Med4U
Bulevardul Vasile Pârvan 2, Timișoara

PARTEA II

PROIECTARE

SISTEM PURTABIL DE SUPRAVEGHERE A STĂRII DE SĂNĂTATE

Programator-șef: Pavel ROȘCA
Ajutor programator-șef: Andrei POPESC
Secretar: Claudia OROS
Programatori: Andreea RUS
Mălina PĂIUȘAN
Denis PETRUȘE
Daniel PAȘCU
Adrian POPESCU
Mihai BABALEAN

Cuprins

1. Arhitectura sistemului

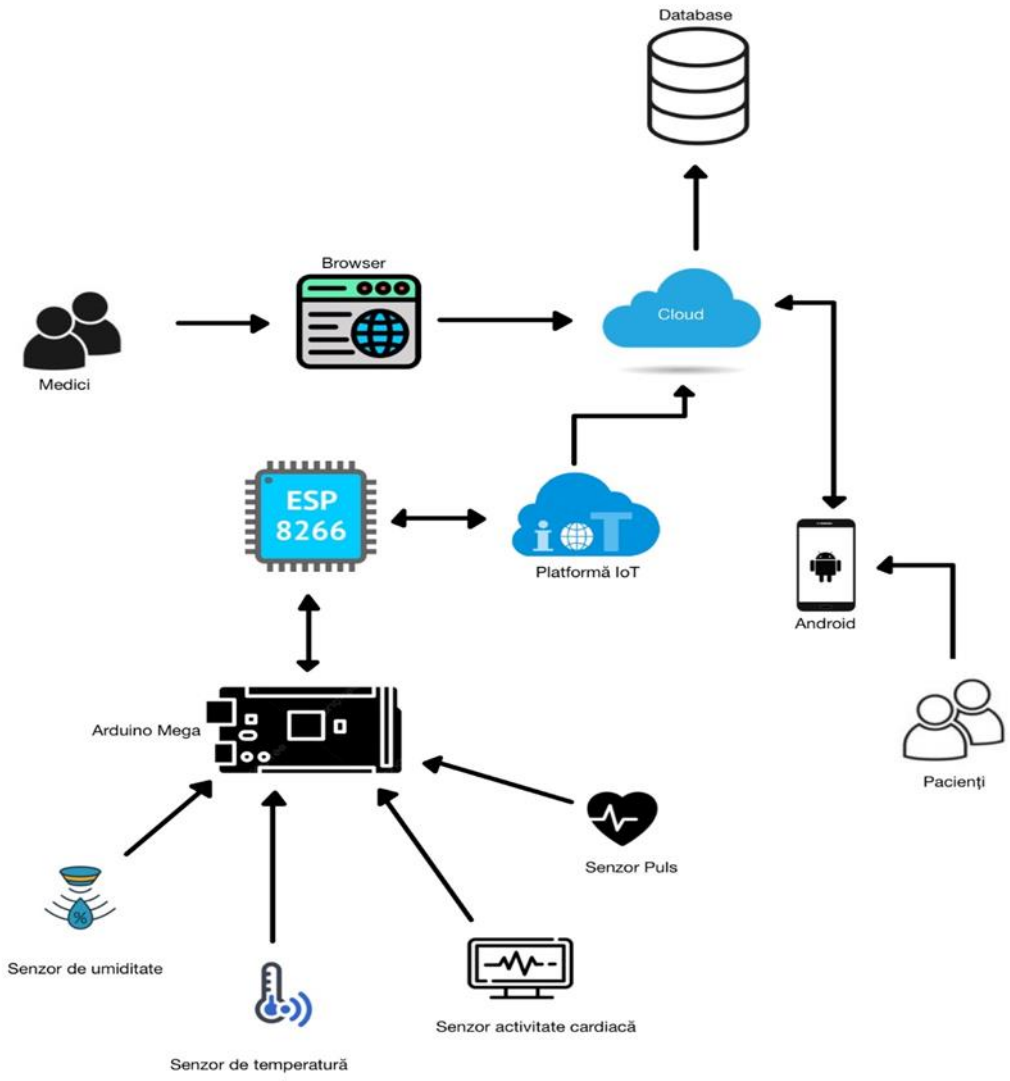


Figura 1. Arhitectura întregului sistem

În cadrul arhitecturii sistemului, fiecare modul reprezintă o componentă distinctă care îndeplinește anumite funcționalități și interacționează cu celelalte module pentru a realiza sarcinile asignate. Segmentarea fiecărui modul și refacerea legăturilor utilizând blocuri simbolice ajută la clarificarea modului în care aceste componente interacționează între ele în cadrul întregului sistem.

În continuare, vom explica fiecare modul în parte pentru a înțelege mai bine rolul și funcționalitățile acestora în cadrul sistemului:

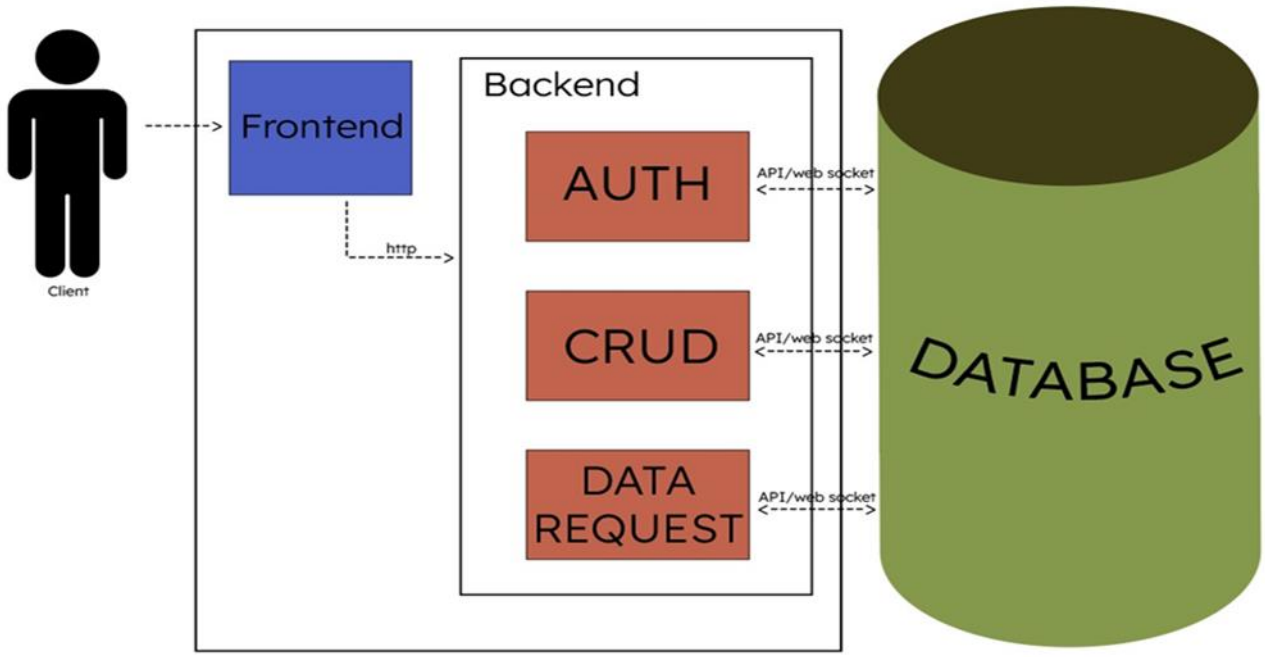


Figura 2. Arhitectura modulului web

Modulul Web/Mobile al aplicației este dezvoltat pe baza arhitecturii client-server, unde clientul reprezintă interfața utilizatorului și serverul gestionează logica și interacțiunea cu baza de date. Pentru dezvoltarea interfeței utilizatorului, se folosește framework-ul React.js, cunoscut pentru eficiența și flexibilitatea sa în construirea aplicațiilor web interactive. Pentru baza de date, se utilizează MongoDB, o bază de date NoSQL scalabilă și flexibilă, care este potrivită pentru stocarea și manipularea datelor într-un mediu cloud.

Comunicarea între client și server se realizează prin intermediul protocolului HTTP și API-urilor (Interfețe de Programare a Aplicațiilor), care permit transmiterea și primirea datelor între componentele modulului. De asemenea, se pot utiliza și websockets pentru comunicare bidirecțională în timp real între client și server.

Pentru dezvoltarea și testarea aplicației, se folosește un mediu de dezvoltare adecvat pentru tehnologiile utilizate, cum ar fi Node.js pentru partea de server, și un set de instrumente care facilitează dezvoltarea, cum ar fi Visual Studio Code.

Sistemul de operare pe care rulează aplicația poate varia în funcție de necesități, dar aplicația poate fi dezvoltată pentru a fi compatibilă cu diferite sisteme de operare, inclusiv Windows, Linux și macOS pentru aplicații desktop, sau iOS și Android pentru aplicații mobile.

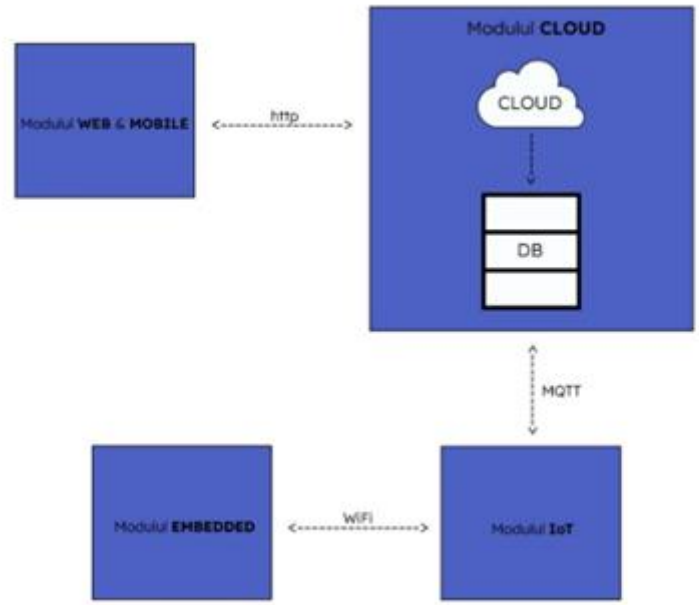


Figura 3. Arhitectura modulului Cloud

Modulul Cloud al aplicației are o arhitectură care implică interacțiunea cu alte module, cum ar fi Modulul Web/Mobile și Modulul IoT. În această arhitectură, modulul Cloud servește ca o infrastructură centralizată pentru prelucrarea și stocarea datelor provenite din diverse surse.

Pentru comunicarea cu Modulul Web/Mobile, modulul Cloud folosește protocolul HTTP, care permite transferul de date între client și server. Acest lucru facilitează interacțiunea utilizatorului cu aplicația prin intermediul interfeței web sau mobile.

În ceea ce privește comunicarea cu Modulul IoT, modulul Cloud utilizează protocolul MQTT (Message Queuing Telemetry Transport), care este ideal pentru transmiterea de date între dispozitive IoT și server. Acest protocol asigură o comunicație eficientă și scalabilă între dispozitivele IoT și modulul Cloud.

Modulul Cloud utilizează MongoDB ca și bază de date, oferind o soluție scalabilă și flexibilă pentru stocarea și gestionarea datelor. MongoDB este o bază de date NoSQL care permite stocarea datelor sub formă de documente JSON, ceea ce se potrivește bine cu datele semistructurate sau nestructurate provenite din diverse surse, inclusiv dispozitive IoT.

Pentru a facilita integrarea datelor provenite din dispozitivele IoT în baza de date MongoDB, modulul Cloud utilizează un plugin sau o componentă de transformare a datelor care convertește datele din formatul specific dispozitivelor IoT în format JSON și le inserează în baza de date.

În ceea ce privește interacțiunea cu Modulul Embedded, modulul Cloud nu este direct implicat în comunicarea cu dispozitivele embedded. Comunicarea dintre Modulul IoT și Modulul Embedded se realizează prin intermediul rețelei locale, cu ajutorul tehnologiei Wi-Fi, iar această comunicare nu necesită intervenția directă a modulului Cloud.

Arhitectura generală a modulului Cloud este orientată spre centralizarea și gestionarea datelor provenite din diverse surse, asigurând interoperabilitatea și scalabilitatea întregii aplicații .

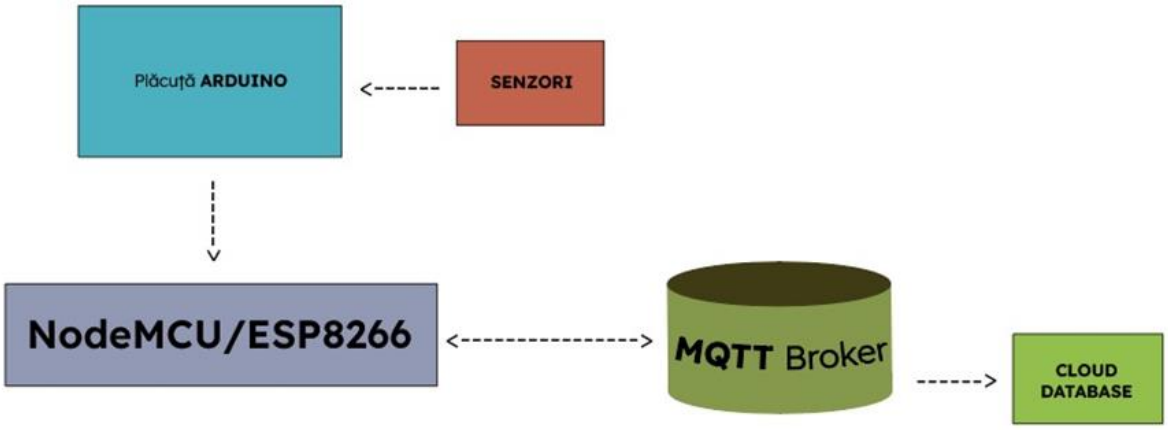


Figura 4. Arhitectura modulului IoT

Modulul IoT are rolul de a colecta datele de la diverse dispozitive și de a le transmite către alte componente ale sistemului. În această arhitectură, modulul IoT comunică cu Modulul Embedded prin intermediul unei conexiuni Wi-Fi, utilizând tehnologia ESP8266 pentru a asigura conectivitatea.

Odată ce datele sunt colectate de către Modulul Embedded, acestea sunt trimise către Modulul Cloud prin intermediul protocolului MQTT (Message Queuing Telemetry Transport). Modulul Cloud este configurat cu platforma Mosquitto de la Eclipse pentru a gestiona mesajele MQTT și pentru a le recepționa.

Pentru a asigura interoperabilitatea și compatibilitatea datelor, Modulul Cloud folosește un plugin care convertește datele primite în formatul JSON și le transmite către baza de date pentru stocare și prelucrare ulterioară.

În ceea ce privește tehnologiile folosite, Modulul IoT se bazează pe ESP8266 pentru comunicare, iar Modulul Cloud utilizează platforma Mosquitto pentru gestionarea mesajelor MQTT și MongoDB ca și bază de date.

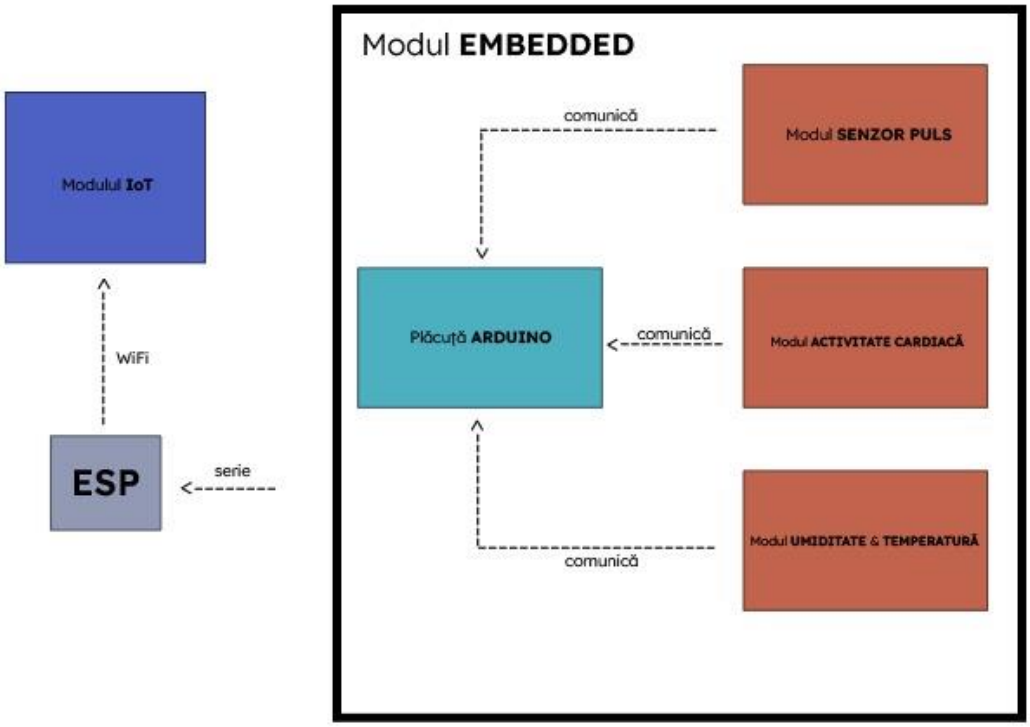


Figura 5. Arhitectura modulului embedded

În cadrul Modulului Embedded, avem o configurație complexă ce implică colectarea datelor de la trei tipuri diferite de senzori: un senzor de puls, un monitor pentru activitatea cardiacă și un senzor de temperatură și umiditate.

1. Senzorul de puls și monitorul pentru activitatea cardiacă sunt conectate la o placă Arduino Mega folosind cabluri și electrozi medicali. Aceste dispozitive sunt responsabile pentru măsurarea ritmului cardiac al utilizatorului și pentru detectarea activității cardiace. Datele colectate de aceste senzori sunt transmise sub formă de semnale electrice către Arduino Mega.

2. Senzorul de temperatură și umiditate HTU21D este conectat, de asemenea, la placă Arduino Mega. Acest senzor măsoară temperatura și umiditatea ambientală și transmite datele colectate către Arduino Mega.

3. Arduino Mega este utilizat ca și mediu de procesare și agregare a datelor provenite de la senzori. Acesta este programat folosind mediu de dezvoltare Arduino IDE pentru a citi datele de la senzori și a le procesa înainte de a le transmite mai departe.

4. ESP8266 este un modul Wi-Fi care este conectat la Arduino Mega și este utilizat pentru a transmite datele către Modulul IoT printr-o conexiune Wi-Fi. ESP8266 preia datele procesate de Arduino Mega și le trimite către Modulul IoT folosind un protocol de comunicație specific, cum ar fi MQTT.

Mediul de dezvoltare utilizat pentru programarea placii Arduino Mega este Arduino IDE, o platformă software open-source care facilitează dezvoltarea și încărcarea de cod pe plăcile Arduino. Aceasta oferă un set de instrumente și biblioteci care permit programatorilor să scrie și să încarce cod pe plăci Arduino într-un mod ușor și eficient.

Arhitectura Modulului Embedded este proiectată pentru a colecta date de la diverse senzori și pentru a le transmite către Modulul IoT pentru prelucrare și analiză ulterioară. Această configurație permite monitorizarea continuă a parametrilor fiziologici și ambientali, furnizând informații valoroase pentru analiza și diagnosticarea stării de sănătate a utilizatorului.

2. Descrierea componentelor (modulelor)

1. Modulul WEB App:

- Scopul/modulele asociate: Permite medicilor și pacienților să acceseze și să interacționeze cu datele de sănătate.
- Tehnologie utilizată: HTML, CSS, JavaScript, framework-uri web (de exemplu, React, Angular, Vue.js).

Intrări:

- Cereri HTTP de la utilizatori (medici și pacienți)
- Datele demografice și medicale ale pacienților
- Recomandările medicilor
- Datele măsurate de la senzorii purtabili (cloud)

Prelucrări:

- Validarea și autentificarea utilizatorilor
- Procesarea și afișarea datelor demografice și medicale ale pacienților
- Generarea și afișarea graficelor pentru evoluția parametrilor fiziologici
- Interpretarea și afișarea recomandărilor medicale
- Detectarea și gestionarea alarmelor/avertizărilor bazate pe valorile măsurate

Ieșiri:

- Interfața grafică a aplicației web, care permite utilizatorilor să vizualizeze și să interacționeze cu datele de sănătate
- Graficele și rapoartele generate în funcție de datele introduse și măsurate
- Recomandările medicale afișate pentru pacienți

Interacțiuni cu alte module:

Comunicarea cu componenta Cloud pentru a accesa și actualiza datele pacienților

Reprezentări UML:

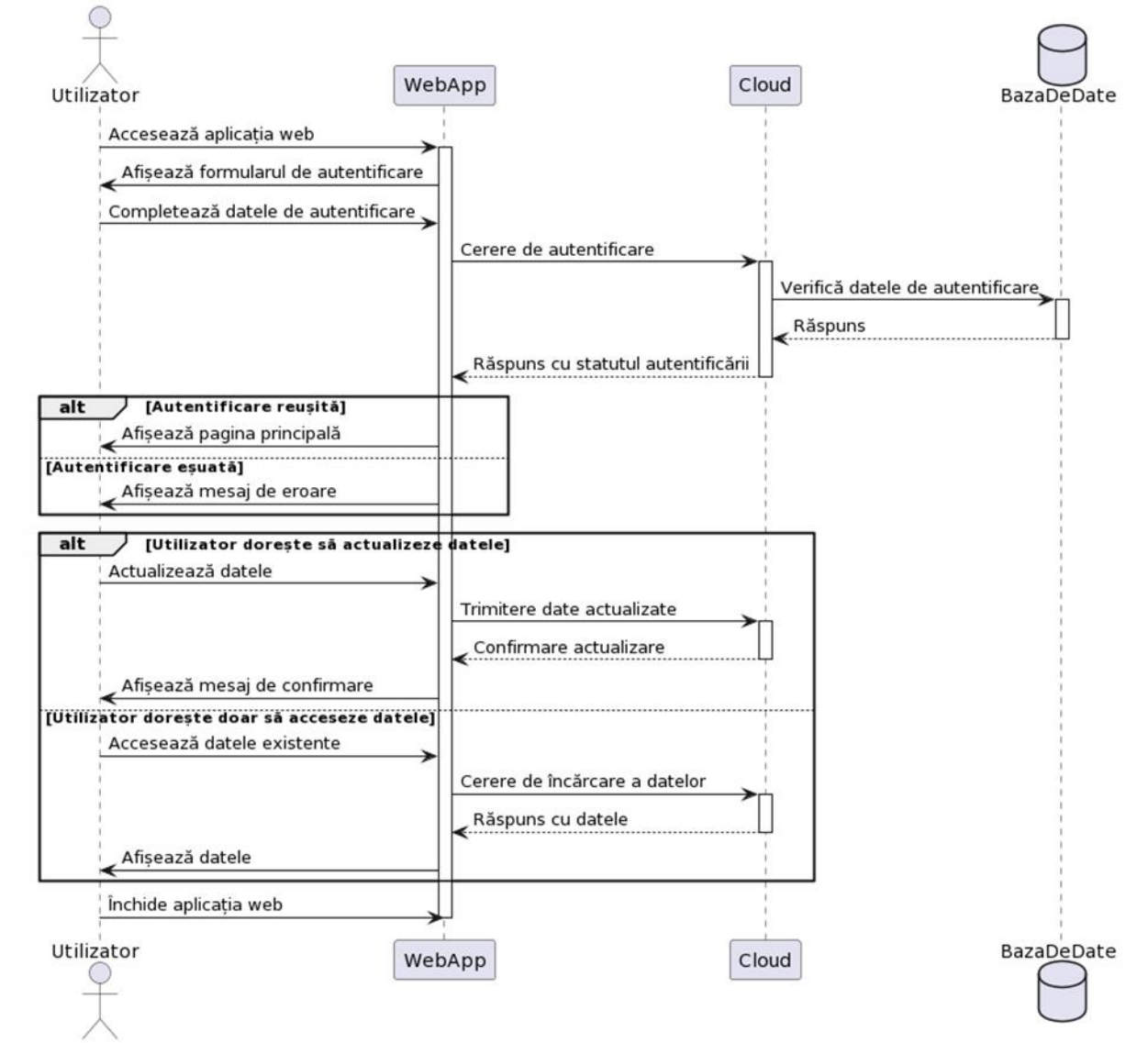


Figura 6. Comunicarea modulului WEB/Mobile cu modulul Cloud

2. Modulul Cloud

- Scopul/modulele asociate: Gestionează datele și interacțiunile între diversele componente ale sistemului, asigurând stocarea, accesarea și prelucrarea acestora cu ajutorul bazei de date MongoDB.
- Tehnologie utilizată: MongoDB pentru stocarea datelor

Intrări:

- Cereri HTTP de la aplicația web pentru autentificare și comunicare
- Datele pacienților și recomandările medicilor
- Informații primite de la platforma IoT și aplicația mobilă

Prelucrări:

- Validarea și procesarea cererilor de autentificare
- Stocarea și accesarea datelor pacienților și recomandărilor medicale în baza de date MongoDB
- Recepționarea și stocarea datelor de la platforma IoT și aplicația mobilă în MongoDB
- Analiza datelor pentru determinarea condițiilor de alarmă și construirea avertizărilor corespunzătoare

Ieșiri:

- Confirmări sau erori de autentificare
- Acces la datele pacienților și recomandările medicale pentru aplicația web și aplicația mobilă
- Alarme și avertizări generate pentru aplicația mobilă

Interacțiuni cu alte module:

- Comunicarea cu aplicația web pentru autentificare și accesarea datelor
- Primirea și stocarea datelor de la platforma IoT și aplicația mobilă în MongoDB
- Transmiterea recomandărilor și avertizărilor către aplicația mobilă

Reprezentări UML:

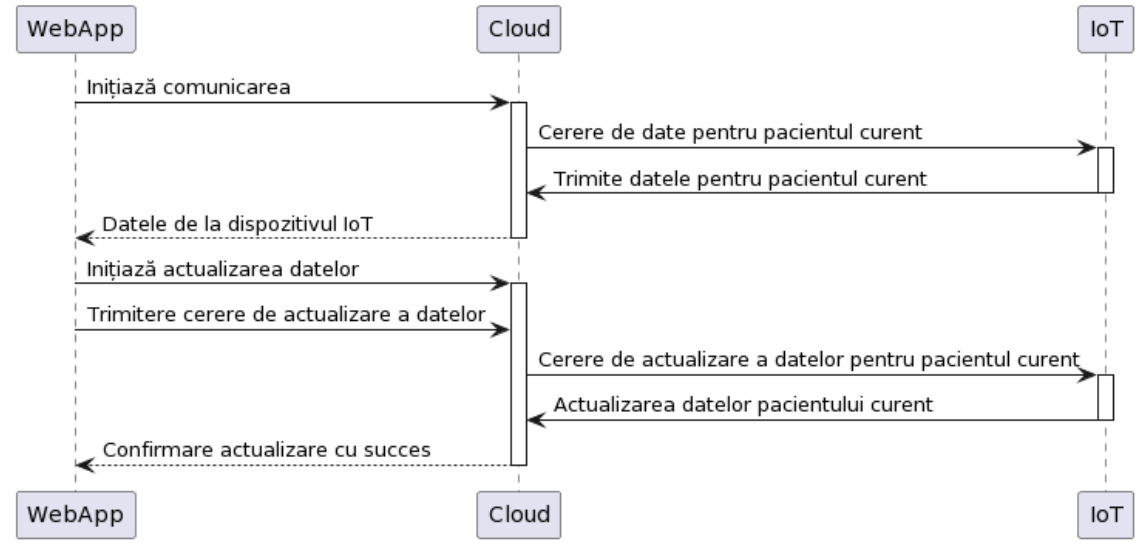


Figura 7. Comunicarea între modulele Web, Cloud și IoT

3. Modulul IoT:

- Scopul/modulele asociate: Gestionează preluarea și transmiterea datelor de la senzorii dispozitivelor IoT către Componenta Cloud, utilizând protocolul MQTT prin intermediul serverului Mosquitto și serviciul de automatizare Node-RED.

- Tehnologie utilizată: Mosquitto pentru implementarea protocolului MQTT, Node-RED pentru automatizare.

Intrări:

- Datele primite de la senzorii dispozitivelor IoT (temperatură, puls, ECG) conectați la ESP8266.
- Comenzi primite de la Componenta Cloud pentru activarea/dezactivarea senzorilor sau alte acțiuni de control.

Prelucrări:

- Transmiterea datelor către Componenta Cloud utilizând protocolul MQTT prin intermediul serverului Mosquitto.
- Preluarea datelor de la senzorii conectați la ESP8266.
- Procesarea comenzilor primite de la Componenta Cloud pentru controlul dispozitivelor IoT sau alte acțiuni de automatizare utilizând Node-RED.

Ieșiri:

- Datele preluate de la senzorii dispozitivelor IoT, transmise către Componenta Cloud prin MQTT.
- Confirmări sau răspunsuri la comenzile primite de la Componenta Cloud.

Interactiuni cu alte module:

- Preluarea comenzilor de la Componenta Cloud și transmiterea datelor către aceasta.
- Utilizarea serverului Mosquitto pentru a facilita comunicarea prin protocolul MQTT între modulul IoT și Componenta Cloud.
- Utilizarea serviciului Node-RED pentru a gestiona automatizarea și prelucrarea datelor primite de la senzorii dispozitivelor IoT.

Reprezentări UML:

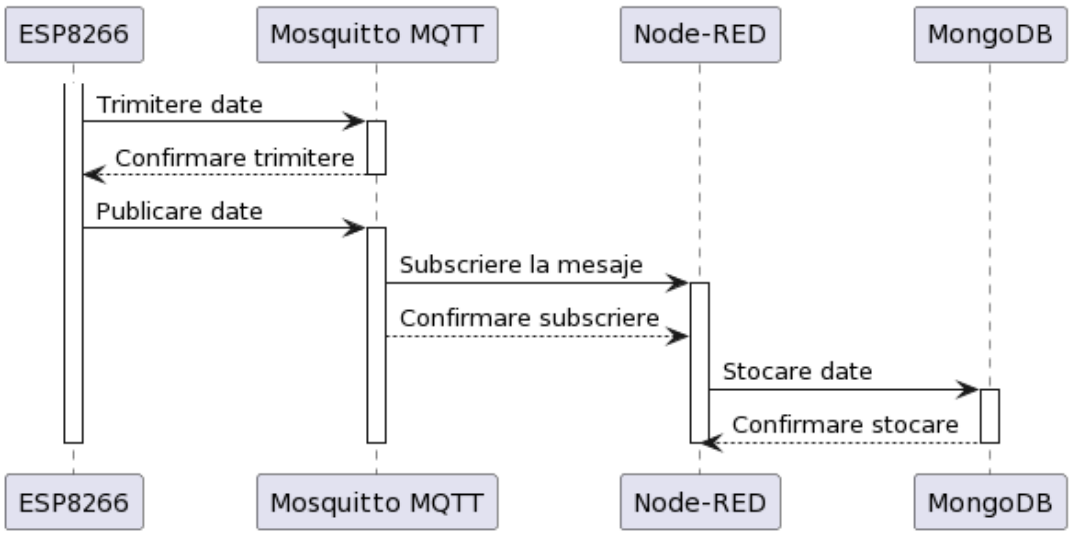


Figura 8. Comunicarea modulului IoT cu modulele Embedded și Cloud

4. Modulul Embedded (Wearable):

- Scopul/modulele asociate: Realizează colectarea și prelucrarea datelor de la diferite tipuri de senzori (senzor de puls, monitor pentru activitatea cardiacă și senzor de temperatură și umiditate), integrându-le în sistemul IoT și pregătindu-le pentru transmiterea către Componenta IoT.
- Tehnologie utilizată: Arduino Mega pentru procesarea și agregarea datelor, ESP8266 pentru conectivitatea Wi-Fi.

Intrări:

- Semnale provenite de la senzorul de puls și monitorul pentru activitatea cardiacă.
- Datele de temperatură de la senzorul HTU21D, conectat la placa Arduino Mega.
- Comenzi primite de la Componenta IoT pentru activarea/dezactivarea senzorilor sau alte acțiuni de control.

Prelucrări:

- Transmiterea datelor către Componenta Cloud utilizând protocolul MQTT prin intermediul serverului Mosquitto.
- Preluarea datelor de la senzorii conectați la ESP8266.
- Procesarea comenzilor primite de la Componenta Cloud pentru controlul dispozitivelor IoT sau alte acțiuni de automatizare utilizând Node-RED.

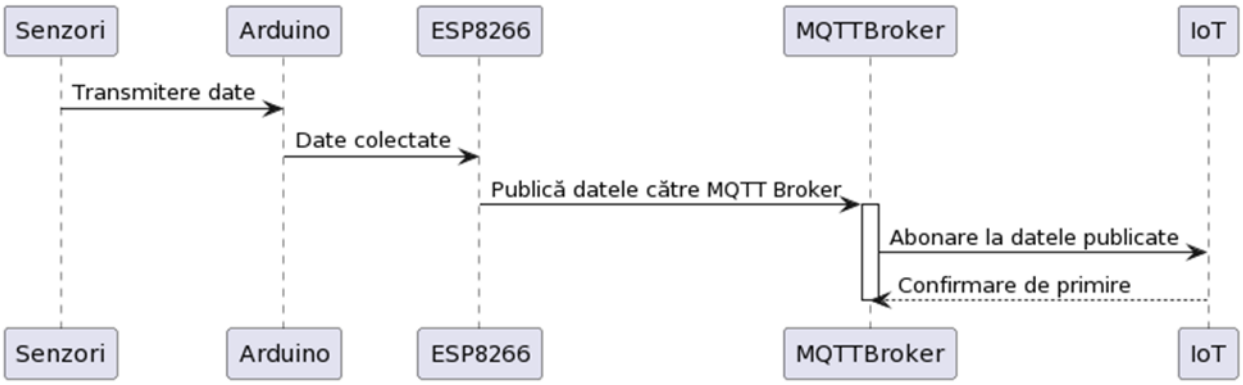
Ieșiri:

- Datele preluate de la senzorii dispozitivelor IoT, transmise către Componenta Cloud prin MQTT.
- Confirmări sau răspunsuri la comenzile primite de la Componenta Cloud.

Interactiuni cu alte module:

- Preluarea comenzilor de la Componenta Cloud și transmiterea datelor către aceasta.
- Utilizarea serverului Mosquitto pentru a facilita comunicarea prin protocolul MQTT între modulul IoT și Componenta Cloud.

Reprezentări UML:



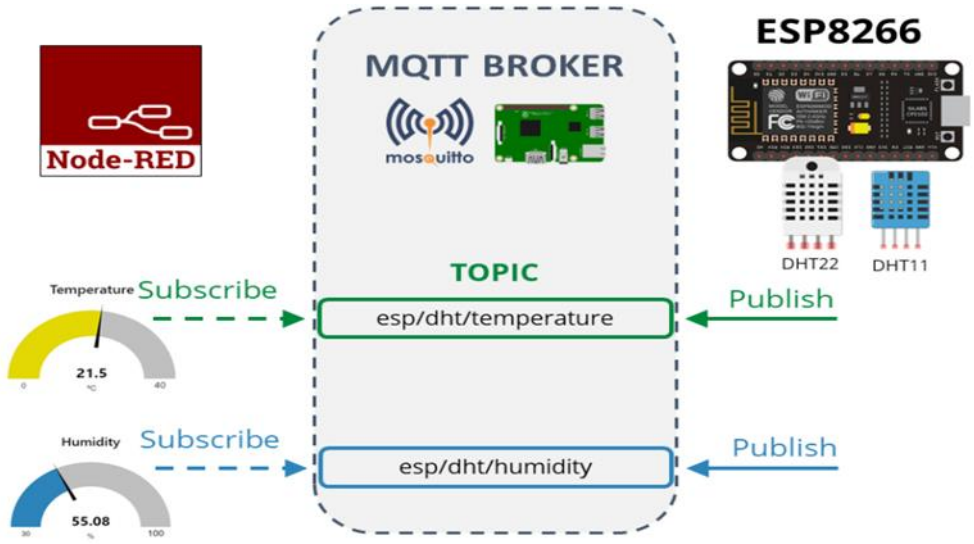


Figura 9,10 Comunicarea modulului Embedded cu modulul IoT

3. Descrierea comunicării între module

3.1 Comunicarea între Modulul Web/Mobil și Modulul Cloud:

Modul Sursă	Modul Destinație	Metoda de Comunicare	Parametri Implicați	Semnificații Parametri	Restricții
Web/Mobil	Cloud	Cereri HTTP POST	Date JSON, Token de Autentificare	Informații despre starea de sănătate, Token de Autorizare	Utilizarea unui protocol sigur pentru transmitere

1. Cereri HTTP POST:

Cererile HTTP POST sunt un tip de cereri pe care clientul le trimite către server pentru a trimite date către acesta. Aceste cereri sunt utilizate pentru a trimite date către un server web pentru a fi procesate. În contextul nostru, modulul web/mobile ar trimite cereri HTTP POST către modulul cloud pentru a trimite date despre starea de sănătate a pacienților sau alte informații relevante.

2. Date JSON:

JSON (JavaScript Object Notation) este un format ușor de citit și de scris pentru schimbul de date între un client și un server. Acesta este adesea folosit în cererile HTTP pentru a structura datele trimise. În cazul nostru, datele despre starea de sănătate ar fi

împachetate în format JSON și trimise împreună cu cererile HTTP POST de la modulul web/mobile către modulul cloud.

3. **Token de Autentificare și Token de Autorizare:**

Acestea sunt identificatori unici care sunt utilizate pentru a autentifica și autoriza cererile efectuate între modulele sistemului. Token-ul de autentificare este folosit pentru a confirma identitatea clientului care face cererea, în timp ce token-ul de autorizare este folosit pentru a verifica dacă clientul are permisiunea de a accesa anumite resurse sau date. Acestea sunt esențiale pentru a asigura securitatea și confidențialitatea datelor în timpul comunicării între module.

4. **Utilizarea unui protocol sigur pentru transmitere**:

Aceasta se referă la utilizarea unui protocol de comunicație care oferă o conexiune sigură și criptată între module pentru a proteja datele împotriva interceptării și manipulării de către terți rău intenționați. De obicei, HTTPS (HTTP Secure) este folosit pentru a asigura securitatea comunicațiilor HTTP, oferind autentificare și criptare între client și server.

Comunicarea dintre modulul web/mobile și modulul cloud se realizează prin intermediul cererilor HTTP POST. Datele sunt transmise sub formă de obiecte JSON, care conțin informații despre starea de sănătate și un token de autorizare pentru autentificare.

Protocolul utilizat pentru transmiterea acestor date este HTTP, iar nu HTTPS, astfel că datele sunt trimise în text clar, fără criptare. Totuși, securitatea comunicației este asigurată de utilizarea unui token de autorizare pentru autentificarea cererilor. Acest token furnizează accesul controlat la serviciile cloud, iar utilizarea lui este esențială pentru a valida și autoriza cererile provenite de la modulul web/mobile.

3.2 Comunicarea între Modulul Cloud și Modulul IoT:

Modul Sursă	Modul Destinație	Metoda de Comunicare	Parametri Implicați	Semnificații Parametri	Restricții
Cloud	IoT	Protocol MQTT	Subiecte MQTT, Date de la Senzori	Organizarea mesajelor, Date de Mediu	Asigurarea unei livrări fiabile a mesajelor

1. **Protocol MQTT**:
   - Protocolul MQTT (Message Queuing Telemetry Transport) este utilizat pentru a facilita comunicarea între Modulul Cloud și Modulul IoT.

- Acest protocol asigură transmiterea eficientă și scalabilă a datelor între dispozitivele IoT și serverul cloud.

2. **Subiecte MQTT și Date de la Senzori**:
   - Subiectele MQTT reprezintă canalele prin care mesajele sunt publicate și abonate între Modulul Cloud și Modulul IoT.
   - Datele de la senzori, cum ar fi temperatură, umiditate etc., sunt împachetate în aceste mesaje și trimise prin intermediul subiectelor MQTT.

3. **Organizarea Mesajelor**:
   - Organizarea mesajelor se referă la structura și formatul în care sunt transmise datele prin intermediul protocolului MQTT.
   - Este esențial să se stabilească un format standard pentru mesajele MQTT pentru a asigura o comunicare eficientă și interpretarea corectă a datelor de către Modulul Cloud și Modulul IoT.

4. **Asigurarea unei Livrări Fiabile a Mesajelor**:
   - Asigurarea unei livrări fiabile a mesajelor este un aspect crucial în comunicarea între Modulul Cloud și Modulul IoT.
   - Protocolul MQTT include mecanisme de confirmare a livrării și retransmiterea automată a mesajelor pentru a asigura că datele sunt livrate cu succes și în condiții de conexiune instabilă sau eșecuri temporare.

Comunicarea între Modulul Cloud și Modulul IoT se realizează utilizând protocolul MQTT. Mesajele sunt transmise pe subiecte MQTT și includ date provenite de la senzorii dispozitivelor IoT. Organizarea mesajelor este esențială pentru a asigura o transmitere eficientă și ordonată a datelor.

Subiectele MQTT servesc drept canale de comunicare prin intermediul cărora mesajele sunt publicate și consumate de către diferitele componente ale sistemului.

Aceste subiecte sunt concepute pentru a organiza mesajele într-un mod care să reflecte structura și fluxul informației în cadrul sistemului.

Un aspect crucial al comunicării prin MQTT este asigurarea unei livrări fiabile a mesajelor. Acest lucru se realizează prin implementarea unor mecanisme de confirmare a livrării și retransmitere a mesajelor în cazul în care acestea nu sunt recepționate cu succes de către destinatar. Astfel, se garantează că datele sunt transmise și procesate corect între

Modulul Cloud și Modulul IoT, chiar și în condiții de instabilitate sau eșecuri temporare de conexiune.

3.3 Comunicarea între Modulul Embedded și Modulul IoT:

Modul Sursă	Modul Destinație	Metoda de Comunicare	Parametri Implicați	Semnificații Parametri	Restricții
Embedded	IoT	Protocol Wi-Fi (ESP8266)	Date de la Senzori	Date de Mediu	Fiabilitatea conexiunii Wi-Fi

1. **Protocol Wi-Fi (ESP8266)**:
   - Modulul Embedded utilizează protocolul Wi-Fi, implementat de către componenta hardware ESP8266, pentru a comunica cu Modulul IoT.
   - ESP8266 este un modul Wi-Fi care oferă capacitatea de a stabili și menține conexiuni Wi-Fi pentru transmiterea datelor între dispozitive.

2. **Date de la Senzori**:
   - Modulul Embedded primește date de la diferiți senzori, cum ar fi senzorul de puls, monitorul pentru activitatea cardiacă și senzorul de temperatură și umiditate.
   - Aceste date sunt colectate și procesate de către Modulul Embedded înainte de a fi transmise către Modulul IoT pentru prelucrare și stocare ulterioară.

3. **Date de Mediu**:
   - Termenul "Date de Mediu" se referă la informațiile colectate de la senzori care reflectă starea mediului înconjurător, cum ar fi temperatura și umiditatea ambientală.
   - Aceste date sunt esențiale pentru monitorizarea condițiilor de mediu și pot fi utilizate pentru a lua decizii informate în ceea ce privește sănătatea și siguranța utilizatorului.

4. **Fiabilitatea Conexiunii Wi-Fi**:
   - Fiabilitatea conexiunii Wi-Fi este un aspect crucial în comunicarea între Modulul Embedded și Modulul IoT.
   - ESP8266 trebuie să mențină o conexiune Wi-Fi stabilă și fiabilă pentru a asigura transmiterea corectă și completă a datelor către Modulul IoT.
   - Este important ca Modulul Embedded să gestioneze în mod adecvat situațiile în care conexiunea Wi-Fi este instabilă sau pierdută, pentru a preveni pierderea datelor sau funcționarea incorectă a sistemului.

Comunicarea între Modulul Embedded și Modulul IoT se realizează prin intermediul protocolului Wi-Fi, cu ajutorul componentei hardware ESP8266. Modulul Embedded primește date de la diferiți senzori și le transmite către Modulul IoT prin conexiunea Wi-Fi. Fiabilitatea conexiunii Wi-Fi este crucială pentru asigurarea unei transmiteri corecte și complete a datelor, iar ESP8266 trebuie să mențină o conexiune stabilă și fiabilă pentru a facilita comunicarea între module.

Datele trimise de la Modulul Embedded la Modulul IoT includ informații despre starea mediului, cum ar fi temperatura și umiditatea ambientală, colectate de la senzori. Este esențial ca Modulul Embedded să gestioneze în mod adecvat situațiile în care conexiunea Wi-Fi este instabilă sau pierdută, pentru a evita pierderea datelor sau funcționarea incorectă a sistemului.

Prin asigurarea unei comunicări eficiente și fiabile între Modulul Embedded și Modulul IoT, se poate garanta colectarea și prelucrarea corectă a datelor de mediu și a informațiilor de sănătate în cadrul sistemului, contribuind la funcționarea corectă și fiabilă a întregului sistem.

4. Structuri de baze de date și fișiere

1. Tabela utilizatori

Conține detalii legate de autentificarea și gestionarea utilizatorilor din cadrul sistemului nostru. Aceasta conține informații generale despre toți utilizatorii, inclusiv administratorii, medicii și pacienții și este utilizată pentru a controla accesul și permisiunile în sistem.

Cuprinde următoarele câmpuri:

- **IdAdmin (PK)** - un identificator unic pentru fiecare administrator, de tip int, dimensiune 15 și este cheie primară;
- **Nume** - numele administratorului, de tip varchar, dimensiune 30;
- **Prenume** - prenumele administratului, de tip varchar, dimensiune 30
- **Adresă_de_email** - adresa de email a administratorului, de tip varchar, dimensiune 50.

2. Tabela Pacienți

Conține detalii personale ale pacienților și are următoarele câmpuri:

- **IdPacient (PK)** - un identificator unic pentru fiecare pacient, de tip int, dimensiune 15, cheie primară;
- **Nume** - numele pacientului, de tip varchar, dimensiune 30;
- **Prenume** - prenumele pacientului, de tip varchar, dimensiune 30;

- **CNP** - Codul Numeric Personal al pacientului, de tip char, dimensiune 13;
- **Adresă** - adresa pacientului, de tip varchar, dimensiune 20;
- **Număr_de_telefon** - numărul de telefon al pacientului, de tip char, dimensiune 10;
- **Adresă_de_email** - adresa de email a pacientului, de tip varchar, dimensiune 50;
- **Profesie** - profesia pacientului, de tip varchar, dimensiune 15;
- **Loc_de_muncă** - locul de muncă al pacientului, de tip varchar, dimensiune 15;
- **Alte_informații** - alte informații medicale relevante despre pacient, de tip text, dimensiune 255.

3. Tabela Medici

Conține informații despre medici, inclusiv nume, prenume, specializare, număr de telefon și adresă de email. Aceasta are următoarele câmpuri:

- **IdMedic (PK)** - un identificator unic pentru fiecare medic, de tip int, dimensiune 15, cheie primară;
- **Nume** - numele medicului, de tip varchar, dimensiune 30;
- **Prenume** - prenumele medicului, de tip varchar, dimensiune 30;
- **Specializare** - specializarea medicului, de tip varchar, dimensiune 15;
- **Număr_de_telefon** - numărul de telefon al medicului, de tip char, dimensiune 10;
- **Adresă_de_email** - adresa de email a medicului, de tip varchar, dimensiune 50.

4. Tabela DateFiziologice

Stochează datele fiziologice măsurate ale pacienților, precum pulsul, temperatura, ECG-ul și umiditatea, împreună cu data și timpul citirii. Aceasta cuprinde următoarele cămpuri:

- **IdMăsurare (PK)** - un identificator unic pentru fiecare măsurare, de tip int, dimensiune 15, cheie primară;
- **Puls** - pulsul pacientului, de tip int, dimensiune 3;

- **Temperatură** - temperatura pacientului în grade Celsius, de tip decimal, dimensiune (5,2);
- **ECG** – electrocardiograma / ritmul cardiac al pacientului reprezentat de mai multe categorii, de tip varchar, dimensiune 30;
- **Umiditate** – umiditatea mediului înconjurător al pacientului reprezentat printr-un procent, de tip decimal, dimensiune (5,2);
- **Data_și_timpul_citirii** – data și timpul citirii parametrilor fiziologici, de tip datetime.

5. Tabela RecomandăriMedicale

Păstrează recomandările medicale date de medici pacienților, inclusiv tipul recomandării, descrierea, durata și alte informații relevante. Ea cuprinde câmpurile următoare:

- **IdRecomandare (PK)** - un identificator unic pentru fiecare recomandare, de tip int, dimensiune 15, cheie primară,
- **IdPacient (FK)** - cheie externă către tabela "Pacienți", indicând pacientul asociat acestei recomandări, de tip int, dimensiune 15;
- **IdMedic (FK)** - cheie externă către tabela "Medici", indicând medicul care a emis această recomandare, de tip int, dimensiune 15;
- **TipRecomandare:** tipul de recomandare emisă, de tip varchar, dimensiune 30;
- **DescriereRecomandare** - descrierea detaliată a recomandării medicale, de tip text, dimensiune 1024;
- **DurataRecomandare** - durata pentru care este valabilă recomandarea în minute, de tip int, dimensiune 3;
- **Alte_informații** - orice alte informații relevante despre recomandare, de tip text, dimensiune 255.

6.Tabela Alarme

Utilizat pentru a înregistra alarmele generate de sistem, inclusiv tipul alarmei, descrierea, data și timpul activării. Ea conține următoarele câmpuri:

- **IdAlarmă (PK)** - Un identificator unic pentru fiecare alarmă, de tip int, dimensiune 15, cheie primară;

- **IdPacient (FK)** - cheie externă către tabela "Pacienți", indicând pacientul asociat acestei alarme, de tip int, dimensiune 15;
- **TipAlarmă** - tipul de alarmă generată, cum ar fi "puls prea mare", "temperatură ridicată" etc., de tip varchar, dimensiune 30;
- **DescriereAlarmă** - descrierea detaliată a alarmei sau a situației care a generat-o, de tip text, dimensiune 1024;
- **Data_și_timpul_activării** - momentul exact când a fost activată alarma, de tip datetime.

7. Tabela Wearable

Este destinată stocării informațiilor transmise de dispozitivele purtabile prin platforma IoT. Aceasta va conține detalii despre datele fiziologice măsurate de dispozitivul purtabil, precum și informații despre pacienții asociați. Ea conține câmpurile următoare:

- **ID_Wearable (PK)** - un identificator unic pentru fiecare dispozitiv purtabil, de tip int, dimensiune 15, cheie primară;
- **ID_Pacient (FK)** - cheie externă către tabela "Pacienți", indicând pacientul asociat dispozitivului purtabil, de tip int, dimensiune 15;
- **Data_Și_Timpul_Măsurării** - momentul exact când au fost efectuate măsurările fiziologice, de tip datetime;
- **Valoarea_Măsurată** - valoarea măsurată de către dispozitivul purtabil (de exemplu, pulsul, temperatura, etc), de tip decimal, dimensiune (5,2);
- **Tipul_Măsurării** - tipul măsurării efectuate de către dispozitivul purtabil (de exemplu, ECG, temperatură, puls, etc).

Relațiile dintre Tabele:

1. Tabela Utilizatori și Tabela Pacienți:

Aceste tabele au o relație one-to-one sau one-to-zero (dacă un utilizator nu este neapărat și un pacient). Relația este realizată prin intermediul cheii primare-externe IdPacient din Tabela de Utilizatori către cheia primară IdPacient din Tabela de Pacienți.

2. Tabela Utilizatori și Tabela Medici:

Similar cu relația anterioră, aceste tabele au o relație one-to-one sau many-to-zero, realizată prin intermediul cheii primare-externe IdMedic din Tabela de Utilizatori către cheia primară IdMedic din Tabela de Medici.

3. Tabela RecomandăriMedicale și Tabela Pacienți:

Aici avem o relație many-to-one sau many-to-zero (dacă nu toate recomandările sunt asociate unui pacient). Fiecare recomandare medicală din Tabela de RecomandăriMedicale este asociată cu un singur pacient din Tabela de Pacienți prin intermediul cheii primare-externe IdPacient.

4. Tabela RecomandăriMedicale și Tabela Medici:

Similar cu relația precedentă, avem o relație many-to-one sau many-to-zero între aceste tabele. Fiecare recomandare medicală din Tabela de RecomandăriMedicale este asociată cu un singur medic din Tabela de Medici prin intermediul cheii primare-externe IdMedic.

5. Tabela Alarme și Tabela Pacienți:

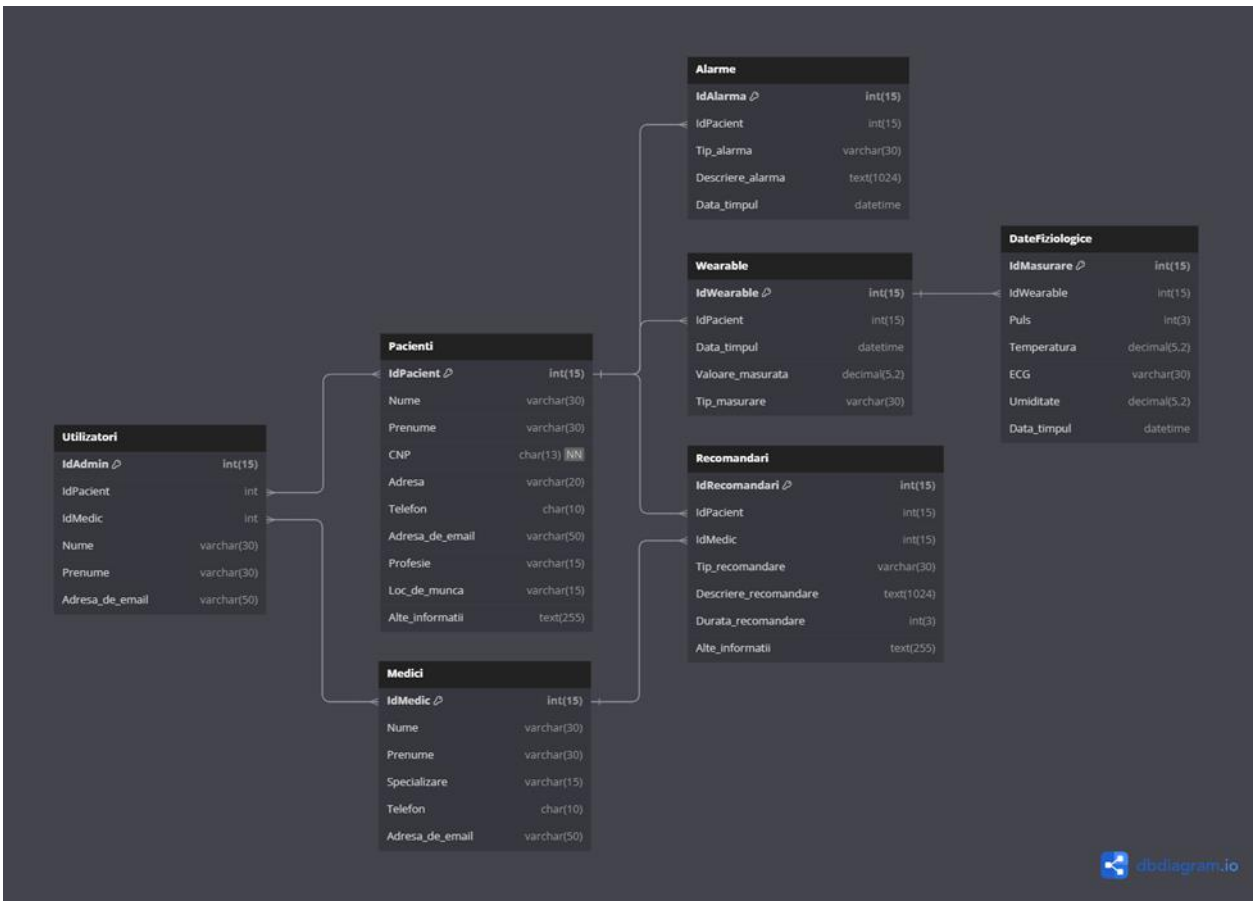
Aici avem o relație many-to-one sau many-to-zero între Tabela de Alarme și Tabela de Pacienți. Fiecare înregistrare din Tabela de Alarme este asociată cu un singur pacient din Tabela de Pacienți prin intermediul cheii primare-externe IdPacient.

6. Tabela Wearable și Tabela Pacienți:

Avem o relație many-to-one sau many-to-zero între aceste tabele. Fiecare înregistrare din Tabela de Wearable este asociată cu un singur pacient din Tabela de Pacienți prin intermediul cheii primare-externe ID_Pacient.

7. Tabela DateFiziologice și Tabela Wearable:

Avem o relație many-to-one sau many-to-zero între aceste tabele. Fiecare înregistrare din Tabela de DateFiziologice este asociată cu un singur dispozitiv din Tabela de Wearable prin intermediul cheii primare-externe ID_Wearable.



5. Planificarea lucrărilor

Pentru a gestiona eficient proiectul nostru de dezvoltare a sistemului purtabil de supraveghere a stării de sănătate, utilizăm o combinație de instrumente și platforme. Diagrama GANTT ne oferă o vizualizare clară a etapelor și a eșalonării lucrărilor pe parcursul proiectului.

Pentru gestionarea sarcinilor și a fluxului de lucru, folosim Trello, care ne permite să organizăm activitățile în panouri și liste, să asignăm sarcini și să monitorizăm progresul în timp real. În plus, utilizăm platforma GitHub pentru gestionarea codului sursă, controlul versiunilor și colaborarea în echipă, asigurându-ne că toți membrii au acces la codul actualizat și că schimbările sunt monitorizate și validate corespunzător.

Prin integrarea acestor instrumente, ne asigurăm că proiectul este gestionat eficient și că comunicarea și colaborarea în echipă sunt streamline și transparente.

Noua diagramă Gantt:

Task	Asigined to	Start	End
Phase 1 Specificatii			
Intelegerea temei	Echipa	14-Mar	14-Mar
Creere logo, nume	Andrei Popesc	13-Mar	18-Mar
Documentatie	Echipa	20-Mar	1-Apr
Impartinrea sarcinilor	Pavel Rosca, Andreea Rus	14-Mar	30-Mar
Phase 2 Proiectarea			
Arhitectura programului	Echipa	14-Mar	14-Mar
Componente	Echipa	13-Mar	18-Mar
Baze de date	Adi Popescu	20-Mar	1-Apr
Phase 2 Implementare			
Modulul WEB	Echipa, Adi POPESCU - Andrei POPESC	25-Apr	23-May
Modulul Cloud	Echipa, Denis PETRUȘE - Daniel PAȘCU	25-Apr	23-May
Modulul Iot	Echipa, Pavel ROSCA - Malina PAIUSAN	25-Apr	23-May
Modulul Wearable	Echipa, Andreea RUS	25-Apr	23-May

Figura 11. Tabelul de Proces

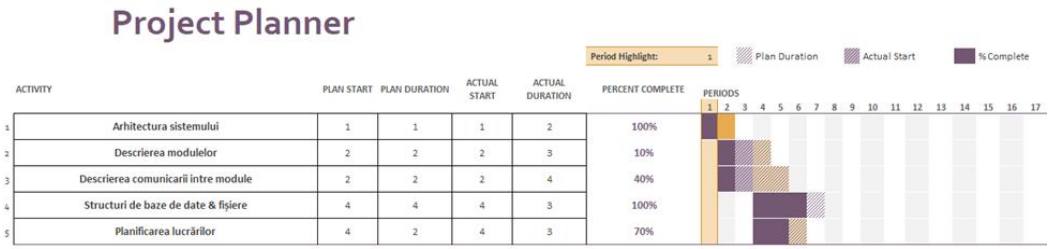


Figura 12. Diagrama Gantt

Aplicația Trello actualizată:

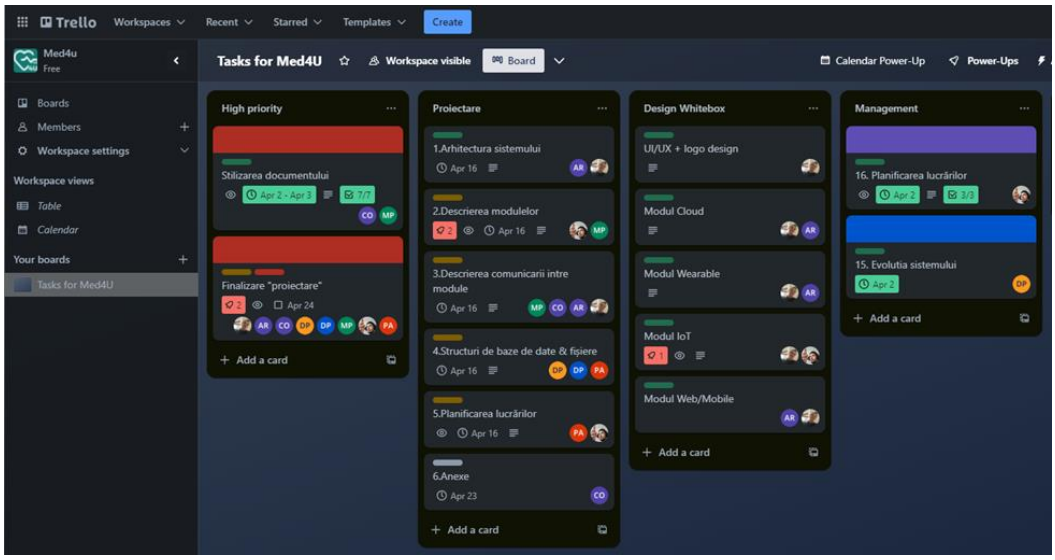


Figura 13. Progresul proiectului conform aplicației Trello

GitHub Depozitar:

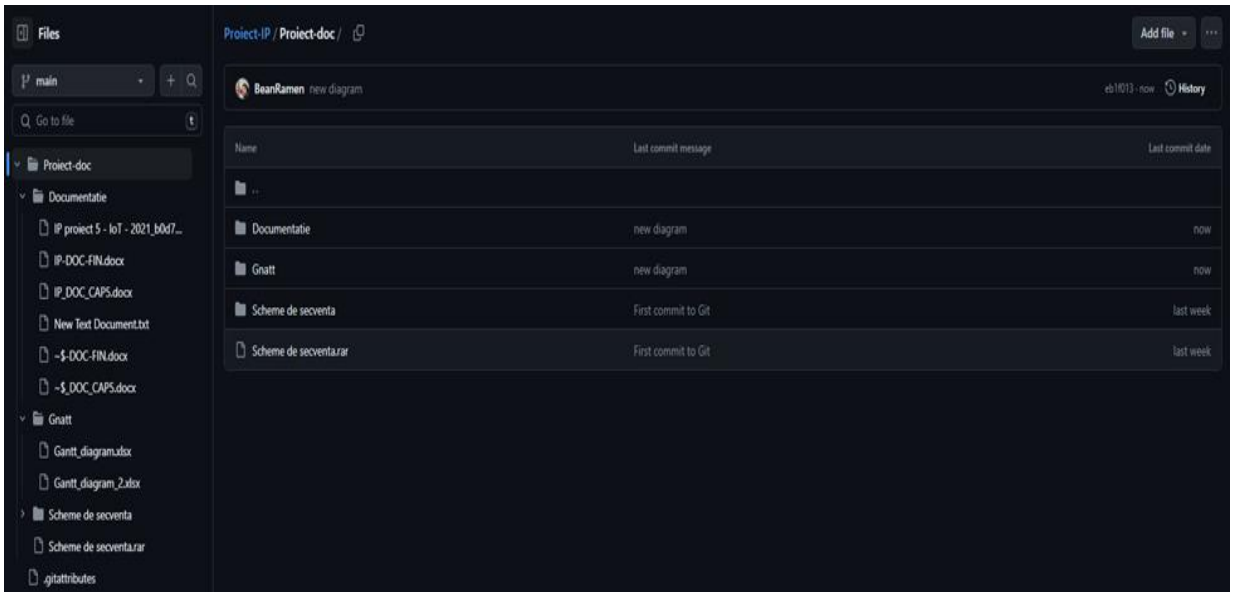


Figura 14. Depozitar GitHub Sistem Purtabil de Supraveghere a Stării de Sănătate

6. Anexe

[1]. https://www.adobe.com/products/illustrator.html(04.04.2024)
[2]. https://react.dev/(04.04.2024)
[3]. https://nodejs.org/en(04.04.2024)
[4]. https://www.mongodb.com/(04.04.2024)
[5]. https://code.visualstudio.com/(08.04.2024)
[6]. https://mosquitto.org/(08.04.2024)
[7]. https://www.robofun.ro/wireless/esp8266-wifi.html(08.04.2024)
[8]. https://www.arduino.cc/en/software(08.04.2024)
[9]. https://ro.farnell.com/iot-wireless-network-protocols(08.04.2024)
[10]. https://mkt-sampath97.medium.com/build-prototype-iot-system-with-node-red-mqtt-and-nodemcu-762407af2917 (08.04.2024)
[11]. https://randomnerdtutorials.com/esp8266-and-node-red-with-mqtt/(08.04.2024)