

Rapport P.F.E :

Elaboration d'un système IOT pour la gestion d'un warehouse

Préparé par:

Abdelkodousse Ybnelhaje
Ali Ouarrirh
Rida Lamiini
Ismail Tazzit

Encadré par:

Mr. Hicham Belhedaoui
M. Nadia Afifi

Table des matières:

Table des matières.....
Remerciements
Chapitre 1 : Etude théorique de l'IOT.....
Chapitre 2 : Spécification des besoins.....
introduction.....
1.Objectif.....
2.Exigences Fonctionnelles.....
3.Description fonctionnelle :
conclusion.....
Chapitre 3 : Gestion de projet.....
introduction.....
1.Choix de la méthodologie.....
2.Rôles et responsabilité.....
3.Sprints
conclusion.....
Chapitre 4 : Analyse et conception (sprint 0).....
introduction.....
1.Méthodologie d'analyse.....
2.Architecture système.....
3.Schéma.....
4.Diagramme de Cas d'utilisation
Conclusion.....
Chapitre 5 : Realisation.....
1.Initialisation de l'environnement (sprint 1).....
Introduction
1.ESP32.....
2.ReactJS.....
3.tailwindCss.....
4.Spring Boot
Conclusion.....
2.Contrôle d'accès et Authentification (sprint 2).....
Introduction
1.Contrôle d'accès.....
- RFID.....
- Installation.....
- Diagramme de séquence.....

2.Authentification.....
- Introduction.....
- Développement de l'api.....
- Jwt.....
- Conclusion.....
3.Integration Azure cloud et Récupération de données(sprint 3).....	
Introduction
1.Installation du MQTT Broker.....
2.Réseaux des capteurs.....
3.Publication des données.....
4.Abonnement au topic.....
5.Websocket.....
6.Diagramme de séquence.....
Conclusion
4.Sécurisation d'accès au chambre privée (sprint 4).....	
Introduction
1.Interface.....
2.Diagramme d'activité.....
3.Abonnement au sujet.....
Conclusion
5.Implementation du DevOps (sprint 5).....	
Introduction
1.Git.....
2.Docker.....
3.Jenkins.....
4.Creation des Instances.....
Conclusion
Chapitre 6: Conclusion.....	

Remerciement :

Nous exprimons notre plus profonde gratitude envers Mme Nadia Afifi et Monsieur Hicham Belhadaoui pour leur précieux accompagnement tout au long de notre Projet de Fin d'Études. Leur disponibilité constante, leurs critiques constructives et leurs conseils précieux ont été d'une aide inestimable. Leur générosité en nous fournissant tout le matériel nécessaire mérite une mention spéciale.

Nous tenons également à exprimer notre sincère reconnaissance envers notre encadrant, Monsieur Hicham Belhadaoui, pour avoir facilité le succès de notre stage en nous accordant l'accès à la salle de recherche.

Un remerciement particulier est adressé à nos familles pour leur soutien sans faille et leurs précieux conseils.

Enfin, nous souhaitons remercier chaleureusement toutes les personnes qui ont contribué, de près ou de loin, à l'élaboration de ce travail. Leur aide a été indispensable dans la réalisation de ce projet, et nous leur exprimons notre reconnaissance la plus profonde.



CHAPITRE 1:

Etude théorique de l'IoT

Introduction général

L'avancement technologique et l'évolution du mode de vie contemporain ouvrent la voie à des logements mieux adaptés et plus intelligents, où l'amélioration du confort et de la sécurité domestique est devenue une priorité. L'intégration de l'informatique a donné naissance aux maisons intelligentes, offrant un environnement de vie plus confortable et sécurisé. Ces habitations permettent aux occupants de contrôler divers aspects via un smartphone, comme l'éclairage, la température et d'autres appareils électroniques, améliorant ainsi considérablement le quotidien en termes de confort et de commodité.

Par ailleurs, un autre aspect crucial est la protection des résidents contre les dangers potentiels. Les systèmes intégrés aux maisons intelligentes peuvent anticiper les situations dangereuses ou réagir rapidement à des événements mettant en péril la sécurité des occupants.

C'est dans ce contexte, et dans le cadre de notre projet de fin d'études, que nous proposons de concevoir un modèle réduit d'une "SMART WAREHOUSE". Ce prototype vise à mettre en œuvre diverses fonctions de domotique telles que la gestion de l'éclairage, la surveillance de la température intérieure, ainsi que la détection de fuites de gaz et de fumée. Ces fonctionnalités seront automatisées grâce à des cartes électroniques exécutant des programmes informatiques spécifiques, ainsi qu'à une application mobile dédiée.

I. Introduction :

L'Internet des objets (IoT) est l'interconnexion d'appareils physiques via des logiciels, des capteurs et une connectivité pour échanger des données. Cette technologie facilite la collecte et le partage de données entre une variété d'appareils, ouvrant la voie à des systèmes plus efficaces et automatisés. Dans un avenir proche, l'IoT révolutionnera de nombreux aspects de la vie quotidienne, y compris la médecine, l'énergie, l'agriculture, les villes intelligentes et les maisons connectées. Il s'agit d'un réseau d'objets interconnectés, tels que des appareils informatiques, des machines et des êtres vivants, tous avec des identifiants uniques, facilitant le transfert de données sur un réseau et permettant des interactions entre humains, entre humains et ordinateurs, et entre les objets eux-mêmes.



II. Histoire de l'IoT :

Ici, vous apprendrez comment l'IOT est impliqué et également, à partir de l'explication de chacun, vous découvrirez comment l'IOT joue un rôle dans ces innovations :

1982 : Premier aperçu de l'IoT avec un distributeur automatique connecté à Internet pour surveiller son inventaire.

1990 : Premier grille-pain connecté à Internet, préfigurant la commodité des appareils domestiques intelligents.

1999 : Kevin Ashton invente le terme "Internet des objets", décrivant le réseau interconnecté d'appareils communiquant et partageant des données.

2000 : Le LG Smart Fridge permet de gérer le contenu du réfrigérateur à distance, illustrant le potentiel de l'IoT dans la vie quotidienne.

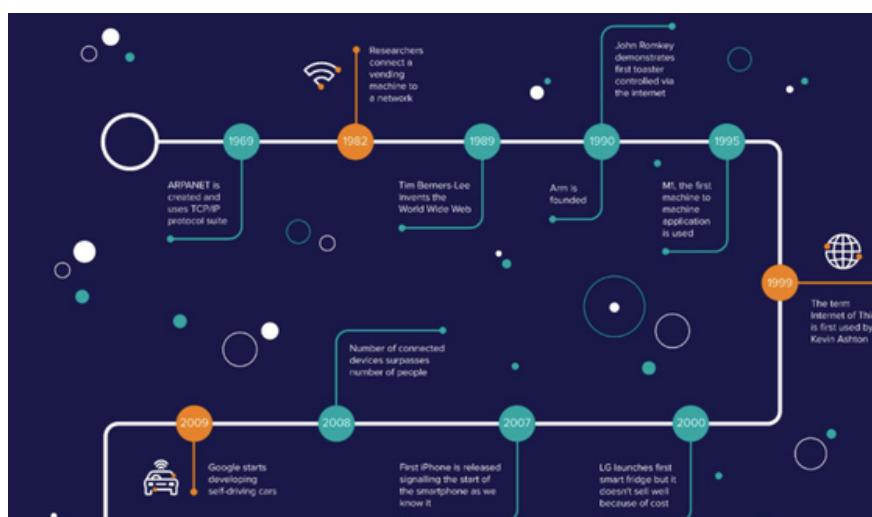
2004 : Introduction de l'IoT dans les technologies portables avec des montres intelligentes offrant un suivi de la condition physique et des notifications en déplacement.

2007 : L'iPhone d'Apple intègre des capacités IoT avec des applications connectant les utilisateurs à divers services et appareils.

2009 : L'IoT entre dans l'industrie automobile avec des capteurs améliorant les véhicules pour les diagnostics en temps réel et les tests à distance.

2011 : Les Smart TV introduisent l'IoT dans les salons, permettant la connectivité Internet pour le streaming et l'utilisation d'applications.

2013 : Google Lens présente le potentiel de l'IoT en matière de reconnaissance d'images sur smartphones.

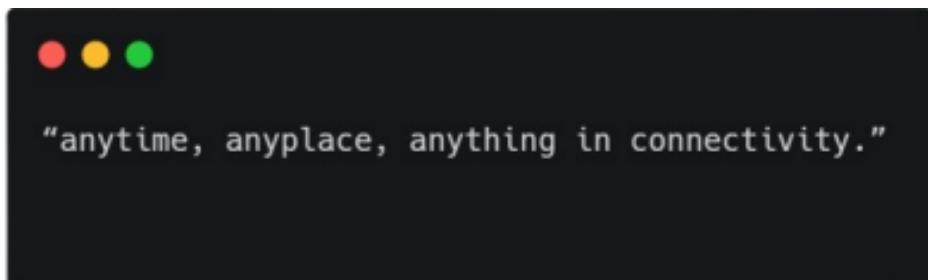


III. Méthodes de développement de l'IoT :

Il existe deux manières de créer l'IoT :

- **Formez** une œuvre Internet distincte comprenant uniquement des objets physiques.
- **Rendre** Internet toujours plus étendu, mais cela nécessite des technologies de base telles qu'un cloud computing rigoureux et un stockage rapide de Big Data (coûteux).

Dans un avenir proche, l'IoT deviendra plus vaste et plus complexe en termes de portée. Cela changera le monde en termes de :



1_IoT_Enablers:

RFID : utilise des ondes radio pour suivre électroniquement les étiquettes attachées à chaque objet physique.

Capteurs : appareils capables de détecter les changements dans un environnement (ex : détecteurs de mouvement).

Nanotechnologie : comme son nom l'indique, il s'agit de minuscules appareils dont les dimensions sont généralement inférieures à une centaine de nanomètres.

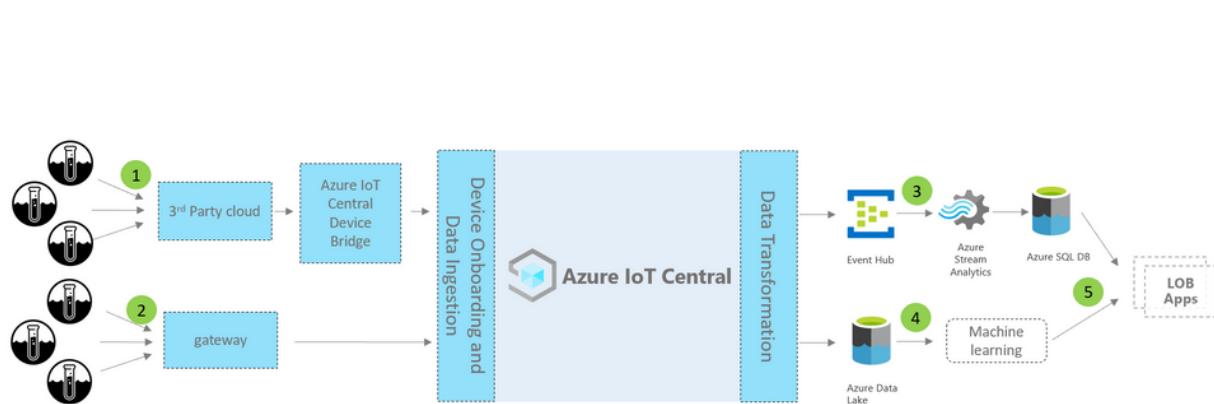
Réseaux intelligents : (ex : topologie maillée)

2 Caractéristiques de l'IoT

- Massivement évolutif et efficace
- L'adressage basé sur IP ne sera plus adapté à l'avenir.
- Une abondance d'objets physiques n'utilisent pas l'IP, ce qui rend l'IoT possible.
- Les appareils consomment généralement moins d'énergie. Lorsqu'ils ne sont pas utilisés, ils doivent être automatiquement programmés pour dormir.
- Un appareil qui est actuellement connecté à un autre appareil peut ne pas être connecté à un autre instant.
- Connectivité intermittente – Les appareils IoT ne sont pas toujours connectés. Afin d'économiser la bande passante et la consommation de la batterie, les appareils seront éteints périodiquement lorsqu'ils ne sont pas utilisés. Sinon, les connexions pourraient devenir peu fiables et donc inefficaces.

3 Qualité souhaitée de toute application IoT

L'Internet des objets (IoT) repose sur plusieurs exigences et caractéristiques clés :



- 1. Interconnectivité** : Tous les appareils IoT doivent pouvoir se connecter et communiquer entre eux, quel que soit le réseau auquel ils sont connectés.
- 2. Hétérogénéité** : Les appareils IoT peuvent présenter diverses configurations matérielles et logicielles, mais ils doivent pouvoir interagir malgré ces différences.
- 3. Dynamisme** : Les appareils IoT doivent pouvoir s'adapter dynamiquement à un environnement changeant.
- 4. Technologie auto-adaptative et auto-configurable** : Les appareils IoT doivent être flexibles pour fonctionner dans différentes conditions, comme les variations météorologiques ou les changements de luminosité.
- 5. Intelligence** : L'analyse des données est essentielle dans l'IoT pour extraire des connaissances et fournir des insights actionnables.
- 6. Évolutivité** : Les configurations IoT doivent être capables de gérer l'expansion en termes de puissance de traitement, de stockage, etc.
- 7. Identité** : Chaque appareil IoT possède une identité unique pour la communication et le suivi.
- 8. Sécurité** : La sécurité des données et des équipements IoT est cruciale pour protéger les informations personnelles des utilisateurs et prévenir les menaces.
- 9. Architecture hybride** : L'architecture IoT doit prendre en charge les produits de différents fabricants pour fonctionner dans le réseau.



4 Modern Applications :

- Smart Grids and energy saving
- Smart homes/Home automation
- Healthcare
- i
 - Radiation detection/hazardous gas detection
 - Water flow monitoring
 - Wearables
 - Robots and Drones
 - Security
 - Heart monitoring implants (Example Pacemaker, ECG real time tracking)
 - Industry

5 Advantages of IoT :

- Improved efficiency and automation of tasks.
- Increased convenience and accessibility of information.
- Better monitoring and control of devices and systems.
- Greater ability to gather and analyze data.
- Improved decision-making.
- Cost savings.

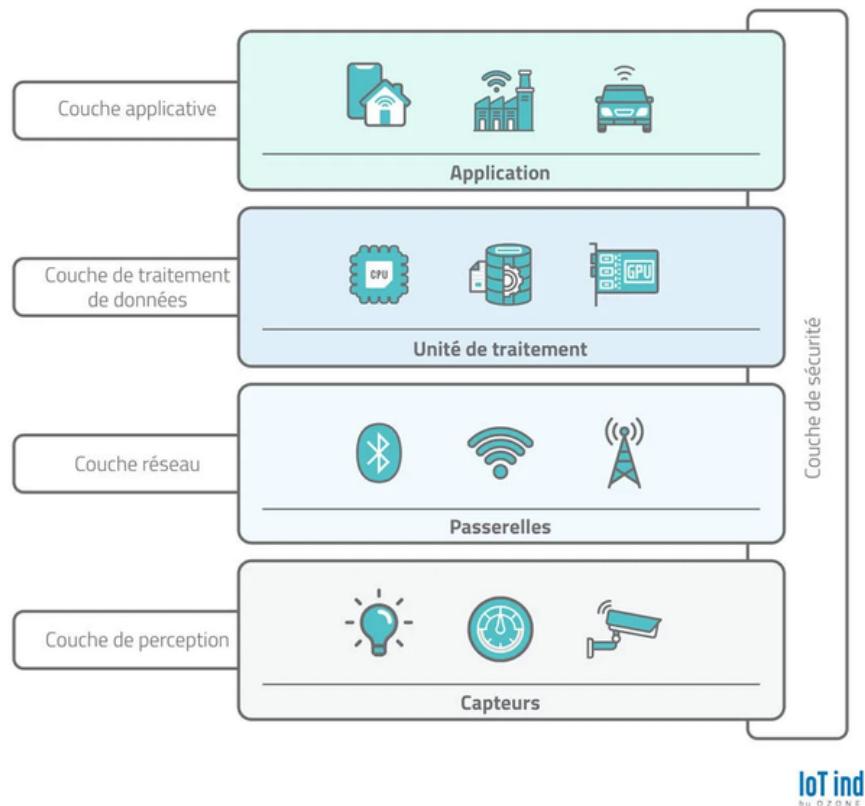
6 Disadvantages of IoT :

- Security concerns and potential for hacking or data breaches.
- Privacy issues related to the collection and use of personal data.
- Dependence on technology and potential for system failures.

7. Architecture de l'IOT :

L' architecture d'une solution IoT varie d'un système à l'autre en se basant sur le type de la solution à mettre en place.

Plusieurs normes architecturales IoT ont été proposées par plusieurs chercheurs et organismes de recherche. L'architecture la plus élémentaire est une architecture à trois couches:



IoT industriel
by OZONE CONNECT

Couche de perception : Cette couche convertit les signaux analogiques en données numériques et vice versa. Elle englobe une variété d'objets IoT, tels que des capteurs, des actionneurs et des machines, qui collectent des données physiques et les transmettent au système IoT.

Couche réseau : Responsable de la transmission des données collectées par les appareils IoT vers d'autres objets intelligents, serveurs et appareils réseau. Elle utilise différentes technologies telles que Ethernet, les réseaux cellulaires (comme la 5G et la 4G), LTE-M, NB-IoT, LPWAN, WiFi, NFC, Bluetooth ou ZigBee pour assurer la connectivité.

Couche de traitement de données : Accumule, stocke et traite les données provenant de la couche de perception. Cette couche comprend deux étapes principales : l'accumulation des données en temps réel et l'abstraction des données pour les rendre utilisables par les applications grand public. Elle utilise des plateformes IoT pour ces tâches.

Couche Applicative : C'est la couche avec laquelle l'utilisateur interagit. Elle fournit des services spécifiques à l'application à l'utilisateur. Les applications peuvent être construites directement sur les plates-formes IoT ou utiliser des API pour s'intégrer à la couche précédente.

Couche de sécurité : Transverse à toutes les couches précédentes, elle est primordiale pour garantir la sécurité de l'IoT. Elle adresse les vulnérabilités potentielles telles que celle du Log4j et assure la protection des données et des appareils IoT contre les menaces potentielles.



ÉCOLE SUPÉRIEURE DE TECHNOLOGIE
UNIVERSITÉ HASSAN II DE CASABLANCA

CHAPITRE 2 :

Spécification des besoins

Introduction

L'entrepôt d'aujourd'hui est un élément central de la chaîne d'approvisionnement, et son efficacité est essentielle pour la compétitivité des entreprises. Face à la croissance du e-commerce et à l'exigence croissante des clients en termes de délais et de traçabilité, les entrepôts doivent constamment s'adapter et optimiser leurs opérations.

Ce chapitre est consacré à la définition précise des besoins du projet d'implémentation d'un système IoT pour la gestion d'une warehouse. Il s'agit d'une étape cruciale qui permettra de garantir le succès du projet en répondant aux attentes des utilisateurs et en optimisant l'utilisation des ressources disponibles.

1 Objectif :

Le projet d'implémentation d'un système IoT pour la gestion d'une warehouse vise à répondre aux enjeux de la gestion moderne d'entrepôt et à améliorer l'efficacité et la performance de la chaîne d'approvisionnement. Les objectifs spécifiques du projet sont les suivants :

- Développer un réseau de capteurs IoT pour la surveillance des conditions d'environnement.
- Créer une application de gestion permettant de visualiser les données collectées par les capteurs.
- Mettre en œuvre des algorithmes pour optimiser la gestion de l'espace et des ressources de l'entrepôt.

2 Description du Besoin :

- **Besoins Principaux**

- Automatisation du suivi d'espace de stockage.
- Visualisation des conditions d'entrepôt.
- Sécurisation d'une zone privée

- **Besoins Spécifiques**

- Système de notification lors de la détection des activités anormale(capteurs ...).
- Interface utilisateur intuitive pour la gestion des opérations d'entrepôt.
- Intégration avec les systèmes existants de gestion des commandes et de la chaîne logistique

3 Exigences du projet :

- **Exigences Fonctionnelles :**

- Le système doit pouvoir identifier et localiser chaque zone de stockage soit elle est occupée ou non, à l'aide des tags RFID
- L'application doit fournir des rapports détaillés sur l'état de l'entrepôt et sa performance et un système de notification pour chaque intervention urgente(Incendie, chambre froide...).

- **Exigences Non Fonctionnelles :**

- Fiabilité et précision des données collectées.
- Sécurité des données, avec des mesures de protection contre l'accès non autorisés.

Conclusion:

Dans ce chapitre nous avons présenté les fonctionnalités qu'on va implémenter et la structure générale de notre SMART WAREHOUSE. Le chapitre suivant portera sur l'analyse et la conception de notre système.



CHAPITRE 3 :

Gestion de projet

Introduction

Dans cette section, nous aborderons la méthodologie de gestion de projet adoptée pour assurer une planification, une exécution et un suivi efficaces du projet. La méthodologie de gestion de projet fournit un cadre structuré pour la réalisation des activités du projet, en tenant compte des contraintes de temps, de coûts, et de ressources, tout en garantissant la satisfaction des parties prenantes.

1 Choix de la méthodologie :

Justification du choix de Scrum

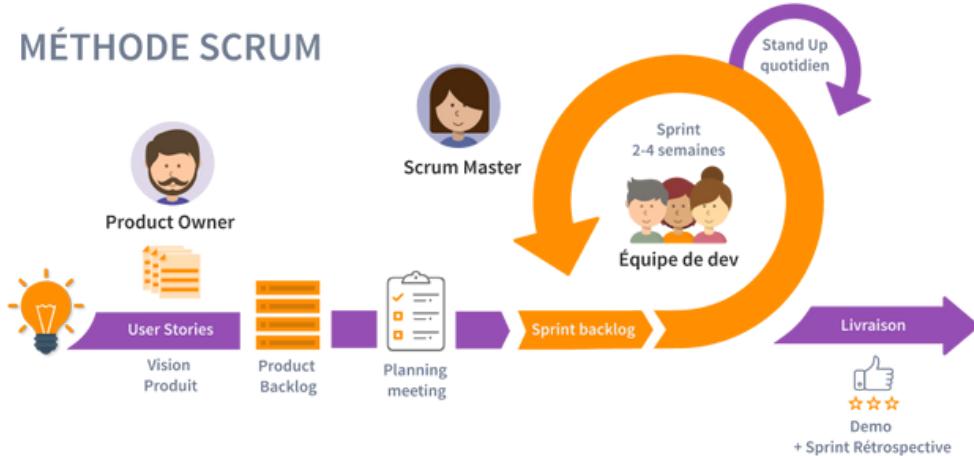
La méthodologie Scrum repose sur plusieurs principes fondamentaux qui favorisent une approche agile et collaborative de la gestion de projet. Voici un bref aperçu de ces principes :

- **Transparence** : Scrum souligne l'importance de fournir une visibilité claire aux informations, aux progrès et aux défis du projet à tous les membres de l'équipe.
- **Inspection** : Il est essentiel d'examiner régulièrement les progrès et les résultats du projet pour détecter précocement les problèmes et les erreurs, permettant ainsi des ajustements en temps opportun.
- **Adaptation** : Scrum encourage l'adaptabilité aux changements en ajustant les plans et les actions en fonction de l'évolution des circonstances, des commentaires et des besoins émergents du projet.
- **Collaboration** : Au cœur de Scrum, la collaboration efficace entre toutes les parties prenantes du projet favorise le partage des connaissances, la résolution collective des problèmes et l'obtention de meilleurs résultats.

2 Rôles et responsabilité :

Les rôles clés dans la méthodologie Scrum sont les suivants :

- **Product Owner** : Le Product Owner est responsable de définir les objectifs du projet, de gérer le backlog du produit et de prendre des décisions éclairées sur les fonctionnalités à développer. Il collabore étroitement avec les parties prenantes pour comprendre leurs besoins et priorités.
- **Scrum Master** : Le Scrum Master est le garant de l'application de la méthodologie Scrum. Il facilite les réunions et les interactions de l'équipe, s'assure du respect des principes et des valeurs de Scrum, et aide à résoudre les problèmes et les obstacles qui pourraient entraver le progrès du projet.
- **L'équipe de développement** : Composée de membres multidisciplinaires, l'équipe de développement est responsable de la réalisation des fonctionnalités du produit. Elle s'engage à collaborer et à s'auto-organiser pour atteindre les objectifs du sprint.



3 __ Sprints :

Sprint 0

- Analyse et conception

Duration : 7 jours

Sprint 1

- Initialisation d'environement

Duration : 2 jours

Sprint 2

- Contrôle d'accès
- Authentification

Duration : 7 jours

Sprint 3

- Integration Azure cloud
- Récupération de données

Duration : 7 jours

Sprint 4

- Sécurisation d'accès à la chambre privée

Duration : 5 jours

Sprint 5

- Implementation du DevOps

Duration : 7 jours

Conclusion:

En conclusion, la gestion de projet joue un rôle essentiel dans la réussite et l'accomplissement des objectifs fixés. Tout au long de ce chapitre, nous avons exploré les différentes dimensions de la gestion de projet et souligné son importance pour assurer une exécution efficace et efficiente.



CHAPITRE 4 :

Analyse et conception (Sprint 0)

Introduction

L'analyse et la conception préalables à la programmation sont essentielles pour comprendre les besoins et planifier efficacement le développement d'un système logiciel. En utilisant UML (Unified Modeling Language), je peux créer des diagrammes et des modèles qui représentent visuellement la structure, les interactions et les fonctionnalités du système. Cette approche facilite la communication, la collaboration et la documentation du projet, tout en offrant des outils puissants pour concevoir une architecture solide et anticiper les évolutions futures.

1 **UML:**

UML, ou Unified Modeling Language, est un langage de modélisation visuelle normalisé largement utilisé. Il constitue un outil puissant de communication et de collaboration entre les parties prenantes impliquées dans le développement et la conception de systèmes.



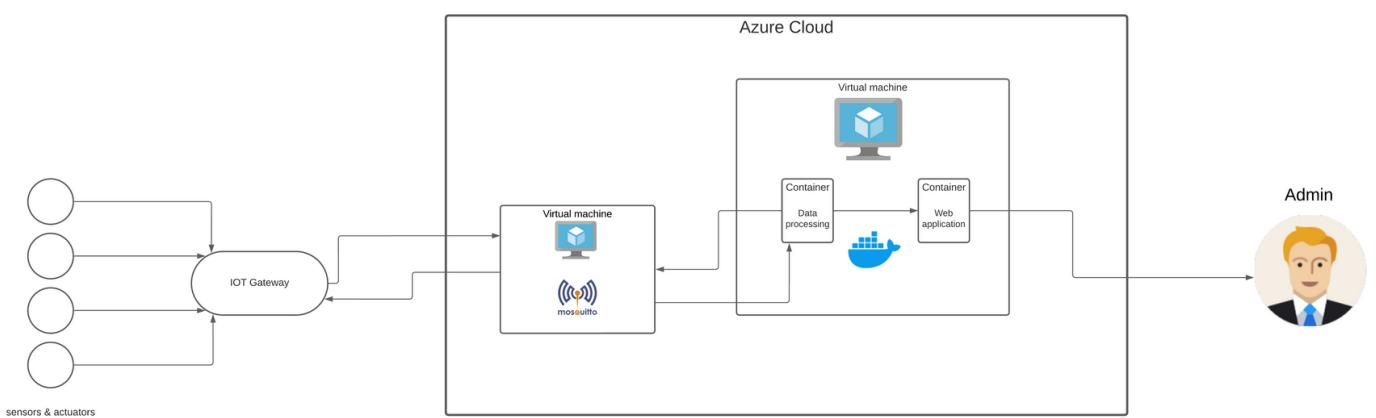
Un des diagrammes UML fondamentaux est le diagramme de classe. Il présente la structure statique d'un système, en mettant en avant les classes, leurs attributs, leurs méthodes et leurs relations. Ce diagramme offre une vision claire des composants du système et de leurs interconnexions.

2 **Conception:**

La conception joue un rôle essentiel dans la réussite d'un projet. Elle permet de définir les bases solides, de minimiser les erreurs et les risques, d'optimiser les ressources, de favoriser la communication et la collaboration, et de stimuler l'innovation. Une conception bien réalisée est cruciale pour atteindre les objectifs du projet de manière efficace et réussie.

- **Architecture système :**

L'architecture du système IoT pour la gestion d'entrepôt repose sur une approche distribuée et évolutive, conçue pour répondre aux besoins de surveillance en temps réel, de traitement des données massives et de gestion des flux d'information. Au cœur de cette architecture se trouve une infrastructure cloud robuste, offrant une scalabilité élastique pour s'adapter aux variations de charge de travail et assurer une disponibilité continue. Les capteurs IoT déployés à travers l'entrepôt collectent des données environnementales et opérationnelles, telles que la température, l'humidité, le mouvement et le poids des marchandises

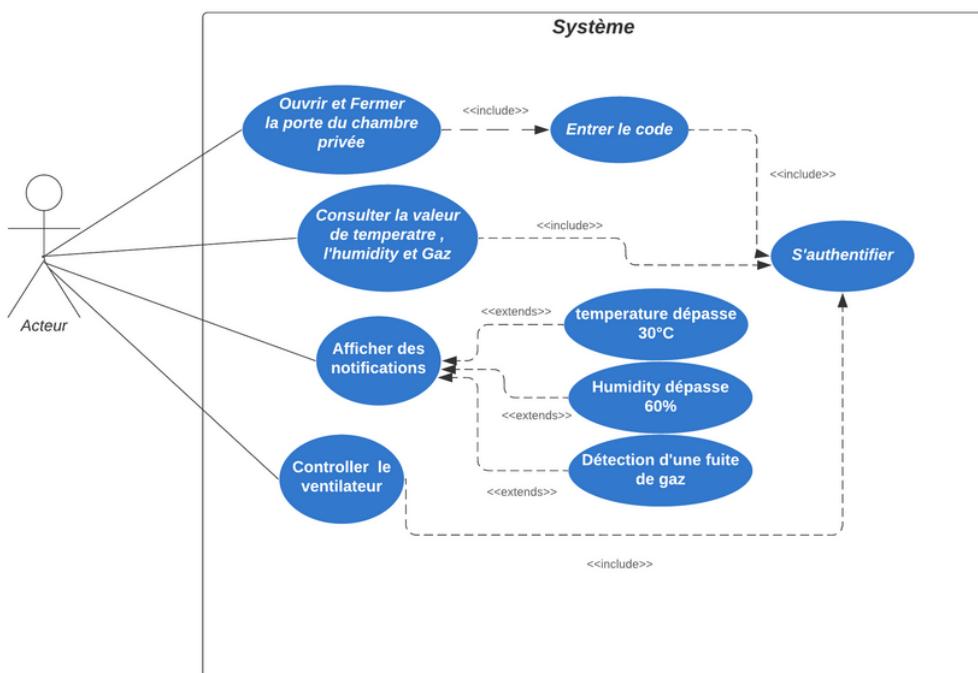


3 _Cas d'utilisation:

C' est un diagramme qui décrit les différents contextes dans lesquels un système peut être appliqué. Les diagrammes de cas d'utilisation nous fournissent un aperçu de haut niveau de ce qu'un système ou un élément d'un système se comporte sans plonger dans les détails de la mise en œuvre.

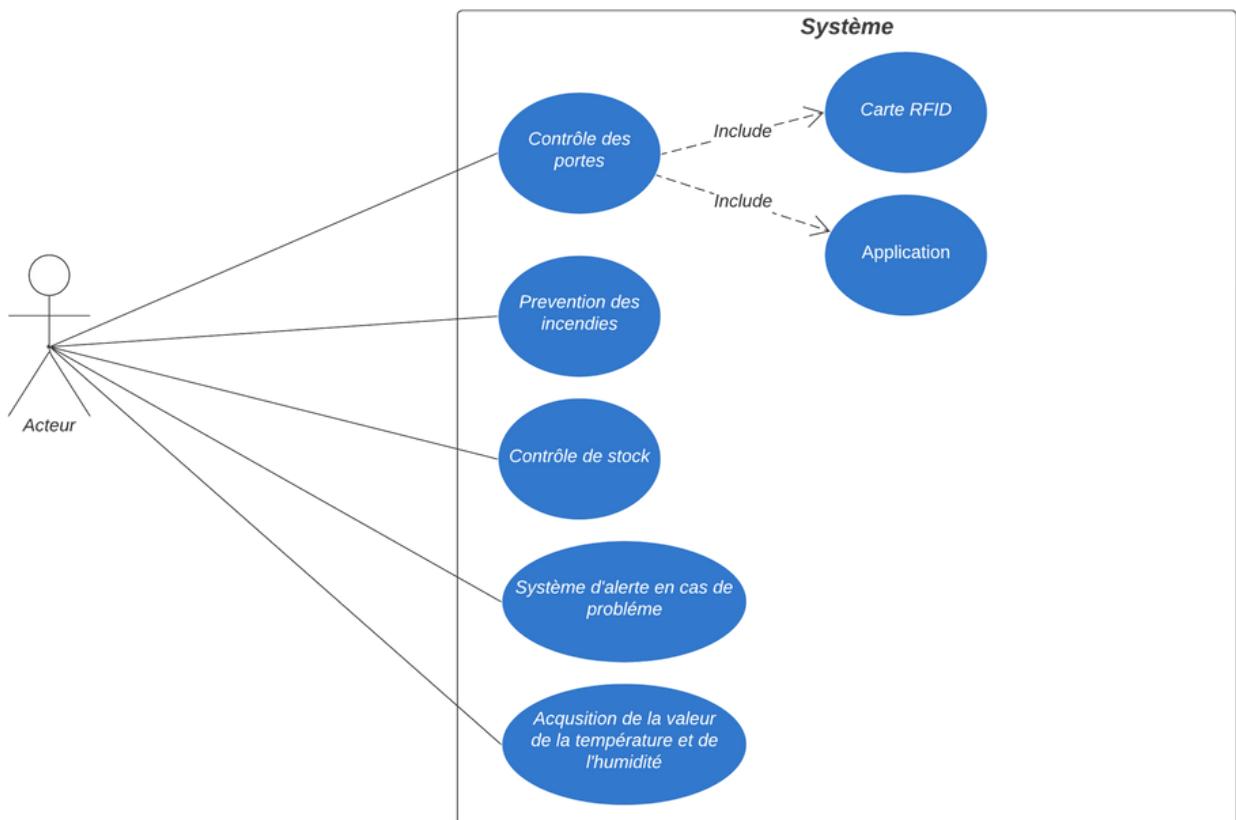
- **Application warehouse:**

Ce diagramme illustre les fonctionnalités de l'application web qui gère notre warehouse



- **Warehouse:**

Le diagramme ci-dessous montre les principales fonctionnalités de notre Warehouse



Conclusion

Dans ce sprint nous avons présenté la partie analyse et conception de notre système. Nous allons décrire dans le chapitre suivant, les composants logiciels et les matériels essentielles pour réaliser notre projet.



CHAPITRE 5 :

Réalisation



Initialisation de l'environnement

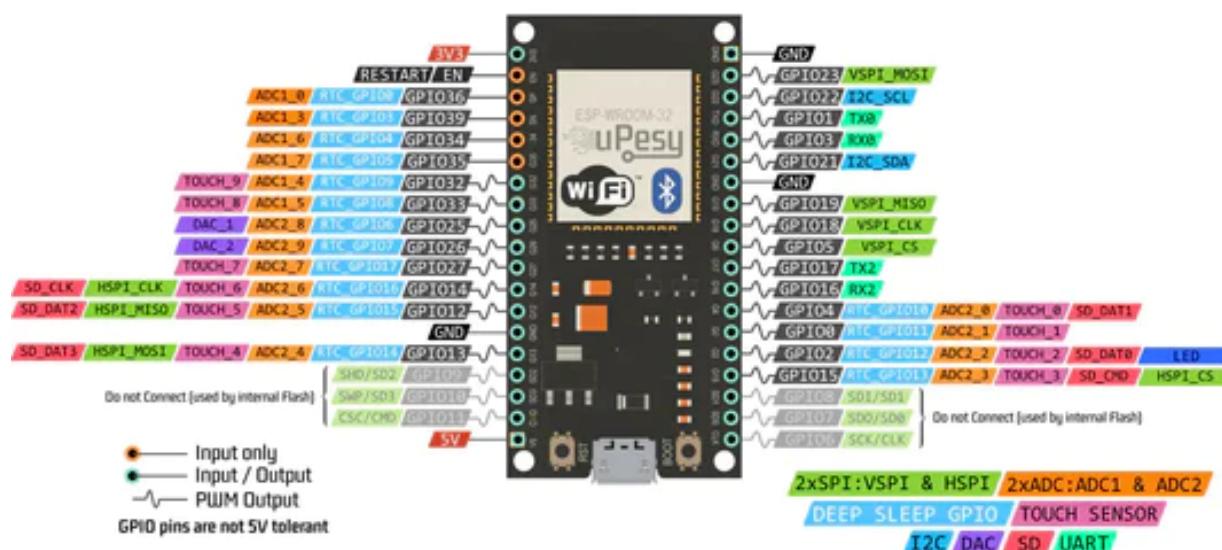
(Sprint 1)

Introduction

Dans ce sprint nous présentons les différentes technologies ainsi que les outils de contrôle utilisés lors de la phase de développement. L'étude technique consiste à mener une analyse des besoins techniques puis trouver une implémentation qui répond à ces besoins indépendamment des choix fonctionnels. Ce chapitre fera donc l'objet de capture des besoins techniques et présentera ensuite la conception générique qui décrit la solution adoptée.

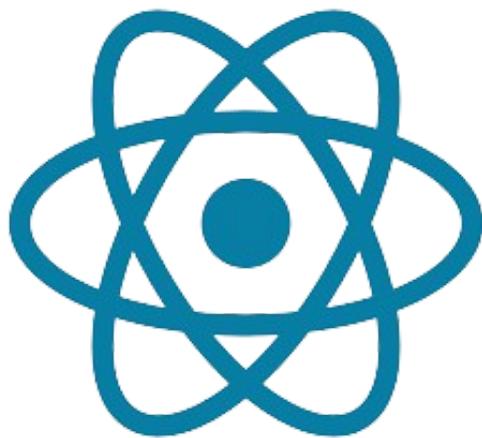
1_ESP32 :

ESP32 Wroom DevKit Full Pinout



L'ESP32 est une série de microcontrôleurs de type système sur une puce (SoC) développée par Espressif Systems, basée sur l'architecture Xtensa LX6 de Tensilica. Cette série intègre la gestion du Wi-Fi et du Bluetooth jusqu'à la version LE 5.0 et 5.1 en mode double, ainsi qu'un processeur numérique de signal (DSP). Largement utilisé dans l'Internet des objets (IoT), l'ESP32 offre une puissance de traitement robuste, une connectivité Wi-Fi et Bluetooth intégrée, et des capacités de faible consommation d'énergie. Avec son support pour Arduino IDE, MicroPython et ESP-IDF, il est accessible aux développeurs de tous niveaux.

2 _Reactjs :



ReactJS, souvent appelé React, est une bibliothèque JavaScript open-source développée par Facebook pour construire des interfaces utilisateur (UI) pour les applications web. Avec une architecture basée sur des composants, React permet aux développeurs de créer des UI réutilisables et complexes efficacement. Ses caractéristiques clés incluent un DOM virtuel pour des mises à jour d'UI efficaces et une approche déclarative pour définir les composants UI.

- Creation :

```
Terminal
> npx create-react-app my-project
> cd my-project
```

3_tailwind css :



Tailwind CSS est un framework CSS open-source qui permet de construire des interfaces utilisateur modernes de manière rapide et efficace. Contrairement aux autres frameworks CSS comme Bootstrap, Tailwind CSS se concentre sur l'utilisation de classes utilitaires pour styliser les éléments HTML, ce qui offre une flexibilité et une personnalisation accrues dans la conception d'interfaces utilisateur. Grâce à sa approche basée sur des classes utilitaires, Tailwind CSS permet aux développeurs de créer des designs réutilisables, modulaires et facilement adaptables, tout en offrant une expérience de développement plus rapide.

• Installation :

```
Terminal
> npm install -D tailwindcss postcss autoprefixer
> npx tailwindcss init
```

A screenshot of a dark-themed terminal window. The title bar says "Terminal". Inside, there are two pinkish-red command-line entries: "npm install -D tailwindcss postcss autoprefixer" and "npx tailwindcss init".

```
tailwind.config.js
/** @type {import('tailwindcss').Config} */
module.exports = {
  content: ["./src/**/*.{html,js}"],
  theme: {
    extend: {},
  },
  plugins: [],
}
```

A screenshot of a code editor showing a file named "tailwind.config.js". The code is written in a syntax similar to JSON or JavaScript. It defines a module with "content", "theme", and "plugins" properties. The "content" property is set to an array containing "src" and its subdirectories with file extensions ".html" and ".js". The "theme" property has an "extend" key. The "plugins" property is an empty array. The entire code block is preceded by a multi-line comment starting with "/* @type {import('tailwindcss').Config} */".

4_Spring boot:

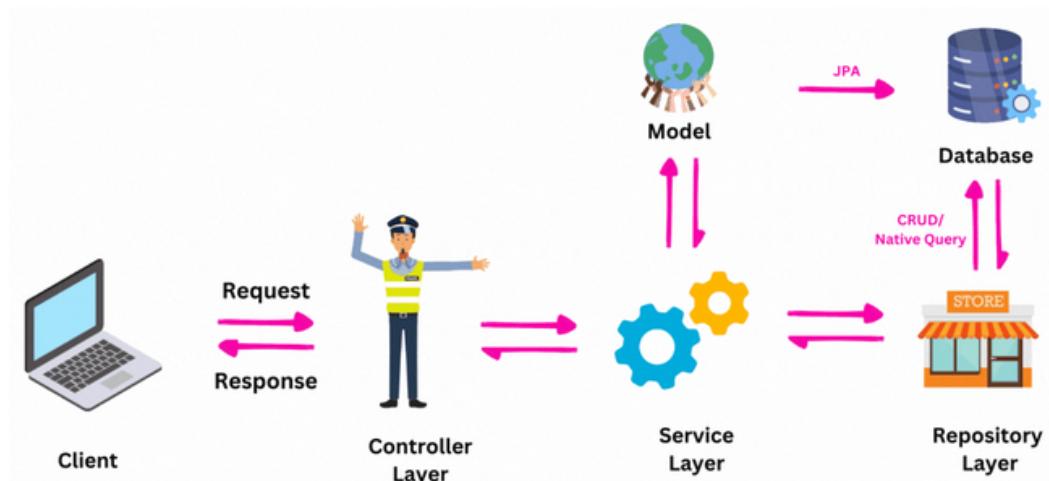


Java Spring Framework (Spring Framework) est un framework open source populaire au niveau de l'entreprise permettant de créer des applications autonomes de niveau production qui s'exécutent sur la machine virtuelle Java (JVM).

Java **Spring Boot** (Spring Boot) est un outil qui rend le développement d'applications Web et de microservices avec Spring Framework plus rapide et plus facile grâce à trois fonctionnalités principales :

- Configuration automatique
- Une approche avisée de la configuration
- La possibilité de créer des applications autonomes

Ces fonctionnalités fonctionnent ensemble pour vous fournir un outil permettant de configurer une application basée sur Spring avec une configuration et une configuration minimales. Les applications Spring Boot peuvent également être optimisées et exécutées avec le runtime Open Liberty.



Conclusion

En conclusion, le premier sprint du projet s'est concentré sur l'analyse, la conception et la configuration de l'environnement.



Contrôle d'accès et Authentification

(Sprint 2)

Introduction

Bienvenue au Sprint 2 de notre projet . Durant cette phase, nous allons nous concentrer sur deux aspects essentiels : l'authentification au niveau de l'application et le contrôle d'accès côté warehouse . Ces composantes clés jouent un rôle vital dans la garantie de la sécurité et du bon fonctionnement de notre système.

1 Contrôle d'accès :

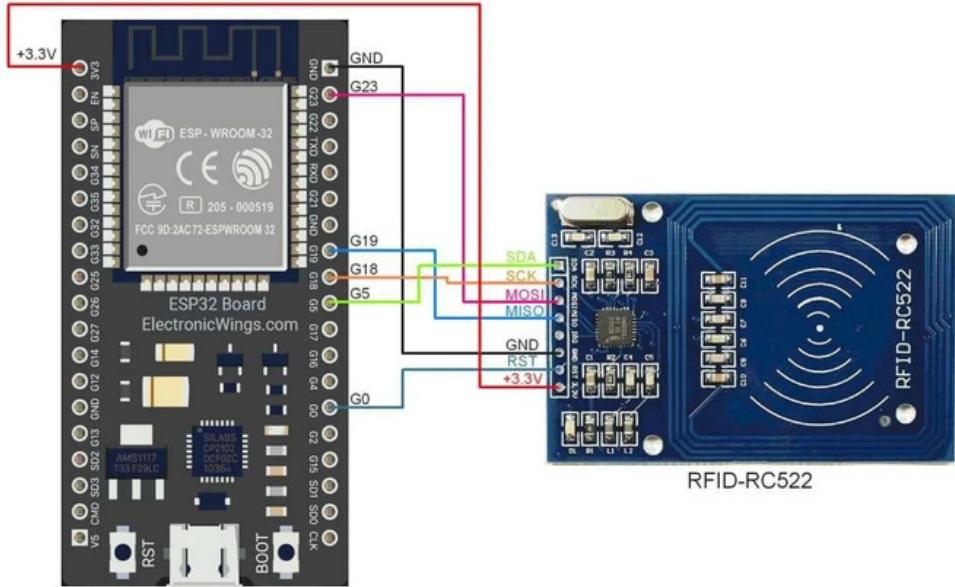
- RFID :

La technologie RFID, ou identification par radiofréquence, est un système de communication sans fil qui utilise des étiquettes, ou tags RFID, pour transférer des données à distance via des ondes radio. Chaque tag RFID contient une puce

électronique et une antenne qui lui permettent de communiquer avec un lecteur RFID. Lorsqu'un lecteur RFID émet des ondes radio, les tags RFID à proximité reçoivent ces signaux et répondent en transmettant leurs données stockées à l'intérieur de la puce. Cette technologie est largement utilisée dans divers domaines tels que la logistique, la gestion des stocks, le suivi des actifs, le contrôle d'accès, le paiement sans contact, et bien d'autres encore, en raison de sa capacité à identifier et à suivre les objets de manière rapide, précise et sans contact.



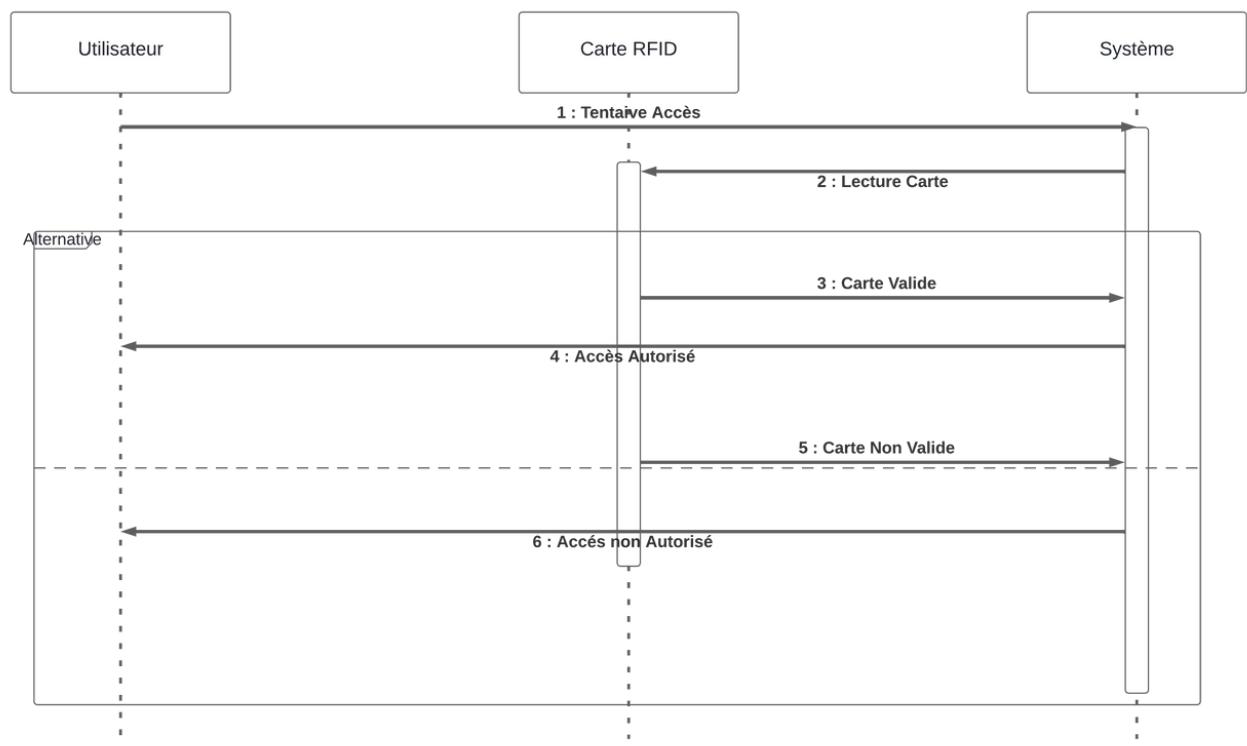
• Installation :



- **SDA** : Communication de données série avec le microcontrôleur. Connectée à la broche SDA du microcontrôleur (par exemple, GPIO 21 de l'ESP32).
- **SCK** : Synchronisation de la communication série. Connectée à la broche SCK du microcontrôleur (par exemple, GPIO 18 de l'ESP32).
- **MOSI** : Transfert des données du microcontrôleur vers le module RFID. Connectée à la broche MOSI du microcontrôleur (par exemple, GPIO 23 de l'ESP32).
- **MISO** : Transfert des données du module RFID vers le microcontrôleur. Connectée à la broche MISO du microcontrôleur (par exemple, GPIO 19 de l'ESP32).
- **IRQ** : Signalement d'interruption au microcontrôleur (optionnel). Peut être connectée à une broche GPIO du microcontrôleur (par exemple, GPIO 5 de l'ESP32).
- **RST** : Réinitialisation du module RFID. Connectée à une broche GPIO du microcontrôleur pour permettre une réinitialisation logicielle (par exemple, GPIO 22 de l'ESP32).

- **Diagramme de séquence :**

Le diagramme ci-dessous montre l'interaction du système de lecture avec la carte Rfid :



• Integration :

```
#include <SPI.h>
#include <MFRC522.h>

#define SS_PIN 22
#define RST_PIN 23

MFRC522 rfid(SS_PIN, RST_PIN);

void setup() {
    Serial.begin(9600);
    SPI.begin();
    rfid.PCD_Init();
}

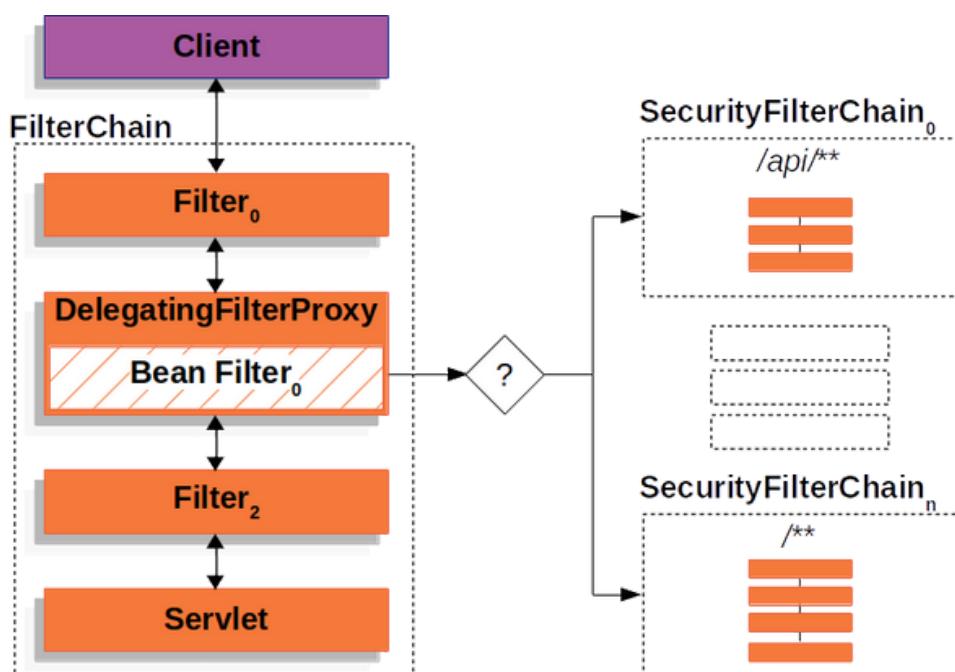
void loop() {
    if (rfid.PICC_IsNewCardPresent() && rfid.PICC_ReadCardSerial()) {
        Serial.println("Carte RFID détectée !");
        MFRC522::StatusCode status;
        byte dataBlock = 1;
        byte readData[18];
        byte bufferSize = sizeof(readData);
        status = rfid.MIFARE_Read(dataBlock, readData, &bufferSize);
        if (status == MFRC522::STATUS_OK) {
            Serial.println("Données lues depuis la carte RFID :");
            for (byte i = 0; i < 16; i++) {
                Serial.print(readData[i], HEX);
                Serial.print(" ");
            }
            Serial.println();
        } else {
            Serial.println("Échec de la lecture des données depuis la carte RFID !");
        }
        delay(1000);
    }
}
```

Ce code utilise un module RFID RC522 avec un ESP32 pour détecter et lire des cartes RFID. Une fois que le programme est téléchargé sur l'ESP32, il surveille en permanence la présence d'une carte RFID à proximité. Lorsqu'une carte RFID est détectée, le programme lit les données stockées sur la carte et les affiche sur le moniteur série. Ces données peuvent être utilisées pour diverses applications telles que le contrôle d'accès, la gestion des stocks, ou toute autre application nécessitant l'identification par carte RFID.

2 Authentification :

Pour garantir la sécurité et la confidentialité des données sensibles manipulées par le système IoT de gestion d'entrepôt, une infrastructure d'authentification robuste a été mise en place, reposant sur les technologies éprouvées de Spring Security et JWT (JSON Web Tokens).

Spring Security est un puissant cadre de sécurité qui fournit une infrastructure complète pour gérer l'authentification, l'autorisation et d'autres aspects de la sécurité dans les applications Java. Voici un aperçu de la façon dont Spring Security fonctionne :



- Integration

```
<dependencies>
    <!-- ... other dependency elements ... -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-security</artifactId>
    </dependency>
</dependencies>
```

JSON Web Tokens (JWT) est un standard ouvert (RFC 7519) qui définit un moyen compact et autonome pour transmettre de l'information de manière sûre entre deux parties, sous la forme d'un objet JSON. Voici un aperçu de son fonctionnement :

- **Génération** : Le serveur génère un JWT contenant des informations utilisateur et le signe avec une clé secrète.
- **Transmission** : Le JWT est envoyé au client après la connexion réussie.
- **Utilisation** : Le client envoie le JWT avec chaque requête protégée vers le serveur.
- **Validation** : Le serveur vérifie la signature du JWT pour s'assurer qu'il est authentique et valide.
- **Autorisation** : Si la validation réussit, le serveur autorise l'accès à la ressource demandée.
- **Expiration** : Les JWT peuvent être configurés avec une date d'expiration, après laquelle ils ne sont plus valides.
- **Renouvellement** : Les clients peuvent renouveler leur JWT en se reconnectant pour obtenir un nouveau jeton valide.

Encoded

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikpvag4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36P0k6yJV_adQssw5c
```

Decoded

```
HEADER:  
  
{  
  "alg": "HS256",  
  "typ": "JWT"  
}  
  
PAYLOAD:  
  
{  
  "sub": "1234567890",  
  "name": "John Doe",  
  "iat": 1516239022  
}  
  
VERIFY SIGNATURE  
  
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  your-256-bit-secret  
) □ secret base64 encoded
```

- Interface

Via cette interface l'utilisateur peut se connecter à l'application en saisissant son login et mot de passe. Il va être dirigé ainsi vers la page d'accueil.

 WELCOME TO ORTA

Welcome to IoT warehouse management system. Register as a member to experience.

Email address

Password [Forgot password?](#)

keep my session alive

Sign in



Conclusion

En conclusion, l'intégration de JSON Web Tokens (JWT) avec Spring Security pour l'authentification et le contrôle d'accès côté entrepôt offre une solution robuste et sécurisée pour gérer l'accès aux ressources sensibles dans le système IoT de gestion d'entrepôt.



Integration Azure cloud et Récupération de données (Sprint 3)

Introduction

Bienvenue au Sprint 3 de notre projet. Durant cette phase, nous allons nous concentrer sur deux aspects essentiels : l'intégration Azure cloud, et la récupération des données. Ces composantes clés jouent un rôle vital dans la garantie de la surveillance des capacités du Warehouse et son bon fonctionnement

1 MQTT broker

- Azure



Azure est une plateforme de cloud computing fournie par Microsoft. Elle offre une variété de services cloud, notamment le stockage de données, le calcul, l'analyse, le développement d'applications et bien plus encore.

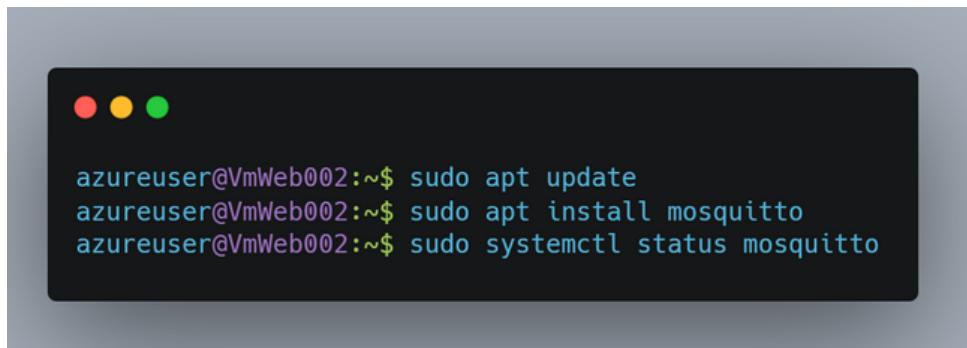
Les entreprises peuvent utiliser Azure pour héberger leurs applications, stocker leurs données, créer des solutions IoT (Internet des objets), mettre en œuvre des solutions de Big Data, déployer des applications web et mobiles, et bien d'autres choses encore. Azure est utilisé par de nombreuses entreprises pour bénéficier des avantages du cloud computing, tels que l'évolutivité, la flexibilité et la réduction des coûts opérationnels.

- Mosquitto



Mosquitto est un courtier de messages (broker) MQTT (Message Queuing Telemetry Transport) open source. MQTT est un protocole de messagerie légère et largement utilisé dans l'Internet des objets (IoT) et d'autres applications de communication entre appareils.

- **Installation :**



```
azureuser@VmWeb002:~$ sudo apt update
azureuser@VmWeb002:~$ sudo apt install mosquitto
azureuser@VmWeb002:~$ sudo systemctl status mosquitto
```

2 Réseau des capteurs:

Dans notre réseau de capteurs, nous avons intégré trois composants essentiels :

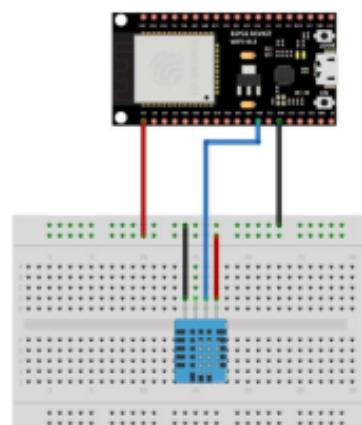
- **DHT11**



le capteur DHT11 mesure la température et l'humidité ambiantes, fournissant des données importantes pour le contrôle climatique et la gestion de l'environnement.

- **Schéma:**

Module	ESP32
DHT11	
GND	GND
VCC	3V3
DATA	14



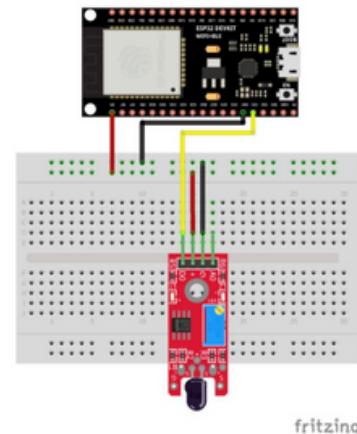
- KY026



Le capteur KY-026 détecte les flammes, permettant ainsi une surveillance en temps réel des risques d'incendie

- Schéma:

Flamme Sensor	ESP32
GND	GND
VCC	3V3
DO	13



fritzing

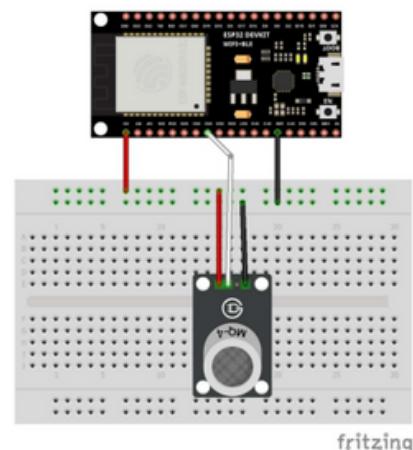
- MQ-4



Le capteur MQ-4 est conçu pour détecter divers gaz inflammables et toxiques, offrant une surveillance précise de la qualité de l'air

- Schéma:

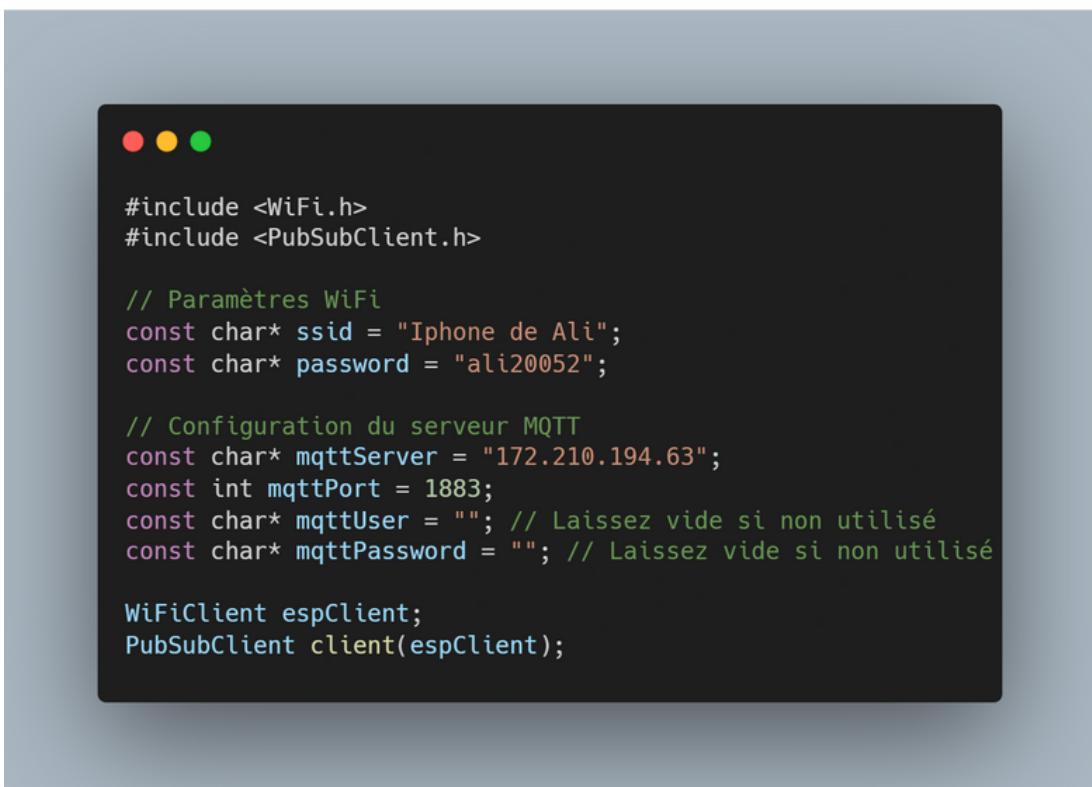
Module	ESP32
MQ2	
GND	GND
VCC	3V3
AO	34



fritzing

- **2 Publication des données :**

La publication des données via le protocole MQTT constitue une étape cruciale dans notre système de communication. En utilisant MQTT, nous adoptons un modèle de publication/abonnement qui garantit une distribution efficace des données entre les différents composants de notre système. Lorsqu'un capteur détecte une donnée pertinente, comme une lecture de température élevée ou la présence de gaz dangereux, il publie cette information sur un sujet spécifique, appelé "topic" dans le langage MQTT



The screenshot shows a terminal window with a dark background and light-colored text. At the top, there are three small colored circles (red, yellow, green). Below them is the C++ code:

```
#include <WiFi.h>
#include <PubSubClient.h>

// Paramètres WiFi
const char* ssid = "Iphone de Ali";
const char* password = "ali20052";

// Configuration du serveur MQTT
const char* mqttServer = "172.210.194.63";
const int mqttPort = 1883;
const char* mqttUser = ""; // Laissez vide si non utilisé
const char* mqttPassword = ""; // Laissez vide si non utilisé

WiFiClient espClient;
PubSubClient client(espClient);
```

initialisation de la connexion Wi-Fi et serveur MQTT

```
void reconnectMQTT() {
    while (!client.connected()) {
        Serial.println("Connexion au serveur MQTT...");
        if (client.connect("ESPDevice", mqttUser, mqttPassword)) {
            Serial.println("Connecté");
            client.subscribe("private_room"); // S'abonner à un topic
        } else {
            Serial.print("Échec, rc=");
            Serial.print(client.state());
            Serial.println(" Nouvel essai dans 5 secondes");
            delay(5000);
        }
    }
}
```

La connexion au serveur MQTT et l'abonnement au topic de publication

```
void loop() {
    // Reconnexion MQTT si nécessaire
    if (!client.connected()) {
        reconnectMQTT();
    }
    client.loop();

    // Lecture et publication périodiques des données du capteur DHT
    unsigned long currentMillis = millis();
    if (currentMillis - lastMessageTime > interval) {
        // Lecture des données du capteur DHT
        float temp = dht.readTemperature();
        float hum = dht.readHumidity();

        // Préparation et envoi des données MQTT
        doc.clear();
        doc["temperature"] = temp;
        doc["humidity"] = hum;

        String message;
        serializeJson(doc, message);
        if (client.publish("sensor_data", message.c_str())) {
            Serial.println("Données publiées avec succès");
        } else {
            Serial.println("Échec de la publication des données");
        }

        // Gestion des autres fonctionnalités...
    }
}
```

La boucle principale pour la récupération des données et la publication

- Réception des données

La Réception des données via le protocole MQTT ce base sur les même concept que ceux de la publication. En utilisant MQTT, nous adoptons un modèle de publication/abonnement qui garantit une réception efficace . Dans cet exemple lorsqu'on reçoit un message sur le topic Ventilateur qui est ON on alimente un Ventilateur avec un relais.

```
void callback(char* topic, byte* payload, unsigned int length) {  
    // Conversion du payload en chaîne de caractères  
    String message = "";  
    for (int i = 0; i < length; i++) {  
        message += (char)payload[i];  
    }  
  
    // Vérification du topic et traitement du message  
    if (strcmp(topic, "Ventilateur") == 0) {  
        if (message.equals("ON")) {  
            // Allumer le ventilateur  
            digitalWrite(15, HIGH); // Broche 15 pour contrôler le relais  
            Serial.println("Ventilateur allumé");  
        } else if (message.equals("OFF")) {  
            // Éteindre le ventilateur  
            digitalWrite(15, LOW); // Broche 15 pour contrôler le relais  
            Serial.println("Ventilateur éteint");  
        }  
    }  
}
```

La fonction callback qui reçoit les messages et exécute les commandes selon le message

La fonction callback est une fonction de rappel utilisée dans les environnements de programmation asynchrones ou événementiels. Dans le contexte de la programmation MQTT (Message Queuing Telemetry Transport), la fonction callback est appelée chaque fois qu'un message est reçu sur un topic spécifique auquel le client MQTT est abonné.

3 Visualisation des données :

- Problème à résoudre

Dans un entrepôt IoT, de nombreux capteurs génèrent continuellement des données sur divers aspects tels que la température, l'humidité, les niveaux de stock, etc. Ces données doivent être collectées, traitées et présentées aux utilisateurs en temps réel pour prendre des décisions rapides et efficaces.

Cependant, les méthodes de communication traditionnelles telles que les requêtes HTTP ne sont pas adaptées à la transmission en temps réel de grandes quantités de données. Elles peuvent entraîner des retards significatifs et une surcharge du réseau.

Et pour ça nous avons choisi de travailler avec le websocket protocole :

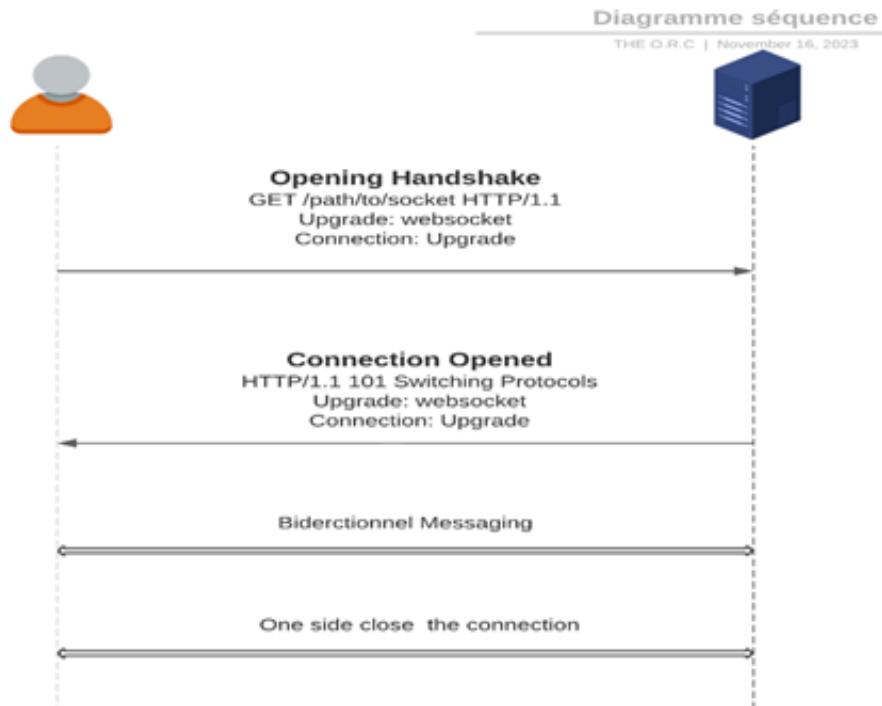
- Websocket :



WebSocket Protocol est un protocole de communication en temps réel entre des clients (généralement des navigateurs web) et des serveurs sur une seule connexion durée.

WebSocket permet une communication full-duplex, ce qui signifie que le client et le serveur peuvent s'envoyer simultanément des messages sans attendre de réponse . Cela diffère de l'HTTP traditionnel, qui est généralement basé sur des demandes et des réponses.

- Diagramme de Séquence pour l'établissement du connexion



- Integration backend

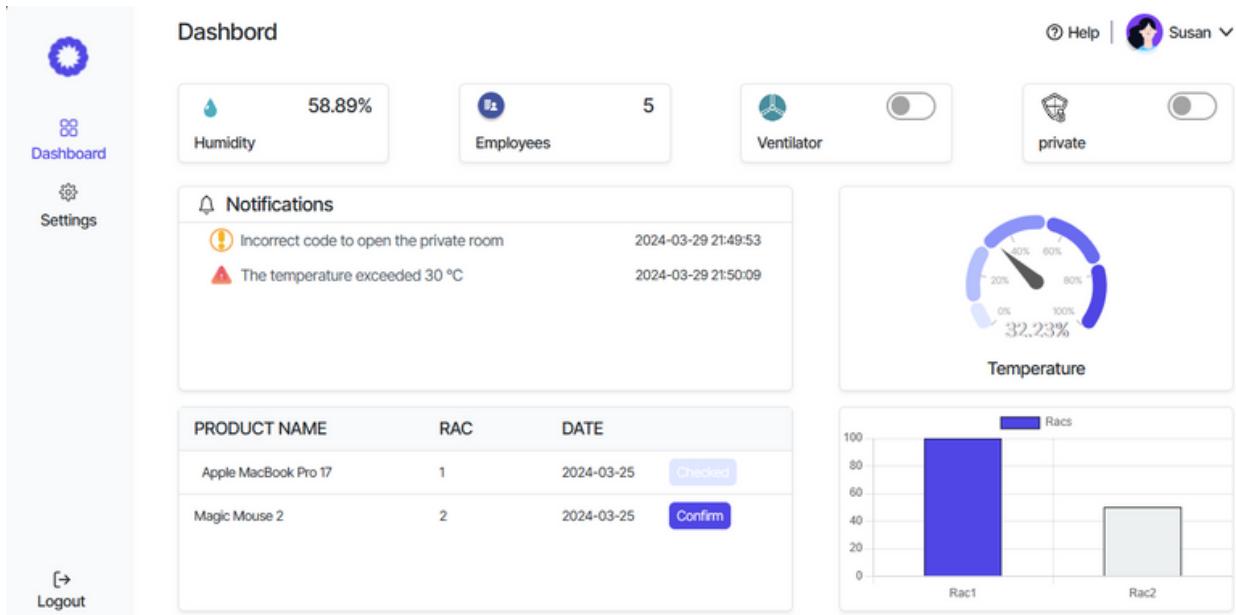
```


    <groupId>org.springframework.integration</groupId>
    <artifactId>spring-integration-mqtt</artifactId>
    <version>5.5.3</version>

  
```

A screenshot of a terminal window displaying Maven dependency configuration code. The code defines a dependency for the 'spring-integration-mqtt' artifact from the 'org.springframework.integration' group, specifying a version of '5.5.3'. Above the code, there are three colored dots (red, yellow, green) typically used in IDEs to indicate the status of the build or the execution of a command.

- Interface :



Conclusion

En conclusion l'intégration de Azure Cloud offre une solution robuste pour le déploiement du serveur MQTT qui sera utile à la suite pour la récupération des données.



ÉCOLE SUPÉRIEURE DE TECHNOLOGIE
UNIVERSITÉ HASSAN II DE CASABLANCA

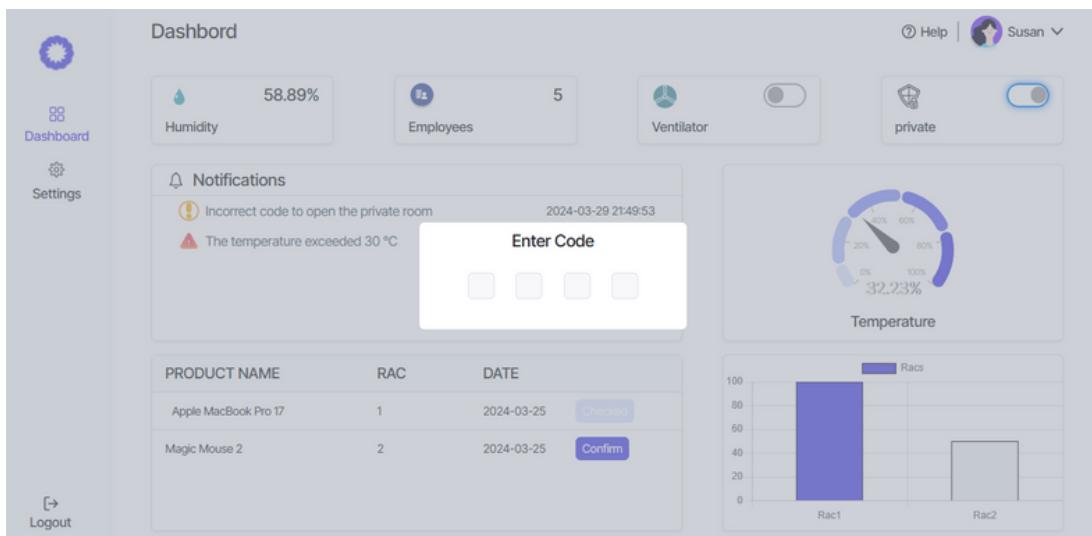
Sécurisation d'accès au chambre privée (Sprint 4)

Introduction

Bienvenue au Sprint 3 de notre projet. Durant cette phase, nous allons nous concentrer sur un aspects essentiels c'est la sécurisation d'une chambre privée à l'aide d'un accès par code contrôle avec l'application.

1 Interface

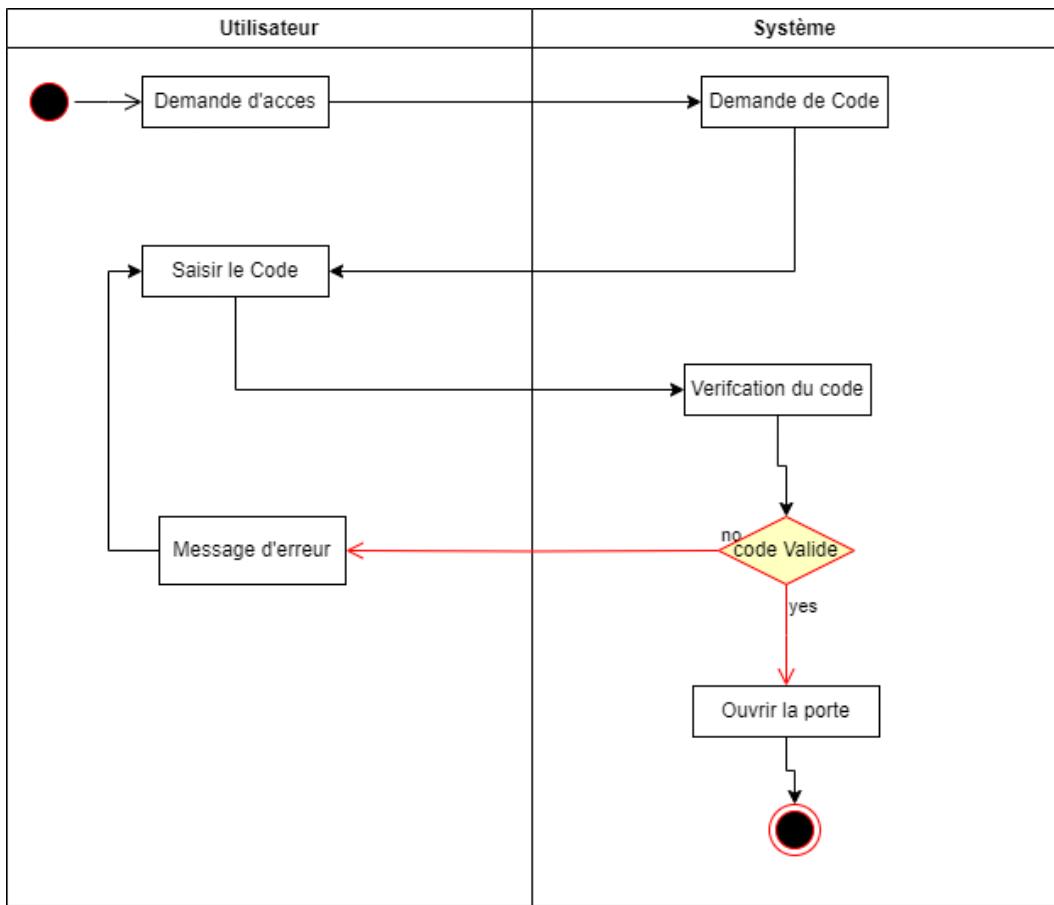
L'interface permet à l'utilisateur de saisir le code pour qui lui permettre d'accéder à la chambre privée .



2 Diagramme d'activité

- Accès chambre privée:

Ce diagramme explique le flux de contrôle nécessaire pour l'accès à la chambre privée.



3 Traitement de message :

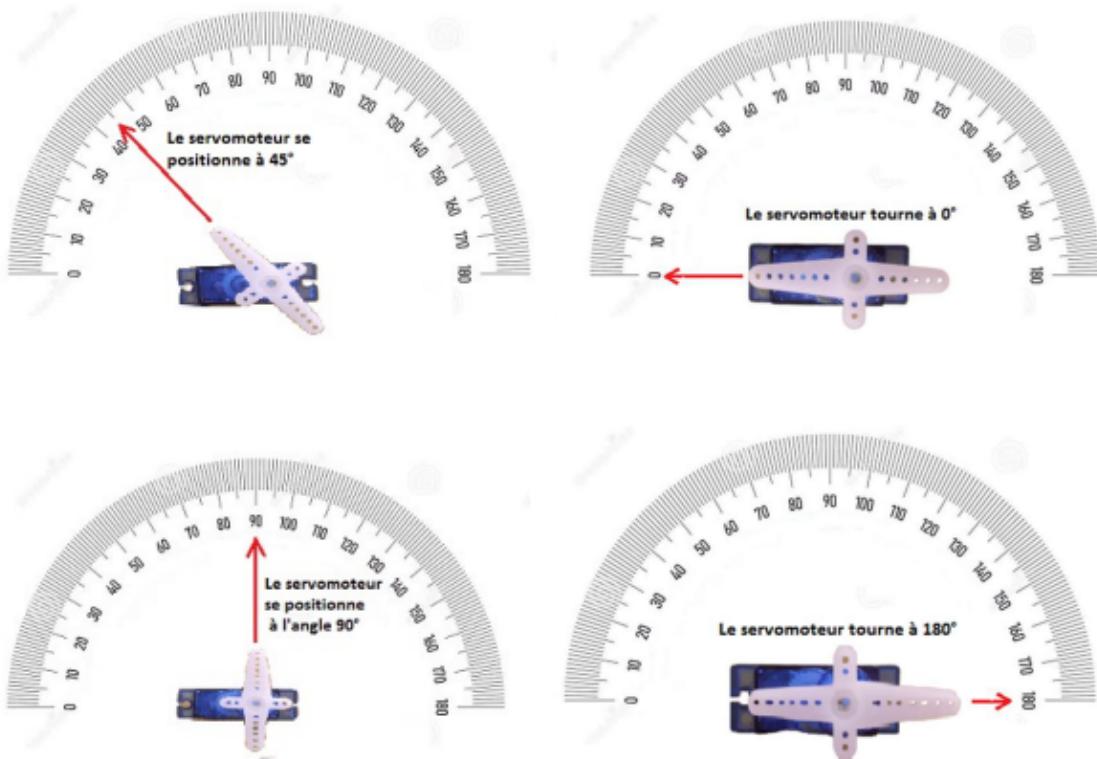
- **Servo Moteur**



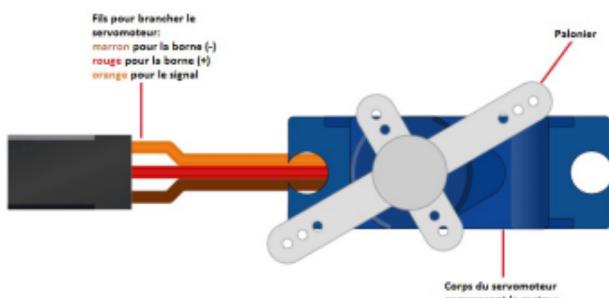
Les servo-moteurs sont des dispositifs électromécaniques capables de contrôler avec précision la position angulaire d'un arbre de sortie. Ils sont largement utilisés dans une variété d'applications, y compris la robotique, l'automatisation industrielle, les drones et les systèmes de contrôle à distance.

Les servo-moteurs sont caractérisés par leur capacité à maintenir une position spécifique même en présence de perturbations extérieures, ce qui les rend idéaux pour les applications nécessitant une précision et une stabilité élevées. En outre, ils peuvent être facilement contrôlés à l'aide de signaux de commande PWM (modulation de largeur d'impulsion), ce qui permet un contrôle précis de la position et de la vitesse du moteur.

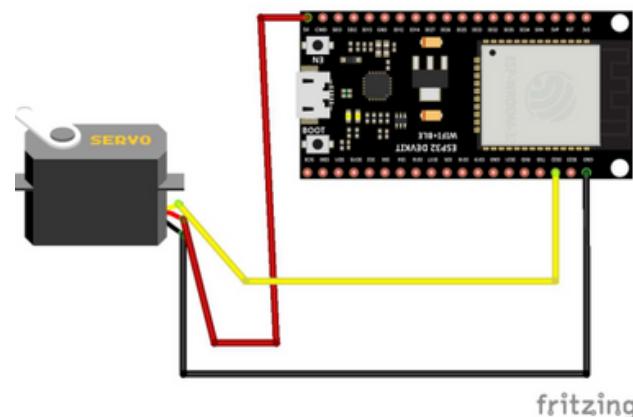
Exemples des positions du servomoteur :



- **Branchements :**



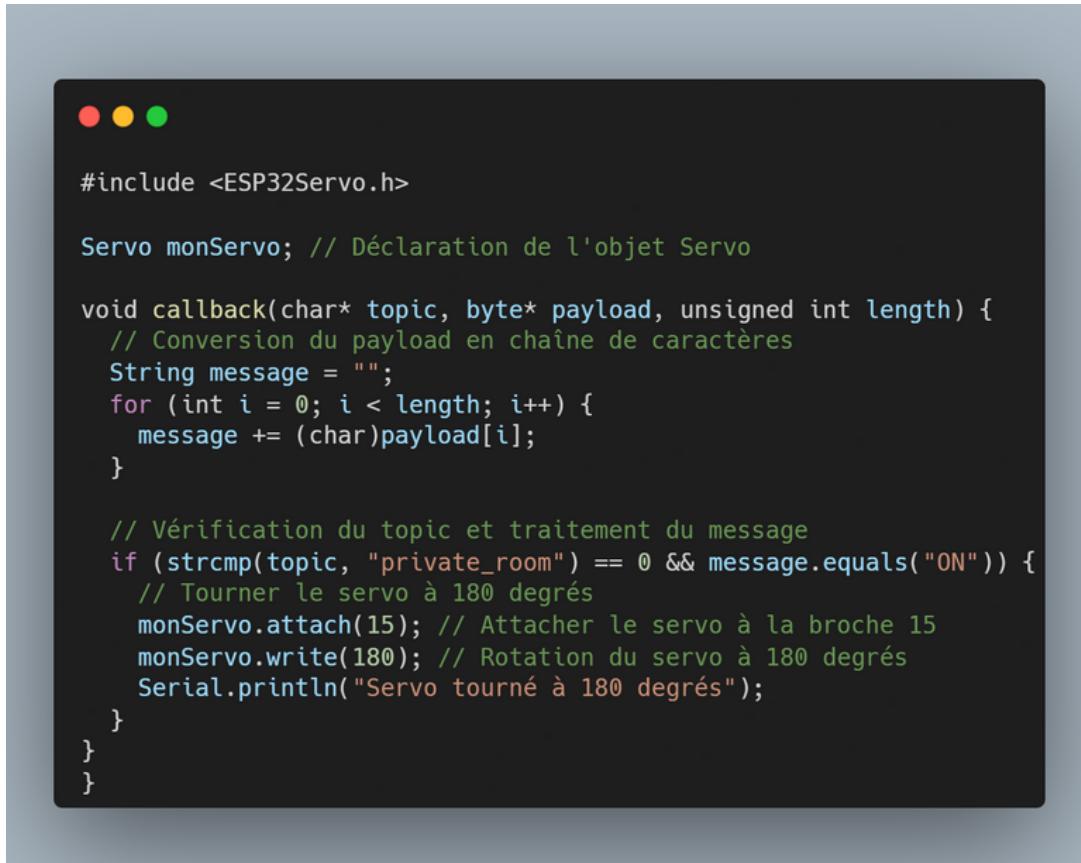
Correspondance des broches		
ESP32	Couleur du fil	Servomoteur SG90
GND	Marron	GND
5V ou 3V3	Rouge	5V
GPIO22	Orange	Signal PWM



- **côté hardware :**

En utilisant la fonction callback mentionnée dans la partie Récuperation des données on va ouvrir une porte à l'aide d'un Servo moteur qui va tourner en fonction du message reçu :

Ce code est une fonction de rappel (callback) destinée à être utilisée dans un environnement IoT basé sur une carte ESP32. Lorsque le dispositif reçoit un message sur le topic "private_room", cette fonction est déclenchée. Elle vérifie si le message est "ON" et si le topic correspond à celui attendu. Si c'est le cas, elle active un servo-moteur connecté à la broche 15, le faisant tourner à 180 degrés.



```
#include <ESP32Servo.h>

Servo monServo; // Déclaration de l'objet Servo

void callback(char* topic, byte* payload, unsigned int length) {
    // Conversion du payload en chaîne de caractères
    String message = "";
    for (int i = 0; i < length; i++) {
        message += (char)payload[i];
    }

    // Vérification du topic et traitement du message
    if (strcmp(topic, "private_room") == 0 && message.equals("ON")) {
        // Tourner le servo à 180 degrés
        monServo.attach(15); // Attacher le servo à la broche 15
        monServo.write(180); // Rotation du servo à 180 degrés
        Serial.println("Servo tourné à 180 degrés");
    }
}
```

- côté application :



```
const { webSocketConnection, setPopup , setFalseCode , setTrueCode} = useWebSocket();
const inputRefs = [useRef(null), useRef(null), useRef(null), useRef(null)];
const Code = Array.from({ length: 4 });

const handleInputChange = (index, e) => {

  const value = e.target.value;
  // Move focus to the next input when a digit is entered
  Code[index] = value
  if (value.length === 1 && index < inputRefs.length - 1) {
    inputRefs[index + 1].current.focus();
  }
  else if (index === inputRefs.length - 1){
    setPopup(false)
    if(webSocketConnection && Code.join("") === "0258"){
      webSocketConnection.send("/app/private",{},"ON")
      setTrueCode(true)
    }else{
      setFalseCode(true)
      setTrueCode(false)
    }
  }
};

};
```

Conclusion

En conclusion la Sécurisation d'accès au chambre privée reste un pilier principale de notre projet puisqu'il gère le côté sécurité.



Implémentation du DevOps

(Sprint 5)

Introduction

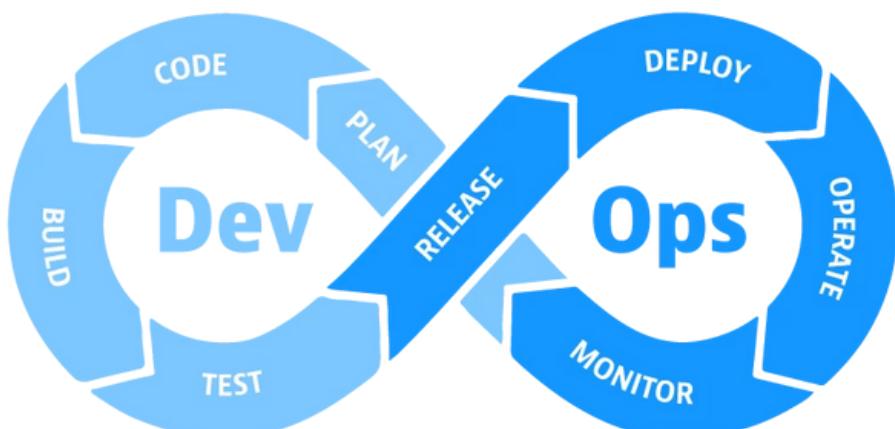
Bienvenue au dernier Sprint de notre projet. Durant cette phase, nous allons nous concentrer sur un aspects essentiels du développement de l'application, le DevOps

1 Devops:

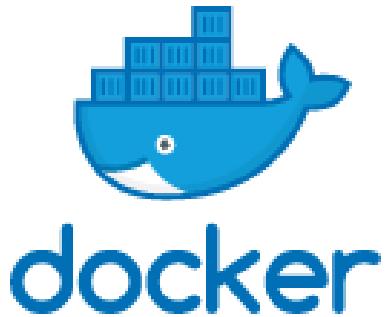
Dans un paysage technologique en constante évolution, où la demande de déploiements logiciels rapides et de haute qualité est devenue la norme, les entreprises se tournent de plus en plus vers le DevOps pour répondre à ces exigences. Le DevOps représente bien plus qu'une simple méthode de développement ou une série d'outils. C'est une approche culturelle et méthodologique qui vise à unifier les équipes de développement et d'exploitation dans un effort commun pour livrer des logiciels de manière rapide, fiable et itérative.

De plus, la complexité croissante des infrastructures logicielles, notamment avec l'essor de l'Internet des Objets (IoT) et des architectures cloud, rend nécessaire une approche plus agile et automatisée pour garantir la fiabilité et la disponibilité des services.

En intégrant le DevOps dans notre système de gestion d'entrepôt IoT, nous cherchons à adresser ces défis en adoptant une approche itérative, automatisée et axée sur la collaboration. Ce faisant, nous visons à accélérer le développement et le déploiement de nouvelles fonctionnalités, à réduire les risques associés aux mises à jour logicielles et à améliorer la satisfaction globale de nos clients en offrant des solutions toujours plus performantes et évolutives



2 Docker



Docker est une plateforme open source qui permet de créer, déployer et exécuter des applications dans des conteneurs. Les conteneurs sont des environnements légers et portables qui encapsulent les applications et leurs dépendances, permettant ainsi de les exécuter de manière cohérente et fiable sur n'importe quel environnement, qu'il s'agisse d'un ordinateur portable, d'un serveur de production ou d'un environnement cloud.

Voici les points clés sur Docker et ses fonctionnalités :

- **Isolation des applications** : Docker utilise la virtualisation au niveau du système d'exploitation pour isoler les applications les unes des autres, assurant ainsi une gestion efficace des ressources.
- **Portabilité** : Les conteneurs Docker sont portables et peuvent être exécutés sur différents systèmes d'exploitation compatibles avec Docker, simplifiant ainsi le processus de développement et de déploiement.
- **Gestion des dépendances** : Docker permet de packager les applications avec toutes leurs dépendances dans un conteneur, garantissant ainsi une cohérence et une fiabilité de fonctionnement quel que soit l'environnement de déploiement.
- **Automatisation du déploiement** : Docker simplifie le déploiement des applications en automatisant de nombreuses tâches telles que la création d'images, le déploiement de conteneurs et la gestion des mises à jour, ce qui permet de déployer rapidement et fréquemment de nouvelles fonctionnalités tout en réduisant les risques.
- **Écosystème riche** : Docker dispose d'un écosystème complet avec de nombreux outils et services complémentaires tels que Docker Compose, Docker Swarm et Docker Hub, facilitant ainsi la gestion des applications et des infrastructures conteneurisées.

- Frontend container :

```
● ● ●

# Build stage
FROM node:18 AS build

WORKDIR /app

# Copy package.json and package-lock.json to the container
COPY package*.json ./

# Install dependencies
RUN npm install

# Copy the rest of the application code
COPY . .

# Build the React app
RUN npm run build

# Serve stage
FROM nginx:1.25.1

# Copy the custom nginx.conf file to the container
COPY .docker/nginx.conf /etc/nginx/nginx.conf

# Copy the built React app from the build stage to the nginx container
COPY --from=build /app/build /usr/share/nginx/html

# Expose port 80
EXPOSE 80

# Start Nginx
CMD ["nginx", "-g", "daemon off;"]
```

- Backend container :

```
● ● ●

# Use the official Maven image as the base image
FROM maven:latest AS build

# Set the working directory
WORKDIR /app

# Copy the pom.xml and any other necessary configuration files
COPY ./pom.xml /app
COPY ./src /app/src

# Build the application
RUN mvn clean package -Dmaven.test.skip=true

# Create a new image for running the application
FROM openjdk:21-jdk

# Set the working directory
WORKDIR /app

# Copy the built JAR file from the previous stage
COPY --from=build /app/target/*.jar app.jar

# Expose the port on which the Spring Boot application will run
EXPOSE 8080

# Command to run the application
CMD ["java", "-jar", "app.jar"]
```

2 Jenkins



Jenkins est un outil open source populaire utilisé pour l'automatisation des tâches liées au développement logiciel. Il est principalement utilisé dans le cadre de l'intégration continue (CI) et du déploiement continu (CD) pour automatiser les différentes phases du cycle de vie du développement logiciel, y compris la construction, les tests et le déploiement d'applications.

- Pipeline :

```
● ● ●

pipeline {
    agent any

    stages {
        stage('Remove Existing Images') {
            steps {
                script {
                    sh 'docker rmi -f backendapp:latest'
                    sh 'docker rmi -f frontendapp:latest'
                    sh 'docker rmi -f abdo001/iot_system:front'
                    sh 'docker rmi -f abdo001/iot_system:back'
                }
            }
        }

        stage('Clean Docker Build for Backend') {
            steps {
                script {
                    sh 'docker build --no-cache -t backendapp backend'
                }
            }
        }

        stage('Clean Docker Build for Frontend') {
            steps {
                script {
                    sh 'docker build --no-cache -t frontendapp frontend'
                }
            }
        }

        stage('Tag Images') {
            steps {
                script {
                    sh 'docker tag frontendapp abdo001/iot_system:front'
                    sh 'docker tag backendapp abdo001/iot_system:back'
                }
            }
        }

        stage('Push Images to Docker Hub') {
            steps {
                script {
                    sh 'docker push abdo001/iot_system:front'
                    sh 'docker push abdo001/iot_system:back'
                }
            }
        }
    }
}
```

Conclusion

Ce projet de fin d'études a représenté une exploration approfondie du développement d'un système IoT dédié à la gestion optimisée d'un entrepôt. À travers les différents chapitres présentés, nous avons abordé une gamme variée de sujets allant de l'étude théorique de l'IoT à l'implémentation pratique des fonctionnalités spécifiques au sein de chaque sprint de développement.

Dans le cadre de cette étude, nous avons d'abord posé les bases théoriques de l'IoT, en explorant ses concepts clés, ses avantages et ses défis, ainsi que son architecture typique. Cette compréhension théorique nous a ensuite permis de spécifier les besoins précis du système à développer, en identifiant ses objectifs, ses exigences fonctionnelles et en décrivant ses fonctionnalités principales.

Par la suite, la gestion de projet a été un aspect crucial de notre démarche, où nous avons choisi une méthodologie Agile basée sur des sprints Scrum pour assurer une gestion efficace des tâches et une communication transparente au sein de l'équipe de développement. Les rôles et responsabilités ont été clairement définis, et les différentes phases du projet ont été planifiées en utilisant des outils tels que les revues et les diagrammes de Pert.

L'analyse et la conception du système ont constitué une étape fondamentale, où nous avons élaboré une architecture système détaillée et identifié les cas d'utilisation clés du système. Cette phase nous a permis de poser des bases solides pour le développement ultérieur du projet.

Enfin, la réalisation du projet a été réalisée à travers une série de sprints successifs, chacun se concentrant sur des fonctionnalités spécifiques du système. De l'initialisation de l'environnement de développement à l'intégration de services cloud comme Azure et à la sécurisation des accès, chaque étape a été soigneusement planifiée et exécutée pour atteindre les objectifs fixés.

En conclusion, ce rapport met en évidence le processus complet de développement d'un système IoT pour la gestion d'entrepôt, en soulignant les défis rencontrés, les solutions apportées et les perspectives futures pour l'amélioration et l'extension du système développé. Ce projet représente une contribution significative à l'optimisation des opérations logistiques dans le contexte des entrepôts modernes, tout en ouvrant la voie à de nouvelles possibilités d'innovation dans le domaine de la logistique et de la gestion des stocks.