

---

# *Remerciements*

Nous remercions piètrement **ALLAH** le tout puissance de m'avoir donné le courage et la volonté à terme ce présent travail.

Nous tenons tout d'abord à remercier notre superviseur **Mme ABDELLATIF Takoua** pour sa générosité, sa constante disponibilité, son extrême bienveillance, sa simplicité alliée à son goût de travail bien fait, ont été d'un apport considérable dans l'élaboration de ce projet.

Nous remercions également **Mr JAOUAD Ibrahim** et **Mr BEZZINE Mohamed Khalil** nos encadrants de la société Whitecape Technologie pour l'intérêt qu'ils nous ont porté tout au long de notre stage ainsi que pour leurs aides et précisions.

A la fin, nous présentons nos sincères remerciements à tous nos enseignants de l'ESSTHS qui nous ont suivis durant notre formation.

---

# *Dédicace*

*Je dédie ce modeste travail :*

*A ma mère SAMIRA, tu m'as donné la vie et le courage pour réussir. Tout ce que je peux t'offrir ne pourra exprimer l'amour et la reconnaissance que je te porte. Tu es pour moi une mère estimée, exceptionnelle et exemplaire.*

*A mon père MOHSEN, mon premier encadrant depuis ma naissance, l'épaule solide, l'œil attentif et le plus digne de mon estime et de mon respect.*

*A mon frère Mohamed et ma sœur Jamila avec qui je partage de merveilleux moments. Je vous souhaite une vie pleine de joie, de bonheur et de réussite.*

*A mon cher binôme et ma soeur Hiba, avec qui j'ai surmonté les difficultés pour aboutir à ce travail. J'ai vécu avec toi des bons moments au cours de ces années. Tu es parfaite et notre amitié n'est pas secrète.*

*A mes proches Farah, Sara, Sirine, Lilya et Ghofrane qui ont toujours été présents pour les bons conseils. Vos affections et vos soutiens m'ont été d'un grand secours au long de ma vie.*

*A mes amis Omar, Jouda, Lotfi, Rihab, Manel et Samer, que j'aime beaucoup et à qui je souhaite le bonheur et le succès.*

*A tous mes professeurs de l'ESSTHS ainsi qu'à mes collègues de promotion.*

*Mes chers, Que Dieu vous protège.  
Je vous aime l'infini*

*Imène*

# TABLE DES MATIÈRES

<b>Introduction Générale</b>	<b>1</b>
<b>1 Présentation du cadre du projet</b>	<b>3</b>
<b>Introduction</b>	<b>3</b>
1.1 Organisme d'accueil . . . . .	3
1.1.1 Données générales : . . . . .	3
1.1.2 Services . . . . .	4
1.1.3 Structure . . . . .	5
1.2 Cadre général . . . . .	5
1.3 Problématique . . . . .	6
<b>Conclusion</b>	<b>6</b>
<b>2 Etat de l'art</b>	<b>7</b>
<b>Introduction</b>	<b>7</b>
2.1 Etude préalable . . . . .	7
2.1.1 Etude de l'existant . . . . .	7
2.1.2 Critique de l'existant . . . . .	7
2.1.3 Solution adoptée . . . . .	8
2.2 Analyse et spécification des besoins . . . . .	9
2.2.1 Besoins fonctionnels . . . . .	9
2.2.2 Besoins non fonctionnels . . . . .	9
2.3 Méthodologie adoptée . . . . .	10
2.4 Planification du travail . . . . .	11
2.5 Environnement de développement . . . . .	12

2.5.1	Environnement matériel . . . . .	12
2.5.2	Environnement logiciel . . . . .	12
2.5.2.1	Environnement de développement : Eclipse IDE . . .	12
2.5.2.2	Serveur : Apache Tomcat . . . . .	13
2.5.2.3	SGBD : PostgreSQL . . . . .	13
2.5.3	Langages et Framework . . . . .	14
2.5.3.1	Langage java . . . . .	14
2.5.3.2	JEE : Java Enterprise Edition . . . . .	14
2.5.3.3	Apache Maven . . . . .	15
2.5.3.4	Spring . . . . .	15
2.5.3.5	Spring MVC . . . . .	16
2.5.3.6	Spring Security . . . . .	16
2.5.3.7	Hibernate . . . . .	17
2.5.3.8	JSP . . . . .	17
<b>Conclusion</b>		<b>18</b>
<b>3 Etude conceptuelle</b>		<b>19</b>
<b>Introduction</b>		<b>19</b>
3.1	Architecture de l'application . . . . .	19
3.2	Conception détaillées . . . . .	22
3.2.1	Vue fonctionnelle du système . . . . .	22
3.2.1.1	Identification des acteurs . . . . .	22
3.2.1.2	Diagramme de cas d'utilisation global . . . . .	23
3.2.1.3	Diagrammes de cas d'utilisation détaillés . . . . .	24
3.2.2	Vue statique du système . . . . .	32
3.2.2.1	Diagramme de paquetage . . . . .	32
3.2.2.2	Diagramme de classe . . . . .	33
3.2.3	Vue dynamique du système . . . . .	35
3.2.3.1	Diagramme de séquence 'S'authentifier' . . . . .	35
3.2.3.2	Diagramme de séquence 'Ajouter Utilisateur' . . . .	36
3.2.3.3	Diagramme de séquence 'Rechercher Utilisateur' . . .	37
3.2.3.4	Diagramme de séquence 'Modifier Utilisateur' . . . .	38
3.2.3.5	Diagramme de séquence 'Supprimer Utilisateur' . . .	39
3.2.3.6	Diagramme de séquence 'Editer facture' . . . . .	40
<b>Conclusion</b>		<b>41</b>

<b>4 Réalisation</b>	<b>42</b>
<b>Introduction</b>	<b>42</b>
4.1 Acquisition des données . . . . .	42
4.2 Présentation de l'application . . . . .	43
<b>Conclusion</b>	<b>50</b>
<b>Conclusion Générale</b>	<b>3</b>
<b>Bibliographie</b>	<b>3</b>

## TABLE DES FIGURES

1.1	Structure de Whitecape technologies . . . . .	5
2.1	Méthodologie adoptée . . . . .	11
2.2	Plannification de travail . . . . .	11
3.1	Architecture globale de l'application . . . . .	20
3.2	Schéma de l'architecture MVC . . . . .	21
3.3	Diagramme de cas d'utilisation global . . . . .	23
3.4	Diagramme de cas d'utilisation 'Gérer les utilisateurs' . . . . .	24
3.5	Diagramme de cas d'utilisation 'Editer rapport' . . . . .	28
3.6	Diagramme de cas d'utilisation 'Editer facture' . . . . .	29
3.7	Diagramme de cas d'utilisation 'Gérer les objectifs' . . . . .	30
3.8	Diagramme de cas d'utilisation 'Consulter l'état' . . . . .	31
3.9	Diagramme de package . . . . .	32
3.10	Diagramme de classe . . . . .	33
3.11	Diagramme de séquence 'S'authentifier' . . . . .	35
3.12	Diagramme de séquence 'Ajouter Utilisateur' . . . . .	36
3.13	Diagramme de séquence 'Rechercher Utilisateur' . . . . .	37
3.14	Diagramme de séquence 'Modifier Utilisateur' . . . . .	38
3.15	Diagramme de séquence 'Supprimer Utilisateur' . . . . .	39
3.16	Diagramme de séquence 'Editer facture' . . . . .	40
4.1	NanoArduino . . . . .	43
4.2	Interface d'authentification . . . . .	44
4.3	Gestion des utilisateurs . . . . .	44
4.4	Rechercher un (des) Utilisateurs . . . . .	45

4.5	Ajouter un utilisateur . . . . .	45
4.6	Modifier un utilisateur . . . . .	46
4.7	Supprimer un utilisateur . . . . .	46
4.8	Gérer les zones . . . . .	47
4.9	Editer facture . . . . .	47
4.10	Gérer es objectifs . . . . .	48
4.11	Consulter sondes . . . . .	48
4.12	Ajouter sonde . . . . .	49
4.13	Consommation temps réel . . . . .	49
4.14	Consommation par secteur . . . . .	50

## LISTE DES TABLEAUX

3.1	Description textuelle de cas d'utilisation 's'authentifier'	25
3.2	Description textuelle de cas d'utilisation 'Ajouter utilisateur'	25
3.3	Description textuelle de cas d'utilisation 'Modifier utilisateur'	26
3.4	Description textuelle de cas d'utilisation 'Supprimer utilisateur'	26
3.5	Description textuelle de cas d'utilisation 'Rechercher utilisateur'	27
3.6	Description textuelle de cas d'utilisation 'Télécharger rapport'	28
3.7	Description textuelle de cas d'utilisation 'Consulter rapports'	28
3.8	Description textuelle de cas d'utilisation 'Afficher la liste des factures'	29
3.9	Description textuelle de cas d'utilisation 'Télécharger facture'	29
3.10	Description textuelle de cas d'utilisation 'Ajouter seuil'	30
3.11	Description textuelle de cas d'utilisation 'Afficher l'historiques'	30
3.12	Description textuelle de cas d'utilisation 'Consulter la consommation'	31



## INTRODUCTION GÉNÉRALE

Durant les dernières années, l'informatique a connu une grande évolution et s'est imposée d'une manière très impressionnante dans les établissements publics. En effet, l'amélioration de la qualité des services est un défi que tout acteur dans le domaine de gestion cherche à atteindre. Afin d'y parvenir, il est crucial de proposer des nouvelles technologies d'informations pour améliorer, d'une part, le fonctionnement et la visibilité des entreprises, et d'autre part, garantir une sécurité des données. Le gain de temps, la performance, la maintenabilité et la facilité de sauvegarde et de manipulation des données restent toujours les objectifs principaux que toutes les entreprises et les établissements cherchent à réaliser dans le domaine de la gestion de l'information.

De nos jours, les entreprises industrielles subissent une lourde charge financière causée par la consommation énorme de l'énergie électrique. Ceci les a poussé à agir pour bien contrôler cette consommation. Durant notre stage on nous a confié la réalisation d'un système logiciel qui assure le suivi de la consommation au sein de l'entreprise industrielle. Ce système est lié à des composants matérielles qui jouent le rôle de sonde. En effet, notre stage ayant comme intitulé « système de suivi de consommation d'énergie », s'est déroulé du 01 Février 2017 au 31 mai 2017 et a lieu à Sousse au sein de l'entreprise Whitecape.

L'objectif de ce projet est de restituer les données depuis des sondes installées à l'intérieur de l'entreprise. les traiter selon le besoin pour les présenter sous un format qui aide la prise de décision par le responsable d'énergie. ce système permettra l'audit et l'optimisation de l'énergie en vue de réduire les dépenses.

Les résultats de nos travaux sont présentés dans ce rapport qui comportera 4 chapitres :

- le premier chapitre est un chapitre introductif dans lequel nous introduirons notre projet ainsi que nous présentons le cadre de notre stage de fin d'études à savoir l'organisme d'accueil Whitecape Technologies.
- le deuxième chapitre est dédié à une étude préalable, une spécification des besoins ainsi que nos choix des outils de développement.
- le troisième chapitre fera l'objet de l'étude conceptuelle de l'application.
- Et enfin, le quatrième chapitre est réservé pour la réalisation et la présentation des interfaces de l'application.

# CHAPITRE 1

## PRÉSENTATION DU CADRE DU PROJET

### Introduction

Dans ce chapitre, nous commençons par la présentation de l'organisme d'accueil dans lequel nous avons effectué notre projet de fin d'études. Ensuite nous nous posons la problématique.

### 1.1 Organisme d'accueil

Whitecape Technologies <sup>1</sup> est une société de service d'ingénierie informatique à responsabilité limitée (SARL), créée en 2008. C'est une société Euro-Tunisienne spécialisée dans la sous-traitance des développements informatiques. Elle apporte à ses clients une présence locale et un suivi de proximité, elle travaille en collaboration avec ses partenaires européens pour offrir à ses clients une forte valeur ajoutée. Elle adopte un modèle Near-shore sécurisée et évolutif.

#### 1.1.1 Données générales :

- **Nom de la Société** : Whitecape Technologie.
- **Associs** : Eric PANSIN et Samy BEN DAOUD.
- **Forme Juridique** : Société à Responsabilité Limitée (S.A.R.L.).
- **Secteur d'Activité** : Service Informatique.

---

<sup>1</sup><http://whitecapetech.com>

- **Date Création** : 2008

### 1.1.2 Services

Whitecape Technologies assiste ses clients pour choisir le meilleur mode de collaboration et définir le type de contrat le plus adéquat à leur contexte.

- Des éditeurs : Whitecape technologies se présente comme un éditeur pour les entreprises ayant des contraintes de «Time To Market», des documents d'analyse incomplets et des budgets sous contrôle, elle propose des solutions et des démarches pour permettre de faire face aux urgences du marché mais aussi aux besoins en développement transversal et de fond.
- Une SSII : Whitecape technologies propose une collaboration entre ses équipes et les équipes des autres entreprises et une collaboration aussi sur les études techniques. De plus, elle offre une équipe de management efficace. Ces solutions sont dédiées aux fournisseurs des services informatiques qui ont des difficultés pour répondre aux cahiers de charges ou de lancer des projets en même temps.
- Une agence web : Dans le domaine du web, Whitecape Technologies met à la disposition des clients son savoir-faire dans :
  - Les développements LAMP avancés et des services web.
  - L'intégration CMS : Magento, Social Engine, Joomla, Drupal, WordPress...
  - La création des sites sur mesures et les réseaux sociaux.
- Des «End User» : Whitecape Technologies intervient comme un accompagnant dans les projets informatiques dans des autres entreprises tout au long de développement en mettant toute son expertise en termes de gestion de projet et technique.

### 1.1.3 Structure

Pour assurer une intégration quasi-totale avec les environnements de ses clients whitecape utilise des canaux VPN cryptés sur des lignes spécialisées à débits garantis et elle fait des investissements de structure nécessaire pour intégrer les environnements de production de ces clients et offrir une plateforme de communication efficace et sécurisée.

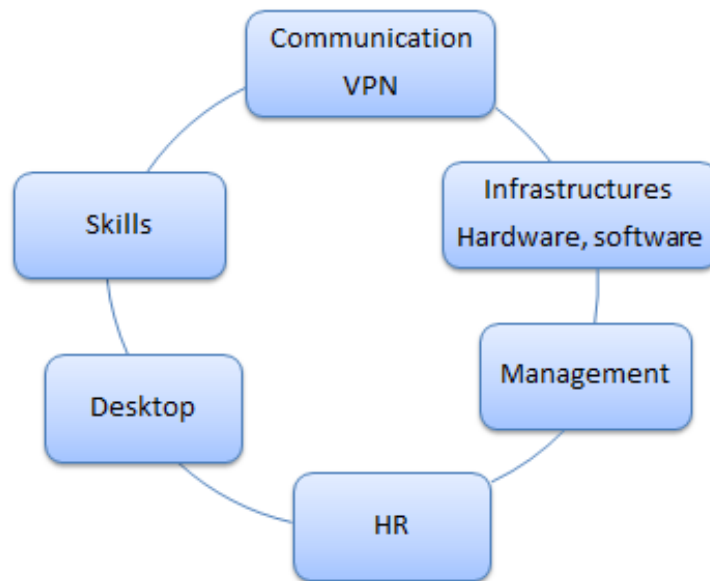


FIGURE 1.1 – Structure de Whitecape technologies

## 1.2 Cadre général

Ce présent travail s'inscrit dans le cadre de stage de fin d'études en vue de l'obtention d'un diplôme du mastère professionnel en Pilotage des Systèmes industriels à l'école Supérieure des sciences et technologies de Hammem Sousse, effectué au sein de la société 'Whitecape Technologies'. Au cours de ce stage, on nous a confié la conception et la réalisation d'une application web dans un environnement purement professionnel.

## 1.3 Problématique

Actuellement, dans la plupart des bâtiments, on remarque l'absence de suivi de consommation de l'électricité, c'est pour cette raison Whitecape a décidé de développer un système de suivi afin de garder cette richesse et de mettre fin au manque d'attention de l'être humain.

- Gaspillage d'énergie.
- Perte des données importantes.
- Absence d'outils qui aide à la prise de décision.
- Absence de vue par secteur.

L'idée de suivi instantané de la consommation d'énergie permet d'identifier l'origine de surconsommation en vue de reviser la répartition de l'énergie.

## Conclusion

Après avoir introduit notre projet et présenté des informations générales de l'organisme d'accueil, nous allons nous intéresser à l'étude de l'état de l'art dans le chapitre suivant.

## CHAPITRE 2

## ETAT DE L'ART

### Introduction

Ce chapitre est dédié, en premier lieu, pour une étude de l'existant suivie de sa critique et une solution proposée. En deuxième lieu, nous nous intéressons à analyser et spécifier les besoins. Et finalement nous présentons notre méthodologie adoptée.

### 2.1 Etude préalable

#### 2.1.1 Etude de l'existant

L'étude de l'existant est une phase importante pour bien comprendre le système actuel et définir ses objectifs. En fait, afin de garantir l'électrification du pays, la société tunisienne d'électricité et du gaz est engagé de nous produire, transporter et distribuer l'énergie électrique.

Aussi, sa principale mission est de s'occuper de notre consommation et de nous distribuer une facture basée sur la quantité reçue du compteur situé dans notre habitation. Cette facture pourra être une facture de consommation réelle ou discrétionnaire.

#### 2.1.2 Critique de l'existant

L'étude de l'existant a permis de dégager un certain nombre de lacunes :  
D'une part, les deux seules solutions pour connaître l'état de notre consommation

qui sont la facture et la lecture directe du compteur électrique, sont des solutions traditionnelles et inefficaces.

Prenons l'exemple d'un utilisateur qui se déplace à l'étranger pour une période. Ce dernier ne recevra pas sa facture et par suite il perd son accès de suivre la consommation de son logement.

D'autre part, la facture bien qu'elle est puissante ne satisfait pas certains besoins des utilisateurs.

En effet, cette facture n'est pas détaillée : elle nous permet de connaître uniquement notre consommation générale. Alors que le suivi devra mieux être plus spécifique : sur différents secteurs (éclairage, prise de courant, climatisation. . .).

De plus, l'utilisateur ne peut pas savoir si sa consommation atteint son maximum ou pas et par conséquent il risque de perdre certains équipements suite à une coupure d'urgence de courant.

### 2.1.3 Solution adoptée

Pour remédier aux différents problèmes cités dans ce qui a précédé. Notre système Imeasure se résume en deux parties l'une pour l'acquisition des données et l'autre se présente comme une application web exploitant ses informations et assurant de nombreuses fonctionnalités. Tel que :

- Restituer les données depuis les sondes.
- Une acquisition correcte et pertinente des mesures.
- Une vue permanente des consommations.
- La possibilité de suivre l'évolution de la consommation par secteur en temps réel :
  - Consommation globale.
  - Consommation d'éclairage.
  - Consommation de climatisation.
  - Consommation des prises de courant.
- Exploitation des données sous forme de graphiques et de statistiques.
- Génération des rapports pour l'aide à la décision.
- Alerte en cas de situations anormales.



- Déceler rapidement un éventuel problème technique.
- Pérennité du système.

## 2.2 Analyse et spécification des besoins

### 2.2.1 Besoins fonctionnels

Il s'agit des fonctionnalités à assurer par l'application. Ce sont les besoins spécifiant le comportement d'entrée/sortie. L'application doit satisfaire les besoins fonctionnels suivants :

- Gestion des utilisateurs.
- Gestion de profil.
- Gestion des objectifs.
- Gestion des sondes/secteurs/zones.
- Consultation de l'état de consommation.
- Edition des rapports/factures.

### 2.2.2 Besoins non fonctionnels

L'application doit assurer certaines contraintes tels que :

- **La fiabilité** : c'est-à-dire, notre application doit être utilisée sans défaillance dans les conditions opérationnelles spécifiques
- **La convivialité** : l'application doit être simple et facile à utiliser, même par des non-experts. Elle doit également présenter un enchainement logique des interfaces qui permet d'assurer une navigation rapide.
- **La disponibilité** : lorsqu'un utilisateur désire consulter le site, ce dernier doit être disponible.
- **La stabilité** : il faut que l'application évite le maximum de problèmes d'exécution ou en cas de changement de système, l'application doit conserver sa position d'équilibre.
- **L'extensibilité** : on devra toujours garder une possibilité d'ajouter ou de modifier de nouvelles fonctionnalités.

- **La sécurité** : l'application doit être hautement sécurisée. Les informations ne devront pas être accessibles à tout le monde, seulement ceux qui ont le droit d'accès. En effet, chaque utilisateur doit s'authentifier pour bénéficier des services offerts par l'application/consulter les services de l'application.

## 2.3 Méthodologie adoptée

La méthodologie est une démarche organisée rationnellement pour aboutir à un résultat. Parmi les différentes méthodologies existantes, nous pouvons citer le modèle en cascade utilisé souvent dans les simples projets dont les besoins sont clairs et bien définis dès le début. D'autres part, on peut citer aussi le modèle en Y qui dissocie et parallélise la résolution des questions fonctionnelles et techniques. Pour bien conduire notre projet et assurer un bon déroulement des différentes phases, nous avons opté pour la méthode agile comme une méthodologie de conception et de développement.

### **Pourquoi méthode agile ?**

Nous avons choisi cette méthode vue ses caractéristiques :

- Très bonne explication et application d'un processus empirique.
- Simple et facile à mettre en place.
- Applicable avec diverses pratiques de logiciel.
- Réduise les risques de « hors sujet » enfin de projet.

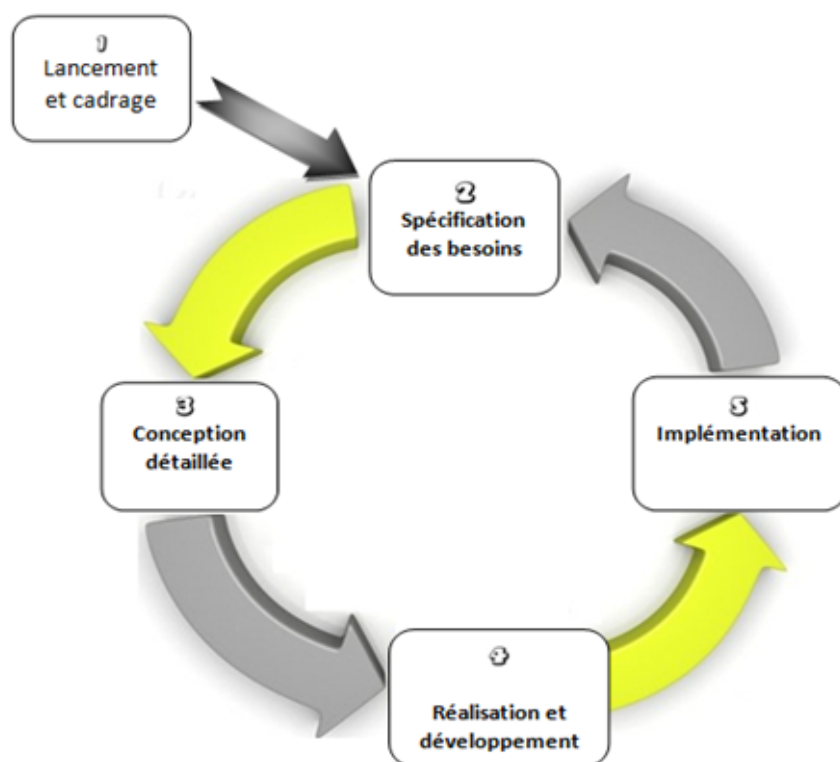


FIGURE 2.1 – Méthodologie adoptée

## 2.4 Planification du travail

Le diagramme de Gantt couramment utilisé en gestion de projet, est l'un des outils les plus efficaces pour représenter visuellement l'état d'avancement des différentes activités (tâches) qui constituent un projet.

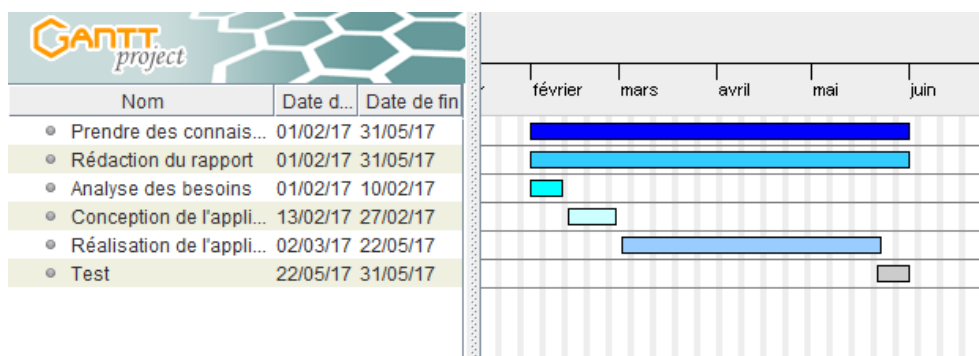


FIGURE 2.2 – Plannification de travail

Les tâches que nous avons effectuées durant notre stage sont illustrées dans ce diagramme comme suit :

- Prendre des connaissances sur Spring : se fait durant toute la période.
- Rédaction du rapport : cette tâche se fait pendant toute la période du stage.
- Analyse des besoins : nécessite dix jours.
- Conception de l'application : dure quinze jours.
- Réalisation de l'application : nécessite environ deux mois.
- Test : c'est une vérification partielle du système, il se fait pendant dix jours.

## 2.5 Environnement de développement

### 2.5.1 Environnement matériel

Pour développer notre application, nous avons utilisé un ordinateur portable ayant les caractéristiques suivantes :

- Processeur Intel Core i3-2350M
- Ecran 15,6 HD WLED
- Disque dur 500 Go
- RAM 4096 Mo
- Chipset graphique Intel HD Graphics

### 2.5.2 Environnement logiciel

Afin de développer notre application, nous avons eu recours à :

#### 2.5.2.1 Environnement de développement : Eclipse IDE

**Description :** Eclipse est un environnement de développement intégré libre, extensible, universel et polyvalent, permettant de créer des projets de développement mettant en œuvre n'importe quel langage de programmation. Eclipse est une communauté pour les individus et les organisations qui souhaitent collaborer sur le logiciel commercialement amicale open source. Ses projets portent sur la construction d'une

plate-forme de développement ouverte composée des cadres extensibles, outils et exécutions pour construire, déployer et gérer des logiciels à travers le cycle de vie.

**Autre Choix :** Netbeans IDE qui présente toutes les caractéristiques indispensable pour à un EDI de qualité, que ce soit pour développer en Java, Ruby, c/c++ ou même PHP.

**Raison :** Eclipse est très loin leader dans les entreprises, il supporte un grand nombre de plugins qui le rendent utilisable pour d'autres langages et d'autres technologies.

#### 2.5.2.2 Serveur : Apache Tomcat

**Description :** Apache Tomcat <sup>1</sup> est un conteneur de servlet J2EE. Issu du projet Jakarta, Tomcat est désormais un projet principal de la fondation Apache. Tomcat implémente les spécifications des servlets et des JSP de Sun Microsystems. Il inclut des outils pour la configuration et la gestion, mais peut également être configuré en éditant des fichiers de configuration XML. Comme Tomcat inclut un serveur http interne, il est aussi considéré comme un serveur HTTP1.

**Autres Choix :** Glassfish, JBoss.

**Raison :** Glassfish et JBoss sont des serveurs d'application alors qu'Apache Tomcat présente un serveur web. Nos besoins pour l'application ne nécessitent pas un serveur d'application, un serveur web est suffisant.

#### 2.5.2.3 SGBD : PostgreSQL

**Description :** PostgreSQL <sup>2</sup> est un système de gestion de base de données relationnelle et objet (SGBDRO). C'est un outil libre disponible selon les termes d'une licence de type BSD. Ce système est concurrent à d'autres systèmes de gestion de base de données, qu'ils soient libres (comme MariaDB, MySQL et Firebird), ou propriétaires (comme Oracle, Sybase, DB2, Informix et Microsoft SQL Server). Comme les projets libres Apache et Linux, PostgreSQL n'est pas contrôlé par une seule entreprise, mais est fondé sur une communauté mondiale de développeurs et d'entreprises.

**Pg admin :** Outil de gestion de base de données et un outil d'administration graphique pour PostgreSQL distribué selon les termes de la licence PostgreSQL.

---

<sup>1</sup>[http :http ://tomcat.apache.org/](http://tomcat.apache.org/)

<sup>2</sup>[https :https ://www.postgresql.org/](https://www.postgresql.org/)

## 2.5.3 Langages et Framework

### 2.5.3.1 Langage java

Java [2] est un langage de programmation moderne et une plate-forme informatique développé par Sun Microsystems (aujourd'hui racheté par Oracle). Il est orienté objet et très puissant en garantissant la portabilité des applications développées avec java.

Nous avons choisi d'utiliser le langage de programmation Java et ceci grâce aux raisons ci-dessous :

- Java permet le développement d'application portable de hautes performances sur une large gamme de plates-formes informatiques.
- Java possède de nombreuses classes et packages dans sa bibliothèque, ce qui ouvre l'horizon de développement d'application professionnels.
- Le langage Java (et/ou JEE) est très utilisé par l'équipe de développement de la société Whitecape, ce qui facilite la maintenance et la modification de l'application au futur.
- Java permet d'écrire des applications puissantes et efficaces pour les Smartphone, qui nous rendent capables de migrer notre solution vers une application mobile.

### 2.5.3.2 JEE : Java Enterprise Edition

**Description :** c'est une plateforme pour la technique Java destinée pour les applications d'entreprise en offrant un ensemble de composants sous forme d'API. JEE comprend des spécifications du serveur d'application, des services à travers des API, c'est-à-dire des extensions Java permettant d'offrir en standard un certain nombre de fonctionnalités.

**Autre choix :** la plateforme .NET qui est un produit proposé par Microsoft pour le développement des applications d'entreprise multi-niveaux basées sur des composants Microsoft.

**Raison :** avec JEE, on se trouve dans une situation plus favorable avec les différents Framework open source et à disponibilité de plateforme d'intégration continue. JEE est riche de plusieurs milliers de classes Java. Ces Classes permettent le développement de tous types d'applications. En effet avec ASP.NET il y a une dépendance forte vis à vis d'un éditeur unique et de sa politique commerciale.

### 2.5.3.3 Apache Maven

**Description :** Apache Maven <sup>3</sup> est un outil logiciel libre pour la gestion et l'automatisation de production des projets logiciels Java, il fournit des moyens de configurations simples basées sur le format XML. Leur concept POM permet d'avoir une compréhension plus simple et plus complète du projet.

**Autre choix :** parmi les autres solutions, on peut trouver Ant (AnotherNeat-Tool) qui est un outil de build multiplateforme écrit en Java. Il sert à l'automatisation des différentes tâches répétitives que l'on fait pendant le développement d'un projet.

**Raison :** Maven possède ses conventions. Il sait où se trouve le code source tandis que Ant est procédural, il faut lui dire exactement ce qu'il doit faire et quand le faire. Il faut lui indiquer de compiler, puis de copier, puis de compresser. Le plugin Maven Compiler place le bytecode dans target/classes et produit un fichier JAR dans le répertoire target. Ant manque une structure standard de projet. Maven est déclaratif. Tout ce que nous avons à faire est la création d'un fichier pom.xml et la mise en place du code source dans le répertoire par défaut de projet. Maven s'occupe du reste. En plus, Maven possède son propre cycle de vie alors que Ant n'a pas de cycle de vie. Avec Ant il n'y a pas de conventions dans l'écriture du XML, ce qui amène de la complexité dans le script.

### 2.5.3.4 Spring

**Description :** Spring [3] est un framework créé et supporté par l'entreprise Spring source. Il permet de simplifier le développement d'application Java et devient un standard dans l'industrie du développement logiciel basé sur la plateforme Java, surtout dans le développement Java EE.

Le framework est organisé en modules, reposant tous sur le module Spring Core :

- Spring Core : implémente notamment le concept d'inversion de contrôle (injection de dépendance). Il est également responsable de la gestion et de la configuration du conteneur.
- Spring Context : Ce module étend Spring Core. Il fournit une sorte de base de données d'objets, permet de charger des ressources (telles que des fichiers de configuration) ou encore la propagation d'événements et la création de contexte comme par exemple le support de Spring dans un conteneur de Servlet.
- Spring AOP : Permet d'intégrer de la programmation orientée aspect.

---

<sup>3</sup>[https :https ://maven.apache.org/](https://maven.apache.org/)

- Spring DAO : Ce module permet d'abstraire les accès à la base de données, d'éliminer le code redondant et également d'abstraire les messages d'erreur spécifiques à chaque vendeur. Il fournit en outre une gestion des transactions.
- Spring ORM : Cette partie permet d'intégrer des frameworks de mapping Object/Relationnel tel que Hibernate, JDO ou iBatis avec Spring. La quantité de code économisé par ce package peut être très impressionnante (ouverture, fermeture de session, gestion des erreurs)
- Spring Web : Ensemble d'utilitaires pour les applications web.. Permet également d'utiliser des requêtes http de type multipart. C'est aussi ici que se fait l'intégration avec le framework Struts.
- Spring Web MVC : Implémentation du modèle MVC.

**Autre Choix :** il y a EJB (Enterprise Java Bean) : qui sont un des éléments très importants de la plateforme Java EE pour le développement d'application distribuées. C'est un conteneur lourd.

**Raison :** on a choisi de travailler avec Spring vu que Spring est un conteneur léger. Ainsi, il nous donne plus flexibilité dans de nombreux aspects du développement de l'application que fait EJB. La configuration peut se faire avec XML ou bien avec les annotations. Spring propose une très bonne intégration avec les autres frameworks open source. C'est un framework qui utilise de simple POJO pour le développement des applications plutôt que d'utiliser des EJB complexe dans un conteneur.

#### 2.5.3.5 Spring MVC

**Description :** C'est l'un des plus importants modules Spring, il offre une implémentation innovante du patron MVC. Spring MVC supporte la création des ressources REST, il propose depuis la version 3.0 un support de paradigme REST, il est ainsi très facile de spécifier à quel type de requête http répondre le contrôleur, ainsi que le type des données qu'il consomme et qu'il produit.

#### 2.5.3.6 Spring Security

**Description :** Spring Security<sup>4</sup>, l'un des projets les plus avancés de Spring. Il propose des options de sécurité pour l'authentification, l'autorisation et le contrôle d'accès. Spring security est un framework qui permet :

- Création d'un formulaire d'authentification.

---

<sup>4</sup><https://projects.spring.io/spring-security/>



- Protection contre les attaques.
- Intégration optionnelle avec Spring mvc qui utilise essentiellement des Taglib.
- Crypter les mots de passe avec plusieurs algorithmes disponibles en standards.
- Implémentation du service d'authentification.
- Des filtres pour gérer les cas d'authentification réussie et en échec.

### 2.5.3.7 Hibernate

**Description :** Hibernate <sup>5</sup> est un framework de mapping objet/Relationnel pour applications Java Il permet de créer une couche d'accès aux données (DAO) plus modulaire, plus maintenable, plus performante qu'une couche d'accès aux données 'classique' reposant sur l'API JDBC

Une solution de gestion de persistance => couche de persistance.

- Générer automatiquement code SQL
- Application plus portable, s'adapte à la base de données cible
- Récupération de données optimisée
- Hibernate fournit plusieurs stratégies pour interroger la base de données, Requête SQL, langage HQL ou api criteria.

### 2.5.3.8 JSP

**Description :** les pages web JSP <sup>6</sup> sont une technologie développée par Sun basée sur Java qui simplifie le processus de développement de sites web dynamiques.

JSP est un langage script côté serveur. Le code de la page est exécuté par le serveur web. Les pages JSP sont des fichiers texte munis de l'extension .jsp, qui remplacent les pages HTML traditionnelles. Les fichiers JSP contiennent du code HTML et du code imbriqué en Java. Cette approche présente cependant plusieurs avantages :

- l'utilisation de Java par les JSP permet une indépendance de la plate-forme d'exécution mais aussi du serveur web utilisé.

---

<sup>5</sup><https://hibernate.org/>

<sup>6</sup><https://www.jsp.ch/>

- la séparation des traitements et de la présentation : la page web peut être écrite par un designer et les tags Java peuvent être ajoutés ensuite par le développeur. Les traitements peuvent être réalisés par des composants réutilisables (des Java beans).
- les JSP sont basées sur les servlets : tout ce qui est fait par une servlet pour la génération de pages dynamiques peut être fait avec une JSP.

## Conclusion

Après avoir mis le projet dans son cadre et après avoir capturé les besoins fonctionnels et non fonctionnels de notre application ainsi que présenter les différents logiciels et technologies pour la réalisation de notre projets, nous allons par la suite entamer la partie de l'étude conceptuelle.

## CHAPITRE 3

## ETUDE CONCEPTUELLE

### Introduction

Dans ce chapitre, nous abordons la partie conception du projet qui présente une phase fondamentale dans le cycle de développement d'une application. Pour cela, nous allons présenter, en premier lieu, notre l'architecture. En deuxième lieu, nous allons commencer l'étude conceptuelle par les diagrammes de cas d'utilisation qui nous permettent d'avoir une vue globale de l'application. Puis, nous allons présenter le diagramme de classe et par la suite le diagramme de paquetage. Enfin, nous finirons par des diagrammes de séquences qui vont présenter chronologiquement les opérations.

### 3.1 Architecture de l'application

Avant d'entrer dans la phase de conception détaillée, il est important de présenter l'architecture générale de l'application. Cette architecture nous aide à décomposer l'application en couches comme décrit dans la figure ci-dessous :

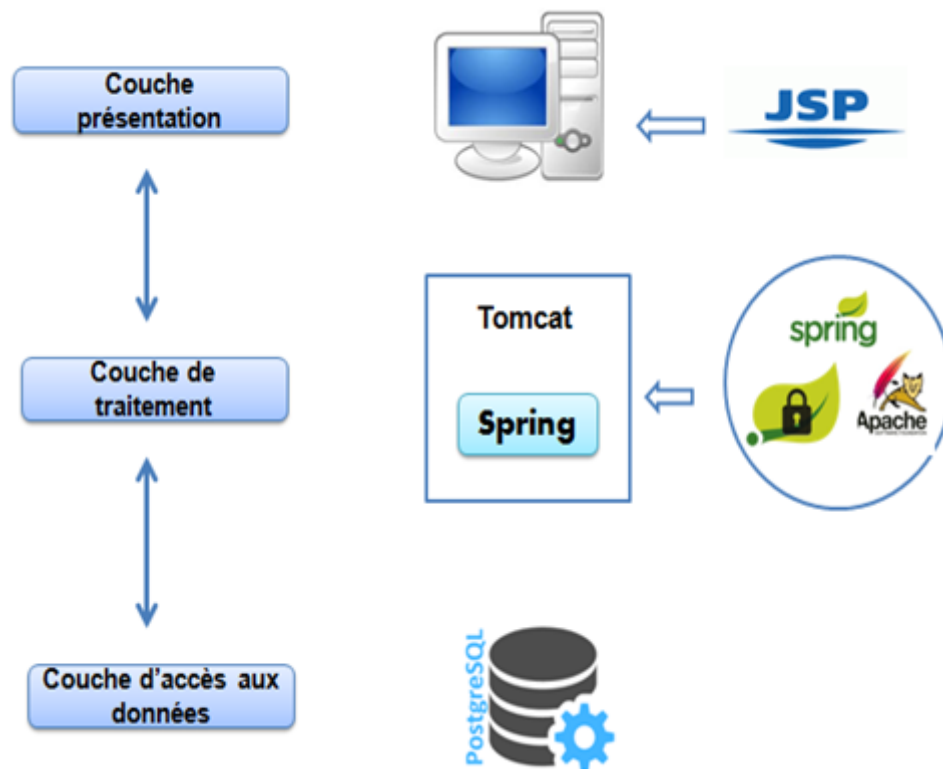


FIGURE 3.1 – Architecture globale de l'application

**Une couche présentation** : Créée avec JSP et qui correspond à la partie visible et interactive de l'application.

**Une couche de traitement** : développée avec Spring MVC, elle correspond à la partie fonctionnelle de l'application, celle qui implémente le logique métier et qui décrit les opérations que l'application opère sur les données en fonction des requêtes des utilisateurs effectuées au travers la couche de présentation.

**Une couche d'accès aux données** : elle correspond à la partie qui gère l'accès aux données à travers l'ORM Hibernate.

Notre architecture suit un modèle MVC présenté dans la figure ci-dessous :

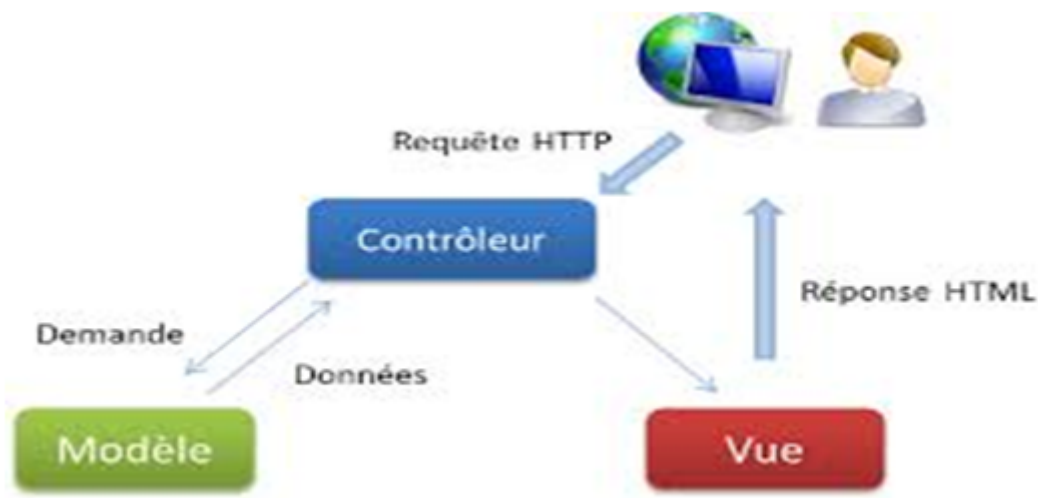


FIGURE 3.2 – Schéma de l'architecture MVC

Le modèle MVC (Model Vue Contrôleur)[1] cherche à séparer les couches présentation, traitement et accès aux données, ce qui assure la clarté de l'architecture et simplifie la tâche de développeur responsable de la maintenance et de l'amélioration du projet. On obtient alors trois couches :

**La couche modèle :** Elle décrit les données manipulées par l'application. Dans le cas typique d'une base de données le modèle offre des méthodes pour mettre à jour ces données (insertion, suppression, modification). Il offre aussi des méthodes pour récupérer ces données. Les résultats renvoyés par le modèle sont dénués de toute présentation. C'est le modèle qui contient toute la logique métier de l'application.

**La couche vue :** La vue correspond à l'interface avec laquelle l'utilisateur interagit. Sa première tâche est de présenter les résultats renvoyés par le modèle qui lui sont passés par le contrôleur. Sa deuxième tâche est de recevoir toutes les actions de l'utilisateur qui seront envoyés au contrôleur. La vue n'effectue aucun traitement, elle se contente d'afficher les résultats des traitements effectués par le modèle.

**La couche contrôleur :** Le contrôleur prend en charge la gestion des événements de synchronisation pour mettre à jour la vue ou le modèle et les synchroniser. Il reçoit tous les événements de l'utilisateur et déclenche les actions à effectuer. Si

une action nécessite un changement des données, le contrôleur demande la modification des données au modèle et ensuite avertit la vue que les données ont changé pour qu'elle se mette à jour.

### Pourquoi MVC ?

L'approche MVC apporte de réels avantages tel que :

- Une conception claire et efficace grâce à la séparation des données de la vue et du contrôleur.
- Un gain de temps de maintenance et d'évolution de l'application.
- Une plus grande souplesse pour organiser le développement entre différents développeurs (indépendance des données, de l'affichage (web désigne) et des actions).

## 3.2 Conception détaillées

### 3.2.1 Vue fonctionnelle du système

#### 3.2.1.1 Identification des acteurs

Tout d'abord, on va introduire la notion d'acteur. En effet un acteur est une entité externe qui interagit avec le système (opérateur, centre distant, autres systèmes ...). En réponse à l'action d'un acteur, le système fournit un service qui correspond à son besoin. Notre système possède trois acteurs :

- **Super admin** : cet acteur désigne la personne responsable qui possède le privilège le plus approfondie. Il est capable de gérer tous ce qui est en relation avec les gestions de système tel que gestion des utilisateurs, gestion des zones et gestion des secteurs. En plus, il a le droit d'éditer les rapports et les estimations des factures.
- **Administrateur** : une fois authentifié, l'administrateur peut suivre les états de consommation, gérer les sondes et les objectifs ainsi éditer les alertes.
- **Agent** : suite à l'authentification, un utilisateur a le droit de consulter l'évolution de la consommation des différents secteurs .

### 3.2.1.2 Diagramme de cas d'utilisation global

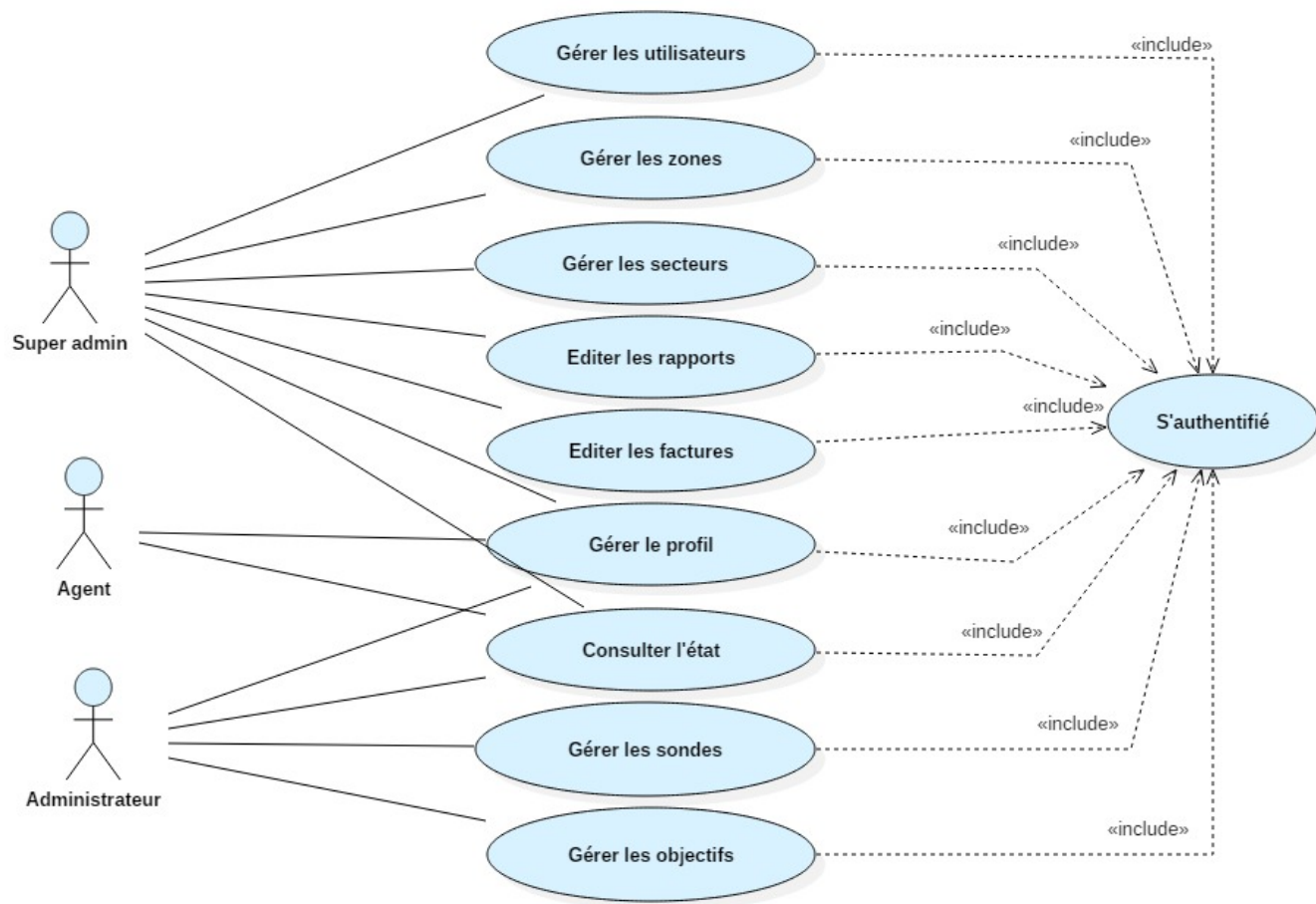


FIGURE 3.3 – Diagramme de cas d'utilisation global

Dans le diagramme de cas d'utilisation ci-dessus, nous avons détaillé les différents besoins fonctionnels nécessaires à l'élaboration de notre projet. En effet, notre application est divisée en trois sessions selon l'utilisateur.

**Session Super admin** : ce mode de connexion est réservé aux super admin qui suite à son authentification peut accomplir les tâches suivantes :

- Gérer les utilisateurs : permet d'ajouter, rechercher, modifier ou supprimer un utilisateur.
- Gérer le profil : permet à l'utilisateur de gérer son profil.
- Gérer les secteurs : permet à l'utilisateur de gérer les secteurs.
- Gérer les zones : permet à l'utilisateur d'ajouter ou de supprimer une zone.

- Consulter l'état : l'interface demandée en cours permet de visualiser l'état de la consommation d'électricité.
- Editer rapport : permet à l'utilisateur d'éditer un rapport, le télécharger ou le supprimer.
- Editer estimation de facture : permet à l'utilisateur d'éditer une estimation de facture, de télécharger ou de la supprimer.

**Session administrateur** : dans cet espace, l'administrateur peut :

- Gérer le profil : permet à l'utilisateur de gérer son profil.
- Gérer les sondes : permet à l'utilisateur d'ajouter ou supprimer une sonde.
- Gérer les objectifs : permet à l'utilisateur de fixer un seuil à ne pas dépasser.
- Consulter l'état : l'interface demandée en cours permet de visualiser l'état de la consommation d'électricité.

**Session agent** : dans cette session un agent peut :

- Gérer le profil : permet à l'utilisateur de gérer son profil.
- Consulter l'état : l'interface demandée en cours permet de visualiser l'état de la consommation d'électricité.

### 3.2.1.3 Diagrammes de cas d'utilisation détaillés

- Diagramme de cas d'utilisation 'Gérer les utilisateurs'

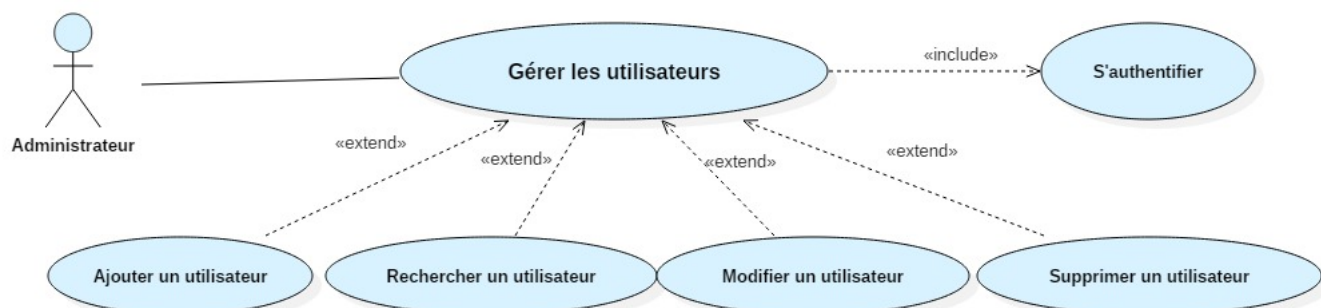


FIGURE 3.4 – Diagramme de cas d'utilisation 'Gérer les utilisateurs'



<b>Cas d'utilisation</b>	S'authentifier
<b>Acteurs</b>	Utilisateur
<b>Pré condition</b>	L'utilisateur doit avoir un compte.
<b>Post condition</b>	Une session s'ouvre.
<b>Scénario nominal</b>	<ol style="list-style-type: none"> <li>1. L'utilisateur demande l'accès à l'application.</li> <li>2. Le système affiche le formulaire d'authentification.</li> <li>3. L'utilisateur remplit le formulaire.</li> <li>4. Le système vérifie les champs et ouvre la session correspondante.</li> <li>5. Si le login ou le mot de passe sont incorrectes, un message d'erreur sera affiché</li> </ol>

TABLE 3.1 – Description textuelle de cas d'utilisation 's'authentifier'

<b>Cas d'utilisation</b>	Ajouter utilisateur
<b>Acteurs</b>	Administrateur
<b>Pré condition</b>	Une authentification préalable.
<b>Post condition</b>	Un utilisateur est bien ajouté.
<b>Scénario nominal</b>	<ol style="list-style-type: none"> <li>1. L'administrateur demande l'affichage de formulaire d'ajout.</li> <li>2. Le système affiche le formulaire.</li> <li>3. L'administrateur remplit le formulaire.</li> <li>4. Le système vérifie et confirme l'ajout de l'utilisateur.</li> <li>5. Le système affiche la nouvelle liste des utilisateurs.</li> </ol>

TABLE 3.2 – Description textuelle de cas d'utilisation 'Ajouter utilisateur '

<b>Cas d'utilisation</b>	Modifier utilisateur
<b>Acteurs</b>	Administrateur
<b>Pré condition</b>	Une authentification préalable.
<b>Post condition</b>	L'utilisateur est modifié.
<b>Scénario nominal</b>	<ol style="list-style-type: none"> <li>1. L'administrateur demande l'affichage de formulaire rempli par les champs à modifier.</li> <li>2. Le système affiche le formulaire demandé.</li> <li>3. L'administrateur modifie l'utilisateur souhaité.</li> <li>4. Le système confirme la modification.</li> </ol>

TABLE 3.3 – Description textuelle de cas d'utilisation 'Modifier utilisateur '

<b>Cas d'utilisation</b>	Supprimer utilisateur
<b>Acteurs</b>	Administrateur
<b>Pré condition</b>	Une authentification préalable.
<b>Post condition</b>	L'utilisateur est supprimé.
<b>Scénario nominal</b>	<ol style="list-style-type: none"> <li>1. L'administrateur demande la suppression de l'utilisateur.</li> <li>2. Le système supprime l'utilisateur.</li> <li>3. Le système retourne la nouvelle liste des utilisateurs.</li> </ol>

TABLE 3.4 – Description textuelle de cas d'utilisation 'Supprimer utilisateur '

<b>Cas d'utilisation</b>	Rechercher utilisateur
<b>Acteurs</b>	Administrateur
<b>Pré condition</b>	Une authentification préalable.
<b>Post condition</b>	Afficher la liste des utilisateurs cherchés.
<b>Scénario nominal</b>	<ol style="list-style-type: none"> <li>1. L'administrateur demande la recherche d'un(des) utilisateur(s).</li> <li>2. Le système recherche d'un(des) utilisateur(s) correspondant(s) au critère de recherche.</li> <li>3. Le système affiche la liste des utilisateurs trouvés.</li> </ol>

TABLE 3.5 – Description textuelle de cas d'utilisation ‘Rechercher utilisateur ‘

• Diagramme de cas d'utilisation 'Editer rapport'

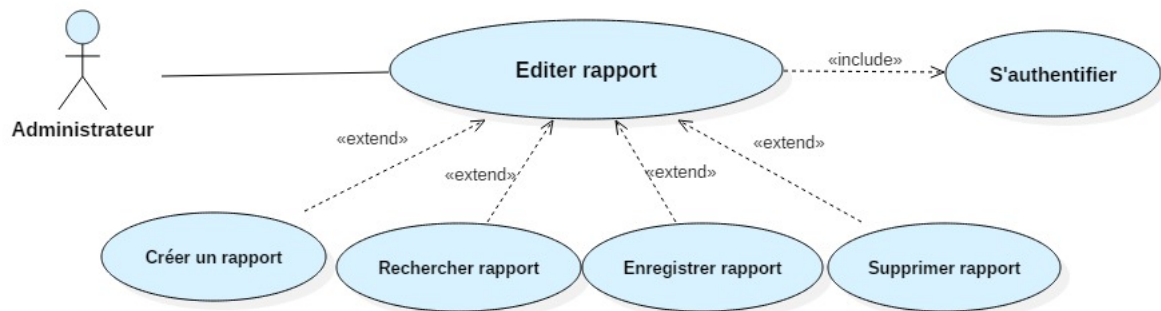


FIGURE 3.5 – Diagramme de cas d'utilisation 'Editer rapport'

<b>Cas d'utilisation</b>	Télécharger rapport
<b>Acteurs</b>	Administrateur
<b>Pré condition</b>	Une authentification préalable.
<b>Post condition</b>	Le rapport est bien téléchargé.
<b>Scénario nominal</b>	<ol style="list-style-type: none"> <li>1. L'utilisateur demande l'enregistrement d'un rapport avec une extension bien défini.</li> <li>2. Le système effectue le téléchargement.</li> </ol>

TABLE 3.6 – Description textuelle de cas d'utilisation 'Télécharger rapport'

<b>Cas d'utilisation</b>	Consulter rapports
<b>Acteurs</b>	Administrateur
<b>Pré condition</b>	Une authentification préalable.
<b>Post condition</b>	La liste des rapports est bien affichée sur l'écran.
<b>Scénario nominal</b>	<ol style="list-style-type: none"> <li>1. L'administrateur demande la consultation des rapports.</li> <li>2. Le système affiche la liste des rapports.</li> </ol>

TABLE 3.7 – Description textuelle de cas d'utilisation 'Consulter rapports'

• Diagramme de cas d'utilisation 'Editer facture'

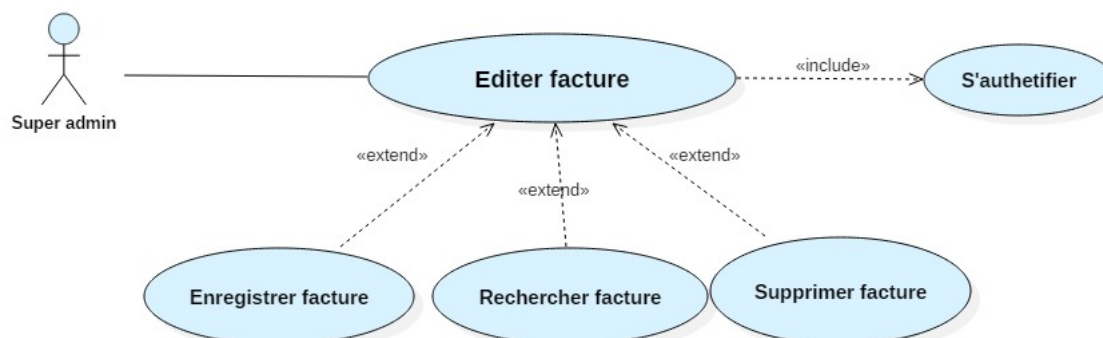


FIGURE 3.6 – Diagramme de cas d'utilisation 'Editer facture'

<b>Cas d'utilisation</b>	Afficher la liste des estimations des factures.
<b>Acteurs</b>	Administrateur
<b>Pré condition</b>	Une authentification préalable.
<b>Post condition</b>	Liste estimations des factures est bien affichée .
<b>Scénario nominal</b>	<ol style="list-style-type: none"> <li>1. L'utilisateur demande l'affichage de la listes des estimations de facture.</li> <li>2. Le système affiche la liste.</li> </ol>

TABLE 3.8 – Description textuelle de cas d'utilisation 'Afficher la liste des factures '

<b>Cas d'utilisation</b>	Télécharger facture .
<b>Acteurs</b>	Administrateur.
<b>Pré condition</b>	Une authentification préalable.
<b>Post condition</b>	La facture est bien téléchargé.
<b>Scénario nominal</b>	<ol style="list-style-type: none"> <li>1. L'utilisateur choisit la facture à télécharger</li> <li>2. L'utilisateur choisit le format sous lequel l'estimation de facture sera téléchargée (PDF, CSV...)</li> <li>3. Le système effectue le téléchargement.</li> </ol>

TABLE 3.9 – Description textuelle de cas d'utilisation 'Télécharger facture '

• Diagramme de cas d'utilisation 'Gérer les objectifs '

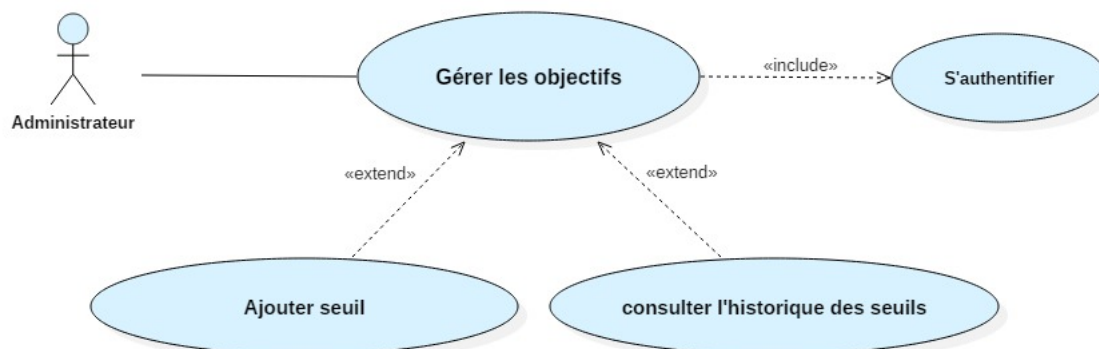


FIGURE 3.7 – Diagramme de cas d'utilisation 'Gérer les objectifs'

<b>Cas d'utilisation</b>	Ajouter seuil
<b>Acteurs</b>	Administrateur
<b>Pré condition</b>	Une authentification préalable.
<b>Post condition</b>	Seuil à ne pas dépasser est bien fixé .
<b>Scénario nominal</b>	<ol style="list-style-type: none"> <li>1. L'utilisateur demande l'affichage du formulaire.</li> <li>2. Le système affiche le formulaire.</li> <li>3. L'utilisateur fixe le seuil.</li> <li>4. Le système enregistre les données.</li> </ol>

TABLE 3.10 – Description textuelle de cas d'utilisation 'Ajouter seuil'

<b>Cas d'utilisation</b>	Afficher l'historique des objectifs
<b>Acteurs</b>	Administrateur
<b>Pré condition</b>	Une authentification préalable.
<b>Post condition</b>	Liste des historiques est affichée.
<b>Scénario nominal</b>	<ol style="list-style-type: none"> <li>1. L'utilisateur demande l'affichage de la liste des historiques.</li> <li>2. Le système affiche la liste.</li> </ol>

TABLE 3.11 – Description textuelle de cas d'utilisation 'Afficher l'historiques '

• Diagramme de cas d'utilisation 'Consulter l'état '

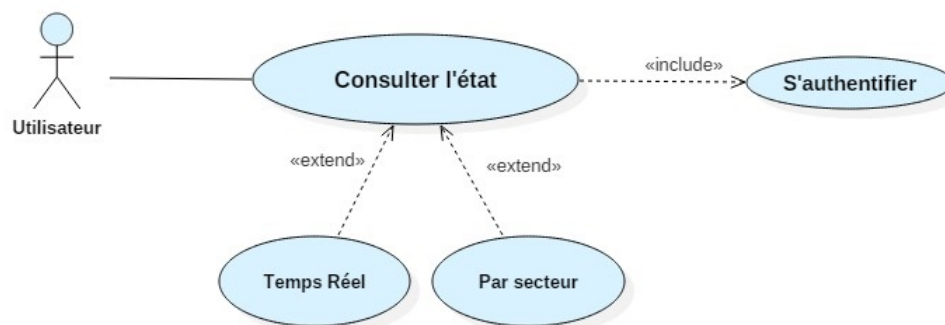


FIGURE 3.8 – Diagramme de cas d'utilisation 'Consulter l'état'

<b>Cas d'utilisation</b>	Consulter l'état de consommation
<b>Acteurs</b>	Administrateur ,Super admin ou agent
<b>Pré condition</b>	Une authentification préalable.
<b>Post condition</b>	Utilisateur a bien consulté sa consommation d'électricité.
<b>Scénario nominal</b>	<ol style="list-style-type: none"> <li>1. L'utilisateur demande la consultation de sa consommation.</li> <li>2. L'utilisateur choisit la manière de consultation : <ul style="list-style-type: none"> <li>– Consommation temps réel</li> <li>– Consommation générale</li> <li>– Consommation par secteurs</li> </ul> </li> <li>3. Le système affiche des courbes de consommation.</li> </ol>

TABLE 3.12 – Description textuelle de cas d'utilisation 'Consulter la consommation'

## 3.2.2 Vue statique du système

### 3.2.2.1 Diagramme de paquetage

Un diagramme de packages est un diagramme UML qui fournit une représentation graphique de haut niveau de l'organisation de notre application.

L'idée est de regrouper des classes en unités appelés 'packages' et de les représenter ainsi que les relations qui lient ces différents paquets.

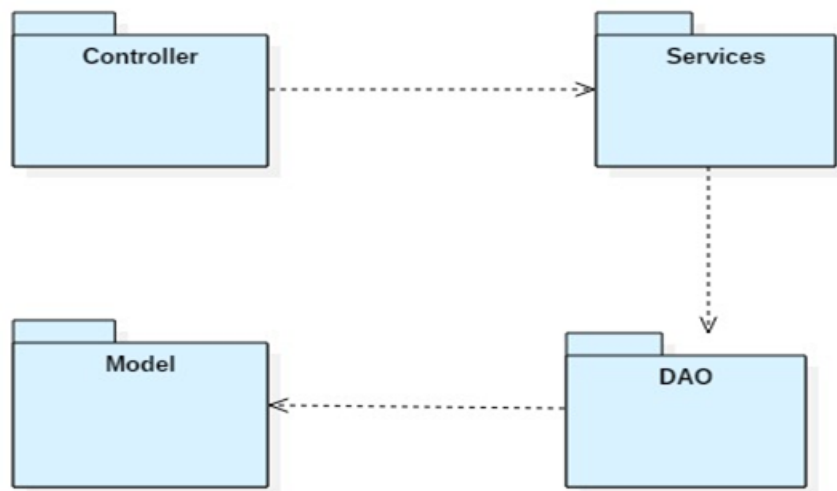


FIGURE 3.9 – Diagramme de package

La figure ci-dessous présente le diagramme de paquetage qui est composé de quatre package.

- le **paquetage Model** : représente les différentes entités de l'application.
- le **paquetage DAO** : représente la couche d'accès aux données.
- le **paquetage Service** : représente la couche de traitement.
- le **paquetage Contrôleur** : représente la couche Controller qui va gérer les requêtes des utilisateurs.



### 3.2.2.2 Diagramme de classe

Le diagramme de classe représente les classes constituant le système et les associations entre elles. Ils expriment de manière générale la structure statique d'un système, en termes de classe et de relations entre ces classes. De même qu'une classe décrit un ensemble d'objets, une association décrit un ensemble de liens ; les objets sont des instances de classe et les liens sont des instances de relation. À l'issu de l'étude et de l'analyse des différents composants de l'application, nous avons pu dégagée le diagramme de classe général suivant permettant de donner une vue claire et générique sur notre application réalisée.

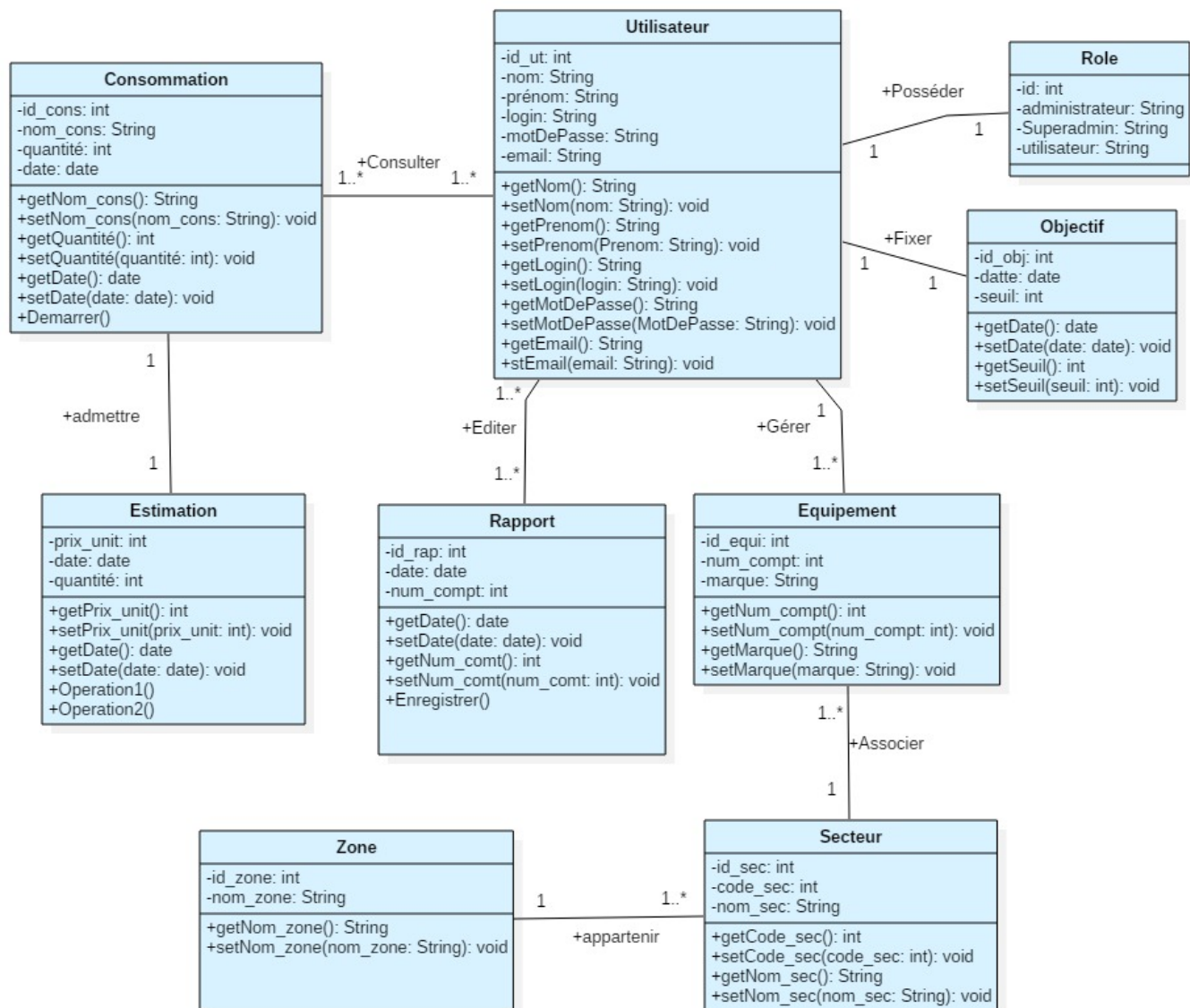


FIGURE 3.10 – Diagramme de classe

Une classe est un ensemble de fonctions (opérations) et de données (attributs) qui sont liées ensemble par un champ sémantique. Une classe décrit aussi un ensemble d'objets et une association décrit un ensemble de liens. Les objets sont les instances des classes et les liens sont les instances des relations.

#### Description détaillée des Classes :

- **Utilisateur** : est une classe qui regroupe tous les utilisateurs de notre application.
- **Rôle** : la classe rôle énumère les différents rôles des différents utilisateurs du système.
- **Consommation** : est une classe qui représente la consommation d'électricité.
- **Objectif** : est une classe qui représente le seuil à ne pas dépasser.
- **Estimation** : est une classe qui représente une estimation de prix de la consommation.
- **Secteur** : est une classe qui énumère les différents secteurs de consommation.
- **Zone** : est une classe qui énumère les différentes zones.
- **Equipement** : est une classe qui présente les équipements(sondes).
- **Rapport** : est une classe désigne l'historique de consommation.

#### Relations entre les classes :

- Un utilisateur possède un et un seul rôle.
- Un utilisateur possède un et un seul rôle.
- Un ou plusieurs utilisateurs peuvent consulter un ou plusieurs consommations.
- Chaque consommation admet une estimation de facture.
- Un ou plusieurs utilisateurs peuvent éditer un ou plusieurs rapports.
- Un utilisateur peut gérer un ou plusieurs équipements(sondes).
- Un ou plusieurs équipements sont associés à un et un seul secteur.
- Un ou plusieurs secteurs appartiennent à une et une seule zone.

### 3.2.3 Vue dynamique du système

#### 3.2.3.1 Diagramme de séquence ‘S’authentifier‘

Le diagramme ci-dessous illustre un scénario d’authentification de l’utilisateur. Ce dernier demande l’interface d’authentification, une requête est envoyée vers le contrôleur HomeController qui retourne l’url de la page d’authentification, puis l’utilisateur saisie les paramètres d’authentification (login et mot de passe). Ces paramètres sont envoyées vers un Filter qui fait appel au service AuthProvider qui traite les données reçues et se connecter dans le cas ou les paramètres sont correctes. Un message d’erreur sera affiché si les paramètres sont incorrectes.

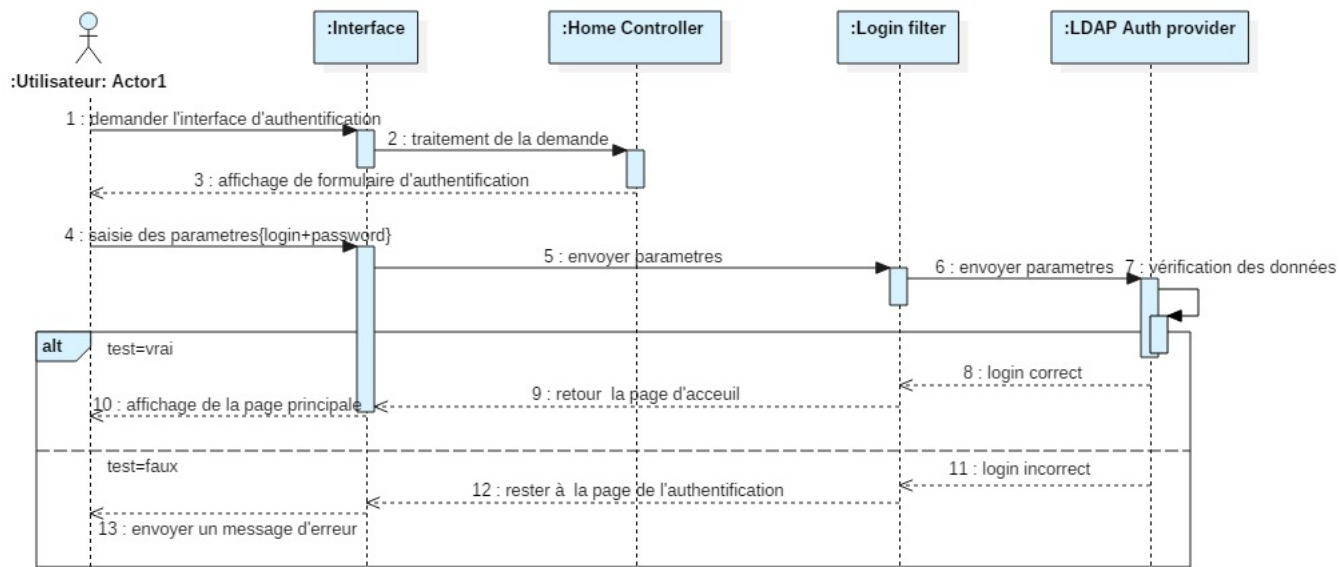


FIGURE 3.11 – Diagramme de séquence ‘S’authentifier‘

### 3.2.3.2 Diagramme de séquence ‘Ajouter Utilisateur’

Le diagramme ci-dessous présente un scénario d’ajout d’un utilisateur. Les paramètres entrées par le Super admin (nom, prénom, email, rôle et image) sont transmis à l’application cliente (client de service web). Cette dernière prend en charge l’invocation de service d’ajouter un nouvel utilisateur. Ce service possède une méthode « create » qui ajoute l’utilisateur après un test des champs.

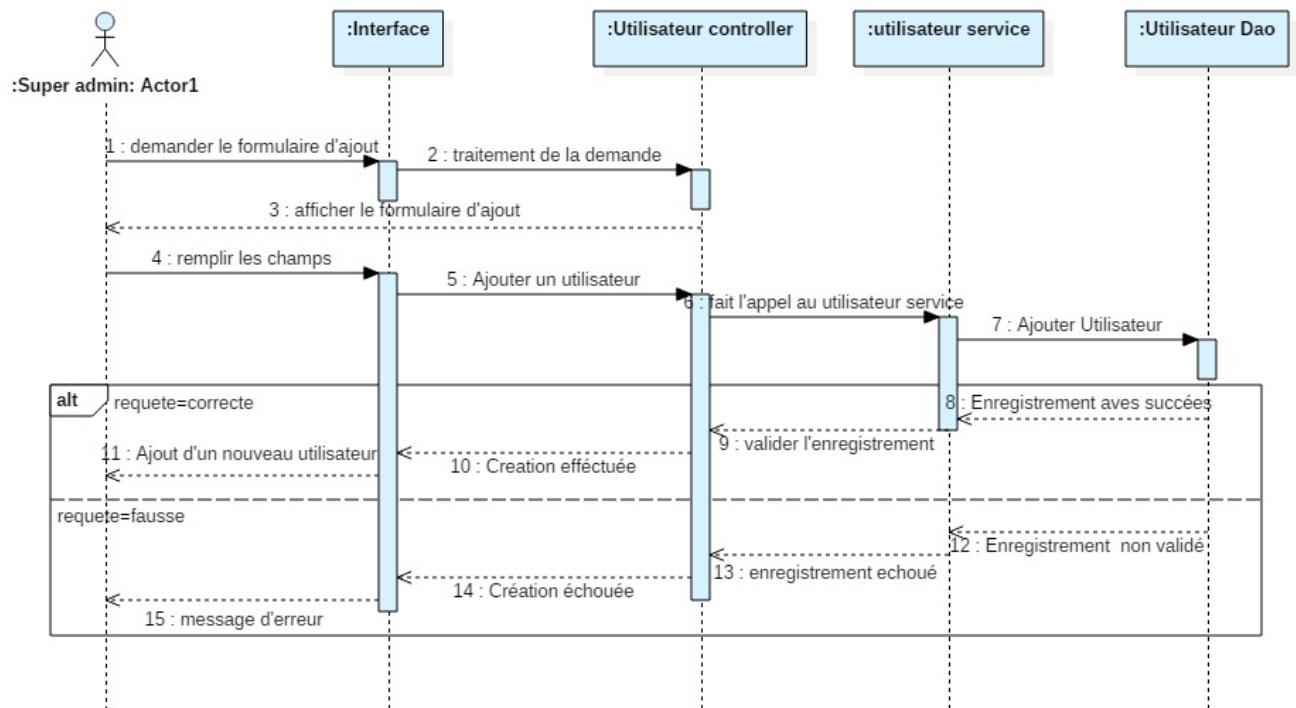


FIGURE 3.12 – Diagramme de séquence ‘Ajouter Utilisateur’

### 3.2.3.3 Diagramme de séquence ‘Rechercher Utilisateur’

Le diagramme ci-dessous présente un scénario de recherche d'utilisateur. Le super admin appuie sur le bouton qui a pour fonction l'affichage de la liste des utilisateurs. Grâce au service «UtilisateurService» qui possède une méthode «listeUser», il peut récupérer tous les enregistrements de la base grâce à la méthode « getUserById ». Le Résultat final est une interface affichée qui contient la liste des utilisateurs trouvée.

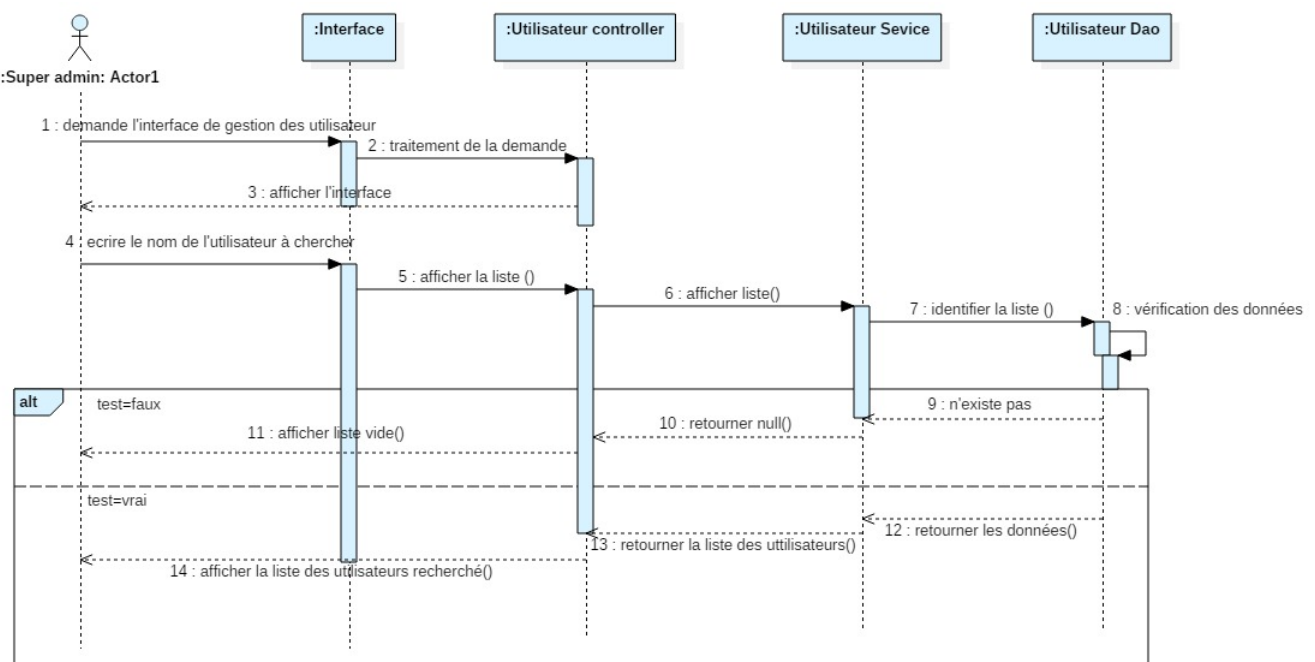


FIGURE 3.13 – Diagramme de séquence ‘Rechercher Utilisateur’

### 3.2.3.4 Diagramme de séquence ‘Modifier Utilisateur’

Le diagramme ci-dessous présente un scénario de modification d’un utilisateur. Premièrement, le Super admin va sélectionner l’utilisateur à modifier et par suite il va faire les modifications nécessaires. Puis, les nouveaux paramètres seront transmis à l’application cliente. Cette dernière prend en charge l’invocation de service de création d’un nouvel utilisateur. Ce service possède une méthode « update » qui retourne les données de l’utilisateur sélectionné pour les mettre à jour.

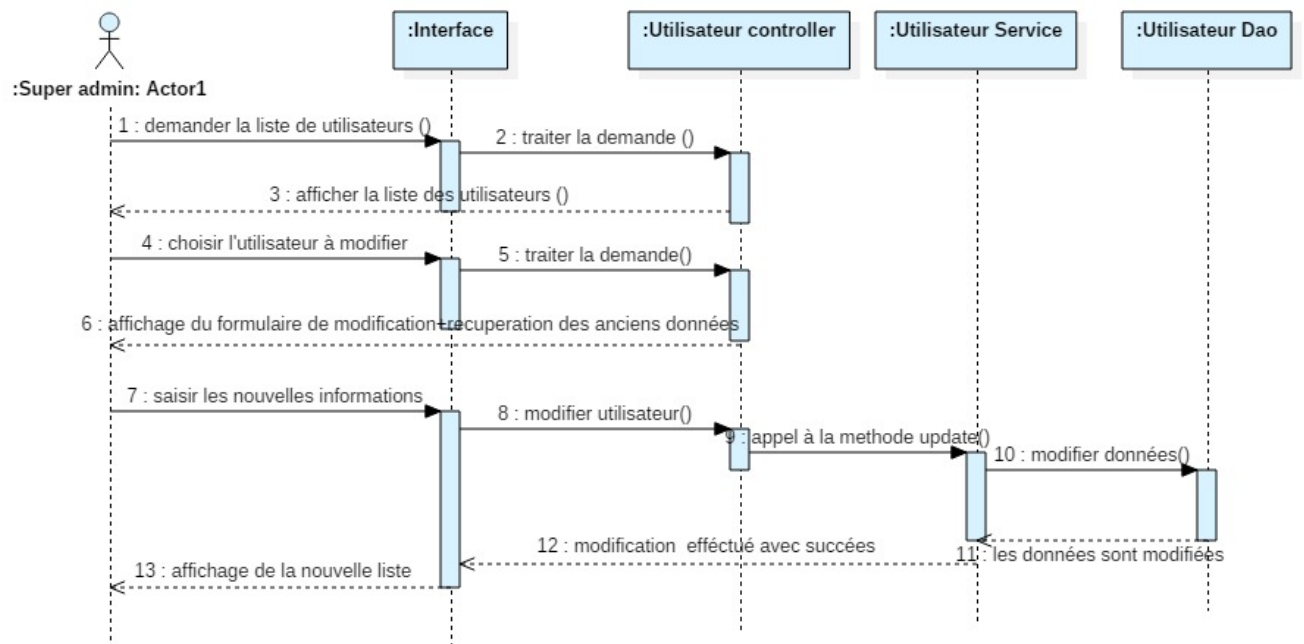


FIGURE 3.14 – Diagramme de séquence ‘Modifier Utilisateur’

### 3.2.3.5 Diagramme de séquence ‘Supprimer Utilisateur’

Le diagramme ci-dessous illustre un scénario de suppression d'utilisateur .Donc il suffit de sélectionner l'utilisateur à supprimer et par suite le contrôleur «UtilisateurController» fait appel au service «UtilisateurService» qui possède une méthode «delete» qui a son tours fait appel à la méthode «remove».

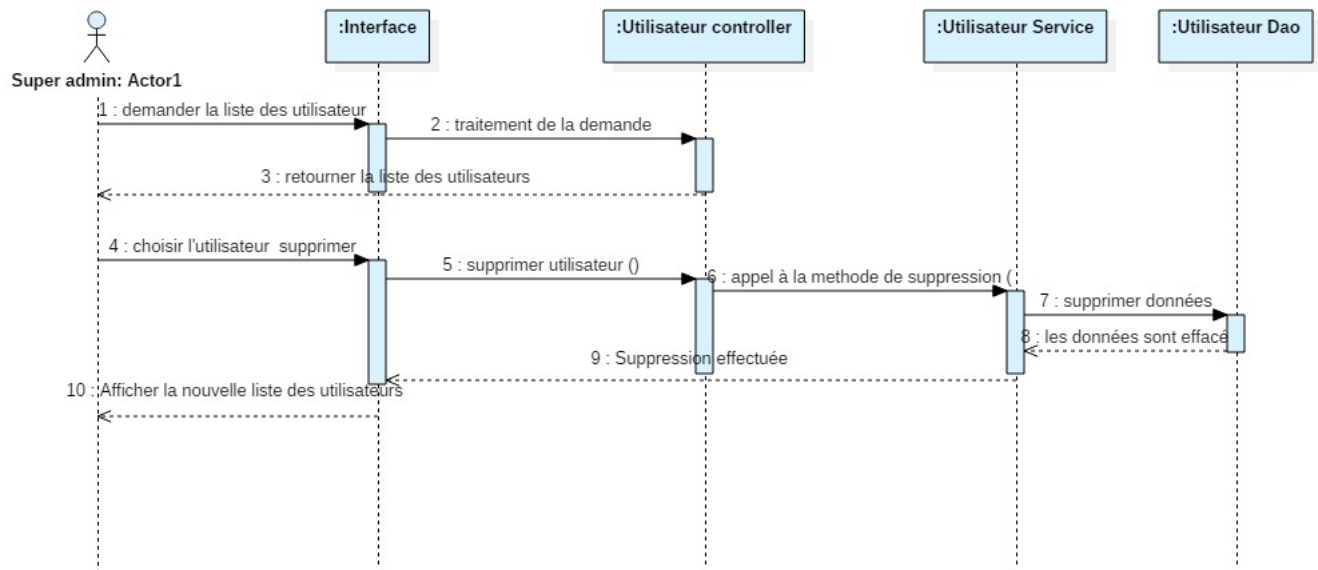


FIGURE 3.15 – Diagramme de séquence ‘Supprimer Utilisateur’

### 3.2.3.6 Diagramme de séquence ‘Editer facture’

Le diagramme ci-dessous présente l’édition de facture. En effet, après son authentification, le Super admin demande l’affichage de l’estimation de facture actuelle : Il a la possibilité d’enregistrer une facture sous forme d’un PDF,CSV... de rechercher une estimation de facture selon une date bien déterminé et de supprimer les estimations indésirables.

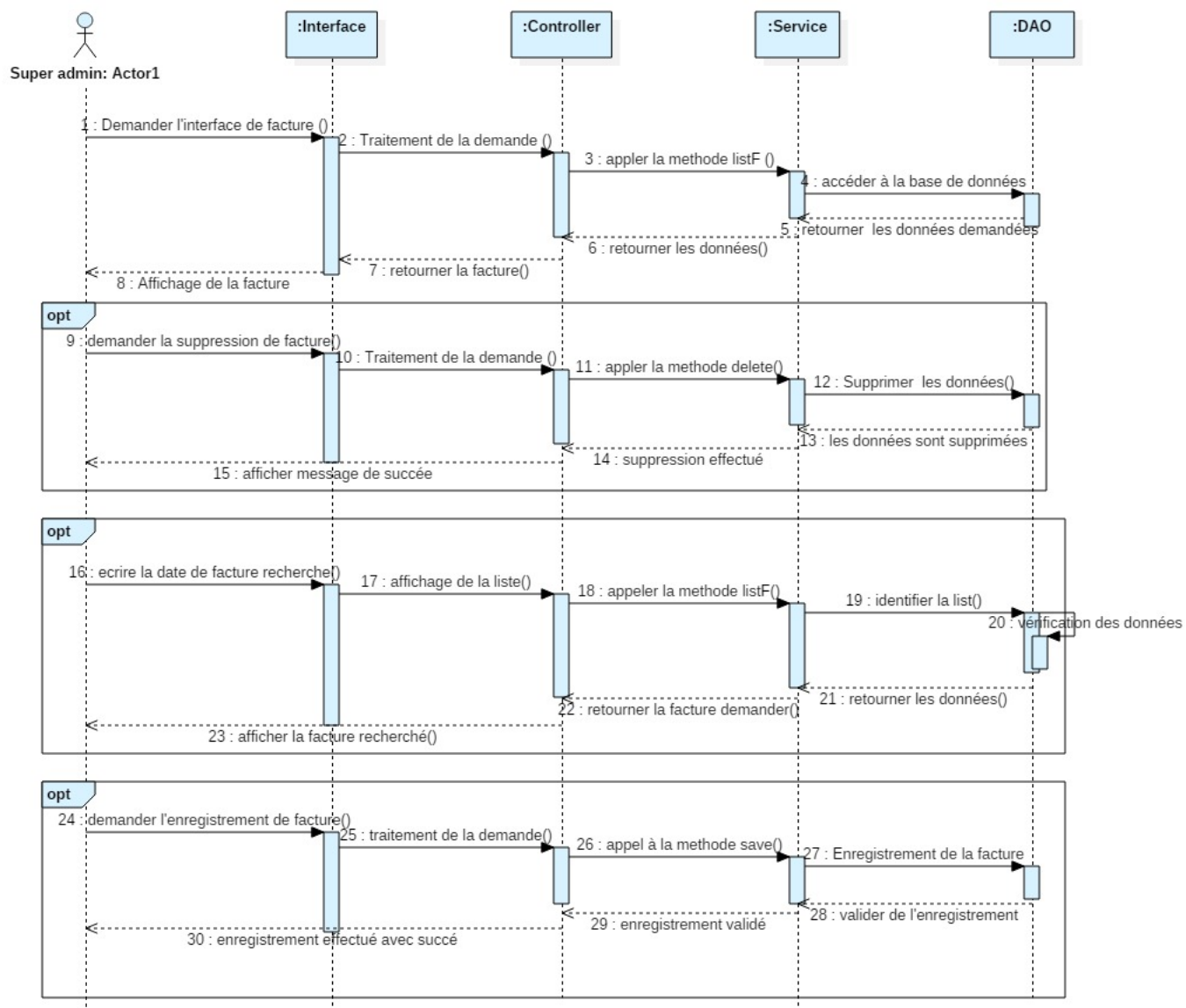


FIGURE 3.16 – Diagramme de séquence ‘Editer facture’



## **Conclusion**

La mise en œuvre d'une conception bien organisée et détaillée nous a facilité les tâches de développement et d'implémentation de notre application et ceci à l'aide de l'imagination des scénarios possibles et des différentes fonctionnalités. Nous avons décrit les différents éléments de l'étude conceptuelle.

Dans le chapitre suivant, nous allons passer à la description des étapes de réalisation de notre application.

## CHAPITRE 4

## RÉALISATION

### Introduction

Une fois les besoins sont définis et la phase de conception est détaillée, nous allons entamer dans ce chapitre la partie réalisation et implémentation qui décrit la phase la plus importante dans le cycle de vie d'une telle application.

Ce chapitre sert à déterminer en premier lieu la partie acquisition des données, puis nous nous intéresserons à la description et l'explication de fonctionnement des interfaces implémentées.

### 4.1 Acquisition des données

Pour le développement de notre système de suivi de consommation, nous avons utilisé une carte Nano Arduino comme un simulateur à travers le port série pour restituer les données provenant des différents capteurs installés sur les compteurs électriques. La récupération des données, donc la communication de notre application avec le Nano Arduino, nécessite l'utilisation d'un API externe qui puisse remplir ce rôle. Après pas mal de recherches nous avons choisi d'utiliser l'API Java RXIX permettant d'établir une connexion avec le port COM série du Nano Arduino puis la communication avec celui-ci en envoyant /recevant des données.



FIGURE 4.1 – NanoArduino

## 4.2 Présentation de l'application

L'authentification est la première interface de notre application. Suivant les données introduites dans les deux champs (login et mot de passe) on peut accéder à trois sessions :

- Session Super admin
- Session Administrateur
- Session Agent



FIGURE 4.2 – Interface d'authentification

- **Session Super admin**

- **Gestion des utilisateurs :**

Cette interface fournit les moyens et les fonctionnalités nécessaires pour contrôler et consulter les utilisateurs du système.

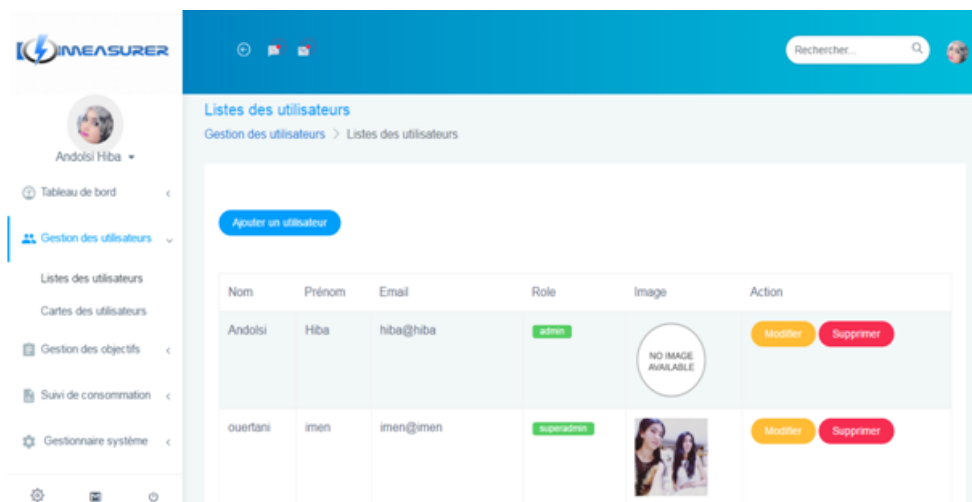


FIGURE 4.3 – Gestion des utilisateurs

Pour rechercher un utilisateur, il faut écrire son nom dans la barre de recherche, de cette façon le système va afficher tous les utilisateurs avec ce nom dans la liste.

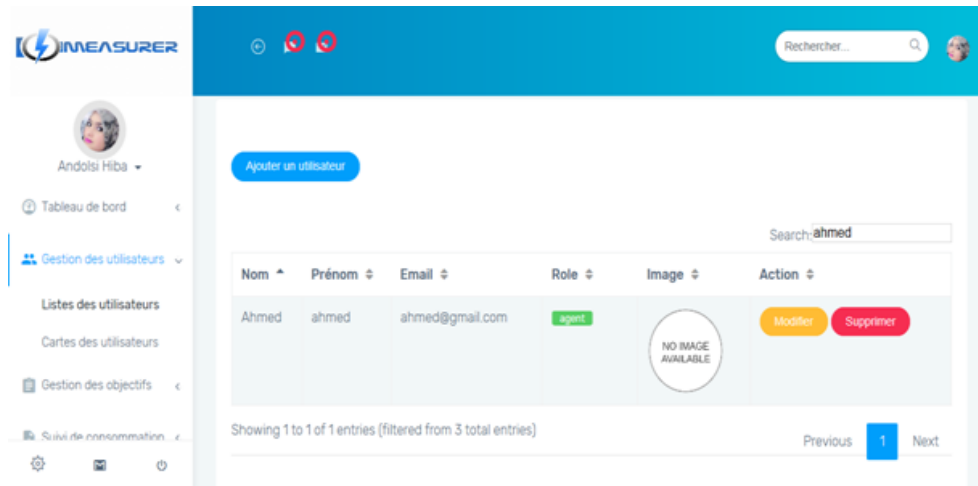


FIGURE 4.4 – Rechercher un (des) Utilisateurs

Pour ajouter un nouvel utilisateur, le Super admin doit remplir le formulaire avec les coordonnées de l'utilisateur.

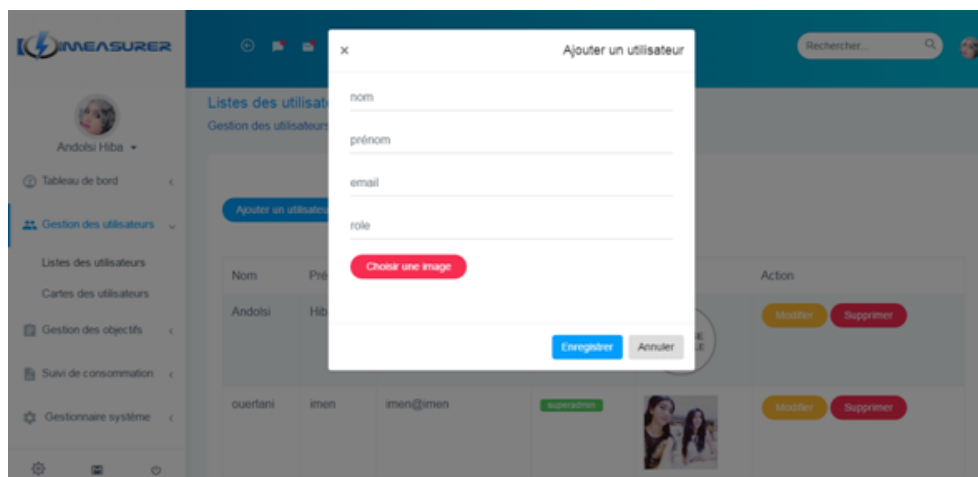


FIGURE 4.5 – Ajouter un utilisateur

Si on veut modifier un utilisateur, le Super Admin doit le sélectionner dans la liste. Automatiquement le formulaire sera affiché rempli avec les coordonnées de cet utilisateur. De cette façon il peut le modifier puis l'enregistrer.

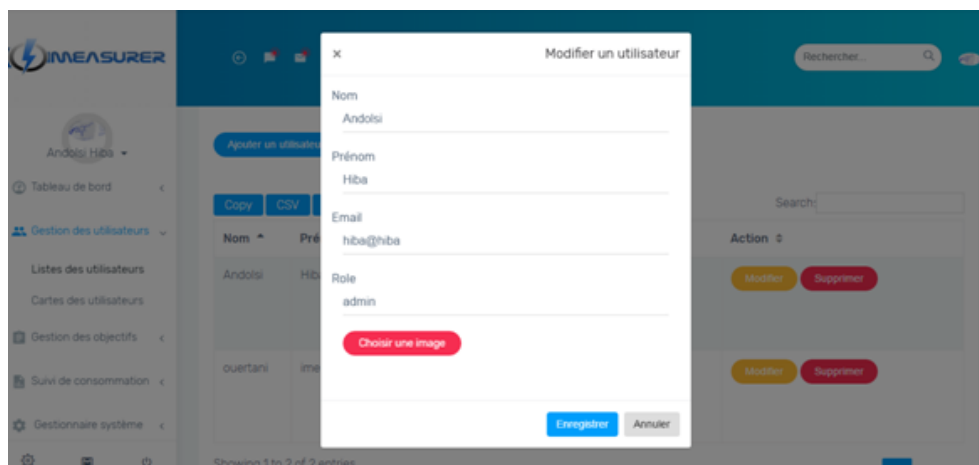


FIGURE 4.6 – Modifier un utilisateur

Afin de supprimer un utilisateur, il suffit de la sélectionner dans la liste et de cliquer sur le bouton «Supprimer ».

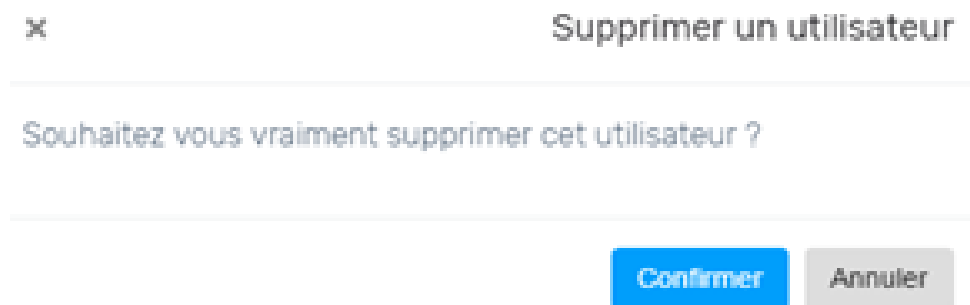


FIGURE 4.7 – Supprimer un utilisateur

– **Gérer les zones :**

Dans cet onglet, l'utilisateur peut ajouter une zone et lui affecter un gérant associé.

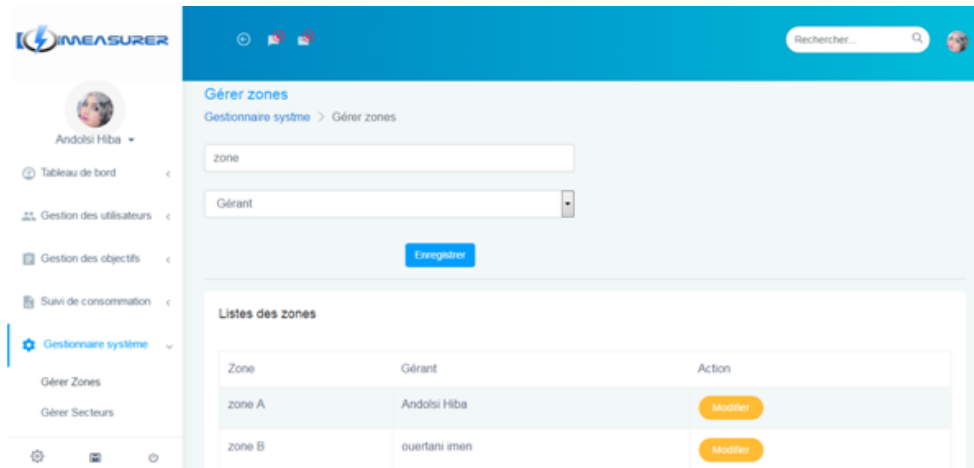


FIGURE 4.8 – Gérer les zones

– **Editer Factures :**

En cliquant sur « Facture » le Super admin peut consulter, enregistrer, rechercher ou supprimer des estimations de facture.

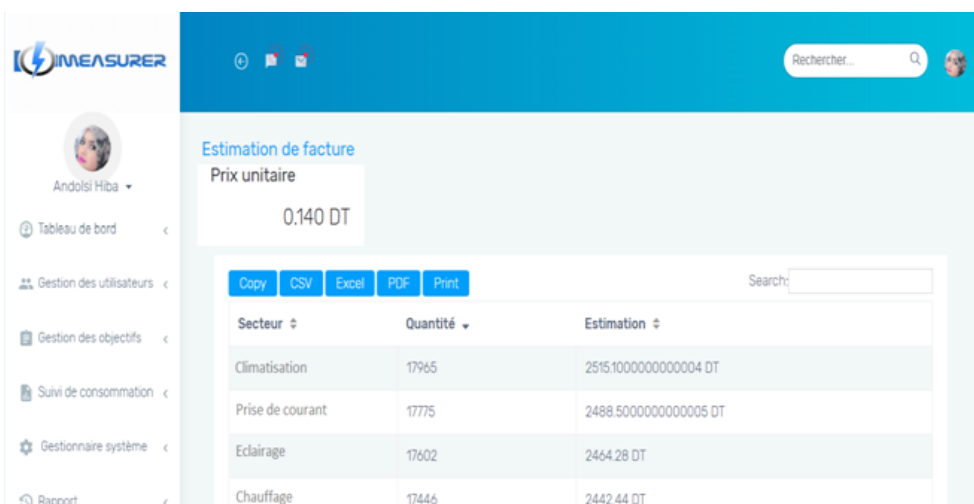


FIGURE 4.9 – Editer facture

- **Session Administrateur**

- **Gérer les objectifs :**

A partir de cette interface, l'utilisateur peut fixer les seuils et voir la liste des seuils fixés précédemment.

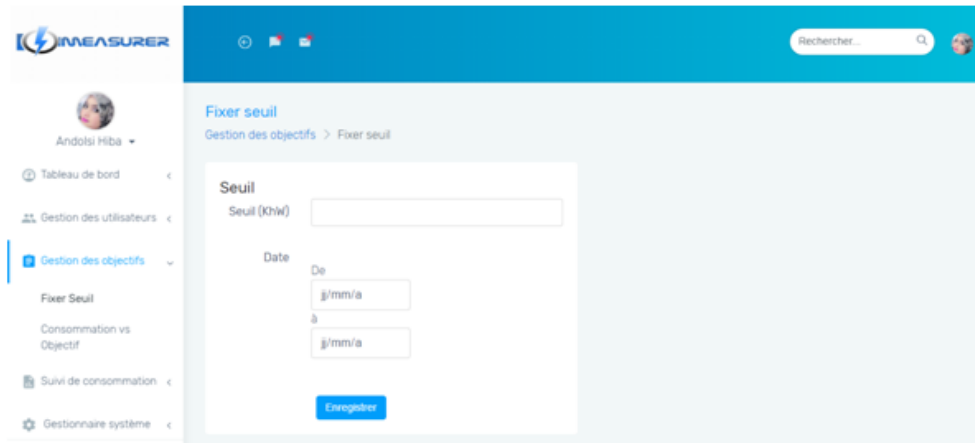
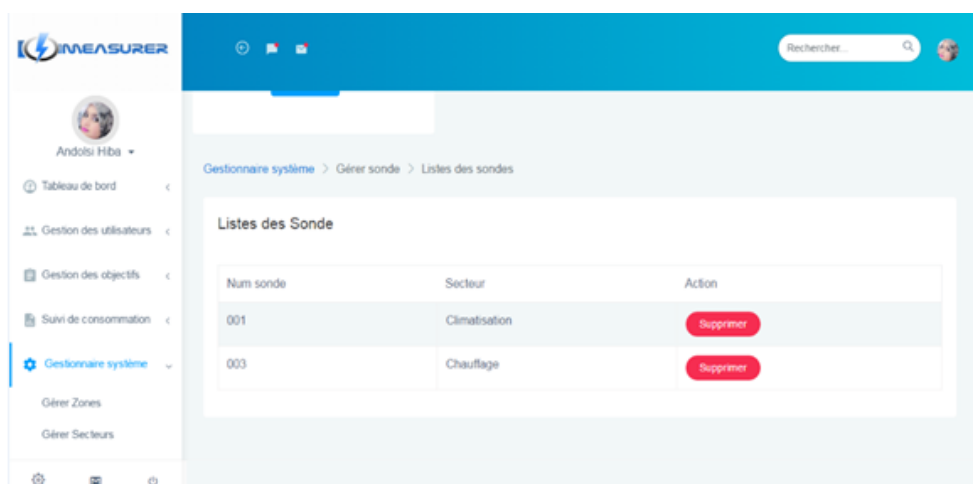


FIGURE 4.10 – Gérer es objectifs

- **Gérer les sondes :**

Dans l'onglet suivant, l'administrateur va ajouter des sondes, les consulter ou les supprimer.



Num sonde	Secteur	Action
001	Climatisation	Supprimer
003	Chauffage	Supprimer

FIGURE 4.11 – Consulter sondes



Pour ajouter une nouvelle sonde, l'administrateur doit remplir le formulaire avec les champs correspondants.

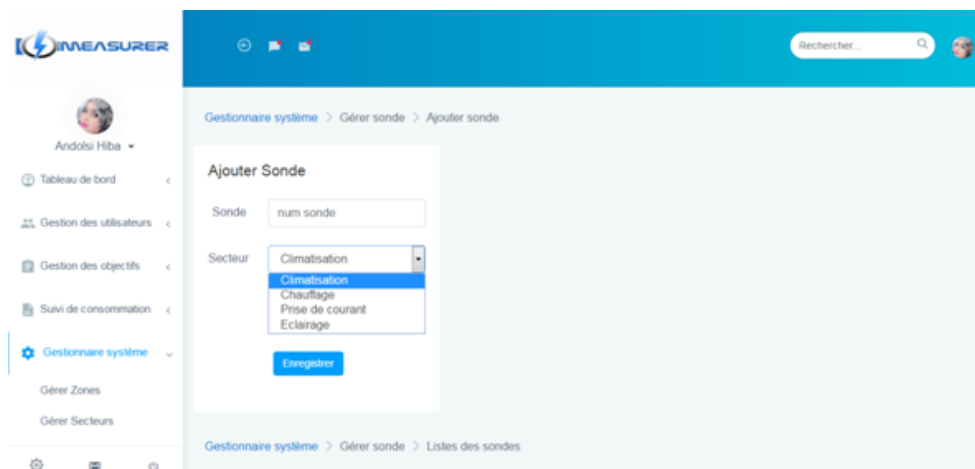


FIGURE 4.12 – Ajouter sonde

- **Session Agent**

Ces deux interfaces sont accessibles à n'importe quel utilisateur : SuperAdmin, Administrateur ou Agent.

- **Consulter l'état de consommation :**

Avec un simple clic sur l'onglet « suivi de consommation », l'utilisateur peut examiner l'état de son consommation.

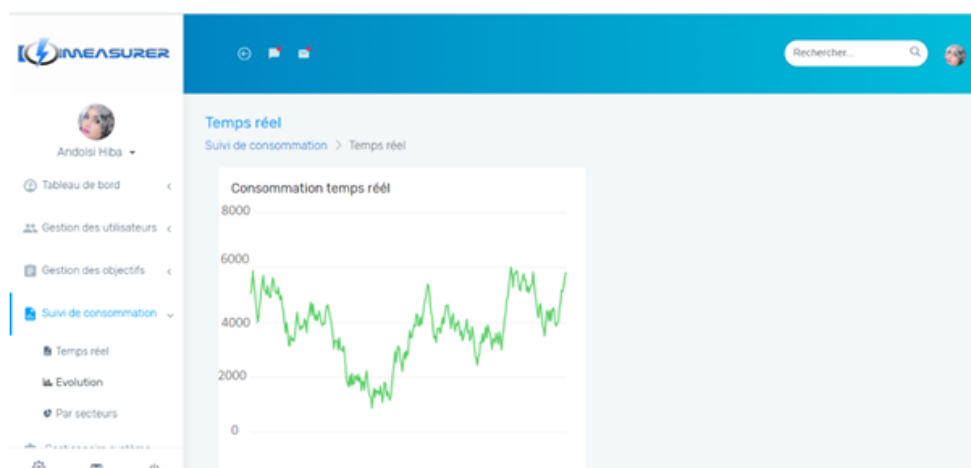


FIGURE 4.13 – Consommation temps réel

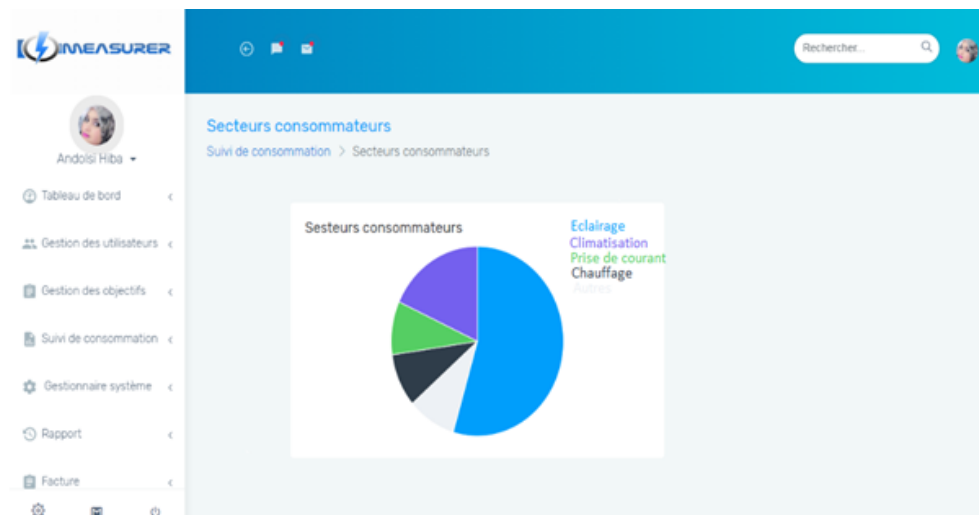


FIGURE 4.14 – Consommation par secteur

## Conclusion

A travers ce chapitre, nous avons détaillé la réalisation de notre application en présentant quelques interfaces graphiques et en décrivant brièvement comment nous avons planifié notre application.

## CONCLUSION GÉNÉRALE

L'objectif de notre projet était de réaliser un système de suivi de consommation d'électricité au sein de la société Whitecape Technologies.

Notre système sert à restituer la consommation en temps réel et les exploiter pour générer des statistiques, des rapports et des factures qui permettent aux décideurs (Super admin, Administrateur) de prendre les bonnes décisions concernant leurs consommations.

Nous avons commencé par une récolte des informations nécessaires pour l'élaboration d'une étude préliminaire ainsi qu'une présentation des outils de développement qu'on a eu recours pour arriver à développer « **Imeasurer** ». Par la suite, nous nous sommes intéressées à une étude conceptuelle qui nous a permis d'approfondir nos connaissances en modélisation UML. Le dernier volet de notre projet était la partie réalisation qui a été consacré à la présentation des quelques interfaces de notre application.

Ce projet nous a permis de découvrir un nouveau domaine de travail, de s'éloigner des projets traditionnels et d'entrer directement dans le monde de professionnalisme. Aussi, il nous a permis d'avoir des nouvelles connaissances dans le développement web en termes de Spring, Maven, Hibernate, JPA...

Comme perspectives, nous pouvons améliorer notre travail en réalisant une application mobile, en ajoutant des nouvelles fonctionnalités.

Enfin, Nous espérons que nos efforts donnent satisfaction aux membres de jury, à nos encadrants et tous nos enseignants.



## BIBLIOGRAPHIE

- [1] Guide pour mvc, dernière consultation 10/05/2017. <http://esi.namok.be/docs/Guide-MVC.pdf>.
- [2] Thierry Groussard. *Les fondamentaux du langage Java*. Edition ENI, 2011.
- [3] Arnaud Cogoluègues. Thierry Templier. Julien Dubois. Jean-Philippe Retailé. *Spring par la pratique*. Eyrolles, 09/07/2009.