

**Master d'Excellence  
Ingénierie Informatique et Systèmes Embarqués - IISE  
Module : Distributed Systems**

# **RAPPORT DE PROJET IOT**

**Sous le thème :**

## **Distributed Smart Irrigation System Based on ESP32**

**Réalisée par :**

**OGAD NOUREDDINE**

**ABBOUD AYA**

**AAMER AYA**

**EL MORNAN WIAME**

**TARKANY MOHAMED**

**Mustapha Chouiar**

**KHADIJA LAHJAOUZI**

**Oualid OUARRACH**

**Imane Abboudrar**

**Encadré par : Pr. Youssef EL HABOUZ**

**Année universitaire : 2025/2026**

## Table des matières

<b>Chapitre 1 Introduction.....</b>	<b>2</b>
1. Contexte général du projet.....	2
2. Technologies utilisées.....	2
3. Contexte et motivation.....	3
<b>Chapitre 2 Architecture Logicielle.....</b>	<b>4</b>
1. Description générale de l'architecture du système.....	4
2. Déploiement distribué.....	6
<b>Chapitre 3 Diagrammes UML.....</b>	<b>9</b>
1. Introduction:.....	9
2. Diagramme de cas d'utilisation.....	9
3. Diagramme de classes.....	9
4. Diagramme de séquence.....	10
5. Diagramme de déploiement.....	12
6. Conclusion.....	13
<b>Chapitre 4 Réalisation et présentation de l'application.....</b>	<b>14</b>
1. Présentation générale de l'application.....	14
2. Fonctionnalités principales.....	14
3. Présentation des interfaces.....	15
4. Montage et Déploiement du Système d'Irrigation.....	21
<b>Chapitre 5 Conclusion Générale du Projet.....</b>	<b>22</b>

# Chapitre 1 Introduction

## 1. Contexte général du projet

Ce projet consiste à concevoir et implémenter un système d'irrigation intelligent distribué, capable de collecter des données environnementales en temps réel, de prendre des décisions automatiques, et de fournir une interface de supervision et de contrôle accessible via le web et une application mobile.

L'objectif principal est de mettre en pratique les concepts des systèmes distribués, en particulier la communication par appels distants, tout en répondant à un besoin concret : l'optimisation de l'irrigation en fonction de paramètres mesurés (humidité du sol, niveau d'eau, température, pluie).

Le système repose sur une architecture distribuée composée de plusieurs composants distincts :

- Un **ESP32** qui collecte les données des capteurs et envoie périodiquement des mesures.
- Un **serveur central RMI**, qui joue le rôle de cœur du système et assure :
  - la logique métier
  - la prise de décision (mode automatique ou manuel)
  - la gestion des alertes
  - et l'accès à la base de données.
- Une **passerelle (Gateway)** exposée via des services REST, utilisée comme intermédiaire entre le serveur RMI et les interfaces clientes.
- Une **interface web** (React) et une **application mobile**, permettant à l'utilisateur de visualiser les données, consulter l'historique, recevoir des alertes et contrôler le système.

## 2. Technologies utilisées

Les principales technologies employées sont :

- **Java RMI (Remote Method Invocation)** : utilisé pour les appels distants entre la passerelle et le serveur central. Le serveur RMI contient toute la logique métier et l'accès aux données.
- **HTTP / REST** : utilisé entre les clients (ESP32, interface web, application mobile) et la passerelle.
- **Spring Boot** : utilisé pour implémenter la passerelle REST exposant les services vers l'interface web.
- **SQL Server** : utilisé pour stocker les lectures des capteurs, l'état des dispositifs, les décisions prises et les alertes.
- **React** : utilisé pour développer l'interface web de supervision.
- **Application mobile** : utilisée comme client léger pour la consultation et le contrôle du système.

### 3. Contexte et motivation

Un système distribué est nécessaire pour ce projet car les différents composants n'exécutent pas les mêmes responsabilités et peuvent être déployés sur des machines distinctes. Cette séparation permet :

- une **meilleure modularité** (chaque composant a un rôle bien défini)
- une **scalabilité** du système (ajout de nouveaux dispositifs ou clients)
- une **centralisation de la logique métier**, garantissant la cohérence des décisions
- une **tolérance aux pannes**, grâce à la séparation entre la collecte, le traitement et la visualisation des données.

Les principaux défis abordés dans ce projet sont :

- la gestion fiable des **communications distantes** entre plusieurs composants
- la synchronisation de l'état des dispositifs dans un environnement distribué
- la gestion des erreurs réseau et des reconnections RMI
- la diffusion d'événements en temps réel (alertes) vers les interfaces clientes
- et l'intégration cohérente entre RMI, REST et interfaces utilisateur.

## Chapitre 2 Architecture Logicielle

### 1. Description générale de l'architecture du système

#### Contenu

Le système d'irrigation intelligent est conçu selon une **architecture distribuée en couches**, où chaque composant possède un rôle clairement défini. Cette architecture vise à séparer la collecte des données, la logique métier, l'accès aux données et la présentation, afin d'assurer la robustesse, la maintenabilité et l'évolutivité du système.

#### Vue d'ensemble de l'architecture

L'architecture globale repose sur quatre composants principaux :

1. Dispositif embarqué (ESP32)
2. Serveur central RMI (cœur du système)
3. Passerelle REST (Gateway – Spring Boot)
4. Clients (Interface Web et Application Mobile)

Ces composants communiquent entre eux via des mécanismes de communication distribuée adaptés à leurs rôles respectifs.

#### Types de communication utilisés

Le système utilise deux types principaux de communication distribuée :

- **HTTP / REST :**  
Utilisé pour :
  - l'envoi des données des capteurs depuis l'ESP32 vers la passerelle,
  - l'accès aux données et aux commandes depuis les interfaces web et mobile.
- **Java RMI (Remote Method Invocation)**  
Utilisé entre :
  - la passerelle REST (client RMI),
  - et le serveur central RMI (serveur distant).

Le serveur RMI est l'unique composant autorisé à contenir la logique métier et l'accès à la base de données. La passerelle agit uniquement comme un traducteur entre REST et RMI.

#### Rôle de chaque composant

##### Serveur RMI (Serveur central)

Le serveur RMI constitue le cœur décisionnel du système. Il est responsable de:

- la réception des mesures des capteurs (via appels RMI)
- la prise de décision concernant l'activation ou non de la pompe (mode automatique),
- la gestion des modes de fonctionnement (AUTO / MANUAL),
- l'enregistrement des données dans la base de données SQL Server,
- la gestion et la diffusion des alertes,
- la fourniture des données de statut, d'historique et de synthèse.

Toutes les décisions critiques sont centralisées à ce niveau, garantissant la cohérence du système.

### **Passerelle REST (Gateway - Spring Boot)**

La passerelle joue un rôle **d'intermédiaire** entre les clients et le serveur RMI. Elle ne contient aucune logique métier.

Ses responsabilités sont limitées à :

- exposer des endpoints REST accessibles aux clients,
- valider la forme des requêtes entrantes,
- traduire les appels REST en appels RMI,
- relayer les réponses du serveur RMI vers les clients,
- diffuser les alertes vers l'interface web via **Server-Sent Events (SSE)**.

Cette séparation permet d'éviter l'accès direct au serveur RMI depuis l'extérieur.

### **Clients (Interface Web et Application Mobile)**

Les clients représentent la couche de présentation du système. Ils permettent :

- la visualisation en temps réel des données des capteurs,
- la consultation de l'historique des mesures,
- l'affichage de l'état des dispositifs (**en ligne / hors ligne**),
- le changement de mode de fonctionnement (**AUTO / MANUAL**),
- le contrôle manuel de la pompe (si autorisé),
- la réception d'alertes en temps réel.

Les clients n'interagissent jamais directement avec la base de données ni avec le serveur RMI.

### **Base de données (SQL Server)**

La base de données est utilisée pour stocker de manière persistante :

- **les dispositifs enregistrés**
- **l'état courant de chaque dispositif**
- **les lectures des capteurs**
- **les décisions prises par le système**
- **les actions de contrôle**

- les alertes générées.

Elle est exclusivement accessible par le serveur RMI, ce qui garantit l'intégrité des données et le respect de l'architecture distribuée.

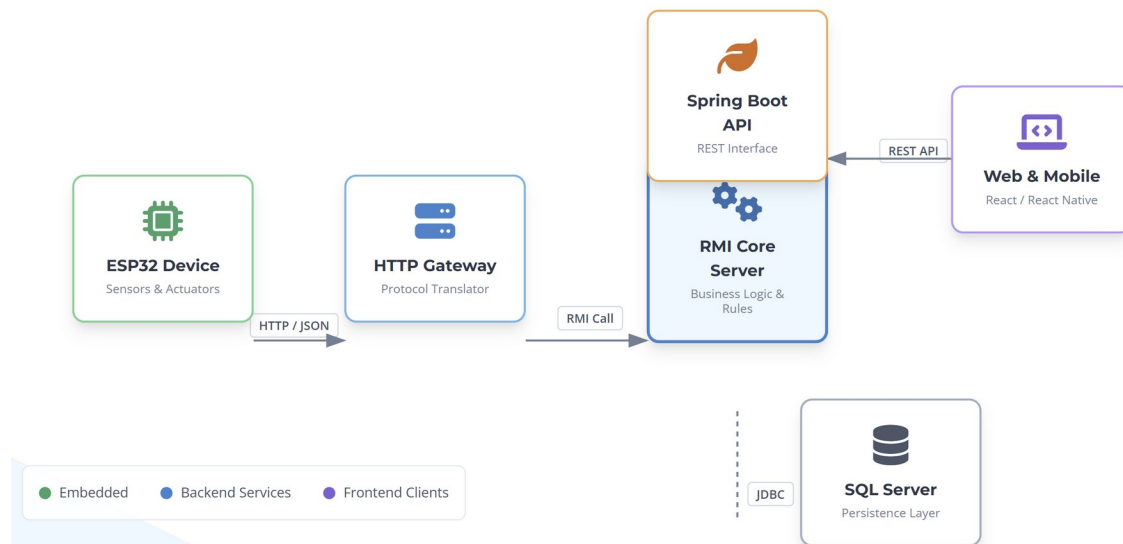
## 2. Déploiement distribué

### Déploiement général du système

Le projet est déployé selon une **architecture distribuée en plusieurs couches**, où chaque composant joue un rôle précis et peut être exécuté de manière indépendante. Cette organisation reflète un système réel de supervision et de contrôle à distance.

Le déploiement repose sur les éléments suivants :

- **Dispositif embarqué (ESP32 + capteurs + pompe)** : déployé physiquement sur le terrain.
- **Serveur RMI (Java)** : serveur central exécutant la logique métier et l'accès aux données.
- **Passerelle REST (Spring Boot)** : exposant des services HTTP pour les interfaces clientes.
- **Base de données SQL Server** : stockant l'état, les mesures et les événements.
- **Interfaces clientes Web et Mobile** : utilisées pour la visualisation et le contrôle.



Dans l'environnement de développement et de test, ces composants sont exécutés sur une même machine. Cependant, leur séparation logique permet un déploiement ultérieur sur des machines distinctes sans modification majeure de l'architecture.

### Topologie réseau et circulation des données

La circulation des données suit une chaîne bien définie :

#### 1. ESP32 → Passerelle REST (HTTP)

Le dispositif embarqué envoie périodiquement des mesures vers la passerelle REST via une requête HTTP POST.

## 2. Passerelle REST → Serveur RMI (RMI)

La passerelle agit comme client RMI et transmet les données reçues au serveur central.

## 3. Serveur RMI → Base de données

Le serveur RMI applique la logique métier, enregistre les données et prend les décisions.

## 4. Serveur RMI → Passerelle → ESP32

La décision concernant la pompe (marche/arrêt) est renvoyée à l'ESP32 via la passerelle.

## 5. Interfaces Web/Mobile → Passerelle REST

Les interfaces clientes consultent l'état du système et envoient les commandes uniquement via la passerelle.

Cette topologie garantit que:

- le serveur RMI reste le point central de décision
- aucun client n'accède directement à la base de données
- la communication est contrôlée et sécurisée.

## Gestion des appels distants et des connexions

Deux types de communication sont utilisés :

- **HTTP/REST** pour :
  - la communication avec l'ESP32
  - les interactions avec les interfaces Web et Mobile.
- **RMI** pour :
  - les appels distants entre la passerelle et le serveur central.

La passerelle implémente un mécanisme de **reconnexion automatique** vers le serveur RMI en cas d'erreur, ce qui améliore la robustesse du système face aux pannes réseau.

Les alertes sont transmises en temps réel vers l'interface web via **Server-Sent Events (SSE)**, permettant une mise à jour continue sans surcharge réseau.

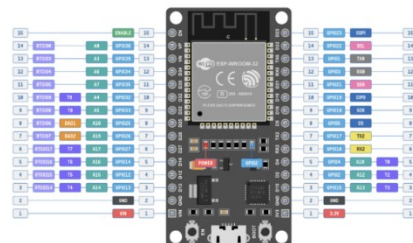
## Dispositif embarqué (ESP32) et capteurs — aperçu dans le contexte distribué

Dans le cadre du système distribué, le dispositif embarqué joue le rôle de nœud périphérique du réseau.

### ✓ ESP32

L'ESP32 est responsable de :

- la collecte des données des capteurs



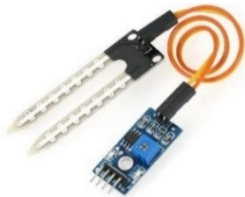


- la construction d'un message JSON respectant un schéma fixe
- l'envoi périodique de ces données vers la passerelle REST
- l'exécution des commandes reçues (activation ou désactivation de la pompe).

Il ne contient aucune logique décisionnelle : toutes les décisions sont prises côté serveur RMI.

### ✓ Capteurs et actionneur (pompe)

Le dispositif embarqué est connecté aux éléments suivants :



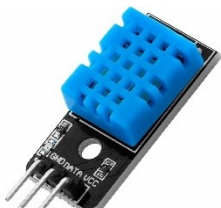
- **Capteur d'humidité du sol** : mesure l'état du sol pour déterminer besoins en irrigation.



- **Capteur de niveau d'eau** : indique la quantité d'eau disponible dans le réservoir.



- **Capteur de pluie** : informe le système de la présence de pluie.



- **Capteur de température et d'humidité ambiante** : fournit des informations environnementales.



- **Pompe (actionneur)** : contrôlée à distance par le serveur central via une commande binaire (ON/OFF).

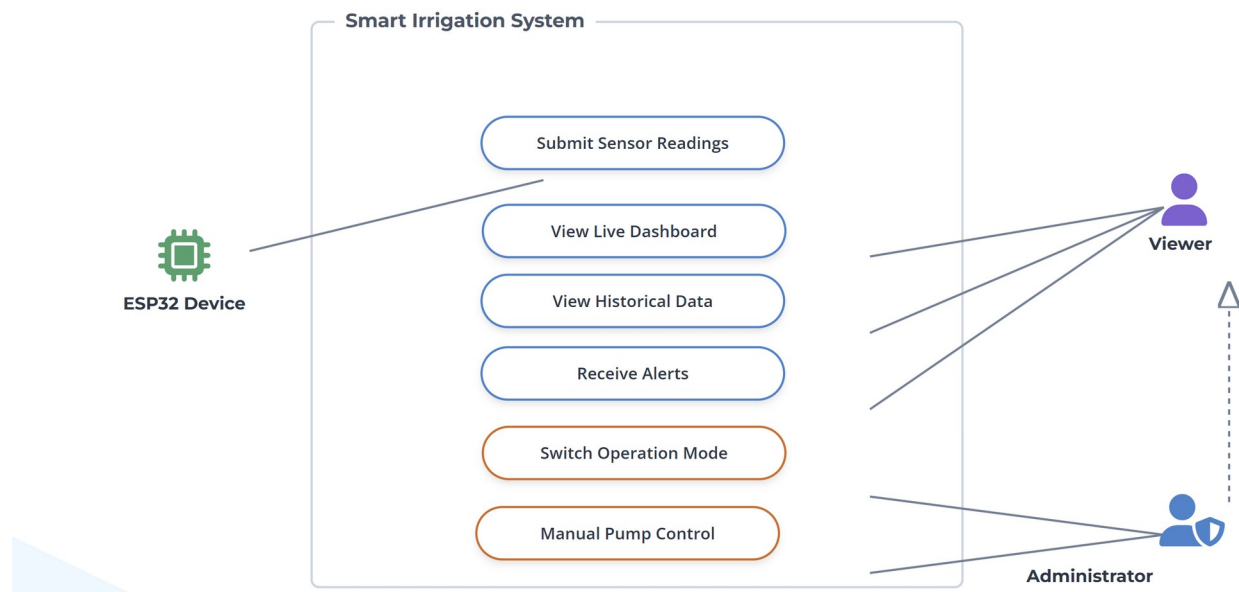
Les capteurs produisent uniquement des données, tandis que la pompe agit comme **actionneur final**, exécutant les décisions prises par le système central.

## Chapitre 3 Diagrammes UML

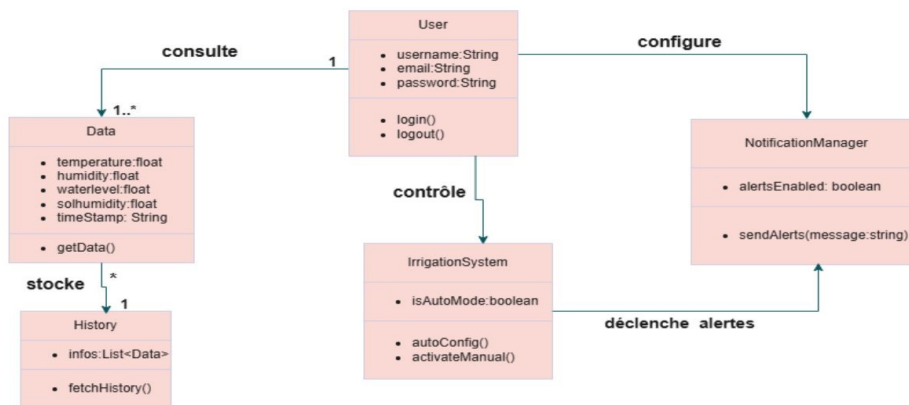
### 1. Introduction:

Dans cette partie, les diagrammes UML sont utilisés afin de modéliser et de mieux comprendre l'architecture et le fonctionnement global du système développé. Ils permettent de représenter de manière structurée les interactions entre le dispositif embarqué ESP32, les capteurs, les serveurs intermédiaires et les applications utilisateur. Les différents diagrammes présentés décrivent aussi bien les aspects fonctionnels que structurels du système. Cette modélisation facilite l'analyse, la maintenance et l'évolution de la solution. Elle offre également une vision claire des flux de données et des mécanismes de contrôle mis en place.

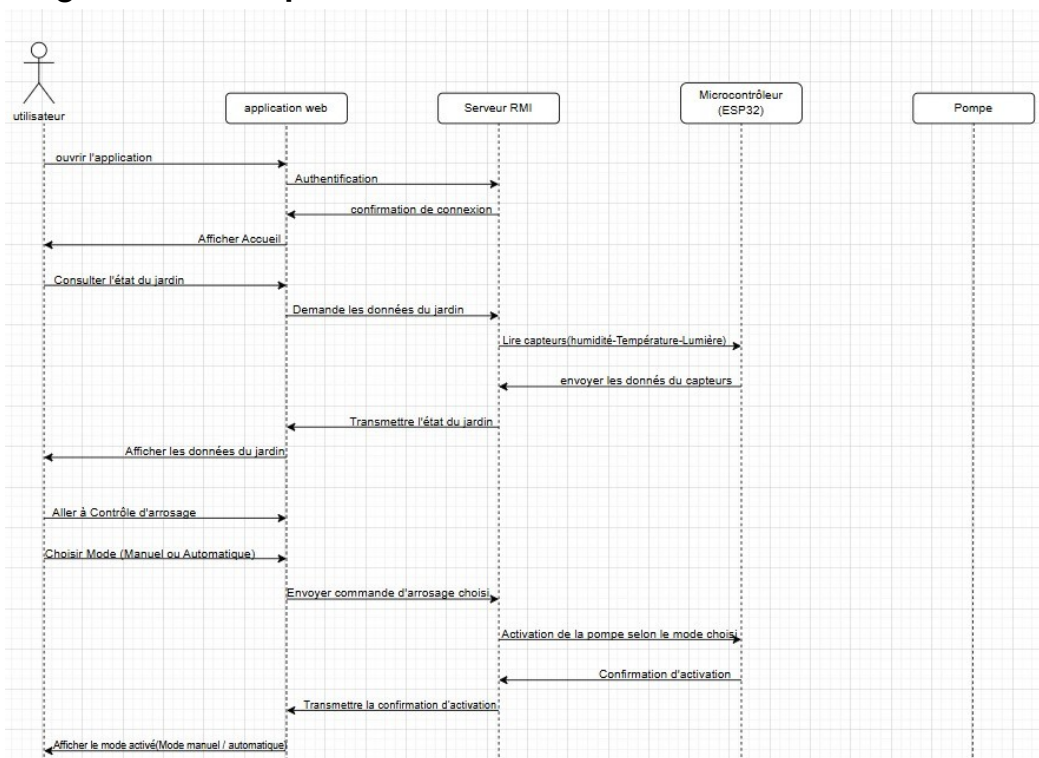
### 2. Diagramme de cas d'utilisation

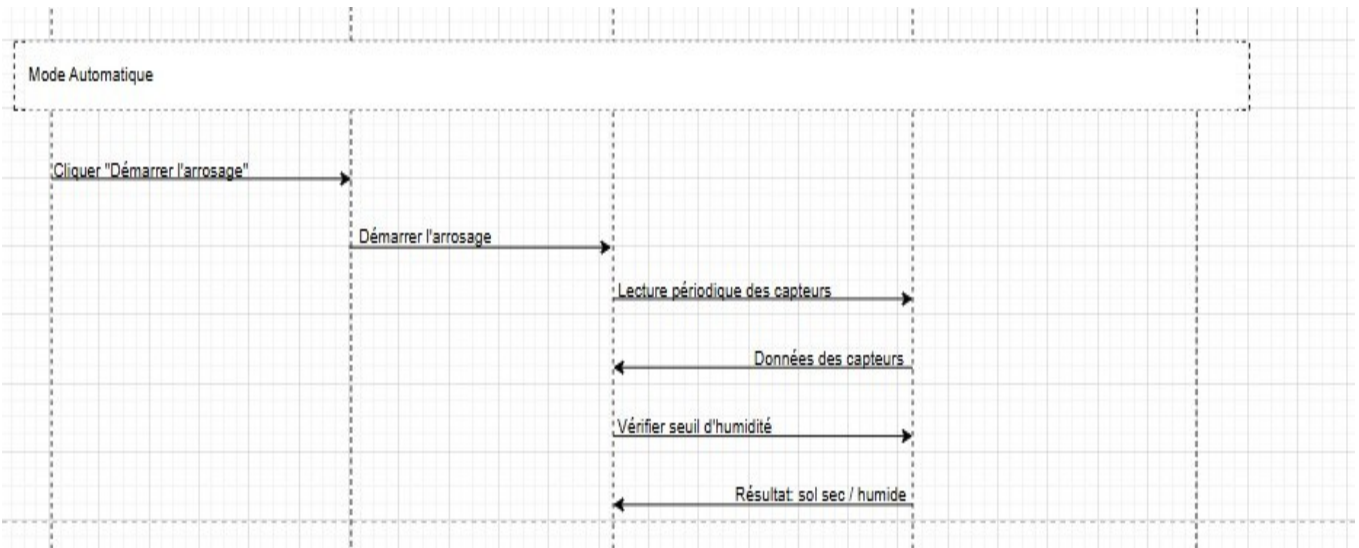
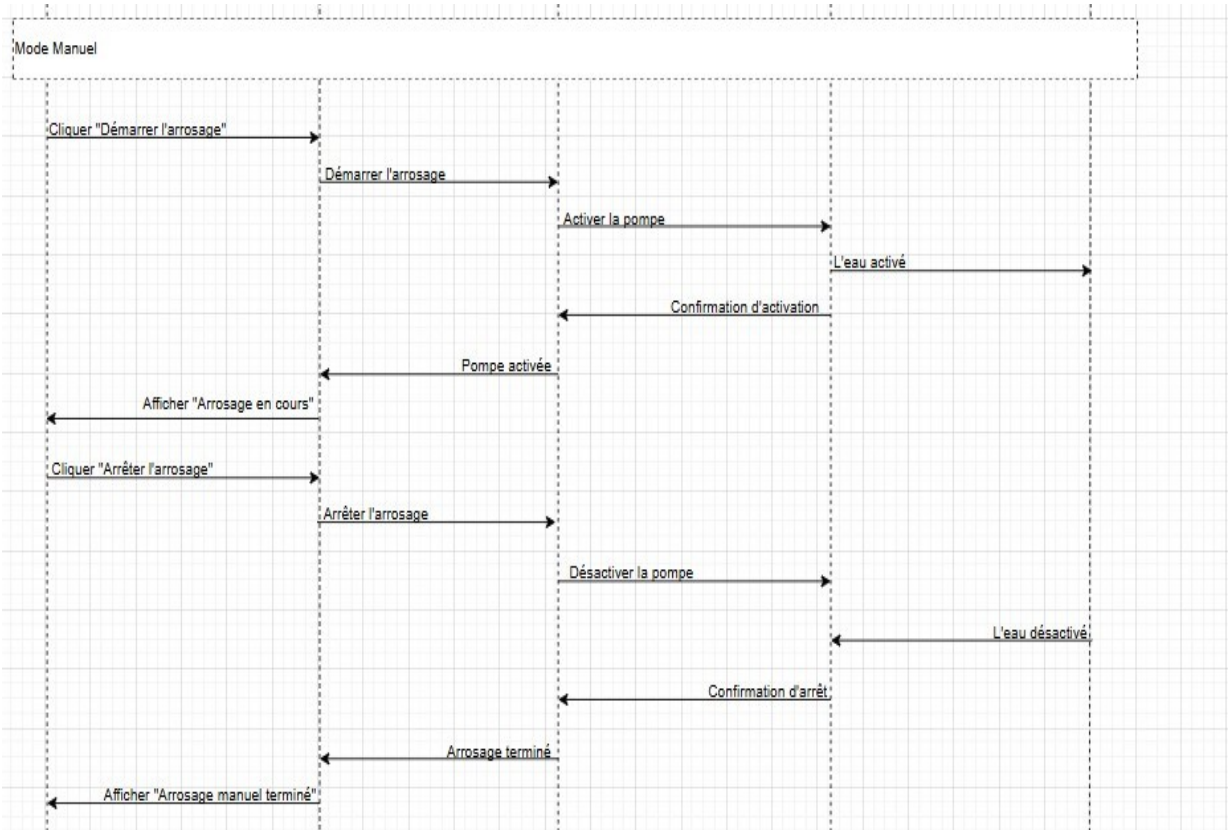


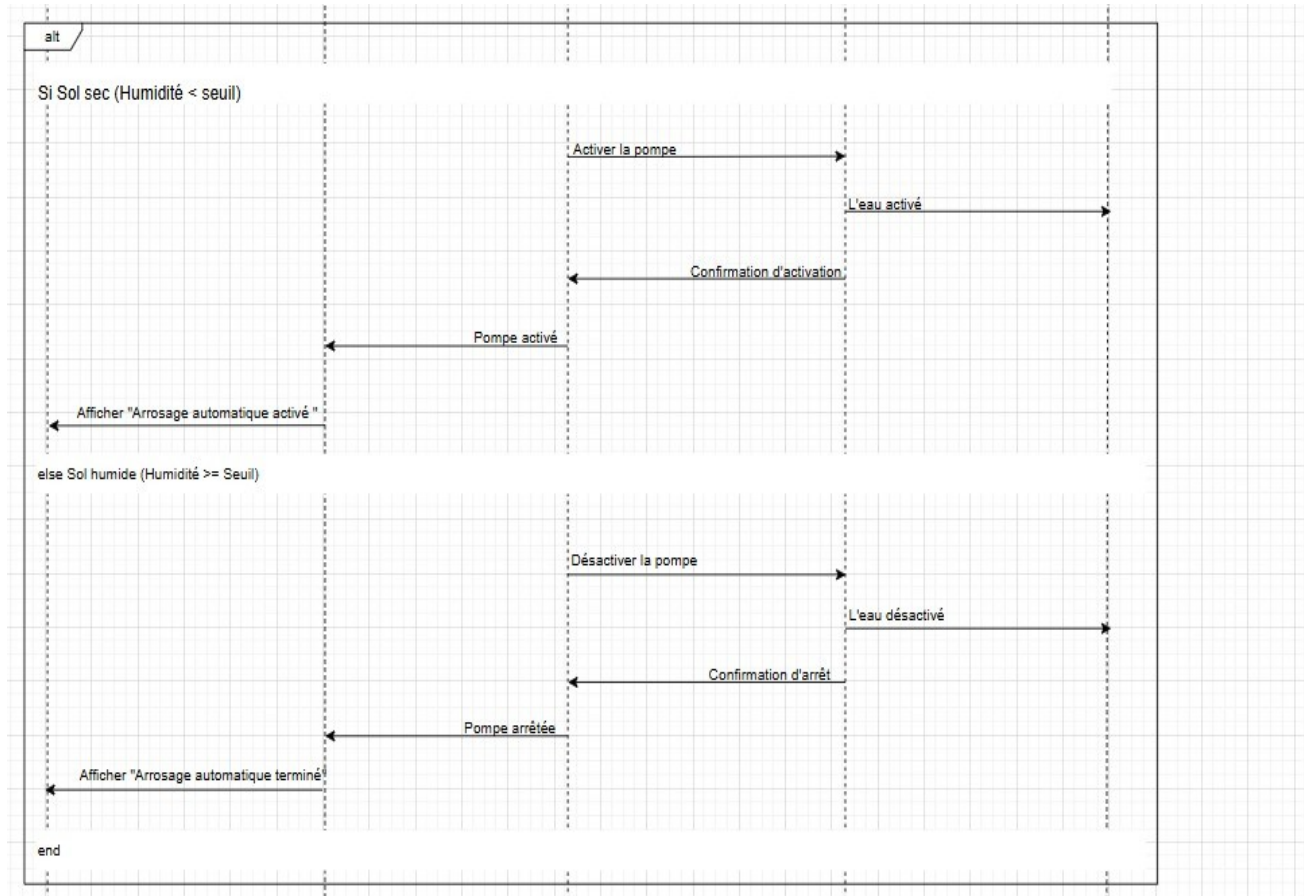
### 3. Diagramme de classes



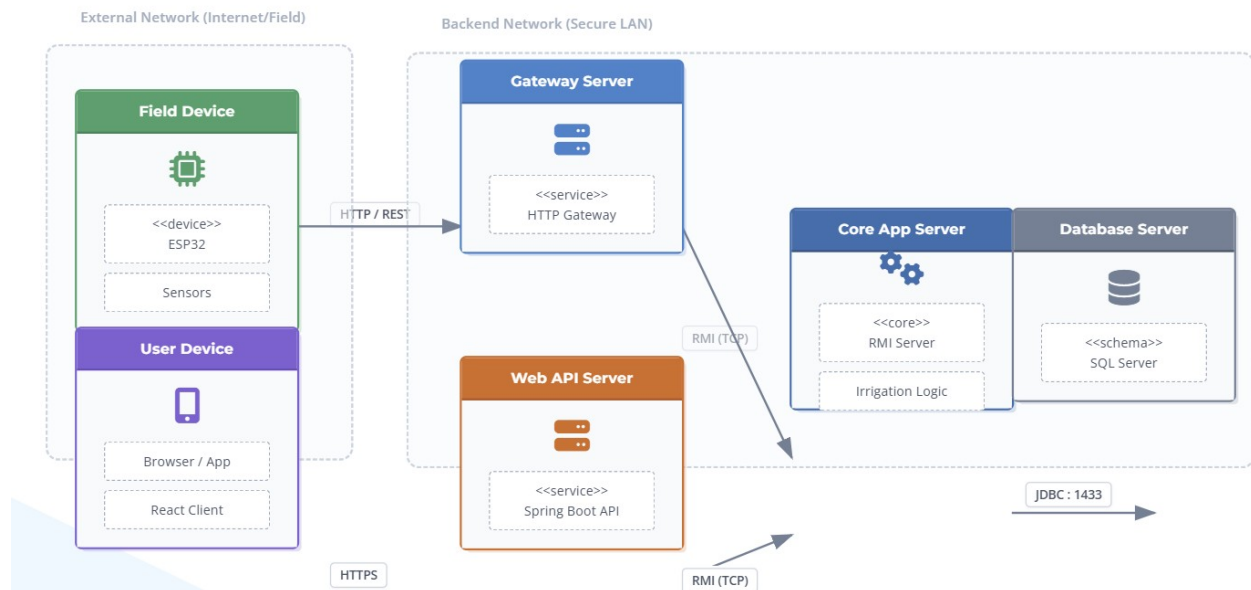
#### 4. Diagramme de séquence







## 5. Diagramme de déploiement



## 6. Conclusion

À travers l'ensemble des diagrammes UML réalisés (diagramme de cas d'utilisation, diagramme de classes, diagramme de séquence et diagramme de déploiement), nous avons pu modéliser de manière claire et cohérente l'architecture et le fonctionnement du système d'irrigation intelligent.

Ces diagrammes permettent de mieux comprendre les interactions entre l'utilisateur, le dispositif embarqué (ESP32), les capteurs, le système de contrôle et les services de notification. Ils constituent une base solide pour l'implémentation technique du projet et facilitent la maintenance ainsi que l'évolution future du système.

Cette modélisation UML assure ainsi une meilleure organisation du développement et une vision globale du système avant son déploiement réel.



## Chapitre 4 Réalisation et présentation de l'application

### 1. Présentation générale de l'application

L'application développée dans ce projet constitue une plateforme de supervision et de contrôle d'un système d'irrigation intelligent basé sur un dispositif embarqué ESP32. Elle permet à l'utilisateur de surveiller en temps réel les paramètres environnementaux tels que la température, l'humidité, l'humidité du sol et l'état du système d'arrosage.

L'application offre une interface intuitive accessible via le web, facilitant l'interaction entre l'utilisateur et le système distribué. Grâce à cette interface, l'utilisateur peut consulter les données collectées par les capteurs, visualiser l'historique des mesures et configurer le mode de fonctionnement du système (manuel ou automatique).

La communication entre les différents composants du système est assurée par des services distribués (REST/RMI), garantissant une transmission fiable et efficace des données entre l'ESP32, le serveur backend et l'interface utilisateur. Le système intègre également un mécanisme de notification permettant d'alerter l'utilisateur en cas de conditions critiques. Cette application vise ainsi à améliorer la gestion de l'irrigation en offrant une solution intelligente, flexible et scalable, adaptée aux besoins de l'agriculture connectée.

### 2. Fonctionnalités principales

L'application développée offre un ensemble de fonctionnalités permettant la gestion complète et intelligente du système d'irrigation, depuis la collecte des données jusqu'au contrôle des actionneurs.

#### • Collecte et transmission des données des capteurs

Le dispositif embarqué ESP32 est chargé de collecter périodiquement les données issues des capteurs environnementaux, notamment l'humidité du sol, la température, l'humidité de l'air ainsi que l'état de la pluie. Ces données sont transmises au serveur via une interface HTTP REST.

#### • Supervision en temps réel

L'application permet de visualiser en temps réel les dernières mesures reçues pour chaque dispositif connecté. L'utilisateur peut consulter l'état courant du système d'irrigation, y compris l'état de la pompe et le mode de fonctionnement actif.

#### • Gestion des modes de fonctionnement

Le système supporte deux modes de fonctionnement :

- ✓ **Mode automatique** : la décision d'activation de la pompe est prise automatiquement en fonction des seuils définis et de la logique métier implémentée côté serveur.

- ✓ **Mode manuel** : l'utilisateur peut forcer l'activation ou la désactivation de la pompe via l'interface web ou mobile.

- **Commande à distance de la pompe**

Grâce aux services distribués (RMI et REST), l'utilisateur peut contrôler la pompe d'irrigation à distance. Les commandes sont transmises de manière sécurisée et exécutées en temps réel par le dispositif embarqué.

- **Historique des données**

L'application permet la consultation de l'historique des mesures collectées, facilitant l'analyse du comportement du sol et des conditions climatiques sur une période donnée.

- **Gestion des alertes**

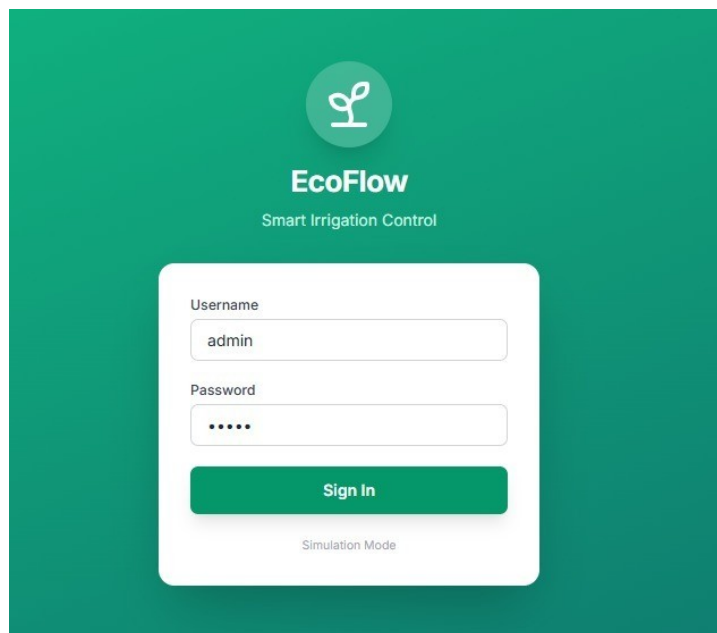
Un système d'alertes est intégré afin de notifier l'utilisateur en cas de conditions critiques, telles qu'un niveau d'humidité du sol trop faible ou un dysfonctionnement du système.

- **Architecture distribuée et scalable**

L'architecture du système permet l'ajout de nouveaux dispositifs ESP32 sans modification majeure de l'application, assurant ainsi une bonne extensibilité et une adaptation facile à un environnement distribué.

### 3. Présentation des interfaces

- ✓ **Écran de connexion**

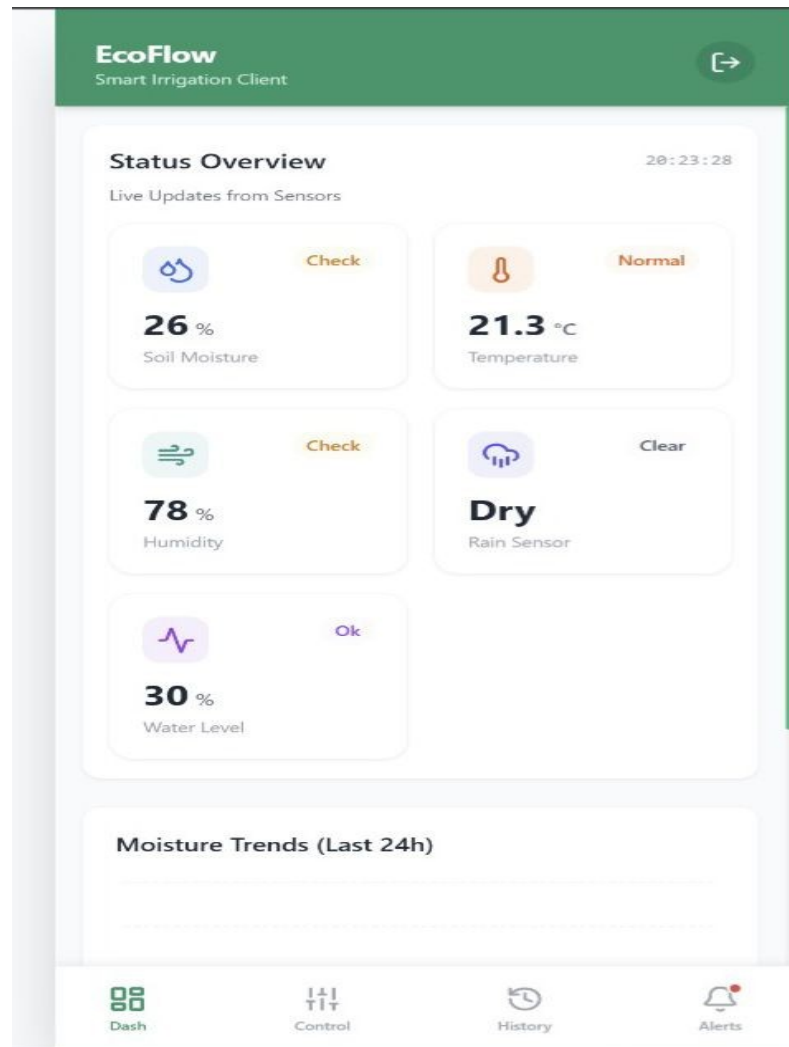




Cet écran représente la page d'authentification de l'application *EcoFlow - Smart Irrigation Control*.

L'utilisateur doit saisir un nom d'utilisateur et un mot de passe afin d'accéder au système. Cette étape permet de sécuriser l'accès à l'application et de limiter les actions de contrôle (changement de mode, activation de la pompe) aux utilisateurs autorisés.

### ✓ Tableau de bord



Cet écran présente une vue globale de l'état du système d'irrigation en temps réel.

Les données affichées proviennent directement des capteurs connectés à l'ESP32 et sont mises à jour via le backend distribué.

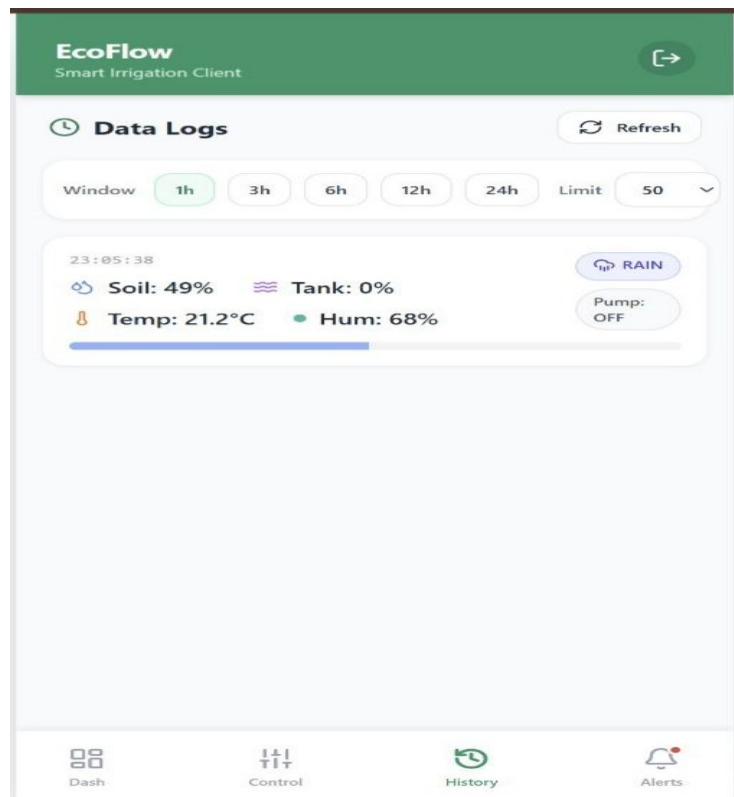
Les informations visibles incluent :

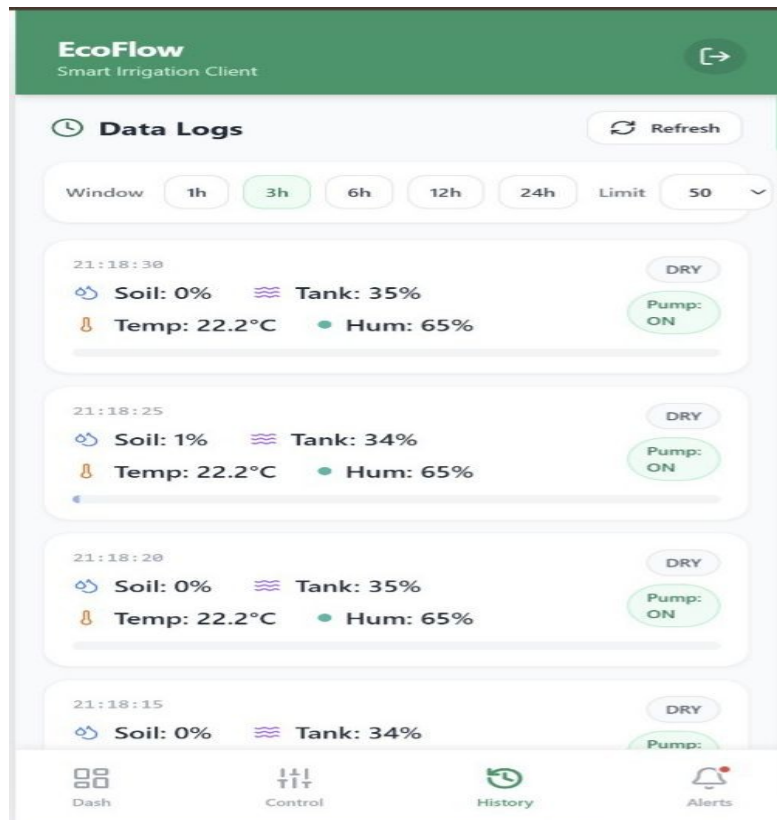
- Le taux d'humidité du sol
- La température ambiante
- Le taux d'humidité de l'air

- L'état du capteur de pluie
- Le niveau d'eau du réservoir

Chaque indicateur est accompagné d'un statut (Normal, Check, Ok, Clear), facilitant la lecture rapide de l'état du système.

### ✓ Historique des données





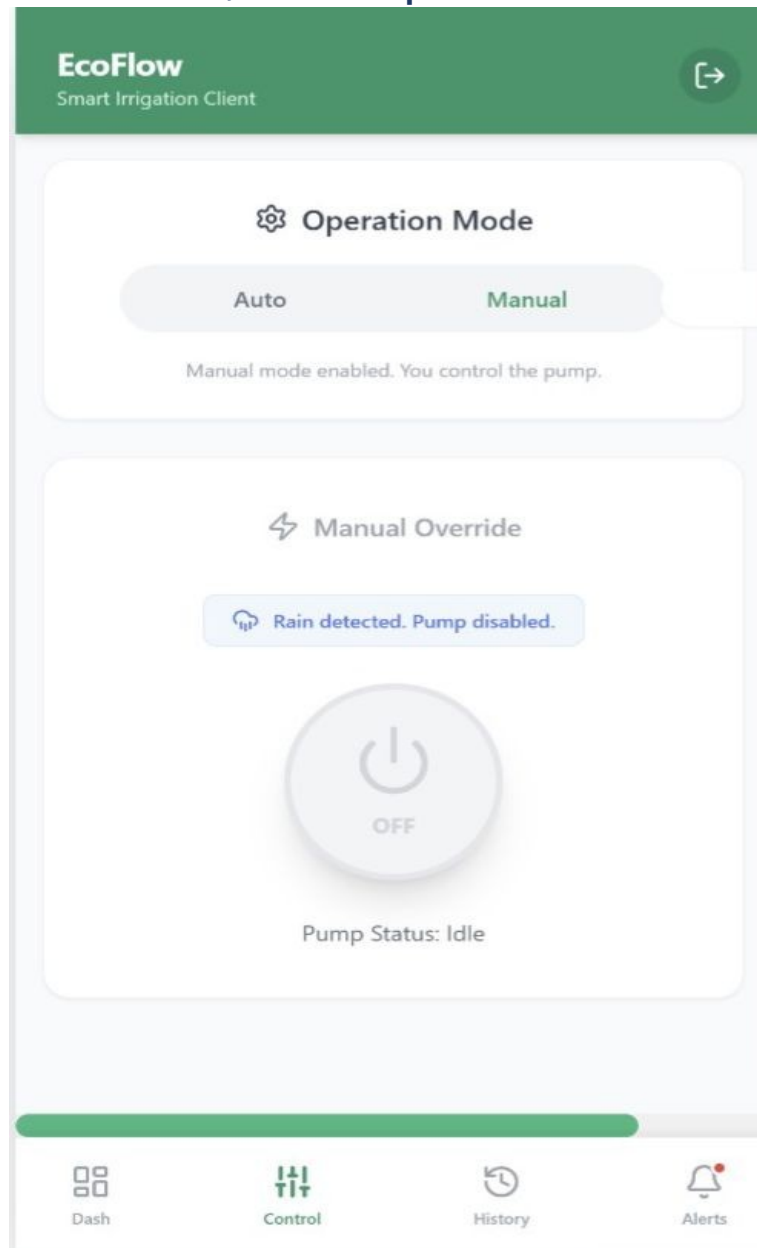
Cet écran permet de consulter l'historique des données collectées par les capteurs. L'utilisateur peut sélectionner une fenêtre temporelle (1h, 3h, 6h, 12h, 24h) afin d'analyser l'évolution des paramètres environnementaux.

Chaque enregistrement affiche :

- Le taux d'humidité du sol
- La température
- Le niveau du réservoir
- L'humidité de l'air
- L'état de la pluie
- L'état de la pompe (ON/OFF)

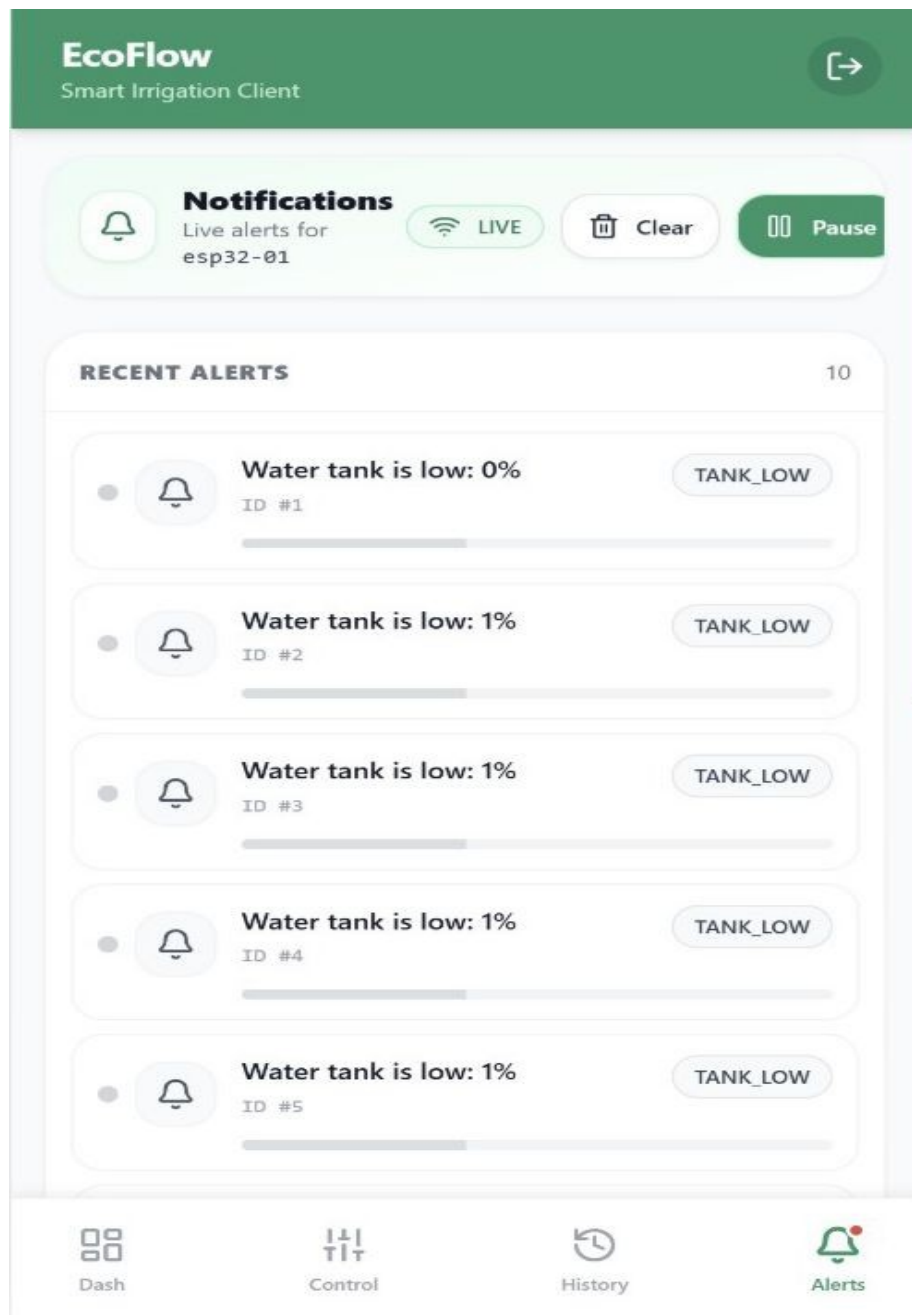
Ces données sont récupérées depuis la base de données via le serveur RMI, garantissant la cohérence et la centralisation des informations.

## ✓ Contrôle manuel / automatique



Cet écran montre l'interface de l'application EcoFlow lorsque le système fonctionne en mode **manuel**. L'utilisateur peut normalement activer ou désactiver la pompe via le bouton central. Cependant, comme une **pluie a été détectée par le capteur**, la sécurité du système bloque automatiquement la commande : un message d'alerte indique que la pompe est désactivée pour éviter un arrosage inutile.

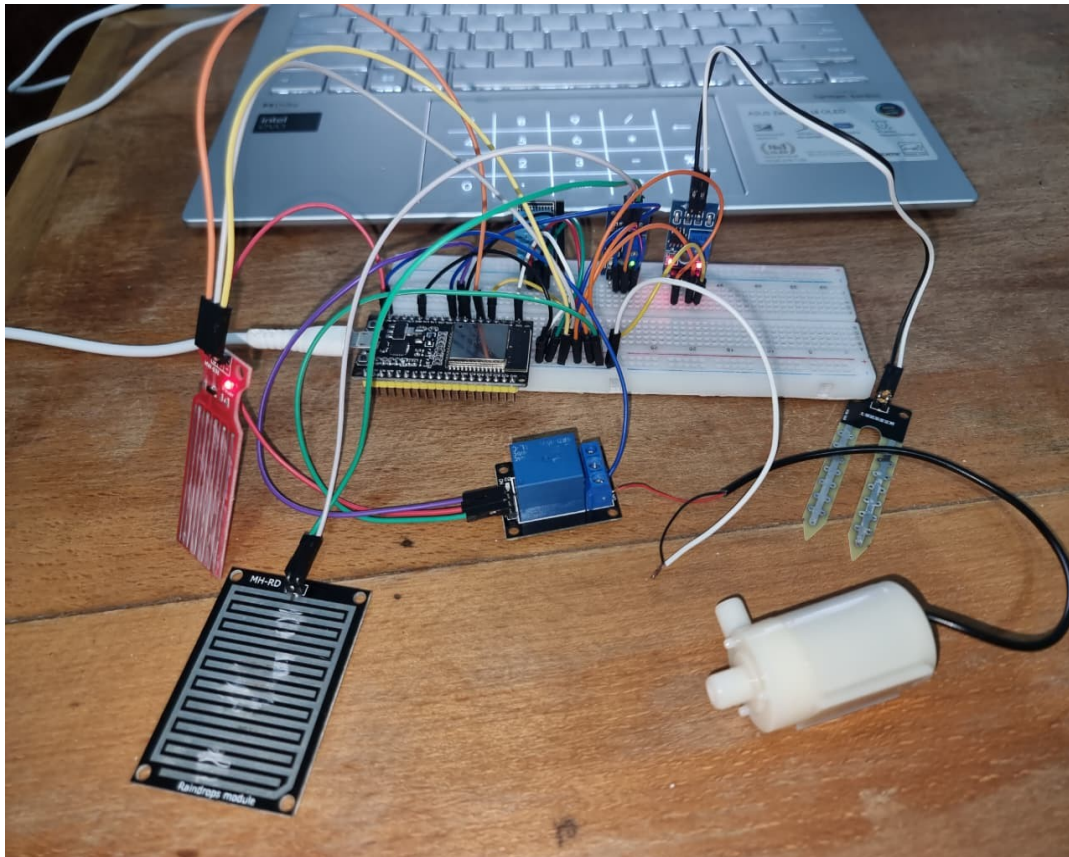
## ✓ Notifications et alertes



Cet écran affiche en temps réel les alertes générées par le système EcoFlow pour le dispositif *esp32-01*. Les notifications indiquent ici que le **niveau d'eau du réservoir est extrêmement**

**bas (entre 0 % et 1 %)**. Chaque alerte est identifiée par un ID unique et classée sous le type **TANK\_LOW**, ce qui permet de tracer l'historique des événements. L'utilisateur peut suivre les alertes en mode **LIVE**, les effacer ou suspendre temporairement leur réception. Ce mécanisme permet de garantir que le système d'irrigation ne fonctionne pas à sec et d'informer rapidement l'utilisateur lorsqu'un remplissage du réservoir est nécessaire.

#### 4. Montage et Déploiement du Système d'Irrigation



## Chapitre 5 Conclusion Générale du Projet

Le projet EcoFlow a permis de concevoir et de réaliser un système d'irrigation intelligent, distribué et entièrement connecté, combinant des technologies embarquées (ESP32), des services distants (RMI / REST) et une interface web moderne. L'objectif principal — automatiser et optimiser l'arrosage tout en garantissant une supervision en temps réel — a été pleinement atteint.

Grâce à une architecture distribuée, le système assure une communication fiable entre les capteurs au niveau du terrain, le serveur d'application et l'interface utilisateur. Les données environnementales (humidité du sol, température, pluie, niveau d'eau) sont collectées en continu, analysées, puis exploitées pour piloter intelligemment la pompe d'irrigation. L'utilisateur conserve en parallèle la possibilité d'intervenir manuellement via l'interface web, tout en bénéficiant d'un mécanisme de sécurité robuste (détection de pluie, réservoir vide, etc.).

Ce projet a permis de mettre en pratique plusieurs concepts essentiels des systèmes distribués : tolérance aux pannes, communication distante, synchronisation des états et persistance des données. Il a également démontré la complémentarité entre électronique embarquée, backend serveur et expérience utilisateur.

Sur le plan fonctionnel, EcoFlow constitue une solution fiable, modulaire et extensible, capable d'évoluer vers un déploiement à grande échelle (agriculture intelligente, serres automatisées, jardins connectés, etc.). Sur le plan pédagogique, le projet a renforcé la compréhension des architectures distribuées et a permis de développer une approche structurée d'analyse, de conception et d'implémentation.

En conclusion, EcoFlow représente bien plus qu'un simple prototype : c'est une plateforme opérationnelle, intelligente et durable, qui ouvre la voie à de futures améliorations telles que l'analyse prédictive, l'optimisation énergétique ou l'intégration cloud avancée. Ce travail prouve qu'une gestion connectée et responsable des ressources en eau est non seulement possible, mais déjà une réalité.