**Documentatie proiectare**

**Claudiu COSTEA**

**Goran CODAUSI**

**Dimitrie DEREVCIUC**

**Adrian CODAUSI**

**Patrick DAMIAN**

**Denisa COTOLAN**

**Sara COSARBA**

**Miruna CALUGAR**

**Diana DRAGOTEL**

1. **Arhitectura Programului**

Arhitectura programului pentru sistemul de monitorizare a sănătății se va baza pe o arhitectură three-tier, în care vor fi implicate următoarele componente:

**Componente:**

**1. Dispozitivele de măsurare (Senzori):**

* Acestea sunt dispozitive specializate care măsoară parametrii fiziologici precum ECG, temperatură și puls. Vor trimite datele către placa de dezvoltare (Arduino/ESP8266) pentru prelucrare și transmitere către gateway.

**2. Placă de dezvoltare (Arduino/ESP8266):**

* Acționează ca o interfață între senzori și gateway-ul de comunicație.
* Primește datele de la senzori, le procesează și le formatează pentru transmitere către gateway.
* Utilizează Arduino IDE sau alte medii de dezvoltare similare pentru programarea și configurarea.

**3. Gateway (ESP8266):**

* Primește datele transmise de placa de dezvoltare și le trimite către brokerul MQTT.
* Realizează conexiunea între dispozitivele fizice și infrastructura de comunicare bazată pe rețea.
* Utilizează biblioteci și funcționalități specifice MQTT pentru comunicare.

**4. Broker MQTT:**

* Componenta centrală de comunicație în cadrul sistemului IoT.
* Primește datele publicate de gateway și le distribuie clienților abonați la topicurile respective.
* Utilizează protocoalele MQTT pentru comunicare.

**5. Client intermediar:**

* Poate fi un alt serviciu sau dispozitiv care se abonează la topicurile MQTT relevante pentru a prelua datele publicate de brokerul MQTT.
* Prelucrarea și transmiterea datelor către serviciul Cloud găzduit pe platforma Heroku.

**6. Serviciu Cloud (găzduit pe Heroku):**

* Primește datele transmise de clientul intermediar și le procesează pentru analiză ulterioară.
* Folosește algoritmi pentru detecția anomaliilor și generează notificări în cazul unor valori anormale.
* Găzduit pe platforma Heroku și utilizează tehnologii web pentru comunicare.

**Sistem de operare:**

* Sistemul de operare utilizat poate fi integrat pe dispozitivele hardware, iar pentru placa de dezvoltare Arduino sau ESP8266 se utilizează sisteme de operare în timp real (RTOS) dedicate pentru dispozitivele de tip microcontroler.
* Pentru serviciul Cloud de pe platforma Heroku, sistemul de operare este gestionat de către infrastructura de cloud și nu necesită intervenția directă a dezvoltatorului.

**Interacțiuni:**

Diagrama UML in anexa\*

**1. Senzori → Placă de dezvoltare (Arduino/ESP8266):**

* Senzorii vor trimite datele lor către placa de dezvoltare.
* Placa de dezvoltare (Arduino/ESP8266) va colecta și prelucra datele de la senzori și le va formata pentru transmiterea către gateway.

**2.** **Placă de dezvoltare → Gateway (ESP8266):**

* Placa de dezvoltare va transmite datele prelucrate și formatate către gateway (alt ESP8266 sau alt dispozitiv dedicat).
* Gateway-ul poate fi configurat pentru a primi aceste date și a le transmite către brokerul MQTT.

**3. Gateway → Broker MQTT:**

* Gateway-ul va publica datele pe topicurile MQTT relevante către brokerul MQTT.
* Datele publicate vor include parametri fiziologici precum ECG, temperatură, puls etc.

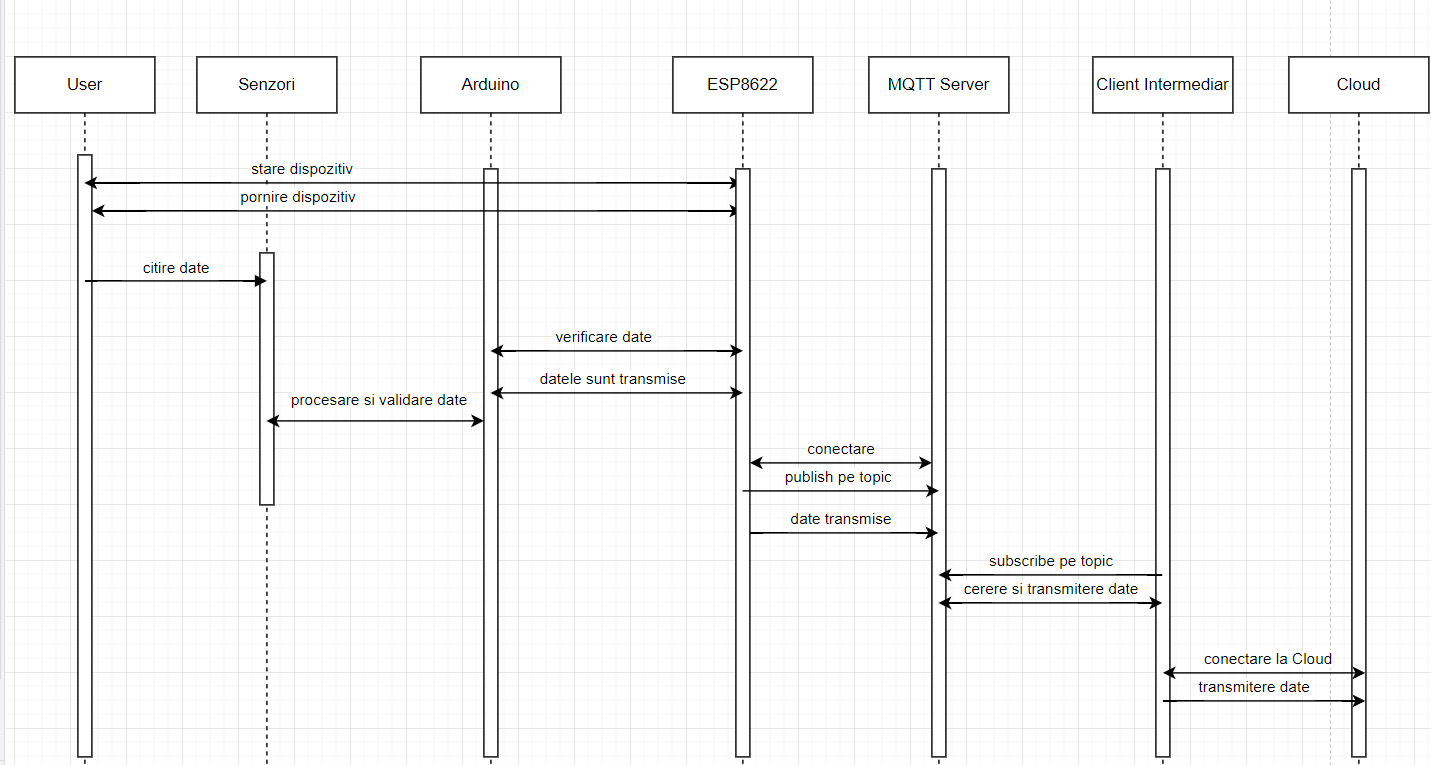
**4. Broker MQTT → Client intermediar:**

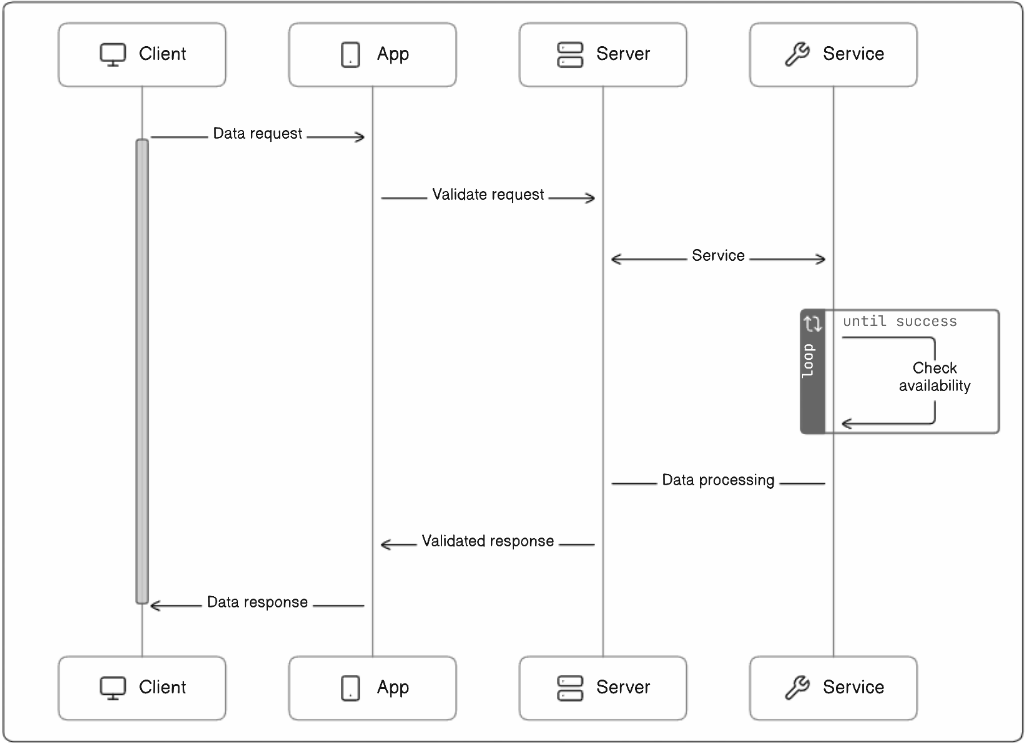
* Clientul intermediar, care poate fi un alt serviciu sau dispozitiv, va abona la topicurile relevante de pe brokerul MQTT pentru a prelua datele publicate.
* Acest client intermediar va prelua datele de la brokerul MQTT și le va transmite către serviciul Cloud găzduit pe Heroku.

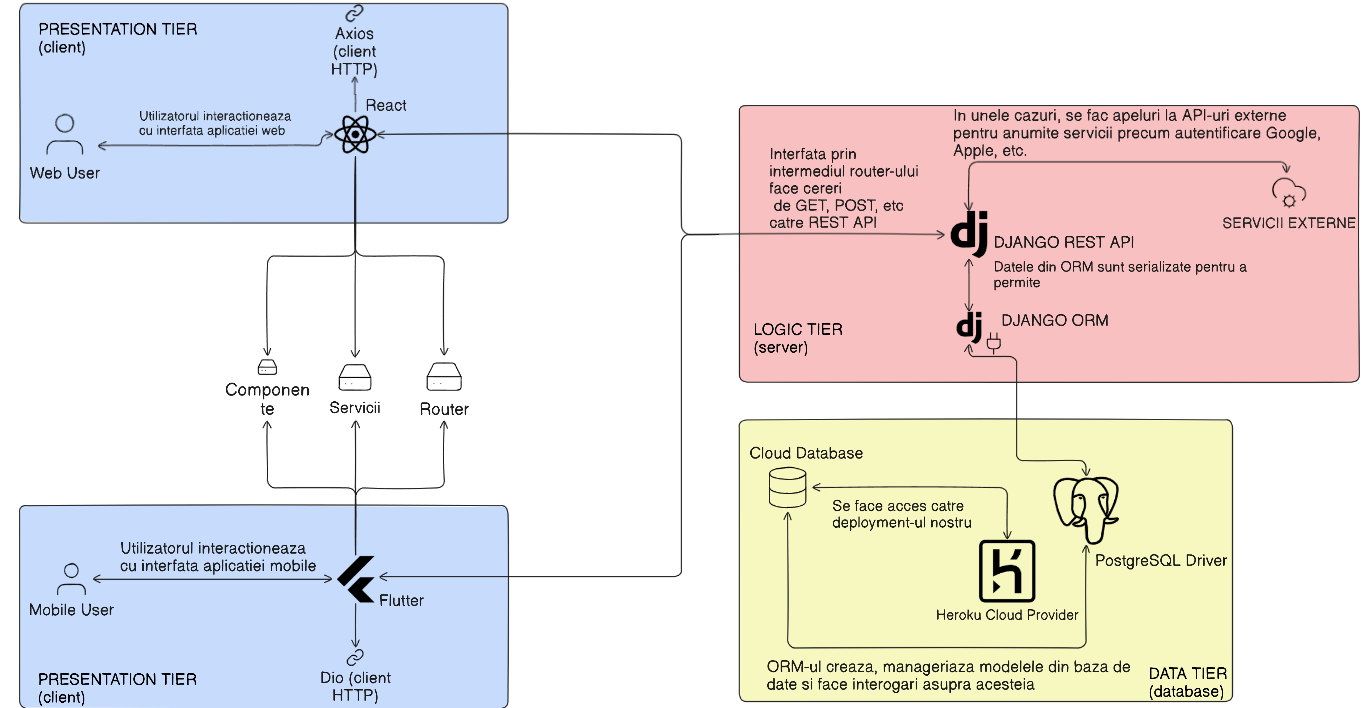
**5. Client intermediar → Serviciu Cloud:**

* Clientul intermediar va transmite datele către serviciul Cloud de pe platforma Heroku.
* Aceste date vor fi recepționate și procesate de serviciul Cloud pentru analiză ulterioară și notificare în cazul unor valori anormale.

Adăugarea acestui client intermediar între brokerul MQTT și serviciul Cloud poate facilita gestionarea și procesarea datelor în mod mai eficient, permițând, de asemenea, să se integreze alte funcționalități sau procese de prelucrare intermediară a datelor înainte de a ajunge la serviciul Cloud.







Arhitectura / Design Frontend : <https://www.figma.com/file/5Nu8CzaXfvCyoK8LMh6ApU/IP-proiect?type=design&node-id=0-1&mode=design&t=phBsOOVKECmVxEn9-0>

1. **Descrierea componentelor**

**Senzori:**

Senzorii vor include dispozitive specializate pentru măsurarea parametrilor fiziologici precum ECG, temperatură și puls. Aceștia vor trimite datele lor către placa de dezvoltare (Arduino/ESP8266) pentru prelucrare și transmitere către gateway.

**Placă de dezvoltare (Arduino/ESP8266):**

Această placă de dezvoltare va acționa ca o interfață între senzori și gateway-ul de comunicație. Va primi datele de la senzori și le va procesa pentru a le formata într-un format adecvat pentru transmiterea către gateway. De asemenea, va efectua prelucrări suplimentare ale datelor, cum ar fi filtrarea sau normalizarea acestora, înainte de a le transmite mai departe.

**Gateway (ESP8266):**

Gateway-ul, care este un ESP8266, va primi datele transmise de placa de dezvoltare și le va transmite către brokerul MQTT. Acesta este un punct de acces între dispozitivele fizice și infrastructura de comunicare bazată pe rețea, facilitând tranzacția datelor între cele două.

**Broker MQTT:**

Brokerul MQTT este componenta centrală de comunicație în cadrul sistemului IoT. Va primi datele publicate de gateway(ESP8266) și le va distribui către toți clienții abonați la topicurile respective. Datele publicate vor include informații precum parametrii fiziologici măsurați (ECG, temperatură, puls etc.), care vor fi transmise către clientul intermediar pentru ulterioara prelucrare.

**Client intermediar:**

Acest client intermediar este un alt serviciu care se abonează la topicurile MQTT relevante pentru a prelua datele publicate de brokerul MQTT. Scopul său este de a prelucra datele și de a le transmite către serviciul Cloud găzduit pe platforma Heroku. Acest pas adițional poate implica, de asemenea, filtrarea, normalizarea sau agregarea datelor înainte de trimiterea lor către serviciul Cloud.

**Serviciu Cloud (găzduit pe Heroku):**

Serviciul Cloud găzduit pe platforma Heroku va primi datele transmise de clientul intermediar și le va procesa pentru analiză ulterioară. Acesta va include algoritmi de detecție a anomaliilor pentru a identifica stările de sănătate anormale ale pacienților. De asemenea, acesta va genera notificări către aplicațiile mobile ale pacienților și ale medicilor pentru a alerta în cazul unor valori anormale.

## **Python și Django**

Python este un limbaj de programare versatil și popular, iar Django este un cadru web Python care facilitează dezvoltarea rapidă și curată a aplicațiilor web. În contextul acestei configurații, Python și Django sunt utilizate pentru a gestiona backend-ul aplicației web, inclusiv logica de afaceri și gestionarea cererilor HTTP.

**PostgreSQL**

PostgreSQL este un sistem de gestionare a bazelor de date relaționale open-source, puternic și sigur. Django oferă suport nativ pentru PostgreSQL, permițând dezvoltatorilor să utilizeze PostgreSQL ca și bază de date pentru aplicațiile lor Django.

1. **Descrierea comunicarii intre module**

Comunicarea între modulele din sistemul dat se va realiza prin intermediul protocoalelor și tehnologiilor specifice fiecărei componente. Iată cum vor comunica modulele între ele:

**Senzori → Placă de dezvoltare (Arduino/ESP8266):**

Senzorii vor trimite datele lor către placa de dezvoltare utilizând interfețe de comunicare specifice senzorului respectiv. Acest lucru poate fi realizat prin intermediul unor interfețe precum I2C, SPI, UART sau analogic, în funcție de tipul de senzor utilizat și de placa de dezvoltare conectată la acesta. Placa de dezvoltare va citi datele trimise de senzori și le va procesa folosind limbajul de programare specific (de exemplu, C/C++ pentru Arduino sau Arduino IDE pentru ESP8266).

**Placă de dezvoltare → Gateway (ESP8266):**

Placa de dezvoltare (Arduino/ESP8266) va transmite datele prelucrate și formatate către gateway-ul (ESP8266) utilizând un protocol de comunicare adecvat, cum ar fi UART, SPI sau I2C. Această comunicare va fi realizată printr-o conexiune fizică directă între placa de dezvoltare și gateway.

**Gateway (ESP8266) → Broker MQTT:**

Gateway-ul (ESP8266) va transmite datele către brokerul MQTT utilizând protocolul MQTT (MQ Telemetry Transport) peste o conexiune de rețea. Acest lucru presupune configurarea și utilizarea bibliotecilor și funcționalităților specifice MQTT disponibile pentru ESP8266, care permit publicarea datelor pe topicurile MQTT relevante către broker.

**Broker MQTT → Client intermediar:**

Brokerul MQTT va distribui datele publicate de gateway către clientul intermediar, care va fi un alt serviciu conectat la același broker MQTT. Această comunicare se va realiza prin intermediul abonării la topicurile relevante de către clientul intermediar, astfel încât să primească automat datele publicate pe aceste topicuri.

**Client intermediar → Serviciu Cloud (găzduit pe Heroku):**

Clientul intermediar va transmite datele către serviciul Cloud găzduit pe platforma Heroku, utilizând protocoale și tehnologii de comunicare convenționale pentru transmiterea datelor peste internet. Acest lucru va fi realizat prin intermediul cererilor HTTP, a protocolului WebSocket sau a altor protocoale de rețea, în funcție de cerințele specifice ale serviciului Cloud și de interfața de programare a aplicațiilor (API) disponibilă pentru integrare.

**Senzori și dispozitive Arduino / ESP8622:**

1. Arduino Core Library: Aceasta este biblioteca standard pentru programarea placelor Arduino și include funcționalități de bază pentru interacțiunea cu hardware-ul.

2. Adafruit Sensor Library: O bibliotecă populară pentru gestionarea diferitelor tipuri de senzori, cum ar fi senzorii de temperatură sau senzorii de puls. Aceasta oferă un mod simplificat de a comunica cu acești senzori.

**MQTT:**

1. PubSubClient (pentru Arduino): O bibliotecă populară pentru Arduino care facilitează comunicarea prin protocolul MQTT.

2. Arduino MQTT (pentru ESP8266 / ESP32): Această bibliotecă oferă suport pentru utilizarea protocolului MQTT pe platformele ESP8266 și ESP32.

**Client intermediar (scris în Python):**

1. Paho MQTT client library: O bibliotecă Python foarte utilizată pentru comunicarea prin protocolul MQTT. Aceasta vă permite să vă conectați la un broker MQTT și să trimiteți și să primiți mesaje.

2. Requests library: Bibliotecă Python simplă și ușor de utilizat pentru a efectua cereri HTTP către aplicația de cloud (Heroku).

3. NumPy și Pandas: Dacă aveți nevoie de analiză și prelucrare a datelor, NumPy și Pandas sunt biblioteci foarte puternice și populare în comunitatea Python.

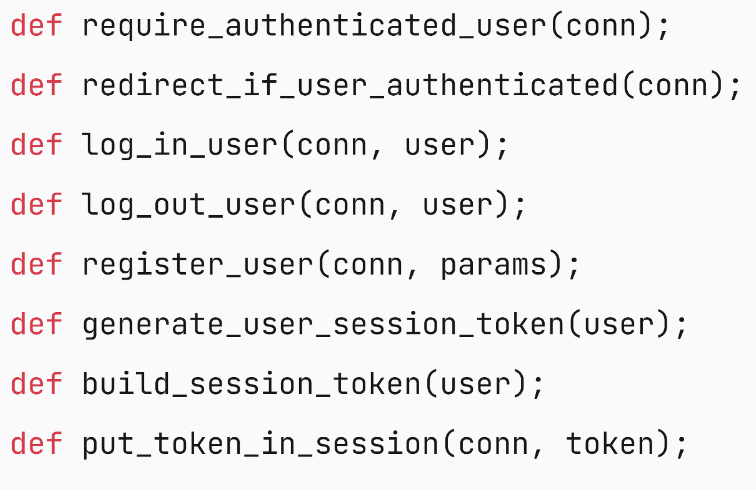
**Heroku (Cloud):**

1. Django: Framework-ur popular pentru dezvoltarea aplicațiilor web în Python pe platforma Heroku.

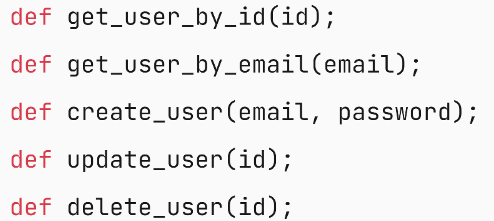
2. psycopg2: Un adaptor PostgreSQL pentru Python, utilizat pentru a conecta aplicațiile Python la baze de date PostgreSQL.

**Pentru comunicarea la nivel de APIs se vor folosi :**

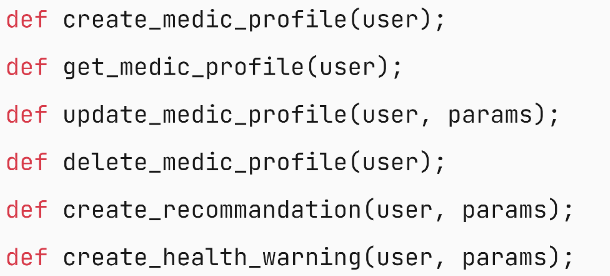
**Atentificator :**



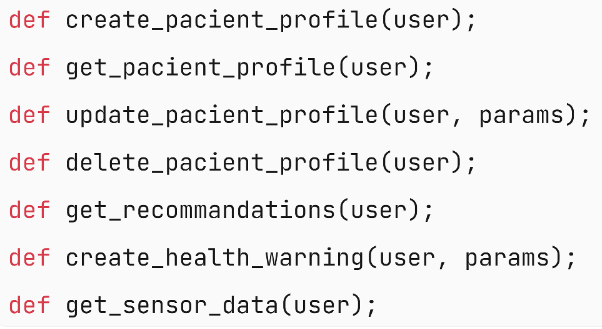
**Utilizator :**



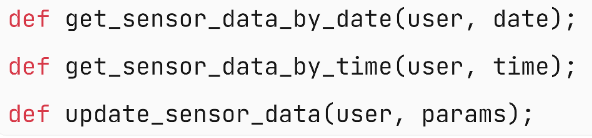
**Medic :**



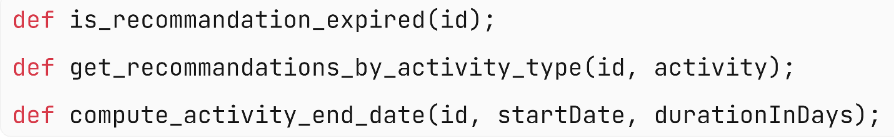
**Pacient :**



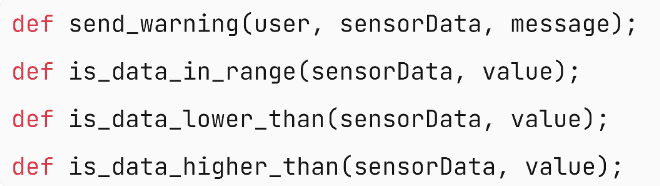
**Masuratoare :**



**Recomandare :**



**Avertizare:**



1. **Structuri de baze de date şi fişiere**

* **Fișiere de cod sursă pentru dispozitivele fizice (Arduino sau ESP)**:
  + Fișiere .ino sau .cpp care conțin codul sursă pentru citirea datelor de la senzori și transmiterea lor către serverul MQTT. Aceste fișiere vor conține logica de programare pentru conectarea la senzori, citirea datelor și comunicarea prin protocolul MQTT.
* **Fișiere de configurare pentru dispozitivele fizice**:
  + Fișiere de configurare sau setări pentru a specifica detalii precum adresa serverului MQTT, porturile de comunicare, detalii de rețea și altele.
* **Fișiere de cod pentru serverul MQTT**:
  + Fișierele de cod sursă pentru serverul MQTT, care pot fi în diverse limbaje de programare, cum ar fi Python, JavaScript, etc. Acestea vor conține logica de gestionare a comunicării între dispozitive și clientul intermediar, precum și logica pentru publicarea și abonarea la topicuri MQTT.
* **Fișiere de configurare pentru serverul MQTT**:
  + Fișiere de configurare care conțin setările serverului MQTT, cum ar fi setările de securitate, limitele de conexiune, configurarea topicurilor, etc.
* **Fișiere de cod pentru clientul intermediar**:
  + Fișierele de cod sursă pentru clientul intermediar care primește datele de la serverul MQTT și le trimite către cloud (Heroku). Acestea vor conține logica de prelucrare a datelor primite și de transmitere a acestora către cloud.
* **Fișiere de configurare pentru clientul intermediar**:
  + Fișiere de configurare pentru clientul intermediar, care pot include detalii despre configurarea serverului MQTT, autentificare, detalii de rețea și altele.
* **Fișiere de cod pentru aplicația de cloud (Heroku)**:
  + Fișierele de cod sursă pentru aplicația de cloud care primește datele de la clientul intermediar și le stoca într-o bază de date sau le prelucrează în alt mod. Acestea vor conține logica pentru gestionarea cererilor HTTP, interacțiunea cu baza de date și alte operațiuni necesare pentru prelucrarea datelor.
* **Fișiere de configurare pentru aplicația de cloud (Heroku)**:
  + Fișiere de configurare pentru aplicația de cloud, care pot include configurările serverului web, setările de securitate, configurările de bază de date și altele.

Datele transmise de catre senzori impreuna cu Arduino si cu ESP vor fi sub forma de fiserel JSON pentru a facilita transmiterea acestora cat mai eficienta, fisierele JSON au aceasta structura :

