Université Mohammed V
Faculté des Sciences
Rabat

RAPPORT DE PROJET « INTERNET DES OBJETS CONNECTES » POUR LA FIN DU MODULE EN MASTER PROFESSIONNEL

Mention : Informatique

Parcours : Informatique Appliquée Offshoring

Intitulé

GUIDE DE PROJET IOT : "SYSTEME SURVEILLANCE DE SANTE A DISTANCE"

Présenté le : 26 Janvier 2025

Présenté par :

- Mr YAYA Mohamedhen – Code Apogée : 23030039
- Mr LEFORT Nomenjanahary Nuno – Code Apogée : 23031093

Encadré par :

- Mme Hafssa BENABOUD – Professeur Titulaire

Table des matières

1.1 Introduction

Ce document fournit un guide étape par étape pour reproduire le projet "Système Surveillance de santé à distance". L'objectif est d'assurer que toute personne ayant accès à ces instructions puisse recréer le projet avec succès sous Windows 10.

1.2 Description Projet

Le projet Système de Surveillance de Santé à Distance vise à développer une solution intégrée permettant de surveiller, analyser et afficher les données vitales des patients en temps réel. En s'appuyant sur une architecture technologique complète, ce système garantit un suivi précis et une interaction fluide entre les différents composants techniques.

1.3 Prérequis système

Avant de commencer, assurez-vous d'avoir :

- Un ordinateur sous Windows 10.
- Une connexion Internet stable pour télécharger les outils et bibliothèques nécessaires.
- Des privilèges administrateur pour l'installation des logiciels.

1.4 Architecture Projet

Le projet se construira comme suite :

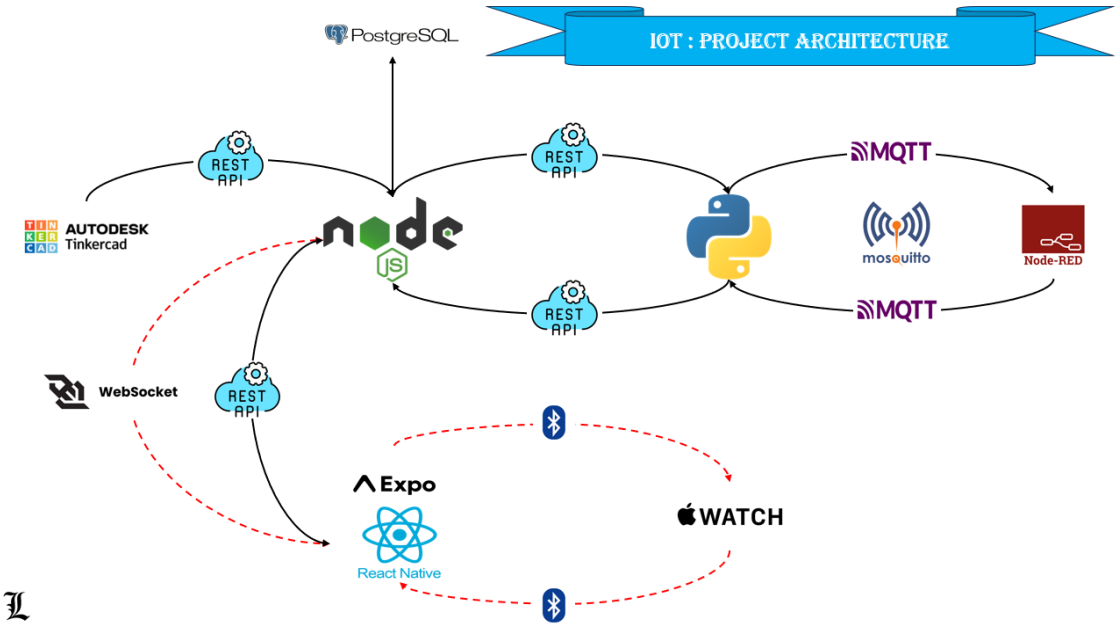

Figure 1: Architecture du projet

1.5 Diagramme de classe UML

La conception nous a amené à dégager ce diagramme UML pour le projet :

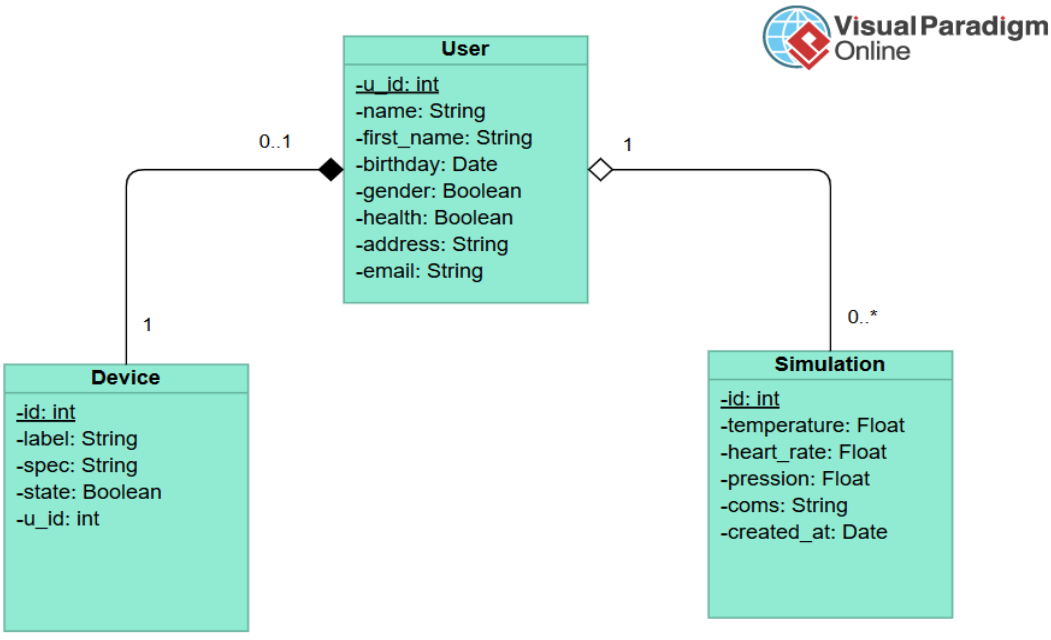

Figure 2: Diagramme de classe

1.6 Implémentation

Le code source complet, ainsi que les fichiers nécessaires pour reproduire ce projet, est disponible sur GitHub :

https://github.com/lefortnuno/IoT-3S

1.6.1 Configuration de ThinkerCAD

Utilité :

Simuler un circuit Arduino Uno pour collecter des données (e.g., fréquence cardiaque, température, taux d'oxygène, …) :

1. Accédez à ThinkerCAD https://www.tinkercad.com/.
2. Créez un compte ou connectez-vous.
3. Lancez un nouveau projet en choisissant 'Circuits'.
4. Ajoutez une carte Arduino Uno et configurez les capteurs nécessaires en suivant le schéma électrique.
5. Chargez un programme Arduino (code) pour transmettre les données via API REST.

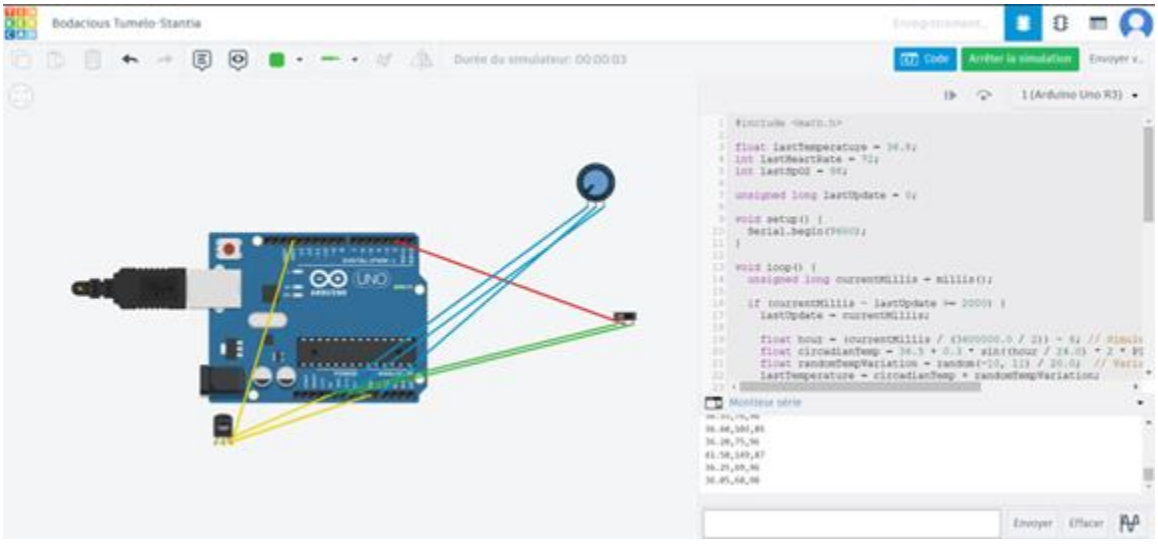

Figure 3: Simulation avec Arduino-Uno

1.6.2 Installation de Node.js et PostgreSQL

Utilité :

Node.js sert à créer une API pour stocker et gérer les données simulées, et PostgreSQL est la base de données pour persister ces données.

Node.js

1. Téléchargez et installez Node.js depuis https://nodejs.org/.
2. Vérifiez l'installation avec :
   node -v
   npm -v
1. Lancez :
   a. Project\IoT-3S\api_service\> npm install
   b. Project\IoT-3S\api_service\> npm start
2. Résultat :

   Lancé sur ADRESS_IP_MACHINE:PORT ....

   Connexion à la base de données '[nom_base_de_donnée]' réussie.

PostgreSQL

1. Téléchargez PostgreSQL depuis : https://www.postgresql.org/download/.
2. Suivez l'assistant d'installation pour configurer une base de données locale avec pgAdmin.
3. Créez une base de données nommée par exemple 'surveillance_sante'.
4. Configurez les tables suivant le fichier :

   Project\IoT-3S\api_service\config\sql arduino-uno-init.sql

1.6.3 Configuration de Python, MQTT et Mosquitto

Utilité :

Python pour traiter les données en temps réel, MQTT pour la communication, et Mosquitto comme broker MQTT.

Python:

1. Téléchargez Python depuis https://www.python.org/.
2. Ajoutez Python à votre PATH lors de l'installation.
3. Installez les bibliothèques nécessaires : pip install paho-mqtt psycopg2
4. Lancez :

```
Project\IoT-3S\python> py main.py
```

Mosquitto:

1. Téléchargez Mosquitto depuis https://mosquitto.org/download/.
2. Installez Mosquitto et démarrez le service en exécutant :

```
C:\Program Files\mosquitto> mosquitto
```

3. Configurez un fichier mosquitto.conf pour ajuster le port ou ajouter un mot de passe si nécessaire.

1.6.4 Installation et Configuration de Node-RED

Utilité :

Afficher les données en temps réel dans un tableau de bord.

1. Installez Node-RED globalement : npm install -g node-red
2. Lancez Node-RED depuis CMD avec : node-red
3. Accédez à l'interface via http://localhost:1880 dans un navigateur.
4. Ajoutez des nœuds MQTT et connectez-les au broker Mosquitto pour recevoir les données.

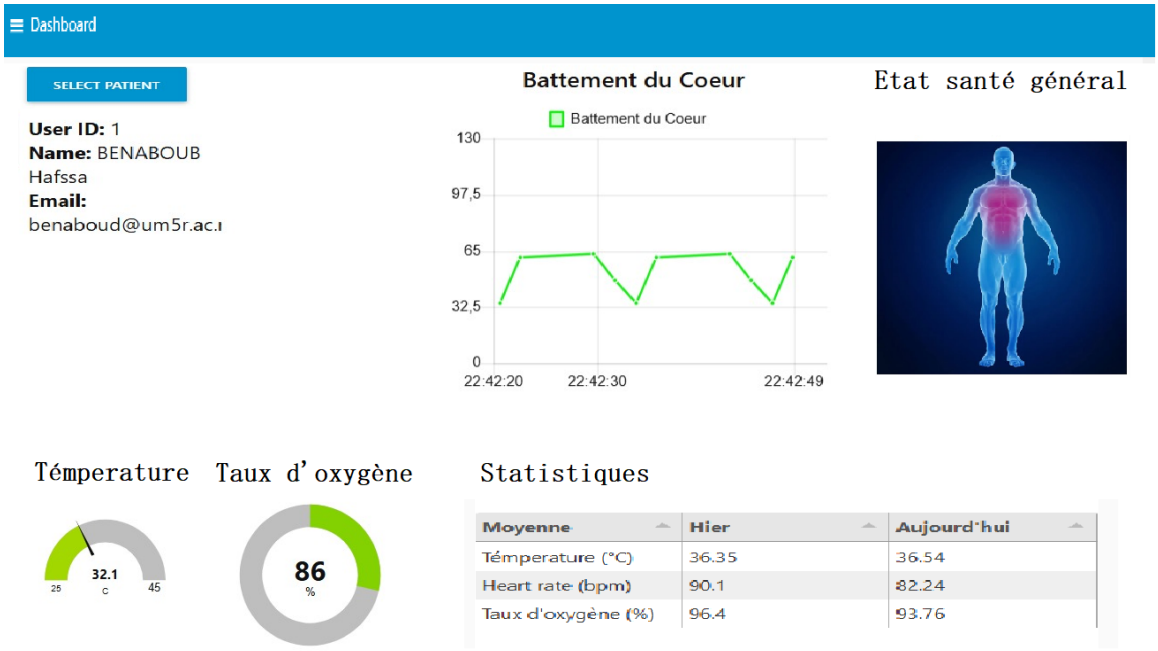

Figure 4: Tableau de bord node-RED

1.6.5 Développement de l'Application Mobile avec React Native

Utilité :

Fournir une interface utilisateur pour les données des patients et effectuer des actions CRUD.

1. 1.Telecharger Expo Go dans App Store(iOs) | Play Store(Android)
2. Lancez :
   Project\IoT-3S\mobile> npm install
   Project\IoT-3S\mobile> npm start
3. 3.Testez l'application sur un appareil physique en scannant le QR code avec Expo Go.

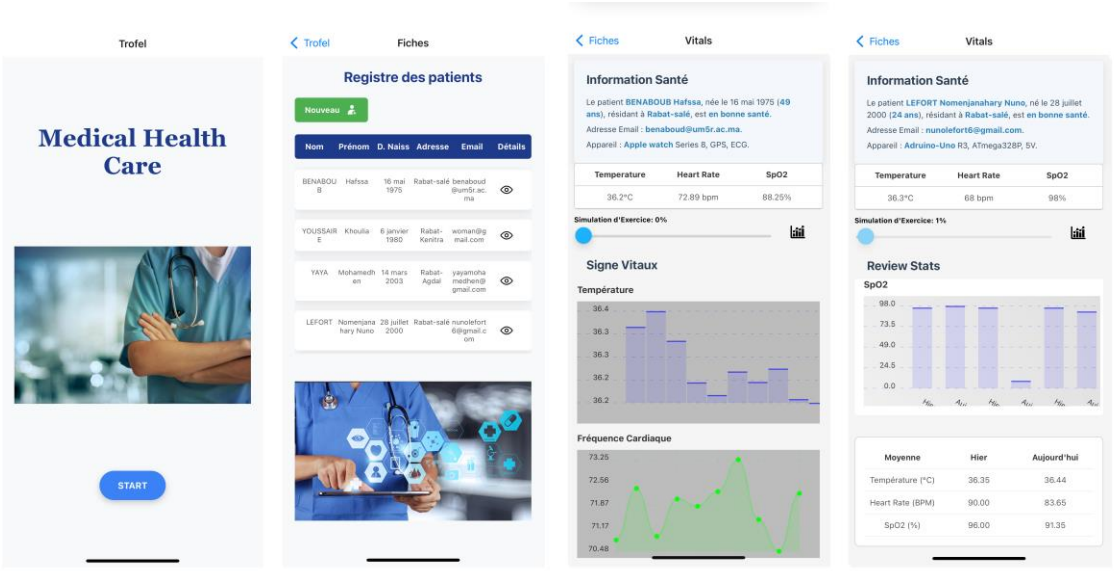

Figure 5: Interface Mobile

1.6.6 Implémentation de l'Envoi d'E-mails

Utilité :

Notifier les utilisateurs pour des alertes ou des événements critiques.

1. Installez Nodemailer : npm install nodemailer
2. Configurez un transporteur SMTP avec Gmail :

```
const nodemailer = require('nodemailer');

const transporter = nodemailer.createTransport({

  service: 'gmail',

  auth: { user: 'votre_email@gmail.com', pass: 'votre_mot_de_passe' }

});
```

3. Ajoutez une fonction pour envoyer des e-mails avec Nodemailer.

1.7 Tâches Réalisées

1.7.1 Collecte des données de santé

Configuration de capteurs simulés (fréquence cardiaque, température corporelle, SpO2) avec Tinkercad pour une émission de données en temps réel.

1.7.2 Transmission des données

Mise en place d'un broker MQTT (Mosquitto) pour la réception des données transmises par les capteurs simulés.

1.7.3 Traitement des données

Développement d'un script Python pour :

- Collecter les données en temps réel depuis le broker MQTT.
- Détecter les anomalies en comparant les données à des seuils critiques.
- Calculer des statistiques comme les moyennes et tendances.

1.7.4 Visualisation et suivi

Création d'un tableau de bord interactif avec Node-RED, incluant des graphiques et jauges pour suivre les paramètres vitaux en temps réel.

1.7.5 Stockage des données

Configuration d'une base de données cloud (ThingSpeak) pour le stockage et l'analyse des données historiques.

1.7.6 Alertes

Intégration d'un service d'alertes en temps réel via e-mail avec IFTTT en cas de détection d'anomalies.

1.8 Difficulté

1.8.1 Collecte des données avec ThinkerCAD Arduino

La collecte des données simulées via ThinkerCAD Arduino a été un véritable défi, car elle nécessitait une configuration précise des capteurs et du code Arduino. Heureusement, la création d'une API Node.js a permis d'automatiser la récupération des données, évitant ainsi une collecte manuelle fastidieuse.

1.8.2 Adaptation à Node-RED

La prise en main de Node-RED a été complexe, en particulier pour la gestion des e-mails. Une limitation notable est que le nœud d'envoi d'e-mails de Node-RED n'est pas dynamique : il impose de saisir manuellement l'adresse du destinataire, ce qui n'est pas idéal pour un système automatisé.

1.8.3 Synchronisation avec Node.js et PostgreSQL

Pendant la phase d'intégration des données collectées dans la base PostgreSQL via l'API Node.js, des erreurs de schéma et des incompatibilités dans le mapping des types de données ont ralenti le processus. Ces problèmes ont été résolus en ajustant les requêtes SQL et en effectuant une validation stricte des données avant leur insertion.

1.9 Perspectives d'amélioration

1.9.1 Collecte des données avec Apple Watch

Nous n'avons pas pu implémenter la collecte de données provenant d'une Apple Watch en raison du manque de matériel disponible pour effectuer les tests nécessaires. Bien que les recherches aient été effectuées et que les bases pour utiliser WatchKit et HealthKit soient comprises, l'absence d'Apple Watch nous a empêchés de finaliser cette fonctionnalité.

1.9.2 Connexion via Socket

La mise en œuvre d'une connexion en temps réel avec Socket.io n'a pas pu être réalisée dans le délai imparti. Cela aurait permis une meilleure réactivité pour la transmission des données entre les composants du système.

1.9.3 Transformation du script Python en API Flask

Bien que le script Python utilisé pour traiter les données soit fonctionnel, il aurait été idéal de le convertir en une API Flask. Cela aurait facilité son intégration avec d'autres parties du système, mais cela n'a pas pu être finalisé par manque de temps.

1.10 Test et Validation

1. Simulez des données avec ThinkerCAD.
2. Vérifiez leur transmission via MQTT ; HTTPS et leur stockage dans PostgreSQL.
3. Assurez-vous que les données apparaissent dans Node-RED et l'application Mobile.
4. Testez l'envoi des e-mails avec des alertes simulées.

1.11 Conclusion

L’IoT révolutionne le suivi médical avec ce « Système de Surveillance de Santé à Distance ». Ce projet, et l’IoT en général, ont requis une précision chirurgicale au niveau algorithmique, ce qui a rendu le processus encore plus passionnant. Travailler dessus a été une expérience incroyablement enrichissante, mêlant rigueur technique, innovation pratique, et la satisfaction de voir des idées se concrétiser pour répondre à des enjeux réels.

C’est avec enthousiasme que nous imaginons les perspectives qu’un tel système pourrait offrir dans le futur !