< Soluție inovativa pentru economisirea apei: Sistem de igienizare a mâinilor prin tehnologie uscată si monitorizare IoT >

# Documentul de proiectare

Cuprins

[1. Introducere 1](#_Toc160527836)

[1.1 Scopul documentului 1](#_Toc160527837)

[2. Prezentare generală și abordări de proiectare 2](#_Toc160527838)

[2.1 Prezentare generală 2](#_Toc160527839)

[2.2 Presupuneri/ Constrângeri/ Riscuri 2](#_Toc160527840)

[2.2.1 Presupuneri 2](#_Toc160527841)

[2.2.2 Constrângeri 2](#_Toc160527842)

[2.2.3 Riscuri 3](#_Toc160527843)

[3. Considerațiii de proiectare 4](#_Toc160527844)

[3.1 Obiective și linii directoare (ghiduri) 4](#_Toc160527845)

[3.2 Metode de dezvoltare 4](#_Toc160527846)

[3.3 Strategii de arhitectură 4](#_Toc160527847)

[4. Arhitectura Sistemului și Proiectarea Arhitecturii 6](#_Toc160527848)

[4.1 Vedere logică 6](#_Toc160527849)

[4.2 Arhitectură hardware 6](#_Toc160527850)

[4.3 Arhitectură software 6](#_Toc160527851)

[4.4 Arhitectura informațiilor 7](#_Toc160527852)

[4.5 Arhitectura de comunicații interne 7](#_Toc160527853)

[4.6 Diagrama de arhitectură a sistemului 8](#_Toc160527854)

[5. Proiectarea sistemului 9](#_Toc160527855)

[5.1 Proiectarea bazei de date 9](#_Toc160527856)

[5.1.1 Obiecte de date și structuri de date rezultante 9](#_Toc160527857)

[5.1.2 Fișiere și baze de date 9](#_Toc160527858)

[5.2 Conversii de date 9](#_Toc160527859)

[5.3 Interfețe utilizator 10](#_Toc160527860)

[5.3.1 Intrări 10](#_Toc160527861)

[5.3.2 Ieșiri 10](#_Toc160527862)

[5.4 Proiectarea interfețelor cu utilizatorul 10](#_Toc160527863)

[6. Scenarii de utilizare 11](#_Toc160527864)

[7. Proiectare de detaliu 12](#_Toc160527865)

[7.1 Proiectare hardware de detaliu 12](#_Toc160527866)

[7.2 Proiectare software de deatliu 12](#_Toc160527867)

[7.3 Proiectare detaliată de securitate 13](#_Toc160527868)

[7.4 Proiectare de detaliu pentru performanța sistemului 14](#_Toc160527869)

[7.5 Proiectare detaliată a comunicațiilor interne (între componente) 14](#_Toc160527870)

[8. Controale pentru verificarea integrității sistemului 15](#_Toc160527871)

[Anexa A: Gestiunea modificărilor documentului 16](#_Toc160527872)

[Anexa B: Acronime 17](#_Toc160527873)

[Anexa C Documente la care se face referire 18](#_Toc160527874)

## Introducere

Instrucțiuni: Furnizați informații de identificare pentru sistemul existent și/sau propus sau situația pentru care se aplică Documentul de Proiectare a Sistemului (de exemplu, numele complet și acronimele proiectului de dezvoltare, sistemul sau situația existentă și propusă, după caz), și evoluția așteptată a documentului. De asemenea, descrieți orice considerații referitor la securitate sau confidențialitate asociate cu utilizarea acestui document..

Proiectul de dezvoltare a unui sistem inteligent de igienizare a mâinilor, denumit în continuare „Sistemul de Igienizare Automatizat”, urmărește crearea unui dispozitiv modern care să îmbunătățească eficiența procesului de igienizare și să contribuie la prevenirea răspândirii bolilor, printr-o abordare automatizată și ușor de utilizat. Acest sistem combină tehnologii avansate de detectare a prezenței mâinilor, control manual și automat al procesului de igienizare, precum și monitorizarea continuă a resurselor și feedback-ul oferit utilizatorului printr-un ecran LCD și aplicația mobilă Blynk.

Proiectul va folosi un **Arduino Mega 2560**, un **modul ESP8266** pentru conectivitate Wi-Fi, **mini relee**, un **senzor de proximitate**, un **senzor de nivel al lichidului** și un **ventilator** pentru difuzarea substanței igienizante. Sistemul va fi capabil să funcționeze atât în mod **automat**, activând igienizarea atunci când este detectată prezența mâinii, cât și în mod **manual**, prin comenzi transmise de la un telefon mobil sau prin butoane fizice de control.

Acest document descrie modul în care cerințele funcționale și non-funcționale, stabilite în **Documentul de Cerințe**, vor fi transformate în specificații tehnice de proiectare a sistemului. Detaliile vor acoperi aspecte ale arhitecturii sistemului la un nivel înalt, precum și descrierea fiecărei componente cheie: hardware, software, comunicații interne și interfețe externe.

În plus, documentul va aborda aspectele legate de **securitate** și **confidențialitate** asociate cu utilizarea acestui sistem, având în vedere că vor exista interacțiuni prin internet și date care pot fi sensibile (precum monitorizarea nivelului lichidului și contorizarea utilizărilor). Toate aceste aspecte vor fi abordate pentru a asigura integritatea și protecția informațiilor utilizatorilor și funcționarea sigură a sistemului.

Astfel, scopul acestui document este de a furniza o descriere detaliată a designului sistemului, incluzând obiectivele și specificațiile de implementare ale fiecărei componente, care vor sta la baza construcției și implementării proiectului.

### Scopul documentului

*Instrucțiuni: Furnizați scopul documentului. Acest document ar trebui să fie adaptat pentru a se potrivi nevoilor unui proiect specific.*

Scopul acestui document este de a descrie arhitectura și designul sistemului pentru **Sistemul de Igienizare Automatizat**, oferind echipei de dezvoltare îndrumări clare pentru implementarea fiecărei componente. Documentul va fi creat și actualizat pe parcursul dezvoltării sistemului, în conformitate cu cerințele și metodologia utilizată.

Publicul țintă include managerul de proiect, echipa de proiect și echipa de dezvoltare, iar anumite secțiuni, precum designul interfeței cu utilizatorul, vor fi împărtășite și cu clientul/utilizatorul. Scopul principal este de a asigura o dezvoltare coordonată și de succes a sistemului, care să răspundă cerințelor stabilite.

## Prezentare generală și abordări de proiectare

Această secțiune descrie principiile și strategiile ce vor ghida proiectarea și implementarea **Sistemului de Igienizare Automatizat**. Abordarea va fi una modulară și scalabilă, axată pe utilizarea tehnologiilor eficiente și fiabile, precum **Arduino Mega 2560**, **ESP8266**, și senzori de proximitate. Sistemul va permite control atât **manual**, prin aplicația mobilă și butoane fizice, cât și **automat**, prin detectarea prezenței mâinilor.

De asemenea, se va pune accent pe **securitatea** datelor și **fiabilitatea** sistemului, pentru a asigura o utilizare sigură și continuă. Comunicarea între componente va fi optimizată pentru a asigura performanța sistemului în condiții de utilizare variabile.

### Prezentare generală

Instrucțiuni: Prezentați succint contextul sistemului și abordarea sau organizarea de bază a proiectării. Oferiți o prezentare generală a arhitecturilor sistemului și software-ului și a obiectivelor de proiectare.

Sistemul de Igienizare Automatizat este conceput pentru a îmbunătăți eficiența procesului de igienizare a mâinilor, oferind o soluție automată și manuală prin control de la distanță sau prin butoane fizice. Arhitectura sistemului include un **Arduino Mega 2560** ca unitate principală de control, un **modul ESP8266** pentru conectivitate Wi-Fi și o serie de **senzori de proximitate** și **senzori de nivel al lichidului** pentru monitorizarea utilizării și a resurselor. Un ventilator va fi utilizat pentru difuzarea substanței igienizante, iar un ecran LCD va furniza informații utilizatorului.

În ceea ce privește software-ul, sistemul va funcționa pe baza unui program ce va permite controlul automat al procesului de igienizare, pe baza detectării prezenței mâinii, și va integra un modul Blynk pentru controlul de la distanță. Obiectivele de proiectare includ fiabilitate, eficiență în utilizare și securitate, pentru a asigura un sistem de igienizare funcțional și sigur.

### Presupuneri/ Constrângeri/ Riscuri

#### Presupuneri

Instrucțiuni: Descrieți orice presupuneri sau dependențe legate de sistem, software și utilizarea sa. Acestea pot viza probleme precum: software sau hardware asociat, sisteme de operare, caracteristicile utilizatorilor finali și posibilele și/sau probabilele modificări ale funcționalității.

 **Hardware**: Se presupune că toate componentele hardware (Arduino Mega 2560, ESP8266, senzori, mini relee, ventilator, LED UV, etc.) vor funcționa conform specificațiilor furnizate și nu vor prezenta defecte de fabricație.

 **Software**: Sistemul va fi dezvoltat pe platforma Arduino IDE, iar aplicația mobilă va folosi platforma Blynk pentru conectivitate. Se presupune că utilizatorii vor avea o conexiune Wi-Fi stabilă pentru a controla sistemul de la distanță.

 **Utilizatori finali**: Se presupune că utilizatorii vor fi familiarizați cu utilizarea unei aplicații mobile pentru controlul de la distanță. De asemenea, se consideră că sistemul va fi utilizat într-un mediu curat și controlat, fără interferențe semnificative de mediu (umiditate extremă, temperaturi foarte scăzute sau înalte).

 **Posibile modificări ale funcționalității**: Este posibil ca în viitor să fie necesare actualizări ale software-ului pentru adăugarea unor noi funcționalități, cum ar fi monitorizarea mai detaliată a utilizării sau integrarea unor noi dispozitive de igienizare. Aceste modificări vor fi gestionate prin actualizări de firmware și software.

#### Constrângeri

 **Mediu hardware**: Dimensiunile și capacitatea de procesare ale **Arduino Mega 2560** pot limita complexitatea funcționalităților, iar numărul de pini disponibili poate impune limitări în ceea ce privește numărul de senzori și relee conectate simultan.

 **Mediu utilizator final**: Utilizatorii finali trebuie să aibă dispozitive mobile compatibile cu aplicația Blynk și o conexiune Wi-Fi stabilă pentru controlul de la distanță.

 **Disponibilitatea resurselor**: Există o dependență de disponibilitatea componentelor hardware (senzori, relee, ventilatoare etc.), iar eventualele întârzieri în livrarea acestora pot afecta termenul de finalizare al proiectului.

 **Cerințe de securitate**: Securitatea datelor este esențială, având în vedere comunicarea prin rețea (Wi-Fi). Toate datele transmise între sistem și aplicația mobilă vor fi criptate pentru a preveni accesul neautorizat.

 **Limitări ale memoriei**: Arduino Mega 2560 are o memorie limitată, ceea ce poate restricționa complexitatea software-ului și numărul de funcționalități avansate implementate.

 **Performanță**: Sistemul trebuie să răspundă rapid la comenzi, iar orice întârziere semnificativă poate afecta experiența utilizatorului. De asemenea, trebuie asigurată o funcționare continuă, fără erori sau întreruperi.

 **Cerințe de testare**: Testarea sistemului va include verificarea fiecărei componente hardware și software pentru a asigura fiabilitatea și performanța corespunzătoare înainte de implementare în mediul final.

#### Riscuri

Instrucțiuni: Descrieți orice riscuri asociate cu designul sistemului și strategiile propuse de reducere a acestora.

 **Defecțiuni hardware**: Componentele hardware pot prezenta defecte de fabricație sau pot fi afectate de condiții de mediu (ex. temperaturi extreme). *Strategia de reducere*: Utilizarea unor componente de calitate și verificarea lor înainte de implementare.

 **Probleme de conectivitate Wi-Fi**: Dacă utilizatorii nu dispun de o conexiune stabilă la internet, sistemul de control de la distanță va fi afectat. *Strategia de reducere*: Oferirea unei opțiuni de control manual prin butoane fizice în cazul în care Wi-Fi nu este disponibil.

 **Întârziere în dezvoltare**: Posibile întârzieri în livrarea componentelor sau dificultăți tehnice în integrarea software-ului. *Strategia de reducere*: Planificarea detaliată a resurselor și evaluarea continuă a progresului proiectului.

 **Probleme de securitate**: Vulnerabilități în comunicarea prin rețea care ar putea permite accesul neautorizat. *Strategia de reducere*: Implementarea criptării și autentificării utilizatorilor în aplicația mobilă.

 **Dependență de actualizări software**: Posibilitatea ca software-ul să necesite actualizări frecvente pentru a se adapta la schimbări de mediu sau cerințe noi. *Strategia de reducere*: Implementarea unui sistem de actualizări OTA (Over The Air) pentru software, ușor de implementat de la distanță.

## Considerații de proiectare

Instrucțiuni: Descrieți problemele care trebuie abordate sau rezolvate înainte de a încerca să elaborați o soluție de design completă.

nainte de elaborarea unei soluții complete de design pentru **Sistemul de Igienizare Automatizat**, trebuie abordate următoarele probleme:

1. **Compatibilitatea hardware**: Asigurarea integrării corecte între toate componentele hardware (Arduino Mega 2560, ESP8266, senzori, relee, ventilator, etc.), ținând cont de limitările pinilor și resurselor disponibile.
2. **Comunicarea între dispozitive**: Stabilirea unei conexiuni Wi-Fi stabile între sistem și aplicația mobilă Blynk, având în vedere posibilele interferențe de rețea sau lipsa de conectivitate.
3. **Gestionarea resurselor**: Asigurarea unui management eficient al lichidului igienizant și al alimentării cu energie, pentru a preveni deficiențele în funcționare.
4. **Securitate**: Protejarea comunicațiilor între dispozitive, prin criptare și autentificare, pentru a preveni accesul neautorizat.
5. **Interfața cu utilizatorul**: Crearea unei aplicații mobile intuitive și ușor de utilizat, care să permită controlul eficient al sistemului atât automat, cât și manual.
6. **Fiabilitatea sistemului**: Testarea fiecărei componente pentru a preveni defectele hardware/software și asigurarea unei funcționări continue și precise a întregului sistem.

### Obiective și linii directoare (ghiduri)

**Obiective de design**:

1. **Performanță vs. resurse**: Sistemul trebuie să fie rapid și să răspundă prompt la comenzi, fără a compromite eficiența memoriei și a resurselor hardware limitate ale Arduino Mega 2560.
2. **Fiabilitate**: Sistemul trebuie să funcționeze continuu, fără erori, asigurându-se că detectarea prezenței și distribuirea igienizantului sunt precise.
3. **Usabilitate**: Aplicația mobilă și interfața sistemului trebuie să fie intuitive și ușor de utilizat de către orice utilizator, fără a necesita instruire complexă.
4. **Securitate**: Datele transmise prin Wi-Fi trebuie să fie criptate pentru a proteja informațiile utilizatorilor.

**Linii directoare de codare**:

1. **Modularitate**: Codul trebuie să fie structurat pe module, fiecare componentă (senzori, relee, aplicație mobilă) având funcții separate, pentru o întreținere mai ușoară și actualizări rapide.
2. **Comentarii și documentație**: Codul trebuie să fie bine comentat, explicând fiecare funcție, mai ales cele critice pentru funcționarea sistemului.
3. **Consistența convențiilor de numire**: Variabilele și funcțiile vor urma convenții clare de numire, de exemplu, folosirea unui prefix standardizat pentru variabilele de tip senzor (ex: sensor\_).

**Strategii de design**:

* **Alegerea hardware-ului**: Se va utiliza un **modul ESP8266** pentru Wi-Fi datorită eficienței sale în termeni de cost și performanță.
* **Utilizarea unui ecran LCD**: Acesta va fi folosit pentru a furniza feedback în timp real utilizatorului despre nivelul lichidului și starea sistemului, fără a crea complexitate suplimentară în aplicația mobilă.

### Metode de dezvoltare

Pentru **designul sistemului și al software-ului**, s-a adoptat o metodă **orientată pe obiecte**, susținută de **diagrama UML** pentru a vizualiza structura și interacțiunile sistemului. Această abordare permite o mai bună organizare și gestionare a componentelor, respectând principiile de modularitate și reutilizare a codului. De asemenea, metoda permite o planificare clară a relațiilor dintre obiectele software și îmbunătățirea colaborării între echipele de dezvoltare.

* **Abordare orientată pe obiecte**: Sistemul este împărțit în clase și obiecte care se ocupă de gestionarea senzorilor, controlul releelor, comunicarea Wi-Fi și interacțiunea cu utilizatorul. Această abordare facilitează extinderea și întreținerea ulterioară a sistemului.
* **Prototipare rapidă și integrare continuă**: Fiecare componentă a sistemului (hardware, software, aplicație mobilă) a fost prototipat și testat separat. După validarea fiecărei componente, acestea au fost integrate într-un sistem complet. Această metodă a permis identificarea timpurie a erorilor și a îmbunătățit iterativ performanța întregului sistem.
* **Utilizarea diagramelor UML**: Diagrama UML a fost folosită pentru a descrie structura și interacțiunile dintre componentele software, inclusiv clasele și relațiile acestora (ex. diagrama de clase, diagrama de secvență). Acest lucru a ajutat la clarificarea arhitecturii și a fluxului de date din sistem.

**Metode formale**: În ciuda utilizării diagramelor UML pentru clarificarea designului, nu s-au adoptat metode formale suplimentare precum J2EE, deoarece complexitatea aplicației nu impune astfel de abordări. Totuși, designul modular, bazat pe obiecte, asigură o separare clară a responsabilităților și o întreținere ușoară.

**Contingente și soluții de rezervă**:

1. În cazul unor dificultăți de conectivitate Wi-Fi, va fi implementat un **mod offline**, permițând controlul manual al sistemului prin butoane fizice.
2. Dacă aplicația mobilă întâmpină probleme de compatibilitate pe anumite dispozitive, se va dezvolta o **interfață web** pentru controlul de la distanță.

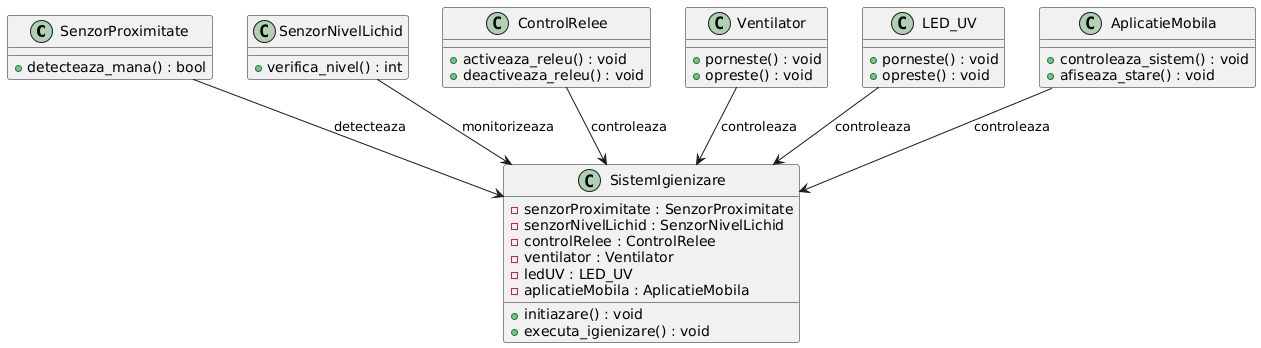


Fig.1 Diagama de clasa

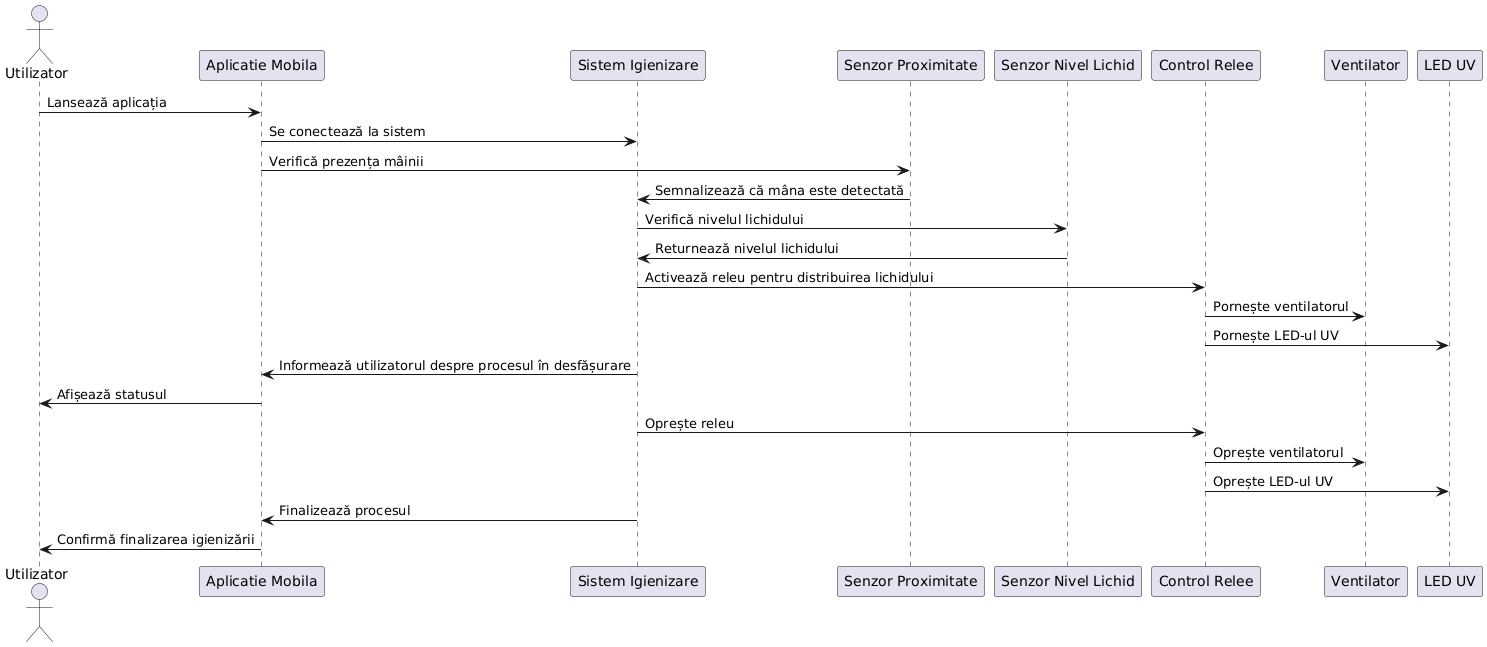


Fig.2 Diagrama de secventa

### Strategii de arhitectură

Arhitectura sistemului se bazează pe o abordare modulară, flexibilă și scalabilă, pentru a asigura eficiență și ușurință în întreținere. Principalele decizii de design și raționamentele pentru alegerea acestora sunt următoarele:

1. **Utilizarea Arduino Mega 2560 și ESP8266**: Alegerea acestora ca platforme hardware se bazează pe costuri reduse, flexibilitate și suport larg din partea comunității. ESP8266 permite conectivitatea Wi-Fi, iar Arduino Mega asigură suficiente porturi și putere de procesare.
2. **Modularitatea software-ului**: Sistemul este împărțit în module distincte (senzori, relee, control ventilator/UV, aplicație mobilă), ceea ce permite extinderea ușoară a funcționalității și îmbunătățirea performanței pe măsură ce cerințele cresc.
3. **Comunicarea între module**: Comunicarea se face printr-o arhitectură client-server (ESP8266 ca server, aplicația mobilă ca client), asigurându-se un control eficient al sistemului prin Wi-Fi. Alte opțiuni de comunicație (de exemplu, Bluetooth) au fost respinse din cauza limitărilor de distanță și viteză.
4. **Detectarea erorilor**: Sistemul include verificări de integritate (de exemplu, nivelul lichidului) și feedback continuu prin aplicația mobilă, pentru a alerta utilizatorul în caz de defecțiuni.
5. **Gestionarea memoriei și performanță**: Alegerea unui limbaj de programare ușor și rapid, precum C++ pentru Arduino, permite un control precis al memoriei și al performanței, optimizând resursele disponibile.

## Arhitectura Sistemului și Proiectarea Arhitecturii

Sistemul este structurat pe o arhitectură modulară, având mai multe subsisteme care colaborează pentru a asigura igienizarea eficientă a mâinilor. Arhitectura generală este împărțită în componente de control hardware și software, care sunt conectate între ele pentru a furniza funcționalitatea dorită.

1. **Componentele subsistemului**:
   * **Controlul hardware**: Arduino Mega 2560 și ESP8266, care gestionează senzorii, releele și interfața cu aplicația mobilă.
   * **Senzori**: Senzorul de proximitate pentru detectarea mâinii și senzorul de nivel lichid pentru monitorizarea lichidului igienizant.
   * **Componente de control**: Releele care activează dispozitivele (ventilator, LED UV).
   * **Aplicația mobilă**: Interfața utilizatorului pentru control manual și monitorizare.
2. **Comunicarea și colaborarea între componente**:
   * **ESP8266** asigură comunicarea între sistemul de igienizare și aplicația mobilă, printr-o arhitectură client-server.
   * Senzorii trimit date către sistem pentru a determina momentul în care să pornească procesul de igienizare, iar controlul releelor inițiază acțiunile (activarea ventilatorului și LED-ului UV).
3. **Motivarea decompunerii**:
   * S-a ales o arhitectură modulară pentru a asigura extensibilitatea și întreținerea ușoară. Separarea funcționalităților (detecție, control dispozitive, interfață utilizator) permite adăugarea de noi funcționalități în viitor fără a afecta restul sistemului.
4. **Justificarea pentru alegerea tiparelor de proiectare**:
   * **Tiparul client-server** este folosit pentru a separa clar rolurile și responsabilitățile între aplicația mobilă (client) și sistemul de igienizare (server), optimizând astfel comunicarea și controlul.
   * **Tiparul de proiectare modulară** este aplicat pentru a crea componente ușor de extins și întreținut.

### Vedere logică

Vederea logică a sistemului descrie modul în care componentele software și hardware sunt organizate pentru a îndeplini funcționalitățile necesare. Sistemul este împărțit în module distincte, fiecare cu un rol clar definit:

1. **Modul de detecție**: Senzorii de proximitate și de nivel al lichidului monitorizează starea și inițiază procesele de igienizare.
2. **Modul de control**: Arduino Mega 2560 și ESP8266 gestionează semnalele de la senzori și controlează releele pentru ventilator și LED UV.
3. **Interfață utilizator**: Aplicația mobilă permite controlul manual, vizualizarea stării sistemului și ajustarea setărilor.

Componentele colaborează printr-o arhitectură client-server, cu ESP8266 ca server și aplicația mobilă ca client. Datele de la senzori sunt transmise către controlerul principal, care activează dispozitivele de igienizare.

### Arhitectură hardware

Sistemul utilizează o arhitectură hardware centralizată, unde procesarea principală este realizată de Arduino Mega 2560 și ESP8266, care gestionează toate componentele și comunicațiile.

1. **Componente hardware principale**:
   * **Arduino Mega 2560**: Placa principală de control, responsabilă pentru gestionarea senzorilor și releelor. Se conectează la ESP8266 pentru comunicațiile wireless.
   * **ESP8266**: Modul Wi-Fi care permite comunicarea între sistemul de igienizare și aplicația mobilă. Acesta funcționează ca un server, gestionând cererile de control ale utilizatorului.
   * **Senzori**: Senzorul de proximitate și senzorul de nivel al lichidului, care sunt conectați direct la Arduino.
   * **Relee**: Folosite pentru a activa dispozitivele de igienizare, cum ar fi ventilatorul și LED-ul UV.
   * **Dispozitive periferice**: Ventilatorul și LED-ul UV sunt controlate de relee pentru a activa procesul de igienizare.
2. **Conectivitate**:
   * **Arduino Mega 2560** comunică prin cabluri cu senzori și relee.
   * **ESP8266** se conectează la rețeaua Wi-Fi locală pentru a permite interacțiunea cu aplicația mobilă.
   * Conexiunile între componente sunt protejate de firewall-uri, iar porturile de comunicație utilizate sunt cele standard pentru Wi-Fi (802.11).
3. **Estimări de resurse**:
   * **Procesor**: Arduino Mega 2560 (8-bit), suficient pentru sarcini de control și monitorizare.
   * **Memorie**: Arduino are 256 KB flash și 8 KB SRAM, suficient pentru stocarea programului și a datelor temporare.
   * **Stocare**: Nu sunt necesare stocări externe, deoarece sistemul lucrează în timp real.
   * **Banda de rețea**: Utilizarea rețelei Wi-Fi prin ESP8266 necesită o bandă de 2,4 GHz, cu un consum minim de lățime de bandă.

### Arhitectură software

#### ****Componente software principale:****

1. **Arduino IDE**: Mediul de dezvoltare utilizat pentru scrierea și încărcarea codului pe **Arduino Mega 2560** (Limbaj de programare: C/C++).
2. **Biblioteci Arduino**: Folosite pentru interacțiunea cu senzorii și controlul releelor (ex. DHT.h, ESP8266WiFi.h).
3. **ESP8266**: Modul Wi-Fi care permite conectivitatea cu internetul, gestionând comunicația între Arduino și platforma **Blynk**.
4. **Blynk**: Aplicația mobilă care permite controlul de la distanță al sistemului prin Wi-Fi, utilizând protocoale HTTP/MQTT pentru transmiterea comenzilor.

#### ****Interfețe și comunicație:****

* **Arduino ↔ ESP8266**: Comunicarea se face prin portul serial (USART).
* **ESP8266 ↔ Blynk**: Folosește protocoale HTTP/MQTT pentru a transmite și recepționa comenzi.

#### ****Raționament design:****

* **Blynk** este utilizat datorită simplității și suportului larg pentru proiecte IoT rapide.
* **ESP8266** este ales pentru costul scăzut și compatibilitatea cu Arduino, permițând conectivitatea Wi-Fi eficientă.

### Arhitectura informațiilor

#### ****Informații stocate în sistem:****

1. **Datele utilizatorilor:**
   * **Tip:** Informații despre utilizatori (ID, nume, date de autentificare)
   * **Format:** Date electronice stocate pe serverele Blynk
   * **Furnizor:** Utilizatorul prin aplicația mobilă Blynk.
   * **Sensibilitate:** Nu sunt informații sensibile.
2. **Datele de utilizare a sistemului:**
   * **Tip:** Numărul de utilizări ale sistemului de igienizare, durata fiecărei utilizări
   * **Format:** Date electronice stocate pe platforma Blynk.
   * **Furnizor:** Sistemul de igienizare, înregistrate automat la fiecare utilizare.
   * **Sensibilitate:** Nu sunt informații sensibile.
3. **Date de monitorizare a lichidului și a stării echipamentului:**
   * **Tip:** Nivelul lichidului de igienizare, statusul releelor (on/off)
   * **Format:** Date electronice stocate pe dispozitivul Arduino și transmise către Blynk.
   * **Furnizor:** Sistemul automatizat (senzori de proximitate și senzor de nivel lichid).
   * **Sensibilitate:** Nu sunt informații sensibile.

#### ****Sistem de stocare:****

* Datele sunt stocate pe platforma **Blynk** pentru acces de la distanță și monitorizare în timp real.
* **Sensibilitate:** Informațiile stocate nu sunt considerate sensibile, deoarece nu includ date personale sau medicale.

#### ****Securitate și confidențialitate:****

* Nu există informații cu caracter sensibil stocate în sistem.
* Comunicarea între Arduino și Blynk se face criptat (prin protocoale HTTPS/MQTT).

.

### Arhitectura de comunicații interne

#### ****Descrierea rețelei de comunicații a sistemului:****

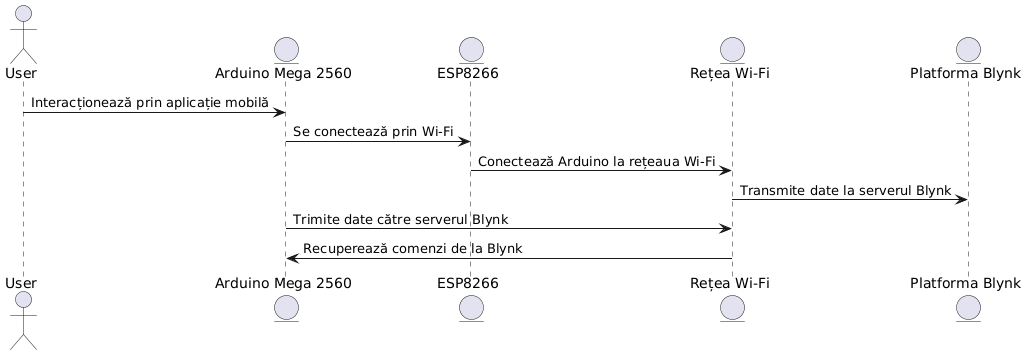
Sistemul utilizează o arhitectură de comunicație **client-server**, în care Arduino 2560 cu modul ESP8266 se conectează la un server extern (platforma Blynk) prin rețeaua Wi-Fi locală.

**Componentele rețelei de comunicații sunt:**

1. **Arduino Mega 2560 cu ESP8266:**
   * **Funcție:** Modulul ESP8266 se conectează la rețeaua Wi-Fi locală pentru a comunica cu serverul Blynk.
   * **Tip de comunicație:** Wireless (Wi-Fi) prin protocolul **HTTP/MQTT** pentru a transmite datele sistemului (nivel lichid, număr utilizări, etc.) și a primi comenzi de la aplicația mobilă.
2. **Platforma Blynk:**
   * **Funcție:** Servește ca server de interfață între utilizator și sistem. Permite monitorizarea și controlul sistemului de igienizare prin aplicația mobilă.
   * **Tip de comunicație:** **HTTPS** și **MQTT** pentru a asigura transmiterea datelor între ESP8266 și serverul Blynk.
3. **Rețea Wi-Fi locală:**
   * **Funcție:** Asigură comunicarea între ESP8266 și serverul Blynk. Rețeaua Wi-Fi este esențială pentru transmiterea și recepționarea datelor.
   * **Echipament necesar:** Router Wi-Fi sau un punct de acces pentru a conecta dispozitivul Arduino la internet.
4. **Firewall-uri și Porturi:**
   * **Funcție:** Securizarea comunicațiilor între componentele sistemului și platforma Blynk.
   * **Porturi:** TCP/UDP pentru comunicarea HTTP și MQTT pe porturile standard (80 pentru HTTP, 1883 pentru MQTT).

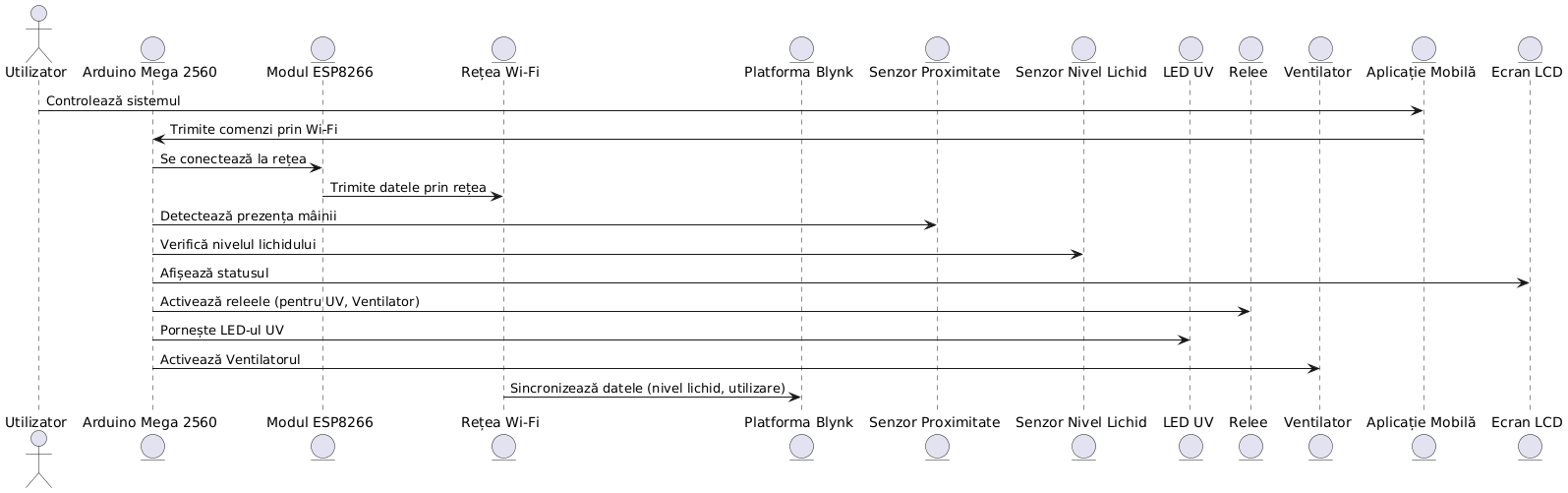
#### ****Estimări ale resurselor necesare:****

* **Lățimea de bandă a rețelei (WAN):** 0.5 - 1 Mbps, suficient pentru transmiterea datelor de la Arduino la platforma Blynk.
* **Lățimea de bandă (LAN):** Rețeaua internă (Wi-Fi local) nu va necesita mai mult de 1 Mbps pentru comunicațiile interne.



.

### Diagrama de arhitectură a sistemuluiâ



## Proiectarea sistemului

### Proiectarea bazei de date

aza de date este destinată să stocheze informațiile esențiale pentru funcționarea sistemului, cum ar fi datele despre utilizatori, dispozitive, sesiuni de igienizare și istoricul utilizării. Se vor implementa tabele pentru fiecare entitate relevantă, iar relațiile dintre acestea vor fi stabilite conform principiilor bazei de date relaționale.

#### ****2. Dicționar de date****

| **Nume element de date** | **Tip** | **Lungime** | **Sursa** | **Reguli de validare** | **CRUD** | **Stocare** | **Ieșiri** | **Aliasuri** | **Descriere** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **user\_id** | INT | - | Aplicație mobilă | Unic, pozitiv | C,R,U,D | SQL | ID utilizator | - | Identificator unic pentru fiecare utilizator |
| **nume\_utilizator** | VARCHAR | 100 | Aplicație mobilă | Nu poate fi gol | C,R,U,D | SQL | Numele utilizator | - | Numele complet al utilizatorului |
| **email** | VARCHAR | 150 | Aplicație mobilă | Unic, valid pentru format de email | C,R,U,D | SQL | Email utilizator | - | Adresa de email a utilizatorului |
| **data\_creare** | DATETIME | - | Sistem | Automat | C,R | SQL | Data înregistrării | - | Data creării contului utilizatorului |
| **nivel\_lichid** | FLOAT | - | Dispozitiv | 0 <= nivel\_lichid <= 100 | C,R,U,D | SQL | Nivel lichid | - | Nivelul lichidului din rezervor |
| **status\_dispozitiv** | VARCHAR | 50 | Dispozitiv | "Operabil", "Neoperabil", "În pauză" | C,R,U,D | SQL | Status dispozitiv | - | Starea curentă a dispozitivului |
| **istoric\_utilizare\_id** | INT | - | Sistem | Unic | C,R | SQL | ID istoric | - | Identificator unic al unui istoric de utilizare |
| **user\_id** | INT | - | Aplicație mobilă | Referință către user\_id din tabelul utilizatori | C,R | SQL | - | - | Legătura între utilizator și istoricul său de utilizare |
| **data\_igienizare** | DATETIME | - | Sistem | Înregistrare automată la fiecare utilizare | C,R | SQL | Data igienizării | - | Data și ora când a fost realizată igienizarea |
| **temperatura** | FLOAT | - | Dispozitiv | -50°C <= temperatură <= 150°C | C,R | SQL | Temperatura | - | Temperatura măsurată de senzor în timpul igienizării |

#### ****3. Structura bazei de date****

Baza de date va include următoarele tabele principale:

1. **Tabelul Utilizatori**: Stochează informațiile despre utilizatori.
   * user\_id: ID-ul unic al utilizatorului (cheie primară).
   * nume\_utilizator: Numele complet al utilizatorului.
   * email: Adresa de email.
   * data\_creare: Data creării contului.
2. **Tabelul Dispozitive**: Stochează informațiile despre dispozitivele de igienizare.
   * dispozitiv\_id: ID-ul dispozitivului (cheie primară).
   * nivel\_lichid: Nivelul de lichid din rezervorul dispozitivului.
   * status\_dispozitiv: Starea dispozitivului (ex: operabil, neoperabil).
3. **Tabelul Istoric\_Utilizări**: Păstrează istoricul utilizărilor dispozitivului.
   * istoric\_utilizare\_id: ID-ul unic al unei înregistrări din istoricul utilizării (cheie primară).
   * user\_id: ID-ul utilizatorului (cheie externă).
   * data\_igienizare: Data și ora în care a avut loc igienizarea.
   * temperatura: Temperatura la momentul igienizării.

#### ****4. Exemple de interogări (SQL)****

**Crearea tabelelor**:

sql

Copy code

CREATE TABLE Utilizatori (

user\_id INT AUTO\_INCREMENT PRIMARY KEY,

nume\_utilizator VARCHAR(100) NOT NULL,

email VARCHAR(150) UNIQUE NOT NULL,

data\_creare DATETIME DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE Dispozitive (

dispozitiv\_id INT AUTO\_INCREMENT PRIMARY KEY,

nivel\_lichid FLOAT DEFAULT 100,

status\_dispozitiv VARCHAR(50) CHECK (status\_dispozitiv IN ('Operabil', 'Neoperabil', 'În pauză')) DEFAULT 'Operabil'

);

CREATE TABLE Istoric\_Utilizări (

istoric\_utilizare\_id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT,

dispozitiv\_id INT,

data\_igienizare DATETIME DEFAULT CURRENT\_TIMESTAMP,

temperatura FLOAT,

FOREIGN KEY (user\_id) REFERENCES Utilizatori(user\_id),

FOREIGN KEY (dispozitiv\_id) REFERENCES Dispozitive(dispozitiv\_id)

);

**Interogare pentru obținerea istoricului unui utilizator**:

sql

Copy code

SELECT u.nume\_utilizator, i.data\_igienizare, i.temperatura

FROM Istoric\_Utilizări i

JOIN Utilizatori u ON i.user\_id = u.user\_id

WHERE u.user\_id = 1; -- Exemplu pentru un utilizator cu ID-ul 1

#### ****5. Proceduri pentru manipularea datelor****

* **Adăugare utilizator**: Inserare de noi utilizatori în tabelul Utilizatori.
* **Actualizare status dispozitiv**: Modificarea nivelului de lichid sau a statusului unui dispozitiv.
* **Ștergere utilizator**: Ștergerea unui utilizator (și a istoricului său de utilizare asociat).

#### ****6. Stocarea și managementul fișierelor non-DBMS****

Pe lângă datele structurate, sistemul va utiliza fișiere non-DBMS pentru:

* Configurarea sistemului (de ex. setările de rețea Wi-Fi).
* Date temporare sau fișiere de log pentru monitorizarea sistemului.

#### ****7. Performanță și scalabilitate****

Baza de date va fi proiectată pentru a face față unui număr mare de utilizatori și de dispozitive. Se va implementa indexarea corespunzătoare pe coloanele frecvent accesate (de exemplu, user\_id, data\_igienizare), iar pentru volume mari de date se vor lua în considerare tehnici de partajare a datelor și replicare a bazei de date.