Home Automation

Proiectare

Membrii echipei:

Istrate Sebastian-Nicolae – programator șef

Ionescu Sergiu-Marian – asistent șef

Moticica Vlad-Florin – secretar

Ionică Florentina-Bianca

Matei Georgiana

Ludică Maria-Alexandra

Mihai Andrei

Negriu Andrei-Valentin

Cuprins

[1. Arhitectura programului 3](#_5j591tmedazs)

[2. Diagrama de secvență pentru întregul sistem 9](#_sylwqs4n7lr2)

[3. Scheme logice 10](#_nifi4fgalt03)

[4. Diagrame de clase 14](#_p32k4phhj5e8)

[5. Scheme cu interacțiunea în interfețe 20](#_aoehw3o40c2i)

[6. Descrierea modulelor din punct de vedere al funcționalității 24](#_kvh7m7figcco)

[7. Explicarea funcționării schemelor logice 28](#_hqzxuj2qqj1t)

[8. Structuri de baze de date și fișiere 30](#_1xo2o56vh69u)

[9. Anexe 33](#_1vi0kx33gmt7)

# 1. Arhitectura programului

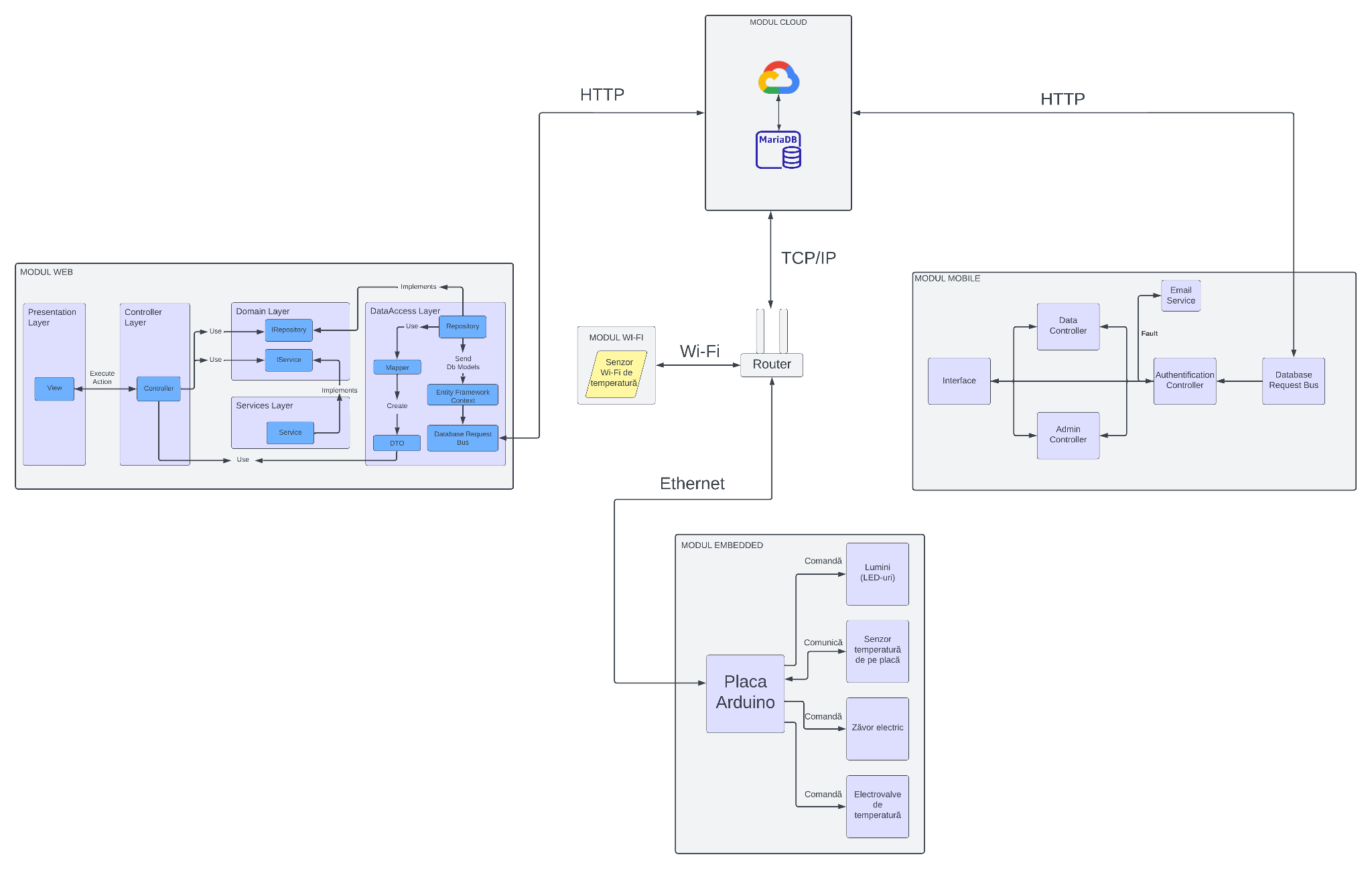


Figura 1. Arhitectura întregului sistem

În Figura 1 se poate observa arhitectura întregului sistem, combinând toate

modulele incluse în proiect (Figura 2 - Figura 5), alături de protocoalele care fac legătura între acestea.

În continuare vom segmenta fiecare modul, refăcând legăturile utilizând blocuri

simbolice, pentru a ușura înțelegerea schemelor.

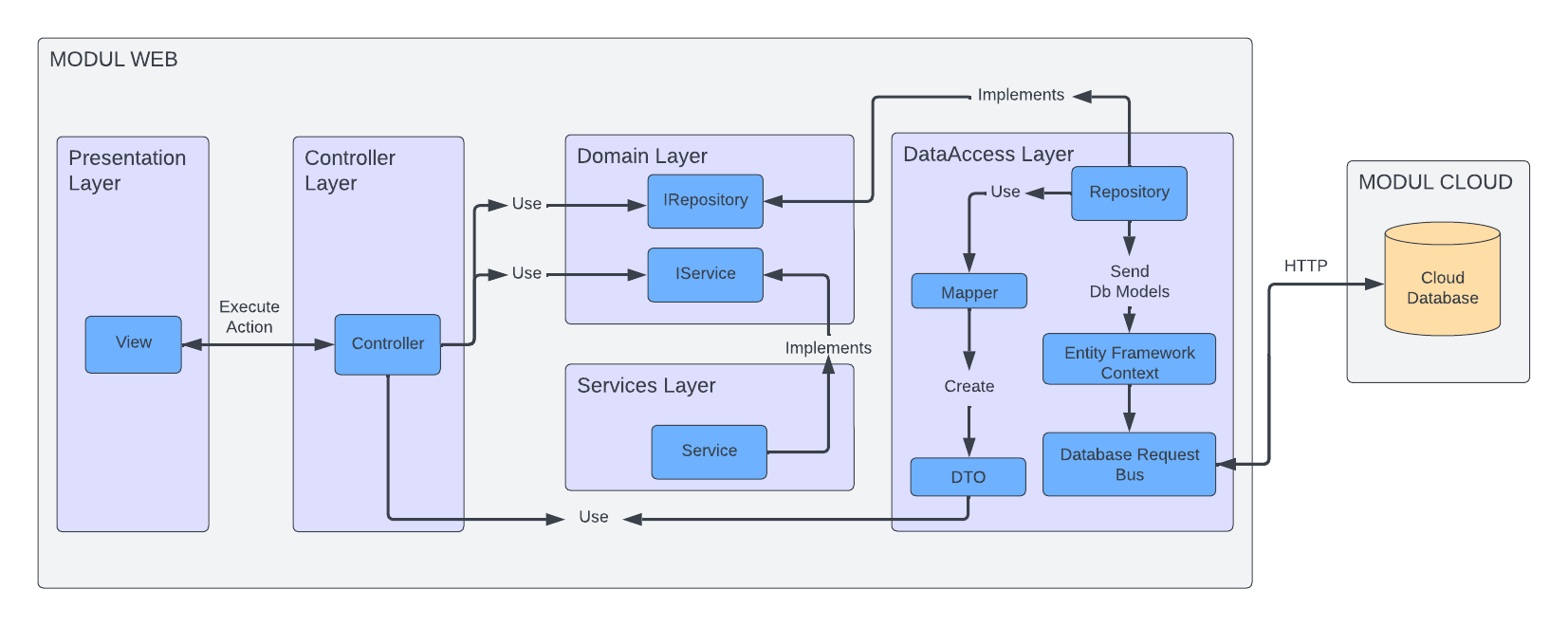


Figura 2. Arhitectura modulului Web

În cadrul modulului WEB se va merge pe o abordare “three layer architecture”, în principiu fiind realizată o separare a straturilor de prezentare, aplicație și acces la date. Modulul WEB va comunica cu modulul Cloud prin intermediul protocolului HTTP.

La nivelul de presentation se regăsește partea de UI a aplicației reprezentată de “views”, acestea fiind componentele corespondențe metodelor din controllere. În controllere vor fi preluate request-urile transmise de view-uri și vor fi procesate în clasele de repository. Clasele repository vor fi responsabile de prelucrarea, aducerea și actualizarea datelor curente și din baza de date, traduse în cod prin clasele modele.

După încheierea prelucrării și în urma unei mapări, datele se vor întoarce din repository către controllere sub forma de DTO-uri (Data Transfer Object) și vor conține doar câmpurile necesare de expus către utilizator.

Vor mai exista două straturi în aplicație: cel de domeniu, unde se vor afla interfețele, și cel pentru servicii care va găzdui clase cu funcționalități suplimentare (ex. trimitere de email-uri).

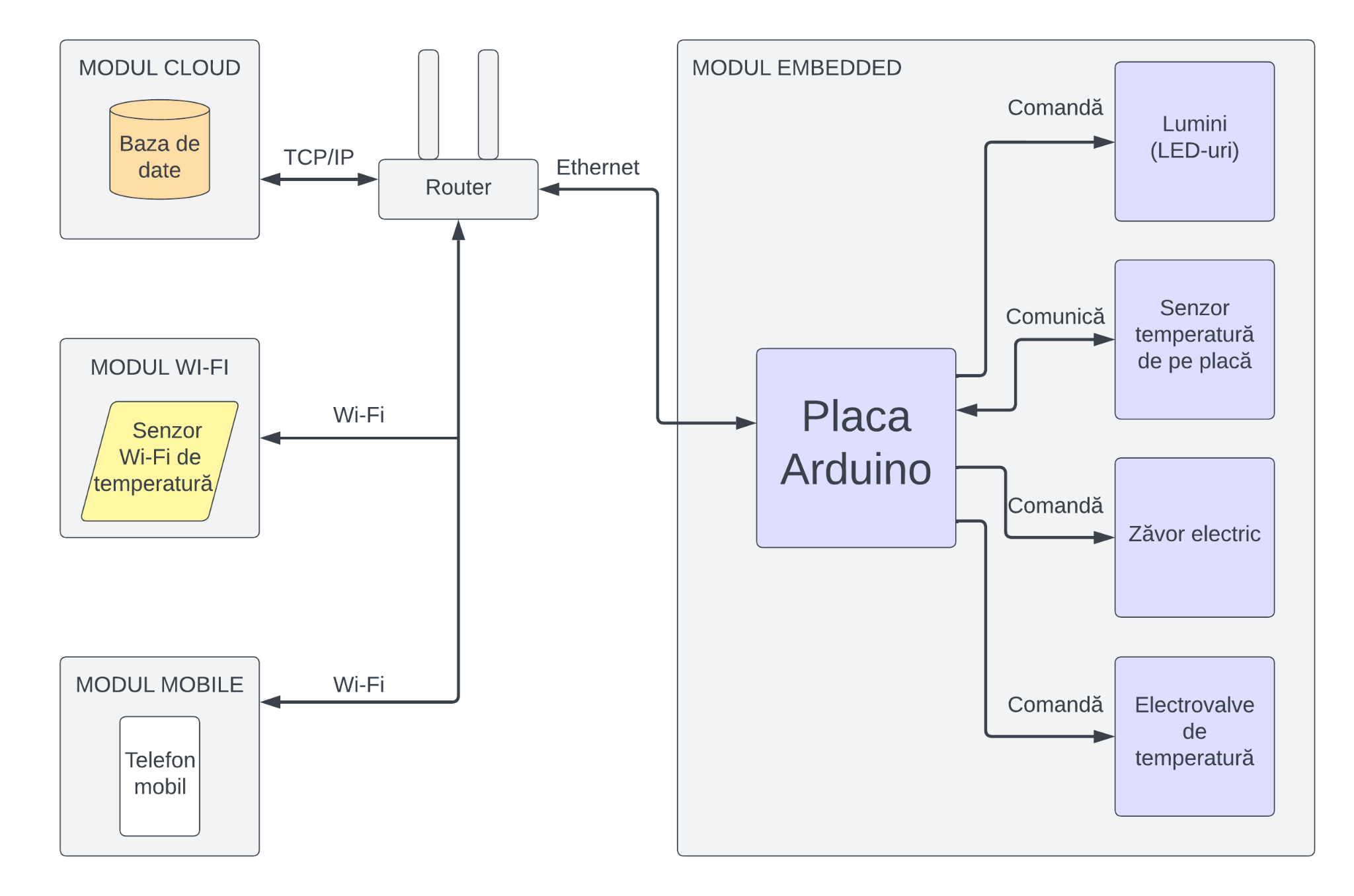


Figura 3. Arhitectura modulului Embedded

Modulul Embedded are la bază placa Arduino, care comandă, respectiv comunică, cu componentele fizice (lumini, electrovalve, zăvor). Modulul comunică cu restul modulelor prin 2 protocoale: Ethernet si Bluetooth.

Prin Ethernet, modulul trimite informații către router, informații destinate modulului Cloud prin protocolul TCP/IP și informații destinate modulului Mobile prin protocolul Wi-Fi. Prin intermediul Ethernet-ului, modulul Embedded primește si temperatura de la modului Wi-Fi, tot prin intermediul routerului.

Modulul are două relații cu procesele sale: comandă și comunicare. Prin relația de comandă, placa Arduino trimite semnale către componente, iar prin procesul de comunicare aceasta trimite un semnal către componente de la care primește un răspuns.

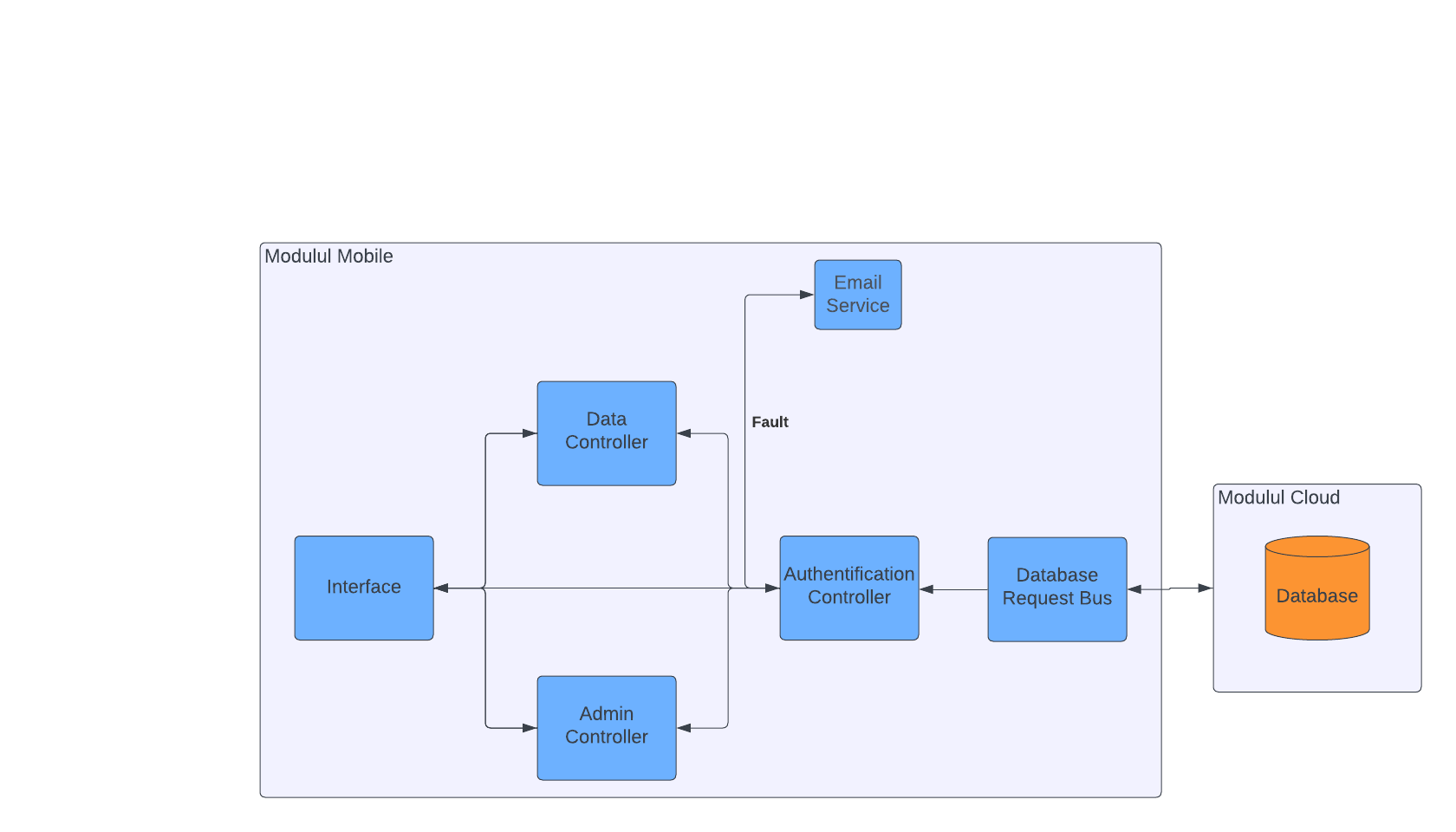


Figura 4. Arhitectura modulului Mobile

Modului Mobile face posibilă accesarea aplicației de pe dispozitivele mobile. Arhitectura modulului Mobile se poate observa în Figura 4.

Aceasta are la baza 3 unități de comandă (Authentication Controller, Data Controller și Admin Controller). Authentication Controller stabilește drepturile utilizatorului actual (User sau Admin) sau trimite un mesaj de eroare prin intermediul Email Service. Data Controller reprezintă unitatea de control specifică clientului prin care acesta poate modifica parametrii, respectiv de a alege sau creea Preset-uri. Admin Controller reprezintă unitatea de control specifică adminului, care monitorizează activitatea fiecărui client și poate adăuga/șterge clienți. Data Controller și Admin Controller se află în relație de dependență cu Authentication Controller.

Conexiunea Modulului Mobile cu Modulul Cloud se face prin protocolul HTTP. Interfața este adaptata dispozitivelor mobile dispunând de butoane mari și comenzi ușor accesibile.

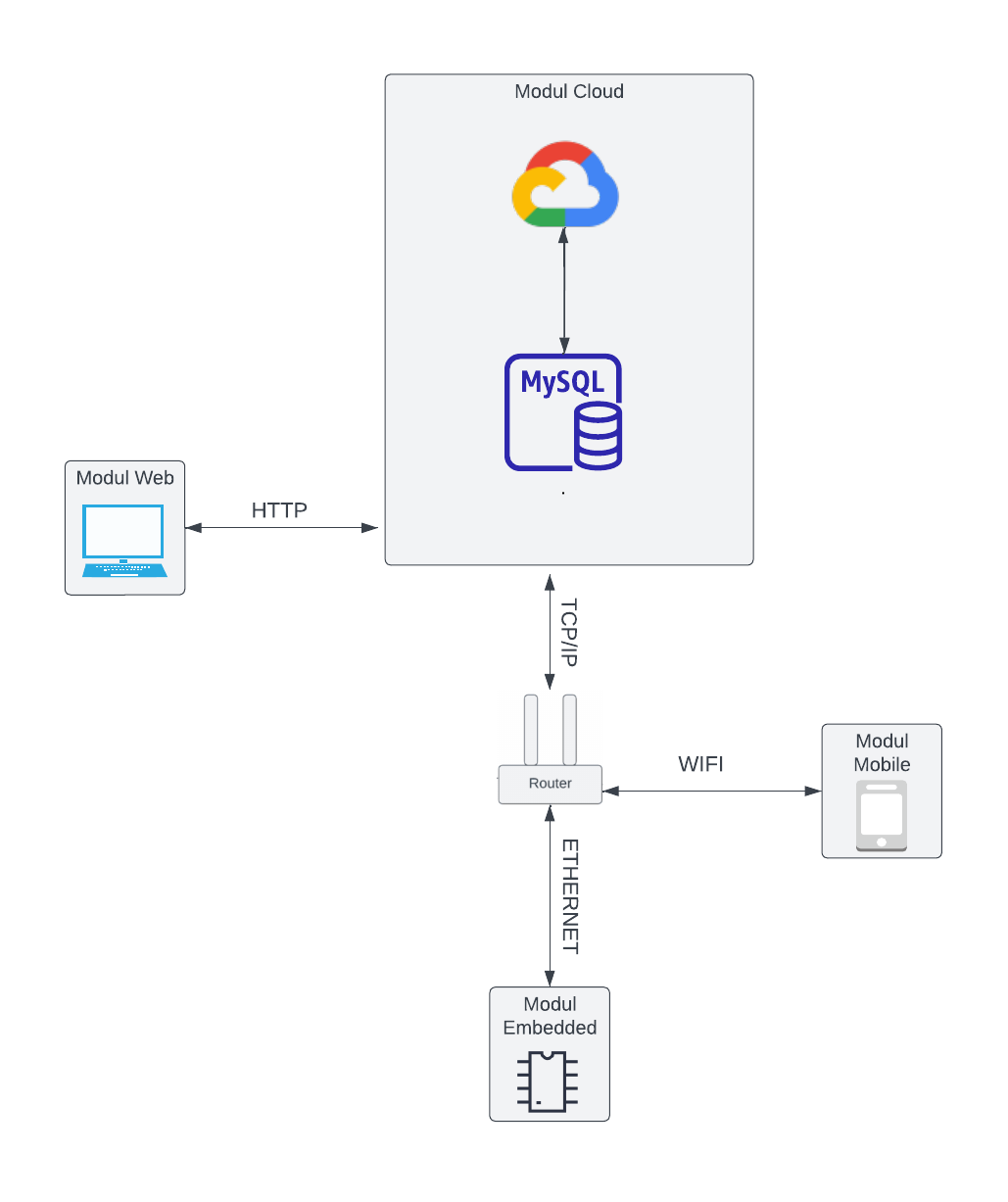


Figura 5. Arhitectura modulului Cloud

Prin intermediul aplicației mobile sau web, utilizatorul poate accesa și gestiona informațiile stocate în Cloud de oriunde și în orice moment, putând modifica valorile de temperatură și statusul luminilor. Atunci când acesta dorește să controleze dispozitivele din sistemul de home automation prin intermediul modulului cloud, aplicația mobilă sau interfața web transmite comenzi către modulul cloud prin intermediul internetului. Modulul cloud procesează apoi comanda și o transmite dispozitivelor corespunzătoare din rețeaua locală prin intermediul routerului.

Pentru comunicarea dintre modulul cloud și router se va folosi protocolul TCP/IP (Transmission Control Protocol/Internet Protocol). TCP/IP este un protocol de comunicație fiabil, care permite transferul de date prin intermediul rețelelor de computere. Protocolul TCP asigură livrarea fiabilă a datelor prin intermediul unei conexiuni de rețea și garantează că datele ajung la destinație în ordinea corectă. Protocolul IP asigură adresarea și rutarea datelor prin intermediul rețelelor de calculatoare, ceea ce permite comunicația între diferite dispozitive din rețea.

Comunicarea dintre modulul cloud și aplicația mobilă sau interfața web poate fi realizată prin intermediul unei conexiuni Wi-Fi și utilizând protocolul TCP/IP. Aplicația mobilă se conectează la rețeaua Wi-Fi la care este conectat și modulul cloud, permițând astfel transmiterea și primirea de date între cele două prin intermediul protocolului TCP/IP.

Comunicarea dintre modulul cloud și modulul embedded se face prin intermediul unei conexiuni Ethernet și a protocolului TCP/IP. Modulul embedded poate iniția comunicarea prin trimiterea de solicitări TCP către modulul cloud, iar acesta va putea să le răspundă cu informațiile cerute sau cu confirmarea acțiunilor efectuate.

În comunicarea dintre modulul web și modulul cloud, se utilizează protocolul HTTP pentru a trimite solicitări către modulul cloud și pentru a primi datele cerute sau confirmarea acțiunilor efectuate.

# 2. Diagrama de secvență pentru întregul sistem

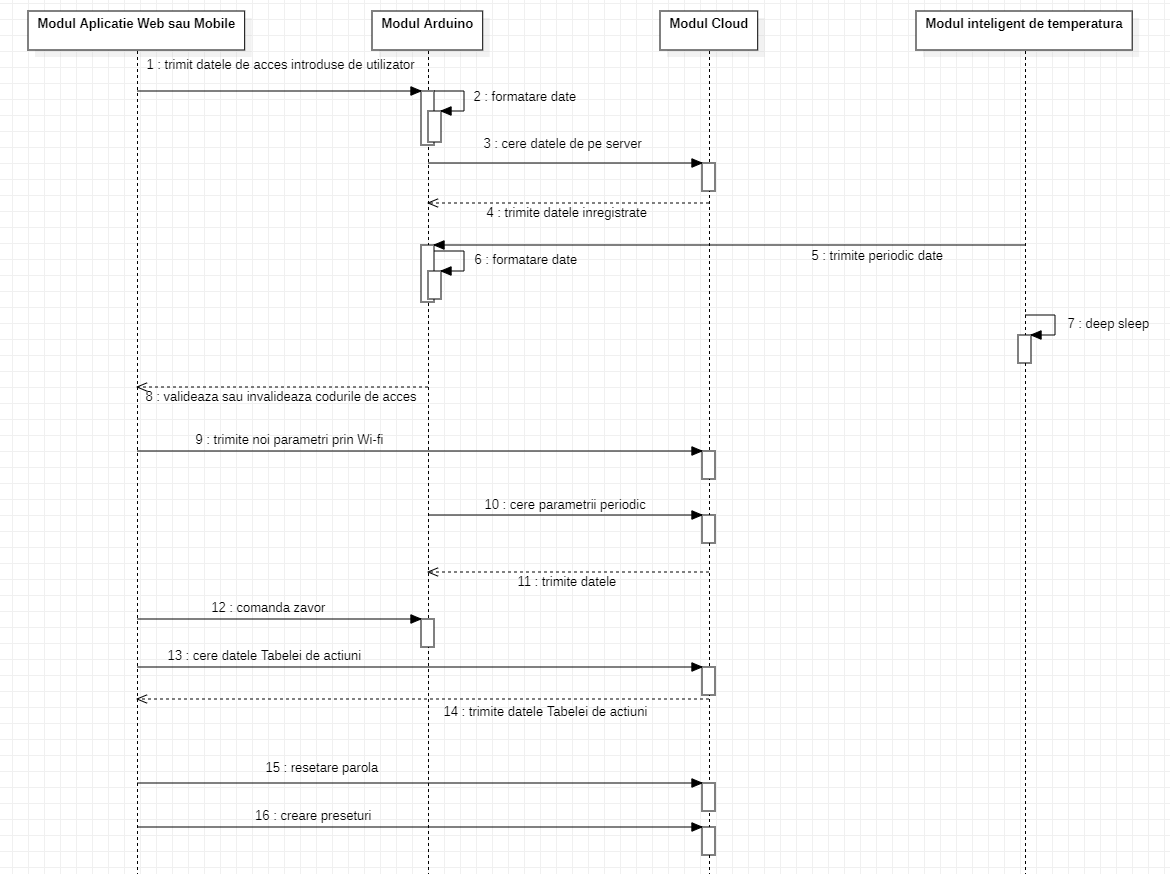


Figura 6. Schema secvențială a sistemului

Primul proces care se realizează este cel de Autentificare, unde utilizatorul va introduce datele de conectare (email și parolă) în aplicația Mobile sau pe site-ul Web. Acestea sunt trimise prin Cloud la Modulul Arduino, unde sunt formatate. Se va face o cerere, de către Modulul Arduino, la Cloud pentru date deja existente de conectare. După ce se va realiza compararea pe Arduino(validează sau nu datele), răspunsul este trimis înapoi în aplicație.

Pentru aplicația Smartphone/site-ul Web, sunt prezentate următoarele funcționalități: utilizatorul trimite noi parametri(prin Wi-fi) către Cloud, de unde sunt cerute de către Modulul Arduino; comandă zăvor prin Bluetooth; cere datele Tabelei de Acțiuni din Cloud pentru afișarea istoricului personal, urmând să primească ca răspuns de la Modulul Cloud aceste date; cere resetarea parolei, acțiune care va modifica parola din tabela cu utilizatori din Cloud; crează preseturi, acțiune care va creea o nouă înregistrare în tabela cu evenimente predefinite din Cloud.

În cazul Modulului inteligent de temperatură, sunt trimise periodic date de temperatură spre Modulul Arduino, unde sunt formatate de acesta, urmând apoi ca primul modul să intre în deep sleep.

# 3. Scheme logice

În aplicația noastră vor exista câteva mecanisme prin care se vor face verificări necesare identificării utilizatorului, construirea de mesaje codificate corespunzătoare preferințelor acestuia în vederea funcționării sistemului ce vor fi transmise în cloud, achiziția periodică de date de la echipamentele din locuință în vederea monitorizării parametrilor de interes, și funcționarea sistemului de Deep Sleep. Aceste mecanisme sunt ilustrate în figurile de mai jos și explicate la punctul 7 al acestui document:

Codul Device ID

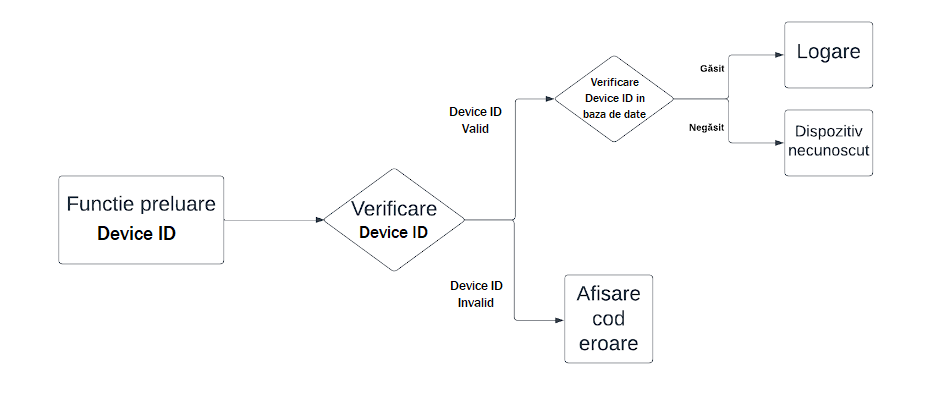


Figura 7. Gestionare Device ID

Codificarea și decodificarea opțiunilor unui preset

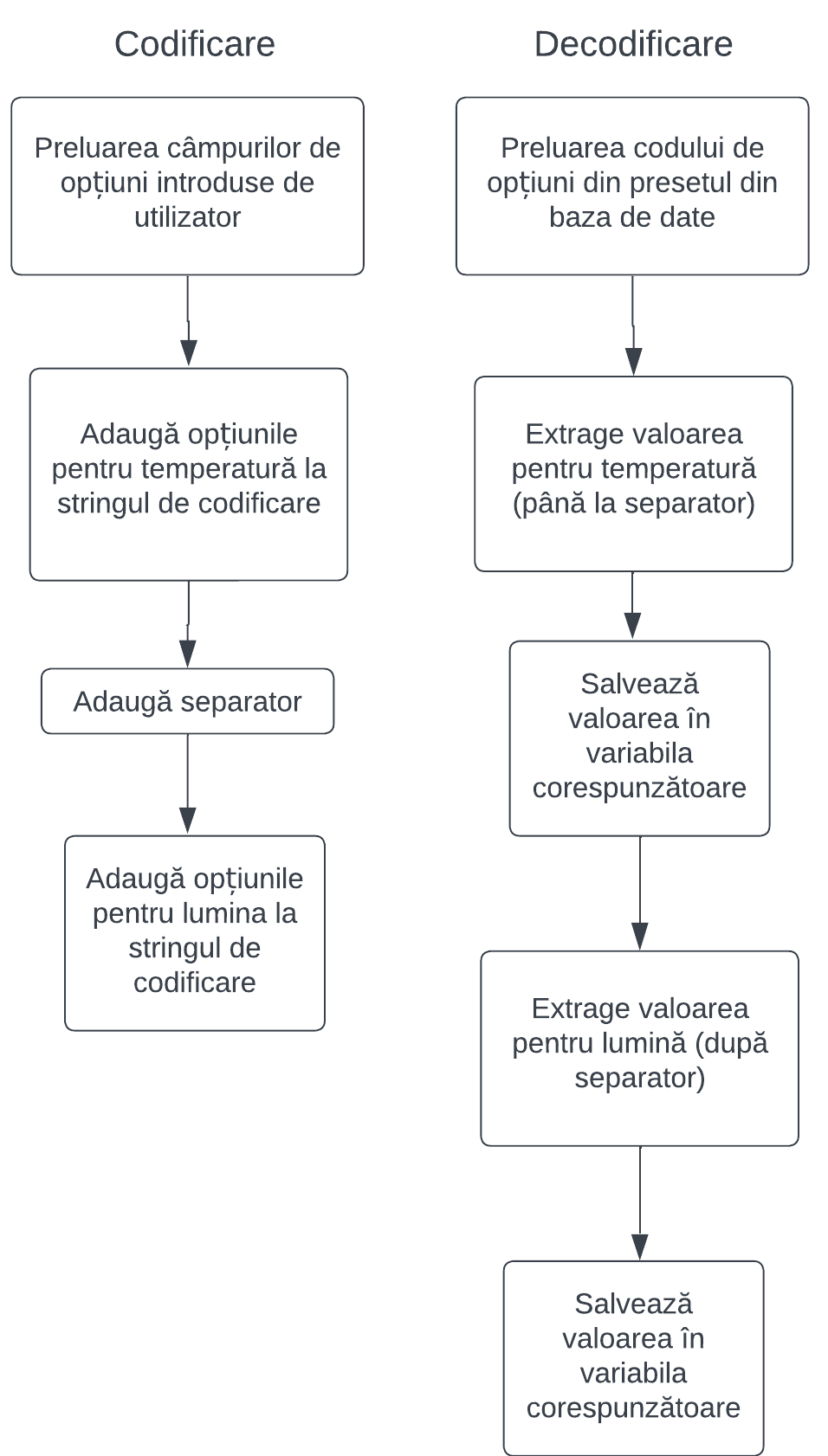
****

Figura 8. Schema logică aferentă codificării/decodificării opțiunilor

Deep Sleep

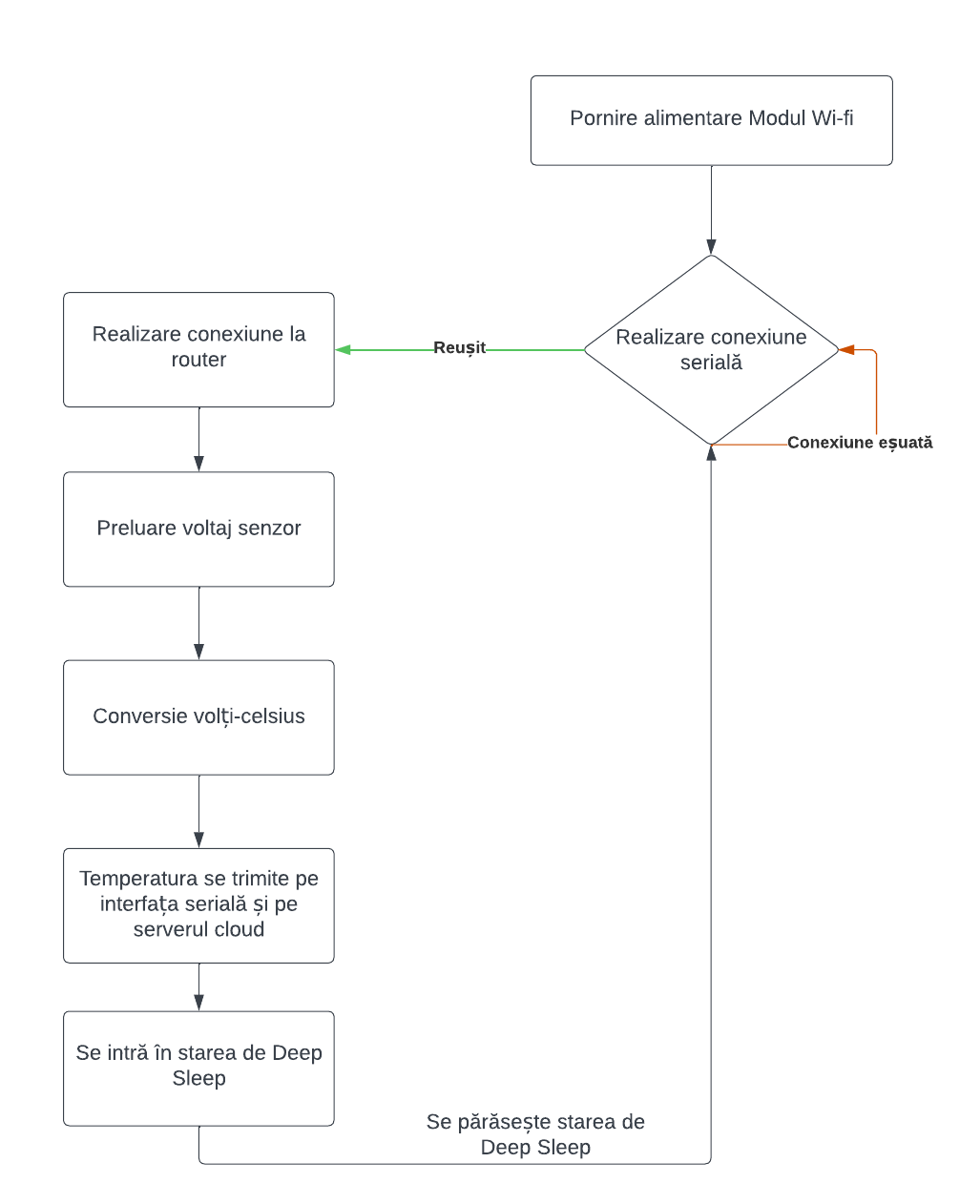
****

Figura 9. Schema logică Deep Sleep

# 4. Diagrame de clase

Modul Web

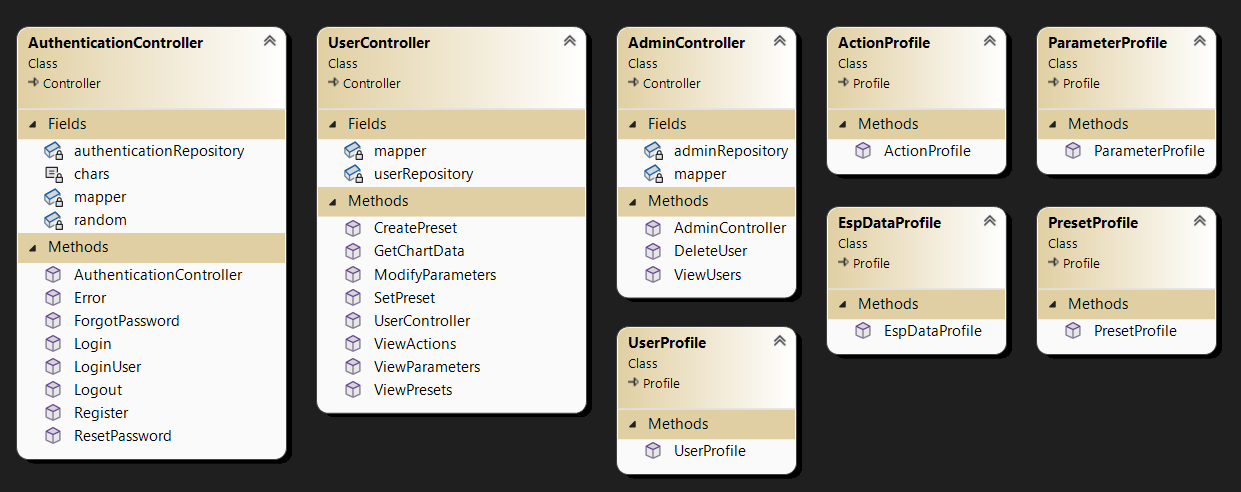


Figura 10. Clasele controller și clasele profil

Cele trei clase controller se vor ocupa de popularea view-urilor cu date și de gestiunea cererilor inițiate de către utilizator din view-uri, întrucât ele asigură legătura cu baza de date prin comunicarea cu clasele repository corespunzătoare.

Clasa AuthenticationController va fi responsabilă pentru acțiunile de login, register, resetare a parolei și logout. Atunci când utilizatorul inițiază una dintre aceste acțiuni, fie prin completarea unui formular sau apăsarea unui buton, metoda vizată va primi, dacă este cazul, modelul cu informația specifică și îl va converti la clasa cu care lucrează metoda corespunzătoare din clasa repository. Apoi se va aștepta răspunsul de la metodă în funcție de care se vor afișa mesaje de succes, eroare, se vor face redirecționări către alte pagini (cum ar fi după login, atunci când utilizatorul va fi redirecționat către pagina pentru vizualizarea parametrilor actuali).

Clasa UserController va conține metodele comune pentru utilizatorii simpli și pentru administrator, accesul la acestea fiind posibil doar după autentificare și autorizare, lucru realizat cu ajutorul adnotarii [Authorize] din clasa Microsoft.AspNetCore.Authorization. Clasa UserController se va folosi de câmpul userRepository de tip IUserRepository pentru a realiza comunicarea cu baza de date și câmpul mapper de tip IMapper pentru maparea către entitățile cu care lucrează clasa UserRepository.

Clasa AdminController conține metodele specifice administratorului, și anume vizualizarea și ștergerea utilizatorilor, accesul fiind restricționat tot prin atributul [Authorize] din clasa Microsoft.AspNetCore.Authorization, la care se adaugă opțiunea (Roles=”ADMIN”). La fel ca restul claselor controller, acesta se va folosi de un repository corespunzător și de un mapper.

Clasele profil sunt construite cu ajutorul pachetului AutoMapper, toate moștenesc clasa “Profile” cu ajutorul căreia se pot construi hărți pentru maparea între clase. Rolul lor este să ajute la construirea claselor ce vor expune doar informația necesară către utilizator și a claselor care cuprind informația necesară pentru trimiterea unei cereri către controller. Clasele respective sunt clasele model din figura 11 care vor fi folosite în view-uri. De exemplu, pentru entitatea Parameter nu se dorește afișarea câmpului Row\_Id din baza de data așa ca se va construi o hartă către clasa ParameterModel unde există toate câmpurile din Parameter, mai puțin cel de Row\_Id.

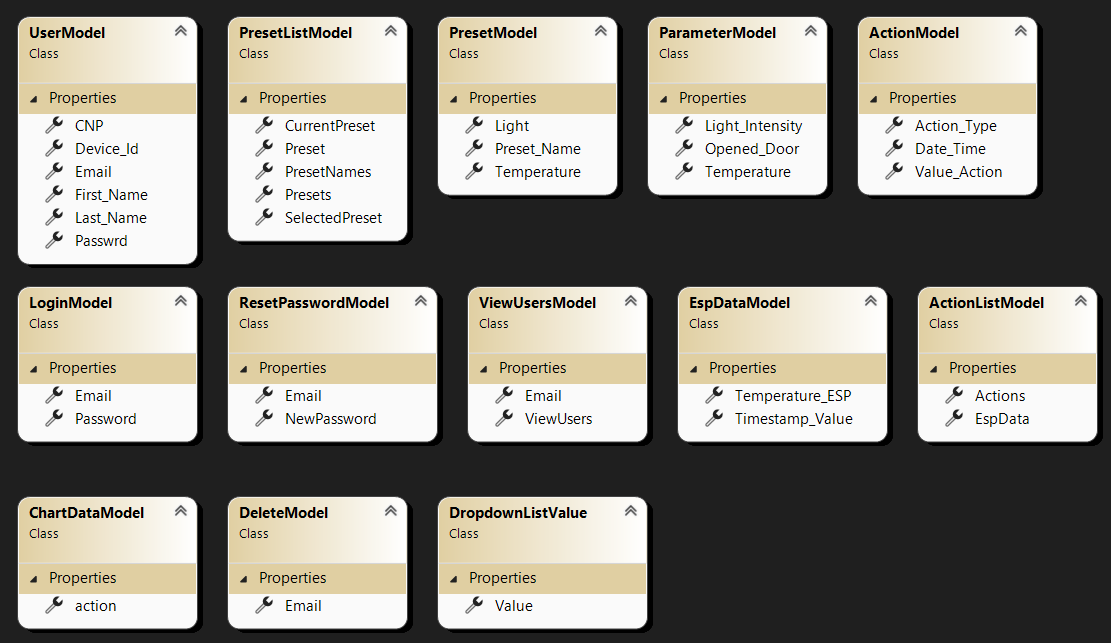


Figura 11. Clasele model

Clasele model vor transmite informația de la view-uri către controllere și din controllere către view-uri, fiind mapate către și din entitățile din baza de date, dacă este cazul. De exemplu, un obiect de tip ActionListModel va fi populat cu o listă de obiecte de tip ActionModel și o lista de obiecte de tip ESPDataModel la accesarea paginii pentru vizualizarea acțiunilor, liste ce vor fi afișate în tabele.

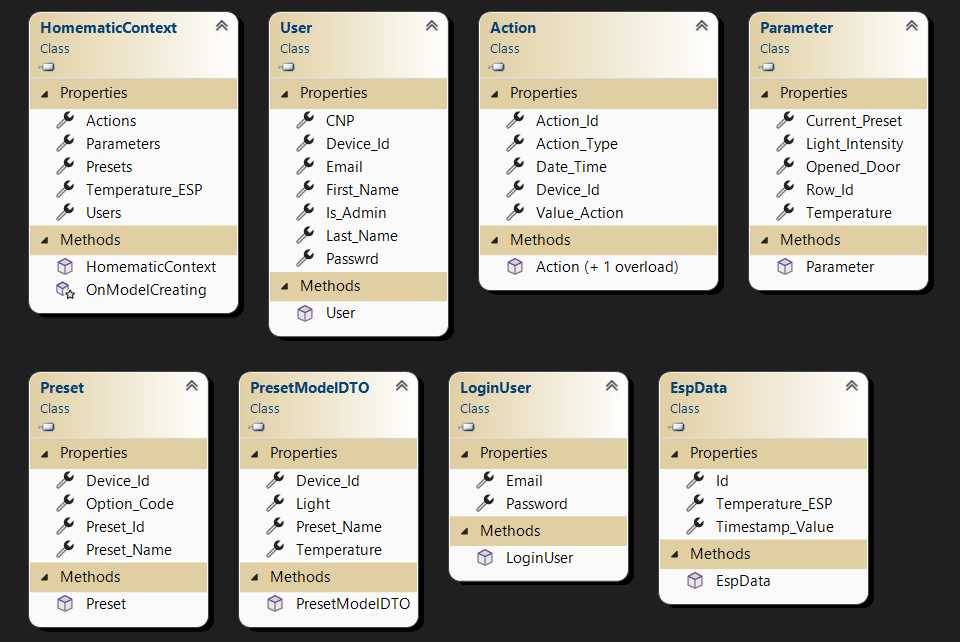


Figura 12. Clasa context, clasele entități din baza de date și clasele DTO

HomematicContext va moșteni clasa DbContext din pachetul EntityFrameworkCore, ajutând la manipularea facilă a tabelelor și înregistrărilor din baza de date și la realizarea legăturii dintre tabele și clasele entități corespunzătoare prin adăugarea câmpurilor de tip DbSet de tipul entității vizate (User, Parameter, Action, Preset sau EspData).

User, Parameter, Action, Preset și EspData sunt clasele corespunzătoare înregistrărilor din baza de date. PresetModelDTO și LoginUser sunt două clase intermediare, necesare datorită unor prelucrări înainte de utilizare (decodificarea codurilor de opțiuni din obiectele preset și verificarea parolei).

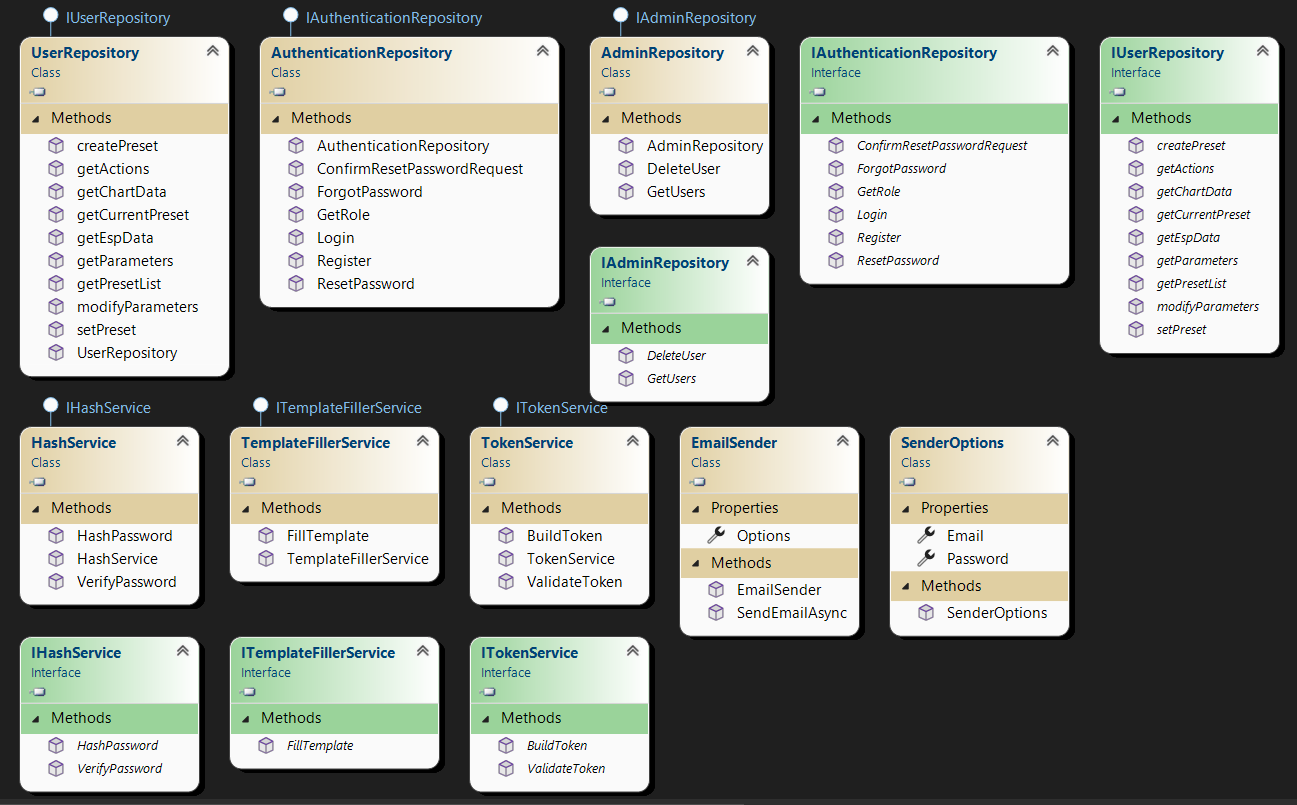


Figura 13. Clasele repository, clasele pentru servicii și interfețele acestora

În clasele repository se va realiza comunicarea propriu-zisă cu baza de date, prin intermediul unui câmp privat, marcat cu “readonly” de tip HomematicContext, injectat prin constructor. În clasele repository se vor găsi metode aproape în oglindă cu cele din controllere, cu diferența că în clase repository se va implementa logica în sine, iar rezultatul va fi transmis clasei controller corespunzătoare. Aici se vor adăuga, modifica, citi și șterge înregistrări din baza de date.

Clasele serviciu îndeplinesc responsabilități suplimentare ce nu țin de clasele controller sau repository. Va exista un serviciu responsabil de criptarea, respectiv verificarea parolei, un serviciu pentru trimiterea email-ului de resetare a parolei și unul pentru construirea șablonului HTML trimis pe mail cu acest mesaj. Clasa TokenService este probabil cea mai importantă, întrucât cu ajutorul ei se construiește tokenul pe baza căruia se realizează autentificarea și autorizarea. Metoda BuildToken va primi datele utilizatorului ce dorește să facă login, va prelua detaliile de construire a tokenului din fișierul appsettings.json și le va folosi pentru a crea un token unic cu valabilitate de 30 de minute.

Modul Mobile

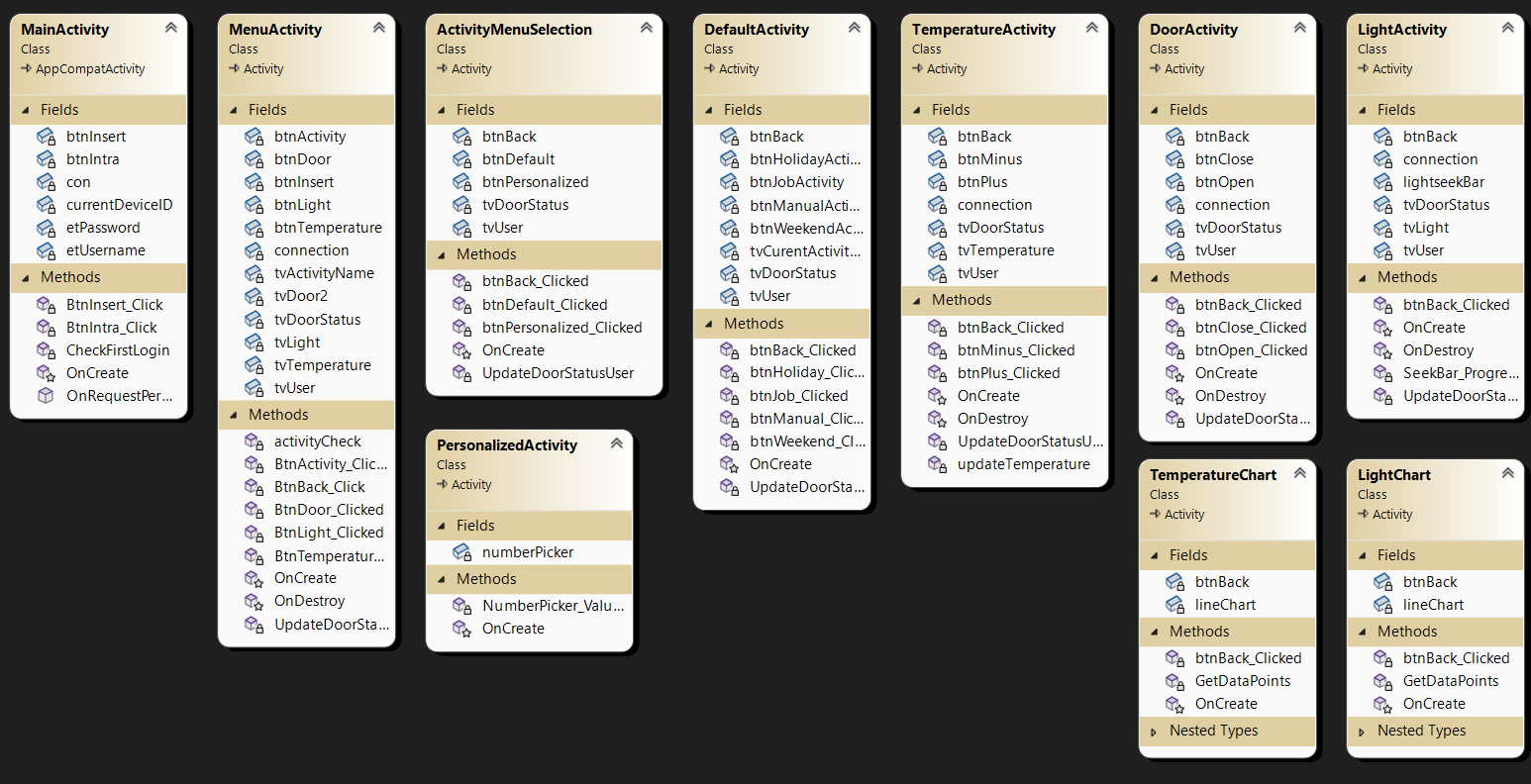


Figura 14.1. Diagrama claselor

Clasa MainActivity reprezinta partea care se ocupa cu logarea și verificarea datelor introduse, precum si a Device\_ID-ului.

Clasa MenuActivity cuprinde meniul principal al aplicației, unde se pot vizualiza parametri și se poate naviga prin aplicație.

Clasele PersonalizedActivity și DefaultActivity permit vizualizarea și selectarea activităților existente, respectiv crearea unor noi activități bazate pe cerințele clientului.

Clasele TemperatureActivity, DoorActivity si LightActivity, permit modificarea manuala a parametrilor aplicatiei.

Clasele TemperatureChart și LightChart permit vizualizarea modificarilor stocate in baza de date, in functie de ora.

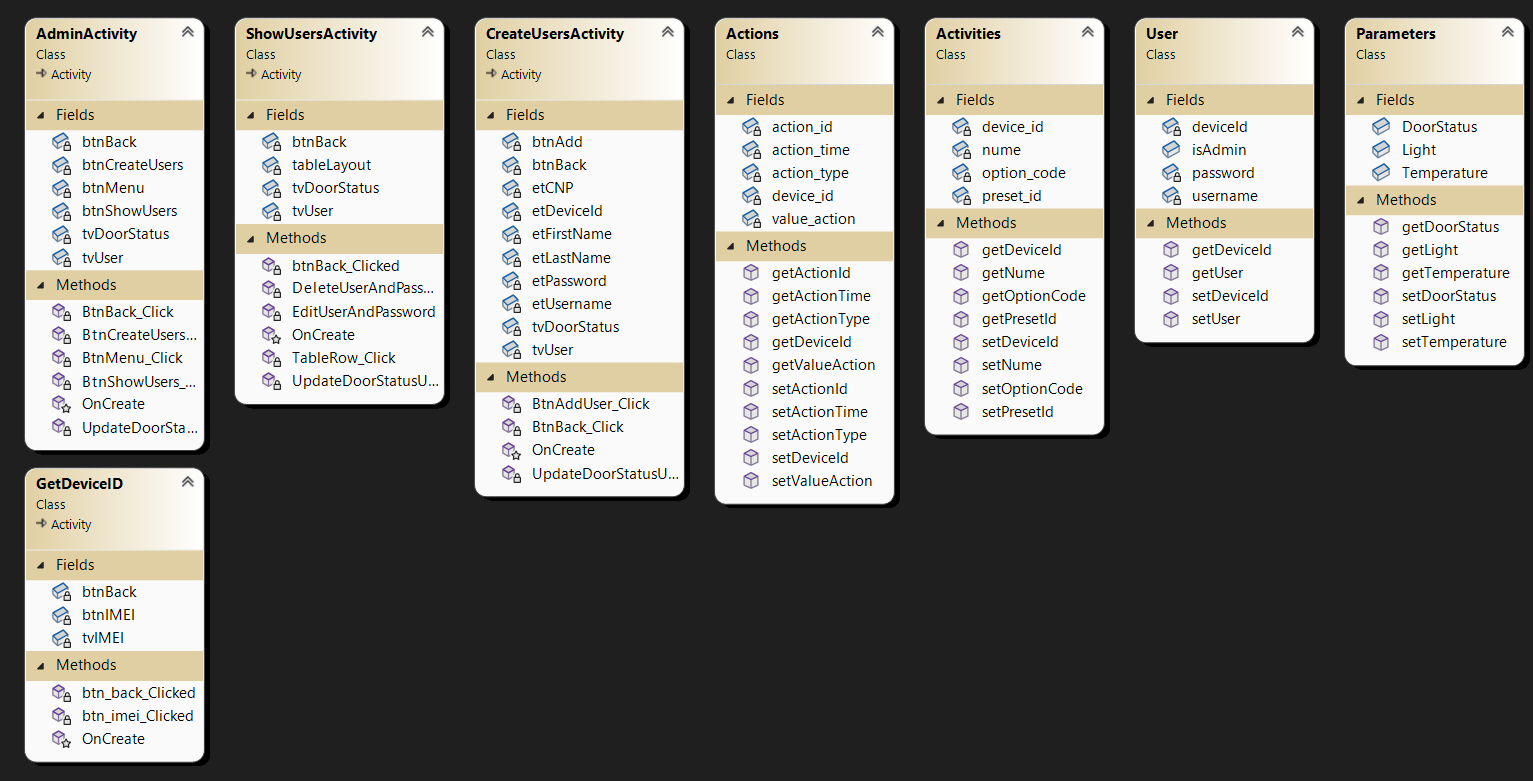


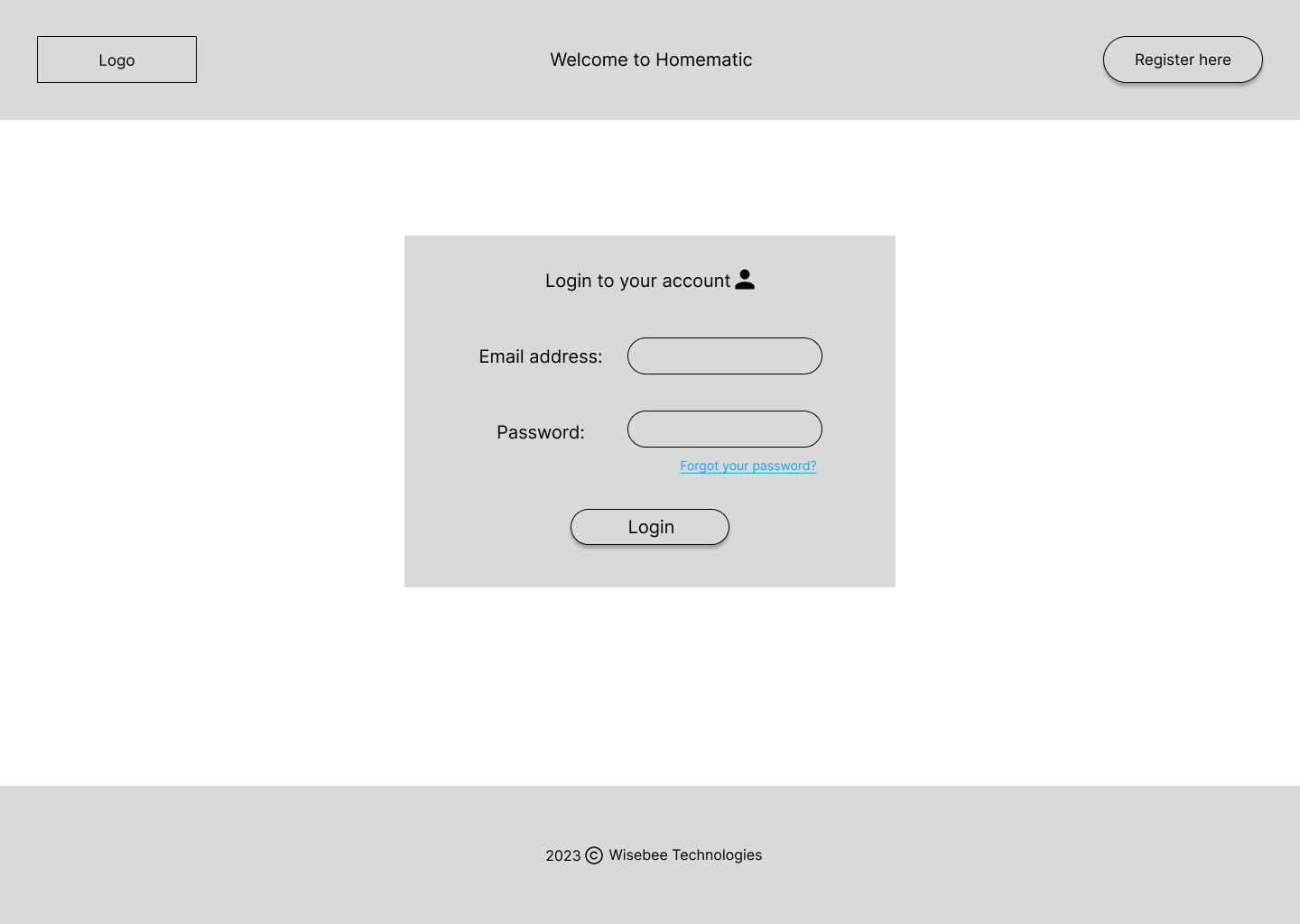
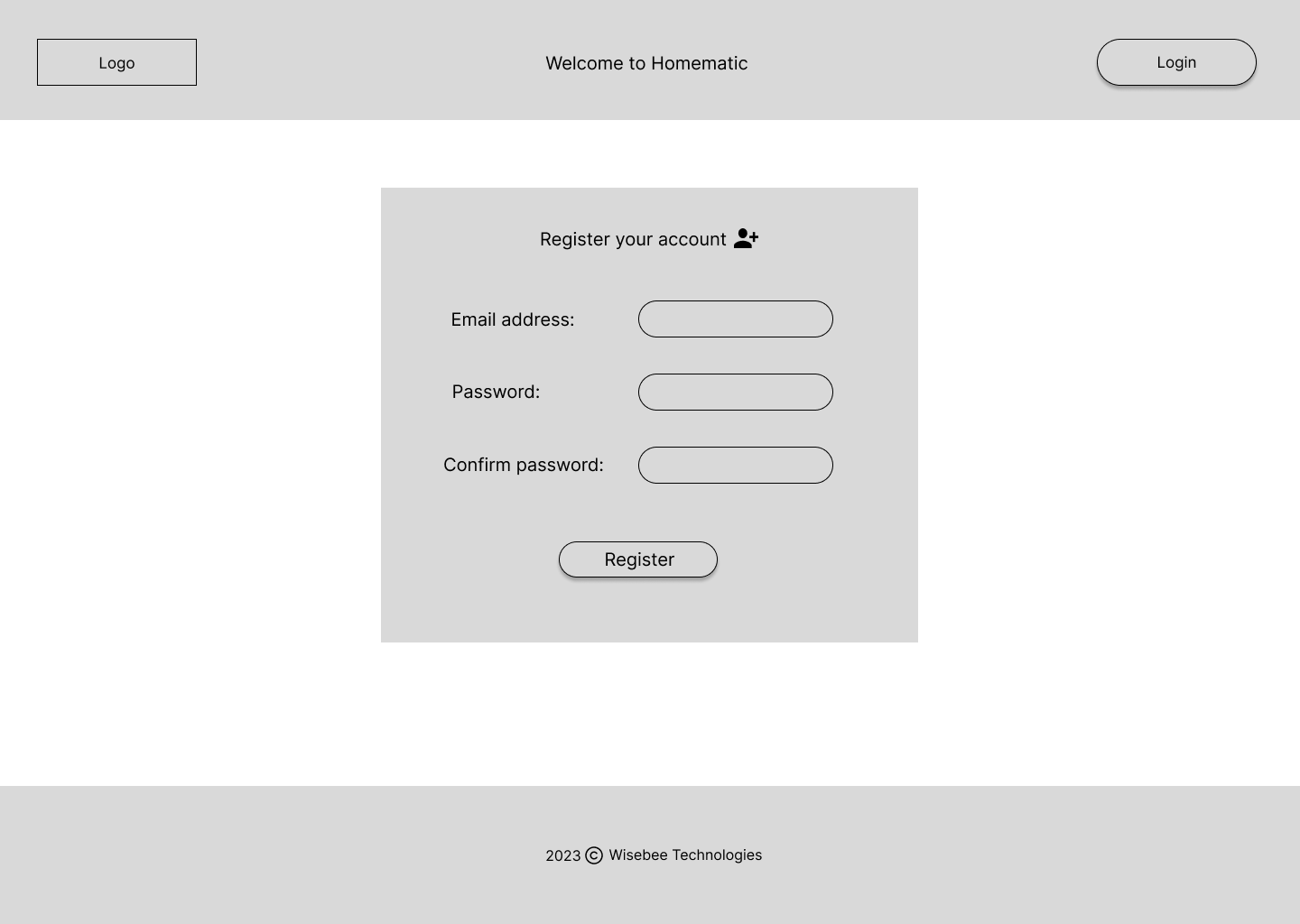
Figura 14.2. Diagrama claselor

Clasa AdminActivity permite navigarea prin aplicatie, ca administrator, și oferă accesul la funcții cum ar fi vizualizarea, modificarea sau stergere utilizatorilor (Clasa ShowUsersActivity) sau crearea unor utilizatori noi (Clasa CreateUserActivity).

Clasele Activities, Actions, User si Parameters se ocupa cu stocarea temporara a datelor in cadrul aplicatiei, înainte de trimiterea lor în baza de date.

# 5. Scheme cu interacțiunea în interfețe

Modul Web



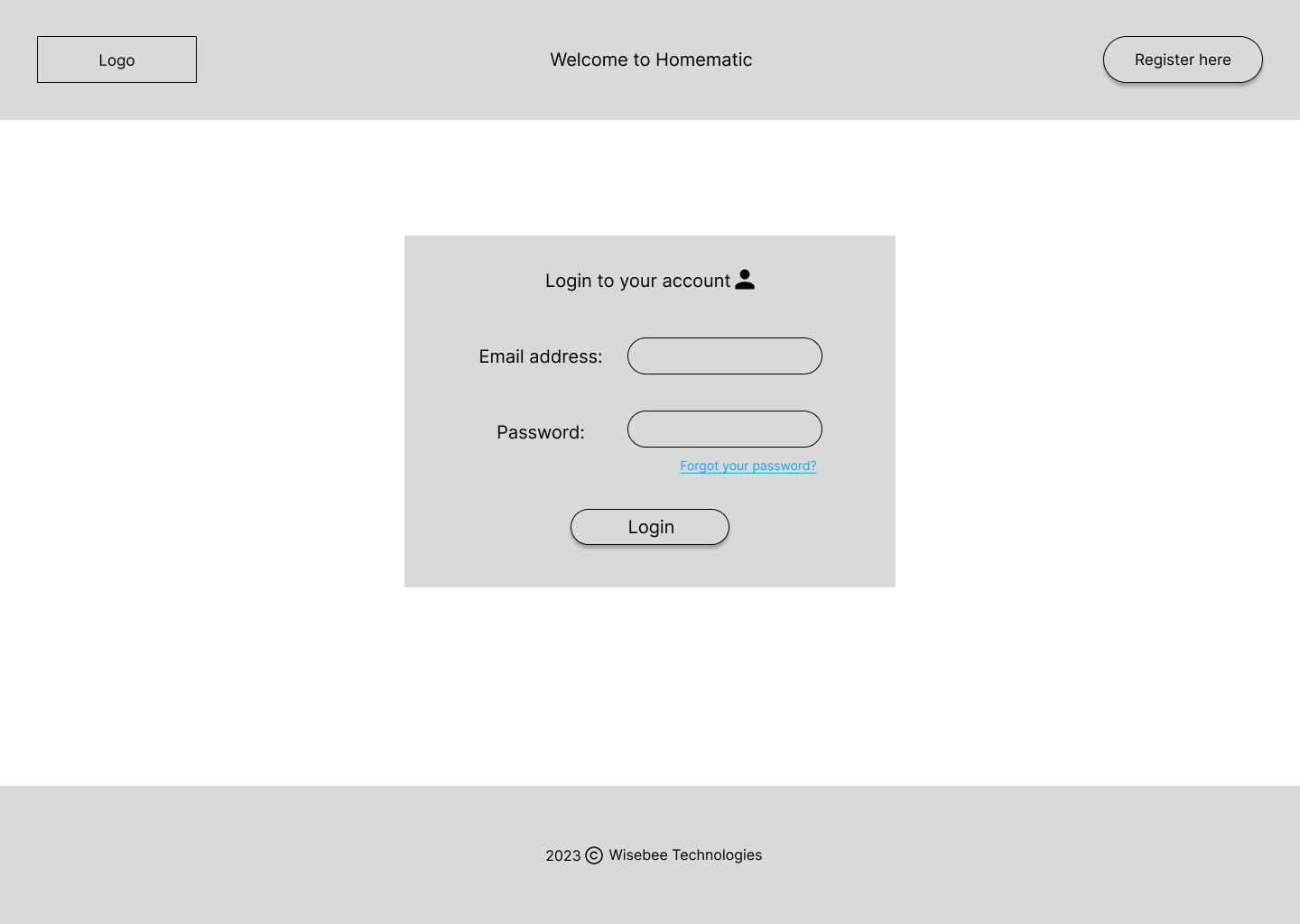
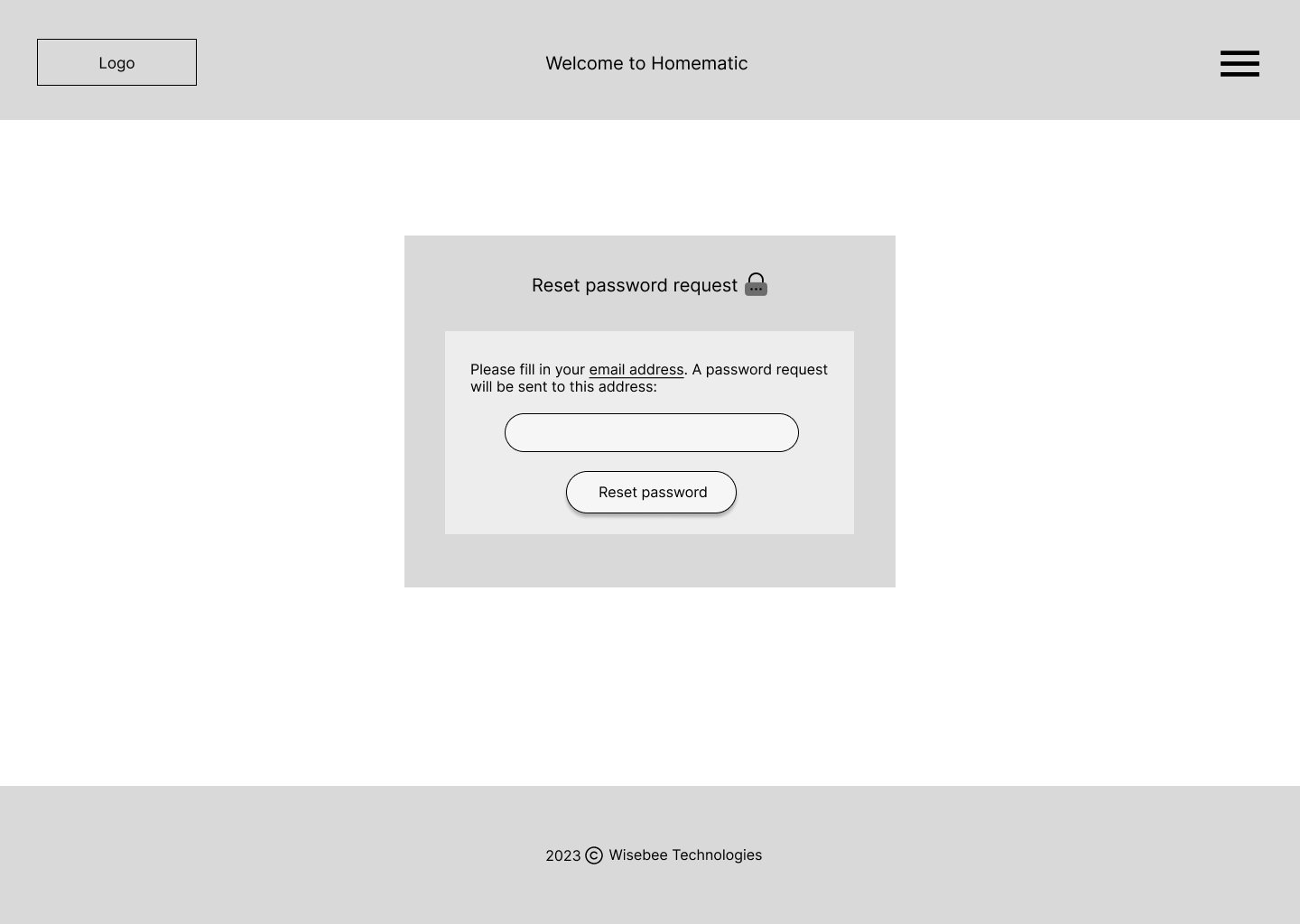
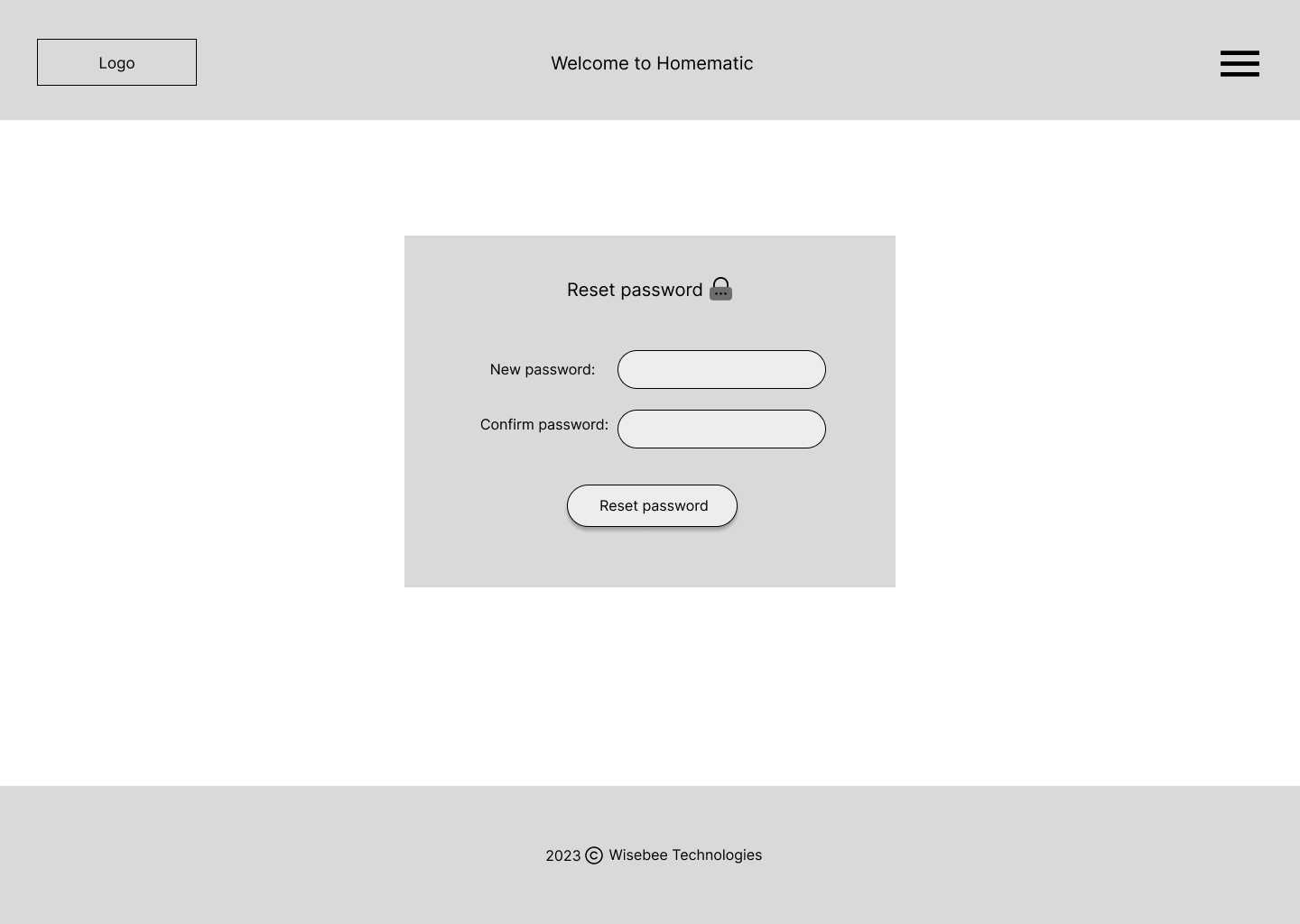


Figura 15. Resetarea parolei

În figura 15, este prezentat scenariul în care un utilizator dorește resetarea parolei. Prima interfață este o pagină de Register, de unde se trece în pagina de Login. Aici utilizatorul sau adminul se pot conecta. În cazul în care acesta va selecta opțiunea “Forgot your password”, se va deschide pagina de Request new Password, unde se va introduce email-ul. Următoarea interfață este aceea de Set new password, unde se va introduce o parolă nouă și se va reveni la pagina de Login.

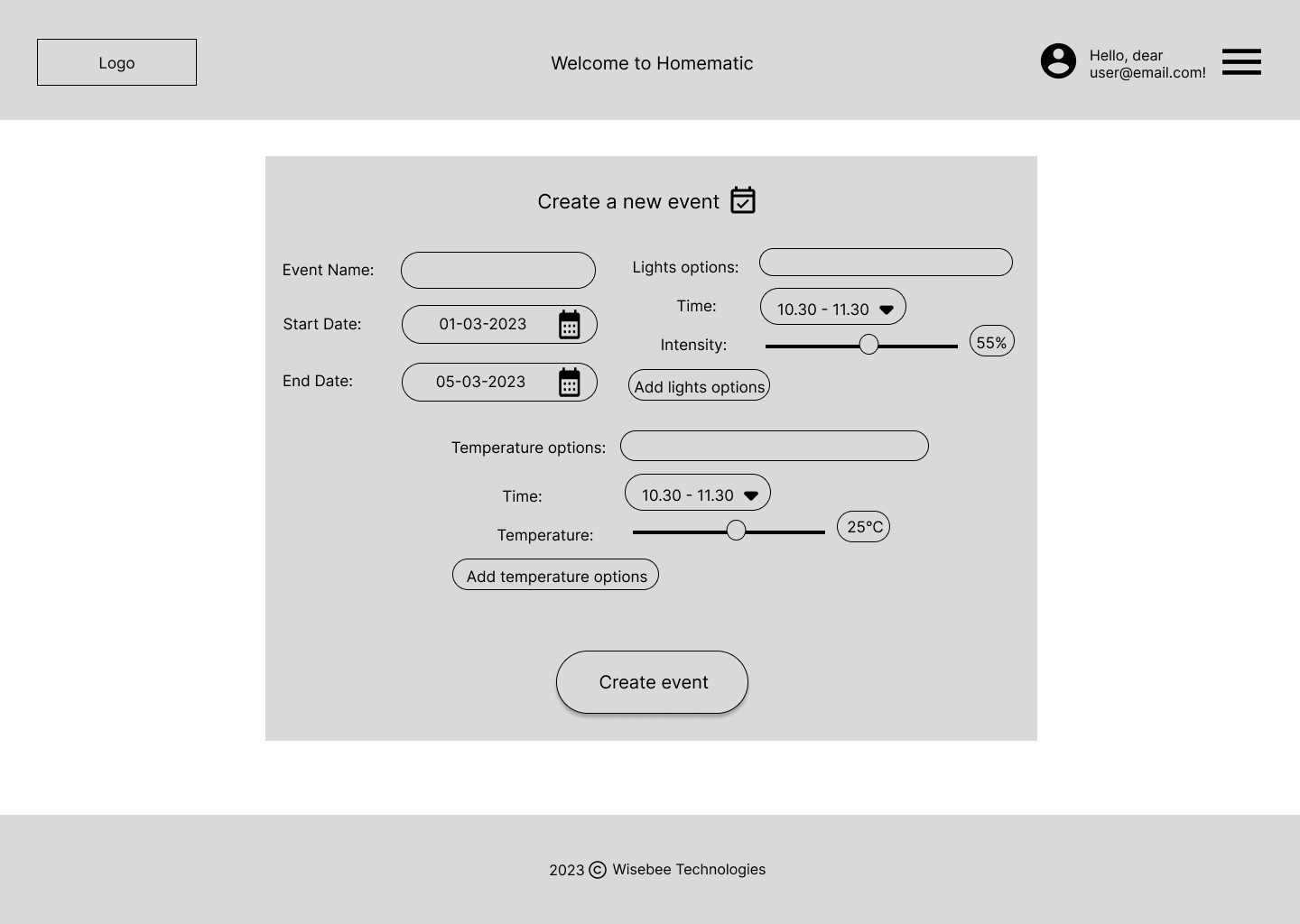
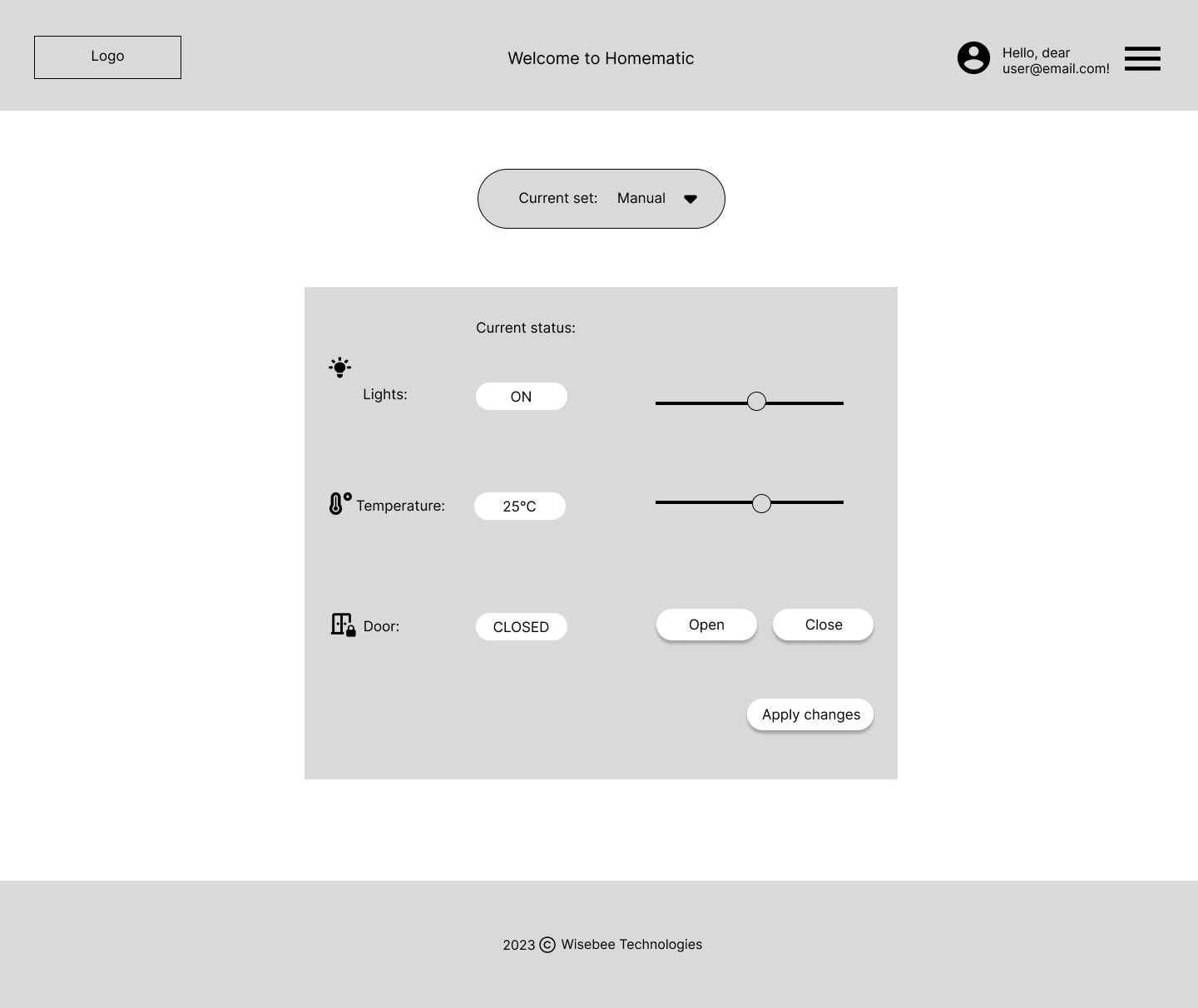
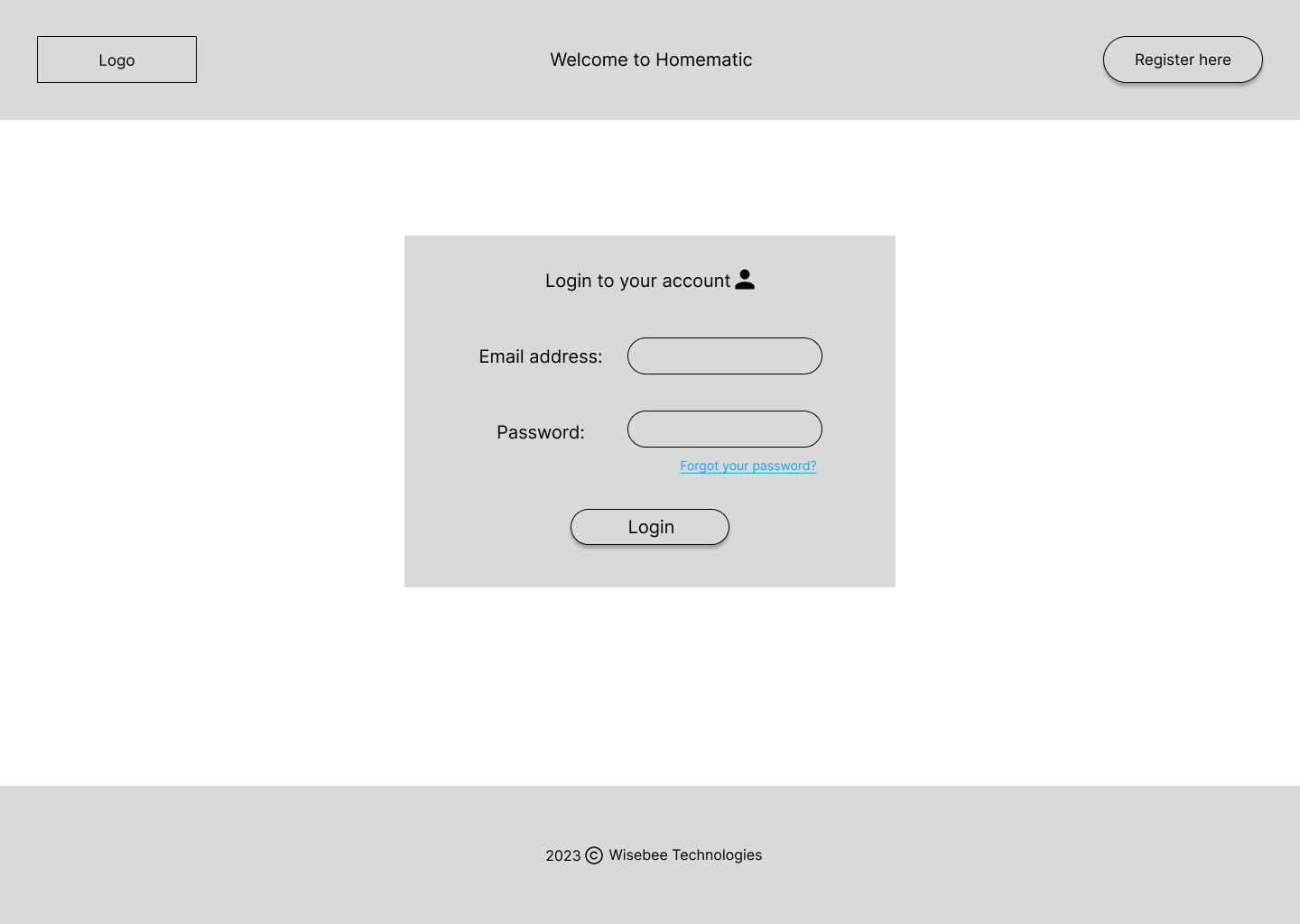
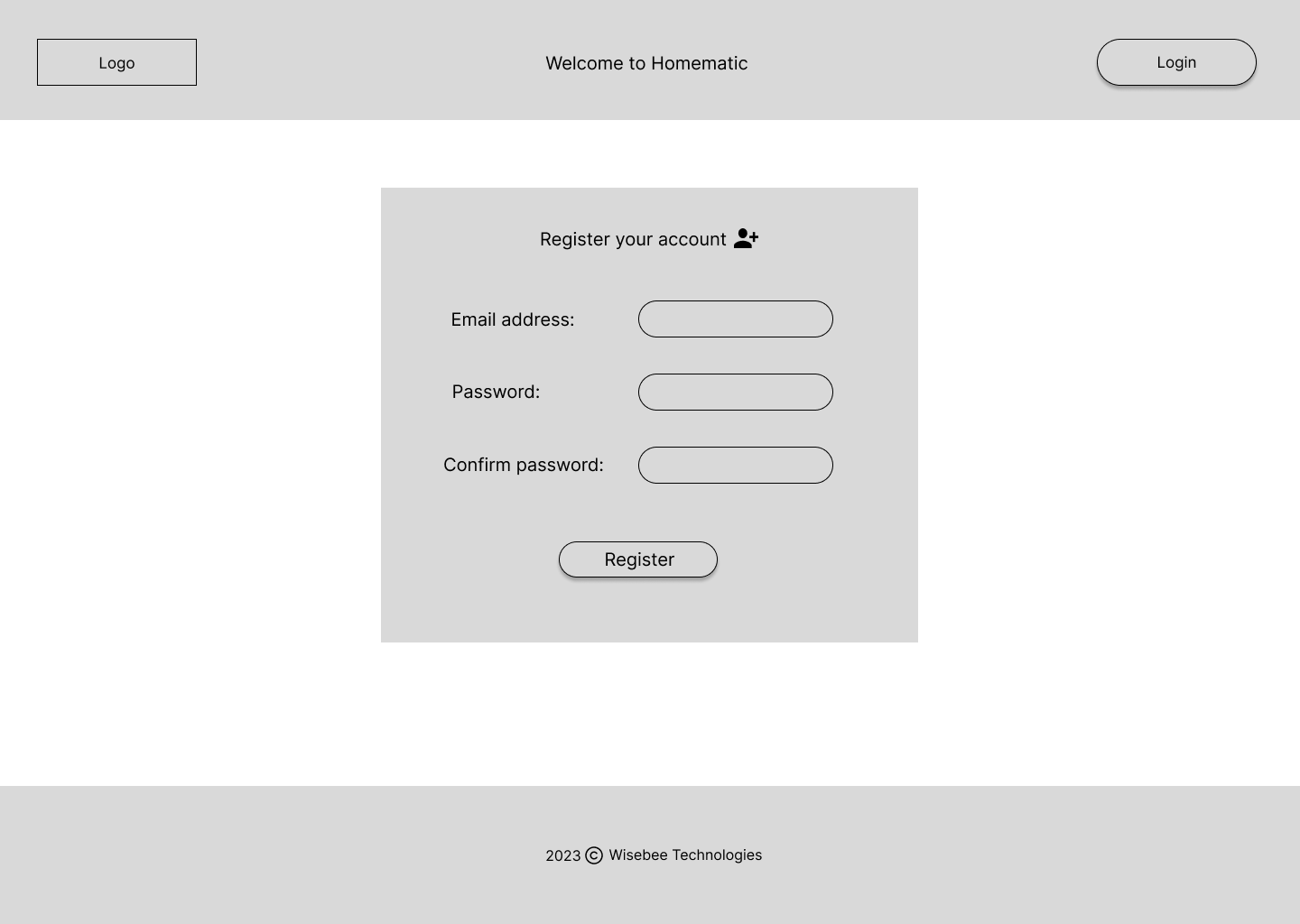
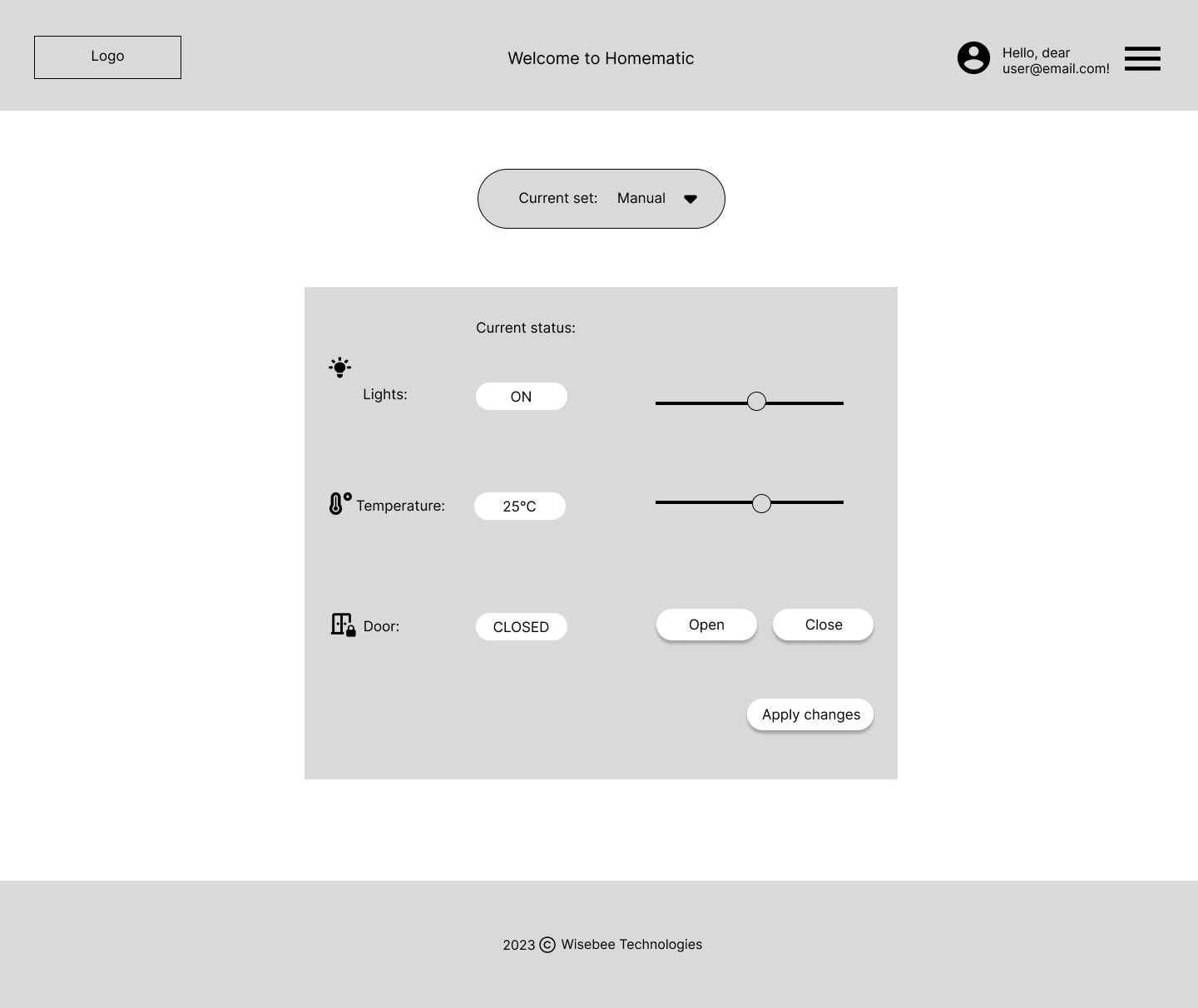
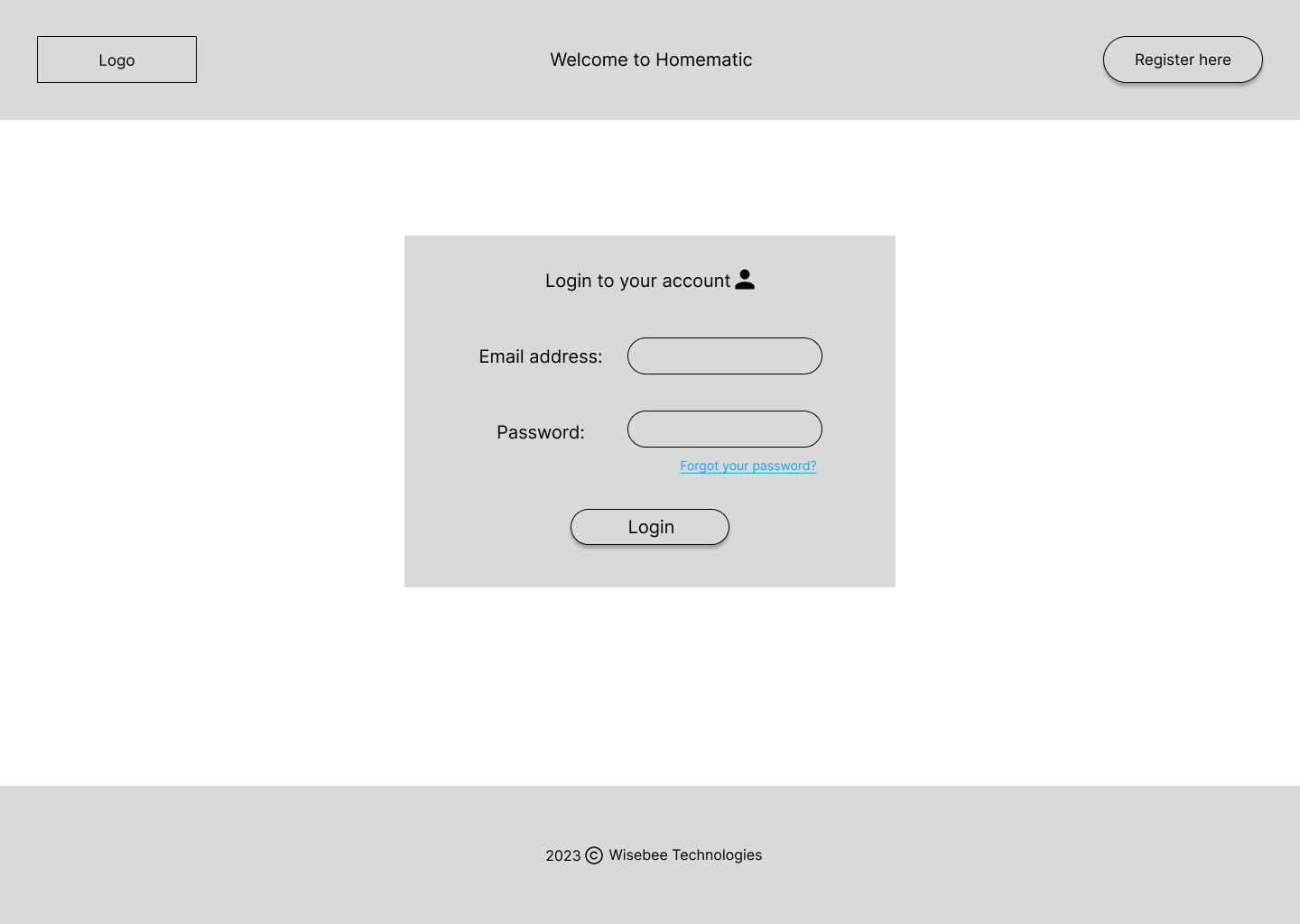
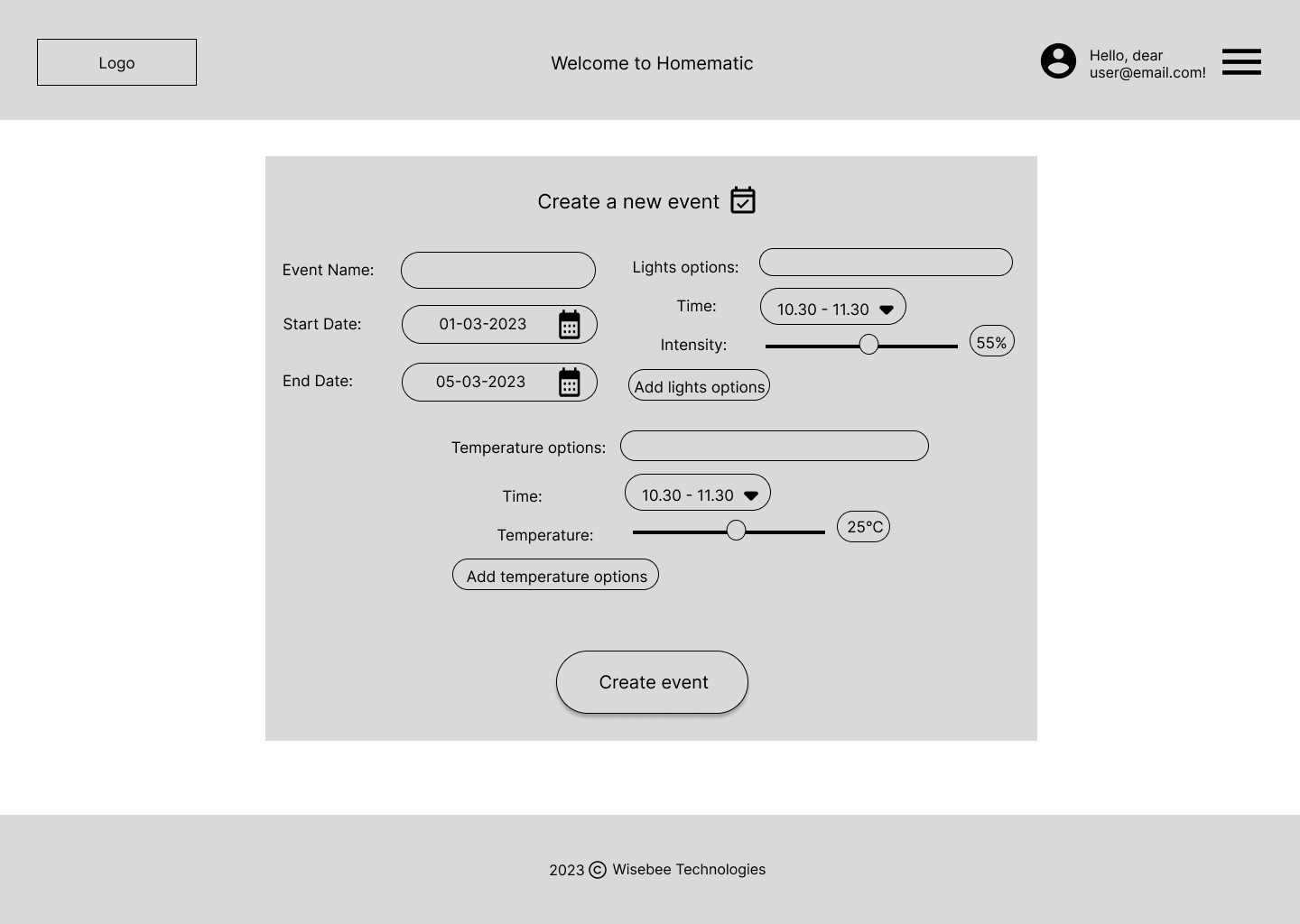
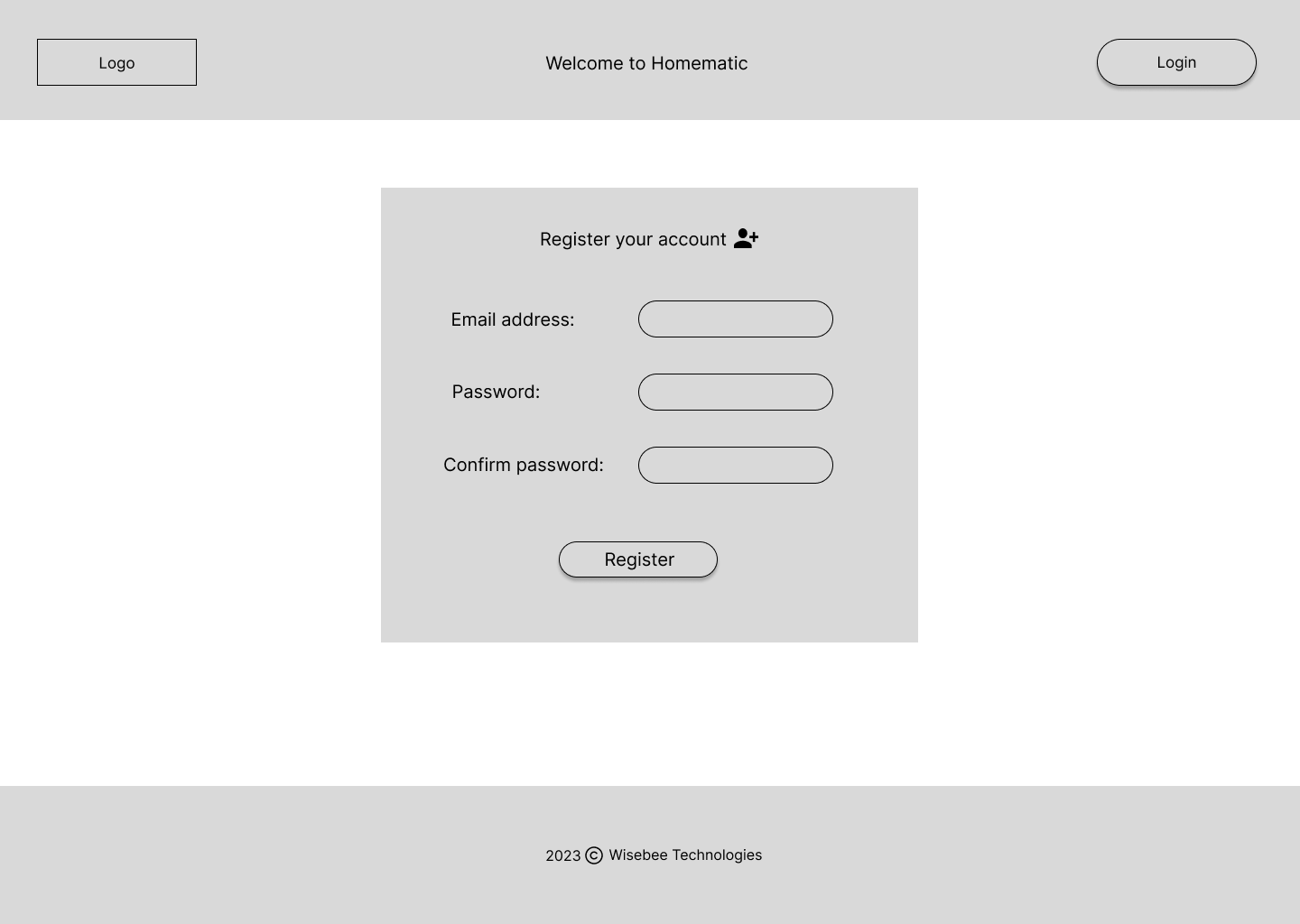




Figura 16. Interacțiunea interfețelor pentru utilizator

În Figura 16, este prezentat modul în care interacționează interfețele site-ului Web pentru un utilizator obișnuit. Prima pagină este cea de Register, de unde se trece în pagina de Login, unde se va realiza logarea utilizatorului folosind un cont existent. Se va deschide apoi pagina de View and modify Parameters, de unde se poate ajunge la interfața View Data. Ultima pagina este cea de Create event.









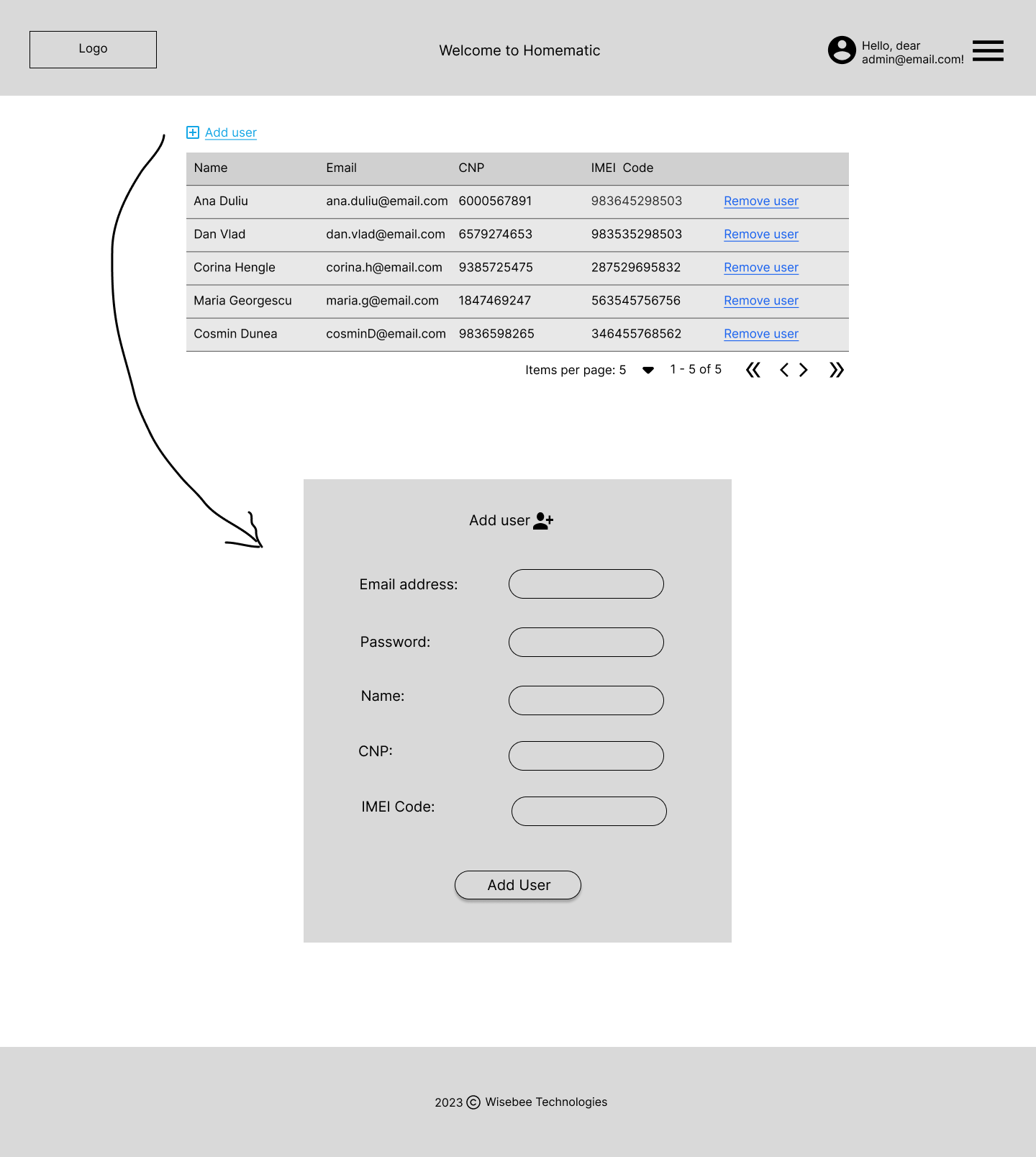


Figura 17. Interacțiunea interfețelor pentru admin

În Figura 17, este prezentată interacțiunea dintre interfețele site-ului Web pentru admin. Se deschide cu pagina de Register, de unde se va ajunge în pagina de Login. Următoarea interfață este cea de View and Modify Parameters, de aici se va trece în pagina de View Data. Se poate ajunge de aici în pagina de Create event. În plus, față de un utilizator obișnuit, adminul va avea acces la o ultimă interfață, View and Add Users.

Modul Mobile

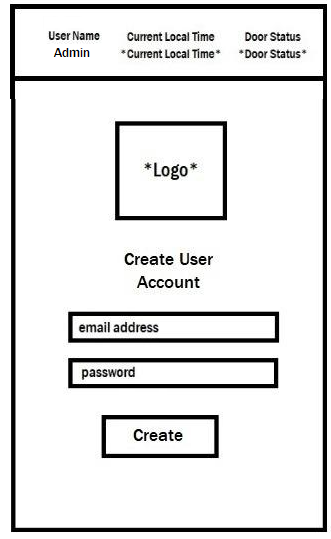
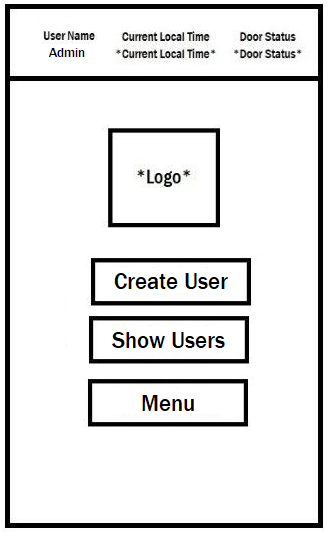
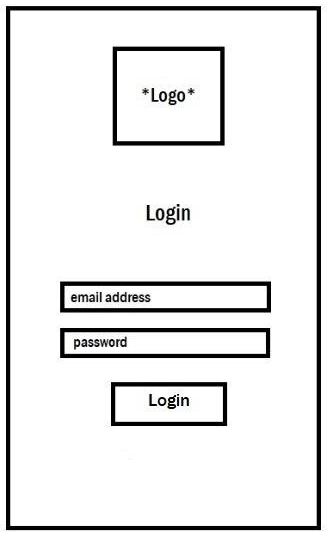
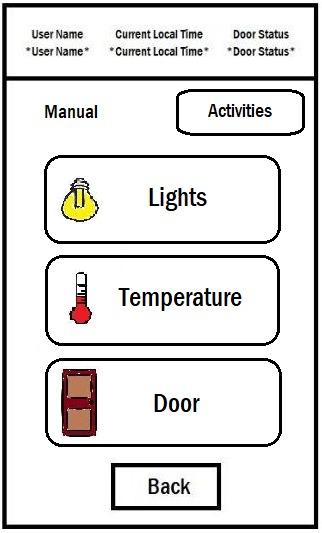


Figura 18. Pagina de Login

Prima interfață este o pagină de Login, unde se realizează logarea utilizatorului cu un cont existent, sau logarea administratorului. În cazul în care cel logat este administrator, următoarea pagină îi permite să creeze un cont nou de utilizator, să vizualizeze toți utilizatorii, sau să acceseze meniul aplicației. În cazul în care este selectată opțiunea “Menu”, administratorul va intra în aplicație, și va avea acces la aceleași funcții precum un utilizator obișnuit.



De la această interfață (Figura 19) este posibilă navigarea printre funcțiile aplicației. Fiecare categorie are o pagină separată, unde este posibilă interacționarea cu sistemul respectiv, și unde este afișat statusul actual al sistemului. De asemenea, se poate urmări și activitatea actuală, dacă este setată una, sau dacă nu, este afișat modul “Manual”.

Figura 19. Navigare funcții aplicație

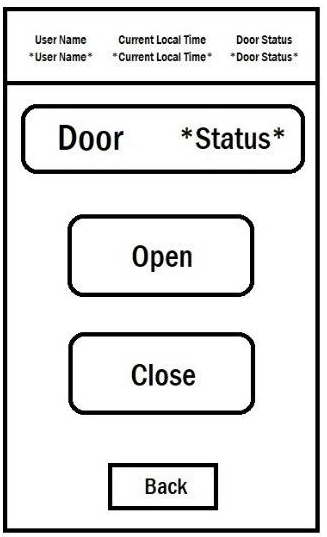
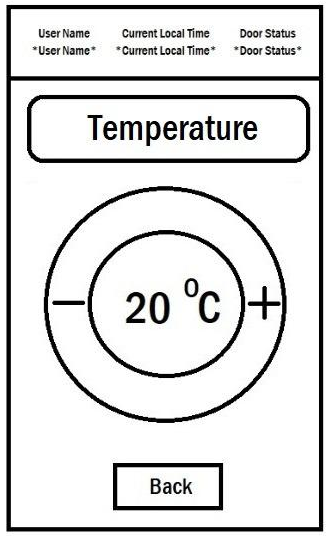
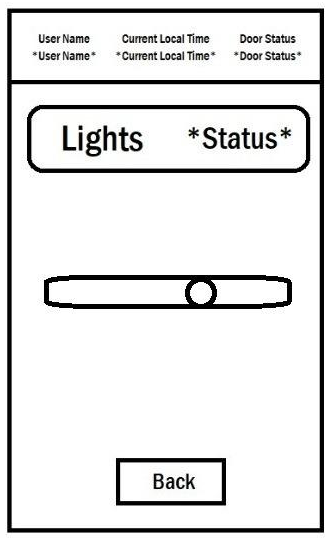


Figura 20. Interfețe pentru modificare parametri

Următoarele 3 interfețe prezentate în figura 20 sunt folosite pentru a aprinde sau stinge lumina, pentru a modifica temperatura sau pentru a închide sau deschide ușa principală.

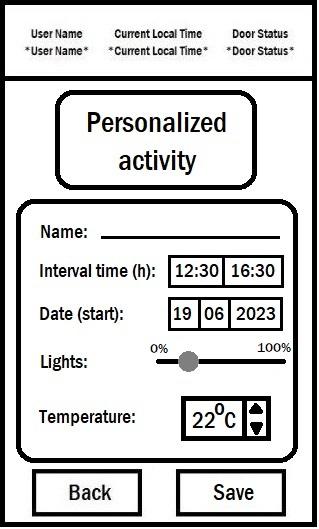
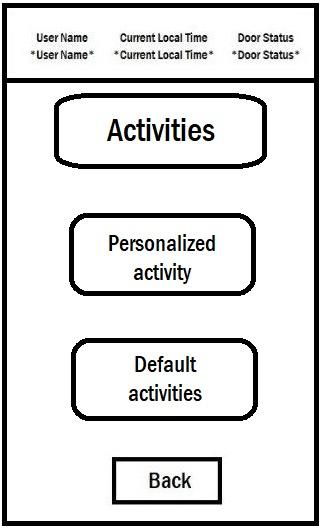
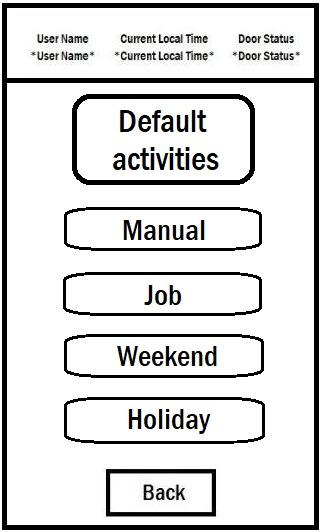


Figura 21. Vizualizare și creare activități

Fiecare activitate poate fi activată, din meniul de activități, sau există posibilitatea de a crea activități noi, personalizate, în funcție de nevoile utilizatorului.

# 6. Descrierea modulelor din punct de vedere al funcționalității

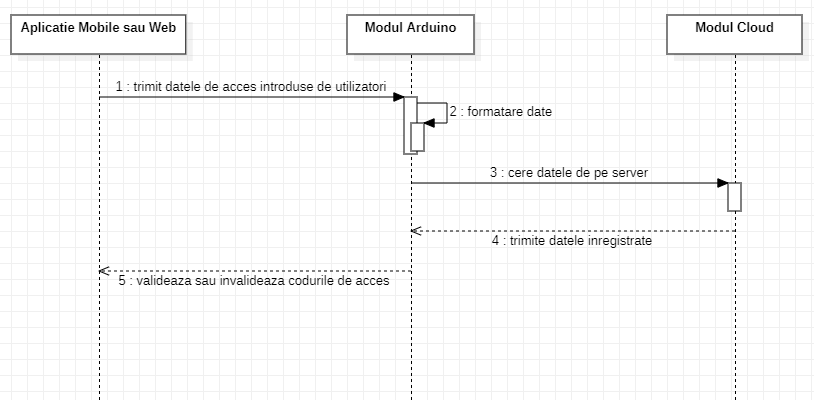


Figura 22. Diagramă de Secvență Proces de Autentificare

În figura 22 este prezentat procesul de autentificare al unui utilizator. Acesta își introduce datele în interfața mobile/web, date care vor fi trimise către modulul Arduino. După primirea datelor, modulul Arduino le formatează și cere datele deja înregistrate pe server pentru a le compara cu cele primite. În cazul în care datele introduse de utilizator se regăsesc printre conturile deja create, modulul Arduino validează codul de acces, iar în caz contrar, acesta le invalidează. În ambele cazuri, răspunsul este trimis înapoi la modului mobile/web.

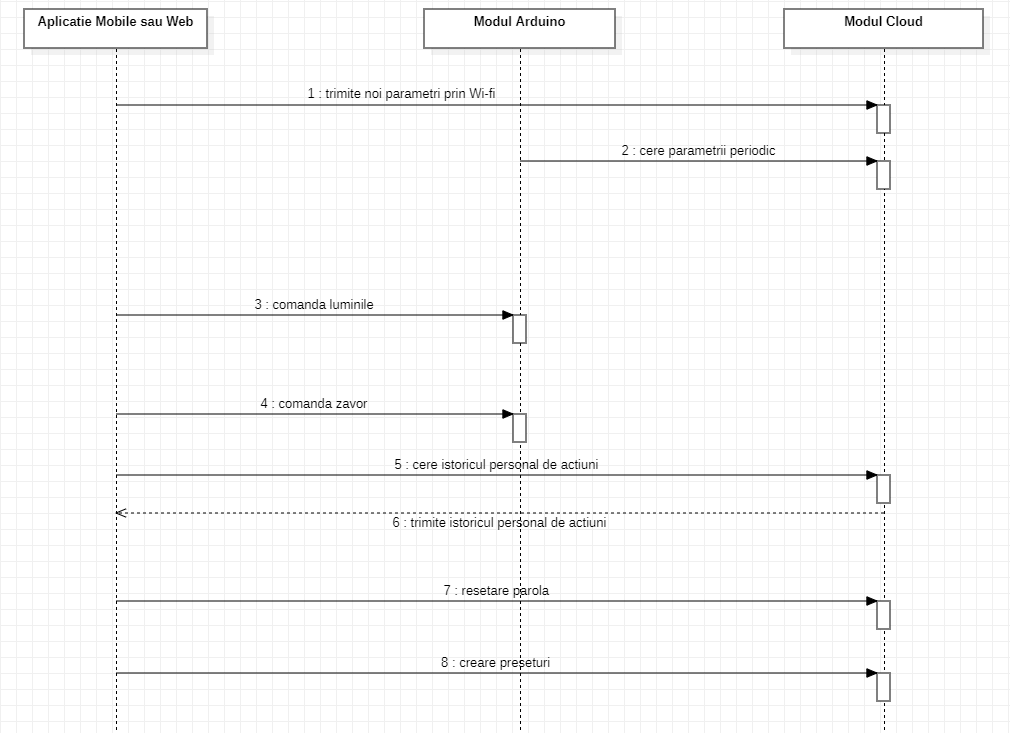
Figura 23. Diagramă de Secvență funcționalități pentru utilizatori

Figura 23 prezintă modurile în care utilizatorul poate modifica sistemul (luminile, încălzirea și starea zăvorului). Utilizatorul introduce noi parametri, care vor fi trimiși către tabela ”parameters” din *Cloud*. La o perioadă predefinită de timp, Modulul *Arduino* verifică dacă datele din aceeași tabelă au fost modificate, urmând să facă schimbările în sistem. Utilizatorul mai poate și să ceară resetarea parolei, acțiune care îi va trimite un mail pentru schimbarea acesteia. Parola va fi modificată în tabela cu utilizatori. De asemenea, utilizatorul își poate creea propriile preseturi care vor fi adăugate în Cloud, în tabela cu evenimente predefinite. Pe lângă acestea, utilizatorul poate cere și vizualizarea istoricului personal, stocat în tabela Acțiuni din Cloud.

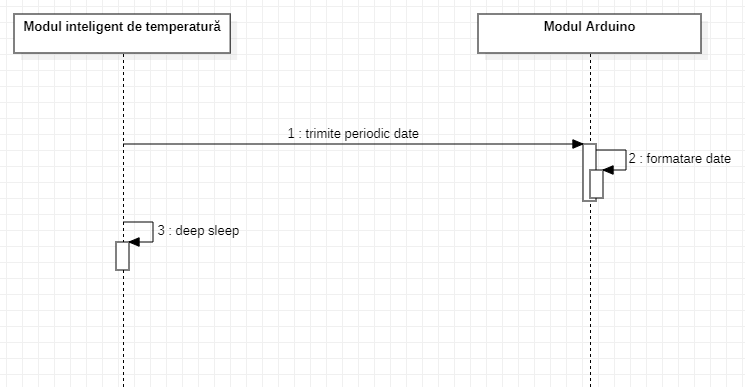


Figura 24. Diagramă de Secvență Modul Wi-fi

În figura 24 este prezentată funcționalitatea modulului *inteligent de temperatură Wi-fi*. Acesta trimite periodic temperatura înregistrată la senzorii săi către modulul Arduino, care formatează datele primite. După ce trimite temperatura înregistrată, acesta intră în modul *Deep Sleep* pentru o perioadă predefinită.

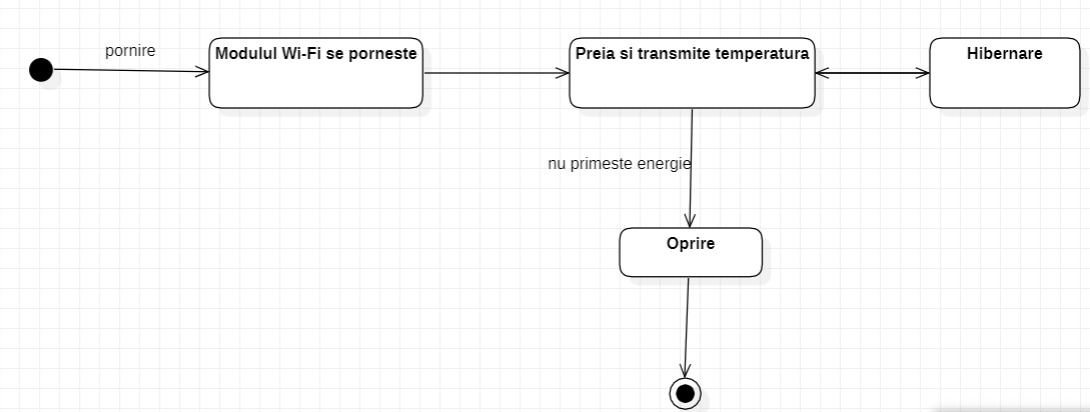


Figura 25. Model de Stare Modul Wi-fi

Modulul *Wi-fi* prezintă mai multe stări reprezentate în figura 25. Modulul intră în prima stare după pornire, stare în care acesta își *inițializează datele*. După inițializare, acesta intră în următoarea stare, unde face *citirea temperaturii de la senzor și o trimite către modulul Arduino* prin Wi-fi. După trimiterea datelor, aceasta intră în cea de-a treia stare, *Deep Sleep* , din care v-a intra înapoi în starea a doua după o perioadă predefinită de timp (perioada de sleep). Există și o a patra stare, cea de *Oprire,* în care modulul intră doar când rămâne fără sursa de alimentare.



Figura 26. Model de stare Modul Cloud

În figura 26 este prezentat modelul de stare al modulului *Cloud*. Starea inițială a modelului verifică ora de schimbare a referinței, aceasta fiind la interval predefinit de timp. Dacă nu s-a ajuns la ora respectivă, se reface verificarea. În caz contrar, se ajunge la următoarea stare, în care ora curentă se compară cu ora programului activ din tabela ”presets”. Dacă ora se potrivește cu cea a programului activ, se intră în următoarea stare, în care parametrii din tabela ”parameters” se modifică în cei ai programului activ și se revine la starea inițială.

# 7. Explicarea funcționării schemelor logice

Codul Device ID

1. Funcția de preluare Device ID, presupune extragerea codului dispozitivului de unde se încearcă accesarea aplicației de către utilizator.

2. Verificarea codului Device ID se bazează pe un algoritm de verificare a lungimi (16 caractere alfanumerice).

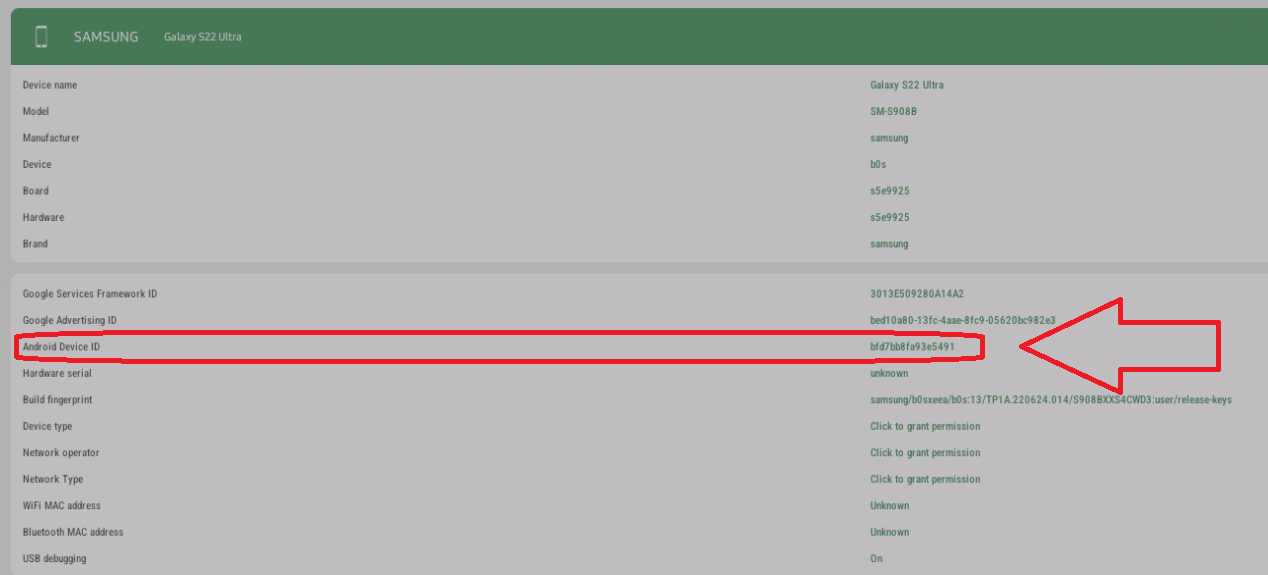


Figura 27. Model Device ID

3.DEVICE\_ID este un număr unic al telefonului, similar codului IMEI. Acesta este generat automat de sistemul de operare la prima pornire a device-ului. Diferenta dintre codul IMEI și Device ID, este aceea ca, in cazul Device ID, se regasesc și caractere nu doar cifre iar lungimea lui este de 16 biți, spre deosebire de 15 biți cum este în cazul IMEI-ului.

4. Logarea este bazată pe interfața de logare unde utilizatorul își introduce datele contului existent, iar dacă acestea sunt valide i se permite accesul la aplicație.

Codificarea opțiunilor unui preset

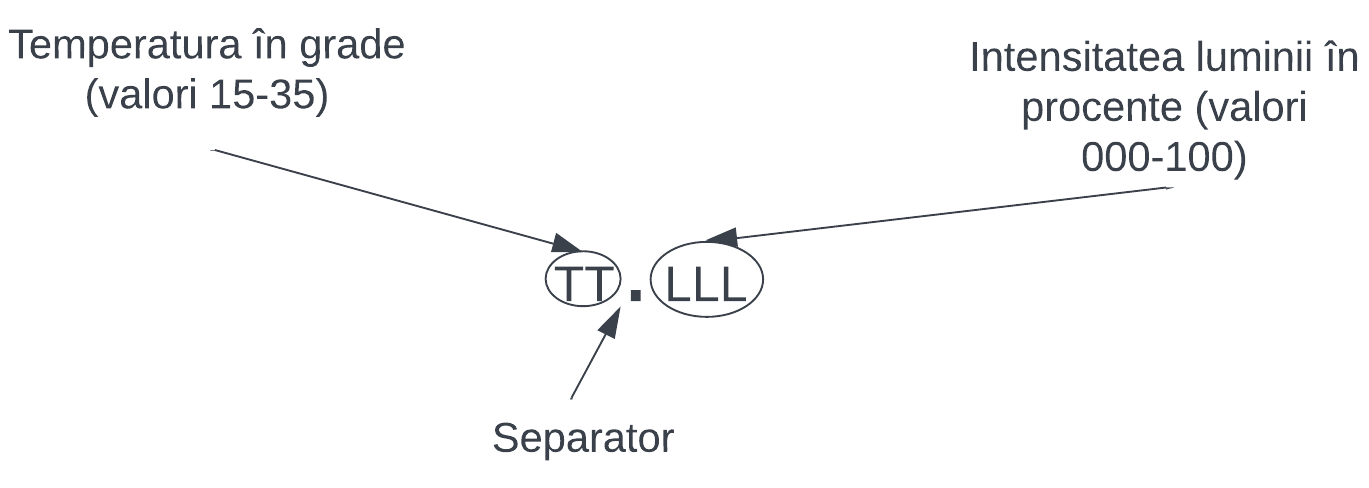
****

Figura 28. Structura mesajului codificat cu opțiuni

Atunci când utilizatorul creează un preset, are posibilitatea de a seta valori diferite pentru temperatură și lumină. Acest lucru va fi codificat printr-un mesaj cu structura prezentată în figura cu numărul 28.

Ex.: Avem codul cu structura:

22.100

Acesta s-ar traduce prin existența opțiunilor: temperatura din locuință va fi setată la 22°C cu luminile aprinse (intensitate de 100%), presetul fiind aplicat din momentul creării acestuia.

Mesajul codificat începe cu un număr între 15-35 corespunzător temperaturii, apoi urmează un separator reprezentat de un punct și valoarea pentru intensitatea luminii. În ceea ce privește construirea mesajului, aceasta este ilustrată printr-o schemă logică la punctul 3 al acestui document.

Deep Sleep

În setup() se face conexiunea la router, implicit la server, și se pornește interfața serială. Se citește voltajul de la senzorul de temperatură al modulului și se convertește în grade Celsius. Temperatura se trimite la serverul Cloud și se afișează pe interfața serială. Se trimite semnalul de Deep Sleep *ESP.deepSleep(Xe6)*, unde argumentul funcției semnifică valoarea in microsecunde, de aici elementul e6 care reprezintă 106, iar X constanta de timp. Programul se va reseta după X secunde. Loop-ul programului nu îndeplinește nici o funcționalitate.

În Deep Sleep, aproape toate funcționalitățile modulului sunt oprite la primirea semnalului *ESP.deepSleep(Xe6)*, ramânand doar RTC-ul (Real-Time Clock) pornit. În modul Deep Sleep, modulul va consuma 20 µA/h.

# 8. Structuri de baze de date și fișiere

În cadrul sistemului nostru, proiectarea tabelelor este necesară deoarece are rol în definirea structurii, tipurilor de date, constrângerilor și a relațiilor dintre tabelele bazei de date. Astfel, datele vor fi stocate și gestionate eficient, lucru foarte important într-o aplicație de Home Automation. De asemenea, proiectarea unor astfel de tabele ne va ajuta în crearea unei baze de date funcționale, care să permită stocarea și accesarea datelor într-un mod organizat, rapid și sigur. Pentru implementarea relațiilor între tabele, o metodă eficientă este utilizarea de chei străine, cu ajutorul cărora se face referință la cheia primară a unei alte tabele. Astfel, în cazul nostru, o cheie străină poate fi folosită pentru a asocia un utilizator cu o acțiune sau cu un eveniment prestabilit.

Sistemul nostru prezintă 5 tabele care relaționează astfel:

Tabela „Users” – Această tabelă conține informații despre utilizatorii care folosesc aplicația noastră. Câmpurile includ un id unic pentru fiecare utilizator, parola, emailul, numele și prenumele utilizatorului, un câmp care va indica dacă utilizatorul are privilegii de administrator și codul numeric personal al acestuia. În privința constrângerilor, avem cheia primară „PRIMARY KEY” cu rolul de a identifica unic fiecare înregistrare din tabel .

Tabela „Actions”- Această tabelă conține informații despre acțiunile programate prin intermediul aplicației, cum ar fi tipul acțiunii, data și ora, respectiv utilizatorul care a efectuat acțiunea. Câmpurile sunt populate cu un id unic pentru fiecare acțiune, un id unic fiecărui dispozitiv, tipul acțiunii(de exemplu, „termostat”, „ușă”, etc.), valoarea pentru care se aplică presetarea, respectiv data și ora. Ca și constrângeri avem o cheie primară care va identifica unic fiecare acțiune prin „action\_id”. De asemenea, avem constrângerea de cheie străină „FOREIGN KEY”, care are rolul de a lega tabela „Actions” de tabela „Users” prin câmpul „device\_id”.

Tabela „Presets”- În această tabelă sunt incluse câmpuri pentru identificarea unică a evenimentului și denumirea acestuia. De asemenea, este inclus și un cod pentru lista de acțiuni care trebuie executate în momentul declanșării evenimentului. O constrângere întâlnită și în tabelele anterioare este cheia primară, care va identifica unic câmpul „preset\_id”. Totodată, cheia străină este prezentă și aici prin „device\_id”, care va face legătură cu tabela „Users” și care va avea și o constrângere de tip NULL pentru preseturile de care dispun toți utilizatorii.

Tabela „Parameters”- Această tabelă va conține înregistrări periodice actualizate cu valorile curente ale temperaturii, intensității luminii și statusul ușii. Constrângerea „PRIMARY KEY” este întâlnită în câmpul „row\_id”, un identificator unic pentru fiecare înregistrare.

Tabela „Temperature\_ESP”- Aceasta tabela va conține înregistrări actualizate periodic cu valori ale temperaturii de la senzorul ESP. Constrângerea “PRIMARY KEY” este întâlnită în câmpul “id” .

Totodată, constrângerea NOT NULL, prezentă în toate tabelele, indica faptul că valoarea NULL nu este permisă în coloana afectată de constrângere.

Având la îndemână aceste date, relațiile dintre tabele vor fi următoarele:

- Tabela "Users" are relație one-to-many cu tabela "Actions", deoarece un utilizator poate efectua mai multe acțiuni, dar o acțiune poate fi efectuată doar de un singur utilizator.

- Tabela "Users" are relație one-to-many cu tabela "Presets", deoarece un utilizator poate crea mai multe presetări, dar o presetare poate fi creată de un singur utilizator.

- Tabela "Actions" are relație many-to-one cu tabela "Users", deoarece mai multe acțiuni pot fi efectuate de același utilizator.

- Tabela "Presets" are relație many-to-one cu tabela "Users", deoarece mai multe presetări pot fi create de același utilizator.

- Tabela "Presets" are relație one-to-many cu tabela "Actions", deoarece o presetare poate implica mai multe acțiuni, dar o acțiune poate fi asociată cu o singură presetare.

-Nu există nici o relație între tabelele “Parameters”, “Temperature\_ESP” și celelalte tabele.

Figura de mai jos explică relațiile dintre tabele:

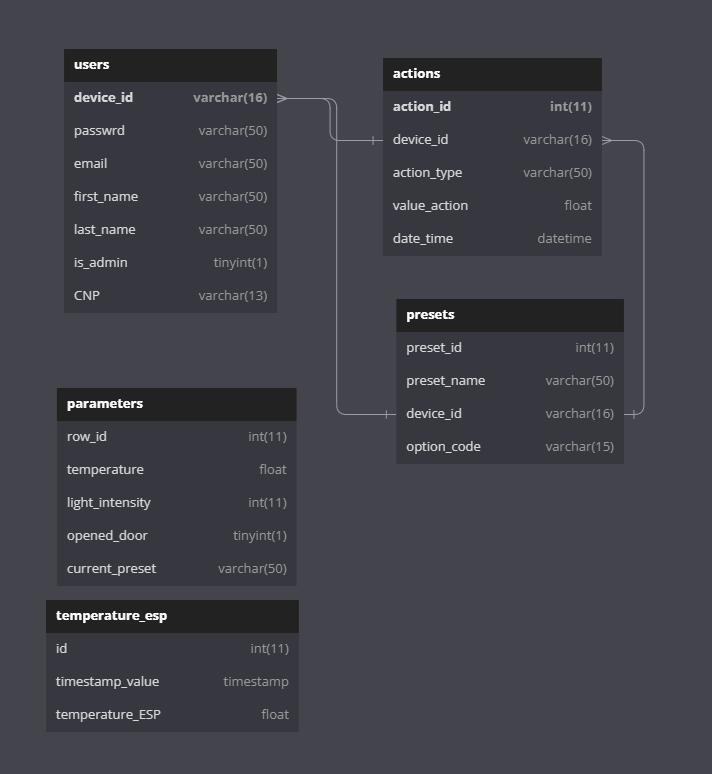


Figura 29. Relațiile dintre tabele

# 9. Anexe

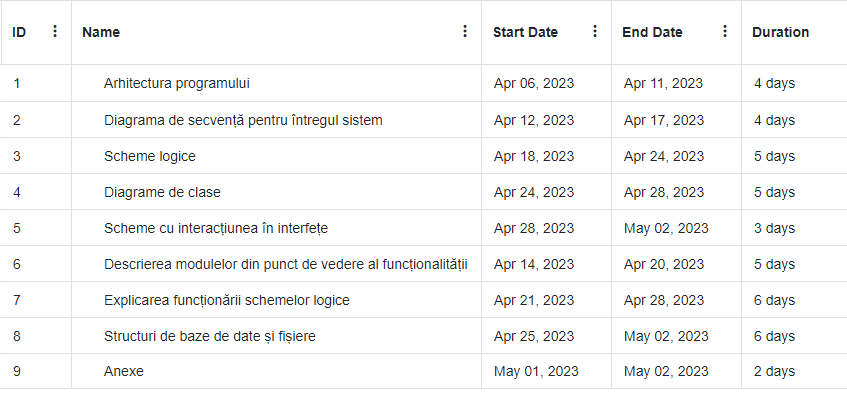


Figura 30. Planificare activități proiectare

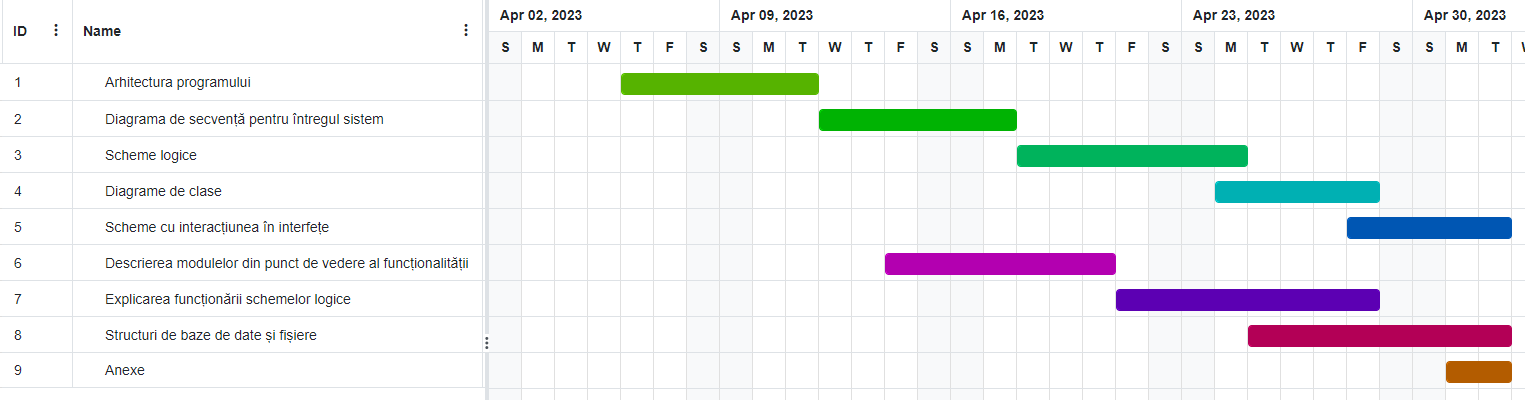
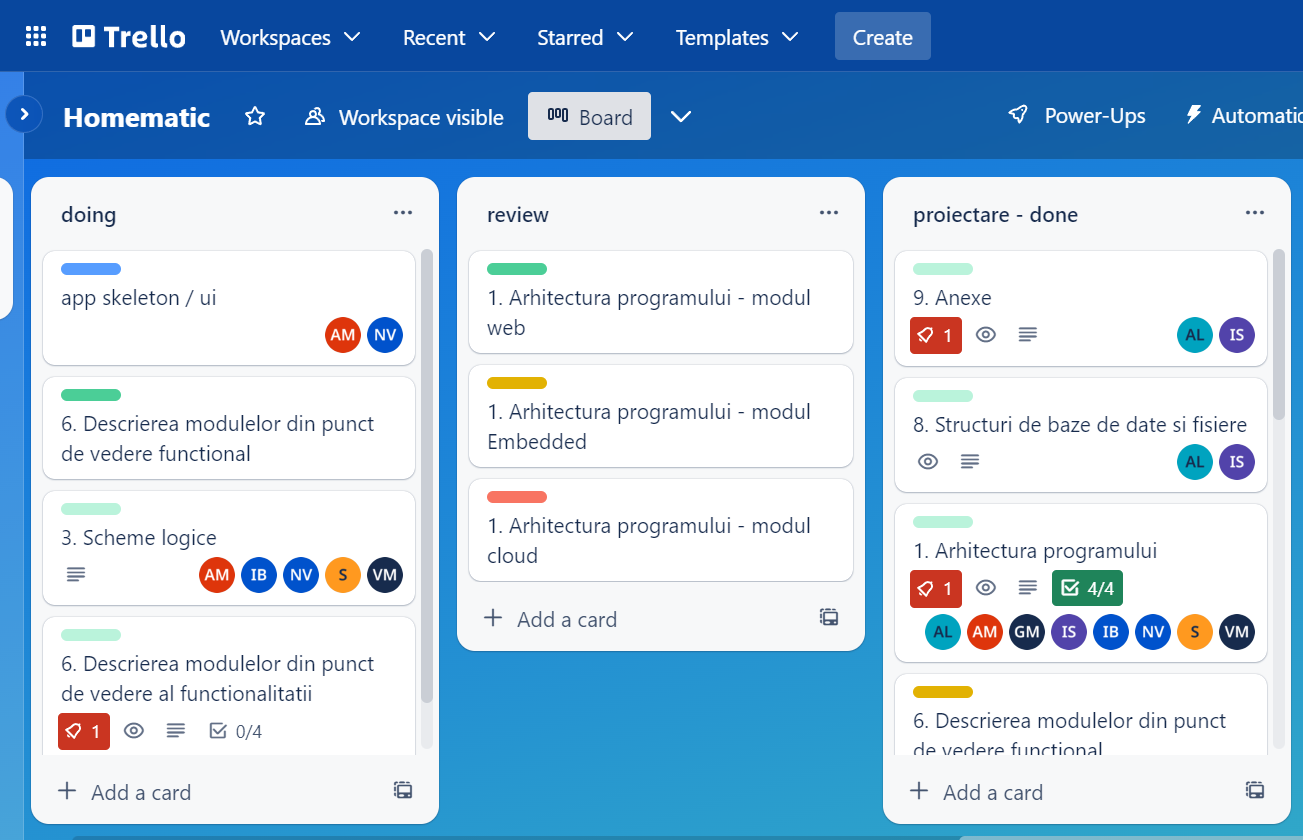


Figura 31. Diagrama Gantt a proiectării



|  |  |
| --- | --- |

Figura 32. Distribuirea atribuțiilor pe platforma Trello