



**Université Ibn
Zohr Faculté
des Sciences à
Agadir**



Centre d'Excellence IT

**Master d'Excellence
Ingénierie Informatique et Systèmes
Embarqués - IISE**

Module : Systèmes répartis et distribués

Rapport

Gestion de la consommation énergétique d'une maison

Soutenu le : 24/06/2025, Par :

Kazaz Mariam

Ben-kari Fatima Ezzahra

EL GOUMIRI Imane

Lamahad Maryam

Encadré par :

Pr. IDRAIS

Pr. ZAKARIA RIDA

Année universitaire : 2024/2025

Remerciements

Nous tenons à exprimer nos plus sincères remerciements à **Monsieur Idrais**, enseignant au sein de notre établissement, pour la qualité de ses cours, son engagement pédagogique, et ses orientations précieuses qui ont enrichi notre compréhension des systèmes distribués et des technologies mises en œuvre.

Nous remercions également à **Monsieur Zakaria Rida**, notre encadrant, pour sa disponibilité, ses conseils avisés, son accompagnement rigoureux et bienveillant tout au long de ce projet. Grâce à sa guidance, nous avons pu avancer avec clarté, méthode et motivation.

Nos remerciements s'adressent également à l'ensemble du corps professoral de la **Faculté des Sciences d'Agadir**, pour les connaissances et les valeurs transmises tout au long de notre formation.

Enfin, nous remercions chaleureusement nos familles, nos camarades et toutes les personnes qui nous ont soutenus de près ou de loin durant cette aventure académique.

Table des matières

Remerciements.....	1
Liste des figures	3
Introduction Générale.....	4
Résumé	5
Chapitre 1 : Contexte énergétique et analyse du système.....	6
1.1 Contexte général	6
1.2 Problématique	6
1.3 Objectifs du projet	6
1.4 Enjeux technologiques et énergétiques.....	6
Chapitre 2 : Analyse et Conception du Système.....	8
Introduction	8
2.1 Analyse fonctionnelle	8
2.1.1 Description des besoins.....	8
2.1.2 Identification des acteurs.....	9
2.2 Conception UML	10
2.2.1 Présentation du langage UML.....	10
2.2.2 Diagramme de cas d'utilisation.....	10
2.2.3 Diagramme de classes	10
2.2.4 Diagramme de séquence	11
2.2.5 Diagramme de déploiement.....	12
Chapitre 3 : Réalisation, tests et résultats	13
3.1 Environnement de développement	13
3.2 Visualisation et résultats	17
3.3 Tests réalisés	18
Conclusion générale.....	19

Liste des figures

Figure 1: Diagramme de cas d'utilisation	10
Figure 2:Diagramme de classe.....	11
Figure 3:Diagramme de séquence.....	12
Figure 4:Diagramme de déploiement	12
Figure 5:JAVA(JDK)	13
Figure 6:JAVA RMI	13
Figure 7:HTML-JS-CSS	14
Figure 8: Tomcat.....	14
Figure 9:VS-Code	15
Figure 10:Maven.....	15
Figure 11:GIT	15
Figure 12:StarUml.....	16
Figure 13:Dashboard	17
Figure 14:Recommandations	18
Figure 15:Messages d'alertes	18
Figure 16:Détection de faible consommation.....	18

Introduction Générale

De nos jours, la consommation énergétique est devenue un enjeu fondamental, notamment dans le cadre de la transition vers des modes de vie plus durables et plus respectueux de l'environnement. L'augmentation constante de la demande énergétique, en particulier dans les habitations modernes, soulève des défis importants en matière de gestion, de contrôle et d'optimisation de l'usage de l'électricité.

La maison, en tant qu'unité de consommation de plus en plus connectée, constitue un terrain propice à l'intégration de systèmes intelligents capables de surveiller et d'adapter en temps réel les comportements de consommation. Ces systèmes doivent permettre aux utilisateurs de visualiser leur consommation, de recevoir des alertes en cas d'anomalie, et de bénéficier de recommandations pour réduire les dépenses énergétiques.

Dans ce contexte, notre projet s'inscrit dans la volonté de développer une solution logicielle distribuée, s'appuyant sur la technologie **Java RMI**, afin de gérer efficacement la consommation énergétique d'une maison. Le système mis en place permet la collecte de données simulées, leur traitement à distance, et l'affichage des résultats à travers une interface web conviviale.

Ce rapport présente en détail le déroulement de ce projet, depuis l'analyse des besoins jusqu'à la mise en œuvre technique, en passant par la conception de l'architecture distribuée. Il met également en lumière les résultats obtenus et propose des pistes d'amélioration pour de futures évolutions.

Résumé

Dans un contexte où la maîtrise de la consommation énergétique est devenue une priorité mondiale, les foyers sont appelés à adopter des solutions intelligentes pour gérer leurs dépenses en électricité. Ce projet s'inscrit dans cette dynamique et propose le développement d'un système de gestion de la consommation énergétique d'une maison à travers une architecture distribuée basée sur Java RMI.

L'objectif principal est de surveiller la consommation des différents appareils domestiques, d'analyser les données collectées afin de détecter les anomalies, et de générer des recommandations personnalisées. Le système repose sur quatre modules principaux : un serveur de surveillance, un serveur d'analyse, un serveur de recommandation, et un serveur de gestion. Les données de consommation sont simulées via des fichiers textes, puis transmises au serveur d'analyse qui déclenche des alertes en cas de dépassement de seuil.

L'ensemble du système est accompagné d'une interface web ergonomique, développée avec le template AdminLTE, permettant à l'utilisateur de visualiser les statistiques de consommation à travers des tableaux et des graphiques dynamiques.

Les résultats obtenus démontrent l'efficacité du système à identifier les comportements énergivores et à fournir des recommandations adaptées, ouvrant ainsi la voie à une gestion plus responsable et intelligente de l'énergie au sein des habitations.

Chapitre 1 : Contexte énergétique et analyse du système

1.1 Contexte général

La consommation d'énergie est devenue un défi majeur du XXI^e siècle. Face à la croissance démographique, à l'urbanisation rapide et à la numérisation des foyers, les besoins en énergie électrique augmentent de façon constante. Dans ce contexte, les maisons modernes sont dotées de nombreux équipements électriques et électroniques, souvent énergivores, ce qui engendre une augmentation significative de la facture énergétique et de l'empreinte carbone.

La maison connectée, ou « smart home », est une réponse à cette problématique. Elle repose sur l'intégration de technologies capables de mesurer, contrôler et optimiser en temps réel l'usage de l'énergie. Ce projet vise à mettre en place un tel système, en s'appuyant sur une architecture distribuée permettant de surveiller la consommation électrique, de détecter les anomalies et d'informer l'utilisateur via des recommandations ciblées.

1.2 Problématique

Dans une maison classique, il est difficile de connaître précisément quels appareils consomment le plus, à quels moments de la journée, et avec quel impact sur la facture d'électricité. L'utilisateur ne dispose généralement que d'un relevé global de consommation sans détails sur les usages spécifiques. Cette absence de visibilité limite toute tentative d'optimisation énergétique.

La problématique que nous nous proposons de traiter est donc la suivante :

Comment concevoir un système intelligent, distribué et accessible, permettant de surveiller la consommation énergétique d'une maison, de détecter les usages excessifs, et de fournir à l'utilisateur des recommandations utiles en vue d'améliorer son efficacité énergétique ?

1.3 Objectifs du projet

Les objectifs principaux de notre projet sont :

- Mettre en place un système modulaire basé sur l'architecture Java RMI
- Lire automatiquement des données de consommation simulées
- Transmettre ces données à un serveur d'analyse distant
- Détecter les anomalies ou les dépassements de seuils prédéfinis
- Envoyer des recommandations appropriées via un module dédié
- Offrir une interface utilisateur simple et intuitive pour visualiser les données et les alertes.

1.4 Enjeux technologiques et énergétiques

La maîtrise de la consommation énergétique dans les maisons modernes nécessite l'intégration de solutions technologiques avancées capables de collecter, traiter et exploiter

des données en temps réel. Le recours à une architecture distribuée, telle que Java RMI, offre des avantages significatifs :

- **Modularité** : Les différents modules (surveillance, analyse, recommandation) sont déployés sur des serveurs distincts, facilitant la maintenance et l'évolution du système.
- **Interopérabilité** : Grâce à RMI, les objets distants peuvent communiquer aisément, indépendamment de leur localisation physique.
- **Scalabilité** : Le système peut être étendu pour gérer davantage de capteurs ou d'utilisateurs sans dégradation notable des performances.

Sur le plan énergétique, une gestion fine de la consommation permet de réduire la facture électrique, de limiter la surconsommation en période de pointe, et de participer activement à la réduction des émissions de gaz à effet de serre.

Chapitre 2 : Analyse et Conception du Système

Introduction

La réussite de tout projet logiciel dépend d'une bonne compréhension des besoins et d'une modélisation rigoureuse des fonctionnalités attendues. Ce chapitre est divisé en deux parties principales :

- La première concerne l'**analyse fonctionnelle** du système, à travers l'identification des besoins et des acteurs.
- La deuxième est dédiée à la **modélisation UML** du système que nous avons conçu pour surveiller et optimiser la consommation énergétique d'une maison.

2.1 Analyse fonctionnelle

2.1.1 Description des besoins

Surveillance de la consommation énergétique : Le système doit être capable de collecter régulièrement les données de consommation électrique provenant de plusieurs appareils domestiques (réfrigérateur, climatiseur, éclairage, etc.). Ces données peuvent représenter la quantité d'électricité consommée en temps réel ou sur une période donnée.

L'objectif est d'avoir une vision détaillée de la consommation par appareil, plutôt que d'un simple relevé global. Cela permettra :

- D'identifier les appareils énergivores.
- De comprendre les habitudes de consommation.
- De préparer les données pour une analyse plus fine.

Dans notre projet, la collecte est simulée via des fichiers textes qui contiennent ces données. Le système doit donc lire ces fichiers régulièrement et extraire les informations pertinentes.

Transmission sécurisée des données : Une fois les données collectées, il faut qu'elles soient transmises de manière fiable et sécurisée aux différents modules du système, notamment au serveur d'analyse. Pour cela, on utilise la technologie Java RMI (Remote Method Invocation) qui permet à des objets Java situés sur des machines différentes de communiquer.

Cette transmission doit garantir :

- La fiabilité des échanges, sans perte ni altération des données.
- La communication en temps réel ou quasi temps réel.
- La possibilité d'extension future à plusieurs serveurs ou utilisateurs.

Ainsi, chaque module peut fonctionner indépendamment, et la transmission via RMI assure que les données sont envoyées correctement.

Analyse des données : Le module d'analyse reçoit les données de consommation et doit les traiter pour détecter des comportements anormaux ou des dépassements de seuils prédéfinis. Par exemple, si un appareil consomme beaucoup plus que la normale, cela peut indiquer :

- Une panne ou un dysfonctionnement.
- Un oubli d'éteindre un appareil.
- Une mauvaise utilisation ou un gaspillage d'énergie.

Le traitement des données comprend :

- La comparaison des valeurs reçues aux seuils définis.
- L'identification d'anomalies selon des règles.
- La préparation d'alertes à transmettre au module de recommandation.

Génération d'alertes et recommandations : Lorsque le système détecte une anomalie ou une consommation excessive, il doit alerter l'utilisateur en temps réel pour lui permettre d'agir rapidement.

Mais il ne s'agit pas seulement d'alerter : le système fournit aussi des recommandations personnalisées, par exemple :

- Éteindre un appareil non utilisé.
- Limiter l'usage de certains équipements aux heures creuses.

2.1.2 Identification des acteurs

Afin de structurer correctement notre système, il est important de déterminer les **acteurs** impliqués dans les différents processus. Un acteur est une entité (humaine ou logicielle) qui interagit directement avec le système.

Les principaux acteurs dans notre application sont :

- **Utilisateur final (propriétaire de la maison) :** consulte la consommation d'énergie, reçoit des alertes et suit les recommandations.
- **Serveur de surveillance :** lit les fichiers de consommation (fichiers texte ou CSV) et envoie les données via RMI.
- **Serveur d'analyse :** reçoit les données, les analyse, détecte les anomalies et décide si une alerte doit être générée.
- **Serveur de recommandation :** reçoit les alertes et génère des messages personnalisés à destination de l'utilisateur.
- **Serveur de gestion :** centralise les données collectées, les enregistre dans une base de données, et les met à disposition pour consultation et analyse via l'interface web.

Cette identification permet de construire les diagrammes UML nécessaires à la modélisation fonctionnelle du système.

2.2 Conception UML

2.2.1 Présentation du langage UML

UML (Unified Modeling Language) est un langage de modélisation standardisé largement utilisé en ingénierie logicielle. Il permet de représenter graphiquement les différents aspects d'un système, facilitant la communication entre les membres de l'équipe et la compréhension globale du projet.

Dans notre projet, UML nous aide à formaliser les interactions entre les différents modules, à décrire la structure des données, ainsi que les comportements attendus.

2.2.2 Diagramme de cas d'utilisation

Ce diagramme illustre les différentes interactions entre les acteurs du système et les fonctionnalités offertes :

- **Acteurs :** Utilisateur final, Serveur de surveillance, Serveur d'analyse, Serveur de recommandation, Serveur de gestion.
- **Cas d'utilisation principaux :**
 - Consulter la consommation énergétique
 - Analyser les données et détecter les anomalies
 - Recevoir les alertes et recommandations
 - Contrôler les appareils

Ce diagramme offre une vue d'ensemble claire des responsabilités de chaque acteur.

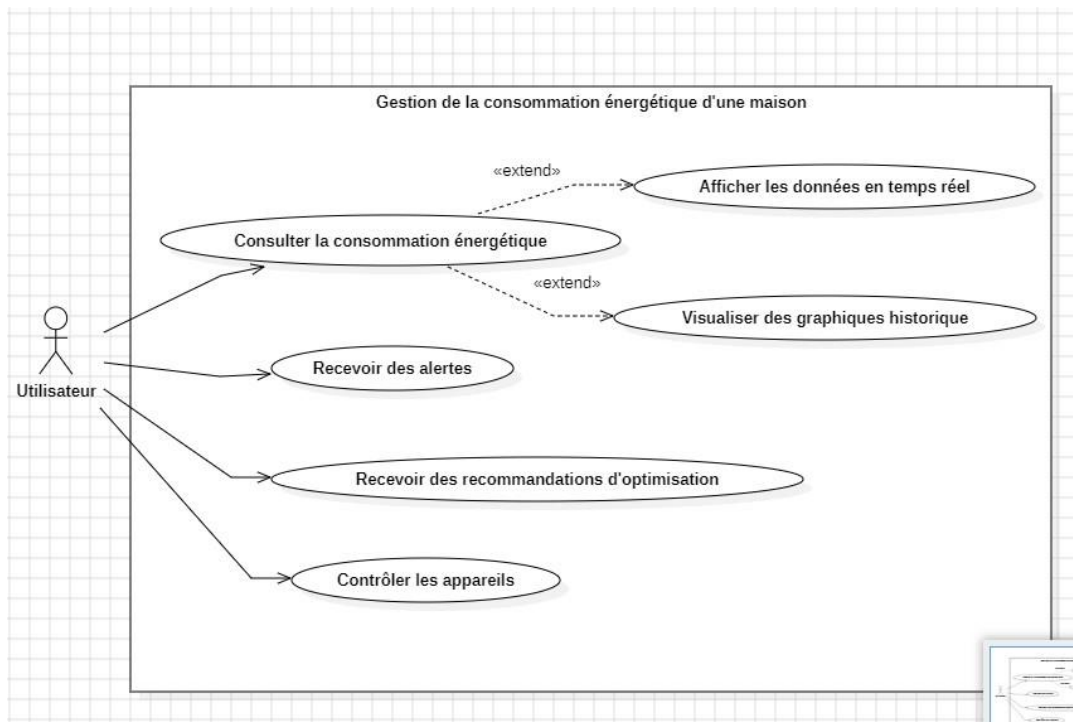


Figure 1: Diagramme de cas d'utilisation

2.2.3 Diagramme de classes

Le diagramme de classes présente les principales classes du système et leurs relations.

Les relations (association, dépendance) montrent comment les modules interagissent.

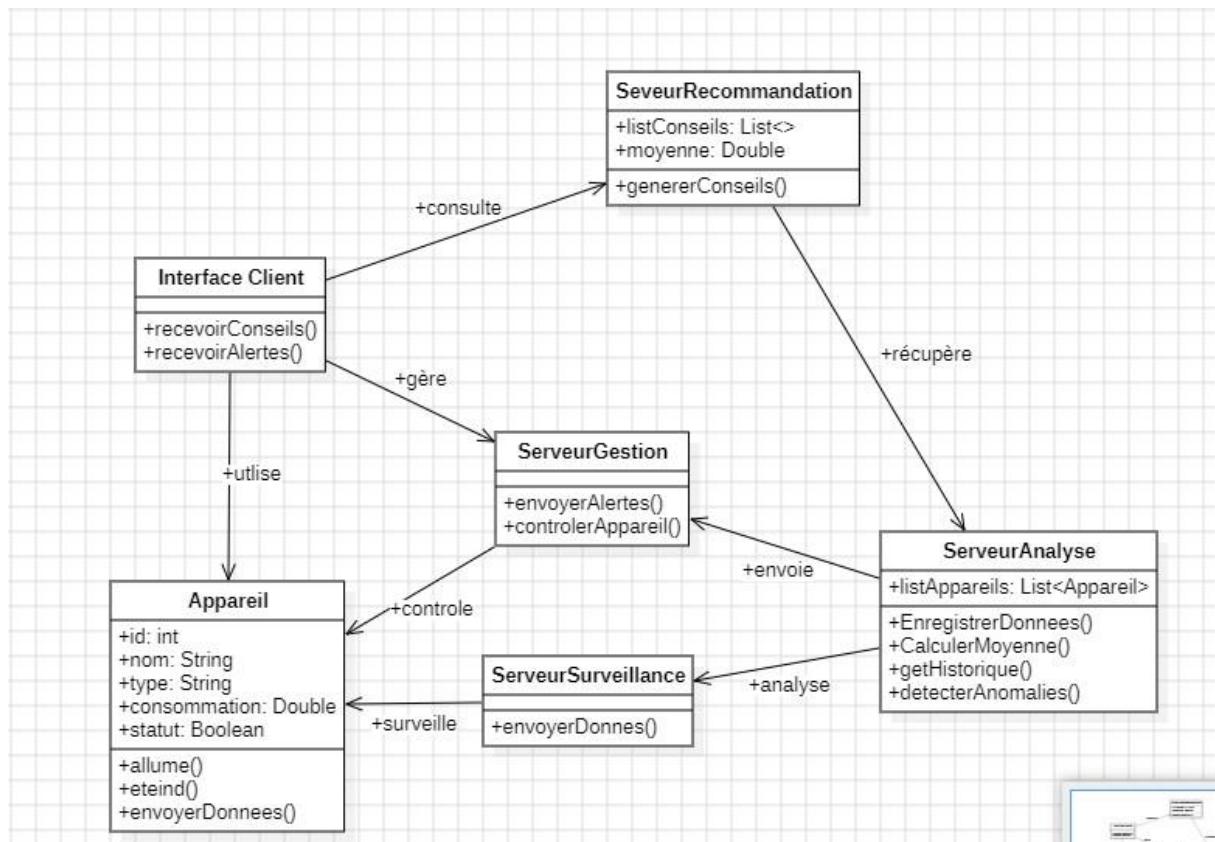


Figure 2:Diagramme de classe

2.2.4 Diagramme de séquence

Ce diagramme détaille la chronologie des interactions entre les modules lors d'un scénario typique, par exemple :

- Le serveur de surveillance lit les données
- Il envoie via RMI au serveur d'analyse
- Le serveur d'analyse compare aux seuils
- Si anomalie, envoie une alerte au serveur de recommandation
- Le serveur de recommandation génère un message et le transmet à l'interface utilisateur

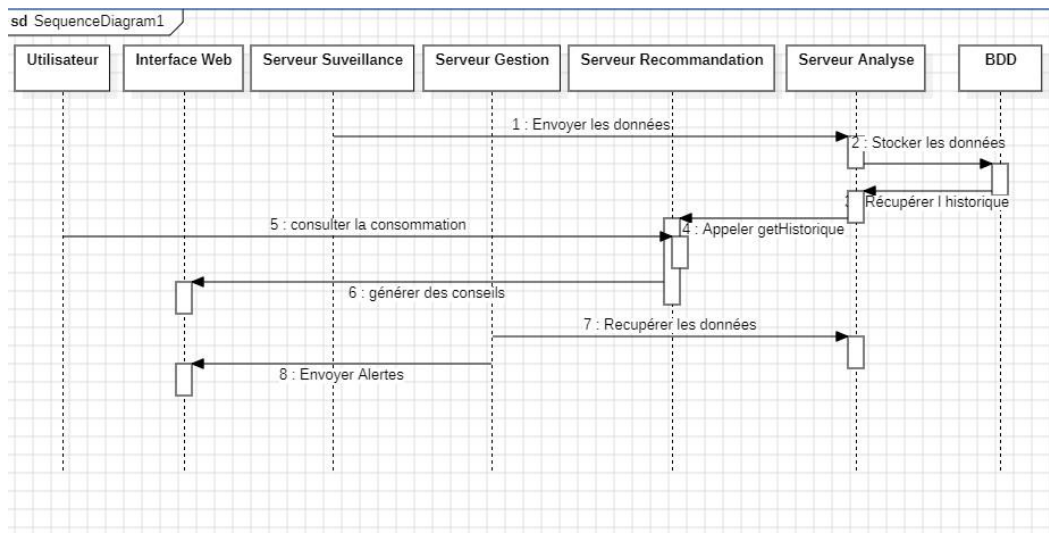


Figure 3:Diagramme de séquence

2.2.5 Diagramme de déploiement

Ce diagramme montre la disposition physique des serveurs et clients :

- Serveur Surveillance (machine 1)
- Serveur Analyse (machine 2)
- Serveur Recommandation (machine 3)
- Serveur Gestion / Base de données
- Client Web

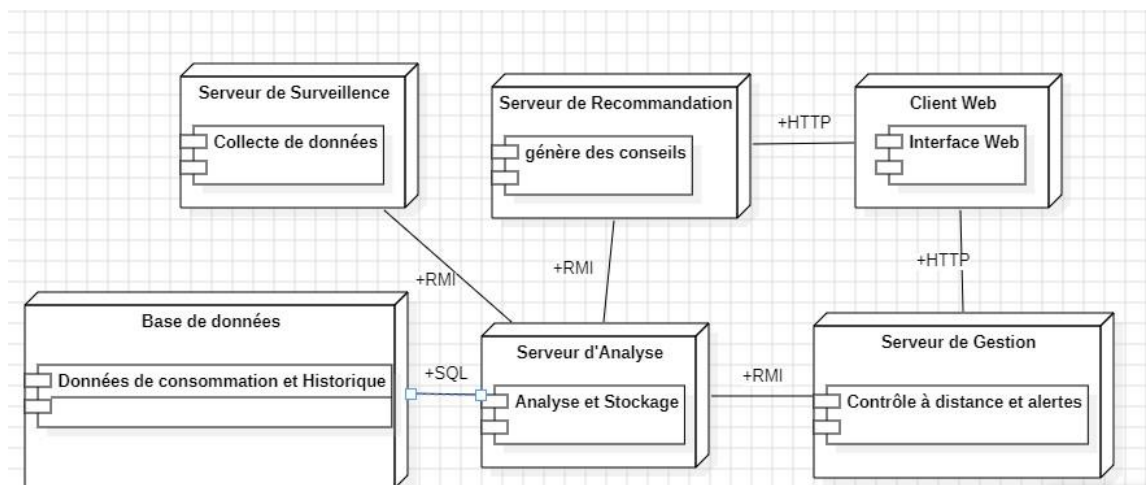


Figure 4:Diagramme de déploiement

Chapitre 3 : Réalisation, tests et résultats

3.1 Environnement de développement

Le développement du système distribué a été effectué dans un environnement Java, propice à la création d'applications réparties et modulaires. Plusieurs outils, bibliothèques et composants ont été mobilisés afin d'assurer la cohérence, la portabilité et la maintenabilité du système. Ci-dessous, une description détaillée des choix techniques :

- **Langage de programmation : Java (JDK 17)**

Le choix de Java s'est imposé naturellement en raison de la robustesse de la plateforme, de sa portabilité entre différents systèmes d'exploitation, et de son support natif de la technologie RMI. L'utilisation de la version JDK 17 garantit une compatibilité avec les fonctionnalités récentes tout en assurant une bonne stabilité.

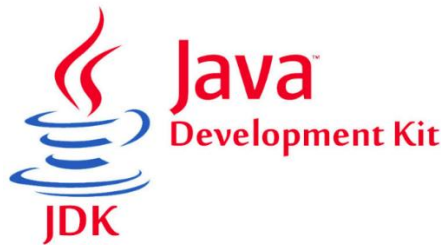


Figure 5:JAVA(JDK)

- **Technologie distribuée : Java RMI (Remote Method Invocation)**

RMI permet d'invoquer des méthodes sur des objets distants, situés sur d'autres machines. Cette technologie a été utilisée pour mettre en place la communication entre les différents modules du système (surveillance, analyse, recommandation, gestion). Chaque module fonctionne sur une JVM indépendante, rendant le système réellement distribué.



Figure 6:JAVA RMI

- **Interface Web : HTML, CSS, JSP (Java Server Pages)**

L'interface utilisateur a été développée en combinant HTML et CSS pour le rendu statique, et JSP pour l'intégration dynamique des données côté serveur. Cela permet à l'utilisateur de visualiser sa consommation énergétique, les alertes et recommandations de manière fluide via un navigateur.



Figure 7: HTML-JS-CSS

- **Intermédiaire de communication : Servlets Java**

Les **servlets** assurent le lien entre l'interface web (JSP) et les objets distants RMI. Elles reçoivent les requêtes de l'utilisateur, invoquent les méthodes RMI appropriées (via `Naming.lookup()`), puis transmettent les résultats à la page JSP.

- **Moteur de servlet : Apache Tomcat**

Tomcat a été utilisé comme serveur d'exécution pour les pages JSP. Il permet d'héberger les composants Web du système, tout en assurant le traitement des requêtes HTTP et l'interfaçage avec les modules RMI. Dans certains environnements de test, le moteur Jetty a également été utilisé pour des déploiements plus légers.



Figure 8: Tomcat

- **Éditeur de code : Visual Studio Code**

Visual Studio Code a été choisi pour son interface conviviale, ses extensions dédiées au développement Java, et sa légèreté. Il permet de gérer efficacement les fichiers sources, de lancer les commandes Maven, et de suivre le projet dans son ensemble.



Figure 9:VS-Code

- **Gestion de projet : Maven**

Maven a été utilisé pour automatiser la compilation, la gestion des dépendances, et la structure modulaire du projet. Grâce à son fichier `pom.xml`, il facilite l'organisation des différents composants et l'intégration continue.



Figure 10:Maven

- **Contrôle de version : Git et GitHub**

Le code source a été versionné à l'aide de Git, ce qui permet de garder un historique des modifications, de travailler en équipe et de synchroniser les contributions via GitHub.



Figure 11:GIT

- **Modélisation UML : StarUML**

Pour la phase d'analyse et de conception, nous avons utilisé **StarUML**, un outil de modélisation UML performant et intuitif. Il nous a permis de représenter graphiquement :

- les **cas d'utilisation** pour identifier les interactions entre les acteurs et le système,
- les **diagrammes de classes** pour modéliser les objets métier et leurs relations,
- les **diagrammes de séquence** pour décrire le déroulement des appels entre les modules RMI,
- ainsi que le **diagramme de déploiement** pour illustrer la distribution physique des composants sur différentes machines.

L'utilisation de StarUML a facilité la communication entre les membres du groupe et assuré une documentation claire et structurée, conformément aux bonnes pratiques d'ingénierie logicielle.



Figure 12:StarUml

- **Outils complémentaires :**
 - **Navigateur web** : pour tester l'interface utilisateur (Google Chrome et Mozilla Firefox principalement).
 - **Terminal système** : pour exécuter les différentes commandes (lancement des serveurs, compilation, etc.).
 - **rmiregistry** : registre RMI nécessaire au bon fonctionnement des appels distants. Il doit être lancé avant l'exécution des serveurs.

Particularités de l'architecture RMI :

L'utilisation de RMI impose une structure particulière dans le déploiement du système :

- Avant toute communication, le **registre RMI** (`rmiregistry`) doit être initialisé pour permettre l'enregistrement des objets distants.
- Chaque classe distante doit être **compilée avec `rmic`** (ou avec des outils récents qui l'intègrent automatiquement via Maven).
- Chaque serveur (surveillance, analyse, recommandation, gestion) est lancé séparément et enregistre ses services auprès du registre.
- Le client (interface Web ou autre) accède aux objets distants à travers leurs interfaces RMI.

Cette organisation permet de simuler un système réellement réparti, où chaque composant peut être exécuté sur une machine différente, tout en maintenant une interopérabilité fluide grâce aux interfaces Java.

3.2 Visualisation et résultats

La page Dashboard constitue l'élément central de l'interface utilisateur du système Smart Energy. Elle offre une vue synthétique, visuelle et interactive des données de consommation électrique, collectées et traitées par l'ensemble des serveurs du système.

Les données sont historisées dans une base de données MySQL via le serveur de gestion, et sont ensuite représentées graphiquement dans le navigateur de l'utilisateur grâce à des composants HTML, CSS, JSP et JavaScript.

Le Dashboard est structuré en quatre blocs principaux, chacun illustrant une dimension essentielle de la consommation :

- [Dernières 12 consommations](#)

Cette section affiche un graphique en ligne représentant les 12 dernières valeurs de consommation en kilowatts (kW). Elle permet de visualiser rapidement l'évolution temporelle et de détecter toute anomalie récente.

- [Moyenne par jour](#)

Ce graphique en barres verticales indique la consommation moyenne quotidienne, en kW, pour chaque jour analysé. Il permet de détecter les journées à forte consommation et de suivre les tendances d'une semaine à l'autre.

- [Pic max quotidien](#)

Cette visualisation montre, pour chaque jour, la valeur maximale atteinte dans la consommation. Elle est utile pour identifier les pics énergétiques inhabituels qui nécessitent une attention particulière.

- [Répartition Jour/Nuit](#)

Un diagramme circulaire (camembert) indique le ratio de consommation diurne et nocturne. Cette information est cruciale pour suggérer à l'utilisateur des optimisations, comme le décalage de l'utilisation de certains appareils vers les heures creuses.



Figure 13: Dashboard

3.3 Tests réalisés

Afin de valider le bon fonctionnement du système distribué, plusieurs scénarios de test ont été réalisés. Le serveur d'analyse a correctement détecté les cas de surconsommation en comparant la puissance transmise aux seuils définis (par exemple, > 300W). Dans ces cas, des alertes ont été générées automatiquement et transmises au serveur de gestion, qui les a affichées à l'utilisateur. Le système a également su détecter des situations de **faible** consommation ou d'appareils en veille, en proposant des recommandations intelligentes telles que l'arrêt ou la désactivation des équipements non essentiels. Ces tests ont permis de valider la chaîne complète de surveillance, d'analyse, de recommandation et d'affichage à travers l'interface

Recommandations intelligentes générées par le système selon l'analyse des seuils de consommation



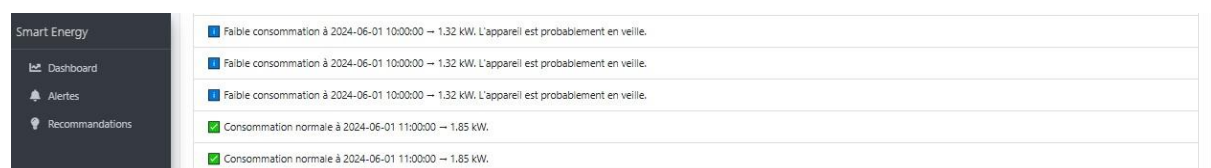
Recommandations pour la maison	
Conseils générés	
Moyenne générale de consommation :	3,07 kW
✓	Consommation normale à 2024-06-01 00:00:00 → 2,75 kW.
✓	Consommation normale à 2024-06-01 00:00:00 → 2,75 kW.
✓	Consommation normale à 2024-06-01 00:00:00 → 2,75 kW.
✓	Consommation normale à 2024-06-01 01:00:00 → 1,61 kW.
✓	Consommation normale à 2024-06-01 01:00:00 → 1,61 kW.
✓	Consommation normale à 2024-06-01 01:00:00 → 1,61 kW.
✓	Consommation normale à 2024-06-01 02:00:00 → 1,89 kW.
✓	Consommation normale à 2024-06-01 02:00:00 → 1,89 kW.
✓	Consommation normale à 2024-06-01 02:00:00 → 1,89 kW.
✓	Consommation normale à 2024-06-01 03:00:00 → 2,37 kW.

Figure 14:Recommandations



⚠	Haute consommation détectée à 2024-06-01 06:00:00 → 5,4 kW. Pensez à éteindre l'appareil.
⚠	Haute consommation détectée à 2024-06-01 06:00:00 → 5,4 kW. Pensez à éteindre l'appareil.
⚠	Haute consommation détectée à 2024-06-01 06:00:00 → 5,4 kW. Pensez à éteindre l'appareil.
✓	Consommation normale à 2024-06-01 07:00:00 → 3,96 kW.
✓	Consommation normale à 2024-06-01 07:00:00 → 3,96 kW.

Figure 15:Messages d'alertes



■	Faible consommation à 2024-06-01 10:00:00 → 1,32 kW. L'appareil est probablement en veille.
■	Faible consommation à 2024-06-01 10:00:00 → 1,32 kW. L'appareil est probablement en veille.
■	Faible consommation à 2024-06-01 10:00:00 → 1,32 kW. L'appareil est probablement en veille.
✓	Consommation normale à 2024-06-01 11:00:00 → 1,85 kW.
✓	Consommation normale à 2024-06-01 11:00:00 → 1,85 kW.

Figure 16:Détection de faible consommation

Conclusion générale

À l'ère de la transition énergétique et de la digitalisation croissante des foyers, la maîtrise de la consommation électrique est devenue une priorité tant économique qu'écologique. Dans ce contexte, notre projet a permis de concevoir et de développer un système distribué intelligent, capable de surveiller, analyser et recommander en temps réel des actions pour une meilleure gestion de l'énergie au sein d'une maison.

En s'appuyant sur la technologie **Java RMI**, nous avons mis en place une architecture modulaire, composée de plusieurs serveurs spécialisés (surveillance, analyse, recommandation, gestion), chacun jouant un rôle déterminé dans le traitement et la circulation des données énergétiques. Ce découpage a favorisé la séparation des préoccupations, la maintenabilité du code, et l'évolutivité du système.

La phase de conception UML a permis de formaliser nos idées, tandis que le développement en Java, appuyé par des technologies comme **JSP**, **HTML/CSS**, **Apache Tomcat** et **Maven**, a abouti à une solution fonctionnelle intégrant une interface web intuitive. L'ensemble du processus a été enrichi par l'utilisation d'outils professionnels tels que **StarUML** pour la modélisation, **GitHub** pour la collaboration, et **Visual Studio Code** pour le développement.

Les résultats obtenus démontrent la capacité du système à détecter des consommations anormales, à générer des alertes pertinentes, et à proposer des recommandations utiles à l'utilisateur. Ce projet constitue ainsi une base solide pour de futures évolutions, telles que l'intégration de capteurs physiques, l'utilisation d'algorithmes d'intelligence artificielle, ou encore le déploiement à grande échelle dans des environnements réels.

En somme, cette expérience nous a permis de mobiliser nos compétences en systèmes distribués, en modélisation logicielle et en développement web, tout en apportant une réponse concrète à une problématique actuelle et sociétale.