MINISTERUL EDUCAȚIEI

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

APLICAȚIE ANDROID PENTRU MONITORIZAREA CALITĂȚII AERULUI

PROIECT DE DIPLOMĂ

Autor: **Anca – Raluca LOMBREA**

Conducător ştiinţific: **Prof. dr. ing. Silviu Corneliu FOLEA**

2023

MINISTERUL EDUCAȚIEI

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

Vizat,

DECAN
Prof. dr. ing. Liviu MICLEA

DIRECTOR DEPARTAMENT AUTOMATICĂ
Prof. dr. ing. Honoriu VĂLEAN

Autor: **Anca – Raluca LOMBREA**

Aplicaţie Android pentru monitorizarea calităţii aerului

1. **Enunţul temei:** *Lucrarea este concepută pentru obţinerea unei soluţii dedicate monitorizării calităţii aerului, folosind o aplicaţie Android pentru preluarea datelor de la dispozitivele Bluetooth Low Energy şi o aplicaţie web pentru prezentarea rezultatelor sub formă de grafice.*

2. **Conţinutul proiectului:** *Părţile componente ale proiectului sunt: Pagina de prezentare, Declaraţie pe proprie răspundere privind autenticitatea proiectului, Sinteza, Cuprinsul, Introducerea, Studiul Bibliografic, Analiză, Proiectare, Implementare şi Testare, Concluzii, Bibliografie.*

3. **Locul documentării:** *Universitatea Tehnică din Cluj-Napoca, Facultatea de Automatică şi Calculatoare.*

4. **Consultanţi:**

5. **Data emiterii temei:** 06.07.2023

6. **Data predării:** 12.07.2023

Semnătura autorului _____

Semnătura conducătorului ştiinţific _____

MINISTERUL EDUCAȚIEI

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

Declaraţie pe proprie răspundere privind

autenticitatea proiectului de diplomă

Subsemnatul(a) **Anca – Raluca LOMBREA**,
legitimat(ă) cu CI seria AX nr. 702402 , CNP 6000302124936,
autorul lucrării:

Aplicaţie Android pentru monitorizarea calităţii aerului

elaborată în vederea susţinerii examenului de finalizare a studiilor de licenţă la **Facultatea de Automatică şi Calculatoare**, specializarea **Automatică şi Informatică Aplicată,** din cadrul Universităţii Tehnice din Cluj-Napoca, sesiunea Iulie 2023 a anului universitar 2022-2023, declar pe proprie răspundere, că această lucrare este rezultatul propriei activităţi intelectuale, pe baza cercetărilor mele şi pe baza informaţiilor obţinute din surse care au fost citate, în textul lucrării, şi în bibliografie.

Declar, că această lucrare nu conţine porţiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislaţiei române şi a convenţiilor internaţionale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în faţa unei alte comisii de examen de licenţă.

În cazul constatării ulterioare a unor declaraţii false, voi suporta sancţiunile administrative, respectiv, *anularea examenului de licenţă*.

Data

06.07.2023

Anca – Raluca LOMBREA

Lombrea

(semnătura)

MINISTERUL EDUCAȚIEI
UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA
FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE

SINTEZA

proiectului de diplomă cu titlul:

Aplicaţie Android pentru monitorizarea calităţii aerului

Autor: **Anca – Raluca Lombrea**

Conducător ştiinţific: **Prof. dr. ing. Silviu Corneliu FOLEA**

1. Cerinţele temei:

Pentru acest proiect s-a propus crearea unui sistem pentru observarea calităţii aerului, atât în interiorul locuinţelor, cât şi în spaţiile exterioare. Datele necasare vor face referire la temperatură, umiditate relativă, presiune atmosferică şi un echivalent al compuşilor organici volatili şi al concentraţiei de dioxid de carbon. După furnizarea acestor informaţii de către senzori, urmează salvarea lor în spaţiul on-line, de unde vor putea fi preluate şi analizate.

2. Soluţii alese:

Pentru preluarea datelor din mediu s-a ales senzorul BME688, care furnizează infomaţiile necesare şi stabileşte pe baza lor un indice de calitate al aerului. S-a folosit mediul de dezvoltare integrat Android Studio pentru realizarea unei aplicaţii Android capabilă să preia datele transmise prin Blouetooth Low Energy de către beacon. Pentru salvarea caracteristicilor preluate de la senzor s-a utilizat o bază de date NoSQL, oferită de Firebase. Acest lucru a facilitat stocarea şi sincronizarea datelor în timp real, urmând să poată fi accesate de către o aplicţie web, unde se pot vedea rapoartele calităţii aerului în funcţie de dată, locaţie şi de senzorul care le-a furnizat.

3. Rezultate obţinute:

Rezultatele priectului constau în implementarea unei aplcaţii Android ce permite capturarea şi salvarea valorilor referitoare la calitatea aerului pe perioadă de timp într-o bază de date cloud. De asemenea, oferă posibilitatea vizualizării rezultatelor sub forma unor grafice sugestive accesibile prin intermediul unei aplicaţii web.

4. Testări şi verificări:

Testarea aplicaţiei Android s-a realizat folosind un dispozitiv fizic cu versiunea Android 12, iar în cazul aplicaţiei web s-au implementat teste utilizând un browser local.

5. Contribuţii personale:

Contribuţiile oferite în cadrul acestui proiect constau în realizarea componentelor software ale sistemului.

6. Surse de documentare:

Sursele de informaţii au constat în articole ştiinţifice, capitole din cărţi sau rapoarte tehnice şi sunt prezentate în cadrul capitolului destinat bibliografiei.

Semnătura autorului _____

Semnătura conducătorului ştiinţific _____

Cuprins

1 Introducere

1.1 Context general

În prezent, poluarea reprezintă una dintre cele mai mari probleme cu care se confruntă omenirea, aceasta fiind cauza a peste 4 milioane de decese înregistrate anual la nivel mondial. Chiar dacă în Uniunea Europeană a avut loc o scădere considerabilă a emisiiolor pentru poluanţi atmosferici în ultimele decenii, indicele de poluare rămâne încă prea ridicat. Potrivit EEA (European Environment Agency), 96% dintre locuitorii din oraşe sunt expuşi la o concentraţie prea mare de particule nocive. [1]

Poluanţii prezenţi în aer sunt un rezultat al activităţii omului sau sunt rezultaţi din surse naturale, pe care nu le putem controla sau ameliora. Erupţiile vulcanice constituie unul dintre procesele în urma cărora rezultă o cantitate mare de particule de dioxid de sulf, un poluant semnificativ care afectează calitatea aerului. Suprafeţele mari de teren fără vegetaţie produc praf, iar fumul şi monoxidul de corbon din incendii constituie şi ele un factor natural care afectează aerul.[26] Omul contribuie la agravarea poluării prin arderea diferiţilor combustibili, defrişări sau prin eliberarea unor compuşi chimici în natură.

Oricare ar fi cauzele care produc poluarea aerului, sănătatea omului are de suferit, efectele fiind vizibile atât pe termen scurt, cât şi după trecerea timpului. Conştientizarea gravităţii acestei probleme este foarte importantă, doar aşa populaţia poate lua măsuri pentru minimizarea treptată a acţiunilor proceselor tehnologice care cotizează la evoluţia poluării aerului.

Oraşele suprapopulate sunt cele mai vulnerabile din punct de vedere al gradului de impurificare al aerului. Cele mai cunoscute surse care cauzează poluarea în oraşe sunt traficul autovehiculelor, industria, comerţul şi combustibilul ars la nivel casnic. Gradul de poluare este proporţional cu creşterea populaţiei, ceea ce alarmează locuitorii şi îi face mai atenţi la activităţile pe care le desfăşoară.

Căile respiratorii iau contact direct cu poluanţii din aer, determinând scăderea bunei funcţionări a plămânilor. Pentru combaterea apariţiei bolilor care afectează sistemul respirator este nevoie în primul rând de prevenirea poluării aerului pe care îl respirăm. Îmbunătăţirea planurilor urbanistice şi a mediului tehnologic (proiectarea unor noi maşinării care produc mai puţini poluanţi) şi introducerea unor noi legi care să restricţioneze arderea combustibililor sunt câteva forme care reduc poluarea aerului. Se estimează că prin reducerea nivelului arderii incomplete a combustibilor fosili, dar şi a nivelului de ozon din stratul inferior al atmoserei, se pot preveni peste 3 milioane de decese premature şi se poate creşte randamentul culturilor cu aproximativ 50 de milioane de tone anual. [2]

Concentraţia poluanţilor din aer este strâns legată de temperatură şi de umiditatea relativă, având concentraţii diferite in timpul iernii faţă de vară, sau atunci când este secetă faţă de zilele în care plouă. De asemenea, în spaţiile interioare, la un nivel constant de poluare, calitatea aerului scade odată cu creşterea temperaturii şi a umidităţii aerului.

Este posibil ca temperatura şi umiditatea relativă să influenţeze în două feluri calitatea aerului: prin amploarea şi compoziţia emisiilor şi prin percepţia intensităţii mirosului. [3]

Emisiile de CO2 sunt cauza principală a încălzirii globale, lucru ce determină organizaţiile internaţionale să caute noi măsuri pentru minimizarea emanaţiilor. Cu toate acestea, nivelul emisiilor a crescut semnifictaiv, de la aproximativ 23 de gigatone în anul 2000, la 33 gigatone în 2019. Acest proces accelerează degradarea mediului global şi este un pericol pentru economia mondială modernă. [4] Dioxidul de carbon este produs de metabolismul uman şi este constant emis în spaţiile interioare, iar o altă sursă o poate fi reprezentată de procesele de ardere. Principalele surse de CO2 din încăperi sunt oamenii, încălzitoarele cu kerosen sau gaz, fumul de ţigară şi aerul provenit din exterior. În afara clădirilor concentraţia de dioxid de carbon variază de la 320 la 400 părţi pe million, iar în spaţiile interioare neventilate poate să ajungă până la 5000 părţi pe million. [5] Expunerea la un nivel crescut din acest gaz poate să determine dureri de cap, diaforeză, hiperventilaţie sau chiar pierderea cunişţinţei.

Compuşii organici volatili biogenici (BVOC) fac parte dintr-o combinaţie complexă de compuşi emanaţi de plante în atmosferă. Datorită reactivităţii lor, pot contribui puternic la schimbarea globală a climei şi pot avea un rol important în compoziţia atmosferică. Emisiile de BVOC contribuie la formarea ozonului şi a altor poluanţi fotochimici, ceea ce reprezintă un adevărat pericol dacă nivelul de O3 va fi stimulat pe termen lung. [6]

Atunci când vine vorba de măsurarea calităţii aerului din interiorul imobilelor sau din apropierea acestora, IAQ (Indoor Air Quality) ne arată pe o scară de la 0 la 500 starea aerului pe care îl respirăm. Calitatea aerului din interiorul clădirilor poate fi influenţată de materialele din care au fost realizate construcţiile, de activităţile desfăşurate în interior, de arderea combustibililor pentu încălzire sau de infiltraţiile din sol, aer şi apă. O calitate a aerului scazută determină apariţia unor probleme de sănătate precum: dureri de cap, greaţă, congestive nazală, probleme ale ochilor, probleme ale gâtului, oboseală sau dureri ale muşchilor. [7] Aceste simptopme sunt mai des întâlnite în clădirile de birouri unde sunt mulţi angajaţi, iar sistemele de aerisire nu sunt create regulamentar.

În această lucrare se va prezenta un sistem care permite observarea temperaturii, presiunii atmosferice, umidităţii relative, echivalentului cantităţii de dioxid de carbon şi a celui pentru cantitatea compuţilor organici volatili biogenici şi a indicelui de poluare a aerului prin intermediul unui dispozitiv Android. Prin deţinerea unei astfel de aplicaţii, utilizatorul poate să verifice starea aerului din jurul său, astfel poate evita efectele negative determinate de petrecerea unui timp îndelungat într-un mediu ostil sănătăţii. Pentru a putea vizualiza datele preluate din mediu, acestea se stochează într-o bază de date online. Vor fi disponibile atât locaţiile unde s-a realizat preluarea informaţiilor, data şi ora achiziţiei, dar şi grafice sugestive din care se poate vedea evoluţia parametrilor monitorizaţi pe o durată fixă de timp.

Pentru achiziţia datelor s-a folosit senzorul BME688, iar beaconul în care este integrat permite transmiterea informaţiilor prin Bluetooth Low Energy, cu un consum scăzut al bateriei şi o gamă largă de operare (de la -40 la 85 °C). Pentru preluarea şi stocarea datelor, dispozitivele Android reprezintă o soluţie optimă datorită gradului

redus de complexitate în raport cu utilizatorul, acestea fiind una dintre cele mai răspândite mecanisme în rândul populației. Salvarea într-o bază de date online a avut ca scop permiterea accesului la baza de date de pe diferite dispozitive, existând posibilitatea observării de la distanță a parametrilor livrați de beacon.

Această lucrare conține patru capitole mari: introducere, studiu bibliografic, analiză și implementare și concluzii. În prima parte are loc descrierea contextului în care s-a dorit realizarea aplicației, motivele alegerii componentelor și specificarea pașilor necesari îndeplinirii obiectivelor. Studiul bibliografic are ca și conținut descrierea lucrărilor anterioare care au inspirat autorul și au ajutat la crearea acestei lucrări, iar în partea de analiză și implementare se vor descrie detaliat pașii urmați pentru finalizarea și testarea aplicației. În plus, în secțiunea dedicată concluziilor se vor expune rezultatele și posibilele dezvoltări ale proiectului.

1.2 Obiective

În realizarea proiectului actual am avut ca principal obiectiv construirea unui sistem pentru observarea parametrilor necesari determinării calității aerului, atât în spațiile interioare, cât și în afara locuițelor. Datele vor fi preluate pe durate predefinite de timp și salvate într-o bază de date online, care permite prelucrarea și afișarea lor de către un program software, urmând să poată fi accesate dintr-un browser web. Se va cere permisiunea utilizatorului pentru folosirea tehnologiei Bluetooth, făcând posibilă preluarea datelor de la beacon și pentru activarea locației pe dispozitiv, urmând salvarea acesteia în baza de date.

Obectivele detaliate sunt expuse în cele ce urmează:

- Determinarea unei soluții în mediul de programare Android Studio pentru scanarea prin Bluetooth Low Energy a dispozitivelor în funție de adresa MAC (Media Access Control address) dorită;
- Preluarea informațiilor de la senzor în funcție de intervalul de timp ales de client pentru menținerea scanării;
- Decodificarea pachetului de date difuzat de către beacon pentru obținerea celor șase parametrii doriți (temperatură, preiune atmosferică, umiditate relativă, echivalentul pentru dioxidul de carbon, echivalentul pentru cantitatea de compuși organici volatili biogenici și indice de calitate al aerului);
- Preluarea locației curente a utilizatorului și transmiterea în timp real a datelor decodificate într-o bază de date online;
- Vizualizarea informațiilor extrase în timpul scanării, fiind posibilă compararea lor cu valorile optime pentru fiecare mărime în parte;
- Crearea unui program software pentru extragerea datelor din baza de date și afișarea lor într-un mod ușor de înțeles și de evaluat pentru utilizator.

Se dorește obținerea acestor obiective cu un cost redus pentru toți deținătorii unui dispozitiv Android care să fie compatibil cu versiunea 21 API (Application Programming

Interface) sau una ulterioară. Costul pentru achiziționarea senzorului BME688 sunt scăzute, astfel este posibilă deținerea unui astfel de dispozitiv de către o gamă largă de persoane.

1.3 Specificații

Din punct de vedere al intercațiunii utilizatorului cu mediul Android, se dorește obținerea unei interfețe ușor de utilizat, unde să fie permisă inițierea scanării prin Bluetooth Low Energy a dispozitivelor compatibile. Se va permite alegerea intervalului de scanare, interval în care datele decodificate vor fi afișate pe ecran alături de locația curentă. Pentru a vedea în timp real cât de ridicată sau scăzută este calitatea aerului, se vor compara valorile obținute cu cele optime.

Pentru salvarea datelor se cere folosirea unei baze de date online care stochează informațiile în funcție de data și ora la care a fost inițiată scanarea. În acest proiect s-a folosit o bază de date NoSQL oferită de Firebase pentru primirea și reținerea pachetelor de date. S-a ales ca securitatea să fie scăzută, oricine care are adresa de la baza de date putând face modificări. Acest lucru facilitează accesarea din mai multe programe a infromațiilor.

Pentru obținerea datelor într-un browser web, se vor studia limbajele de programare JavaScript (JS), HTML (Hyper Text Markup Language) și CSS (Cascading Style Sheets) și se va crea o aplicație cu interfață ușor de urmărit. Mărimile de la senzori vor fi afișate grafic în funcție de data preluării și de locația din care au fost extrase. De asemenea, la fiecare interceptare a unui nou pachet de la senzor, mesajele decriptate vor fi afișate pe ecranul dispozitivului Android alături de o atenționare, în cazul înregistrării unor valori periculoase pentru starea de sănătate a persoanelor aflate în proximitatea locației unde a avut loc preluarea informațiiolor.

Se va utiliza o metodă precisă de aflare a locației dispozitivului Android, datorită acurateței cerute în cadrul programului implementat în Android Studio, folosind limbajul de programare Java. Valorile transmise de senzor sunt reprezentate în numere întregi, fiind suficientă această precizie pentru aflarea calității aerului.

Limitările programului sunt reprezentate de necesitatea cunoașterii adresei MAC a beaconului, dar și de faptul că este nevoie de informații privind decodificarea pachetelor primite în urma scanării. Dacă nu se va ține cont de aceste aspecte, calculele folosite pentru aflarea valorilor mărimilor cerute vor fi eronate, iar valorile furnizate de aplicație vor fi irelevante.

2 Studiu bibliografic

Observarea calităţi aerului reprezintă un punct de interes în prezent în favoarea căruia au fost studiate şi implementate diferite metode cât mai eficiente din punct de vedere al energiei consumate, dar şi uşor de procurat privind preţul de achiziţionare al dispozitivelor necesare. Pentru realizarea acestui obiectiv, s-au folosit dispozitive hardware pentru procurarea informaţiilor relevante din mediu (temperatură, compuşi organici volatili, presiune atmosferică, concentraţie de dixid de carbon). Transmiterea datelor, folosind protocoale speciale de comunicare, permite vizualizarea şi stocarea informaţiilor în baze de date specializate. Astfel, clientul este uşor informat asupa schimărilor produse in mediu şi asupra calităţii aerului din spaţiile interioare, dar şi din afara imobilelor.

Este foarte important să cunoaştem beneficiile şi provocările pe care ni le oferă noile tehnologii dezvoltate pentru crearea unor locuinţe şi totodată oraşe inteligente, acestea incluzând şi monitorizarea calităţii aerului. În lucrarea [18] s-a discutat despre caracteristicile dispozitivelor care folosesc Bluetooth Low Energy pentru preluarea informaţiilor de la senzori, despre aplicaţiile acestor sisteme şi posibilităţile de dezvoltare. Una dintre cele mai utilizate tehnologii de comunicaţie este Wi-Fi (Wireless-Fidelity), care reprezintă o modificare a standardului WLAN-IEEE802.11ax. Această schimbare a fost necesară datorită creşterii cerinţei lăţimii de bandă pentru livrarea video, care depăşesc capacitatea vechilui standard [19]. Alte alternative ale soluţiei Wi-Fi sunt: ZigBee (protocol bazat pe standardul IEEE 802.15.4 care susţine dispozitivele de monitorizare şi putere redusă[20]), LoRaWAN, RFID (Radio Frequency Identification) şi comunicarea prin Bluetooth, aceasta fiind o comunicaţie wireless pentru distanţă scurtă, alături de o altă parte a sa numită Bluetooth Low Energy.

Beacon-urile BLE sunt dispozitive care servesc unei game largi de aplicaţii ce ne îmbunătăţesc calitatea vieţii, fiind utilizaţi în domeniul sănătăţii, în dezvoltarea unor sisteme inteligente pentru locuiţe sau pentru localizarea precisă a oamenilor sau echipamentelor. Componentele hardware ale unui beacon sunt: sursa de energie, microcontrolerul şi un chip radio BLE. Proiectarea unui astfel de device trebuie să ţină cont de costul de producţie, dimensiunile finale, consumul de energie sau memorie integrată în chip. Astfel, mărimea bateriei trebuie calculată cu atenţie, iar aceasta poate varia de la valori de peste 200 mAh până la 2000mAh, care determină o perioadă de viaţă mai lungă, dar şi dimensiuni considerabil mai mari. Pe lângă acet tip de alimentare se mai poate lua în considerare şi folosirea unui cablu USB sau folosirea celulelor solare pentru încărcarea de la lumina solară.

Pentru a controla funcţionalitatea componentelor hardware este necesară programarea firmware-ului astfel încât să îndeplinească cerinţele utilizatorului. Se poate programa puterea de transmisie sau intervalul în care să se realizeze publicarea pachetelor cu informaţii, ambele criterii influenţează nivelul în care se consumă energia în sistem. Cu cât se doreşte preluarea datelor de la o distanţă mai mare, sau cu cât se foloseşte un interval scurt de timp între timpii de publicare a pachetelor, se va creşte şi consumul de putere.

În contextul oraşelor inteligente beacon-urile ocupă un loc foarte important, aceste dispozitive putând să se împartă, în funcţie de amplasarea lor, în dispozitive cu locaţie fixă, cum ar fi în restaurante, şcoli, staţii de autobuz, şi în beacon-uri cu locaţie mobilă, cum ar fi pe biciclete, agăţate de ghiozdane sau în bagaje în aeroport. În această tehnologie sunt prezente şi dezavantaje cu privire la securitate cum ar fi: dezasamblarea dispozitivelor aflate în spaţii publice şi, astfel, accesarea componentelor hardware ale acestora, clonarea sau folosirea UUID-urilor în alte aplicaţii fără permisiunea producătorului.

Calitatea aerului în spaţiile în care elevii îşi desfăşoară activitatea reprezintă un factor important pentru menţinerea sănătăţii acestora, dar şi pentru creşterea gradului de productivitate şi atenţie al copiilor. În lucrarea [8] s-a descris proiectarea unui sistem pentru evaluarea calităţii aerului din interiorul unităţilor de învăţământ folosind o soluţie IoT (Internet of Things) pentru un mediu de învăţare sănătos. Prin acest sistem se colectează datele necesare stabilirii calităţii aerului, se distribuie ocupanţilor datele actuale şi se ţine un istoric cu informaţiile pentru a putea fi studiate.

Folosirea unei reţele distribuite de senzori (DSN = Distributed Sensor Network) pentru realizarea măsurătorilor a fost soluţia aleasă de autorii lucrării pentru a creşte eficienţa, dartorită numărului mic de cabluri necesare pentru instalare. Structura generală a unei astfel de reţele este formată dintr-un set de noduri senzori, un set de elemente de prelucrare şi o reţea de comunicare prin intermediul căreia se stabileşte comunicarea între elementele de prelucrare. Informaţiile sunt transferate de la senzori spre elementele de prelucrare asociate fiecăruia, unde are loc integrarea datelor [9]. În lucrarea citată anterior, senzorii sunt aşezaţi în locaţii şi înălţimi diferite pentru a prelua inforaţii atât din sălile de clasă, cât şi de pe holurile şcolii. Pentru a realiza conectarea wireless cu fiecare nod la intervale de timp predefinite şi pentru a face disponibile măsurătorile pentru server, s-a implementat un nod central gateway. În IoT, un nod gateway este un intermediar care conectează senzorii la internet sau alte reţele. În cazul actual, s-a folosit BLE pentru a colecta datele şi interfaţa Wi-Fi pentru a conecta sistemul la internet. Serverul colectează şi transmite un semnal de alarmă dacă valorile adunate sunt periculoase şi este nevoie de intervenţia omului pentru a restabili aceşti parametri la un nivel normal. Serverul poate să fie o maşină standard Linux care rulează pe un dispozitiv fizic sau pe o maşină virtuală în Cloud.

Pentru stabilirea cât mai exactă a calităţii aerului, este esenţial ca senzorii să fie capabili de transmiterea unor parametrii relevanţi pentru acest subiect. Datele au fost adunate în funcţie de concentraţia de dioxid de carbon, compuşii organici volatili totali (TVOC), temperatură, umiditatea relativă, presiunea aerului, lumina ambientală (dacă lumina este pre scăzută prvoacă somnolenţă, iar dacă este prea ridicată poate cauza stres), indexul ultraviolet (poate indica un echipament care nu funcţionează în condiţii de siguranţă) şi nivelul de zgomot. Pentru a colecta toate aceste elemente s-a folosit un kit de dezvoltare bazat pe un sistem on-chip wireless care suportă protocolul de comunicare Bluetooth Low Energy.

Datorită faptului că senzorii vor fi amplasaţi în săli diferite şi la înălţimi variabile, ar fi necesare prize de la care să se poată încărca nodurile senzor. Însă în cazul în care nu

este în apropierea o astfel de priză, se poate opta şi pentru alimetarea cu baterii. Autorii au decis folosirea bateriilor NiMH de tip AA datorită costurilor reduse comparativ cu bateriile cu litiu. Bateriile bazate pe hidrura de niche-metal (NiMH) reprezintă o tehnologie de baterii reîncărcabile care sunt importante din punct de vedere comercial pentru industrie şi pentru micii consumatori [10]. S-a estimat perioada de viaţă fără încărcare de până la 13 zile, fiind uşor de extras şi reîncărcat datorită suportului pentru baterie din carcasa amplasată pe senzor.

Software-ul care rulează pe fiecare placă de dezvoltare iniţiază un server Bluetooth Generic Attribute Profile (GATT) prin care se publică datele, după ce are loc preluarea lor din mediu. Pachetele publicate sunt formate din atribute care conţin caracteristicile unui profil, reprezentând rezultatele măsurătorilor şi alte informaţii suplimentare de care nu este nevoie în scopul prezentului proiect. Protocolul BLE este format din trei straturi principale: controller, gazdă şi aplicaţie. Controllerul cuprinde stratul fizic şi cel de legătură, iar gazda este formată din funcţionalităţile stratului superior, inclusiv Generic Attribute Protocol (GATT). Informaţiile care trebuie transmise de la un nod BLE şi recepţionate la alt nod sunt plasate într-o bază de date numită server GATT. Aceste măsurători pot fi transmise unei alte baze de date aflate în alt nod numit client. Clientul trimite o cerere serverului, ceea ce provoacă un mesaj de răspuns din partea acestuia. Pentru a transfera datele de la serverul GATT spre client se poate face citirea explicită a carecteristicilor transmise de la senzor, sau se poate realiza notificarea clientului atunci când o valoare a caracteristicilor transmise este schimbată [11]. În mod normal, placa de dezvoltare folosită în acest sistem este capabilă să transmiă datele timp de 30 de secunde, urmând să treacă într-un mod care consumă mai puţină energie numit sleep mode. Datorită faptului că firmware-ul plăcuţei de dezvoltare este de tip open-source, a fost posibilă modificarea modului său de operare la unul continuu. Firmware-ul este software-ul care controlează funcţionarea dispozitivelor hardware, iar faptul că acesta este open-source ne arată că utilizatorul este capabil să vizualizeze şi să modifice codul, reuşind să satisfacă cerinţele specifice ale clientului.

Pentru a face legătura dintre senzori şi server avem nevoie de un element gateway, care colecteaza datele de la fiecare senzor, le converteşte în formatul dorit şi le trimite mai departe pentru a putea fi procesate. Avem nevoie de adresa MAC a plăcii de dezvoltare pentru a crea un adaptor Bluetooth, astfel se poate crea conectarea. Datele măsurate sunt convertite într-o formă care poate să fie înţeleasă de utilizator, ulterior sunt salvate în fişiere separate, iar adaptorul este închis. Acest proces este luat de la început şi sunt extrase alte pachete de date.

Pentru procesarea datelor s-a utilizat Simple Network Management Protocol (SNMP), acesta fiind un protocol pentru observarea şi administrarea dispozitivelor de reţea. S-a implementat un pachet SNMP pentru nodul gateway, care include informaţiile de la senzor în MIB (Managment Information Bases). MIB este o baza de date care stochează datele despre dispozitivele de reţea, sau alte obiecte administrate de SNMP. Pentru stocarea şi afişarea măsurătorilor sub formă de grafice s-a apelat la CACTI. CACTI este un instrument capabil să ofere o interfaţă web pentru prezentare şi trimitere de notificări atunci când apare un nivel anormal sau periculos pentru sănătate la unul dintre senzori.

Testarea a avut loc pe o mașină virtuală Linux, folosind doi senzori. S-a creat un fișier în care s-au introdus adresele MAC ale senzorilor, caracteristica Bluetooth UUID (folosită într-ul service BLE pentru a defini caracteristici precum formatul datelor transmise sau permisiunile de acces) și intervalul de timp la care se dorește să se realizeze măsurarea. Senzorii au fost amplasți în două clase de elevi, iar după 24 de ore de măsurători s-au putut vedea schimbările factorilor care influețează calitatea aerului reprezentate grafic.

Un scop important în momentul de față, ca urmare a dezvoltării conceptului Internet of Things, îl reprezintă reducerea costurilor de întreținere și încărcare a instalațiilor folosite. Astfel este nevoie de tehnologii care să țină cont de acest aspect și să ofere noi soluții pentru optimizarea proceselor care au nevoie de surse de energie care să reziste o perioadă lungă de timp. În lucrarea [12] s-a prezentat colectarea informațiilor prinvind temperatura, presiunea atmosferică și umiditatea folosind o placă Raspberry Pi la care s-au conectat senzori capabili să ofere datele cerute. Prin construirea unei librării API s-au putut extrage datele de la senzori și, de asemenea, s-au transmis aceşti termeni în conținutul iBeacon.

Apple a dezvoltat protocolul iBeacon, care a fost lansat în anul 2013 în cadrul Worldwide Developer Conference, iar pentru a putea identifica un dispozitiv ca iBeacon este nevoie de licență oferită de această companie. Un iBeacon este un dispozitiv care are integrat un chip BTLE, folosind tehnologia Bluetooth Low Energy pentru a transmite datele specifice. Acest tip de aparat este un emițător pasiv care transmite semnale, acestea putând fi preluate de alte dispozitive. Singurul mod prin care se realizează interacționea cu un iBeacon este configurarea valorilor pentru UUID, valoarea majoră, valoarea minoră și puterea de transmitere [13]. UUID, sau Universally Unique Identifier, este un un șir de caractere unic pentru fiecare dispozitiv, în acest fel se pot identifica și distinge iBeacon-urile unele de altele.

În lucrarea prezentată s-a dorit înlocuirea unui beacon cu o placă Raspberry Pi care are integrat un modul Bluetooth Low Energy. După preluarea datelor de la senzori, se vor transmite informațiile utilizând protocolul SSH (Secure Shell) dispozitivelor care folosesc aceeași rețea de internet. Aceasta este o soluție pentru a lucra cu Raspberry Pi la distanță. Dacă este detectată o creștere periculoasă a unuia dintre cei trei parametrii observați (temperatură, umiditate, presiune atmosferică), se va trimite o alertă utilizatorului folosind mail-ul sau numărul de telefon setat din program. În funcție de mesajul de alertă primit, utilizatorul poate să acceseze Raspberry Pi și utizând API-urile să adune informațiile primite de la senzori. În acest mod se vor examina valorile preluate din mediu și se va crea o idee clară despre pericolele care au fost detectate, utilizând un telefon mobil aflat în vecinătatea plăcuței.

Din punct de vedere al consumului de energie și al costurilor necesare menținerii programului activ, soluția aleasă de autorii acestui proiect este foarte avantajoasă, comparativ cu dispozitivele comerciale disponibile în magazine. S-a testat funcționalitatea într-un spațiu de birouri pe o durată de 24 de ore, fiind conectați 4 senzori la placa Raspberry Pi (doi senzori de temperatură/umiditate, un senzor de sunet și unul pentru presiune). Costurile estimate în decursul a o lună de zile de funcționalitate, în aceleași condiții, s-au estimat ca fiind de aproximativ de trei ori mai scăzute față de o

soluţie comercială disponibilă la scară largă. Astfel se pot vedea beneficiile folosirii unui harware ca Raspberry Pi pentru o perioadă mai lungă de timp.

Poluarea este un factor de risc, mai ales pe teritoriul oraşelor suprapopulate, iar oferirea unor informaţii cu privire la porţiunile cu cel mai mare indice de poluare este benefic pentru locuitori. Autorii lucrării [14] au propus o soluţie pentru crearea unei hărţi în care să fie disponibile date despre nivelul compuşilor dăunători sănătăţii prezenţi în aer. Folosind un smartphone, orice persoană poate să contribuie la îmbogăţirea bazei de date cu măsurători noi, oferind o mai mare acurateţe afişării acestora pe ecran. Toate acestea se vor realiza ţinând cont de cerinţele de consumul de energie scăzut şi de costul de achiziţie a unui astfel de sistem.

Sistemul se bazează pe un senzor MiCS-OZ-47, acesta fiind un modul PCB (Printed Circuit Board) bazat pe microprocesor, gândit pentru aplicaţii de măsurare a concentraţiei de ozon. Semnalul brut primit de la senzor este procesat de către microprocesor şi ulterior, are loc stocarea datelor relevante [15]. Principiul de funcţionare al senzorului se bazează pe generarea unei reacţii chimice în prezenţa ozonului, concentraţia de ozon fiind proporţională cu puterea acestei reacţii. Ozonul reprezintă un poluant rezultat în urma reacţiei dintre oxizii de azot şi compuşii organci volatili prezenţi în aer [16], cantitatea de acest compus fiind proporţională cu gradul de poluare. Pentru realizarea comunicării dintre placă şi smartphone, este necesară o interfaţă RS232-TTL, care este direct conectată cu telefonul prin cablu USB. Datorită faptului că versiunile mai vechi de smartphone nu erau capabile să ofere energie prin USB dispozitivelor la care erau conectate, s-a decis adăugarea unei surse externe de energie pentru a alimenta senzorul. Iar prin folosirea modulului automat al senzorului, acesta îşi foloseşte ceasul intern pentru a performa măsurători o dată la două secunde.

Dispozitivele Android conectate la senzorul de ozon vor utiliza o bibliotecă software, care permite citirea datelor seriale. Aplicaţia disponibilă telefoanelor Android face posibilă interacţiunea simplă cu utilizatorul, acesta reuşind să realizeze măsurători ale ozonului, să calibreze senzorul şi să încarce datele preluate. Calibrarea senzorului este foarte importantă deaorece se evită salvarea în baza de date a unor valori neconforme cu realitatea. Măsurătorile preluate într-un interval de timp mai mic de zece minute sau o distanţă sub 400 de metri trebuie să fie foarte asemănătoare, în caz contrar se presupune o eroare a senzorului, ceea ce duce la necesitatea calibrării acestuia.

Pentru a testa funcţionalitatea acestui sistem, autorii lucrării au realizat măsurători pe o perioadă de două luni. Datele au fost preluate de la mai mulţi senzori montaţi pe biciclete care s-au mişcat pe teritoriul unui oraş. În urma colectării, au rezultat aproape 3000 de puncte distribuite pe suprafaţa oraşului, aceste informaţii fiind încărcate pe un server care face parte dintr-o reţea globală de senzori. Din acest punct datele sunt disponibile oricui, iar în ajutorul interpretării mai concrete se adaugă şi locaţia din care acestea au fost extrase. Pentru a forma harta pentru poluarea aerului, s-au luat în considerera 3 scenarii: aer curat corespunzător culorii verzi, stadiu mediu de poluare marcat cu galben şi roşu pentru porţiunile foarte poluate. După ce utilizatorul alege zona dorită de pe hartă, aceasta se va împărţi în forme pătrate, având latura de 35 pixeli. Pentru fiecare parte se va calcula media ozonului preluat de la toate dispozitivele care au efectuat

măsurători în acea zonă. În acest fel persoanele pot să îşi facă o idee de ansamblu asupra nivelului de poluare dintr-un oraş, dar au posibilitatea să privească şi în detaliu anumite zone de interes. Un astfel de proiect, care permite citirea datelor din mediu cu un dispozitiv conectat direct la telefonul mobil, este avantajos din punct de vedere al energieie conumate, dar şi pentru accesul la o bază de date în care pot să se adune informaţii de la mulţi senzori, ceea ce oferă o mai mare acurateţe a rezultatelor finale.

În cadrul dezvoltării continue a soluţiilor care fac parte din conceptul Internet of Things, autorii lucrării [17] au oferit o prezentare a unui sistem de monitorizare a calităţii aerului în spaţiile interioare, dar şi în exterior cu un consum scazut de energie folosind dispozitive încăcabile cu o singură baterie. S-a propus folosirea unui sistem de dimensiuni reduse, care poate să fie uşor de transportat, dar şi cu cost scăzut, pentru a putea fi achiziţionat de orice persoane. Parametrii monitorizaţi sunt temperatura, umiditatea relativă, presiunea atmosferică şi concentraţia de gaze din aer, pe baza cărora se va genera un indice care arată gradul de poluare din zona din care au fost preluate datele. Preluarea datelor se va realiza prin decodificarea pachetelor transmise de beacon, folosind modul de publicitate al tehnologiei de comunicare Bluetooth Low Energy.

Dispozitivul dezvoltat are patru componente principale: un circuit în care au fost amplasaţi toţi senzorii necesari, sursa de energie, un circuit de stabilizare a tensiunii (regulator low-dropout care asigură alimentarea stabilă a circuitului prezentat, dar reduce şi pierderile de tensiune) şi un microcontroller cu modul BLE integrat. În această lucrare s-a prezentat şi varianta anterioară a dispozitivului descris iniţial, dar cu dimensiuni şi consum de energie mai mari. Acest produs nu necesita schimbarea bateriei deoarece avea integrată o sursă de energie care se încărca prin celule solare, care converteau energia luminoasă în energie electrică. Acest mod de colectare a energiei a venit şi cu dezavantajul de a fi nevoie ca dispozitivul să fie amplasat o perioadă lungă la lumină solară, ceea ce duce la încălzirea celorlalte componente integrate şi, prin urmare, la deteriorarea calităţii măsurătorilor furnizate de senzori. Dezavantajele prezentate au condus la realizarea produsului actual, având îmbunătăţite atât dimensiunile şi costurile de fabricaţie, cât şi calitatea informaţiilor extrase din mediu.

Ideile de implementare depind de cerinţele utilizatorului, având posibilitatea de implementare a unei aplicaţii care rulează pe un dispozitiv mobil sau care are o locaţie fixă. Ambele variante primesc datele preluate de la senzori în pachete livrate prin BLE după implementarea unui GATT observer. De asemenea, informaţiile acumulate vor putea fi salvate într-o bază de date cloud sau în una locală. Autorii lucrării au realizat experimente folosind platforma Raspberry Pi, stocând datele într-o bază de date locală SQLite. Două tabele au fost necesare pentru vizualizarea informaţiiloor, unul care stochează datele preluate de fiecare beacon, iar altul cu adresele MAC (Media Access Control) ale dispozitivelor. Testele efectuate au fost relevante în privinţa energiei consumate, recoltării energiei, elementelor de stocare ale energiei, puterii de transmisie, mărimii pachetelor primite, comparării cu alte sisteme, dar şi măsurătorilor efectuate în viaţa reală.

Din punct de vedere al consumului de energie, beacon-ul este programat să alterneze perioadele de somn cu perioade în care preia şi transmite datele, acestea fiind

setate la intervale de 3 secunde. În acest fel, în cadrul unui eveniment de trezire se consumă aproape 28 mA, urmând aproximativ 200 de ms cu o valoare de 9 mA. Folosind procesul anterior se ajunge la o perioadă de viaţă a bateriei de până la două luni şi jumătate. După realizarea mai multor teste s-a ajuns la un compromis între consumul de putere şi perioada de publicarea a pachetelor, astfel s-au propus 3 secunde între publicarea pachetelor de date şi 5 minute pentru perioadele de realizare a măsurătorilor. Tensiunea optimă pentru alimentare s-a calculat ca fiind de 3 V, aceasta se poate obţine cu mai multe tipuri de baterii împreună cu un circuit integrat LDO. Distanţa maximă la care se poate transmite semnalul de la senzor este de maxim 40 de metri în linie dreaptă pentru putere de alimentare cuprinsă între 2.6 V şi 3.6 V. În ceea ce priveşte mărimea pachetului de date, acesta poate fi micşorat prin eliminarea informaţiilor cu privire la nivelul la care se află bateria, modul de fucţionare al dispozitivului, acurateţea sau măsurarea presiunii atmosferice. Presiunea fiind un termen care îşi schimbă valoarea greu, ar fi suficient un singur bit prin care se indică dacă creşte sau scade în decursul unui interval de timp.

În lucrarea descrisă anterior măsurătorile au arătat că energia consumată în realizarea măsurătorilor din mediu este mult mai mare faţă de cea pentru publicarea pachetelor de date. De acest aspect trebuie ţinut cont în configurarea unui dispozitiv destinat monitorizării unui spaţiu pe o perioadă lungă de timp. De asemenea, s-a demonstrat posibilitatea de dezvoltare pe scară largă a unui sistem folosind sute de dispozitive, care preiau date şi creează pe baza acestora rezultate care să fie transformate în statistici sau în grafice relevante.

3 Analiză, proiectare, implementare

3.1 Analiză şi proiectare

Internet of Things este o tehnologie de ultimă oră care permite preluarea informaţiilor de la dispozitivele situate în jurul nostru, atât în interiorul locuiţelor, cât şi în mediul exterior. Alimentarea lor se face prin tehnologii de vârf care îmbină rezistenţa îndelungată cu dimensiunile reduse şi cu posibilitatea preluării energiei din surse naturale, cum ar fi lumina solară. Identificarea şi urmărirea lor în timp real se realizează prin tehnologii precum RFID (Radio-Frequency IDentification) care foloseşte undele radio şi permite conectarea la internet şi transmiterea informaţiilor identificabile. De asemenea, prin WSN (Wireless Sensor Networks) se acceptă construirea unor reţele de senzori interconectaţi care efectuează măsurători pentru diferiţi parametrii ambientali. Un astfel de dispozitiv este capabil să colecteze datele prin intermediul senzorilor şi să le proceseze, pentru ca mai apoi să se efectueze acţiuni automate pe baza deciziilor luate în urma procesării informaţiilor.

În figura 3.1 sunt prezentate straturile arhitecturale ale IoT. Arhitectura IoT trebuie să creeze o punte între componentele fizice şi cele virtuale, pentru a se putea realiza toate interacţiunile necesare prelucrării şi transmiterii informaţiilor. Interacţiunea fizică, în timp real între obiecte determină ca arhitectura să fie adaptabilă, pentru a permite dispozitivelor să interacţioneze dinamic între ele.

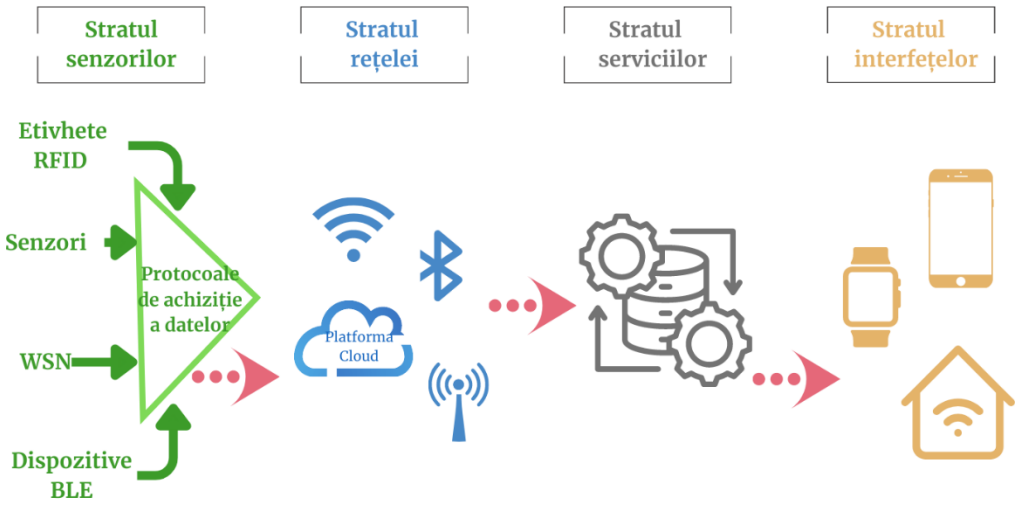


Figura 3. 1 Straturile arhitecturii IoT

Arhitectura IoT este formată din patru straturi: [21]

- Stratul senzorilor: Este responsabil cu integrarea senzorilor în obiectele disponibile în cadrul ecosistemului IoT. Aceştia capturează datele din mediu şi monitorizează starea obiectelor.

- Stratul rețelei: Furnizează instalația necesară pentru stabilirea conexiunilor, cu sau fără fir, între obiectele din rețeaua IoT. Asigură o comunicare sigură și fiabilă între dispozitive.
- Stratul serviciilor: Este responsabil pentru crearea și gestionarea serviciilor necesare utilizatorilor sau aplicațiilor. Se ocupă de sarcini precum preluarea datelor, stocarea și analizarea acestora, dar și de furnizarea serviciilor pe baza datelor colectate.
- Stratul interfeței: Cuprinde metode de intercațiune între utilizatori și sitemul IoT. Include interfețele utilizatorului, interfețele de programare ale aplicațiilor (API-ul) și alte protocoale care fac posibilă comunicarea cu dispozitivele IoT.

3.1.1 Android

Sistemul de operare Android este unul dintre cele mai populare sisteme de operare dezvoltate de Google și se bazează pe nucleul Linux, acesta fiind baza pentru gestionarea resurselor hardware și pentru a oferi funcționalități fundamentale. Android este popular printre utilizatori, dar și printre dezvoltatori datorită naturii open-source, având capacitatea de a îmbunătăți caracteristicile sistemului pentru a se adapta la cerințele tehnologiei actuale. În plus, este disponibil Android Software Development (SDK) prin care se oferă un depanator, un emulator de dispozitive bazat pe Quick Emulator, biblioteci, documentație și manual de utilizare. Acest set de instrumente facilitează procesul de dezvoltare și testare, oferind uneltele necesare pentru a crea aplicații performante și funcționale.

3.1.1.1 Arhitectura Android

În figura 3.2 este prezentată arhitectura Android, referindu-se la structura și organizarea componentelor sistemului de operare Android. Aceasta cuprinde diferite straturi și module care conlucrează pentru a obține interacțiunea dintre aplicații și hardware-ul dispozitivului.

Arhitectura Android este formată din patru nivele: [22]

- Aplicațiile: Stratul aplicațiilor se referă la partea superioară a arhitecturii Android, unde se dezvoltă și execută aplicațiile. Se proiectează folosind Java, Kotlin sau alte limbaje de programare și vin împreună cu un set de aplicații native (aplicații de sistem) precum: calendarul, browserul, contactele, hărțile sau altele.
- Infrastructura aplicațiilor: În arhitectura Android persoana care dezvoltă o aplicație are permisiunea să acceseze toate API-urile framework-ului de bază ale programului. Datorită acestui lucru, se pot gestiona comunicarea cu alte aplicații, manipularea datelor sau interfața cu utilizatorul. Infrastructura aplicației simplifică modul prin care se refolosesc unele componente, iar alte aplicații le pot accesa și le pot crește eficacitatea.
- Bibliotecile și Android Runtime: Bibliotecile sunt seturi de cod predefinite, scrise în limbejele de programare C sau C++, prin care se pot adăuga

funcţionalităţi extinse şi, de asemenea, se poate reduce timpul de dezvoltare. Android Runtime este compus dintr-o bibliotecă Java Core şi o maşină virtuală Dalvik. Biblioteca Java Core conţine clasele şi funcţiile de bază Java necesare pentru dezvoltarea aplicaţiilor. Aceasta oferă un se complet de funcţionalităţi de bază, cum ar fi manipularea şirurilor de caractere, gestionarea fişierelor sau tratarea excepţiilor.

- Nucleul Linux: Oferă practicile esenţiale, precum: gestionarea memoriei interne, protocoalele de internet, gestionarea proceselor şi alte servicii. Nucleul Linux asigură interacţiunea dintre hardware-ul şi software-ul dispozitivelor, permiţând dezvoltatorilor să creeze aplicaţii complexe şi inovatoare.

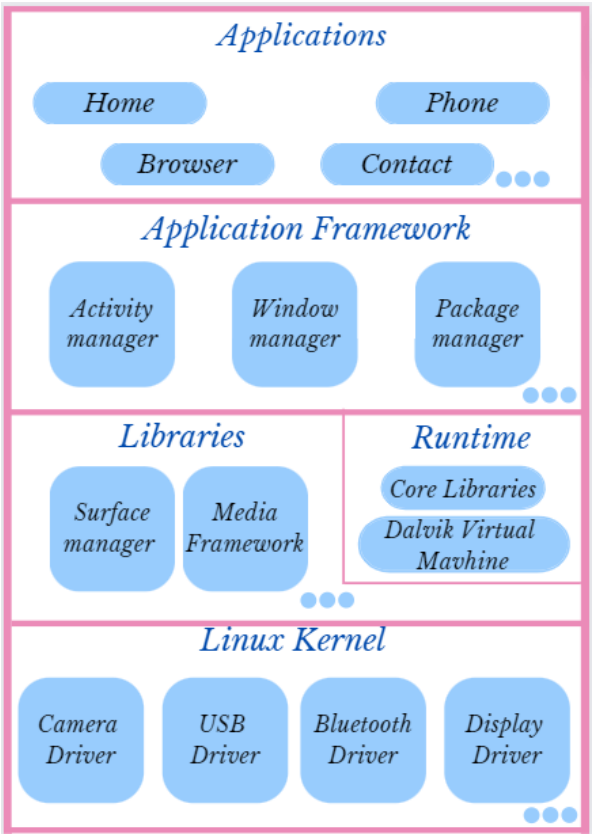


Figura 3. 2 Arhitectura Android

3.1.1.2 Permisiuni

În cadrul aplicaţiilor Android, permisiunile controlează accesul aplicaţiilor la diferite resurse şi funcţionalităţi, precum: locaţia, lista contactelor, stocarea, folosirea camerelor sau activarea Bluetooth-ului. Aceste permisiuni se pot grupa în trei nivele: normale, periculoase şi de sistem. Permisiunile normale protejează accesul la apelurile API care pot incomoda utilizatorul, dar nu pot cauza daune acestuia. Ele au fost introduse pentru a proteja clienţii împotriva schimbării constante a imaginii de fundal sau al altor setări nesemnificative, fără a putea accesa sau modifica datele personale sau sensibile ale utilizatorului. Permisiunile periculoase pot avea consecinţe potenţial dăunătoare, cum ar fi cheltuirea banilor sau colectarea informaţiilor de natură privată. Deoarece implică un

risc pentru securitatea şi confidenţialitatea celui care foloseşte aplicaţia, ele trebuie acordate explicit de către utilizator în timpul instalării sau rulării programului. Permisiunile de sistem permit accesul la cele mai periculoase setări, cum ar fi capacitatea de a şterge pachetele de aplicaţii. Sunt acordate doar programelor care sunt semnate cu certificatul specific al producătorului dispozitivului.

3.1.1.3 Android API

Andoroid API (Application Programming Interface) reprezintă o colecţie de clase, interfeţe şi construcţii pentru a se putea realiza între sistemul de operare şi resursele hardware ale dispozitivelor. Apelurile API sunt manipulate în mai multe etape: apelarea API-ului public din bibliotecă, invocarea unei interfeţe private în cadrul bibliotecii şi trimiterea unei cereri către un serviciu de sistem pentru a finaliza o sarcină. Platforma Android este în continuă evoluare, iar API-urile avansează cu fiecare versiune majoră a sistemului de operare. Acest lucru duce la necesitatea ca dezvoltatorii să ţină pas cu schimbările pentru a beneficia de noile funcţionalităţi şi îmbunătăţiri aduse de Android.

3.1.1.4 Componentele aplicaţiilor Android

Sistemul de operare Android permite intarea în aplicaţie din diferite puncte, nu există o funcţie principală care să îndeplinească acest rol. Există patru componente principale într-o aplicaţie Android (prezentate în figura 3.3),care îndeplinesc funcţii distincte.

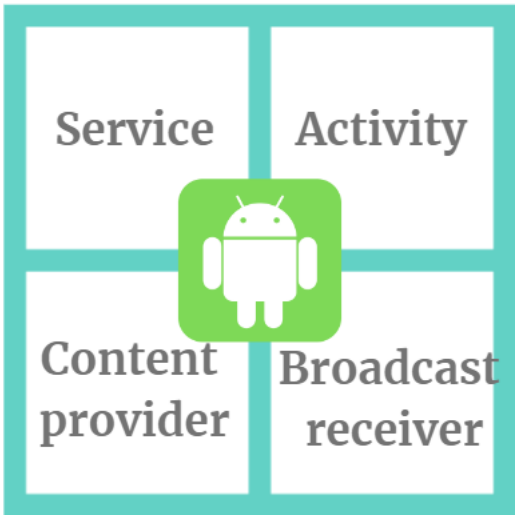


Figura 3. 3 Componentele aplicaţiilor Android

Serviciile sunt componente care rulează în fundal, fără să aibă o interfaţă prin care să comunice cu utilizatorul. Ele pot efectua operaţii de lungă durată sau pot procesa date în fundal, precum descărcarea fişierelor sau redarea muzicii. Componentele de serviciu rulează în mod implicit pe firul principal, iar pentru operaţiile intensive se recomandă rularea pe un fir de execuţie secundar iniţiat de dezvoltator.

O activitate este o interfaţă grafică prin care se comunică cu utilizatorul, fiind o subclasă a clasei Activity. O aplicaţie conţine de obicei o activitate principală care permite lansarea acesteia, din care se poate ulterior intra şi în alte activităţi. De fiecare dată când o activitate este pornită, cea anterioară este oprită, iar sistemul o păstrează într-o stivă. Aşa cum se poate vedea în figura 3.4, există şase stări în care se poate afla o aplicaţie

Android asociate cu metodele onCreate(), onStart(), onResume(), onPause(), onStop() şi onDestroy(). Atunci când aplicaţia este creată, aceasta nu este încă în funcţiune şi devine vizibilă utilizatorului abia după startare. În acest nivel activitatea este lansată, devinind interactivă şi activă atunci când trece în stadiul resumed. Atunci când o altă activitate este afişată peste cea curentă, aceasta trece în starea de pauză, rămânând încă în memorie. După închiderea ferestrei curente, închiderea aplicaţiei sau înlocuirea acesteia cu alta, o aplicaţie Android se află în starea de oprit, urmând distrugerea ei prin eliberarea tuturor resurselor. Ciclul de viaţă al aplicaţiilor este controlat de sistemul de operare, dar dezvoltatorul poate implementa diferite metode pentru a controla comportamentul

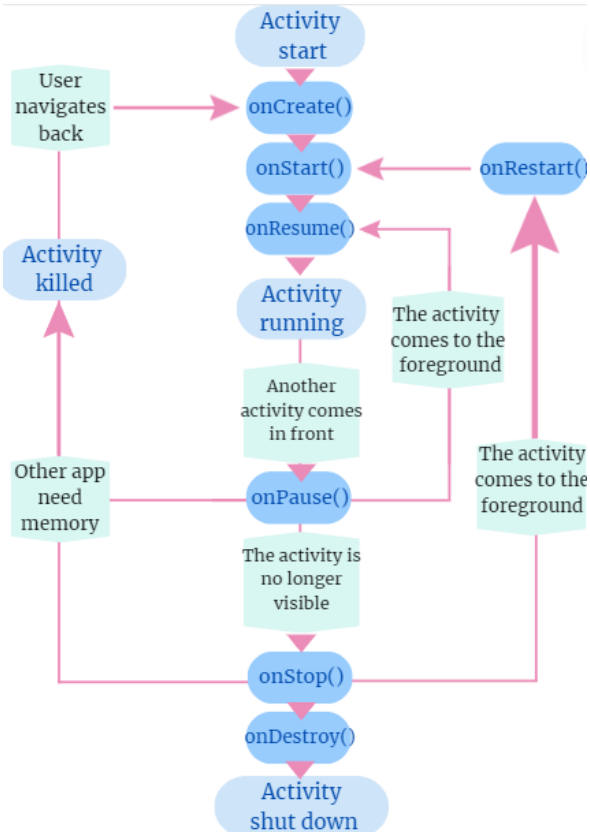


Figura 3. 4 Ciclul de viaţă al aplicaţiilor Android

aplicaţiei în fiecare stare.

Un content provider este o componentă a platformei care permite partajarea şi gestionarea datelor între diferite aplicaţii. Se defineşte schema datelor şi metodele prin care alte sisteme pot interacţiona cu acestea, dar nu se apelează direct metodele unui content provider, ci se foloseşte un obiect. Acesta facilitează posibilitatea ca alte aplicaţii să efectueze operaţiile de inserare (Create), interogare (Read), actualizare (Update) şi ştergere (Delete).

Un broadcast receiver în Android este o componentă ce face posibilă primirea şi răspunderea la difuzări (broadcasts), trimise de sistem sau alte aplicaţii. Un broadcast receiver este o cale de acces spre alte componente şi poate crea o comunicare sau poate alerta utilizatorul când există o informaţie de difuzat. Activarea activităţilor, a serviciilor

şi a broadcast receiver-ilor se realizează printr-un mesaj asincron numit intent. Aceste obiecte leagă între ele componentele individuale în timpul execuţiei [23].

3.1.1.5 Dezvoltarea aplicaţiilor Android (Arhitecturi)

Arhitecturile aplicaţiilor Android reprezintă nişte structuri prin care se organizează codul, se despart sarcinile şi se facilitează întreţinerea acestora. Trei dintre cele mai populare arhitecturi sunt: Model-View-Controller, Model-View-Presenter şi Model-View-ViewModel şi sunt descrise în lucrarea [24].

Model-View-Controller (MVC) este cea mai populară şi stă la baza celorlalte arhitecturi, are ca scop divizarea componentelor aplicaţiei în trei părţi principale: Model, View şi Controller. Modelul stochează datele în clase obiect sau date primare, acesta făcând posibilă şi preluarea informaţiilor de la distanţă. În Android, această componentă este reprezentată prin clase obiect java, încapsulează sursele de date, cum ar fi bazele de date, este responsabilă pentru logica de business a aplicaţiei, astfel oferă metode pentru obţinerea, manipularea sau actualizarea datelor. Elementul View este format din interfaţa disponibilă utilizatorului, acesta afişează datele din Model şi primeşte intrările de la persoana care utilizează aplicaţia. Este o combinaţie între fişierele cu resursele pentru afişare şi activităţi sau fragmente, cele din urmă fiind necesare deoarece fişierele XML nu au control complet asupra interfeţei utilizatorului (UI). Controllerul primeşte intrările corespunzătoare acţiunilor consumatorului, actualizează modelul în funcţie de aceste intrări şi, în unele cazuri, transmite schimbările înapoi pentru a se actualiza afişajul. Un exemplu de parte a controllerului sunt metodele onClickListener(), prin acestea se înregistrează o serie de evenimente din view şi efectuează activităţi corespunzătoare cererilor clientului.

În figura 3.5 este prezentată interacţiunea dintre componentele arhitecturii MVC. Există trei posibilităţi de interacţiuni între cele 3 părţi: controllerul notifică schimbarea unei stări în view, de exemplu schimbarea culorii unei secveţe text; controllerul notifică modelul, care acţionează asupra componentelor necesare îndeplinirii taskurilor; controllerul trimite un mesaj părţii view, aceasta cere modelului noile stări, iar apoi se actualizează pe ea însăşi.

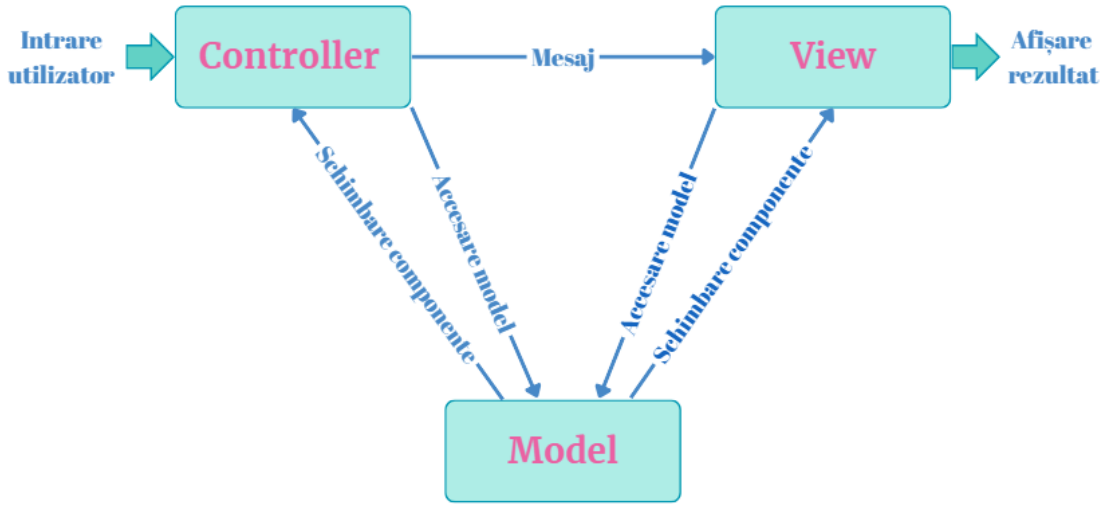


Figura 3. 5 Interacţiunea componentelor arhitecturii MVC

Model-View-Presenter (MVP) se bazează pe conceptul MVC, modelul şi view-ul având aceleaşi funcţionalităţi ca şi în arhitecctura de bază. În plus conţine trei elemente încapsulate în partea de Presenter (command, selector, interactor). Presenterul primeşte acţiunile clientului din View, procesează aceste informaţii şi acţionează asupra modelului sau interfeţei corespunzător. Selectorul precizează submulţimile de date cu care urmează să se lucreze în aplicaţie, conform cerinţelor utilizatorului. Comanda reprezintă o listă cu acţiunile care pot să fie întreprinse, iar componenta interactor se ocupă cu procesarea şi manipularea datelor prin care se implementează logica aplicaţiei.

În figura 3.6 este vizibilă interacţiunea dintre componentele arhitecturii MVP. Diferenţa dintre MVC şi MVP porneşte de la faptul că în cazul actual, componenta View trimite comenzile de la utilizator către Presenter. În cele mai simple cazuri componentele din Presenter iau anumite decizii în funcţie de cerinţa clientului şi actualizează direct interfaţa cu utilizatorul, dar există şi cazuri mai complexe în care este nevoie ca modelul să fie accesat pentru citirea unor date sau pentru actualizarea acestora.

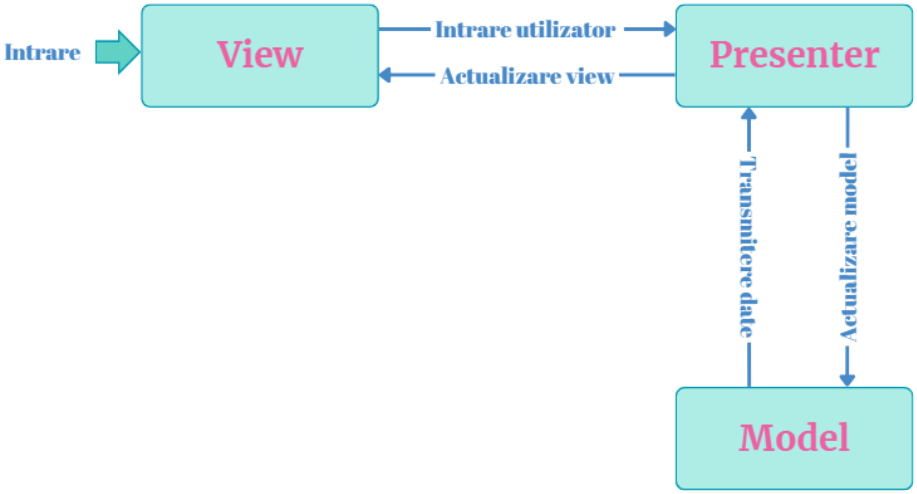


Figura 3. 6 Interacţiunea componentelor arhitecturii MVP

Model-View-ViewModel (MVVM) este relevant în aplicaţiile în care componenta View este realizată de un proiectant, nu de un dezvoltator clasic de aplicaţii. View este interfaţa cu care interacţionează utilizatorul, Modelul este conceput din date, la fel ca şi la cele două arhitecturi anterioare. View-Model este destinat transmiterii datelor şi operaţiilor spre View, dar şi pentru a gestiona logica părţii vizuale.

Interacţiunea dintre componentehe arhitecturii MVVM este arătată în figura 3.7.

Există două tipuri de conexiuni între View şi ViewModel: conexiunea clasică, folosită şi în cazurile MVC şi MVP şi conexiunea de legare a datelor (databinding connection). Databinding este este un mecanism prin care datele sunt legate automat între View şi ViewModel, în acest fel nu este necesară controlarea manuală a actualizărilor de date petrecute în cele două componente ale arhitecturii. Legăturile de date sunt stabilite prin expresii, astfel fiecare modificare a informaţiilor este vizibilă în View fără să fie nevoie de cod suplimentar.

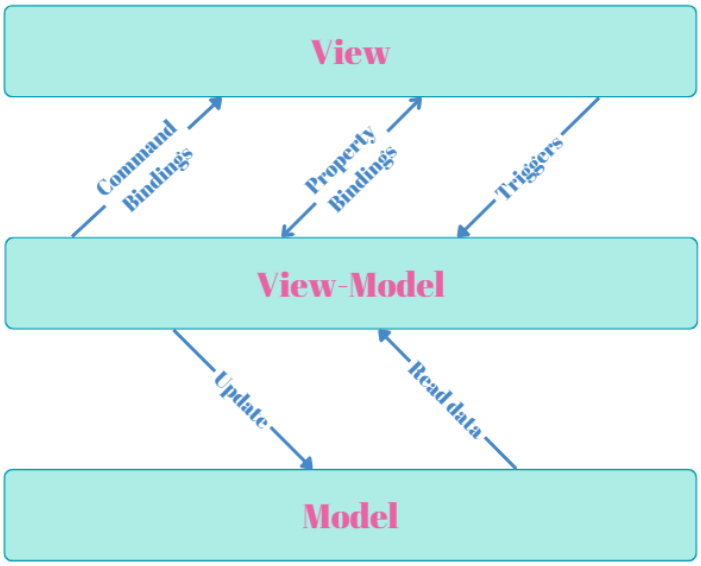


Figura 3. 7 Interacţiunea componentelor arhitecturii MVVM

3.1.2 Bluetooth Low Energy

Tehnologia de comunicare wireless Bluetooth a fost concepută pentru a permite dispozitivelor să schimbe informaţii între ele, limitându-se fizic la o distanţă redusă între acestea. Odată cu evoluţia tehnologiei, au fost adăugate mai multe cazuri complexe de utilizare, ceea ce a dus la necesitatea creşterii lăţimii de bandă. Bluetooth Low Energy este proiectat ca o tehnologie complementară pentru Bluetooth, dar urmăreşte îndeplinirea altor cerinţe, respectiv optimizarea consumului de energie, adresându-se unor segmente diferite de piaţă. Folosind BLE nu se obţin rate ridicate de transfer şi nu se poate menţine conexiunea pe o durată lungă de timp, dar este potrivit pentru o gamă largă de aplicaţii datorită cerinţelor energetice.

Pentru a înţelege arhitectura BLE trebuie să discutăm despre cele trei componente principale (aplicaţie, host şi controller), aşa cum se poate vedea în figura 3.8. Stratul aplicaţiei este format din interfaţa cu utilizatorul, arhitectura generală a aplicaţiei şi logica din spatele acesteia. Aplicaţia accesează datele BLE oferite de dispozitivele periferice prin folosirea bibliotecilor sau API-urilor specifice.

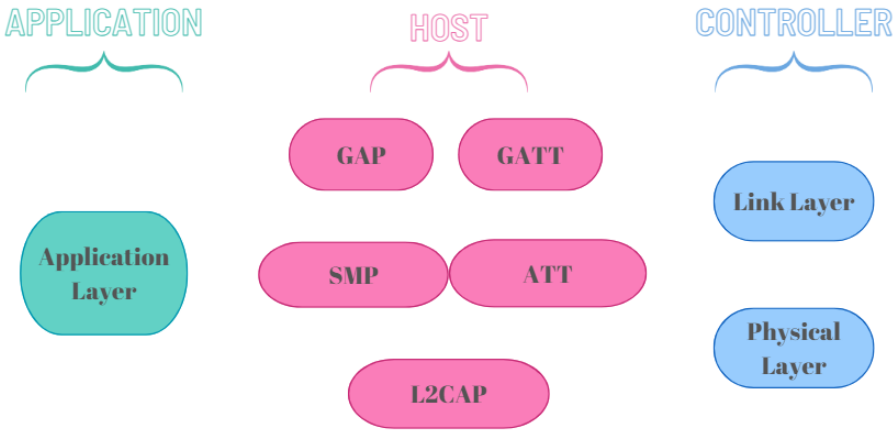


Figura 3. 8 Arhitectura BLE

Host-ul este partea software care controlează comunicarea şi interdependenţa cu dispozitivele BLE, fiind implementat pe un dispozitiv gazdă, cum ar fi un telefon inteligent, un microcontroller sau un calculator.

Generic Attribute Profile (GATT) defineşte un format şi un ansamblu de reguli necesare pentru procesul de transferare a datelor între dispozitivele BLE. Aceste dispozitive pot să indeplinească două roluri fundamentale (client şi server), şi astfel se creează interacţiunea dintre ele. Clientul este programat să trimită şi să primească cereri de la server, fiind necesară şi procedura de descoperire a serviciului sau atributului, deoarece nu sunt cunoscute în prealabil. Atributele sunt valori numerice, cuvinte sau flag-uri prin care se păstrează informaţiile din cadrul unui serviciu (valoarea curentă a presiunii atmosferice), iar serviciile sunt grupări de atribute prin care se descrie o funcţionalitate. Atributele sunt compuse dintr-un UUID, un set de permisiuni, o valoare şi un handle de 16 biţi, utilizat pentru a accesa şi manipula atributul. Serverul are scopul de a transmite actualizări, de a stoca şi a pune la dispoziţie clientului datele caracteristice utilizatorului.

Prin organizarea datelor furnizate de GATT, prezentată în figura 3.9, se permite reutilizarea atributelor şi se facilitează comunicarea dintre client şi server. Caracteristicile stochează datele utilizatorului, fiecare este identificată printr-un UUID, iar descriptorii sunt formaţi mereu dintr-un singur atribut prin care se oferă informaţii suplimentare despre o caracteristică sau un serviciu.

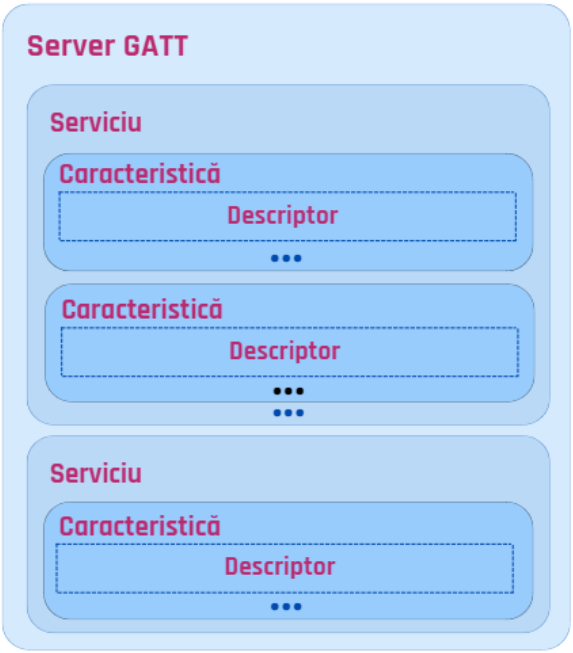


Figura 3. 9 Ierarhia GATT

Generic Access Profile (GAP) este un profil prin care se setează modul în care dispozitivele BLE comunică între ele şi accesează informaţiile dorite de la alte mecanisme asociate. În procesul de conectare a acestora există două ruluri pe care le poate avea un dispozitiv: observer (se vizualizează pachetele şi se stabileşte dacă este saun nu nevoie să se iniţieze o conexiune, în funcţie de cazul de utilizare) şi broadcaster (publică periodic pachete de date care conţin informaţii despre dispozitiv, serviciile oferite de acesta şi alte

detalii relevante). GAP însoţeşte faza de acces şi descoperire şi este esenţial pentru a stabili conexiunea cu mecanismul periferic de la care se citesc datele, ulterior GATT se va ocupa de interacţiunea cu datele furnizate.

Attribute Protocol (ATT) este protocolul de bază în BLE şi controlează transmisia de date între dispozitive. Prin urmare, prin intermediul ATT este fixat un set de reguli legate de accesarea şi manipularea atributelor în cadrul dispozitivelor ce utilizează tehnologia BLE. Operaţiile realizabile în cadrul protocolului sunt: citirea valorilor atributelor, scrierea, indicaţiile şi notificările. Scrierea unei valorii se poate executa fără să se aştepte un răspuns din partea dispozitivului asociat, cu confirmare sau folosind o semnătură pentru a autentifica datele. Indicaţiile sunt înregistrări asincrone deprinse de server pentru client, care sunt iniţiate dacă partea clientului solicită actualizarea valorilor atributelor. Notificările sunt similare cu indicaţiile, diferenţa constă doar în faptul că cele din urmă necesită confirmare din partea clientului atunci când sunt recepţionate informaţiile.

Security Manager Protocol (SMP) este o componentă cheie prin care se definesc regulile privind procesul de autentificare şi pentru asigurarea confidenţialităţii datelor transmise între dispozitive. Logical Link Control and Adaptation Protocol (L2CAP) este un protocol prin care se face posibilă gestionarea conexiunilor logice şi de accesare a unor caracteristici care permit transmiterea datelor între dispozitivele BLE.

Controllerul este format din hardware-ul real care are încorporat un modul Bluetooth, ce asigură comunicarea prin care se favorizează transmiterea şi primirea semnalelor. Această componentă a arhitecturii BLE este compusă din două straturi: stratul fizic şi cel de legătură. Stratul fizic constă din circuitele analogice care transmit şi recepţionează semnalele radio necesare pentru comunicarea BLE. Nivelul de legătură este responsabil cu scanarea, crearea şi menţinerea conexiunilor între dispozitive şi poate să se găsească în una dintre următoarele cinci stări: aşteptare, publicitate, scanare, iniţiere, conexiune şi sincronizare.

În figura 3.10 se găsesc stările în care se poate afla un beacon şi comunicarea dintre aceste etape. Stadiul de aşteptare poate să fie introdus din orice altă stare şi reprezintă momentele în care nu există nici transmisii şi nici recepţii de pachete. În stadiul de publicitate, dispozitivul se numeşte advertiser şi emite pachetele de date pentru a anunţa prezenţa sa sau pentru a permite altor dispozitive să iniţieze o conexiune. Scanarea este responsabilă pentru detectarea şi citirea pachetelor transmise, iar după detectarea unui dispozitiv în urma scanării se iniţiază procesul de conectare. În acest timp se trimite o cerere de conexiune şi se aşteaptă un răspuns din partea aparatului ţintă, urmând schimbul de date după trecerea în starea de conectare. Stratul de legătură în care se aşteaptă periodic pachete de date se numeşte nivelul de sincronizare şi poate să fie introdus din modul de aşteptare.

Stratul fizic (Physical Layer) reprezintă cel mai inferior nivel al stivei BLE, care se ocupă cu transmiterea şi recepţionarea datelor la nivel fizic. BLE, la fel ca şi Bluetooth clasic, foloseşte banda de 2.4 GHz şi transmite mesajele în mod repetat folosind trei canale comune (37, 38 şi 39) dintr-un total de 40. Intervalul de publicare al pachetelor variază între 20 ms şi 10.24 s, în funcţie de cerinţele utilizatorului şi posibilităţile de

implementare pe care le are dezvoltatorul. După fiecare interval de publicare urmează o întârziere, care poate să fie nulă sau să crească până la ordinul zecilor de milisecunde.

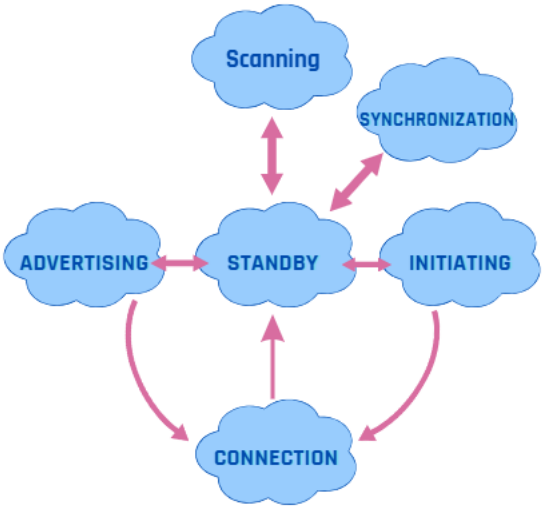


Figura 3. 10 Stratul de legătură

BLE are două moduri de funcţionare (conecatre şi advertisement). Pentru cazul în care nu este necesară conectarea, datele sunt transferate unidirecţional, doar de la dispozitivele terminale către scannere. Odată cu primirea mesajelor, scannerul poate să trimită cereri de scanare, care primesc un răspuns constituit şi din informaţii suplimentare. De asemenea, există posibilitatea de a iniţia şi conexiuni, în timpul cărora se stabileşte un canal dedicat de comunicare între ambele dispozitive şi se utilizează pentru schimbul de date[25].

În figura 3.11 sunt arătate scanarea şi publicarea datelor în cadrul modului de funcţionare fără conectare. Se poate observa că scannerele schimbă canalele de comunicare la fiecare interval de scanare şi aşteaptă primirea mesajelor de pe fiecare canal în timpul intervalului scan window.

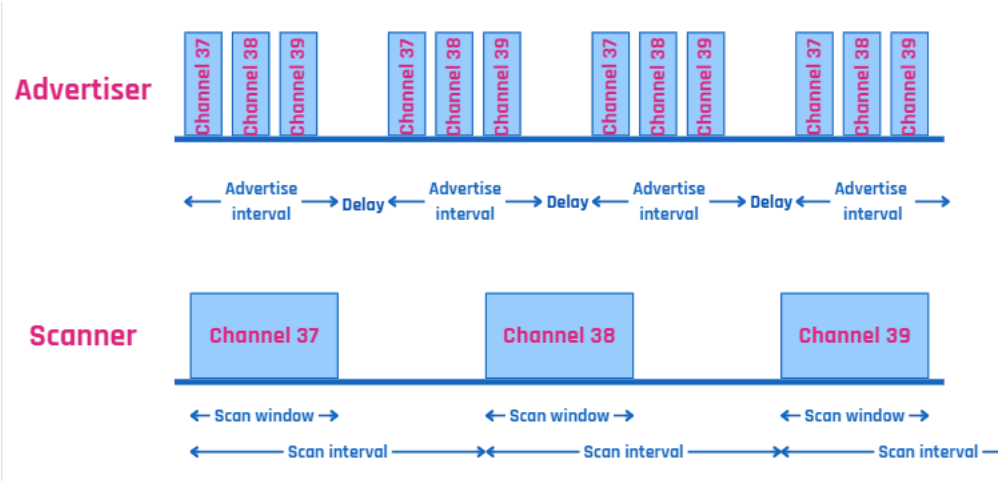


Figura 3. 11 Modul de funcţionare fără conectare

În arhitectura Bluetooth Low Energy, stratul aplicaţiei (Application Layer) se ocupă de funcţionalităţile specifice interacţiunii cu utilizatorul, furnizând interfaţa şi modulele

necesare interacţionării cu dispozitivele BLE. Acest nivel accesează serviciile şi caracteristicile, administrează conexiunile, gestionează evenimentele şi implementează logica aplicaţiei.

3.1.3 Dispozitive Bluetooth Low Energy

Un dispozitiv BLE este dedicat pentru comunicarea wireless eficientă şi un consum redus de energie, folosit într-o gamă largă de aplicaţii. Beaconul utilizat în cadrul proiectului a fost conceput din punct de vedere al hardware-ului şi al software-ului încorporat, care controlează funcţionalităţile şi comportamentul aparatului, de către prof. dr. ing. Silviu Corneliu Folea și prof. dr. ing. George Dan Mois.

Figura 3. 12 Dispozitivul BLE

Beaconul a avut mai multe stadii de dezvoltare, versiunea anterioară având dimensiuni mai vaste şi un consum mai mare de energie, încărcarea bateriei realizându-se prin celulele solare. Dispozitivul actual are proporţii reduse, un consum mai mic de energie şi elimină necesitatea aşezării pe o perioadă mai lungă la soare pentru încărcare, deoarece foloseşte o baterie de 3V CR2032. În figura 3.12 se poate vizualiza dispozitivul descris în următoarea parte a lucrării.

În figura 3.13 este redată arhitectura pe care se bazează proiectul prezentat în lucrarea de faţă pentru monitorizarea calităţii aerului. Dispozitivele dedicate preluării informaţiilor din mediu vor transmite datele aparatelor mobile, care folosesc sistemul de operare Android. Această procedură se realizează folosind tehnologia Bluetooth Low Energy, telefoanele inteligente iniţiază procedura de scanare şi preiau pachetele de date publicate de beacon-uri. Aceste informaţii sunt salvate în timp real într-o bază de date cloud, odată cu locaţia curentă a utilizatorului. Din această bază de date, utilizatorii pot vizualiza măsurătorile efectuate într-un interval de timp dorit, sub formă de grafice, într-o aplicaţie web uşor de folosit.

Conform lucrărilor [17] şi [27], dispozitivul folosit este capabil să măsoare valoarea temperaturii, presiunii atmosferice, umidităţii relative, un echivalent al compuşilor organici volatili biogenici şi al concentraţiei de dioxid de carbon, iar la final să stabilească un indice referitor la gradul de poluare numit IAQ. Măsurătorile se efectuează o dată la cinci minute şi pachetele cu datele rezultate vor fi publicate la fiecare zece secunde pentru a păstra consumul de energie cât mai redus. Deci, aplicaţia care scanează trebuie să

determine când un nou set de date este disponibil. Distanţa maximă la care se pot transmite datele este 40 m, acest lucru reprezentând o restricţie în ceea ce priveşte dezvoltarea unor aplicaţii care preiau datele de la beacon.

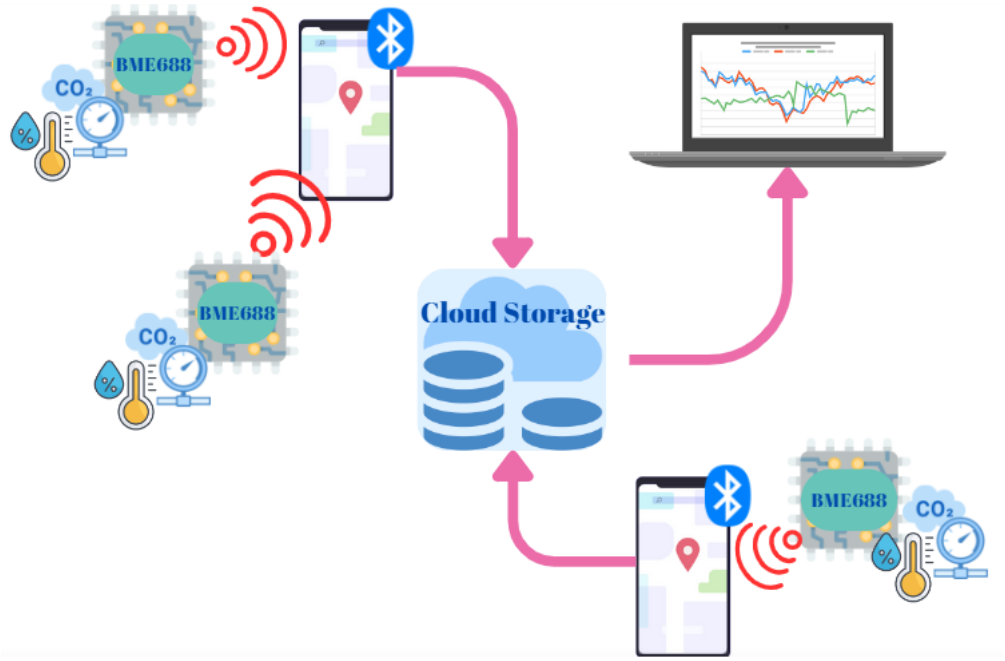


Figura 3. 13 Arhitectura sistemului de monitorizare a aerului

Arhitectura dispozitivului folosit se bazează pe patru elemente, aşa cum se poate observa în figura 3.14. O componentă importantă este modulul Bluetooth CYBLE-222014 -01 de la Cypress Semiconductor, care asigură o soluţie inteligentă pentru comunicarea BLE şi este adesea utilizat în dezvoltarea dispozitivelor portabile şi a senzorilor inteligenţi. Bateria CR2032 este suficientă pentru menţinerea dispozitivului activ pe o perioadă de aproximativ 4 luni. Butonul şi ledul RGB conectate la dispozitiv au rolul de a arăta utilizatorului calitatea aerului în momentul apăsării butonului, verde însemnând cel mai bun caz, în care aerul nu este poluat.

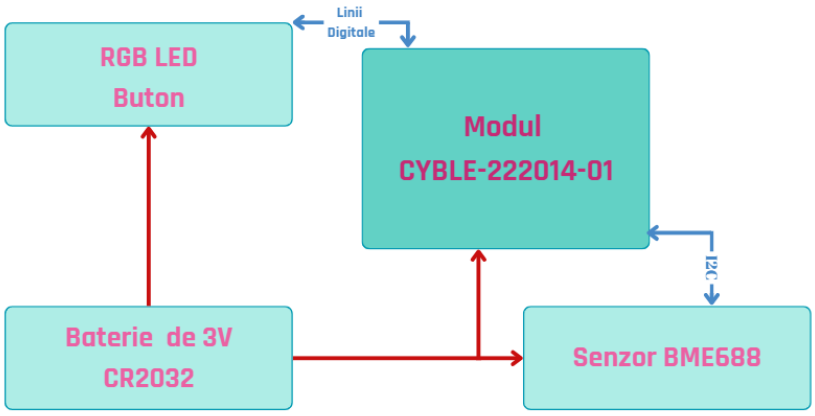


Figura 3. 14 Arhitectura dispozitivului BLE

Sezorul BME688 produs de Bosch este primul senzor de gaz cu Inteligenţa Artificială (AI) care este integrat într-un pachet robust, de dimensiuni reduse (3.0 x 3.0 x 0.9 mm^3). Senzorul de gaz detectează compuşii organici volatili şi alte gaze, acestea fiind măsurate

în părți pe milion. Un astfel de tip de senzor poate să fie folosit într-o gamă largă de aplicații mobile sau să fie integrat în dispozitive specifice caselor inteligente. Scenariile în care deviceul poate să fie utilizat sunt: măsurarea calității aerului în spațiile interioare sau exterioare, detectarea mirosului neobișnuit a unor gaze (ceea ce poate preveni un incediu), detecția incendiilor din locurile sălbatice sau depistarea mâncărurilor alterate, astfel se poate indica prezența unor bacterii[28].

3.1.4 Aplicație web

Aplicațiile web au devenit din ce în ce mai populare în lumea digitală datorită versatilității și ușurinței cu care sunt livrate serviciile. Acestea sunt aplicații software care rulează pe servere și sunt accesate de către utilizatori prin intermediul unui browser web. Există o mare asemănare între aplicațiile web și site-urile web, diferența dintre acestea constând în capacitatea clientului de a influența starea logicii de pe server. Sistemul în care serverul web face posibilă influențarea funcționalității prin intermediul browserelor web, este considerat o aplicație web.

Arhitectura unei aplicații web se referă la felul în care funcționalitatea aplicației este oferită prin comunicarea dintre diferitele module ce o compun. Prin proiectarea unei structuri se oferă performanță și ușurință în mentenanță. În figura 3.15 se poate vedea arhitectura tipică unei aplicații web consituită dintr-un client browser, un server pe care rulează un program și o bază de date a cărei resurse vor putea fi accesate prin intermediul internetului.

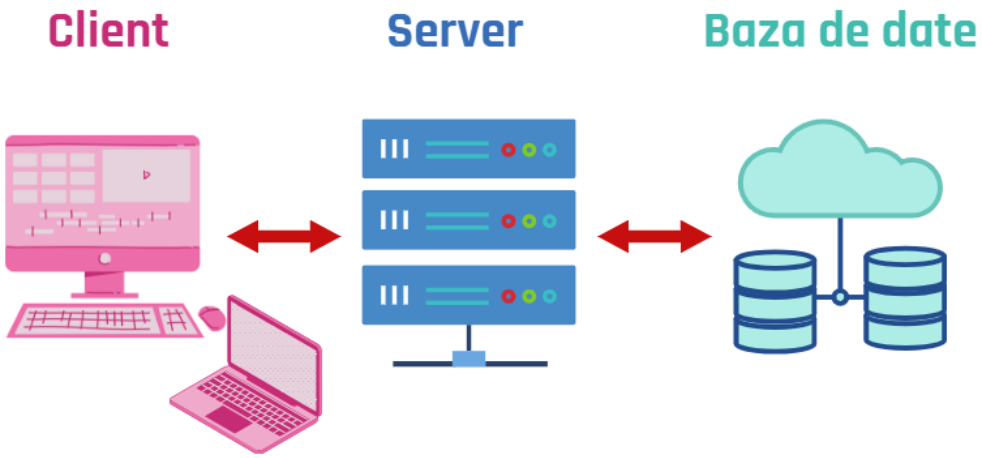


Figura 3. 15 Arhitectura aplicațiilor web

În contextul acestei arhitecturi, serverul este responsabil de interacțiunea cu utilizatorul și de afișarea datelor și a interfeței grafice. Cu ajutorul limbajelor de programare HTML, CSS sau JavaScript se poate dezvolta frontendul unei aplicații, în plus există posibilitatea utilizării unor framework-uri care să ofere o serie de facilități aplicațiilor. Cele mai populare framework-uri cunoscute sunt: Angular, React sau Django. Baza de date stochează și gestionează datele de care este nevoie într-o anumită aplicație, iar serverul o poate utiliza pentru a salva și accesa informațiile necesare pentru buna funcționare a programului.

Serverul gestionează cererile primite de la client şi furnizează răspunsurile corespunzătoare conform logicii implementate de dezvoltator. Există o varietate largă de servere web care pot fi utilizate pentru realizarea aplicaţiilor web, unele dintre aceste pot fi: Apache HTTP Server, Nginx, Microsoft IIS sau Apache Tomcat. Unul dintre cele mai populare este Node.js, acesta fiind o platformă construită pe baza motorului JavaScript al browserului Chrome. În cadrul acestui server se utilizează un model de intrare/ieşire bazat pe evenimente, numit asincron, care ajută la procesarea mai multor cereri simultan, fără să fie blocat firul de execuţie principal.

Arhitectura Node.js prezintă o soluţie software în care nu există concurenţă la nivelul aplicaţiei, datorită rulării codului pe un singur fir de execuţie, iar această abordare simplifică posibilitatea de extindere aplicaţiei. Modelele gândite să folosească câte un fir de execuţie per cerere sunt mai puţin optime deoarece nu pot administra eficient un număr foarte mare de conexiuni fără a consuma resursele sistemului. Acest tip de arhitectură cu mai multe fire de execuţie este mai uşor de gândit pentru dezvoltatori, însă mai puţin performant. În figura 3.16 se observă arhitectura specifică platformei Node.js.

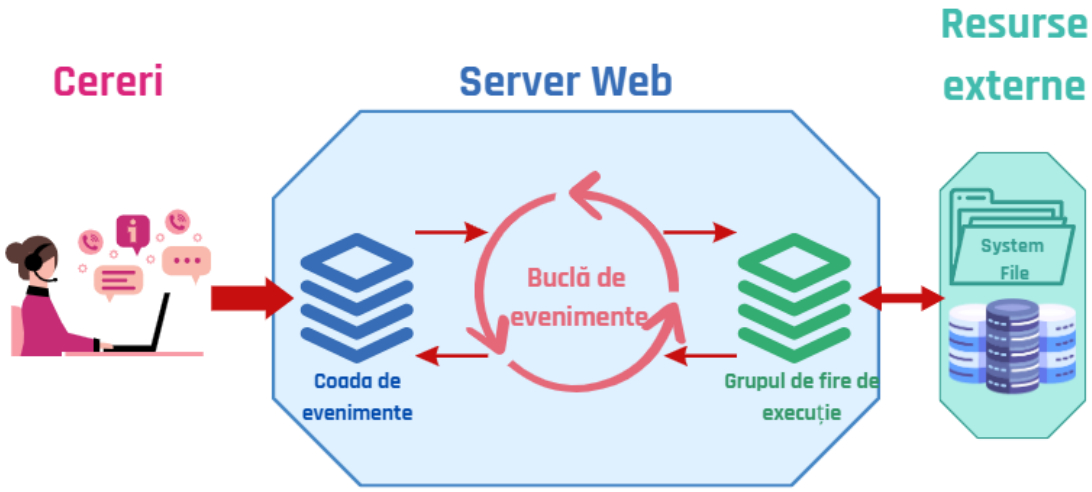


Figura 3. 16 Arhitectura detaliată a platformei Node.js

Utilizatorul formulează diferite cereri pe care serverul trebuie să le proceseze şi să furnizeze răspunsuri corespunzătoare. Aceste solicitări pot să fie simple sau complexe (blocante), care necesită un timp semnificativ pentru a fi finalizate şi blochează execuţia ulterioară a codului până când operaţie va fi terminată. Serverul Node.js preia cererile de la clienţi şi le răspunde fiecăruia în parte. Aceste request-uri vor fi stocate într-o coadă de evenimente şi trimise una câte una spre bucla evenimentelor. Coada funcţionează pe principiul FIFO (First-In-First-Out), adică întâmplările sunt procesate în ordinea în care au fost adăugate. Event loop-ul (bulca de evenimente) le preia din coadă, le procesează şi rulează în mod continuu în timp ce trimite evenimentele spre funcţiile de tratare corespunzătoare. Grupul de fire de execuţie este format din firele disponibile în serverul Node.js, care vor prelua sarcinile pentru a reuşi să îndeplinească toate cererile preluate de la utilizatori. Resursele externe permit ca aplicaţia să solicite informaţii sau să execute operaţii folosind rezerve din afara serverului[29].

Comunicarea dintre server şi baza de date se bazează pe protocoale web precum HTTP (Hypertext Transfer Protocol) care intermediază transferul de date, cereri de

autentificare sau schimb de informaţii. Cele mai comune acţiuni întreprinse prin HTTP sunt cererile GET, POST, PUT, DELETE. De exemplu, prin cererea GET se obţin resursele de la server, cum ar fi o pagină web sau imagini. Pentru a specifica adresele resurselor accesate sunt utilizate URL-uri (Uniform Resource Locators), ele conţin adresa serverului, calea către resursă sau alte informaţii. Pentru interpretarea rezultatului unei cereri se folosesc coduri de stare (200 pentru cerere reuşită, 404 pentru resursă inexistentă). În aces fel, clientul este atenţionat în cazul unei erori, explicitând cauza acesteia [30].

În cadrul unei aplicaţii web, protocolul HTTP ajută la transferul fişierelor între client şi server, acestea putând fi de diferite tipuri: HTML, CSS, JavaScript, imagini, fişiere multimedia sau alte tipuri mai complexe. Fişierele HTML (Hypertext Markup Language) au în componenţa lor codul relevant pentru structura, conţinutul şi modul în care se prezintă paginile web, adică reprezintă scheletul structurii unei aplicaţii web. Pot include şi elemente interactive precum: link-urile către alte pagini web, imagini sau scripturi JavaScript prin care se implementează funcţionalitatea dinamică. Pentru prezentarea vizuală a unei aplicaţii se folosesc fişiere CSS, în care se scriu reguli şi stiluri prin care se controlează modul în care elementele HTML sunt plasate şi formatate într-un browser. Delimitarea stilului în acest fel de colecţii de date este benefică pentru reutilizabilitatea codului, eficienţă şi performanţă şi pentru posibilitatea integrării în alte tehnologii. Iar pentru manipularea datelor şi asigurarea comunicării cu serverul sunt cerute fişierele JavaScript, acestea aducând funcţionalităţi dinamice. Acestea pot verifica datele introduse de utilizator sau furniza mesaje de eroare, iar prin împărţirea codului în module separate, facilitează întreţinerea codului.

Unul dintre cele mai importante elemente într-o aplicaţie web reprezintă API-urile datorită rolului pe care în joacă în cadrul sistemului. Acestea facilitează comunicarea între componentele aplicaţiei si serviciile externe. API-urile sunt împărţite în două tipuri principale: interne (dezvoltate în carul aplicaţiei) şi externe (disponibile datorită unor servicii externe). Api-urile interne sunt create de dezvoltatorii aplicaţiei şi sunt utilizate pentru a gestiona datele şi pentru a efectua operaţii de bază, dar asigură şi comunicarea între componentele programelor software. În schimb, cele externe permit accesarea şi manipularea datelor din surse externe precum: procesarea plăţilor online, afişarea hărţilor interactive sau extragerea unor date din conturi de pe reţelele de socializare. Avantajele folosirii API-urilor în cadrul aplicaţiilor web pot fi: posibilitatea reutilizării proiectelor, beneficierea de funcţionalităţi complexe sau integrarea serviciilor externe populare.

3.2 Implementare

3.2.1 Aplicaţia Android

Aplicaţia Android reprezintă partea componentei software care permite utilizatorului să iniţieze scanarea pentru dispozitivele BLE, salvează valorile preluate de la senzori într-o bază de date cloud, afişează rezultatele primei scanări şi prezintă în acelaşi timp şi locaţia pe hartă a clientului. În figura 3.17 se pot observa toate scenariile care se pot efectua în cadrul aplicaţiei.

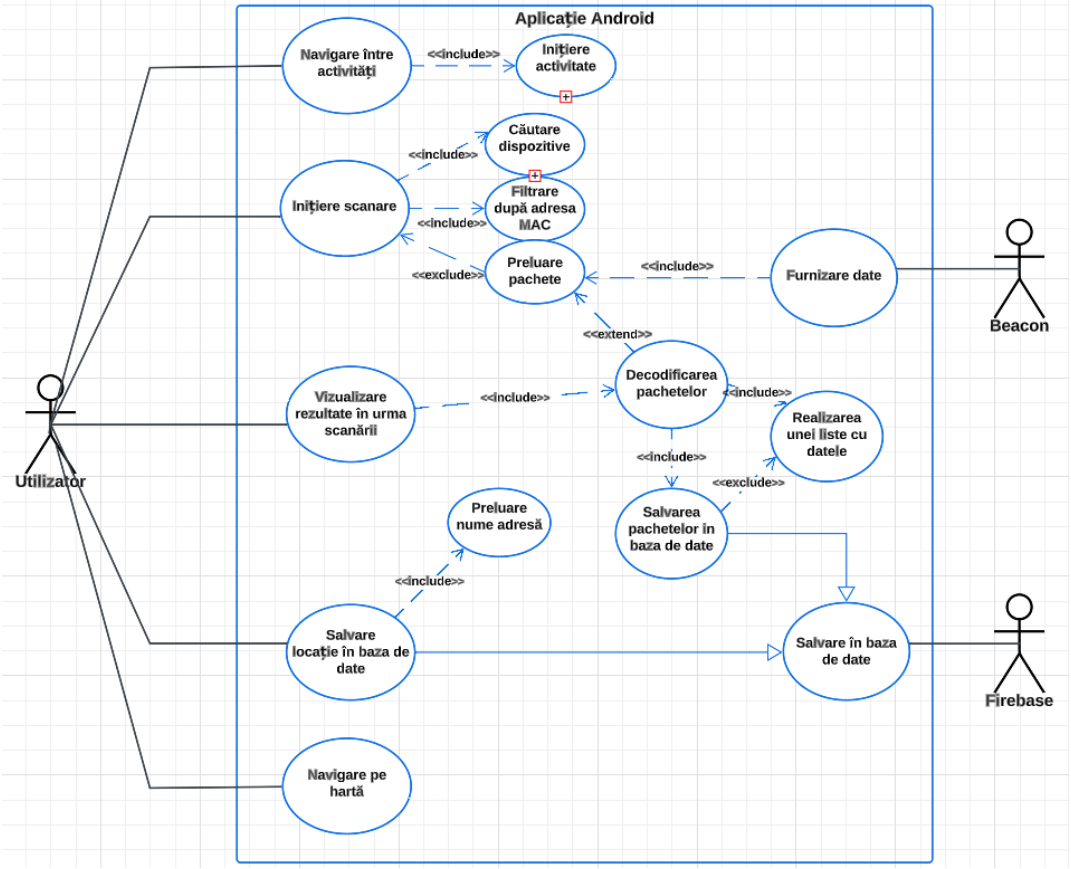


Figura 3. 17 Diagrama cazurilor pentru aplicaţia Android

Implementarea codului a fost făcută în mediul integrat de dezvoltare Android Studio, folosind limbajul de programare Java. Cel mai important pas pentru dezvoltarea acestui program software, dedicat telefoanelor mobile care folosesc sistemul de operare Android, este de a intermedia procesul de scanare a dispozitivelor care folosesc BLE situate în vecinătatea utilizatorului. Pentru a efectua această procedură este nevoie în prealabil să se includă acordurile corespunzătoare tehnologiei Bluetooth. În fişierul AndroidManifest.xml se vor preciza permisiunile necesare accesării funcţiilor generale Bluetooth, cereri care oferă aplicaţiei drepturi administrative pentru a realiza acţiuni avansate legate de Bluetooth (scanare, conectare, schimbarea stărilor) şi permisiuni pentru a accesa informaţiile legate de localizarea dispozitivului mobil. Aşa cum se poate vedea în figura 3.18, acordul pentru localizarea precisă şi cea mai puţin explicită este cerută pentru aplicaţiile care interacţionează cu aparate BLE, doarece aceste date sunt utilizate pentru a filtra dispozitivele în funcţie de proximitate.

După precizarea acestor permisiuni în fișierul Manifest, este nevoie să solicităm în interiorul codului, la momentul potrivit, pornirea lor dacă nu sunt deja activate. Aşadar, în momentul în care este deschisă aplicaţia, utilizatorul primeşte un mesaj care cere activarea Bluetooth-ului sau a locaţiei, dacă acestea nu sunt deja iniţiate pe dispozitiv.

```
<uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />
<uses-permission android:name="android.permission.BLUETOOTH_SCAN" />
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

Figura 3. 18 Perminiuni necesare pentru BLE

Primul pas în realizarea scanării dispozitivelor BLE este obţinerea unei instanţe a clasei BluetoothAdapter şi a unei referinţe a adaptorului Bluetooth prin apelarea metodei getDefaultAdapter(). Această clasă din Android permite interacţiunea cu modulul Bluetooth, adică activarea, dezactivarea tehnologiei de comunicare, scanarea beacon-urilor disponibile, conectarea şi comunicarea cu dispozitivele. De asemenea, trebuie iniţiată o instanţă a unui obiect furnizat de clasa BluetoothLeScanner pentru a avea acces la metodele necesare pornirii şi opririi scanării BLE, dar şi pentru a face posibilă gestionarea rezultatelor scanării.

Pentru a primi notificări cu rezultatele scanării s-a folosit metoda onScanResult() din interiorul clasei onScanCallback. De asemenea, pentru a căuta explicit după beacon-uri care publică pachete corespunzătoare calitatăţii aerului, s-a implementat un filtru care depinde de adresa MAC. Deoarece cunoaştem primele caractere din adresele dispozitivelor de interes, vom decodifica doar pachetele publicate de astfel de servere. În acest pas s-a folosit clasa ScanFilter pentru a crea un obiect “filter” în care se precizează adresele MAC căutate (“00:A0:50:B5:2A:92”, “00:A0:50:B8:F9:9D” ..). Ulterior, va fi folosit atunci când se apelează metoda startScan(), corespunzătoare începerii procesului de scanare, făcând astfel posibilă detectarea dispozitivelor de interes. Procesul de scanare şi cel de oprire a scanării se realizaeză conform apelurilor de mai jos:

```
bluetoothLeScanner.startScan(Collections.singletonList(filter), settings, leScanCallback);
bluetoothLeScanner.stopScan(leScanCallback);
```

Pentru preluarea pachetelor de date publicate de dispozitivul BLE, se foloseşte obiectul “result” transmis ca parametru în metoda suprascrisă onScanResult(), furnizată de clasa ScanCallback. În acest fel se pun la dispoziţie utilizatorului numele dispozitivului, adresa MAC a acestuia, puterea semnalului recepţionat (rssi) şi datele măsurate de senzor. Şirul de bytes preluat se va transforma în hexazecimal, iar caracterele corespunzătoare informaţiilor despre calitatea aerului sunt extrase.

Pentru calcularea valorilor corespunzătoare celor şase paramatrii, este nevoie ca şirul de numere în hexazecimal să fie convertit în binar, şi abia apoi în zecimal pentru a putea fi interpretate de utilizator. Acest proces se vede în figura 3.19, unde este prezentată împărţirea în octeţi a datelor. În cazul temperaturii, primul bit al octetului este reprezentativ pentru semnul valorii, adică determină dacă este pozitivă (bitul = 1) sau

negativă (bitul = 0). Pentru presiunea atmosferică cei mai semnificativi biți se află în cadrul octetului dedicat parametrului IAQ, de aceea trebuie avută grijă atunci când sunt preluate datele. Toate datele sunt reprezentate prin numere întregi, având unități de măsură diferite.

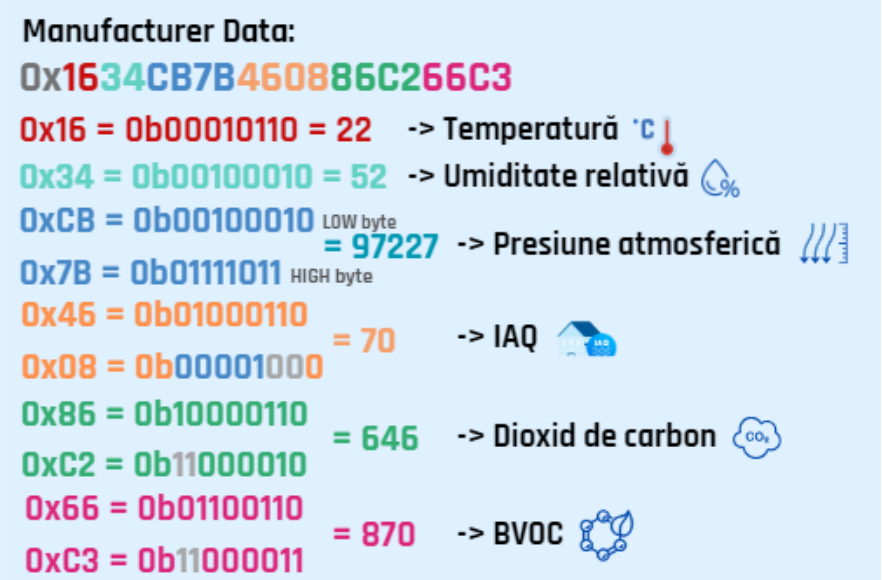


Figura 3. 19 Decodificarea pachetului de date

În program se va crea o instanță a clasei AdvertisedData:

```
processedData = new AdvertisedData(scanHex);
```

Clasa AdvertisedData are șase atribute, corespunzătoare parametrilor măsurați, iar constructorul acesteia primește ca parametru șirul de bytes transformat în sistemul de numerație hexazecimal. Valorile pentru fiecare caracteristică sunt calculate folosind metode moștenite din superclasa DataProcessing.

```
public class AdvertisedData extends DataProcessing{

  int temperature;
  int humidity;
  int pressure;
  int IAQ;
  int CO2;
  int BVOC;
  public AdvertisedData(){}
  public AdvertisedData(String scanRecordHex){
    temperature = temperature(scanRecordHex);
    humidity = humidity(scanRecordHex);
    pressure = pressure(scanRecordHex);
    IAQ = IAQ(scanRecordHex);
    CO2 = CO2(scanRecordHex);
    BVOC = BVOC(scanRecordHex);
  }
}
```

Un exemplu de cod utilizat pentru procurarea valorilor unui element din şirul de valori hexazecimale este prezentat mai jos:

```
public class DataProcessing {
 // Metodă pentru extragerea valorii presiunii atmosferice
 public int pressure(String data){

      char[] data_char = data.toCharArray();
       // String din caracterele care ne intereseaza asezate in ordinea corecta a byte-ilor
      String press_binary = hexadecimalToBinary(String.valueOf(data_char[10]) +
data_char[11] + data_char[6] + data_char[7] + data_char[4] + data_char[5]);  //
transformare in binar
      // Eliminarea biţilor corespunzători parametrului IAQ
      String p1= press_binary.substring(24-press_binary.length()+1,6);
      String p2 = press_binary.substring(press_binary.length()-16);
      press_binary = p1.concat(p2);
      return Integer.parseInt(press_binary,2);
    }

   // Metodă pentru transformarea din hexazecimal în binar
   public String hexadecimalToBinary(String dataHex) {
      int integer = Integer.parseInt(dataHex, 16);
      return Integer.toBinaryString(integer);
   }
}
```

Instanţa conformă cu clasa AdvertisedData se crează de fiecare dată când este trecut filtrul pentru adresa MAC a dispozitivelor şi se extrage pachetul cu informaţiile relevante pentru calitatea aerului. Acestea vor fi scrise într-un container de timp HashMap, având cheie timpul la care este preluată informaţia de la beacon. Astfel, se va transmite un singur obiect spre activitatea locaţiei, unde vor fi salvate datele în baza de date online. Trimiterea containerului spre cealaltă activitate este realizată printr-o structură de date de tip Intent. Se ataşează datele suplimentare la un Intent folosind metoda putExtra(), utilizând o cheie string:

```
intent.putExtra("Pressure", processedData.pressure);
```

Comunicările se vor salva în baza de date doar la finalul intervalului de scanare, deci dacă acţiunea este întreruptă, valorile preluate până în acel moment se vor pierde. S-a utilizat o bază de date în timp real furnizată de către platforma Firebase, permiţând testarea, dezvoltarea şi gestionarea aplicaţiei Android. Există puţine servere asemănătoare cu această platformă, cum ar fi: Amazon Web Services Mobile Hub, CloudKit dezvoltat de Apple sau Perse Server dezvoltat de Facebook. Firebase livrează o bază de date NoSQL, în care se stochează elementele sub formă de obiecte JavaScript Object Notation (JSON). Din acest lucru se înţelege că nu se inserează, şterg, adaugă sau actualizează datele prin interogări.

Un serviciu furnizat de Firebase este baza de date în timp real, bazată pe cloud, care permite sincronizarea şi partajarea datelor în timp real între clienţi. În Android, pentru a putea insera valori în cloud, este nevoie să se obţină o referinţă către baza de date:

```
databaseReference = FirebaseDatabase.getInstance().getReference("Location_firebase");
```

Pentru a salva datele s-a folosit metoda setValue() prin care s-a setat valoarea unei referinţe, iar pentru gestionarea evenimetelor s-a utilizat metoda addValueEventListener(). Prin a doua metodă se pot vedea în timp real modificările efectuate în cadrul bazei de date, astfel devenind mai uşor de controlat fiecare acţiune a clientului. În figura 3.20 se poate observa forma în care sunt salvate informaţiile în baza de date sub formă de obiecte JSON.

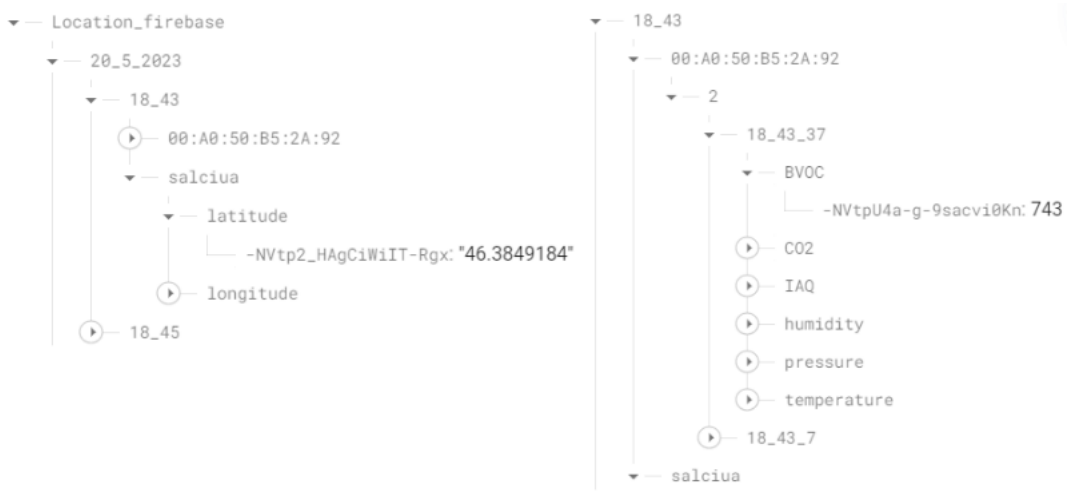


Figura 3. 20 Structura din baza de date

Folosind opţiunea Configure Image Assert din meniul "File" al mediului de dezvoltare Android Studio, se poate alege personalizarea mărcii vizuale a aplicaţiei. Iconul este afişat pe ecranul de start, în meniul aplicaţiilor sau în alte locuri în care utilizatorii interacţionează cu sistemul. În figura 3.21 se poate vedea imaginea realizată pentru acestă aplicaţie, astfel încât să fie sugestivă persoanei care foloseşte programul.

Figura 3. 21 Iconul aplicaţiei

După intrarea în aplicaţie, utilizatorul este întâmpinat cu o imagine prin care este prezentat contextul programului. Această imagine se mai numeşte şi "splash screen" şi determină prima impresie a clientului, reuşind să capteze atenţia şi să capteze interesul în a explora aplicaţia. Această imagine este creată într-o activitate separată din care este

lansată fereastra principală după depăşirea intervalului de 1500 de milisecunde. Deschiderea activităţii principale se realizează prin crearea unei instanţe a clasei Intent, iar programarea executării acestei părţi de cod este posibilă folosind motoda postDelayed() a clasei Handler.

Figura 3. 22 Splash screen

Din fereastra MainActivity, utilizatorul alege un interval de timp pentru care să se realizeze scanarea, sau altfel spus, pentru ce perioadă doreşte să monitorizeze starea aerului din locul în care se află. În acest sens, sunt disponibile 3 pictograme corespunzătoare intervalului de 30 de minute, 10 minute sau scanării singulare. Clientul are şi posibilitatea de a alege propriul interval de timp scris în ore sau în minute, astfel se simplifică procesul de monitorizare, deoarece se poate modela după nevoile de moment ale persoanei ce foloseşte aplicaţia.

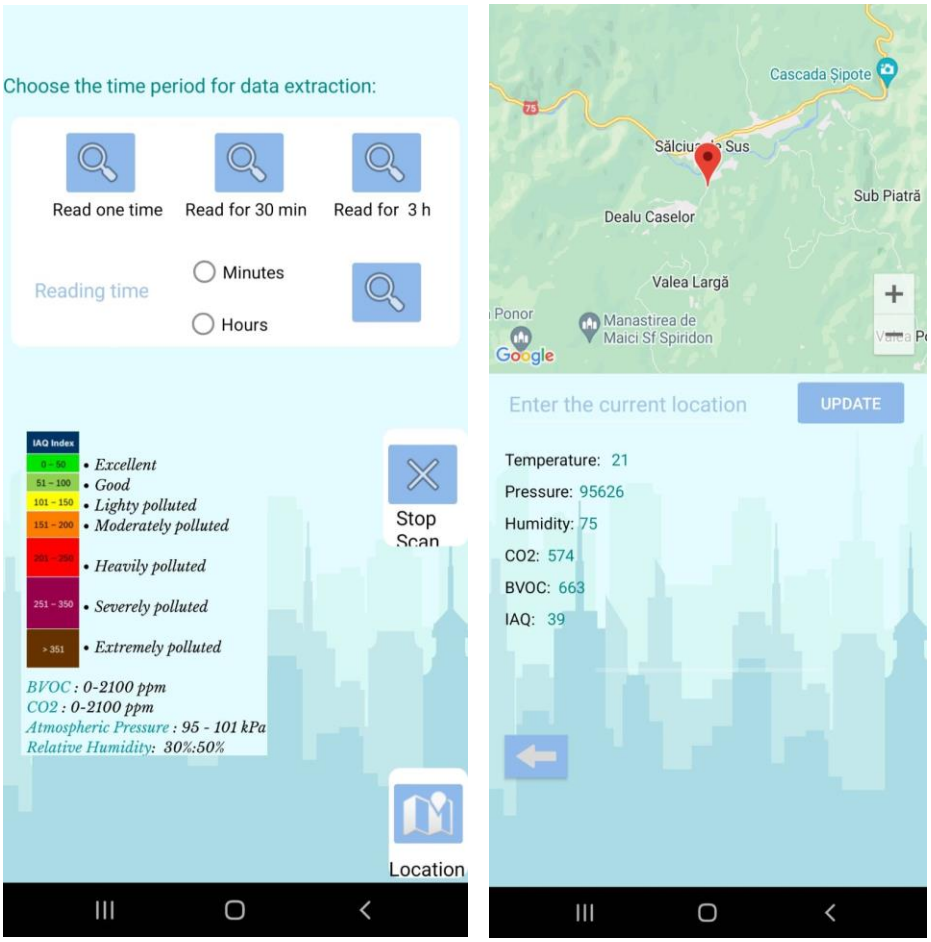
În cazul în care s-a atins din greşeală o opţiune greşită, perioada de timp aleasă se poate şterge şi se iniţiază una nouă după apăsarea imaginii care opreşte scanarea. De asemenea, în colţul din dreapta jos a ecranului, este vizibilă o căsuţă din care se poate naviga în fereastra locaţiei, fără să fie nevoie de pornirea scanării în prealabil. Utilizatorul are ocazia să vizualizeze valorile normale ale parametrilor de interes, astfel reuşind să îşi contureze o idee despre datele pe care le primeşte de la beacon. În figura 3.23-a se poate vedea fereastra principală a aplicaţiei, care face posibilă iniţierea scanării.

Un alt obiectiv al programului este de a arăta pe hartă locaţia curentă a telefonului de pe care se realizează scanarea. Aceste informaţii vor fi salvate în aceeaşi bază de date, alături de valorile citite de la senzor. Fereastra locaţiei se poate deschide folosind butonul din MainActivity sau apare automat după ce în urma scanării este găsit primul dispozitiv cu adresa MAC cerută. Navigarea dintr-o activitate în alta se face prin construirea unei instanţe de tip Intent.

Alături de fragmentul în care se vizualizează locaţia pe hartă, activitatea actuală conţine o pictogramă pin care se poate reveni în MainActivity, un câmp în care se

introduce de către utilizator numele adresei în care se află în acel moment. După apăsarea butonului aflat în centrul ferestrei, se salvează în baza de date numele introdus în câmpul menţionat anterior şi latitudinea şi longitudinea preluate de către program folosind Google Service Location API. În acest fel, vom avea acces la funcţionalităţi avansate pentru obţinerea locaţiei împreună cu latitudine şi longitudine, dar şi pentru a beneficia de precizie. Pentru a utiliza acest API este nevoie de includerea dependenţelor necesare în fişierul Manifest şi de solicitarea permisiunilor pentru locaţia generală şi cea precisă. În figura 3.23 -b este prezentată activitatea corespunzătoare locaţiei.

În plus, se va afişa şi o listă cu temperatura, presiunea atmosferică, umiditatea relativă, echivalentul de CO2 şi BVOC şi indicele IAQ corespunzător primului pachet de date primit în urma scanării, dar şi o bară de progres în care se vede cât timp mai durează până se va încheia intervalul de scanare ales. Utilizatorul este avertizat atunci când sunt găsite valori periculoase pentru sănătate, lucru evidenţiat prin creşterea peste o limită stabilită a valorilor pentru dioxidul de carbon, compuşii organici volatili şi pentru indicele IAQ. Atunci când este detectat un astfel de caz, se va colora cu roşu numărul corespunzător parametrului periculos. În acest fel, utilizatorul poate să vadă direct pe ecranul telefonului dacă este expus la un risc şi poate să ia în acel moment măsuri pentru a diminua factorii negativi.



a. b.

Figura 3. 23 Activităţile aplicaţiei: a. MainActivity, b. Location Activity

Aplicaţia pentru monitorizarea calităţii aerului poate să ruleze pentru o durată scurtă, de doar câteva minute, sau poate să fie funcţională pe o perioadă mai lungă de timp, în funcţie de necesităţile utilizatorului. Din acest motiv este necesară funcţionarea în fundal a aplicaţiei, adică atunci când nu este în prim-plan sau când dispozitivul este în repaus.

3.2.2 Aplicaţia web

După implementarea aplicaţiei Android, există necesitatea de a crea un sistem software care să acceseze baza de date cloud, să extragă datele dorite şi să ofere consumatorului o privire de ansamblu asupra valorilor corespunzătoare celor şase parametrii măsuraţi de beacon. O astfel de soluţie a fost găsită prin dezvoltarea unei aplicaţii web utilizând editorul de cod Visual Studio Code.

Aplicaţia web este implementată astfel încât să ofere posibililor utilizatori informaţii legate de datele referitoare la caliatatea aerului dintr-o anumită dată aleasă. Clienţii au posibilitatea de a vizualiza aceste date sub forma unor grafice uşor de interpretat şi, în plus, pot observa şi poziţionarea pe hartă a locurilor în care au fost preluate valorile de la dispozitivele BLE. În figura 3.24 sunt prezentate scenariile în care funcţionează aplicaţia.

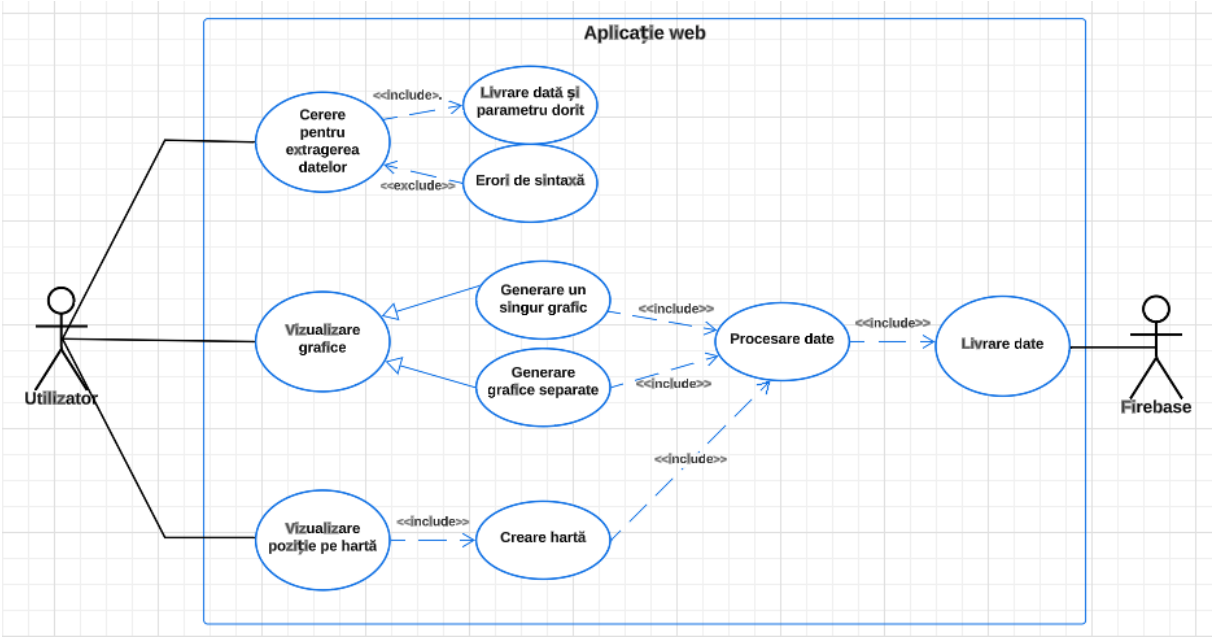


Figura 3. 24 Diagrama cazurilor pentru aplicaţia web

Aplicaţia este concepută pe baza unui server Node.js prin intermediul căruia se programează modul de afişare al paginii, logica din spatele acţiunilor clienţilor şi interacţiunea cu baza de date cloud. Serverul livrează conţinutul static către utilizatori, adică fişierele HTML, JavaScript, CSS, imaginile sau alte resurse statice care alcătuiesc interfaţa.

Primul pas pentru realizarea serverului web este crearea unui director nou în panoul de explorare sau utilizând bara de comandă, prezente în editorul de cod Visual Studio Code. Alte variante de editoare care ar putea fi folosite în acest scop sunt: Atom, Sublime Text, JetBrains WebStorm, Brackets, IntelliJ IDEA. După alcătuirea folderului, se va iniţia proiectul Node.js prin rularea comenzii “npm init” şi se instalează modulele

necesare utilizând “npm install nume-modul”. Se crează automat un fişier “package.json”, acesta conţine detaliile şi dependenţele proiectului. NPM (Node Package Manager) este un sistem de gestionare a pachetelor pentru platforma Node.js şi este folosit pentru a instala şi gestiona dependenţele şi modulele necesare. Pentru a scrie codul esenţial serverului, se proiectează un fişier JavaScript, acesta fiind punctul de intrare al aplicaţiei.

Serverul web va citi fişierele dorite din directorul specificat în prealabil şi va gestiona cererile de la client folosind protocolul de comunicare HTTP. Un client web trimite cerere HTTP către un server web pentru a solicita resursele (fişierele şi imaginile necesare implementării aplicaţiei). În cadrul aplicaţiei curente s-au folosit trei tipuri de fişiere principale: HTML (se programează structurarea conţinutului paginii, integrează alte tehnologii web şi se definesc aspectul şi stilul elementelor prezente în pagină), JavaScript (se gestionează evenimentele corespunzătoate interacţiunii clientului cu pagina web şi datele extrase din baza de date) şi CSS (se controlează aspectul şi prezentarea elementelor HTML).

Utilizatorul este întâmpinat de pagina în care se cere introducerea datei din care să fie extrase informaţiile şi, de asemenea, variabila ţintă a cărei valori o să fie afişate grafic. În figura 3.25 se poate vedea imaginea de intrare, înainte de introdecerea cererilor de către client.

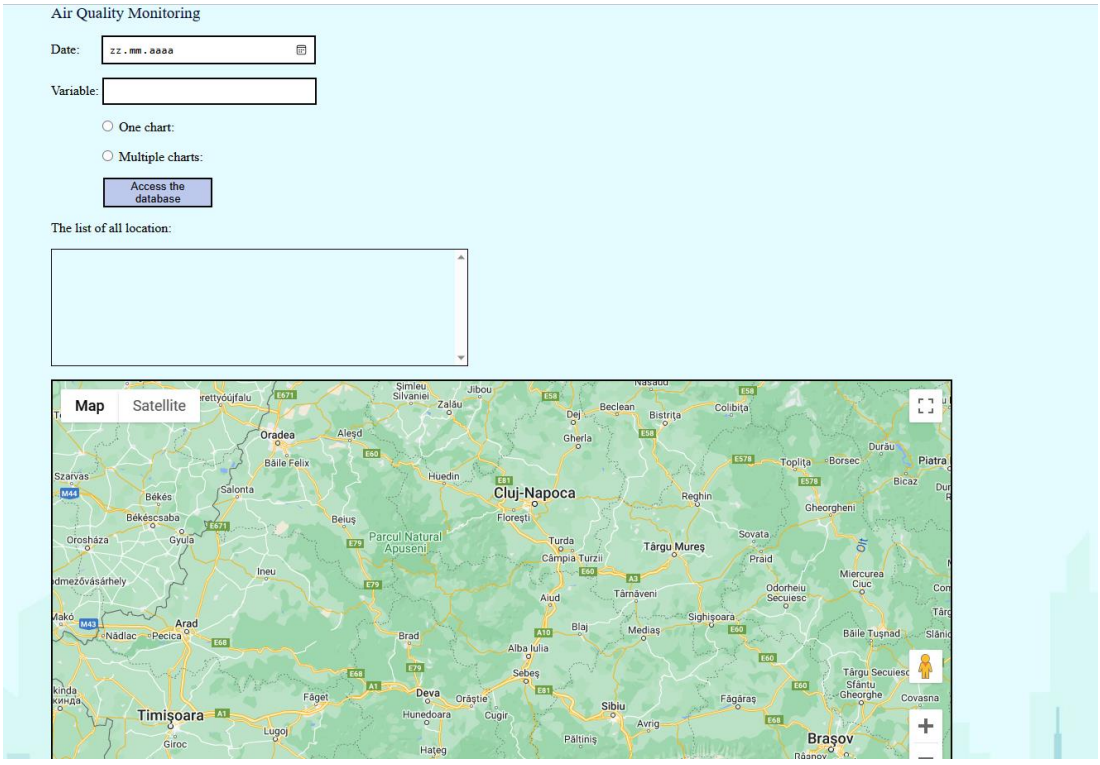


Figura 3. 25 Pagina de pornire

După introducerea datelor de către utilizator în fereastră, se va interoga baza de date, iar dacă se găsesc informaţii prelevate în ziua specificată, se vor afişa grafic. Pentru a intermedia cererile de citire din baza de date, este necesar iniţial să se importe modulele Firebase App şi Firebase Analytics. Primul pas în utilizarea Firebase este iniţierea aplicaţiei cu informaţiile de configurare specifice proiectului Firebase prin intermediul

funcţiei "initializeApp". De asemenea, trebuie folosită funcţia "getAnalytics" care oferă o instanţă a Firebase Analytics, utilizată pentru urmărirea şi analizarea evenimentelor din cadrul unei aplicaţii. Informaţiile de configurare specifice Firebase includ cheia de autentificare (apiKey), URL-ul bazei de date, domeniul de autentificare, ID-ul priectului, bucket-ul de stocare, ID-ul expeditorului de mesaje, ID-ul aplicaţiei şi ID-ul de măsurare pentru Firebase. Toate aceste informaţii se vor copia de pe site-ul Firebase din descrierea proiectului în care se află baza de date.

În figura 3.26 se poate vedea codul coerspunzător accesării şi manipulării bazei de date Firebase Realtime Database. Se obţine o referinţă către baza de date utilizând funcţia "ref", specificând nodul rădăcină sau calea dorită, astfel se pot efectua operaţii de citire, scriere, actualizare sau ştergere. Precizând referinţa către baza de date şi numele cu care este salvată, se accesează un nod specific prin folosirea funcţiei "get" în combinaţie cu "child".

```
import { initializeApp }
    from "https://www.gstatic.com/firebasejs/9.21.0/firebase-app.js";

const app = initializeApp(firebaseConfig);

import{getDatabase, ref, child, get}
    from "https://www.gstatic.com/firebasejs/9.21.0/firebase-database.js";

const database = getDatabase();
const databaseRef = ref(database);
```

Figura 3. 26 Cod folosit pentru accesarea şi manipularea bazei de date

Căutarea valorilor dorite din Firebase se realizează în funcţie de data introdusă de utilizator. Se vor extrage datele referitoare la variabila pe care acesta o alege din listă (temperatură, presiune atmosferică, umiditate relativă, echivalentu de dioxid de carbon şi de compuşi organici volatili, dar şi indicele IAQ) şi se va crea un grafic în care va fi uşor de urmărit. Graficele pot să conţină informaţii pe o durată de cel mult 24 de ore, ulterior dacă dorim să vedem date din zilele următoare vom fi nevoiţi să solicităm acest lucru explicit.

Vor exista două posibilităţi de afişare grafică: generarea graficelor separate în conformitate cu adresele MAC ale dispozitivelor (cel mult 5 grafice, pentru mai multe ar fi prea greu de urmărit) şi realizarea unui singur grafic în care să fie vizibile datele de la toate beacon-urile care au extras date în perioada de timp specificată. Această opţiune se va alege prin validarea unuia dintre cele două RadioButton-uri amplasate în partea superioară a butonului principal din fereastra iniţială. În primul caz, titlul graficului va semnifica numele adresei MAC, iar liniile, colorate conform datelor din legendă, vor fi adresele introduse de utilizator în timpul procesului de scanare. În acest fel, se vor observa diferenţele dintre locaţiile în care a fost amplasat beacon-ul în cadrul zilei alese de client. O imagine cu astfel de grafice se poate vedea în figura 3.27 pentru valorile corespunzătoare parametrului IAQ.

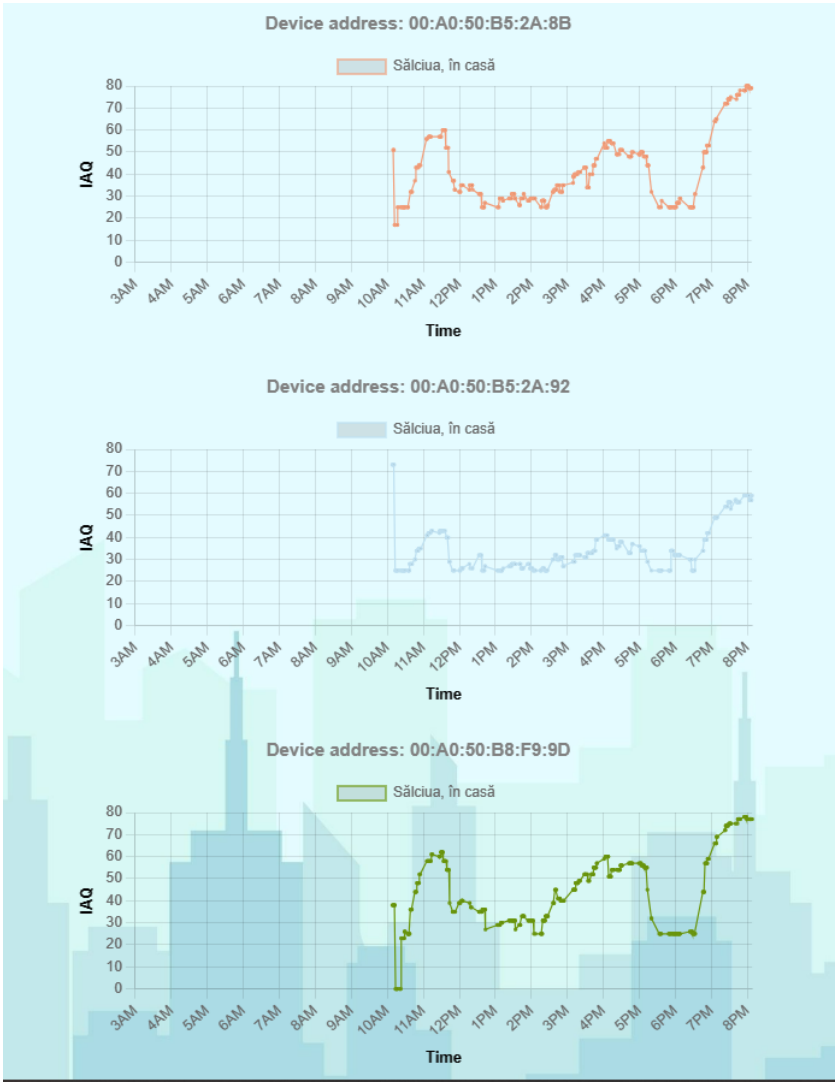


Figura 3. 27 Grafice generate pentru parametrul IAQ

Pentru cazul în care utilizatorul alege ca toate datele să apară într-un singur loc, va fi validată căsuța corespunzătoare opțiunii “Multiple charts”. Se va genera un grafic care va avea câte o linie pentru fiecare dispozitiv găsit în baza de date în perioada specificată, așa cum se poate observa și în figurile 3.28, 3.29, 3.30, 3.31, 3.32 și 3.33.

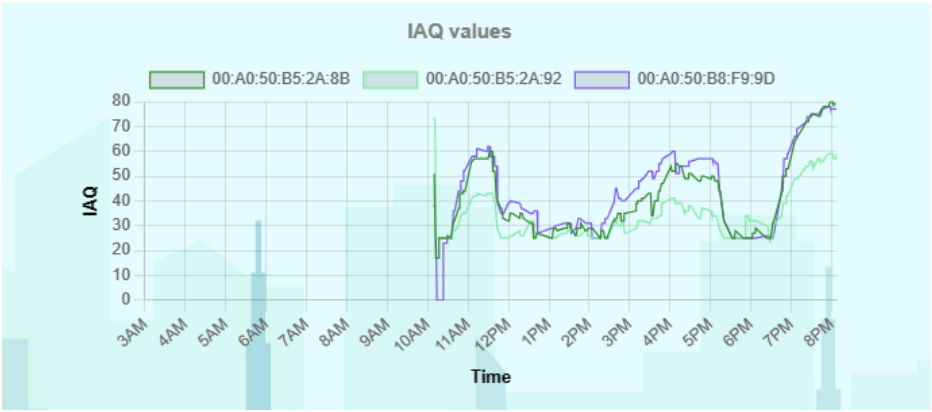


Figura 3. 28 Grafic pentru parametrul IAQ

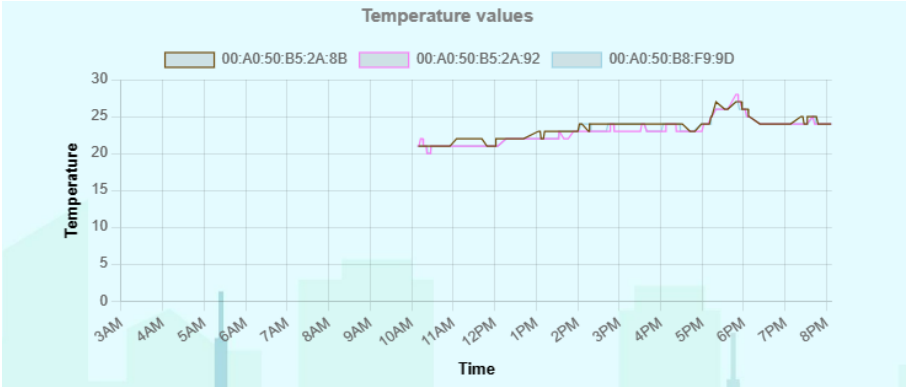


Figura 3. 29 Grafic pentru temperatură

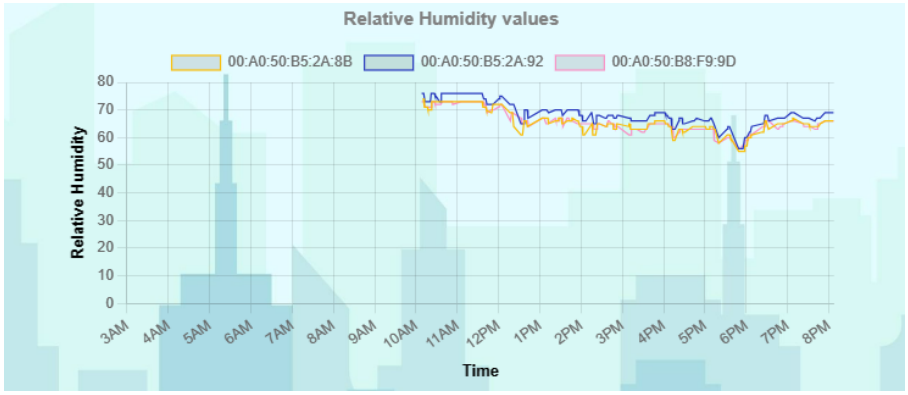


Figura 3. 30 Grafic pentru umiditatea relativă

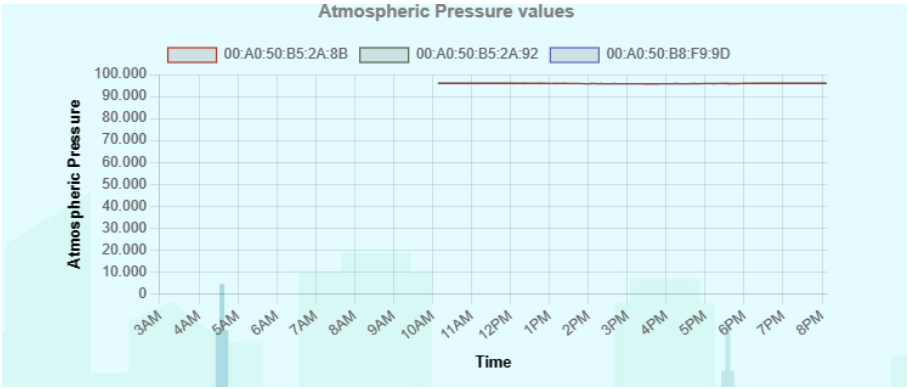


Figura 3. 31 Grafic pentru presiunea atmosferică

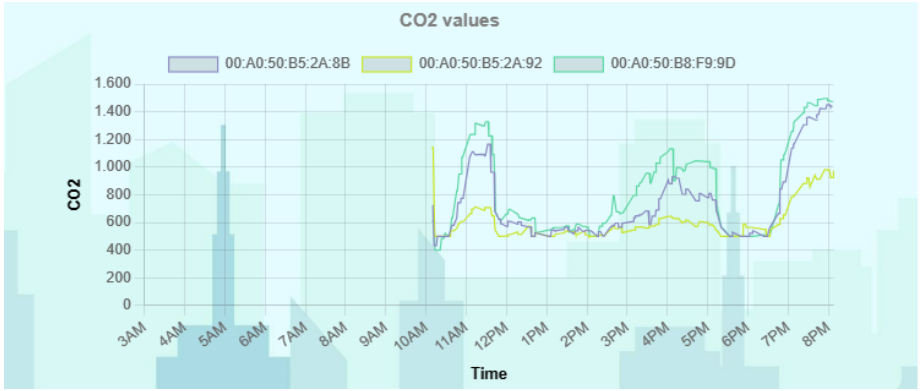


Figura 3. 32 Grafic pentru echivalentul dioxidului de carbon

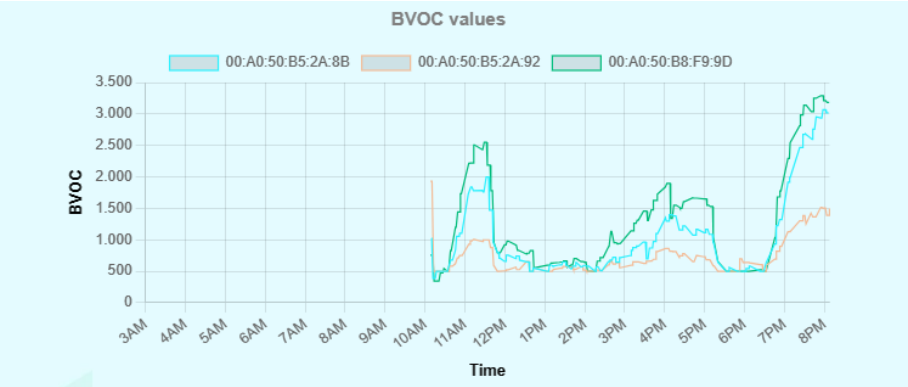


Figura 3. 33 Grafic pentru echivalentul BVOC

În cadrul bazei de date cloud sunt şi informaţii referitoare la locaţia pe care dispozitivul Android a avut-o în timpul scanării. Pentru a determina poziţionatrea pe hartă, este nevoie de coordonatele geografice, latitudine şi longitudine, dar în plus se poate găsi şi numele locaţiei introdus în timpul scanării, în fereastra corespunzătoare locaţiei. Primul pas în realizarea unei hărţi în carul unei aplicaţii web este obţinerea unei chei de autentificare utilizând Google Cloud Platform pentru a accesa Google Maps API. Ulterior, cheia prelevată va fi încărcată în proiectul ce are nevoie de aceasta şi se va iniţializa adresa cu care va fi întâmpinat clientul în momentul în care se deschide aplicaţia.

După apăsarea butonului dedicat interogării bazei de date, în lista situată în partea superioară a hărţii vor fi introduse automat numele punctelor în care s-au preluat valori de la dispozitivele BLE. Dacă clientul doreşte să vadă pe hartă unde se situează exact aceste adrese, este necesar să se selecteze o locaţie din listă, iar ulterior aceasta va apărea pe ecran. Aşa cum se poate vedea şi în figura 3.34, alături de reprezentarea pe hartă, va fi disponibilă şi adresa formată din număr, stradă, localitate şi cod poştal.

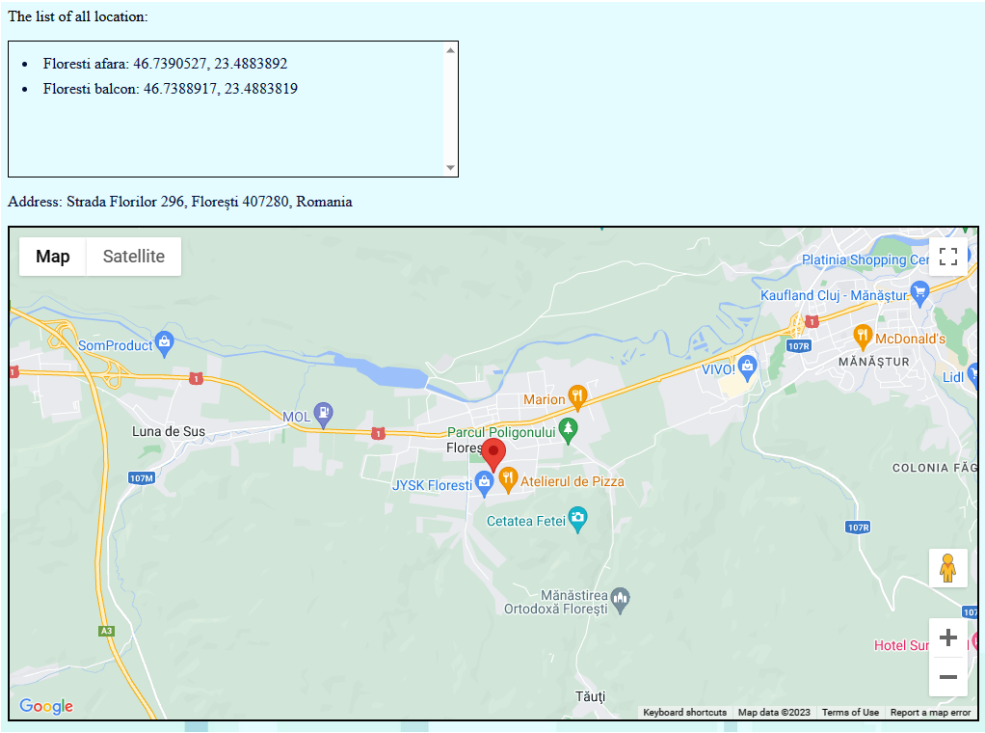


Figura 3. 34 Reprezentarea pe hartă a locaţiei utilizatorului

După generarea hărții, consumatorul are la dispoziție mai multe acțiuni pe care poate să la întreprindă, cum ar fi: schimbarea tipului mapei (vedere din satelit sau prezentare grafică a terenului, străzilor și clădirilor), apropierea sau depărtarea față de un punct ales (în cod este setată o anumită apropiere față de locația aleasă prin intermediul funcției "setZoom()"), mărirea hărții pe ecranul complet al dispozitivului sau navigarea pe străzile găsite în apropiere. De asemenea, cu ajutorul clasei "google.maps.Marker" se va crea un marker cu sprijinul căruia va fi evidențiată locația aleasă în cadrul mapei. Poziția acestuia este setată prin parametrul în care se află latitudinea și longitudinea extrasă din baza de date în funcție de preferințele persoanei care accesează aplicația web.

Fișierele utilizate pentru implementarea aplicației au fost de tipul HTML, JavaScript, CSS și imagini folosite pentru fundalul ecranului. Limbajul HTML este un standard acceptat și folosit pentru dezvoltarea aplicațiilor web, care facilitează crearea și structurarea paginilor. Din interiorul fișierului de bază de acest tip, se vor accesa pe rând modulele .js sau paginile care determină stilul componentelor aplicației. Fiecare element prezent pe ecranul acestui sistem (buton, etichete, lista locațiilor, căsuțele text sau radioButton-urile) se creează în fișierul .html. De asemenea, și inițializarea hărții sau amplasarea locațiilor în cadrul mapei sunt programate tot în interiorul fișierului creat cu acest limbaj de programare.

Pentru procesarea datelor, accesarea bazei de date cloud, crearea graficelor sau intermedierea relației cu utilizatorul sunt folosite fișierele .js scrise în limbajul de programare JavaScript. În plus, comunicarea cu serverul pentru a obține sau trimite date se poate construi cu ajutorul limbajului JS. Pentru accesarea elementelor create în HTML se va folosi metoda "document.getElementById()", iar pentru atașarea unui eveniment unei astfel de noțiuni se utilizează metoda " addEventListener()". Un exemplu de cod conceput în cadrul acestui tip de fișiere este pentru întocmirea graficelor pentru vizualizarea valorilor preluate din baza de date.

```
datasetObject.data = hourData;
datasetObject.borderColor = getColorRGBA();
datasetObject.borderWidth = borderWidthValue;
datasetObject.pointRadius =2;
datasetArray.pointStyle = 'line';
datasetArray[i] = datasetObject;
```

Aici, pentru fiecare set de date, se ceează un obiect cu proprietățile necesare unui grafic dintr-o aplicație web (valoarea reprezentată, culorile pentru liniile graficului, lățimea lor, raza punctelor pentru fiacare dată și stilul de afișare).

```
var datasetsValue = [];
datasetsValue = datasetArray;
const data = {
     labels: [new Date(insertedDate), new Date(insertedDate)],
     datasets: datasetsValue
};
```

Obiectele rezultate vor fi puse într-un șir "datasetsValue" pe baza căruia se fac graficele. Culoarea liniilor pentru fiecare adresă MAC va fi generată aleator prin funcția

getColorRGBA(). În acest mod se vor seta atât graficele în care apare o singură adresă MAC, dar şi în cel în care se pot examina rezultatele comparând valorile de la toate diszpozitivele BLE găsite în urma scanării.

Stilul şi aspectul vizual al aplicaţiei web se defineşte prin utilizarea fişierelor CSS (Cascading Style Sheets), acestea sunt conectate la pagină utilizând eticheta <link> din cadrul HTML. În cadrul etichetei se specifică calea către fişier prin "href" şi că este conectat la o resursă de tip CSS folosind rel="stylesheet". Va fi afectat aspectul elementelor din pagină conform regulilor şi declaraţiilor găsite în CSS.

3.3 Testare şi validare

3.3.1 Testarea aplicaţiei Android

Pentru procesul de testare s-au realizat procese automate şi manuale, atât pentru aplicaţia Android, cât şi pentru cea web. În ceea ce priveşte aplicaţia destinată telefoanelor mobile care folosesc sistemul de operare Android, pentru testarea automată s-a folosit tot mediul de dezvoltare integrat Android Studio.

Pentru evaluarea comportamentului secvenţelor de cod şi a metodelor din interiorul activităţilor, care nu influenţează comportamentul general al aplicaţiei, s-au scris teste unitare folosind biblioteca JUnit. JUnit este utilizată în cadrul limbajului de programare Java şi a devenit standardul în ceea ce priveşte testele unitare. Aceste procese de examinare asigură buna funcţionare a comportamentului claselor.

Primul pas în întocmirea testelor este adăugarea dependenţelor necesare în fişierul "build.gradle", urmând crearea unei clase destinate testelor unitare în secţiunea dedicată acestora din proiectul în care se realizează aplicaţia. Pentru fiecare metodă de testare se utilizează anotarea "@Test", lucru care permite executarea lor în timpul rulării testelor. Ele se crează, în mod normal, înaintea scrierii codului pentru metodele verificate, astfel primul test mereu va fi negativ deoarece nu are ce valori să compare. Testele unitare create au avut ca scop vizualizarea comportamentului programului în timpul procesării datelor primite de la dispozitivul BLE şi verificarea corectitudinii claselor "AdvertisedData" şi "LocationData" corespunzătoare modelului. În figura 3.35 este prezentat un exemplu de validare a metodelor prin care se extrag caracterele corespunzătoate temperaturii şi presiunii atmosferice şi se transformă conform regulilor furnizate de producător. Rezultatul obţinut prin rularea funcţiilor din clasa "DataProcessing" este comparat cu valoarea calculată manual de către dezvoltator. Dacă testele sunt trecute, un mesaj o să anunţe acest lucru, iar dacă acestea pică dintr-un motiv anume, se prezintă un mesaj de eroare prin care se poate vedea ce a cauzat oprirea lor.

Un alt tip de teste necesare în cadrul unei aplicaţii Android este cel al testelor de integrare, care au scopul de a verifica interacţiunea şi integrarea eficientă a unor elemente în cadrul aplicaţiei. În acest sens, s-a verificat lansarea activităţilor din cadrul proiectului (MainActivity, LocationActivity şi InitialViewActivity), dar şi prezenţa corectă a unor

secvențe text. Pentru rularea testelor de integrare s-a folosit clasa AndroidJUnit4ClassRunner pentru care a fost necesară stabilirea unor reguli.

```
3   import static org.junit.Assert.*;
4   import org.junit.Test;
5
6   public class DataProcessingTest {
7
8       @Test
9       public void testTemperature(){
10          DataProcessing dataProcessing = new DataProcessing();
11          String data = "1634CB7B460886C266C3";
12          int result = dataProcessing.temperature(data);
13          assertEquals( expected: 22, result);
14      }
15
16      @Test
17      public void testPressure(){
18          DataProcessing dataProcessing = new DataProcessing();
19          String data = "1634CB7B460886C266C3";
20          int result = dataProcessing.pressure(data);
21          assertEquals( expected: 97227, result);
22      }
```

Figura 3. 35 Teste unitare pentru verificarea metodelor din cadrul clasei DataProcessing

De asemenea, pentru verificarea interacțiunii corecte și a interogării eficiente a bazei de date Firebase s-au implementat teste instrumentale, care se încadrează tot în categoria testelor de integrare. Pentru a rula un astfel de test, s-a utilizat framework-ul de testare Android Instrumentation Test din cadrul mediului de dezvoltare Android Studio. După crearea unui proiect Firebase se poate seta modul de configurare al testelor, așa cum se poate observa în figura 3.36. Inițierea testului se face în Firebase Test Lab și se va încărca fișierul APK al aplicației și al testului în Cloud Storage Bucket.

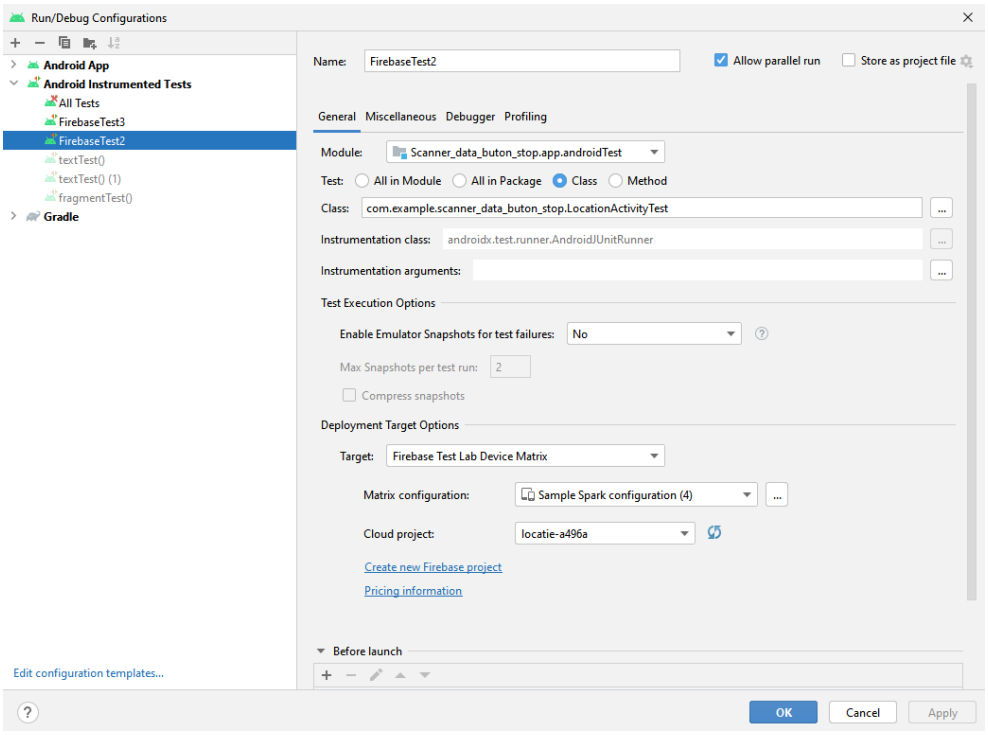


Figura 3. 36 Configurarea testelor

Testele se pot realiza atât pe un dispozitiv virtual, cât şi pe unul real, în funcţie de preferimţele testerului. Pentru aplicaţia curenta s-a ales un dispozitiv virtual Android 13, API Level 33, iar pe baza lui s-a monitorizat performanţa dispozitivului şi s-a înregistrat durata testului, 4 secunde. În figura 3.3 se pot vedea mesajele din timpul rulării testelor, iar informaţiile despre datele din Firebase se vor vedea în secţiunea Logcat a programului. În această parte vor fi furnizate date despre pachete, inclusiv versiunile protocoalelor sau ID-ul aplicaţiei, dar şi informaţii despre evenimentele care au avut loc, cum ar fi vizualizările de ecran.

```
SM-X900, Samsung | Android 13, API Level 33 (T) | Romanian | Portrait
    Starting attempt 1.
SM-X900, Samsung | Android 13, API Level 33 (T) | Romanian | Portrait
    Started logcat recording.
SM-X900, Samsung | Android 13, API Level 33 (T) | Romanian | Portrait
    Preparing device.
SM-X900, Samsung | Android 13, API Level 33 (T) | Romanian | Portrait
    Retrieving Performance Environment information from the device.
SM-X900, Samsung | Android 13, API Level 33 (T) | Romanian | Portrait
    Enabled network logging on device.
SM-X900, Samsung | Android 13, API Level 33 (T) | Romanian | Portrait
    Setting up Android test.
SM-X900, Samsung | Android 13, API Level 33 (T) | Romanian | Portrait
    Started performance monitoring.
SM-X900, Samsung | Android 13, API Level 33 (T) | Romanian | Portrait
    Starting Android test.
SM-X900, Samsung | Android 13, API Level 33 (T) | Romanian | Portrait
    Completed Android test.
SM-X900, Samsung | Android 13, API Level 33 (T) | Romanian | Portrait
    Stopped performance monitoring.
SM-X900, Samsung | Android 13, API Level 33 (T) | Romanian | Portrait
    Tearing down Android test.
SM-X900, Samsung | Android 13, API Level 33 (T) | Romanian | Portrait
    Downloading network logs from device.
SM-X900, Samsung | Android 13, API Level 33 (T) | Romanian | Portrait
    Stopped logcat recording.
SM-X900, Samsung | Android 13, API Level 33 (T) | Romanian | Portrait
    Done. Test time = 4 (secs)
SM-X900, Samsung | Android 13, API Level 33 (T) | Romanian | Portrait
    Starting results processing. Attempt: 1
```

Figura 3. 37 Secţiunea run din timpul desfăşurării testelor

Testarea manuală a ajutat la verificarea funcţionalităţii aplicaţiei pe dispozitive reale (pe emulatoare nu se poate verifica procesul de scanare a dispozitivelor BLE), prin interacţiunea utilizatorului cu interfaţa grafică a aplicaţiei. S-au folosit două telefoane Android care folosesc versiunile 11 şi 12 ale acestui sistem de operare. În acest fel se va vedea compatibilitatea aplicaţiei cu diferite dispozitive, dar se vor putea depista posibile erori specifice versiunii. S-au luat în considerare mai multe cazuri de testare:

- Deschiderea aplicaţiei fără a avea activată conexiunea Bluetooth şi locaţie.
- Încercarea de introducere a informaţiilor în baza de date fără ca dispozitivul să fie conectat la internet.
- Trecerea din activitatea principală în cea destinată locaţiei şi invers.
- Oprirea scanării dacă utilizatorul solicită acest lucru.
- Încercarea de a scana folosind butonul pentru un interval de timp ales de client, fără ca utilizatorul să introducă o valoare în câmpul dedicat în acest sens sau fără să aleagă una din opţiunile “Minutes” sau “Hours”.

- Navigarea în fragmentul pentru hartă.
- Detectarea unor valori periculoase şi semnalizarea lor pe ecranul din LocationActivity.
- Funcţionarea ProgressBar-urilor din activitatea principală şi din locaţie.

În primul caz, dacă telefonul mobil nu este conectat la Bluetooth sau locaţie, pe ecranul dispozitivului vor apărea mesaje prin case se va cere permisiunea de activare a acestor opţiuni, aşa cum se poate vedea în figura 3.38.

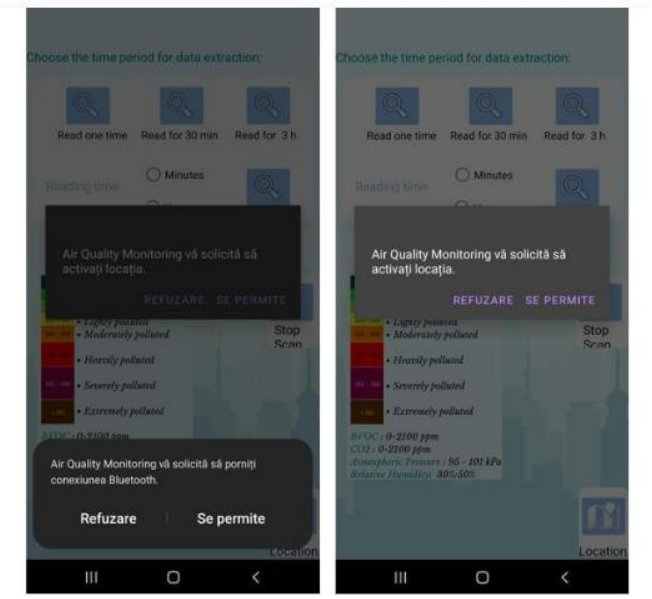


Figura 3. 38 Activarea permisiunilor în Android

Dacă utilizatorul realizează scanarea pentru beacon-uri în momentul în care nu are acces la internet, datele preluate împreună cu coordonatele geografice vor fi salvate în baza de date după ce persoana va activa conexiunea la o reţea de internet. Pentru cazul în care se apasă butonul pentru scanare într-un interval de timp ales, fără a specifica durata în câmpul “Reading time”, sau fără a preciza dacă se doreşte să se preia datele în ore sau minute, scanarea nu va avea loc. Adică, după apăsarea butonului se aşteaptă în continuare introducerea detaliilor necesare, acest lucru fiind vizibil şi prin faptul că ProgressBar-ul care face vizibil procesul de scanare nu apare pe ecran.

Aşa cum se poate vedea în figura 3.34, în fragmentul în care se construieşte harta, clientul are posibilitatea de a mări, micşora sau de a naviga în alte locuri dorite de pe hartă.

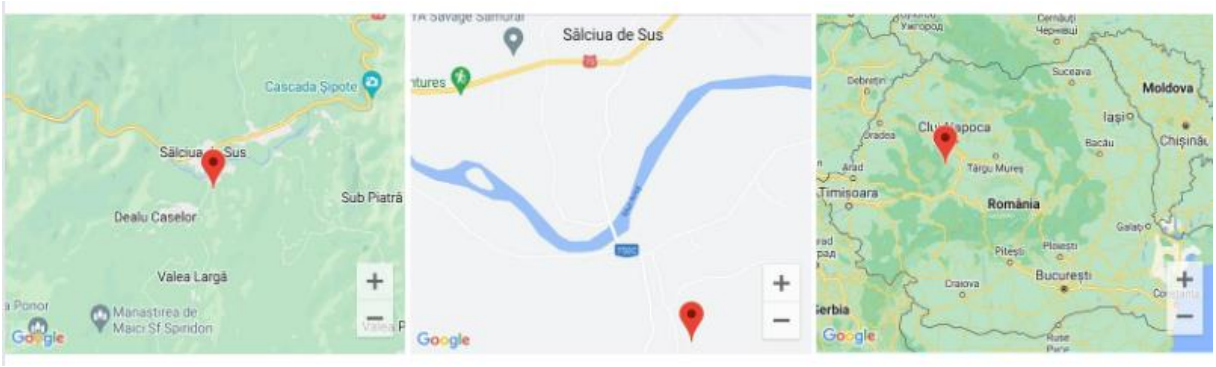


Figura 3. 39 Fragmentul pentru hartă

Detectarea unor valori prea mari ale parametrilor BVOC sau CO2 au efecte negative asupra sănătăţii persoanelor care se expun unui aer de acest fel. Atunci se detectează date periculoase se vor colora în roşu parametrii afectaţi, iar în acest fel utilizatorul poate să ia măsuri (să deschidă geamul). În figura 3.40 se poate vedea un astfel de caz.

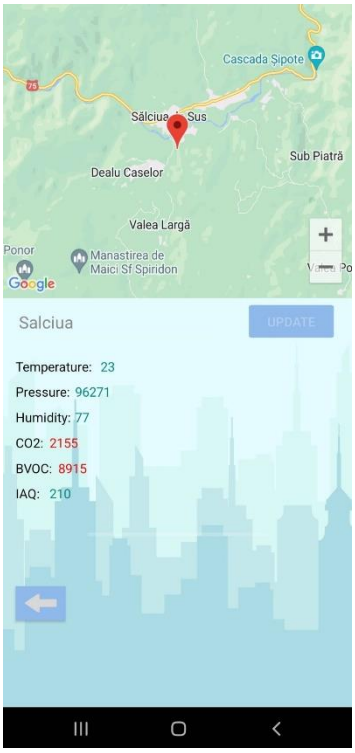


Figura 3. 40 Detectarea unor valori crescute ale BVOC şi CO2

3.3.2 Testarea aplicaţiei web

În cazul aplicaţiei web, testarea performaţei sistemului s-a realizat cu ajutorul ApacheBench (ab), o unealtă de testare a serverelor web dezvoltată de Apache Software Foundation. Se vor măsura timpii de răspuns prin generarea unui număr mare de cereri simultane de la mai mulţi utilizatori. S-au simulat diferite cazuri, pornind de la o singură cerere şi ajungând până la numărul miilor, cu diferite scenarii de cereri concurente.

Primul pas pentru îndeplinirea acestui scop este descărcarea şi instalarea Apache Benchmark, care este disponibil gratuit ca parte a pachetului Apache HTTP Server. Ulterior, se va identifica adresa URL a aplicaţiei web care urmează să fie testată şi se va asigura faptul că aceasta este accesibilă şi rulează, în cazul de faţă aceasta rulând prin intermediul serverului Node.js. Se va deschide linia de comandă şi folosind comanda “cd” se va naviga în directorul în care sunt salvate fişierele de configurare şi de binare asociate serverului web Apache. Prin folosirea comenzii “ab” se vor efectua teste asupra serverului pentru a evalua performanţa şi rezistenţa acestuia. Parametrii testului sunt setaţi prin comanda “-n” (indică numărul total de cereri care vor fi trimise serverului) şi “-c” (arată numărul de cereri concurente trimise serverului în acelaşi timp).

În figura 3.41 se poate vedea rezultatul testului executat pentru 100 de cereri, având 10 cereri realizate simultan. Vom putea vedea informaţii despre hostname-ul serverului

web la care s-au trimis cererile, portul pe care rulează, lungimea documentului în octeți, timpul necesar finalizării testelor, numărul de cereri finalizate, numărul cererilor eșuate, timpul mediu necesar pentru a procesa o singură cerere sau timpul în care se stabilește conexiunea cu serevrul.

```
C:\AB\Apache24\bin>ab -n 100 -c 10 http://localhost:5000/ > result

C:\AB\Apache24\bin>ab -n 100 -c 10 http://localhost:5000/
This is ApacheBench, Version 2.3 <$Revision: 1903618 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient).....done


Server Software:
Server Hostname:        localhost
Server Port:            5000

Document Path:          /
Document Length:        31473 bytes

Concurrency Level:      10
Time taken for tests:   0.106 seconds
Complete requests:      100
Failed requests:        0
Total transferred:      3157300 bytes
HTML transferred:       3147300 bytes
Requests per second:    947.74 [#/sec] (mean)
Time per request:       10.551 [ms] (mean)
Time per request:       1.055 [ms] (mean, across all concurrent requests)
Transfer rate:          29221.72 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.3      0       1
Processing:     5   10   3.0      9      18
Waiting:        3    9   2.8      9      16
Total:          5   10   3.0      9      18

Percentage of the requests served within a certain time (ms)
  50%      9
  66%     10
  75%     10
  80%     12
  90%     16
  95%     17
  98%     17
  99%     18
 100%     18 (longest request)
```

Figura 3. 41 Rezultatul unui test de performanță

Pentru a oferi o privire de ansamblu asupra performanței sistemului, s-au realizat 10 seturi de teste pentru cazul în care nu se execută simultan mai multe cereri, cu număr diferit de cereri totale (pornind de la 1, până la 3000). În figura 3.42 se observă rezultatele obținute în urma testarilor reprezentate grafic .

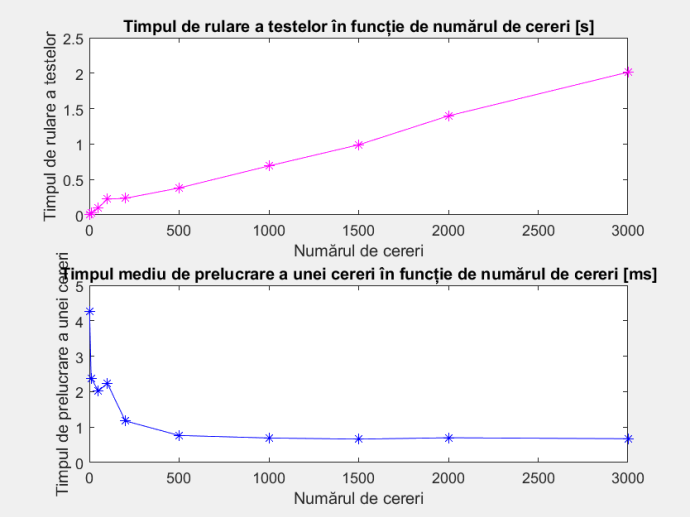


Figura 3. 42 Timpul de rulare a testelor și de executare a unei cereri pentru -n constant

Astfel se vede creşterea timpului de rulare a testului, odată cu creşterea numărului de cereri, de la 0.004 secunde la 2.014 secunde în cazul cu un număr de 3000 de request-uri. În ceea ce priveşte timpul necesar rezolvării unei cereri, acesta este invers proporţional faţă de numărul de request-uri indicat în parametrii iniţiali, lucru explicat prin faptul că serverul poate procesa cererile mai rapid deoarece există un mai mare paralelism intern în procesarea acestora, chiar dacă sunt primite secvenţial.

Alte scenarii verificate au fost cele în care s-a menţinut un număr constant de cereri trimise serverului (1000 de cereri), dar creşterea cererilor concurente, de la 1 la 100. Aşa cum se vede în figura 3.43, va cerşte atât timpul de rulare al testului (de la 1.66 secunde, la 1.378 secunde), dar şi cel de prelucrare a unei cereri (de la 0.666 milisecunde, la 551 milisecunde). Aceste rezultate se datorează latenţei şi timpului de procesare mai îndelungat.

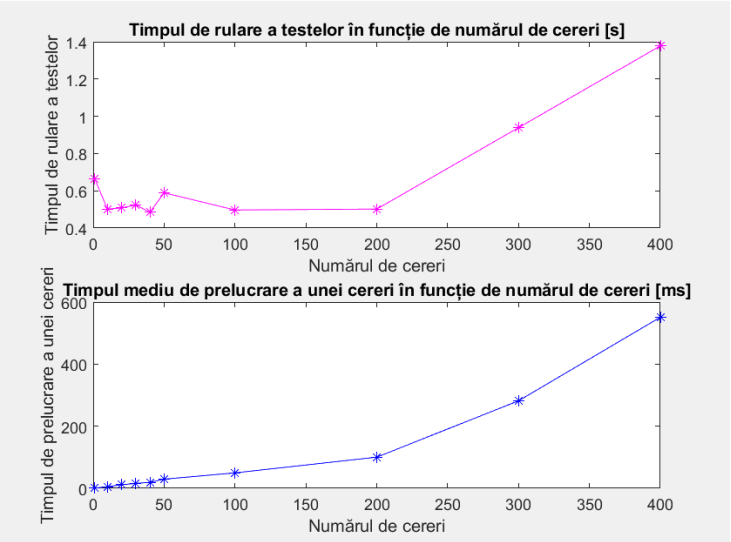


Figura 3. 43 Timpul de rulare a testelor şi de executare a unei cereri pentru -c constant

Pentru verificarea calităţii interacţiuni utilizatorului cu interfaţa web s-au realizat teste manuale. Scenariile au fost analizate folosind rularea locală a proiectului, luând în considerare următoarele cazuri:

- Încercarea de a accesa baza de date fără a avea introduşi parametrii în căsuţele destinate datei şi tipului de variabile dorită.
- Alegerea unei date din calendar în care nu s-au efectuat procese de scanare.
- Selectarea ambelor tipuri de grafice simultan.
- Selectarea pe rând a locaţiilor din lista destinată lor.
- Navigarea în cadrul hărţii spre alte locaţii dorite de utilizator.

Atunci când se apasă butonul “Access the database” fără introducerea parametrilor necesari interogării bazei de date (data din ziua în care se preiau informaţiile şi parametrul ca care să facă referire graficele), aceste două câmpuri vor fi colorate în roşu. În acest fel utilizatorul poate să înţeleagă greşeala şi să remedieze eroarea prin alegerea unor valori pentru aceşti parametrii, aşa cum se poate vedea şi în figura 3.44, unde sunt evidenţiate câmpurile ce necesită atenţie.

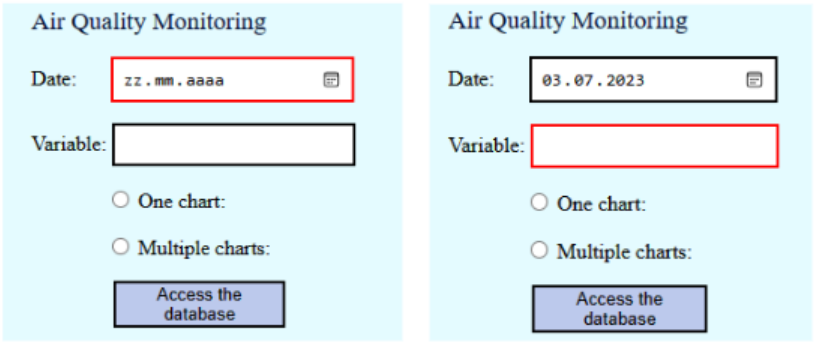


Figura 3. 44 Atenționarea utilizatorului atunci când sunt câmpuri necompletate

În cazul în care se inserează o dată din calendar în care nu sunt realizate măsurători, nu vor apărea mesaje de eroare, deoarece atenționarea în acest sens nu este o prioritate. Pentru momentul în care se aleg ambele tipuri de grafice, marcând RadioButton-urile se vor genera grafice pentru fiecare adresă MAC găsită în baza de date. În ceea ce privește evidențierea pe hartă a locațiilor alese din listă, de fiecare dată când utilizatorul selectează un nou element din listă, harta va fi actualizată conform coordinatelor corespunzătoare elementului ales. Prin acest set de teste i s-a asigurat utilizatorului un mediu ușor de manipulat, prin care să îi fie furnizate datele de care are nevoie.

4 Concluzii

4.1 Rezultate obținute

Pentru acest proiect s-a propus realizarea unui sistem de monitorizare a calității aerului folosind dispozitive Android pentru preluarea datelor prin Bluetooth Low Energy și reprezentarea rezultatelor sub formă de grafice sugestive într-o aplicație web. Soluția găsită urmează să fie folosită, atât pentru verificarea parametrilor măsurați de un senzor dedicat extragerii informațiilor referitoare la temperatură, presiune atmosferică, umiditate relativă, un echivalent al dioxindului de carbon și compușilor organici volatili, cât și pentru înregistrarea acestora într-o bază de date cloud, pe o perioadă mai lungă de timp. Astfel, vor fi disponibile atât locațiile unde s-a realizat preluarea informațiilor, data și ora achiziției, dar și grafice sugestive din care se poate vedea evoluția parametrilor monitorizați.

Există o gamă largă de dispozitive disponibile care pot prelua informații de la alte sisteme folosind tehnologii precum Bluetooth Low Energy (BLE), sau alte tipuri de comunicare. Cu toate acestea, alegerea s-a orientat către platforma Android datorită popularității sale extinse ca sistem de operare. Această decizie permite extinderea posibilității de a instala și utiliza aplicația pe un număr mai mare de dispozitive, oferind astfel un grad mai mare de accesibilitate și utilizabilitate.

În cadrul aplicației Android, s-a finalizat un mecanism prin care utilizatorul sistemului este capabil să inițieze scanarea și preluarea informațiilor de la dispozitivele BLE cu o adresă MAC compatibilă, pentru o perioadă definită de timp. Se poate observa pe ecranul telefonului, atât primul set de date prelucrate în urma scanării, cât și locația la care se află în momentul respectiv persoana. O averizare vizuală este generată dacă se detectează valori care pot influența negativ sănătatea omului. În plus, aplicația își continuă activitatea în background, dacă ecranul telefonului este oprit sau dacă sunt deschise alte aplicații.

Extragrerea pachetelor de date de la beacon se execută în timpul procesului de scanare, urmând să fie decodificate conform unor reguli știute în prealabil, furnizate de către producătorul dispozitivului. În funcție de intervalul de scanare ales, informațiile vor fi inserate într-o listă, iar la terminarea timpului alocat pentru preluare, se vor salva într-o bază de date cloud în funcție de ora la care au fost obținute. Astfel, dacă utilizatorul oprește procesul de scanare înainte de terminarea intervalului de timp, datele procesate până în acel moment se vor pierde.

Pentru aplicația web s-a implementat un sistem prin care sunt extrase informațiile de interes din baza de date cloud și sunt reprezentate grafic, alături de o hartă în care se poate vedea locul unde au fost colectate. Se pot compara valorile obținute de la mai multe dispozitive simultan, pe același grafic (în cazul în care acestea sunt situate în aceeași încăpere) sau pe grafice separate. Utilizatorul are doar permisiunea de a observa datele măsurate în cadrul aplicației web, nefiind posibilă ștergerea sau editarea acestora. Acest

aspect este menit să prevină modificarea informaţiilor, care ar putea compromite rezultatele obţinute.

Pentru crearea aplicaţiei web s-a studiat şi utilizat un server web care intermediază cererile clientului şi oferă drept răspuns fişierele disponibile din directorul proiectului. Acestea constau din fişiere dedicate stilizării aspectului elementelor aplicaţiei, surse prin care se creează componentele paginii, dar şi fişere în care s-a implementat logica din spatele fiecărei acţiuni a utilizatorului.

Contribuţiile aduse în cadrul acestui proiect sunt:

- Crearea mediului cu care interacţionează utilizatorul (activităţile din aplicaţia Android şi pagina aplicaţiei web).
- Alegerea bazei de date cloud în care să fie salvate pachetele.
- Implementarea logicii pentru decodificarea pachetelor de date.
- Tratarea şi remedierea cazurilor care produc erori de sistem.
- Crearea legăturilor necesare obţinerii resurselor externe.
- Implementarea logicii pentru afişarea locaţiei utilizatorului şi salvarea ulterioară a acesteia în baza de date.
- Interogarea bazei de date cu scopul de a furniza informaţiile necesare creării graficelor în cadrul aplicaţiei web.

4.2 Direcţii de dezvoltare

În prezent, aplicaţia oferă doar o notificare vizuală în cazul în care primul set de date primit de la dispozitivul BLE depăşeşte limitele considerate sigure pentru parametrii măsuraţi. Unul dintre obiectivele viitoare este implementarea notificărilor audio de avertizare atunci când se înregistrează valori periculoase pentru sănătatea utilizatorului. Astfel, chiar şi atunci când aplicaţia rulează în fundal sau când clientul nu este atent la ecranul telefonului, acesta va fi avertizat cu privire la posibilele pericole şi va putea lua măsuri corespunzătoare în această privinţă. Această îmbunătăţire va asigura o monitorizare mai eficientă şi o protecţie sporită a utilizatorului.

În viitor, se doreşte implementarea unei funcţionalităţi prin care să se elimine datele duplicate generate consecutiv. Acestea ocupă spaţiul de stocare, dar nu ne oferă nici un nou detaliu cu privire la calitatea aerului. Prin ştergerea acestor pachete, dacă sunt extrase la intervale scurte de timp unul după celălalt, datele vor deveni mai uşor de interpretat de către utilizator. Astfel, utilizatorii vor beneficia de informaţii mai precise şi mai utile pentru a înţelege schimbările din mediul înconjurător.

De asemenea, se pot face îmbunătăţiri şi în ceea ce priveşte aspectul aplicaţiilor Android şi web, adăugând noi opţiuni pentru utilizator. O astfel de opţiune ar putea fi vizualizarea pe ecranul telefonului a tuturor valorilor măsurate şi preluate de aplicaţie, nu doar prima pereche de date. De asemenea, un aspect important ar mai putea fi şi dezvoltarea unei logici prin care datele preluate până într-un anumit punct să poată fi

salvate în baza de date, chiar dacă nu s-a încheiat perioada de scanare. În acest mod, nu se vor pierde valori importante și nu se vor consuma resursele telefonului degeaba.

5. Abrevieri

- EEA - European Environment Agency
- BVOC – Compuşi organici volatili biogenici
- BLE – Bluetooth Low Energy
- IAQ – Indoor Air Quality
- MAC – Media Access Control
- API – Application Programming Interface
- CSS – Cascading Style Sheets
- HTML – Hyper Text Markup Language
- RFID – Radio Frequency Identification
- UUID – Universal Unique Identifier
- GATT – Generic Attribute Profile
- GAP – Generic Attribute Protocol
- ATT – Attribute Protocol
- SNMP - Simple Network Management Protocol
- PCB - Printed Circuit Board
- RFID - Radio-Frequency Identification
- WSN - Wireless Sensor Networks
- SDK - Software Development Kit
- SMP - Security Manager Protocol

6. Bibliografie

[1] K. Cromar şi N. Lazrak, "Risk communication of ambient air pollution in the WHO European Region," WHO Regional Office for Europe, Copenhagen, 2023. [Online]. Available: https://apps.who.int/iris/handle/10665/365787

[2] M. P. Sierra-Vargas, L. M. Teran, "Air pollution: Impact and prevention," în *Air Pollution and Lung Health*, Huitzilac, 2012. [Online]. Available: https://doi.org/10.1111/j.1440-1843.2012.02213.x

[3] L. Fang, G. Clausen şi P. O. Fanger, "Impact of Temperature and Humidity on the Perception of Indoor Air Quality," *Indoor Air*, vol. 8, no. 1, 2004, pp. 80-90. [Online]. Available: https://doi.org/10.1111/j.1600-0668.1998.t01-2-00003.x

[4] C. Aller, L. Ductor, and D. Grechyna, "Robust determinants of CO2 emissions," *Energy Economics*, vol. 96, Apr. 2021, pp. 105154. [Online]. Available: https://doi.org/10.1016/j.eneco.2021.105154

[5] W. M. Alberts, "Indoor air pollution: NO, NO2, CO, and CO2," *Journal of Allergy and Clinical Immunology,* vol. 94, no. 2, 1994, pp. 289-295. [Online]. Available: https://doi.org/10.1053/ai.1994.v94.a56007

[6] C. Calfapietra, E. Pallozzi, I. Lusini, and V. Velikova, "Modification of BVOC Emissions by Changes in Atmospheric [CO2] and Air Pollution," *Tree Physiology 5*, 2013, pp. 10. [Online]. Available: https://doi.org/10.1007/978-94-007-6606-8_10

[7] A.P. Jones, "Indoor air quality and health," *Atmospheric Environment*, Volume 33, Issue 28, 1999, Pages 4535-4564. [Online]. Available: https://doi.org/10.1016/S1352-2310(99)00272-1

[8] A. K. Kanál and K. Tamás, "Assessment of Indoor Air Quality of Educational Facilities using an IoT Solution for a Healthy Learning Environment," *2020 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, Dubrovnik, Croatia, 2020, pp. 1-6, [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9129231

[9] Hairong Qi, S.Sitharama Iyengar, Krishnendu Chakrabarty, "Distributed sensor networks—a review of recent research," *Journal of the Franklin Institute*, Volume 338, Issue 6, 2001,Pages 655-668. [Online]. Available: https://doi.org/10.1016/S0016-0032(01)00026-6

[10] M.A. Fetcenko, S.R. Ovshinsky, B. Reichman, K. Young, C. Fierro, J. Koch, A. Zallen, W. Mays, T. Ouchi, "Recent advances in NiMH battery technology," *Journal of Power Sources*, Volume 165, Issue 2, Pages 544-551, 2007. [Online]. Available: https://doi.org/10.1016/j.jpowsour.2006.10.036

[11] F. J. Dian, A. Yousefi and S. Lim, "A practical study on Bluetooth Low Energy (BLE) throughput," *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, Vancouver, BC, Canada, 2018, pp. 768-771. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8614763

[12] V. S. Srinivasan, Saleem, T. Kumar and D. K. Yasarapu, "Raspberry Pi and iBeacons as environmental data monitors and the potential applications in a growing BigData ecosystem," *2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, Bangalore,

India, 2016, pp. 961-965. [Online]. Available: https://ieeexplore.ieee.org/document/7807971

[13] M. Kohne and J. Sieck, "Location-Based Services with iBeacon Technology," *2014 2nd International Conference on Artificial Intelligence, Modelling and Simulation*, Madrid, Spain, 2014, pp. 315-321. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7102480

[14] D. Hasenfratz, O. Saukh, S. Sturzenegger, and L. Thiele, "Participatory Air Pollution Monitoring Using Smartphones," *Computer Engineering and Networks Laboratory*, ETH Zurich, Switzerland, 2012. [Online]. Available: https://www.cdiweb.com/datasheets/e2v/mics-oz-47.pdf

[15] B. Maag, Z. Zhou and L. Thiele, "A Survey on Sensor Calibration in Air Pollution Monitoring Deployments," in *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4857-4870, Dec. 2018. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8405565

[16] J. Zhang, Y. Wei, and F. Zhangfu, "Ozone Pollution: A Major Health Hazard Worldwide," Front. Immunol., vol. 10, p. 2518, 2019. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fimmu.2019.02518

[17] S. C. Folea and G. D. Mois, "Lessons Learned From the Development of Wireless Environmental Sensors," in *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 6, pp. 3470-3480, June 2020. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8818347

[18] P. Spachos and K. Plataniotis, "BLE Beacons in the Smart City: Applications, Challenges, and Research Opportunities," in *IEEE Internet of Things Magazine*, vol. 3, no. 1, pp. 14-18, March 2020. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9063401

[19] C. Deng *et al.*, "IEEE 802.11be Wi-Fi 7: New Challenges and Opportunities," in *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2136-2166, Fourthquarter 2020. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9152055

[20] C. M. Ramya, M. Shanmugaraj and R. Prabakaran, "Study on ZigBee technology," *2011 3rd International Conference on Electronics Computer Technology*, Kanyakumari, India, 2011, pp. 297-301. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/5942102

[21] P. Gokhale, O. Bhat, S. Bhat, "Introduction to IOT," *International Advanced Research Journal in Science, Engineering and Technology*, 2018. [Online]. Available: (PDF) Introduction to IOT (researchgate.net)

[22] L. Ma, L. Gu, J. Wang, "Research and Development of Mobile Application for Android Platform," *International Journal of Multimedia and Ubiquitous Engineering*, 2014, pp. 187-198. [Online]. Available: Journal Paper Format (gvpress.com)

[23] J. Liu and J. Yu, "Research on Development of Android Applications," *2011 4th International Conference on Intelligent Networks and Intelligent Systems*, Kuming, China, 2011, pp. 69-72. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6104696

[24] T. Lou, "A comparison of Android Native App Architecture - MVC, MVP and MVVM," 2016. [Online]. Available: https://research.tue.nl/en/studentTheses/a-comparison-of-android-native-app-architecture

[25] M. Nikodem and M. Bawiec, "Experimental Evaluation of Advertisement-Based Bluetooth Low Energy Communication," *Sensors*, vol. 20, no. 1, Jan. 2019, Art no. 107. [Online]. Available: (PDF) Experimental Evaluation of Advertisement-Based Bluetooth Low Energy Communication (researchgate.net)

[26] M. P. Choudhary, V. Garg, "Causes, Consequences and Control of Air Pollution," 2015. [Online]. Available: (PDF) Causes, Consequences and Control of Air Pollution (researchgate.net)

[27] T. V. Sântejudean, G. Dan Mois, T. Sanislav and S. C. Folea, "Edge Computing in Wireless Sensing Applications," *2022 11th Mediterranean Conference on Embedded Computing (MECO)*, Budva, Montenegro, 2022, pp. 1-4. [Online]. Available: https://ieeexplore.ieee.org/document/9797161

[28] BME688 - Datasheet. [Online]. Available: https://www.bosch-sensortec.com/products/environmental-sensors/gas-sensors/bme688/

[29] A. Ali, "NodeJs Web Server architecture," mai 2021. [Online]. Available: https://medium.com/nerd-for-tech/nodejs-web-server-architecture-a21d02a33bad

[30] An overview of HTTP. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview