Une image contenant texte, Police, capture d’écran, logo

Description générée automatiquement

|  |
| --- |
| Université de Pau et des Pays de l’Adour – Collège STEE |
| Documentation Application Smart Garden |

 Une image contenant plante, herbe, plein air, serre

Description générée automatiquement

PICIURA Ander

DAVID Thomas

Etudiants en L3-NEC

10/01/2025

Table des matières

[Présentation 3](#_Toc187401502)

[Contexte 3](#_Toc187401503)

[Modélisation 3](#_Toc187401504)

[Architecture du projet 4](#_Toc187401505)

[Technologies 4](#_Toc187401506)

[Capteurs 4](#_Toc187401507)

[Broker1 4](#_Toc187401508)

[App 4](#_Toc187401509)

[BD 4](#_Toc187401510)

[API 4](#_Toc187401511)

[Avancement 5](#_Toc187401512)

[Obstacles 5](#_Toc187401513)

[Finalité 5](#_Toc187401514)

[Glossaire 6](#_Toc187401515)

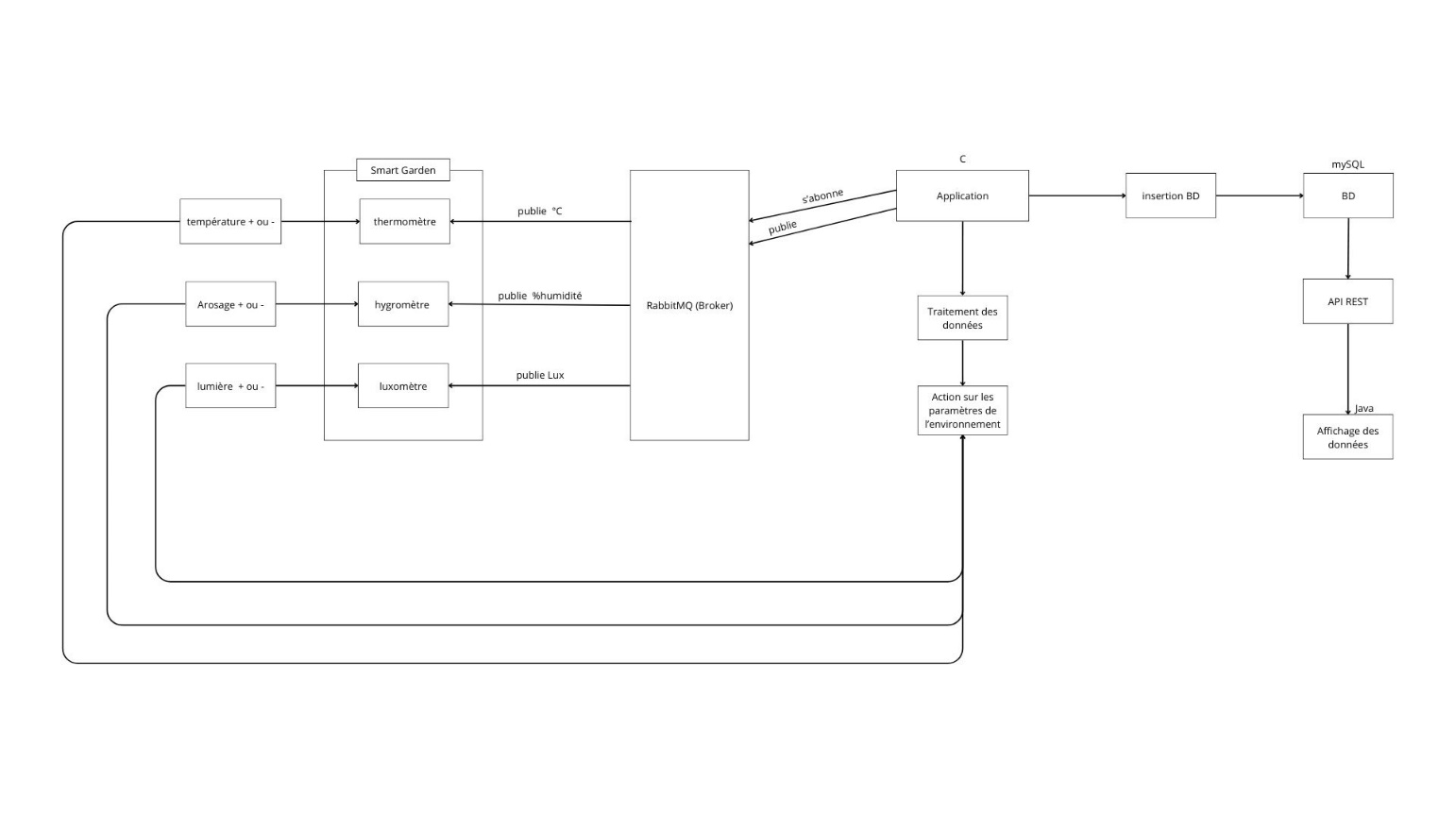
# Présentation

## Contexte

L’idée de ce projet est de développer un système permettant le suivi puis l’automatisation d’une plantation quelconque. En surveillant par exemple la température de l’environnement, l’humidité ou la luminosité.

## Modélisation

Nous avons commencé par la création d’un diagramme afin de prévoir les fonctionnalités de notre système



L’objectif est donc de simuler différentes données à envoyer ( température, humidité et luminosité ), de les publier à un Broker, de les récupérer sur une application afin de les insérer dans une base de données pour pouvoir ensuite les afficher via une API REST. Et l’idéal, si le temps nous le permet, serait de, via l’application, pouvoir intéragir sur l’environnement ( changer la température, humidité et luminosité ).

## Architecture du projet

Une image contenant texte, capture d’écran

Description générée automatiquement

## Technologies

### Capteurs

En l’absence de données à capturer, nous avons simmulé les capteurs à l’aide de programmes en Java.

### Broker1

Pour envoyer et reçevoir les données de notre jardin, nous avons utilisé un broker nommé [RabbitMQ](https://www.rabbitmq.com/) car simple à déployer, à utiliser, scallable et fiable.

Plus précisément nous avons utilisé le modèle orienté souscription2 de RabbitMQ.

### App

Notre Application a aussi été réalisée en Java.

### BD

Nous avons utilisé une base de donnée MySQL et un affichage de celle-ci est possible via une interface phpMyAdmin :

* identifiant : root
* Mot de passe : rootpassword
* URL : <http://localhost:8080>

### API

Une API a aussi été déployée, accessible par l’url : <http://localhost:8085/api/temperatures>

# Avancement

## Obstacles

Prise en main des technologies assez compliquée :

* Prise en main de RabbitMQ
* Utilisation de la BD sur Docker depuis un programme Java
* Mise en place d’une API REST
* Manque de temps pour l’implémentation de certaines fonctionnalités

## Finalité

Toutes les fonctionnalités ne sont pas encore implémentées, mais en à peine quelques petites heures supplémentaires cela devrait être un jeu d’enfant.

Les fonctionnalités manquantes sont : la publication de données autre que la températue : l’humidité (en %) et la luminosité (en Lux), ainsi que leur réception par l’application et affichage via l’API.

Le traitement des données ainsi que l’action sur l’environnement ne sont pas encore implémentés non plus.

Cependant, l’envoi de températures, leur réception, insertion dans la BD, et l’affichage via une API REST sont fonctionnels.

Une image contenant texte, diagramme, croquis, Plan

Description générée automatiquement

# Glossaire

Broker : Le Broker permet la communication entre deux éléments informatiques, ici les capteurs de données et l’application. Il peut reçevoir, stocker et envoyer des messages aux personnes désignées, ici en suivant le modèle orienté sousciption.

Modèle de broker orienté sousctiption : ce modèle permet à des « clients » de s’abonner au Broker afin de, quand ce dernier reçoit un message, il peut le transmettre aux clients  « abonnés » à ce filtre de données