

MINISTERUL EDUCAȚIEI



UNIVERSITATEA TEHNICĂ

DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

APLICAȚIE WEB PENTRU MONITORIZAREA CALITĂȚII AERULUI ÎN MEDIUL EXTERIOR

PROIECT DE DIPLOMĂ

Autor: **Cristina-Elena CAIA-HOANĂȘ**

Conducător științific: **Sl.dr.ing. Gabriel HARJA**

2023



Vizat,

DECAN

Prof. dr. ing. Liviu MICLEA

DIRECTOR DEPARTAMENT AUTOMATICĂ

Prof. dr. ing. Honoriu VĂLEAN

Autor: **Cristina-Elena CAIA-HOANĂȘ**

Aplicație web pentru monitorizarea calității aerului în mediul exterior

1. **Enunțul temei:** Dezvoltarea unei aplicații web care oferă utilizatorului posibilitatea de monitorizare a calității aerului prin evaluarea indexului acestuia în diverse locații. Serverul aplicației este dezvoltat folosind cadrul de lucru (framework-ul) Spring Boot cu baza de date Firebase Database. Aplicația dispune de autentificarea implementată folosind funcționalitățile Firebase Authentication, astfel datele utilizatorilor fiind securizate. Partea de client este implementată cu ajutorul bibliotecii JavaScript, React și alte componente care îmbunătățesc aspectul aplicației, dar și experiența utilizatorului. Aceștia vor avea la dispoziție date despre senzorii care înregistrează valori ale poluanților gazoși și a stării atmosferice. În ansamblu se va realiza o perspectivă globală asupra stării mediului înconjurător.
2. **Conținutul proiectului:** Pagina de prezentare, Declarație privind autenticitatea proiectului, Sinteza proiectului, Cuprins, Introducere, Studiu bibliografic, Analiză și proiectare, Implementare, Testare și rezultate, Concluzii, Bibliografie.
3. **Locul documentării:** Universitatea Tehnică din Cluj-Napoca
4. **Data emiterii temei:** 6 octombrie 2022
5. **Data predării:** 6 iulie 2023

Semnătura autorului

Semnătura conducătorului științific



**Declarație pe proprie răspundere privind
autenticitatea proiectului de diplomă**

Subsemnatul(a) **Cristina-Elena CAIA-HOANĂȘ**, legitimat(ă) cu CI seria NZ nr. 178689, CNP 6000226271542,

autorul lucrării: **Aplicație web pentru monitorizarea calității aerului în mediul exterior**

elaborată în vederea susținerii examenului de finalizare a studiilor de licență la **Facultatea de Automatică și Calculatoare**, specializarea **Automatică și Informatică Aplicată**, din cadrul Universității Tehnice din Cluj-Napoca, sesiunea Iulie 2023 a anului universitar 2022-2023, declar pe proprie răspundere, că această lucrare este rezultatul propriei activități intelectuale, pe baza cercetărilor mele și pe baza informațiilor obținute din surse care au fost citate, în textul lucrării, și în bibliografie.

Declar, că această lucrare nu conține porțiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislației române și a convențiilor internaționale privind drepturile de autor.

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

În cazul constatării ulterioare a unor declarații false, voi suporta sancțiunile administrative, respectiv, *anularea examenului de licență*.

Data

05.07.2023

Prenume NUME

Cristina-Elena CAIA-HOANĂȘ



SINTEZA

proiectului de diplomă cu titlul:

Aplicație web pentru monitorizarea calității aerului în mediul exterior

Autor: **Cristina-Elena CAIA-HOANĂȘ**

Conducător științific: **Sl.dr.ing. Gabriel HARJA**

1. Cerințele temei:

Dezvoltarea unei aplicații software care are ca scop principal monitorizarea calității aerului exterior și stării atmosferice din diverse zone, indicele fiind determinat de valorile poluanților gazoși măsurați în interval orar;

Proces de autentificare pentru securizarea datelor, cu posibilitatea de gestionare a contului utilizatorului;

Stocarea datelor senzorilor (cu posibilitatea de modificare, adăugare și ștergere), cu reprezentarea lor pe o harta Google Maps;

Înregistrări cu măsurătorile senzorilor și diagrame cu acestea pentru ultimele ore. Mesaje de sănătate în funcție de valoarea indicelui și valorile recomandate;

Dimensiuni ale componentelor adaptive pentru folosirea aplicației pe diferite dispozitive.

2. Soluții alese:

Implementarea aplicației în Spring Boot pentru partea de server, structurat în 3 componente: Controller, Service și Firebase;

Partea de client este dezvoltată cu ajutorul React JS, biblioteca principală pentru stilizarea componentelor fiind MUI-Material;

Manipularea datelor este efectuată în baza de date Firebase Database. Procesul de securizare a datelor, autentificarea utilizatorilor și validarea acestora este efectuat în consola Firebase Authentication.



3. Rezultate obținute:

Dezvoltarea cu succes a aplicației de tip web care monitorizează indicele de calitate a aerului, stochează datele senzorilor și valorile măsurărilor acestora în tabele și diagrame, gestionează procesul de autentificare a utilizatorilor.

Funcționalitățile aplicației oferă utilizatorilor o navigare plăcută prin implementarea diferitelor scurtături: căutarea locațiilor pe hartă, meniul de navigare pentru redirecționarea pe paginile dorite și dimensiunile ecranelor adaptive.

4. Testări și verificări:

Funcționalitatea codului scris pe partea de server s-a testat constant pe parcursul dezvoltării folosind aplicația Postman. De aici s-au trimis cereri de tip GET, POST, PUT sau DELETE către backend.

Testarea manuala s-a realizat pentru componentele implementate în partea de frontend. S-au testat diferite scenarii prin introducerea de date incorecte sau manipularea eronată a componentelor pentru a se asigura că dezvoltarea este corectă și aplicația răspunde cu succes cerințelor propuse.

5. Contribuții personale:

Studiul actual asupra efectelor negative privind aerul exterior expirat și metodelor de calculare a indicelui de calitate a acestuia;

Dezvoltarea unui sistem flexibil, optim și ușor de utilizat care redă indicele de calitate cu scopul de a influența utilizatorii să conștientizeze importanța aerului din mediul în care trăiesc și efectele dăunătoare în cazul expunerii la aerul poluant.

6. Surse de documentare:

Lucrări științifice, articole și cărți care tratează tema poluării mediului cu mare interes, logică de funcționalitate implementată în aplicații similare și specificații pentru dezvoltarea codului în tehnologiile utilizate.

Semnătura autorului

Semnătura conducătorului științific

Cuprins

1	INTRODUCERE	3
1.1	CONTEXT GENERAL.....	3
1.2	OBJECTIVE.....	4
1.2.1	<i>Obiectivul principal</i>	<i>4</i>
1.2.2	<i>Obiective secundare</i>	<i>4</i>
1.2.3	<i>Obiective funcționale.....</i>	<i>4</i>
1.2.4	<i>Obiective non-funcționale</i>	<i>6</i>
1.3	SPECIFICAȚII.....	7
2	STUDIU BIBLIOGRAFIC	9
3	ANALIZĂ ȘI PROIECTARE	16
3.1	METODA DE CALCUL PENTRU INDICELE DE CALITATE A AERULUI.....	16
3.2	ARHITECTURA SOFTWARE.....	17
3.2.1	<i>Structura aplicației.....</i>	<i>17</i>
3.2.2	<i>MCV.....</i>	<i>18</i>
3.3	CONCEPTE TEORETICE	19
3.3.1	<i>Baza de date</i>	<i>19</i>
3.3.2	<i>Partea server.....</i>	<i>19</i>
3.3.3	<i>Partea client.....</i>	<i>19</i>
3.3.4	<i>REST.....</i>	<i>20</i>
3.4	RESURSE SOFTWARE.....	20
3.4.1	<i>Firebase.....</i>	<i>20</i>
3.4.2	<i>Spring Boot.....</i>	<i>21</i>
3.4.3	<i>React JS.....</i>	<i>24</i>
3.4.1.1	<i>React Google Maps API</i>	<i>25</i>
3.4.1.2	<i>Material UI și Material-Table</i>	<i>26</i>
3.4.1.3	<i>Formik</i>	<i>26</i>
3.4.1.4	<i>Highcharts React.....</i>	<i>27</i>
3.4.1.5	<i>Reach UI</i>	<i>28</i>
3.4.4	<i>Firebase Authentication</i>	<i>28</i>
4	IMPLEMENTARE	30
4.1	CONFIGURAREA FIREBASE ȘI SPRING BOOT	30
4.1.1	<i>Crearea proiect Firebase.....</i>	<i>30</i>
4.1.2	<i>Crearea proiect Spring Boot</i>	<i>30</i>
4.1.3	<i>Cheia privata Firebase.....</i>	<i>31</i>
4.1.4	<i>Conexiunea Spring Boot la Firebase Database</i>	<i>31</i>
4.2	STRUCTURA BAZEI DE DATE FIRESTORE DATABASE.....	32
4.3	STRUCTURA SPRING BOOT	33
4.3.1	<i>Spring Boot Service</i>	<i>33</i>
4.3.2	<i>Securizarea datelor Spring Boot cu Cloud Firebase</i>	<i>38</i>
4.3.3	<i>Controller.....</i>	<i>39</i>
4.3.4	<i>Utilizatorii</i>	<i>39</i>
4.3.5	<i>CORS.....</i>	<i>40</i>
4.3.6	<i>Generarea de senzor și măsurători.....</i>	<i>40</i>
4.4	STRUCTURA REACT JS.....	41

4.4.1	AXIOS.....	41
4.4.2	Firestore Authentication cu React JS și utilizatorii în Firestore Database	42
4.4.3	Harta Google Maps cu senzorii și câmpul de căutare pentru locații	43
4.4.4	Indicele de calitate a aerului și măsurătorile	44
4.4.5	Sidebar – meniul din lateralul paginii și dimensiunea paginilor adaptivă	45
4.4.6	Tabelul senzorilor și modificarea datelor acestora. Adăugarea unui senzor și ștergerea	45
4.4.7	Gestionarea rutelor	45
5	TESTARE ȘI REZULTATE	47
5.1	MOD DE UTILIZARE	47
5.1.1	Set-up backend și frontend.....	47
5.2	TESTARE	47
5.2.1	Testare Postman pentru serverul Spring Boot	47
5.2.2	Testare manuală pentru funcționalitățile aplicației; Rezultate	48
5.2.2.1	Testare înregistrare, logare și recuperare cont	48
6	CONCLUZII.....	53
6.1	OBIECTIVE PROPUSE.....	53
6.2	DIRECȚII DE DEZVOLTARE.....	54
7	BIBLIOGRAFIE.....	55

1 Introducere

1.1 Context general

În conformitate cu EEA (European Environment Agency) [1], în anul 2020, 96% din locuitorii urbani au fost expuși la concentrații îngrijorător de ridicate a poluării aerului exterior. [2]

Principalele particule care influențează calitatea aerului într-un mod negativ sunt particulele în suspensie, PM_{2.5} și PM₁₀ (particule cu diametrul aerodinamic mai mic decât 2.5, respectiv 10 $\mu\text{g}/\text{m}^3$ - microni pe metru cub), O₃ (ozonul), NO₂ (dioxidul de azot) și SO₂ (dioxidul de sulf). Sursele protagoniste contaminării aerului includ generarea de energie folosind centrale electrice pe bază de cărbune, emisiile vehiculelor, fumul de la producția chimică etc. [3]

Poluarea aerului din zonele naturale, rurale și urbane devine o preocupare majoră pentru cercetători, iar Organizația Mondială a Sănătății (WHO) tratează acest subiect cu mare interes deoarece acest fenomen are efecte considerabile asupra modului de viață.

Expunerea populației la aerul ambiental poluat cauzează, la nivel global, aproximativ 4.2 milioane de decese anual. Consecințele poluării aerului sunt manifestate în general prin afecțiuni cardiovasculare, cancer pulmonar și afecțiuni respiratorii acute.

Studiile EPA (United States Environmental Protection Agency) confirmă că poluarea aerului are de asemenea un impact dăunător asupra climei și ecosistemului la scară mondială. Ozonul (O₃) din atmosfera încălzește clima, în timp ce PM conține carbon negru fiind cunoscut unul dintre cei mai mari contribuitori la încălzirea globală.

Monitorizarea și informarea populației asupra calității aerului exterior este soluția propusă în această lucrare, reprezentând un prim pas esențial în a ajuta oamenii să perceapă efectele adverse ale poluării aerului și să fie conștienți de importanța și valoarea aerului curat pe care l-ar putea consuma. Scopul de lungă durată este de a determina oamenii să fie implicați în combaterea poluării pentru a fi eliminată, prin mici schimbări adăugate în obiceiurile zilnice.

Sistemul rezultat furnizează cele mai recente valori ale poluanților și calculează un indice de calitate a aerului [4], aceste informații fiind afișate pe interfața grafică a aplicației web, disponibilă utilizatorilor.

1.2 Obiective

1.2.1 Obiectivul principal

Obiectivul primar al lucrării este dezvoltarea unei platforme software sub formă de aplicație web în care se monitorizează indicele de calitate a aerului determinat din datele poluanților PM10, PM2.5, SO2, NO2 și O3 înregistrate la fiecare interval de o ora. Aceste date sunt dispuse ca rezultat a măsurătorilor senzorilor plasați pe o harta interactiva Google Maps. Senzorii pot fi vizualizați sub formă de marcarea pe harta, dar și într-un tabel. Adăugarea unui senzor, modificarea datelor sau ștergerea lui este de asemenea una dintre opțiunile disponibile.

Având la dispoziție indicele de calitate, tabel de referința cu limitele impuse de Organizația Mondială a Sănătății (WHO) și un mesaj de informare în funcție de valoare, populația își va crea o imagine de ansamblu a aerului respirat și în conformitate cu rezultatul obținut, ea va fi motivată să ia decizii pentru îmbunătățire și să acționeze în consecință.

1.2.2 Obiective secundare

Un obiectiv secundar este ca utilizatorii să dispună și de starea atmosferei: temperatură, umiditate și presiune atmosferică, aceasta fiind accesibilă în locațiile unde există un senzor activ. Pentru o experiență cat mai plăcută și eficientă a identificării senzorilor pe hartă, utilizatorul dispune de un câmp de căutare în care poate introduce locațiile reale cat mai exacte, astfel încât să îi fie vizibilă zona dorită într-un timp foarte rapid.

Pentru o vizualizare mai clară a ultimelor 12 ore de înregistrare a datelor poluanților, următorul scop secundar al lucrării este de a reprezenta măsurătorile obținute într-un mod clar și coerent cu ajutorul unor grafice. O simplă reprezentare vizuală va contribui la identificarea tendințelor și relațiilor între date într-un mod mai intuitiv.

Aplicația mai dispune de înregistrarea persoanelor dornice de a utiliza aplicația prin crearea unui nou cont de utilizator, proces de resetare a parolei în caz de uitare a acesteia și autentificare cu adresa de email. Userul autentificat poate mai apoi să își gestioneze datele contului prin modificarea anumitor câmpuri.

1.2.3 Obiective funcționale

În contextul aplicațiilor web, obiectivele funcționale reprezintă scopurile sau funcțiile esențiale pe care aplicația trebuie să le realizeze pentru a-și atinge scopul și pentru a satisface nevoile utilizatorilor. Aceste obiective definesc comportamentul aplicației și au un rol important în orientarea procesului de proiectare și implementare.

Obiectivele funcționale care se doresc a fi implementate în cadrul aplicației sunt prezentate în Tabelul 1.1.

Tabel 1.1 Obiectivele funcționale ale aplicației

Descriere obiectiv funcțional
Înregistrare utilizator nou
Autentificare pe bază de roluri: admin și user
Conectare utilizând un cont existent
Resetare parola în caz de uitare a acesteia
Redirecționarea pe pagina de înregistrare, conectare sau resetare parola în cazul în care utilizatorul nu este conectat
Gestionarea contului
Vizualizare harta Google Maps
Oferirea unei navigări rapide și eficiente prin intermediul meniului adaptiv diferitelor dimensiuni de ecrane din lateralul paginii
Câmp de căutare a unei locații cât mai exacte pentru redirecționarea utilizatorului în acel loc pe harta și vizualizarea senzorilor din acea zona
Vizualizare senzorii pe hartă
Indicele de calitate a aerului și data actualizării
Vizualizare stării calității aerului exterior prin generarea unei culori care corespunde unor limite impuse de autorități
Legenda explicativă pentru fiecare culoare redata a indicelui măsurat
Valorile poluanților și starea atmosferei cu ultima data de actualizare
Tabel cu limitele poluanților împreună cu un mesaj de sănătate corespunzător intervalelor
Explicația metodei de calculare a indicelui de calitate a aerului
Grafice pentru ultimele 12 ore cu valorile înregistrate pentru poluanți și starea atmosferei
Tabel cu senzorii din aplicație cu posibilitatea de ordonare a lor după nume sau data plasării lor în locație, paginare și câmp de căutare
Funcție de modificare a datelor unui senzor
Posibilitate de ștergere a senzorilor
Adăugarea unui senzor, cu posibilitatea de a defini tipul de măsurători
Funcționalitate de deconectare din cont
Dimensiuni ale ecranului adaptive pentru o utilizare a aplicației și pe alte dispozitive electronice: telefoane mobile, tablete.

1.2.4 Obiective non-funcționale

Obiectivele non-funcționale reprezintă aspecte legate de experiența utilizatorului, scalabilitatea, securitatea și performanța aplicației. Acestea asigură calitatea și eficiența, influențând aspecte precum capacitatea de gestionare a datelor, timpul de răspuns, protecția împotriva amenințărilor cibernetice, gestionarea erorilor și nu în ultimul rând compatibilitatea cu diferite platforme.

Tabelul 1.2 reprezintă exemple de obiective non funcționale ale aplicației propuse spre implementare.

Tabel 1.2 Obiectivele non- funcționale ale aplicației

Obiectiv non-funcțional	Descriere
Performanță	Aplicația trebuie să dispună de o performanță receptivă și rapidă, timpii de răspuns fiind scurți, încărcarea paginilor fiind una rapidă.
Scalabilitate	Capabilitatea aplicației de adaptare ușoară a creșterii volumului de cererilor simultane din partea numărului mare de utilizatori și în acest timp, performanța fiind neafectată.
Disponibilitate	Accesibilitate și funcționalitate în majoritatea timpului, cu perioade de nefuncționare planificate minime și o strategie de backup și recuperare în caz de eșec.
Securitate	Aplicația trebuie să asigure o securitate robustă pentru a preveni accesul neautorizat și a proteja datele utilizatorilor.
Ușurință în utilizare	Interfața aplicației trebuie să fie intuitivă și prietenoasă, facilitând utilizatorilor interacțiunea fără efort și atingerea rapidă a obiectivelor lor.
Portabilitate	Aplicația trebuie să fie compatibilă cu o varietate de platforme și browsere web, asigurând astfel utilizarea sa pe diverse dispozitive și medii.
Extensibilitate	Posibilitatea de adăugare de noi funcționalități sau de a efectua modificări ulterioare fără a afecta negativ structura sau performanța generală.
Suport și explicații	Datele prezentate trebuie să fie explicate printr-un sistem oferit de aplicație, împreună cu suport pentru a ajuta utilizatorii să înțeleagă și să utilizeze eficient aplicația.

1.3 Specificații

Pentru ca obiectivele menționate mai sus să fie îndeplinite, primul pas în obținerea aplicației cu specificațiile dorite este crearea unei baze de date care permite stocarea și organizarea informațiilor astfel încât datele sunt gestionate, accesate și manipulate corespunzător cerințelor. Se va folosi Firebase Database, baza de date NoSQL, care se bazează pe un model de stocare nelimitat și flexibil, pentru o manipulare mai eficientă și scalabilitate mai ușoară a datelor nestructurate. [5]

Cartea [6] oferă o analiza detaliată a arhitecturii aplicațiilor web și evidențiază avantajele lor în cadrul procesului de dezvoltare software. Autorii oferă studii de caz și exemple practice pentru a ilustra beneficiile aplicațiilor web. De exemplu, în dezvoltarea unei aplicații de comerț electronic, beneficiile ar putea fi: accesibilitate globală, reducerea costurilor operaționale pentru un magazin fizic, personalizarea produselor pe placul utilizatorului și analiza detaliată a comportamentului utilizatorilor, performanța produselor și rezultatelor campaniilor de market. Sunt explorate, de asemenea, și diverse aspecte ale acestor exemple precum gestionarea datelor, securitatea și scalabilitatea.

Aplicația software creată și dezvoltată este o aplicație web, caracterizată de accesibilitatea și ușurința de utilizare prin intermediul unui browser web. Aceasta funcționează pe un server și pune la dispoziția utilizatorilor funcționalități și servicii prin intermediul internetului.

Dezvoltarea părții de backend, cât și a celei de frontend, sunt acțiunile esențiale în crearea aplicației. Backend-ul reprezintă partea server-side a aplicației, responsabilă de gestionarea logicii și comunicarea cu baza de date. Limbajul de programare ales pentru partea de server este Java [7] cu framework-ul Spring Boot [8], o combinație ce facilitează dezvoltarea rapidă și eficientă.

Partea de frontend reprezintă mediul de construire a interfeței utilizatorului cu o interactivitate a componentelor, iar aceasta este dezvoltată cu ajutorul bibliotecii JavaScript, React.js [9].

Gestionarea și citirea la fiecare 60 de minute a indicelui de calitate, a valorilor poluanților, a stării atmosferice și a ultimei date în care acestea au fost actualizate se realizează pentru fiecare senzor în parte, plasat pe o hartă Google Maps [10] din librăria React. Această componentă oferă o integrare convenabilă cu API-ul Google și o manipulare ușoară pentru utilizator.

Senzorii pot fi găsiți foarte ușor pe hartă cu ajutorul componentei Combobox, o fundație accesibilă a sistemului de design bazat pe biblioteca React [11], deoarece aceasta facilitează implementarea unui câmp de căutare a unei locații specifice. În acest fel, utilizatorul introduce locația dorită, primește opțiuni sugerate și este redirecționat în acea parte a hărții verificând astfel dacă există senzori plasați în acea zonă și analizând calitatea aerului.

Senzorii pot fi gestionați de asemenea într-un tabel din biblioteca material-table React [12], Această componentă dispune de sortarea datelor, căutarea după cuvinte cheie și

paginarea pentru a ușura procesul de navigare în pagină. Senzorii pot fi eliminați sau datele lor pot fi modificate. Dacă senzorul devine inactiv (prin modificarea câmpului „active” de tip boolean), atunci datele măsurătorilor nu vor mai fi disponibile.

Dacă se dorește o analizare mai profundă a datelor măsurătorilor unui anumit senzor, utilizatorul poate naviga către o pagina unde sunt grafice reprezentând valorile măsurate în ultimele ore. Pentru aceasta se folosește biblioteca Highcharts React [13], care dispune de o integrare simplă cu animații și interactivitate pentru a genera o experiență cat mai plăcută utilizatorului.

Aplicația beneficiază de serviciul de autentificare folosit de Firebase, numit Firebase Authentication care ușurează gestionarea și autentificarea utilizatorilor.

Dacă se dorește crearea unui nou cont, pe parte de frontend se va folosi metoda `createUserWithEmailAndPassword`, care va genera un nou cont de utilizator cu o parolă și o adresă de email, vizibilă de asemenea în console Firebase Authentication. Atunci când utilizatorul este înregistrat și dorește să se autentifice, metoda `signInWithEmailAndPassword` este utilizată pentru a verifica dacă credențialele introduse corespund.

În caz de uitare a parolei unui cont existent, se va folosi metoda care va trimite un email de resetare a parolei, `sendPasswordResetEmail`.

Partea de backend folosește structura de date `FirebaseToken` din biblioteca `com.google.firebase` [14]. Aceasta conține informații de autorizare și autentificare pentru a valida identitatea utilizatorului și solicitările acestuia prin verificarea token-ului trimis de frontend la autentificarea user-ului.

2 Studiu bibliografic

Când vorbim despre respirație, aproape toate viețuitoarele Pământului dispun de aceasta abilitate, însă nu toate beneficiază de același aer curat. Calitatea vieții și sănătatea viețuitoarelor poate fi amenințată de unele substanțe toxice din atmosfera emise de forțe naturale și activități umane. Orice agent chimic, biologic sau fizic poate modifica caracteristicile naturale ale atmosferei. [15]

Ozonul (O₃), dioxidul de azot (NO₂), dioxidul de sulf (SO₂) și particulele suspendate în aer (PM 2.5, PM10) sunt substanțele incluse în poluanții care prezintă o preocupare importantă pentru sănătatea publică.

PM2.5 și PM10 reprezintă o combinație complexă de solide și aerosoli, formată din mici picături de lichid, fragmente solide uscate și nuclee solide acoperite cu lichide. Particulele variază semnificativ în ceea ce privește dimensiunea, forma și compoziția chimică, cu posibilitatea de a conține ioni anorganici, compuși metalici, substanțe provenite din scoarța terestră. Sursele PM2.5 și PM10 sunt de multe ori diferite, compozițiile lor fiind distincte. Poluarea cu PM2.5 poate proveni din arderea de benzină, ulei, motorină sau lemn. În adăție, PM10 include și praful provenit din apropierea șantierei de construcții, incendiile de pădure și arderea vegetației/deșeurilor, surse industriale, polenul și fragmentele de bacterii.

Expunerea la PM produce efecte adverse asupra sănătății, cauzând boli precum afecțiuni cronice ale inimii și a sistemului respirator. Conform unui articol publicat de către Agenția Internațională pentru Cercetarea Cancerului, particulele suspendate în aerul exterior cauzează cancerul pulmonar. [16] Copii, persoanele astmatice și adulții cu afecțiuni pulmonare cronice sau de inimă fac parte din grupul cu cel mai mare risc de a experimenta efecte adverse asupra sănătății ca urmare a expunerii atât la PM2.5, cât și la PM10. [17]

Dioxidul de azot (NO₂) este unul dintre gazele extrem de reactive, fiind considerat o principala sursă de aerosoli de nitrat. Sursele de poluare cu acest gaz se formează din emisiile la nivelul solului asociate arderii combustibililor fosili în principal proveniți de la vehiculele rutiere, centrale electrice, echipamente pentru grădinarit, întreținerea peluzelor.

Căile respiratorii pot fi iritate de către o concentrație ridicată de NO₂, copii astmatici fiind afectați în mod specific în urma expunerii pe termen lung la acest gaz deoarece simptomele de bronșită se accentuează. [18]

Cel mai întâlnit poluant din atmosfera este dioxidul de sulf (SO₂), fiind prezent în concentrații ridicate în zonele urbane și industriale. În prezența umidității și altor poluanți, SO₂ contribuie la formarea unor straturi de coroziune vizibile. Principala sursă umană a acestui gaz este arderea combustibililor fosili, iar inhalarea acestui rezultat în concentrații ridicate poate cauza stimularea nervilor din căile respiratorii, rezultând modificări patologice, inclusiv edem pulmonar – afecțiune datorată acumulării de lichid în plămâni. [19]

Ozonul (O₃) este un poluant care cuprinde un ansamblu de compuși chimici rezultând într-o serie de reacții complexe în atmosfera, în urma transferului de dioxid de azot (NO₂) atunci când acestea absorb lumina provenita de la radiația solară.

Organizația Mondială a Sănătății (World Health Organization), cunoscută și sub numele de Agenția Națiunilor Unite are ca scop, după cum numele sugerează, să promoveze stilul de viață sănătos, să mențină lumea în siguranță și să fie de ajutor pentru cei vulnerabili, astfel încât lumea să poată trăi apogeul nivelului de sănătate, toate metodele adoptate fiind bazate pe pură știință.

Unul dintre cele mai importante subiecte abordate de către cercetătorii implicați în WHO este poluarea aerului. Ei afirmă că poluarea atmosferei rezultă din prezenta unei sau mai multe contaminări în atmosfera (fum, gaz, praf, ceață sau vapori), iar aceasta expusă în cantități și perioade de timp mari este o amenințare pentru viețuitoarele planetei. Principalele boli care vin în consecința expunerii oamenilor la poluarea aerului sunt accidente vasculare cerebrale, boli cardiace, rezultate adverse ale sarcinii, diabet, tulburări cognitive și boli neurologice.

Organizația Mondială a Sănătății recomandă un set de limite pentru poluanți și propune 3 etape de urmat pentru a reduce progresiv poluarea aerului, recomandate mai ales în zonele cu risc ridicat de poluare. În alta ordine de idei, aceste etape reprezintă niveluri de poluanți din atmosfera care sunt mai ridicate în comparație cu nivelurile recomandate de calitate a aerului, dar pe care autoritățile din zonă le pot impune pentru a dezvolta politici de diminuare a poluării, realizabile într-un interval de timp realist. Acestea sunt de asemenea considerate pași către atingerea nivelurilor recomandate de către WHO.

Tabelul de mai jos prezintă nivelurile recomandate de calitate a aerului (AQG - air quality guidance level) și țintele intermediare (Interim target) în 2 intervale medii pentru 365 zile, respectiv 24 ore. [20]

Pollutant	Averaging time	Interim target				AQG level
		1	2	3	4	
PM_{2.5}, µg/m³	Annual	35	25	15	10	5
	24-hour ^a	75	50	37.5	25	15
PM₁₀, µg/m³	Annual	70	50	30	20	15
	24-hour ^a	150	100	75	50	45
O₃, µg/m³	Peak season ^b	100	70	–	–	60
	8-hour ^a	160	120	–	–	100
NO₂, µg/m³	Annual	40	30	20	–	10
	24-hour ^a	120	50	–	–	25
SO₂, µg/m³	24-hour ^a	125	50	–	–	40
CO, mg/m³	24-hour ^a	7	–	–	–	4

^a 99th percentile (i.e. 3–4 exceedance days per year).

^b Average of daily maximum 8-hour mean O₃ concentration in the six consecutive months with the highest six-month running-average O₃ concentration.

Figura 2.1, Nivelurile recomandate de către WHO pentru calitatea aerului, WHO [20]

Rezultatele cercetărilor demonstrează că majoritatea (99%) populației globale respira aer care conține nivele înalte de poluanți, aceasta depășind limitele prezentate anterior, aproximativ 2.4 miliarde de oameni fiind expuși la nivele ridicate de pericol.

Nu doar sănătatea publică este în pericol, dar poluanții din atmosfera au un impact negativ asupra ecosistemului și climei pământului la nivel global. În particulele fine (PM) se găsește carbon negru care este considerat unul dintre cei mai influenți contribuitori la încălzirea globală. Acesta încălzește atmosfera pământului absorbind lumina soarelui, accelerând astfel topirea gheții și a zăpezilor.

Ca urmare, WHO propune sugestii pentru îmbunătățirea calității aerului, punând la dispoziția populației informații prognozate a poluării aerului împreună cu date de monitorizare a acestuia în timp aproape real. În plus, organizația a elaborat și implementat o strategie de conștientizare a oamenilor cu privire la riscul de contaminare a aerului prin monitorizarea indicelui de calitate. [4]

Majoritatea țărilor europene folosesc un indice comun propus de European Environment Agency. În plus, acesta oferă informații despre sub populațiile afectate, consecința care reprezintă simptomele probabile și în final recomandări specifice pentru a diminua expunerile și riscurile pentru sănătate. Un exemplu este platforma online bazată pe un sistem de informații geografice pentru monitorizarea în timp real a

calității aerului în Europa. Aceasta este accesibilă atât sub forma de aplicație web, cât și aplicație pentru telefoane mobile. Datele sunt raportate sub forma EAQI (European Air Quality Index), concept dezvoltat de către European Environment Agency. EAQI reprezintă "European Air Quality Index" (Indicele European al Calității Aerului). Acest indice este calculat în funcție de nivelurile de poluare, inclusiv particule în suspensie (PM10 și PM2.5), dioxid de azot (NO2), ozon (O3), dioxid de sulf (SO2) și monoxid de carbon. (CO). Valoarea indicelui reprezintă maximum dintre valorile poluanților gazoși, cu o condiție: valorile pentru PM2.5 și NO2 sunt obligatorii în acest calcul. Alte informații accesibile sunt locația stației pe hartă și un mesaj corespunzător bazat pe nivelul indicelui curent.

Health Effect Institute (HEI) este o organizație de cercetare cu privire la efectele poluării aerului asupra sănătății, înființată în 1980. Epidemia de cancer pulmonar din anii 1950 din Statele Unite ale Americii și Europa de Vest a fost un impuls pentru cercetători să examineze efectele aerului poluat din exterior, fiind considerată drept o cauză. Deși consumul de tutun a avut un rol central în această epidemie, rezultatele studiilor demonstrează că agenții cancerigeni sunt eliberați în aerul exterior și inspirarea acestora poate duce la cancerul pulmonar. [21]

Contaminarea aerului este un factor major care provoacă milioane de decese în fiecare an, înregistrându-se 6.67 milioane la nivel global în 2019. În același an, poluarea aerului a fost al patrulea principal factor de risc pentru deces, impactul său total fiind depășit doar de hipertensiune arterială, riscuri alimentare și consumul de tutun. [22]

Este îngrijorător faptul că mai mulți oameni mor ca urmare a expunerii la poluarea aerului decât în urma accidentelor rutiere, un număr estimat la 1.28 milioane în 2019 sau bolilor cronice.

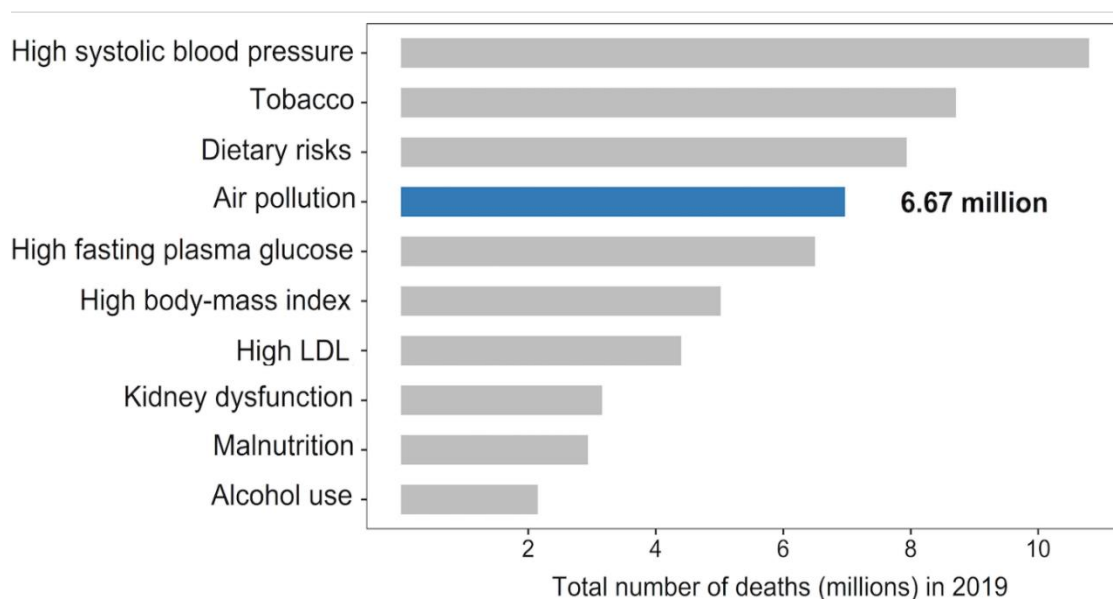
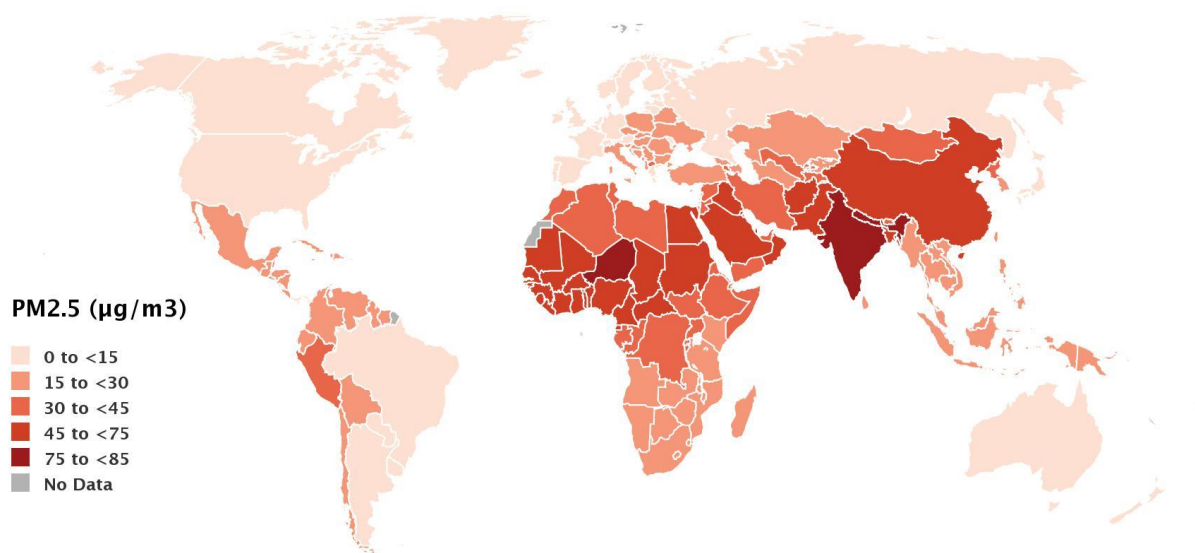


Figura 2.2 Clasamentul global al factorilor de risc în funcție de numărul total de decese din toate cauzele în anul 2019 – State of Global Air [22]

State of Global Air [22] este soluția Health Effect Institute, reprezentată de un website care produce o analiză corespunzătoare a nivelurilor și tendințelor privind calitatea și sănătatea aerului pentru fiecare țară din lume. Un exemplu de raport preluat dintr-un studiu de caz [23] este prezentat în figura de mai jos, unde în 2019 mai mult de 90% din populația lumii a experimentat o concentrație care a depășit limitele anuale (tabelul de mai sus) WHO (10 micrograme pe metru cub) pentru PM2.5. Cele mai mari expuneri au fost observate în Asia, Africa și Orientul Mijlociu: India – cu 83.2, Nepal – cu 83.1, Niger 80.1, Qatar – cu 76.0 etc, iar cele mai mici expuneri (concentrații medii anuale ponderate sub 8 $\mu\text{g}/\text{m}^3$ în funcție de populație) sunt surprinse în Australia, Brunei (Indonezia), Canada, Estonia, Finlanda, Islanda etc.

Average Annual Population–Weighted PM2.5 Concentrations in 2019



State of Global Air 2020

Figura 2.3 Concentrații ale poluantului PM2.5 în anul 2019 – State of Global Air [22]

Proiectul State of Global Air oferă o cuprinzătoare și comparativă analiză a principalilor factori de risc care contribuie la apariția bolilor și decesului prematur la nivel mondial. În concluzie, aceasta este o baza esențială pentru stabilirea priorităților în îmbunătățirea sănătății oamenilor. Conștientizarea în ceea ce privește pericolul bolilor ce pot apărea în urma poluării aerului este importantă și vitală pentru planificare acțiunilor de reducere a poluării în moduri cât mai eficiente.

O altă platformă care are ca scop combaterea poluării aerului este Airly [24]. Sistemul constă în diferite forme de monitorizare ca senzori, o platformă web sau o aplicație mobilă. Senzorii măsoară indexul de calitate a aerului folosind valorile particulelor suspendate în aer PM1, PM2.5 și PM10 de la cea mai apropiată stație din

zona, iar sursele de poluare principale sunt traficul rutier, ferme și incendiile pădurilor. Normele acceptate de către Airly pentru PM2.5 și PM10 sunt cele recomandate de WHO, iar utilizatorul dispune de o scară cu 7 nivele pentru index, începând cu verde (calitate excelentă) până la maro (nivel îngrijorător de ridicat).

Populația României este informată în timp real de parametrii de calitate a aerului, monitorizați în mai mult de 100 de stații care cuprind toată suprafața țării și care, împreună formează Rețeaua Națională de Monitorizare a Calității Aerului (RNMCA) [25]. Datele sunt actualizate în fiecare ora, iar poluanții care sunt luați în considerare pentru a calcula indicele de calitate sunt SO₂, NO₂, O₃, PM_{2.5} și PM₁₀. Indicele general este exprimat pe o scară de la 1 la 6, 1 reprezentând o calitate foarte bună, iar 6 una extrem de rea. Fiind conștiente de poluarea aerului cu ajutorul RNMCA, autoritățile locale pot lua măsuri prompte în timp util pentru eliminarea sau minimizarea poluării sau în cazuri de urgență, să avertizeze și protejeze oamenii.

În Figura 2.4 se poate observa valoarea pentru PM_{2.5} înregistrată de către stație CJ-3 din județul Cluj, la fiecare interval orar, reprezentată atât sub forma de grafic, dar și tabel.

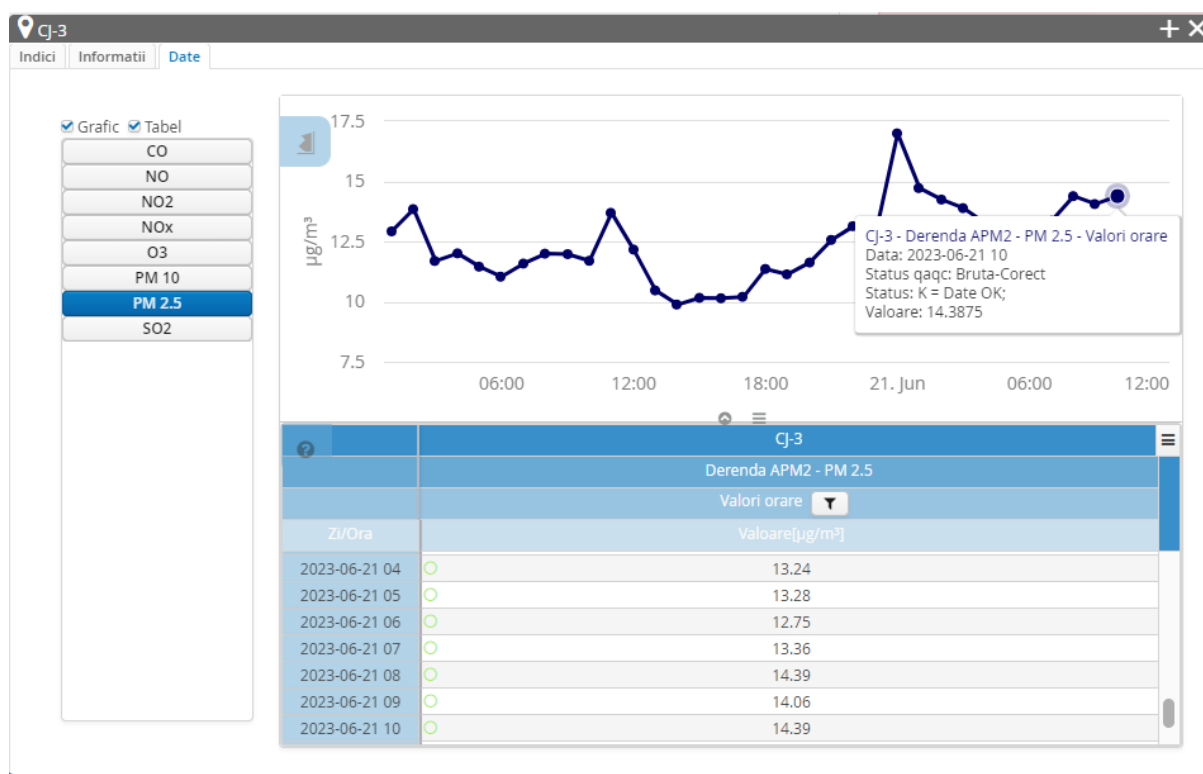


Figura 2.4, Senzor colectare a datelor poluanților și grafice corespunzătoare în județul Cluj, România [25]

Referitor la articolul publicat de WHO, multe țări europene, pe lângă indicele sugerat de EEA și concentrațiile monitorizate ale poluanților, oferă și valori prognozate. Aceste valori sunt suficient de fiabile pentru a raporta concentrațiile de poluanți pentru următoarea zi, aceste precizii fiind asigurate în mare parte de variațiile zilnice ale calității aerului. [26]

În continuare se vor enumera câteva țări, respectiv aplicațiile pe care acestea le folosesc și la care se raportează atunci când vine vorba de monitorizare a aerului.

Germania – Portalul de date privind calitatea aerului actuală

Valorile în portal sunt măsurate și comunicate folosind o harta care acoperă întreaga suprafața a țării, iar graficele reprezintă date istorice. În plus, se oferă recomandări și sfaturi de sănătate pentru diferite grupuri de populație [27]

Portugalia – aplicația QualAr

Se redau informații zilnic despre calitatea aerului, dar și prognoze pentru locațiile selectate. Se furnizează avertismente privind sănătatea folosind un sistem care notifica utilizatorul în funcție de indicele QualAr estimat.

Polonia – Portalul de calitate a aerului (Polski Indeks Jakości Powietrza)

Stațiile de măsurare automate oferă date actuale privind calitatea aerului, așa cum sunt furnizate de la autorități, însoțite de mesaje privind sănătatea bazate pe valoarea indicelui, plus o prognoza pentru următoarele câteva zile.

Poluarea aerului este unul dintre cele mai discutate subiecte la nivel mondial, cercetătorii studiază cu interes atât cauzele, cât și consecințele acestuia, iar multe studii arată că este o sursă negativă de boli grave. De aceea, monitorizarea aerului exterior îi va determina pe oameni să conștientizeze gravitatea situației și să participe la măsurile care se iau la nivel global.

3 Analiză și proiectare

3.1 Metoda de calcul pentru Indicele de Calitate a Aerului

Determinarea indicelui de calitate a aerului se bazează pe evaluarea a 5 valori ale parametrilor cheie: PM10 și PM2.5 - particulele în suspensie cu diametrul mai mic decât 10, respectiv 2.5 microni, NO₂ - ozonul, SO₂ - dioxidul de sulf și NO₂ - dioxidul de azot. Înregistrarea valorilor poluanților și a stării atmosferice se va face la fiecare ora, dar exista posibilitatea ca unele valori să lipsească de la oră la oră, de aceea este important să se ia acest aspect în considerare. NO₂ și PM2.5 sunt poluanți gazoși emiși în special de industrie, centrale termice, vehicule și alte surse. Pentru a evalua nivelul general de poluare a aerului, monitorizarea acestor poluanți este esențială iar prin evaluarea atentă a acestor concentrații, putem obține informații cruciale despre calitatea aerului și posibilele riscuri pentru sănătatea publică. WHO susține că NO₂ și PM2.5 sunt cele mai periculoși poluanți gazoși, reprezentând principalele cauze de decese premature. [28]

Valorile pentru concentrațiile poluanților gazoși PM2.5 și PM10 sunt exprimate în $\mu\text{g}/\text{m}^3$ (micrograme pe metru cub), pentru NO₂, SO₂ și O₃ sunt exprimate în $\mu\text{g}/\text{m}^3$, părți pe milion (ppm) sau părți pe miliard (ppb), iar indicele de calitate a aerului se exprimă în $\mu\text{g}/\text{m}^3$, unde ppb reprezintă volumul de poluant gazos din 10⁹ volume de aer.

În această lucrare, senzorii înregistrează valorile măsurătorilor (poluanților gazoși) în părți pe miliard, de aceea este necesară conversia din ppb în $\mu\text{g}/\text{m}^3$, iar aceasta depinde de masa moleculară a poluantului gazos și densitatea aerului.

Conversia pentru poluanți gazoși este deja cunoscută și ușor accesibilă, așadar aceasta este prezentată în Tabelul 3.1. [29]

Tabel 3.1 Conversia între ppb și $\mu\text{g}/\text{m}^3$ pentru poluanții gazoși folosiți

Poluant gazos	
SO ₂	1 ppb = 2.62 $\mu\text{g}/\text{m}^3$
NO ₂	1 ppb = 1.88 $\mu\text{g}/\text{m}^3$
O ₃	1 ppb = 1.96 $\mu\text{g}/\text{m}^3$

Astfel, pentru a se efectua calculele, valorile poluanților PM10 și NO₂ sunt obligatorii, iar valoarea maximă dintre cele existente reprezintă indicele de calitate a aerului exterior. Metoda de calcul este de asemenea prezentată și pe "European Air Quality Index", un website oficial al Uniunii Europene. [30]

$$\text{Indicele de Calitate a Aerului} = \max(\text{PM10}, \text{PM2.5}, \text{NO}_2, \text{SO}_2, \text{O}_3) \left[\frac{\mu\text{g}}{\text{m}^3} \right]$$

unde PM10 și NO₂ sunt valorile necesare.

Rezultatul calculelor se va raporta mereu cu valorile recomandate de către Organizația Mondială a Sănătății, iar în funcție de indicele rezultat, exista mesaje ce conțin recomandări de activități ce se pot desfășura de către oameni în acea perioada.

Tabelul 3.1 reprezintă intervalele pentru fiecare poluant gazos împărțit în 6 nivele. “Foarte bine” reprezintă o calitate excelentă, iar „Extrem de rău” reprezintă o calitate foarte proastă.

Tabel 3.2 Intervalele recomandate de OMS pentru poluanții gazoși

Poluant gazos	Foarte bine	Bine	Moderat	Rău	Foarte rău	Extrem de rău
PM2.5	0-10	10-20	20-25	25-50	50-75	>75
PM10	0-20	20-40	40-50	50-100	100-150	>150
NO2	0-40	40-90	90-120	120-230	230-340	>340
O3	0-50	50-100	100-130	130-240	240-380	>380
SO2	0-100	100-200	200-350	350-500	500-750	>750

3.2 Arhitectura software

3.2.1 Structura aplicației

Arhitectura software în cadrul unei aplicații web definește modul în care componentele și modelele interacționează între ele, cum acestea sunt organizate și cum sunt distribuite responsabilitățile.

Stabilirea unei structuri arhitecturale solide pentru o aplicație web este de o importanță vitală în organizarea și structurarea codului, aceasta permițând adaptabilitatea și oportunitatea spre extindere a aplicației. De pe urma acestei implementări, se va putea reutiliza și modulariza codul, testa în mod eficient și îmbunătăți nivelul de securitate al aplicației.

Arhitectura aplicației web în aceasta lucrare este bazată pe niveluri principale (layered architecture), acestea 3 fiind: baza de date (nivelul de acces la date), backend (nivelul de logică) și frontend (interfața cu utilizatorul). Conectarea acestora rezultă dependențe reciproce care asigură funcționarea corectă a aplicației.

Figura 3.1 prezintă schema arhitecturii aplicației bazată pe niveluri.

Arhitectura de nivel inalt a aplicației web

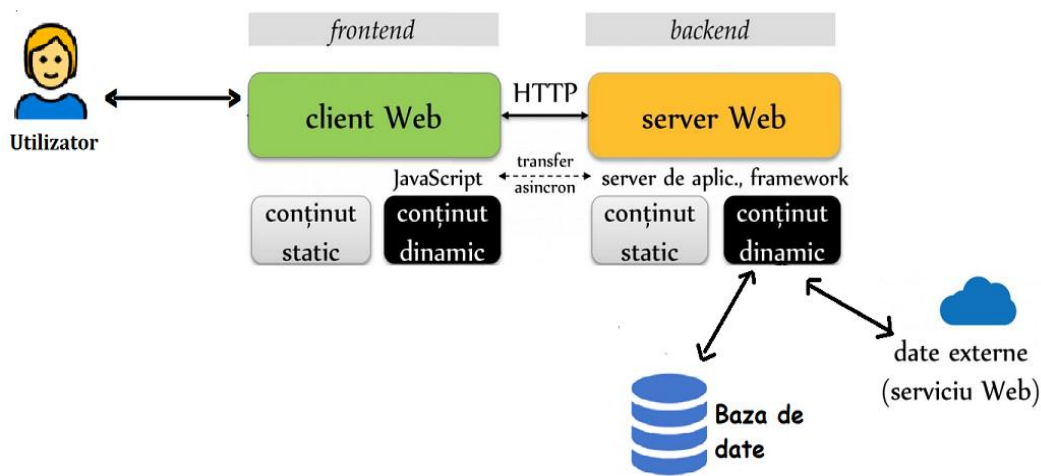


Figura 3.1 Arhitectura de nivel inalt a aplicației web

3.2.2 MCV

MCV (Figura 3.2) este un concept de proiectare folosit în dezvoltarea aplicațiilor web, în special în partea de frontend. Avantajele folosirii acestui concept sunt: posibilitatea de a reutiliza codul și testarea și modularitatea cu ușurință a fiecărei componente în parte. Logica se va împărți în 3 componente distincte:

1. Modelul (Model) gestionează datele și accesul la acestea, dar este responsabil și de interacțiunea cu baza de date.
2. Controlorul (Controller): manipulează logica aplicației și acționează ca intermediar între model și vizualizare. Primește cereri și acțiuni de la utilizatori și ia decizii referitoare la datele necesare din model și la modul de prezentare a acestora utilizatorului.
3. Vizualizarea (View): este responsabilă de afișarea informațiilor utilizatorului într-un mod adecvat. Preia datele din model și le prezintă într-o formă ușor de înțeles.

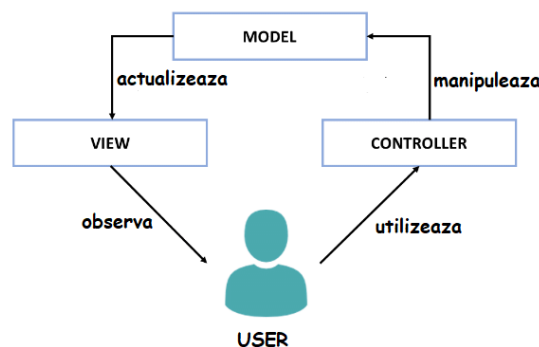


Figura 3.2 Arhitectura MCV

3.3 Concepte teoretice

3.3.1 Baza de date

Baza de date este reprezentată printr-o colecție de informații, organizate, structurate și stocate electronic. Aceasta este organizată în așa fel încât aplicația web să poată accesa, gestiona și actualiza datele primite.

3.3.2 Partea server

Partea server (backend) este responsabilă pentru logica și procesarea datelor în spatele aplicației. Funcționarea corectă și interacțiunea eficientă sunt acțiunile principale care se desfășoară în această parte.

Backend-ul gestionează rutele și punctele de acces (endpoint-uri) ale aplicației, direcționând cererile către modulele sau serviciile adecvate. Acesta implementează logica aplicației, inclusiv reguli de validare, calcule complexe și manipulări de date, asigurând funcționarea corectă a aplicației conform scopului său.

Serverul se conectează la baza de date pentru a salva, accesa și actualiza datele, utilizând limbaje de interogare pentru a manipula informațiile. De asemenea, securitatea aplicației și a datelor utilizatorilor este asigurată prin autentificare și protecție împotriva atacurilor.

3.3.3 Partea client

Partea client (frontend) reprezintă partea vizibilă a aplicației web cu care utilizatorul interacționează. Aici este inclusă interfața grafică, elementele de navigare și funcționalitățile interactive.

Frontend-ul este responsabil de proiectarea și implementarea interfeței utilizatorului, aceasta implicând crearea structurii și aspectului aplicației, a componentelor vizuale, a stilizării și a animațiilor.

Partea client implementează funcționalități care rulează în browser-ul utilizatorului, cum ar fi validarea datelor, gestionarea interacțiunilor și manipularea stării aplicației. Frontend-ul interacționează în legătură strânsă cu backend-ul, integrând astfel interfața utilizatorului cu funcționalitatea server-side. Aceasta implică definirea și utilizarea API-urilor, transmiterea și recepționarea datelor între cele două părți și asigurarea unei comunicări eficiente.

Nu în ultimul rând, partea de client eficientizează aplicația pentru diverse dispozitive și dimensiuni de ecran, asigurându-se că interfața se redimensionează și se adaptează la diferite platforme. Se optimizează performanța aplicației prin reducerea timpului de încărcare, minimizarea cererilor către server, perfecționarea codului JavaScript și CSS, implementarea cache-ului și a metodei de îmbunătățire a vitezei de răspuns.

3.3.4 REST

REST este un stil arhitectural pentru aplicațiile web, bazat pe principii și restricții. Acesta separă aplicațiile în client - frontend și server - backend, folosește cereri fără stare (între cereri, serverul nu este nevoit să păstreze nicio stare despre client), definește o interfață uniformă și permite stocarea în cache a răspunsurilor. REST facilitează dezvoltarea aplicațiilor web adaptabile, compatibile și ușor de administrat.

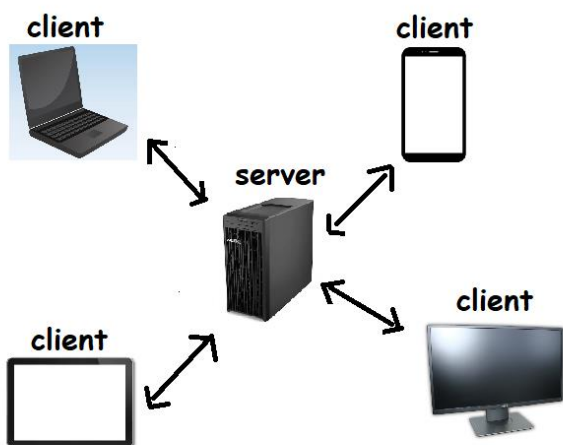


Figura 3.3 Arhitectura client-server

Principalele metode REST sunt:

- GET: obține date despre o sursă prin intermediul unui URL
- POST: creează o resursă nouă în server
- PUT: actualizează o resursă existentă
- DELETE: șterge o resursă specificată prin URL

3.4 Resurse software

3.4.1 Firebase

Firebase reprezintă o platformă deținută de Google, specializată în dezvoltarea aplicațiilor de tip Web și mobile. Aceasta pune la dispoziție un set complet de servicii backend, care facilitează implementarea și gestionarea aplicațiilor într-un mod simplu și eficient. Baza de date Firestore de tip NoSQL permite stocarea și sincronizarea datelor în timp real între diferite dispozitive. Firestore dispune de asemenea de serviciu de autentificare și autorizare, utilizând metode simple pentru utilizatori. Firestore storage este un alt serviciu oferit reprezentat de un sistem de stocare scalabil și fiabil pentru fișiere.

În scopul realizării acestei aplicații, s-a utilizat Firestore Database, caracterizată printr-o structură flexibilă și complexă, cu o capacitate de stocare mare. Datele sunt organizate în colecții, iar aceasta poate conține mai multe documente care sunt

identificate fiecare printr-un ID unic, conținând câmpurile necesare. De asemenea, este posibil să se organizeze documentele în categorii mai mici numite subcolecții, ceea ce facilitează crearea unei structuri ierarhice complexe.

Figura 3.4 descrie modul de structurare Firestore Database.

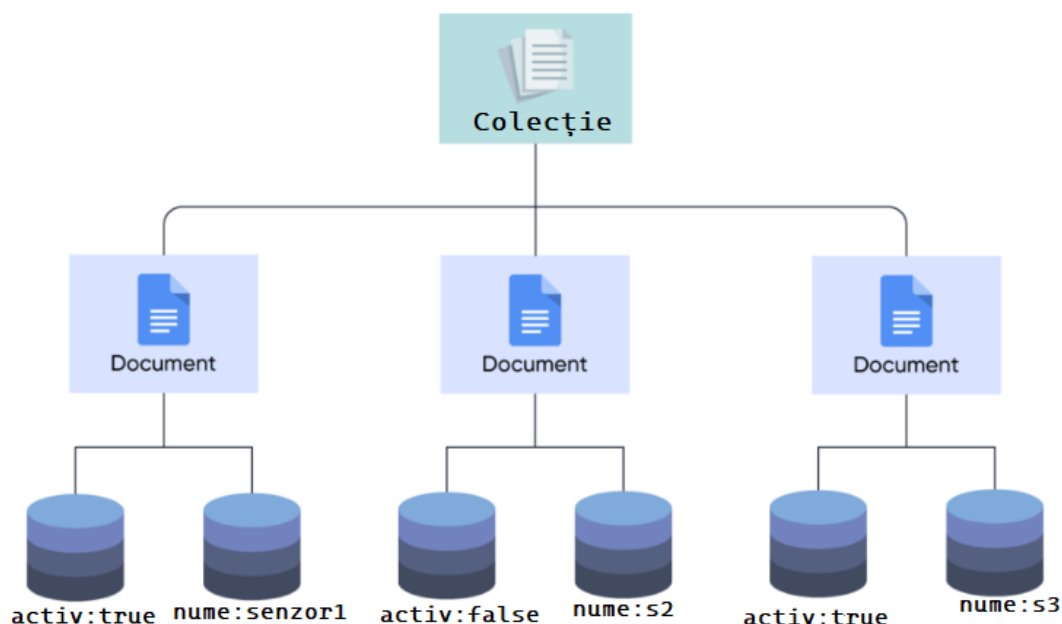


Figura 3.4 Modul de structurare a datelor în Firestore Database

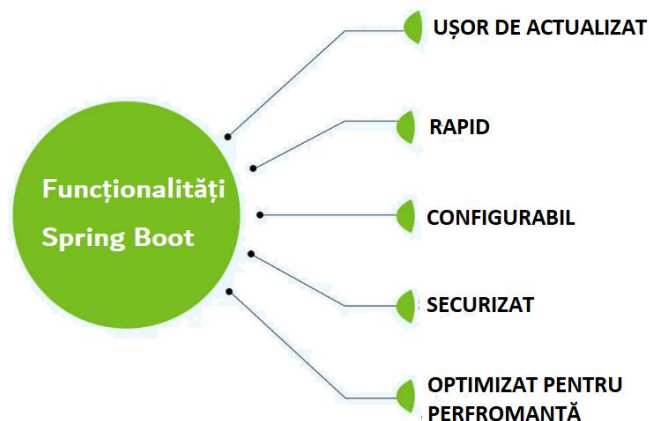
3.4.2 Spring Boot

Spring Boot este un cadru de lucru (framework) pentru dezvoltarea aplicațiilor Java, conceput pentru a simplifica și ca un rezultat, a accelera procesul de creare a aplicațiilor robuste și scalabile, deținut și susținut de către o divizie a VMware, compania Pivotal Software.

Spring Boot pune la dispoziție un ansamblu complet de funcționalități, reducând considerabil nivelul de dificultate și necesitatea configurării manuale a unei aplicații Java standard. Principiul este bazat pe "convenție în loc de configurare", ceea ce înseamnă că framework-ul vine cu setări implicite bine stabilite, astfel dezvoltatorii se concentrează mai mult asupra logicii aplicației și mai puțin asupra configurării infrastructurii.

Caracteristicile principale și alte beneficii oferite de SpringBoot (prezentate și în Figura 3.5):

- Livrarea unui set de funcționalități efectuate de către actuator pentru monitorizarea și gestionarea aplicațiilor în timpul execuției.
- Permite integrării cu alte framework-uri și biblioteci Java prin intermediul gestionării dependentelor și instrumentelor de construire a proiectului Maven.



Figură 1.5 Funcționalitățile principale oferite de Spring Boot

Arhitectura părții de backend a acestei lucrări (Figura 3.6) este reprezentată de următoarele componente:

1. Clasa responsabilă de implementarea logicii de lucrării, componenta a modelului de design „Service Layer” este numită „Service” și adnotată cu „@Service” pentru a fi marcată ca fiind un bean de serviciu. Aici se descrie funcționalitatea aplicației, în principiu metodele de creare, citire, actualizare și ștergere a informațiilor din baza de date.
2. FirebaseInitialize este responsabilă de inițializarea bazei de date Firestore Database. Spring Boot recunoaște și aceasta componentă ca fiind un serviciu, de aceea se folosește adnotarea @Service.
3. Componenta Controller este responsabilă de gestionarea interacțiunii între partea de frontend și backend. Adnotarea „@RestController” va automatiza serializarea și deserializarea obiectelor în format JSON. Acestea vor fi returnate direct ca răspuns HTTP în formatul dorit. Deoarece metodele din clasă pot gestiona operațiile HTTP (GET, POST, PUT, DELETE), se va simplifica implementarea API-urilor REST-ful.

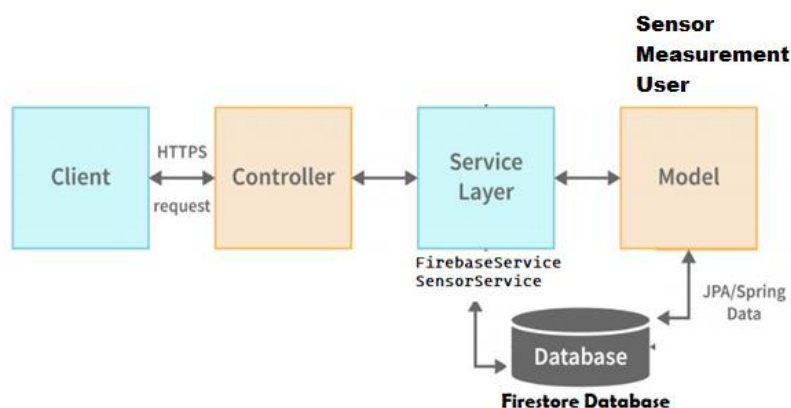


Figura 3.6 Arhitectura MVC

Adnotarea „@RequestMapping” în Controller este folosită pentru a mapa un URL de acces către o metodă specifică din controller. De exemplu, se va folosi adnotarea @PostMapping pentru a mapa cererea HTTP POST către metoda de creare din clasa Service. Figura 3.6 prezintă interacțiunea între o aplicație Java MVC și partea de client a unei aplicații web într-un browser web.

Clasa „Sensor” reprezintă o Entitate utilizată pentru a manipula și ilustra datele unui senzor, asociată unui document în baza de date Firestore. Atributele clasei „Sensor” corespund field-urilor documentului.

Prin utilizarea adnotărilor Lombok @AllArgsConstructor, @NoArgsConstructor și @Data se generează un constructor care primește toți parametrii, respectiv niciun parametru pentru clasa Sensor și se generează automat metode comune (gettere, settere).

Pentru a se realiza transferul datelor între diferite componente ale aplicației, se va utiliza DTO (Data Transfer Object) în Spring Boot (Figura 3.7). Aceste obiecte sunt create pentru a simplifica comunicarea și pentru a separa modelul de date intern de cel utilizat în interacțiunea cu utilizatorii sau alte sisteme externe.

Un exemplu de DTO integrat și folosit în aplicație pentru clasa „Sensor” se poate vizualiza în Figura 3.8.

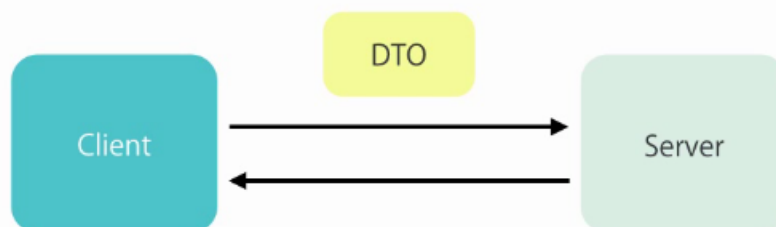


Figura 3.7 DTO (Data Transfer Object) în Spring Boot

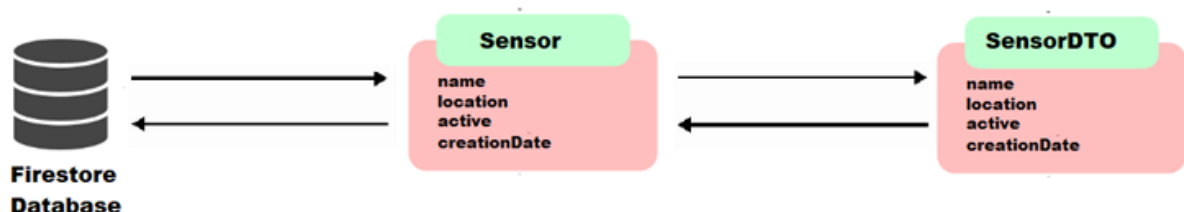


Figura 3.8 DTO integrat în aplicație

3.4.3 React JS

React.js este o bibliotecă JavaScript open-source creată în mare parte de echipa de dezvoltatori Facebook.

Cu ajutorul conceptului React.js, reprezentat de “componente”, se pot crea interfețe utilizator interactive și reactive pe web, care reprezintă părți mai mici și individuale din cadrul interfeței. Acestea pot fi reutilizate în diverse părți ale aplicației și pot comunica între ele, îmbunătățind astfel dezvoltarea și întreținerea aplicațiilor complexe. Atunci când doar anumite date se schimbă, React.js nu redă întreaga pagină, ci utilizează un mecanism eficient de “re-rendering virtual”, actualizând doar părțile relevante. Acest lucru asigură o experiență fluidă și o performanță îmbunătățită.

Alte funcționalități oferite de React.js sunt:

1. DOM-ul virtual - făcând comparații eficiente, acesta utilizează un arbore virtual în memorie pentru a actualiza doar elementele necesare în interfață.

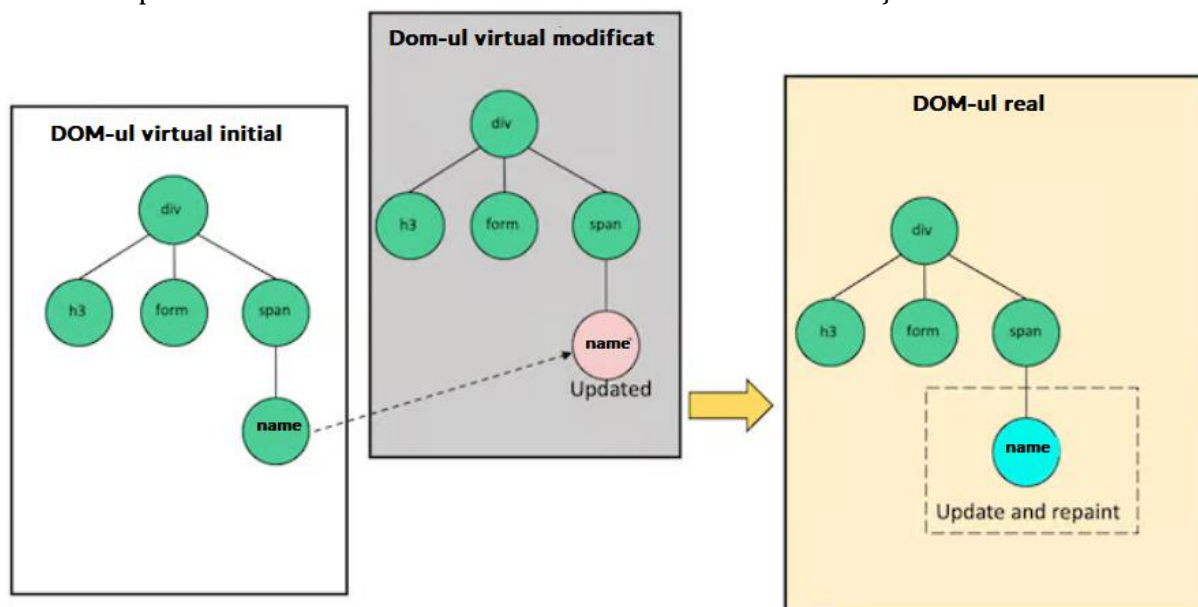


Figura 3.9 DOM-ul virtual React

În exemplul din Figura 3.9, observăm că pentru a modifica datele din câmpul „name”, React va crea un nou arbore cu DOM-ul virtual actualizat și îl va compara pe acesta cu DOM-ul virtual inițial pentru a identifica modificările necesare. După acest proces, React utilizează o bibliotecă de redare, cum ar fi „ReactDOM”, care preia informațiile despre diferențe pentru a actualiza aplicația redată. Astfel, se asigură că DOM-ul real primește și redesenează doar nodurile actualizate.

2. JSX: Permite utilizarea sintaxei JSX, combinând JavaScript și HTML într-un mod expresiv și ușor de înțeles.

3. Integrare ușoară: Se integrează cu alte biblioteci sau framework-uri, oferind extensibilitate.

În aplicația prezentată, React implementează diferite biblioteci pentru a oferi funcționalități și aspecte grafice predefinite, îmbunătățind astfel dezvoltarea și aspectul aplicației.

Se vor enumera bibliotecile utilizate și principiile după care sunt definite acestea:

3.4.1.1 React Google Maps API

Biblioteca React Google Maps API integrează și utilizează hărți Google Maps. Aplicația frontend va folosi componente și funcționalități ale acestei biblioteci pentru afișarea unei hărți interactive și gestionarea datelor afișate pe aceasta:

Pentru a permite încărcarea și inițializarea API-ului Google Maps, se va folosi funcția „useLoadScript” împreună cu cheia API furnizată. Cheia se va obține prin crearea unui proiect în consola Google Cloud și activarea serviciului Google Maps JavaScript API. Prin utilizarea acestei componente, se va realiza încărcarea asincronă a scripturilor necesare pentru funcționarea hărții.

Pentru a marca senzorii pe harta, se va folosi componenta „Marker”, aceasta fiind reprezentată de un pin pe hartă. „InfoWindow” este componenta care permite afișarea de conținut informativ asociat unui marker. Atunci când utilizatorul face clic pe un marker, se deschide un modal ce conține informații despre indicele de calitate și ultimele măsurători, aceste 2 informații fiind însoțite de data de actualizare.

Prin utilizarea React Google Maps API, se vor gestiona evenimentele de interacțiune din partea utilizatorului cu harta sau pinii marcați pe aceasta și navigarea (inclusiv zoom out, zoom in).

Figura 3.10 reprezintă arhitectura configurării API-ului Google Maps

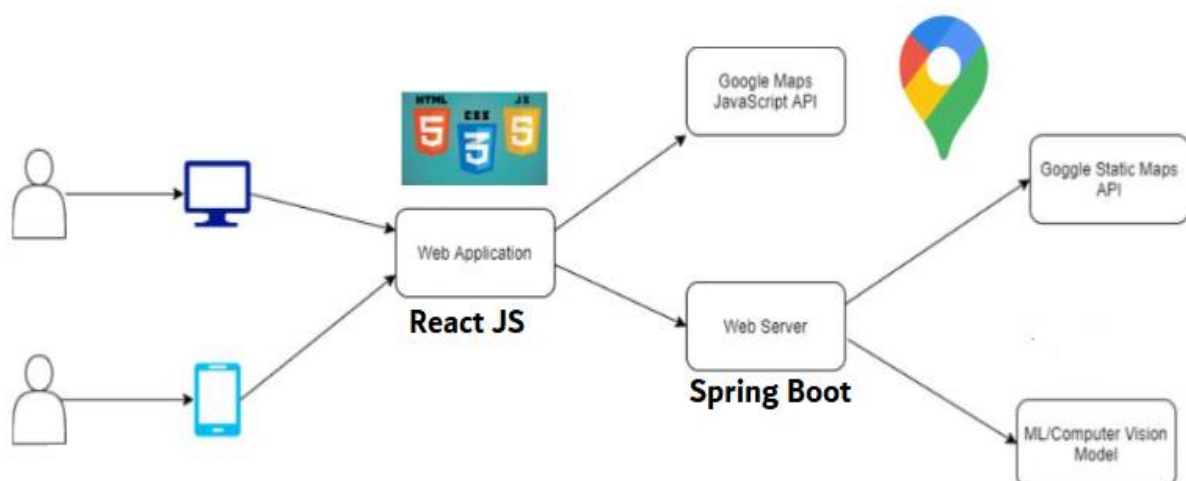


Figura 3.10 Arhitectura de servicii pentru Google Maps

3.4.1.2 Material UI și Material-Table

Biblioteca MUI – Material UI oferă dezvoltatorilor de aplicații React componente și stilizarea predefinită a acestora pentru a crea interfețe utilizator (UI) precum butoane (Button MUI), componente de navigare (Sidebar MUI sau NavBar MUI), liste (Lists MUI), tabele (Table MUI), ferestre modale (Dialog MUI) etc.

Folosirea și implementarea MUI oferă flexibilitate, iar componentele pot fi personalizate în funcție de nevoile aplicației (culori, dimensiuni, stiluri). Redimensionarea ecranelor este un alt beneficiu pe care MUI îl pune la dispoziție deoarece componentele se vor adapta la diferite dimensiuni de ecrane și dispozitive, astfel asigurând faptul că aplicația este ușor de utilizat pe diverse platforme.

Material-Table este o bibliotecă React JS pentru afișarea și manipularea datelor tabulare. Aceasta include sortarea, filtrarea (introducând termenii de căutare într-o fereastră), paginarea (datele pot fi afișate pe mai multe pagini pentru o vizualizare mai rapidă și ușoară) editarea și actualizarea datelor, selectarea rândurilor, acțiuni personalizate (ștergerea unui rând ce reprezintă un document în baza de date) și suport pentru date externe. Este ușor de personalizat și este utilizată pentru a crea tabele interactive și structurate în aplicațiile web. Un exemplu de randare a Material-Table în React JS, cu 2 coloane: „Name” și „Creation Date” este prezentat în Figura 3.11.



```
<MaterialTable
  columns={[
    {
      title: "Name",
      field: "name",
    },
    {
      title: "Creation Date",
      field: "creationDate",
    },
  ]}
  data={sensors}
/>
```

Figura 3.11 Randare Tabel în material-table

3.4.1.3 Formik

Pentru a crea, valida și manipula datele din formularele aplicației (Înregistrare user existent, Autentificare user nou, Modificare date senzor etc, adăugare sensor nou) într-un mod eficient și ușor de utilizat, se va folosi biblioteca Formik.

Aceasta implementare urmărește starea formularului prin implementarea unor pași: introducerea valorilor de intrare, urmărirea stării de atingere a câmpurilor, a erorilor de validare și a trimiterii formularului. Pentru a construi o logica de validare a datelor, se folosește biblioteca Yup, aceasta furnizează un mod puternic de definire a regulilor de validare pentru datele introduse.

După ce utilizatorul introduce datele într-un form Formik și acestea nu sunt în conformitate cu regulile de validare, yup va genera diferite mesaje de eroare, blocând accesul la trimiterea formularului mai departe. O schema Yup poate „modela” date

precum șiruri de caractere (nume senzor, parola), data (data plasării senzorului), numere, obiecte (locația GeoLocation din Firebase) etc.

Un exemplu de validare Yup se poate observa în Figura 3.12. Parola este obligatorie, minimul de caractere acceptat este 12, iar maximul 128.

```
export const ValidareParolaYup = object().shape({
  password: string()
    .nullable()
    .label('Password')
    .required('Required')
    .min(12, 'Password should be at least 12 characters')
    .max(128, 'Password is too long'),
})
```

Figura 3.12 Schema de validare pentru parola Yup

De asemenea, Formik furnizează metode și componente React pentru a gestiona evenimente precum submit, change și blur. Aceasta facilitează captarea și prelucrarea evenimentelor formularului, iar odată ce datele au fost validate, trecând prin filtrele Yup, Formik trimite datele completate către server.

Figura 3.13 reprezintă integrarea bibliotecii Formik cu Material-Table. În Form-ul de actualizare a senzorului, sunt afișate valorile câmpurilor din fiecare document senzor din baza de date, iar în tabel sunt afișate numele și data plasării acestuia. La fiecare schimbare a datelor senzorilor, baza de date va fi actualizată, care mai departe, va schimba interfața afișată.

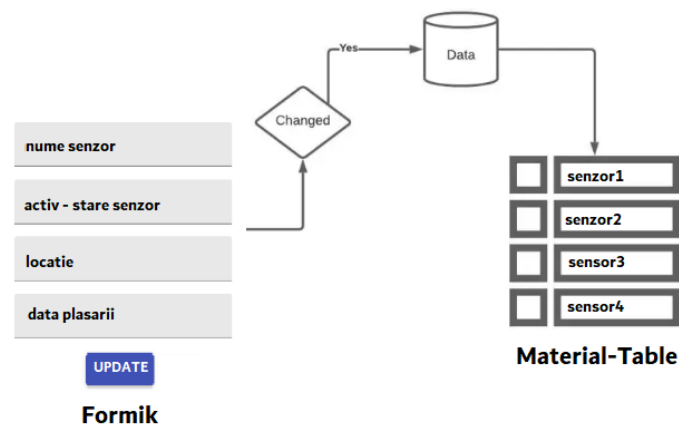


Figura 3.13 Formik și Material-Table

3.4.1.4 Highcharts React

Biblioteca Highcharts React permite integrarea ușoară a graficelor, reprezentând o înaltă performanță și interactivitate pentru vizualizarea datelor. Core (diagrame standard), Stock (diagrame cronologice), Maps (hărți geografice interactive) și Gantt (evenimente de activitate) sunt principalele modalități de a reprezenta grafice utilizând Highcharts.

Valorile ultimelor 12 ore de înregistrare a poluanților gazoși sunt interpretare intra-un mod clar și coerent cu ajutorul Highcharts React. O simpla reprezentare vizuala sa contribui la identificarea tendințelor și relațiilor între date intra-un mod mai intuitiv.

Figura 3.14 reprezintă un model de arhitectura pentru implementarea diagramelor. De obicei, acestea se găsesc în secțiunea “Dashboard” a aplicațiilor Web, în acest caz, harta Google Map.

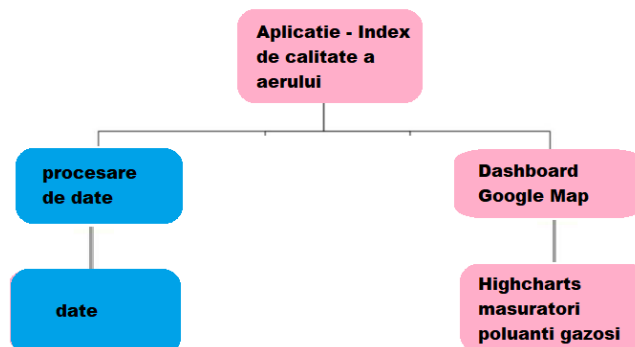


Figura 3.14 Structura de implementare Highcharts

3.4.1.5 Reach UI

În aplicație este implementată o funcționalitate de căutare unor locații pe harta Google Maps. Libraria Reach UI este folosită pentru a crea această funcționalitate și a marca locația dorită. Pentru a obține locațiile pe harta, se va folosi hook-ul „usePlaceAutocomplete”, care oferă geocodul și coordonatele geografice ale locației selectate.

Utilizând această funcționalitate, experiența utilizatorului se va îmbunătăți deoarece are posibilitatea de a găsi rapid locațiile dorite și de verifica dacă în zona selectată există senzori și care sunt valorile măsurătorilor (indicele de calitate, valoarea poluanților gazoși, starea atmosferică).

3.4.4 Firebase Authentication

Funcționalitățile de autentificare și autorizare în aplicațiile web se referă la procesul de identificare a utilizatorilor și controlul accesului lor la resurse și funcționalități. Se va valida identitatea utilizatorilor prin cereri de autentificare și confirmarea datelor de identificare.

Acest serviciu definește permisiuni și restricții pentru fiecare utilizator autentificat, determinând ce resurse și acțiuni pot fi accesate. Aceste funcționalități asigură securitatea datelor și oferă o experiență personalizată și protejată utilizatorilor.

În cadrul proiectului, procesul de autentificare și autorizare este definit de serviciul oferit de Firebase, numit Firebase Authentication cu ajutorul căruia se va gestiona înregistrarea și autentificarea utilizatorilor.

Utilizatorii care folosesc aplicația prezentată în această lucrare vor avea posibilitatea de crearea a unui cont folosind email și parola. Se vor autentifica folosind

aceste credențiale, iar în caz de uitare a parolei, vor putea recupera contul primind un email de resetare a parolei.

Firebase Authentication oferă următoarele principale funcționalități:

- Diverse metode de autentificare folosite de utilizatori: email și parola, autentificarea folosind conturi de rețele sociale (Google, Facebook, GitHub etc), folosind numărul de telefon (SMS) sau autentificarea anonimă.
- Funcții de gestionare a conturilor utilizatorilor, inclusiv crearea de noi conturi, actualizarea datelor utilizatorului, resetarea parolelor și ștergerea conturilor.
- Identificarea utilizatorului autentificat prin procesul de gestionare automată a sesiunilor de autentificare și furnizarea unui token.
- Integrarea ușoară cu alte servicii Firebase, precum Cloud Firestore sau Cloud Functions.

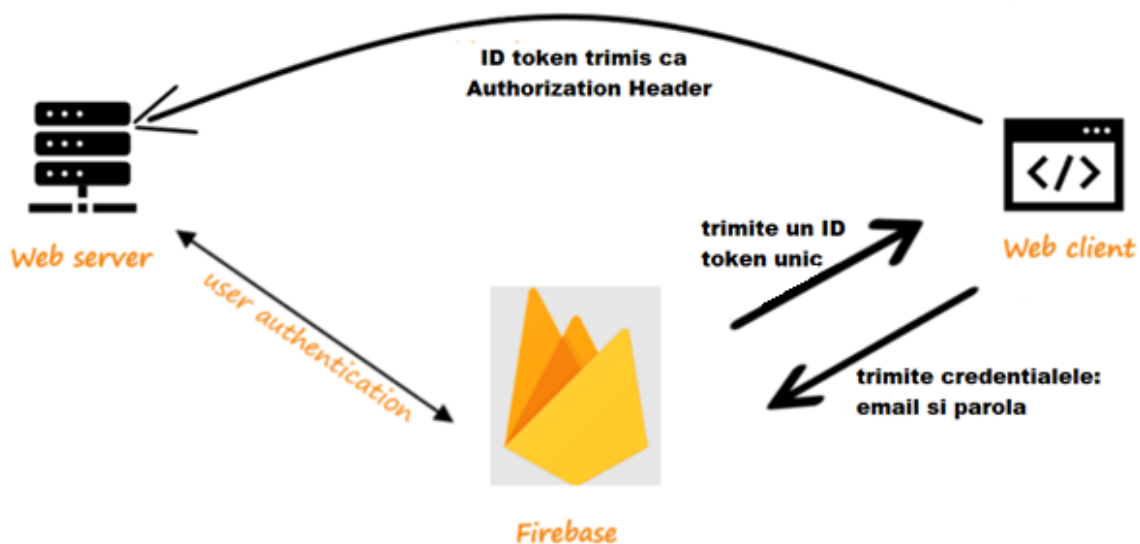


Figura 3.15 Arhitectura Firebase Authentication

Pentru a permite comunicarea cu toate serviciile Firebase de care avem nevoie pe partea de frontend, se va conecta acesta cu serverul prin intermediul Firebase SDK.

Figura 3.15 reprezintă arhitectura Firebase Authentication, iar procesul de autorizare și accesare a datelor utilizatorului este prezentat astfel:

La autentificarea sau înregistrarea utilizatorului, credențialele vor fi trimise către server într-un mod securizat utilizând metodele Firebase SDK: `createUserWithEmailAndPassword` (înregistrare) sau `signInWithEmailAndPassword` (autentificare). Pe partea de server, Firebase validează aceste credențiale și trimite ca răspuns un token de autentificare, iar în acest fel vom accesa datele (email, nume, prenume) în partea de frontend. Având token-ul, îl putem trimite ca header la partea de backend API și pe baza lui datele utilizatorului vor putea fi accesate iar în cele din urmă, aceste vor fi salvate în baza de date Firebase Database în documentele din colecția "users".

4 Implementare

4.1 Configurarea Firebase și Spring Boot

4.1.1 Crearea proiect Firebase

Stocarea datelor se va realiza în baza de date Firestore Database, serviciu de baza de date Google Cloud. Aceasta oferă un sistem de stocare și sincronizare a datelor într-un format JSON organizat pe colecții și documente. Firestore Database oferă capacități de stocare și sincronizare în timp real.

Primul pas în realizarea configurării bazei de date a aplicației, este autentificarea în contul google (sau mai întâi crearea unui nou cont, dacă este cazul) din consola console.firebase.google.com. Se va selecta opțiunea „Create a project” sau “Add project” care va redirecționa utilizatorul către procesul de creare (este necesară doar definirea numelui proiectului și acceptarea termenilor și condițiilor). Odată creat proiectul, se va configura tipul de baza de date dorit („Firestore Database”) din secțiunea „Build”. Ultimii pași în acest proces sunt selectarea modului de producție „product mode” pentru a securiza datele definind reguli de acces și setarea locației proiectului.

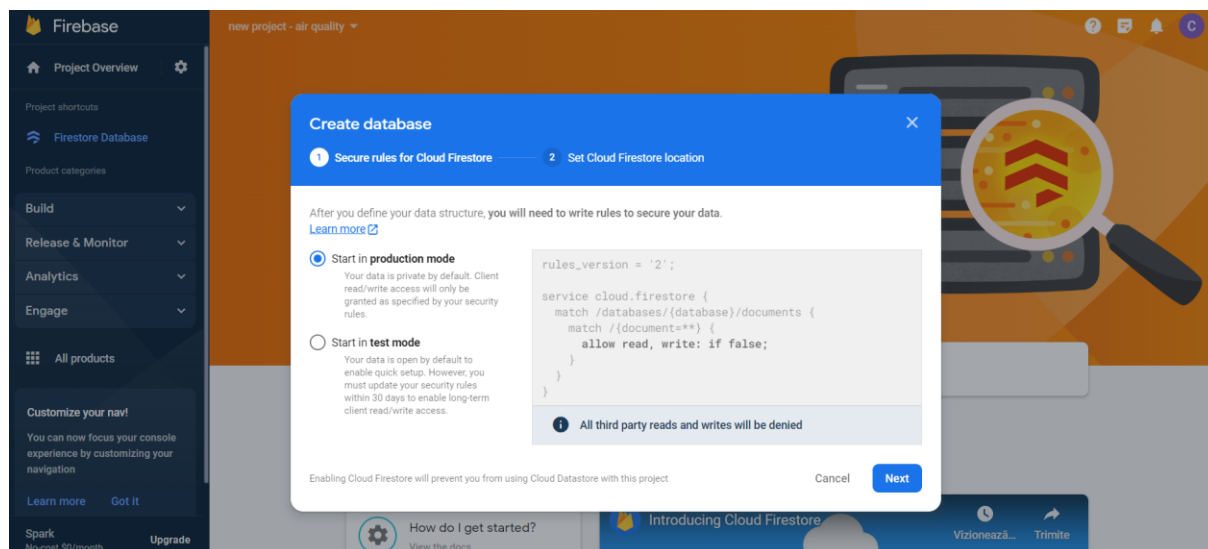


Figura 4.1 Creare proiect nou Firebase Database

4.1.2 Crearea proiect Spring Boot

Pentru a crea partea de backend, am folosit serviciul online, „Spring initializr”, accesat la adresa start.spring.io. Acesta generează schemele unui proiect Spring Boot prin specificarea configurărilor și dependentelor dorite. Din secțiunea „Project”, se va selecta Maven, limbajul de programare Java și versiunea, apoi se vor completa câmpurile care definesc proiectul: Group, Artifact, Name etc și dependențele dorite: Spring Web, Lombok, Firebase Admin etc, care vor fi adăugate în fișierul pom.xml.

Prin confirmarea specificațiilor adăugate, proiectul generat este pregătit pentru dezvoltare, inclusiv structura de directoare și fișiere necesare. Folosind Spring initializr se va simplifica semnificativ procesul de inițializare a unui proiect.

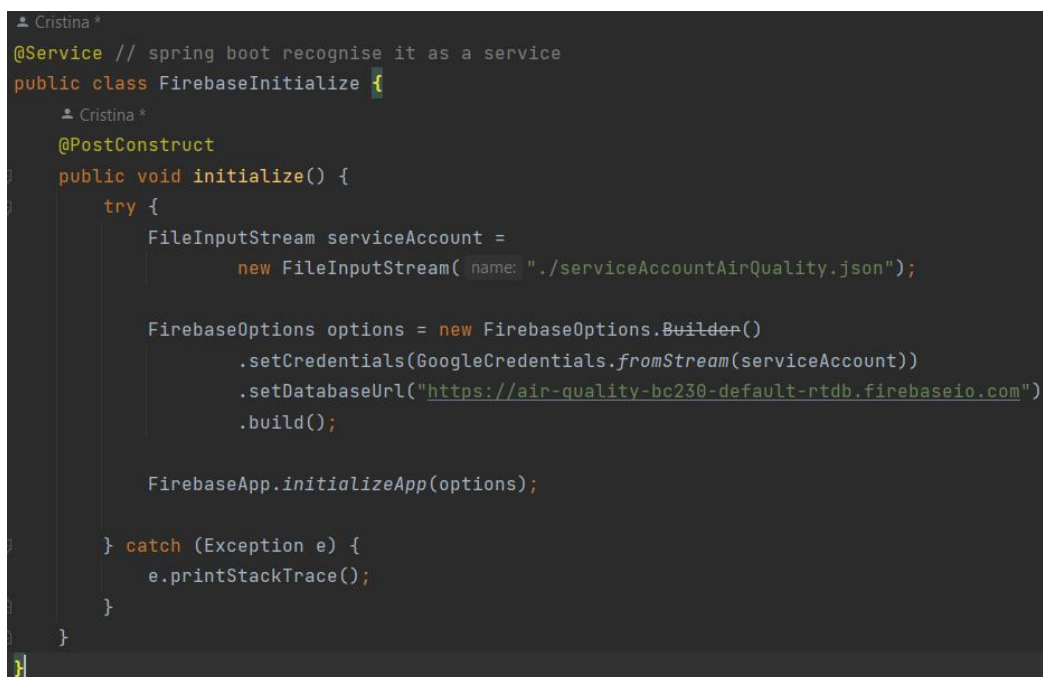
4.1.3 Cheia privata Firebase

Următorul pas consta în autorizarea aplicației locale Spring Boot și conectarea acesteia cu baza de date Firestore Database. Pentru aceasta, se va folosi o cheie secreta care poate fi generata de Firebase prin apăsarea butonului „Generate new private key” (Java) din secțiunea „Project Overview” -> „Service accounts” -> „Database secrets”. Fișierul JSON generat conține date de identificare a proiectului (tip, id-ul, cheia privata și id-ul ei etc). Aceasta este valoarea unica de autentificare utilizata în cererile de acces la baza de date. Pentru a face inițial accesul cheie din codul de server, se va salva scriptul în folder principal al proiectului pentru a putea fi accesat ulterior.

4.1.4 Conexiunea Spring Boot la Firebase Database

Se va crea mai departe clasa „FirebaseInitialize” în pachetul Java „service” pentru a defini și permite conexiunea către Firebase.

Metoda „initialize()” descrie detaliile de conectare folosind cheia generată la pasul anterior, citind mai întâi datele fișierului cu ajutorul obiectului `FileInputStream`. Înainte de a conecta, se vor defini opțiunile firebase, cum ar fi credențialele și URL – ul bazei de date. Deoarece conexiunea este necesara la pornirea aplicației, se va folosi adnotarea `@Service` (Spring Boot va recunoaște aceasta clasa ca fiind un serviciu) și `@PostConstruct`. Pentru a trata posibilele erori din timpul conectării, se folosește blocul „try-catch”.



```
@Service // spring boot recognise it as a service
public class FirebaseInitialize {
    @PostConstruct
    public void initialize() {
        try {
            FileInputStream serviceAccount =
                new FileInputStream( name: "./serviceAccountAirQuality.json");

            FirebaseOptions options = new FirebaseOptions.Builder()
                .setCredentials(GoogleCredentials.fromStream(serviceAccount))
                .setDatabaseUrl("https://air-quality-bc230-default-rtdb.firebaseio.com")
                .build();

            FirebaseApp.initializeApp(options);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Figura 4.2 Conexiunea Spring Boot la Firebase Database

4.2 Structura bazei de date Firestore Database

Odată ce conexiunea Spring Boot și Firebase este realizată, se va defini structura bazei de date. Planificarea colecțiilor și documentelor trebuie structurată în funcție de necesitățile aplicației.

Identificarea entităților principale

Aplicația dispune de parte de autentificare și autorizare și de actualizarea datelor contului, deci este nevoie de o entitate „User” pentru a stoca datele utilizatorilor. Datele de măsurare a indicelui de calitate a aerului sunt definite de către senzorii plasați în diferite locații pe harta, așadar „Sensor” este cea de-a 2-a entitate principală.

Definirea colecțiilor

Colecțiile sunt reprezentate de entitățile principale. Rezulta ca baza de date are 2 colecții: „Users” și „Sensors”.

Definirea documentelor din colecția „users”

Fiecare document din cadrul colecției reprezintă o înregistrare individuală, așadar numele colecției este email-ul userului. Câmpurile documentelor utilizatorilor conțin „email”, „firstName” și „lastName” (toate de tipul string).

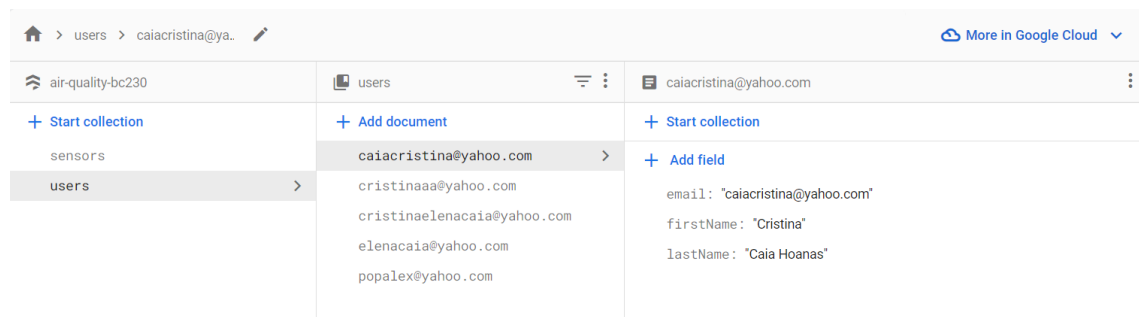


Figura 4.3 Colecția „users” în Firebase Database

Definirea documentelor din colecția „sensors” și a subcolecțiilor „measurements”

Senzorii sunt reprezentați de documentele din cadrul acestei colecții, iar un senzor este definit de următoarele câmpuri: „active”: boolean, „creationDate”: Timestamp Firebase, „location”: GeoPoint Firebase, „measurementsType”: array de tip string, „name”: string și „uuid”: string (acesta este și ID ul documentului).

Fiecare document senzor poate conține o subcolecție „measurements” reprezentând tipurile și valorile măsurătorilor înregistrate. Câmpurile pentru o

măsurătoare sunt: „instantTime”: Timestamp Firebase, „type”: string, „unit”: string, „value”: double.

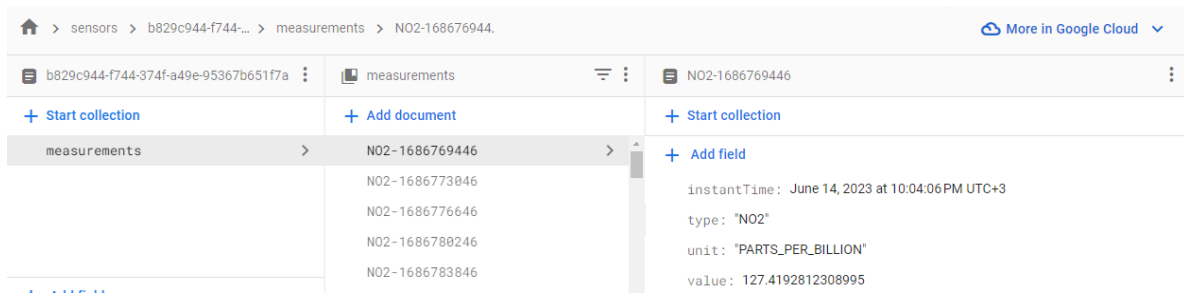


Figura 4.4 Subcolecția "measurements" în Firebase Database

ID-ul documentelor măsurătorilor reprezintă tipul măsurătorii (NO2) și timestamp-ul (secundele acestuia) la care s-a creat acestui document.

4.3 Structura Spring Boot

Clasele „SensorService” și „FirebaseService” din pachetul Java service se vor adnota cu @Service, deci vor fi considerate servicii de Spring Boot. Aici se va inițializa și dezvolta logica de implementare a aplicației și vor defini metodele de PUT, GET, POST și DELETE pentru manipularea datelor în Firestore.

Pentru a obține conexiunea la baza de date definită în **Conexiunea Spring Boot la Firebase Database**, se va folosi metoda „getFirestore()” din „FirestoreClient” astfel: *FirestoreClient.getFirestore()*. Fiecare funcție definită în acest pachet va folosi aceasta conexiune.

4.3.1 Spring Boot Service

Datele manipulate în aplicație sunt reprezentate de entități. Acestea sunt salvate sub formă de clase în pachetul “entities” după cum urmează: “Sensor” pentru senzori, “Measurement” pentru măsurătorile înregistrate de senzor, “User” pentru utilizatori și “airQualityIndexWithType” pentru indicii de calitate a aerului.

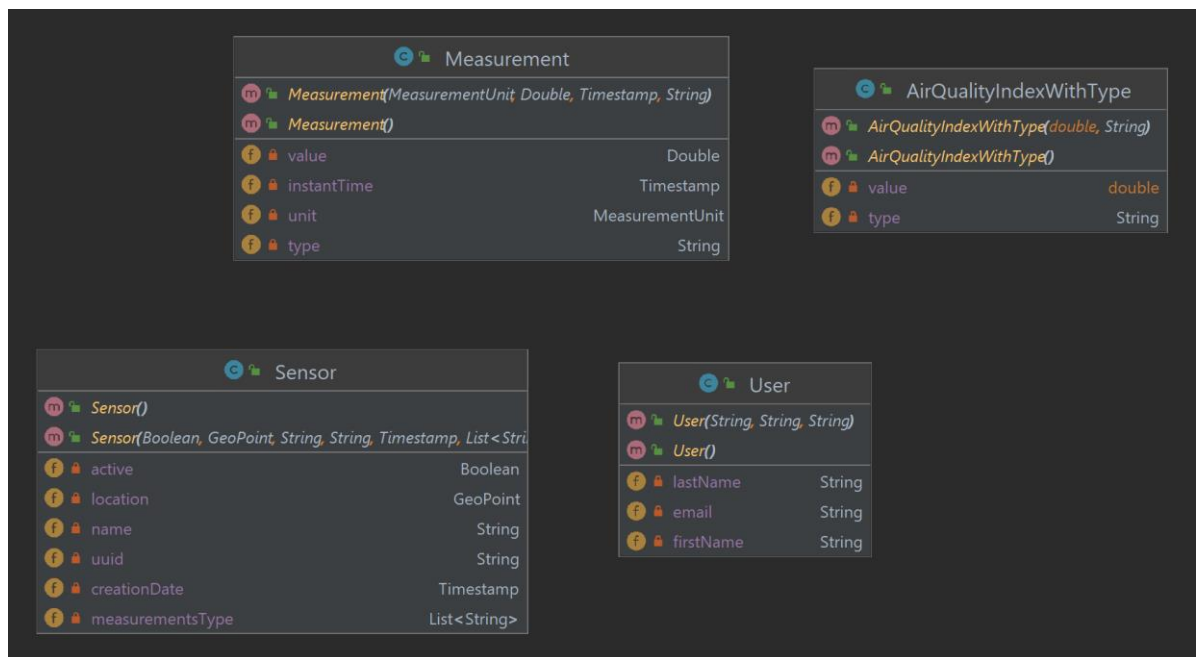


Figura 4.5 Diagrama claselor pentru entitățile aplicației

Clasa SensorService implementează logica pentru entitățile senzorilor, principalele metode fiind cele operațiile de bază reprezentate de CRUD (Create, Read, Update, Delete). [31]

Create - crearea unor noi înregistrări în baza de date

Read – citirea informațiilor existente din baza de date

Update – actualizarea datelor existente în baza de date

Delete – ștergerea înregistrărilor existente din baza de date

Pentru a putea scrie, șterge, obține și actualiza datele unui senzor, se va obține referința documentul senzorului respectiv din colecția „sensors” astfel:

- obținem colecția „sensors”:

```
sensorsCollectionReference = getFirestore().collection(„sensors”)
```

- obținem documentul cu ID unic din colecție:

```
sensorsCollectionReference().document(documentId)
```

```

3 usages  Cristina
public SensorDTO createSensor(SensorDTO sensorDTO) throws ExecutionException, InterruptedException {
    final ZonedDateTime zonedDateTime = ZonedDateTime.now();
    final UUID uuid = UUID.nameUUIDFromBytes(zonedDateTime.toString().getBytes());
    sensorDTO.setUuid(uuid.toString());
    sensorDTO.setCreationDate(LocalDate.now());

    final Sensor sensor = entityConverter.convert(sensorDTO);
    sensor.setCreationDate(Timestamp.now());

    ApiFuture<WriteResult> sensorDocument = getFirestore().collection(SENSOR_COLLECTION)
        .document(sensor.getUuid())
        .create(sensor);

    final String timestamp = sensorDocument.get().getUpdateTime().toString();
    final String logMessage = String.format("[%s] Sensor with uuid: %s was successfully created!", timestamp, sensor.getUuid());
    Log.info(logMessage);

    return dtoConverter.convert(sensor);
}

```

Figura 4.6 Crearea unui document senzor

Figura 4.6 reprezintă crearea unui document senzor în Firestore. Câmpul „uuid” reprezintă identificadorul unic generat pe baza valorii obținute din data și ora curenta, transformate în conținutul șirului de octeți.

Metoda primește ca parametru un senzor DTO, deci se va face conversia în senzor pentru a crea aceasta entitate în baza de date, unde câmpul „creationDate” reprezintă momentul scrierii.

Se va crea documentul folosind metoda „create(sensor)”, unde obiectul „ApiFuture<WriteResult>” reprezintă rezultatul viitor al unei operații de scriere în Firestore Database.

Tabelul reprezintă metodele CRUD folosite de Spring Boot pentru manipularea a datelor în Firebase în mod general.

Tabel 4.1 CRUD în Spring Boot și corespondențele în Firebase

Metoda HTTP	Metoda Spring Boot	Rezultat asincron Firebase
POST	create()	ApiFuture<WriteResult>
GET	get()	ApiFuture<QuerySnapshot>
PUT	setName()	ApiFuture<DocumentSnapshot>
DELETE	delete()	ApiFuture<WriteResult>

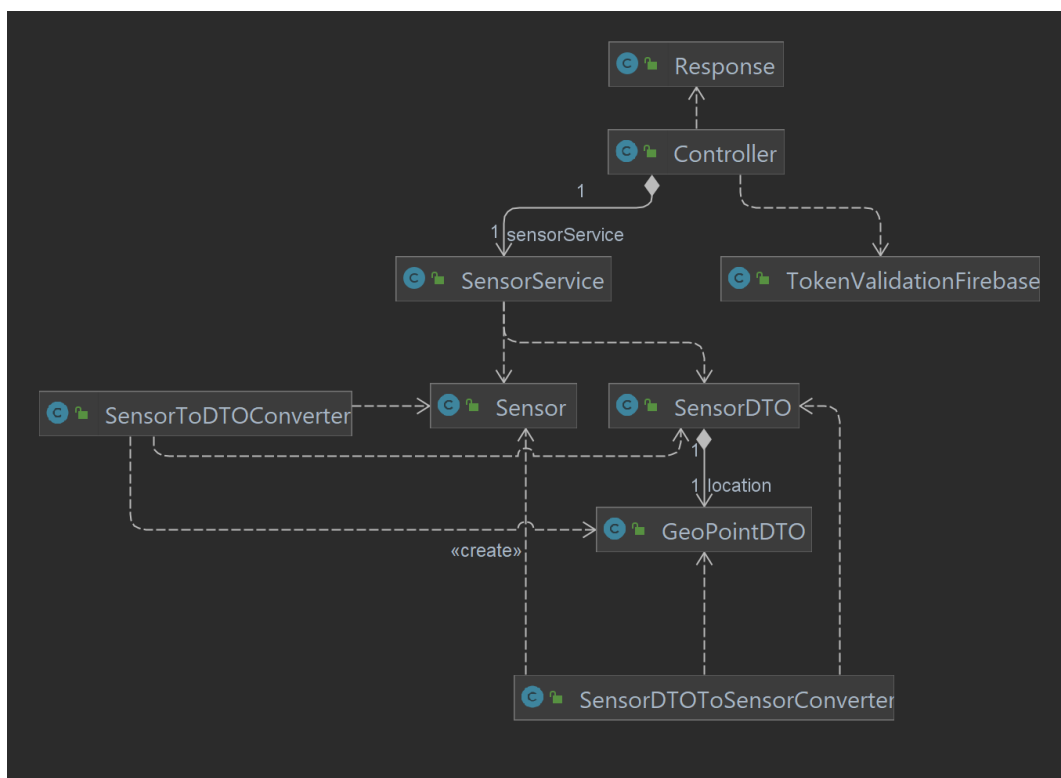


Figura 4.7 Diagrama claselor pentru logica din SensorService

Data Transfer Object – SensorDTO

Clasele SensorDTOToSensroConverter și SensorToDTOConverter implementează metodele “convertInternal(source, target)” pentru a face conversia din DTO în entitate și viceversa.

Firebase Database lucrează cu obiecte entității senzor, iar serviciul Spring Boot folosește obiecte DTO senzor. Metodele “convert” ale acestor clase se vor apela în “SensorService” pentru a realiza conversia entitate -> DTO.

Câmpul “location” a senzorului este de tip GeoPoint Firebase, obiect cu field-urile latitudine și longitudine, deci se va crea un DTO și pentru locație.

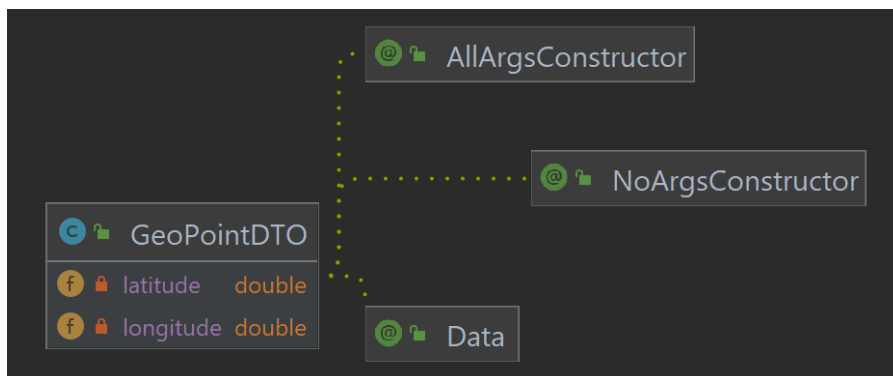


Figura 4.8 Obiectul Java GeoPointDTO

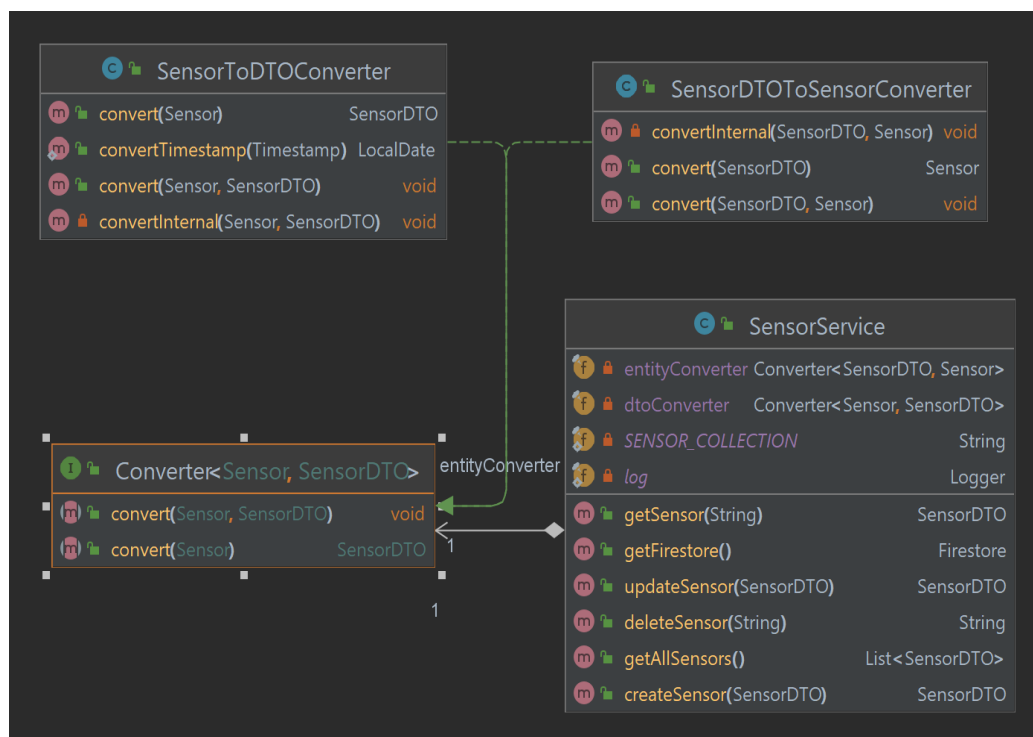


Figura 4.9 Diagrama implementării SensorDTO

Clasa FirebaseService implementează în principal metodele de CRUD pentru utilizatorii aplicației, măsurătorile senzorilor și metoda de obținere a indicelui de calitate.

Aplicația pune la dispoziție utilizatorilor funcționalități precum înregistrare pentru un nou cont, autentificare în contul existent și resetare parola. Pentru aceasta, se va implementa autentificarea și autorizarea.

Pentru crearea unui unei măsurătoare, metoda primește ca parametrii ID-ul senzorului și obiectul de tip măsurătoare. Mai întâi, se va verifica dacă ID-ul senzorului trimis ca parametru exista și dacă senzorul corelat ID-ului are definit acest tip de măsurătoare. Se va scrie o noua măsurătoare în baza de date, reprezentând un document cu ID-ul unic.

Pentru a obține ultimele măsurători - `getLastMeasurementsOfLastHour(seznor ID)`, se vor lua ultimele documente a fiecărui tip de măsurătoare: PM2.5, PM10, NO2, SO2, O3, temperatura, umiditate, presiune – în funcție de existența, iar acestea se vor filtra astfel încât să rămână doar acele înregistrări din ultima ora (acelea a cărui timestamp este maxim).

Valoarea indicelui de calitate a aerului este obținută din valorile poluanților gazoși (PM2.5, PM10, NO2, SO2, O3) înregistrați în ultima ora (folosind metoda `getLastMeasurementsOfLastHour`). Mai întâi se va verifica dacă valorile minime necesare (PM2.5 și NO2) exista, apoi se va calcula maximul dintre înregistrările obținute. Acest rezultat reprezintă indicele de calitate a aerului pentru senzorul respectiv. Metoda `airQualityIndex` returnează indicele și poluantul definitor (cel a cărui valoare este maxima). Aceste informații se vor folosi în partea de front-end corespunzător.

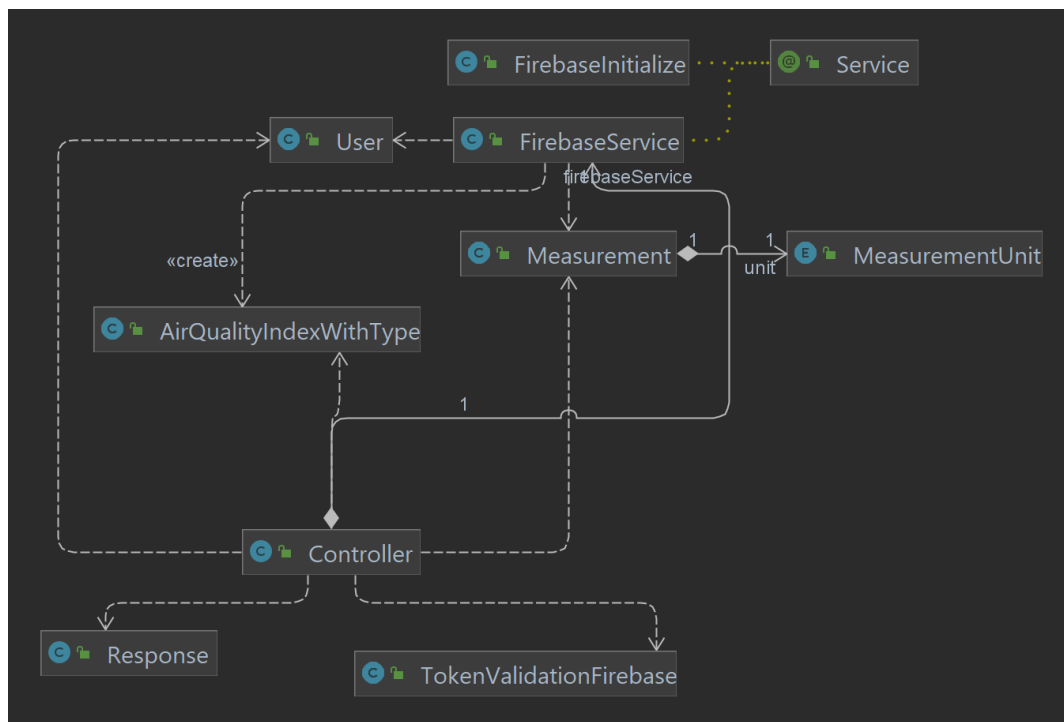


Figura 4.10 Diagrama claselor pentru logica din FirebaseService

4.3.2 Securizarea datelor Spring Boot cu Cloud Firebase

Securizarea datelor în Spring Boot consta în verificarea token-ului trimis din frontend. Se va crea în pachetul „auth” o clasa denumita „TokenValidationFirebase”, care va verifica tokenul trimis ca parametru prin header în metodele care se doresc a fi securizate. FirebaseAuth este o clasa din pachetul com.google.firebase.auth.

```

4 usages  Cristina
public class TokenValidationFirebase {
    3 usages  Cristina
    public static FirebaseToken validateToken(String idToken) throws FirebaseAuthException {
        return FirebaseAuth.getInstance().verifyIdToken(idToken);
    }
}

```

Figura 4.11 Metoda de validare a tokenului

Senzorii și datele acestora sunt securizate și pentru a fi livrate este nevoie de validarea tokenului. Metoda “validateToken” verifică datele utilizatorului și permisiunile acestuia. Dacă tokenul este valid, se va putea accesa permisiunea la date, iar senzorii vor putea fi vizualizați pe harta. Metoda de mai jos este implementată în clasa “Controller”.

```
Cristina *
@GetMapping("/sensor/list") // pe harta
public List<SensorDTO> getSensors(@RequestHeader String idToken) {
    try {
        TokenValidationFirebase.validateToken(idToken);
        return sensorService.getAllSensors();
    } catch (FirebaseAuthException e) {
        return null;
    }
}
```

Figura 4.12 Validarea tokenului pentru afișarea senzorilor pe harta

4.3.3 Controller

Clasa controller din pachetul „controller” apelează metodele din clasele service pentru a trata solicitările primite de la client (frontend). Aici se vor specifica rutele și metodele HTTP (GET – getUser, POST – createMeasurementForSpecificSensor, PUT – updateSensor, DELETE – deleteSensorByUUID) și se va verifica token-ul pentru validarea cererilor utilizatorului.

4.3.4 Utilizatorii

Utilizatorii sunt creați în baza de date la crearea unui nou cont. Datele unui utilizator se pot actualiza, iar un user se poate obține cu adresa de email.

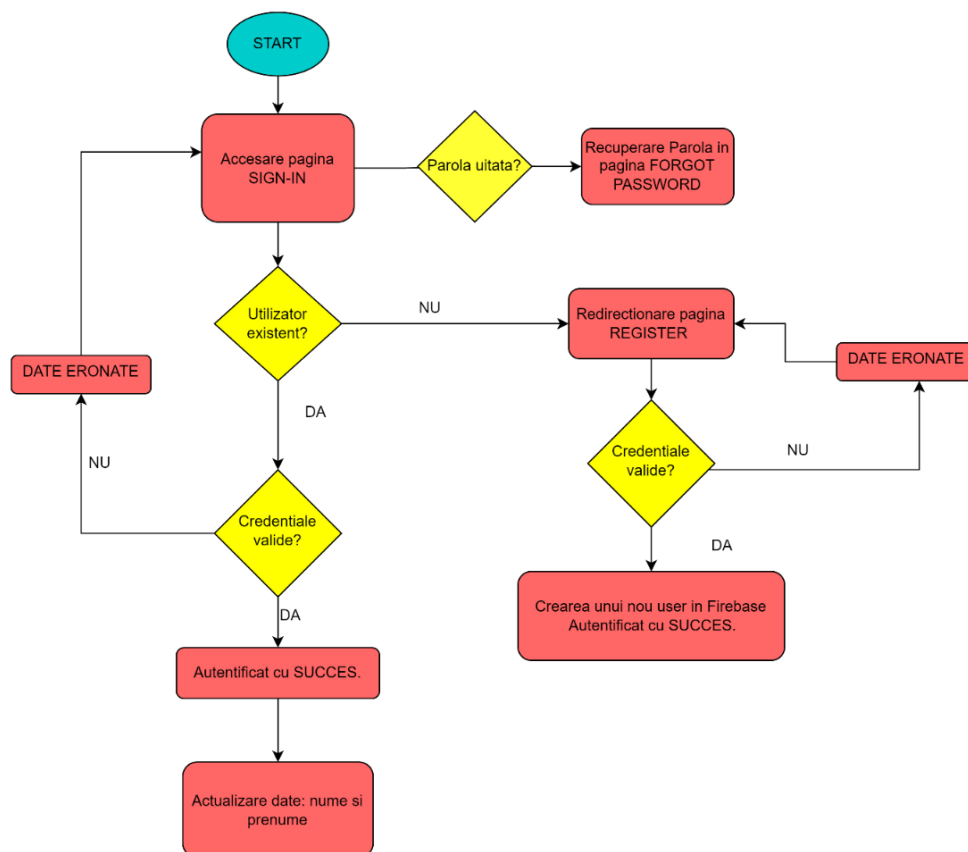


Figura 4.13 Diagrama Flow-Chart pentru autentificarea userilor în aplicatie

4.3.5 CORS

Se vor adăuga configurări pentru CORS (Cross-Origin Resource Sharing) utilizând metoda `addCorsMapping`. Folosind CORS, se va permite accesul front-end la resursele din backend, cat și utilizarea oricăror metode HTTP.

4.3.6 Generarea de senzor și măsurători.

Generarea măsurătorilor in lucrarea prezentata este importanta pentru a se evalua performanta, a se optimiza resursele, acestea reprezentând monitorizarea aplicației. Aceste performante oferă date concrete in ceea ce privește timpul de răspuns și utilizarea resurselor, fiind utilizate pentru o îmbunătățire continua a funcționalităților. Se vor identifica problemele si optimiza performantele in scopul de a oferi o experiența mai buna utilizatorilor.

Metoda de generare a datelor va genera atât senzorul, cat și măsurătorile specifice acestuia. Logica este implementata astfel încât măsurătorile sa fie create la un interval orar. Se pot genera oricâte măsurători este nevoie. Se va trimite o cerere HTTP folosind Postman pentru a apela metoda.

4.4 Structura React JS

Partea de frontend dezvoltată în React JS este structurată după cum se observă în Figura, sub forma mai multor fișiere, organizate în mai multe foldere.

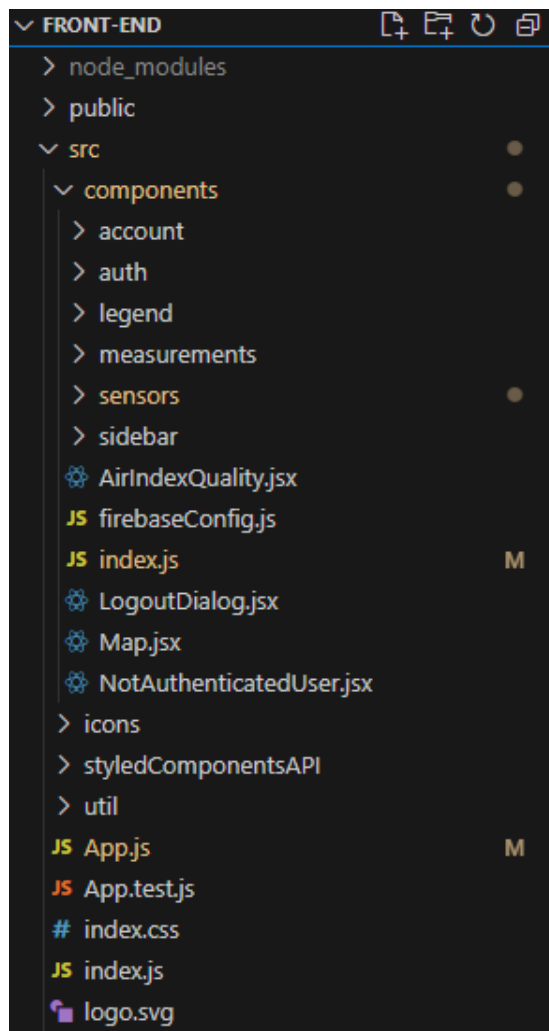


Figura 4.14 Structura frontend

4.4.1 AXIOS

Cererile HTTP în mediul de browser se efectuează cu metodele și end-pointurile din biblioteca „axios” din JavaScript.

Tabel 4.2 Metodele axios în React JS și endpoint-urile corespunzătoare

Metoda AXIOS	Endpoint
axios.post	api/createSensor
axios.get	api/getSensorByUUID
axios.put	api/updateSensor
axios.delete	api/deleteSensorByUUID

4.4.2 Firebase Authentication cu React JS și utilizatorii în Firebase Database

Pentru a configura autentificarea pe partea de client, mai întâi se va instala dependența „firebase” folosind comanda „npm install firebase” în folderul principal al proiectului. Mai apoi, se va crea un fișier în care se va defini obiectul care conține configurația necesară pentru inițializarea Firebase.



```
import "firebase/auth";

const firebaseConfig = {
  apiKey: "AIzaSyB...",
  authDomain: "air-quality-bc230.firebaseio.com",
  projectId: "air-quality-bc230",
};

const app = initializeApp(firebaseConfig);
export const auth = getAuth(app);
```

Figura 4.15 Configurare Firebase

Se folosesc 3 metode pentru funcționalitatea de autentificare în partea de frontend. React JS implementează metodele de înregistrare user nou, autentificare în contul deja existent și resetare parolă (Firebase trimite un form pe email pentru resetarea parolei). Acestea sunt folosite în cele 2 pagini: RegisterForm, LoginForm și ForgotPasswordForm. Se folosește formik pentru a crea formul, valida datele introduse în campurile „firstName”, „lastName”, „email”. Acestea, cu ajutorul metodei „createUserWithEmailAndPassword” vor fi trimise mai departe pentru a fi create în consola Firebase Authentication, dar și la server, pentru a le salva în baza de date.

Atunci când un utilizator se va autentifica cu credențialele existente, se stochează tokenul de validare în localStorage și se va redirecționa către pagina principală, reprezentată de harta Google Map.

Dacă utilizatorul nu este autentificat și va dori să navigheze pe una dintre paginile aplicației schimbând URL-ul, se va afișa o pagină ce oferă opțiunile de autentificare, înregistrare sau recuperare cont.

Utilizatorul se poate deconecta prin implementarea funcției „signOut” din „firebase/auth” cu parametrul „auth” din aceeași bibliotecă.

```

    },
    onSubmit: (values, { setSubmitting, setErrors }) => {      Blame Crist
      signInWithEmailAndPassword(auth, values.email, values.password)
        .then((userCredential) => {
          userCredential.user.getIdToken().then((token) => {
            localStorage.setItem("accessToken", JSON.stringify(token));
          });
          navigate("/map");
        })
        .catch((error) => {
          const errorCode = error.code;
          const errorMessage = error.message;
          if (errorCode === "auth/wrong-password") {
            setErrors({ password: "Invalid password" });
          } else if (errorCode === "auth/user-not-found") {
            setErrors({ email: "User not found" });
          } else {
            console.log(errorCode, errorMessage);
          }
        })
        .finally(() => {
          setSubmitting(false);
        });
    });
  });

```

Figura 4.16 Funcția *signInUserWithEmailAndPassword*

Numele și prenumele userului pot fi modificate în secțiunea “Account” a aplicației. Acest formular folosește Formik, făcând mai întâi o cerere de tip “get” către backend pentru a primi utilizatorul current, apoi va trimite datele modificate sub formatului unui obiect.

4.4.3 Harta Google Maps cu senzorii și campul de cautare pentru locatii

Pentru a afișa harta din biblioteca „react-google-maps/api” pe pagina “Map”, se definește hook-ul “useLoadScript” format din cheia API. Se personalizează pinul reprezentat de componenta “Marker”, cat și “InfoWindow” (modalul ce se deschide la apăsarea pe un pin). Pentru a vedea toți senzorii, se face o cerere “get” către backend și acesta verifica mai întâi tokenul de validare trimis prin header din (tokenul este valabil în “localStorage”. Dacă utilizatorul este autentificat, primește datele și acestea se stochează într-un hook “useState” denumit “sensors”. Toata această logica se va implementa într-un hook useEffect, care va afișa datele cerute doar o singură data, la prima randare a paginii.

Câmpul de căutare este definit în componenta “AutocompletePlaces”, care folosește hook-ul usePlacesAutocomplete() pentru ca atunci cand userul caută un loc, să poată vedea sugestii. La selectarea locului dorit căutat, “useState” va seta latitudinea și longitudinea acestuia, iar punctul central al hărții se va muta acolo.

4.4.4 Indicele de calitate a aerului și măsurătorile

La apăsarea pe pinul reprezentat de senzor, se va afișa în cadrul unei ferestre indicele de calitate a aerului, data în care acesta a fost măsurat, ultimele măsurători și un buton care permite redirectionarea către vizualizarea diagramelor măsurătorilor pentru ultimele 12 ore. Dacă senzorul devine inactiv, datele acestuia în modalul de pe hartă nu vor mai putea fi vizualizate.

Pentru definirea culorii și încadrarea indicelui într-un status, se implementează o componentă numită "SeverityPill". Aceasta are ca referință tabelul cu intervalele și în funcție de valoarea și măsurătoarea care definește valoarea, se va genera o culoare. Statusul este primit de la server atunci când se face o cerere de tip GET către metoda de calculare a indexului. Aceasta returnează valoarea și poluantul definitor.

Biblioteca Highcharts folosește "chartOptions" pentru a defini opțiunile diagramelor. Acestea sunt: tipul de diagramă ("line" și "column"), datele măsurătorilor, data în care s-a făcut ultima măsurătoare, legenda, unitatea de măsură, tooltip (pentru a vizualiza măsurătorile pe ultimele ore) și mesaj în cazul în care nu există înregistrări. Datele sunt primite în urma unei cereri "get" și se vor afișa sub formă de linie și coloana, iar ultima valoare va fi de o culoare mai evidențiată.

```
<Grid item>
  <ParentPaper>
    <SpaceBetweenGrid marginBottom={1}>
      <KPITitleTypography>Humidity</KPITitleTypography>
      <TypographyHealthKitAndDate>
        Weather data
      </TypographyHealthKitAndDate>
    </SpaceBetweenGrid>
    <SpaceBetweenGrid marginBottom={1}>
      <KPITypography color={"blue"}>Hourly</KPITypography>
      <TypographyHealthKitAndDate>
        {humidityTimestamp.length > 0
          ? lastDateTimestamp(humidityTimestamp[0])
            : "-"}
      </TypographyHealthKitAndDate>
    </SpaceBetweenGrid>
    <HighchartsReact
      highcharts={Highcharts}
      options={chartOptions(
        humidityValues.reverse(),
        "#5637BC",
        "column",
        "%"}
    >
  </ParentPaper>
</Grid>
```

Tabel 4.17 Codul pentru randarea diagramei cu datele de umiditate

4.4.5 Sidebar – meniul din lateralul paginii și dimensiunea paginilor adaptivă

“Drawer” este o componenta din biblioteca “mui/material” și are dimensiunile adaptive ecranelor. Se folosește `useState` pentru interschimbarea mărimilor și o dimensiune exactă a componentei. Acesta oferă navigarea către harta, lista de senzori, formularul de adăugare a unui senzor, detaliile contului și delogarea din cont.

Mai mult, fiecare pagina este implementată în așa fel încât aceasta să fie adaptivă la diferite dimensiuni. Principala componentă care permite dimensionarea conținutului este “Grid”.

```
<Grid item md={8} sm={12} xs={12}>  
  {props.content}  
</Grid>
```

Figura 4.18 Dimensionare conținut în pagina

4.4.6 Tabelul senzorilor și modificarea datelor acestora. Adăugarea unui senzor și ștergerea

Senzorii pot fi de asemenea vizualizați și într-un tabel “MaterialTable” din biblioteca “material-table”. Acesta are 2 coloane: numele senzorului și data plasării lui în locație. La apăsarea unui rând, se va redirectiona către pagina de editare a datelor senzorului. Aici se pot modifica datele (nume și stare: active sau inactiv) prin apăsarea butonului de “Update sensor” în formularul Formik. Tipurile de măsuratori nu pot fi modificate, acestea reprezentând specificațiile unice ale senzorului.

Senzorul se poate șterge, iar la apăsarea butonului pentru a face o cerere de delete către server, va apărea un modal numit “Dialog” pentru confirmarea ștergerii.

Adăugarea unui senzor se realizează printr-o metoda de tip post. Se va verifica mai întâi tokenul și dacă utilizatorul este conectat în cont, după care se va trimite obiectul senzor la server. Latitudinea și longitudinea sunt capuri obligatorii de definite în schema de validare Yup Formik. Dacă senzorul este adăugat cu succes, se va afișa pe ecran un mesaj corespunzător.

4.4.7 Gestionarea rutărilor

Componentele din biblioteca “react-router” gestionează rutările (componenta definitoare a modului de afișare și manipulare a diferitelor pagini ale aplicației) și acestea depind de URL-ul din browser. În componenta “App”, “Routes” conține o serie de componente “Route”, fiecare cu un “path” (URL-ul asociat) și o proprietate “element”, reprezentând componenta care trebuie afișată atunci când URL-ul se potrivește cu ruta respectivă.

```

<Routes> Blame Cristina (25 days ago)
<Route path="/signin" element={<LoginForm />} />
<Route path="/register" element={<RegisterForm />} />
<Route path="/forgotPassword" element={<ForgotPasswordForm />} />

<Route path="/account" element={<Account />} />
<Route path="/map" element={<Map />} />
<Route path="/legendExplanations" element={<LegendExplained />} />
<Route
  path={`map/measurementsChart`}
  element={<SensorMeasurementsCharts />}
/>
<Route path="/sensors" element={<SensorsTableContent />} />
<Route path={`sensors/sensorDetails`} element={<SensorDetailsPage />} />
<Route path="/addSensor" element={<AddSensorContentPage />} />
</Routes>

```

Figura 4.19 Rutarea paginilor aplicatiei

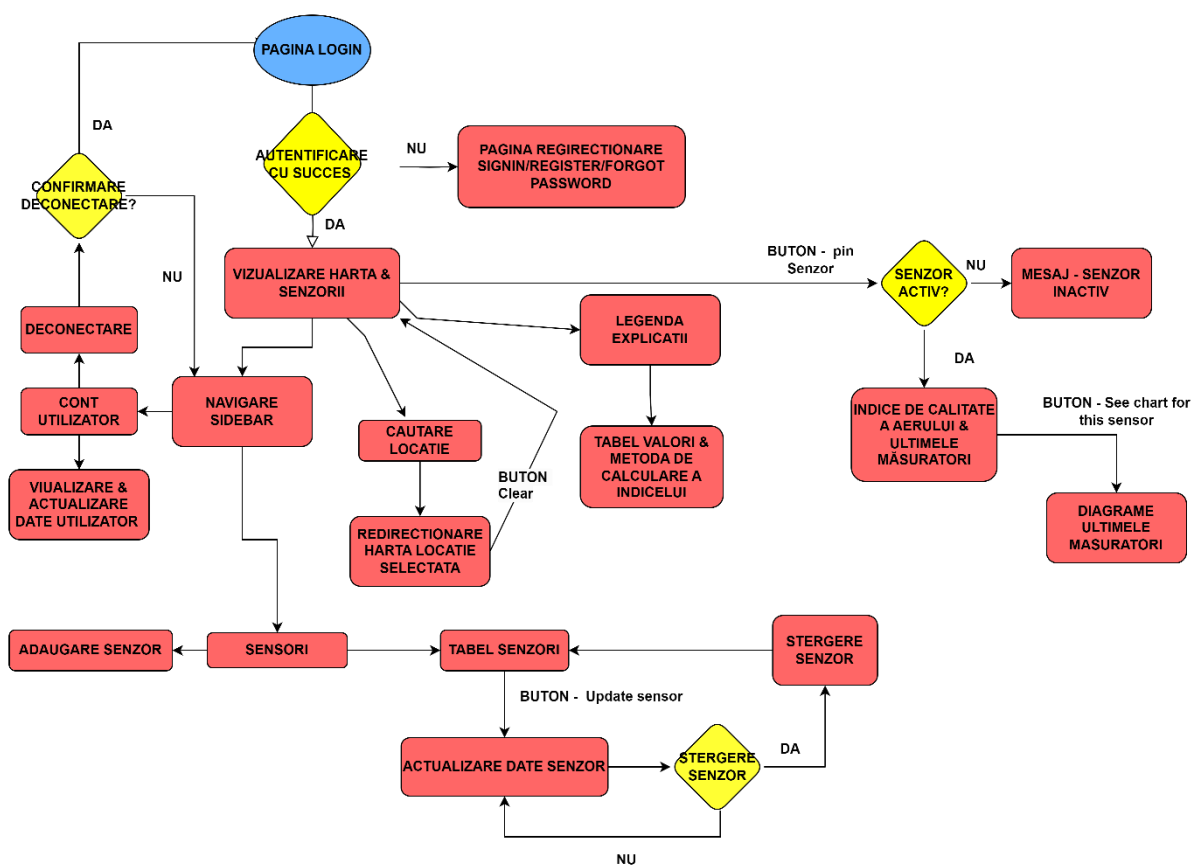


Figura 4.20 Diagrama flow-chart pentru funcționalitățile aplicației

5 Testare și rezultate

5.1 Mod de utilizare

5.1.1 Set-up backend și frontend

Backend – cerințe sistem - instalarea următoarelor

1. Java Development Kit (JDK) 8 sau versiune mai recentă
2. Maven pentru gestionarea dependentelor
3. Editor de cod (IDE) – exemplu: IntelliJ IDEA sau Eclipse

Frontend – cerințe de sistem

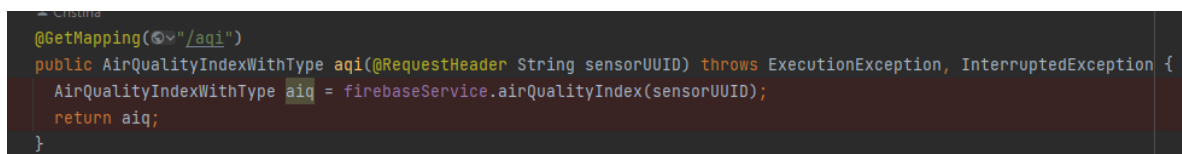
1. Instalare Node.js
2. Instalare NPM (Node Package Manager) sau yarn
3. Rularea comenzii „npm install” în terminalul IDE (Visual Studio Code, de exemplu) sau „yarn install” pentru a instala dependențele proiectului (acestea vor fi actualizate în scriptul package.json)
4. Rularea aplicației folosind comanda „npm run start” sau „yarn start”

5.2 Testare

5.2.1 Testare Postman pentru serverul Spring Boot

Platforma Postman pune la dispoziție dezvoltarea și testarea API-urilor, având un mediu simplu și puternic pentru a trimite cereri HTTP către API-ul serverului și a primi răspunsuri de la acesta.

Pentru a testa o metoda de tip GET cu scopul de a obține indicele de calitate a aerului, setăm id-ul senzorului („sensorUUID”) în header și introducem adresa URL corespunzătoare, așa cum este specificat în clasa Controller.



```
@GetMapping("/aqi")
public AirQualityIndexWithType aqi(@RequestHeader String sensorUUID) throws ExecutionException, InterruptedException {
    AirQualityIndexWithType aiq = firebaseService.airQualityIndex(sensorUUID);
    return aiq;
}
```

Figura 5.1 Metoda GET aqi în Controller

Răspunsul acestei cereri este “200 OK”, ceea ce înseamnă că indicele de calitate a aerului este calculat, rezultatul fiind sub formă JSON, unde “value” reprezintă valoarea indicelui, iar “type” este tipul de măsurătoare pe baza căruia s-a calculat acest indice.

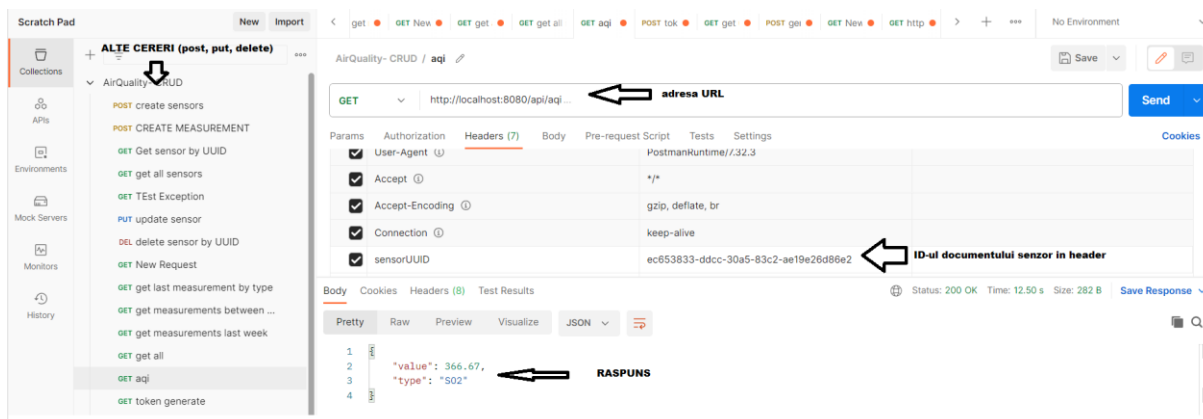


Figura 5.2 Cerere de tip GET în Postman

5.2.2 Testare manuala pentru functionalitatile aplicatiei; Rezultate

Mai jos se va descrie testarea aplicației, aceasta reprezentând fluxul funcționalității aplicației, dar și un manual de utilizare pentru utilizatori. Scopul testării manuale este în principal pentru o asigurare a faptului ca și partea de front-end funcționează conform așteptărilor. Aceasta se va efectua constant de către dezvoltator, pe tot parcursul implementării.

5.2.2.1 Testare înregistrare, logare și recuperare cont

Oricine dorește să utilizeze aplicația, va avea dreptul sa își creeze un cont în aplicație, sa se înregistreze cu un cont deja existent sau sa își recupereze contul prin resetarea parolei.

Utilizatorul va introduce numele, prenumele și o adresa de email existenta și valida pentru a se putea înregistra.

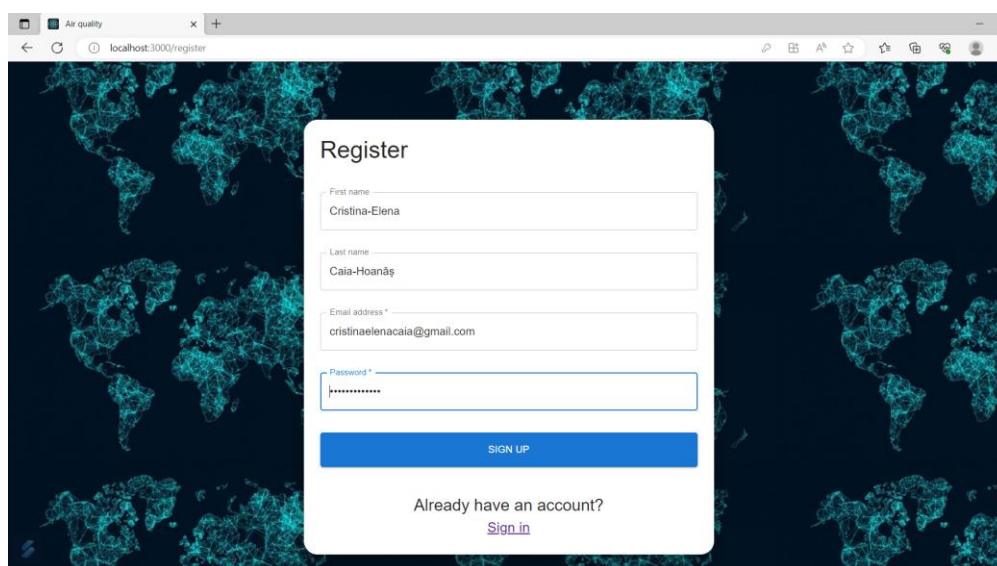


Figura 5.3 Pagina de creare cont utilizator

Odată ce datele introduse sunt valide (adresa de email obligatorie, valida, nu mai exista alt utilizator cu accesai adresa de email, parola obligatorie, conține minim 6 caractere), se va crea un nou utilizator atât în Firebase Authentication, cat și în Firebase Database (sub forma de document, în colecția “users” – vezi secțiunea 4.2).

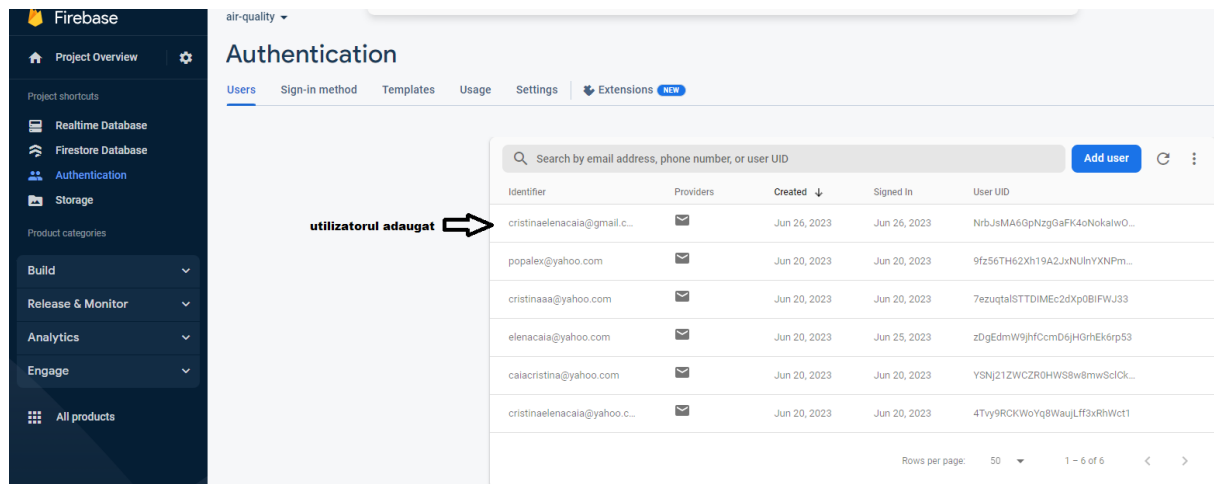


Figura 5.4 Noul utilizator adăugat în firebase Authentication

Daca un utilizator existent nu-și mai amintește parola, acesta își poate recupera contul. Dupa completarea emailului în pagina de “Forgot password”, va primi un email cu ajutorul căruia își va putea reseta parola.

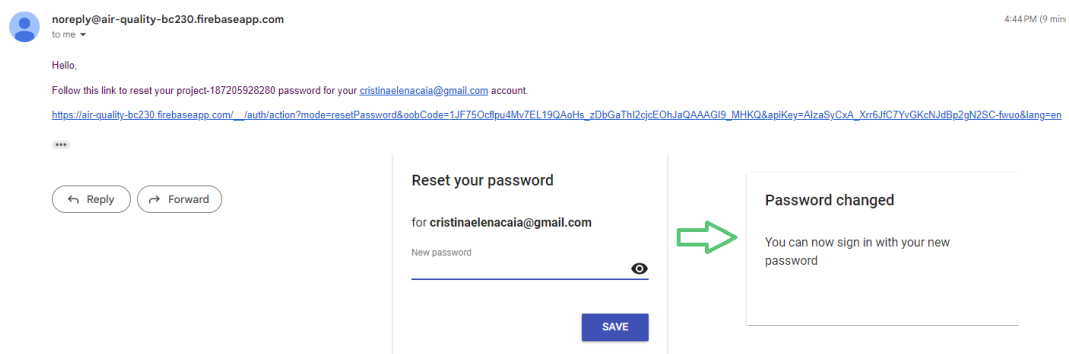


Figura 5.5 Procesul de recuperare cont prin resetarea parolei

Mai departe, utilizatorul se va putea loga în aplicație, iar prima pagină care apare este cea cu harta. Acolo va putea fie naviga prin pagini cu ajutorului meniului “Sidebar”, fie va studia senzorii de pe hartă care redau indicele de calitate din locația unde este senzorul, valorile poluanților, diagramele pentru ultimele 12 ore de măsurare. În pagina “Legend explanations” se găsește tabelul cu valorile recomandate de către Organizația Mondială a Sănătății și intervalele de feedback, cu mesajele de sănătate în funcție de valoarea indicelui de calitate și metoda de calcul a acestuia.

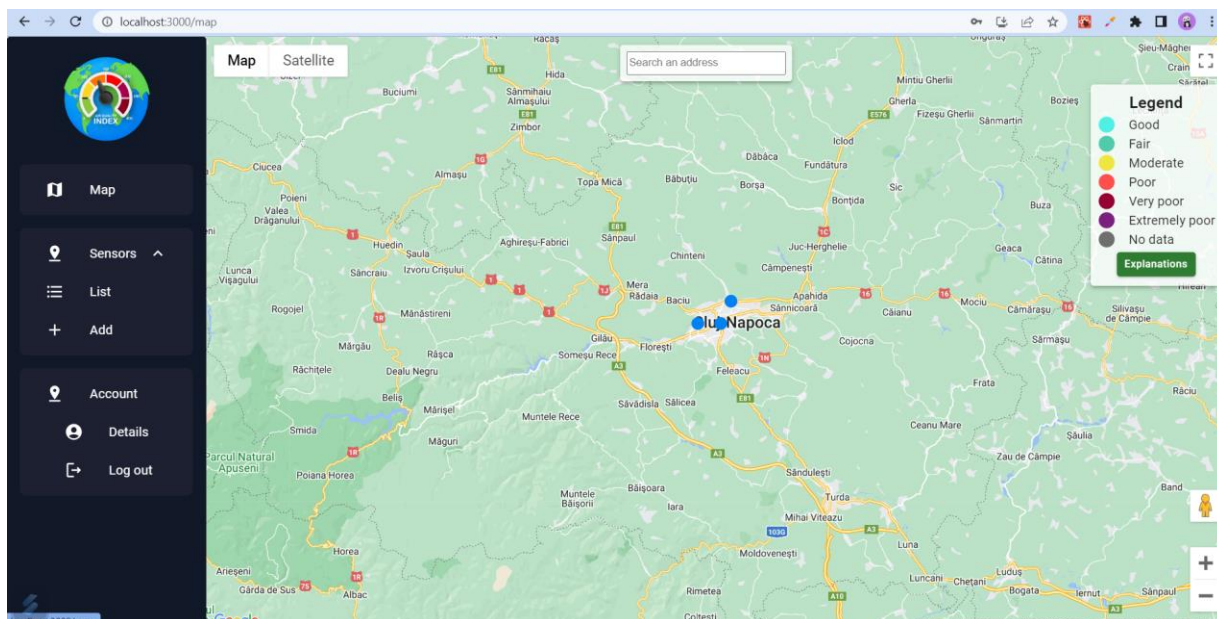


Figura 5.6 Pagina principală a aplicației

În cazul în care se va accesa una dintre paginile aplicației prin manipularea URL-ului, dar utilizatorul nu este conectat în cont, acesta este redirecționat către o pagină în care i se explică faptul că are acces interzis, având opțiunile de creare cont, autentificare și recuperare cont.

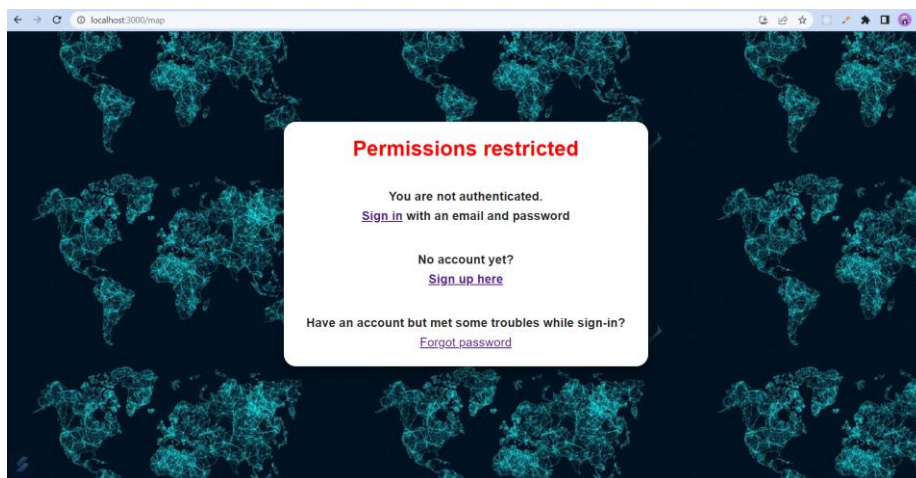


Figura 5.7 Permișune interzisă în cazul în care utilizatorul nu s-a autentificat

Pentru o navigare mai ușoară și a verifica dacă există senzori într-o locație anume, se va introduce în câmpul de căutare locația, iar centrul de vizionare al hărții este redirecționat acolo (pin-ul din Figura 5.8). La apăsarea butonului „Clear”, se va reseta punctul central al hărții.

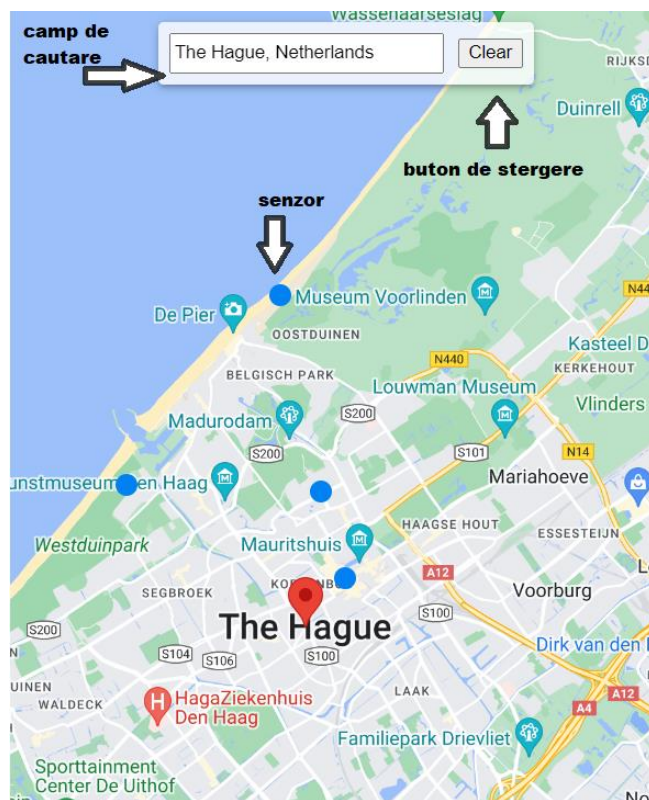


Figura 5.8 Camp de cautare locatie pe harta

În tabelul cu senzori („Sensors -> List”), utilizatorul va putea identifica un senzor anume introducând în câmpul de căutare atât numele cât și data plasării („Creation Date”).

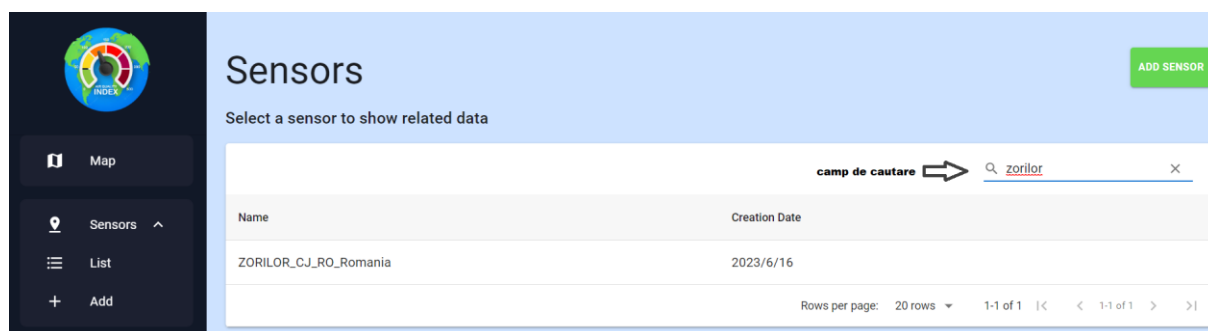


Figura 5.9 Filtrare senzori

Pentru a testa dacă indicele de calitate este calculat corect, vom lua acest exemplu: Tipurile măsurătorilor, respectiv ultimele valori sunt: PM2.5 – 26.34 $\mu\text{g}/\text{m}^3$, PM10 – 58.38 $\mu\text{g}/\text{m}^3$, NO2 – 14.06 $\mu\text{g}/\text{m}^3$, O3 – 232.83 $\mu\text{g}/\text{m}^3$ și SO2 – 119.76 $\mu\text{g}/\text{m}^3$. Valoarea indicelui reprezintă valoarea maximă dintre poluanții gazoși enumerați mai sus și afișați în Figura 5.10.

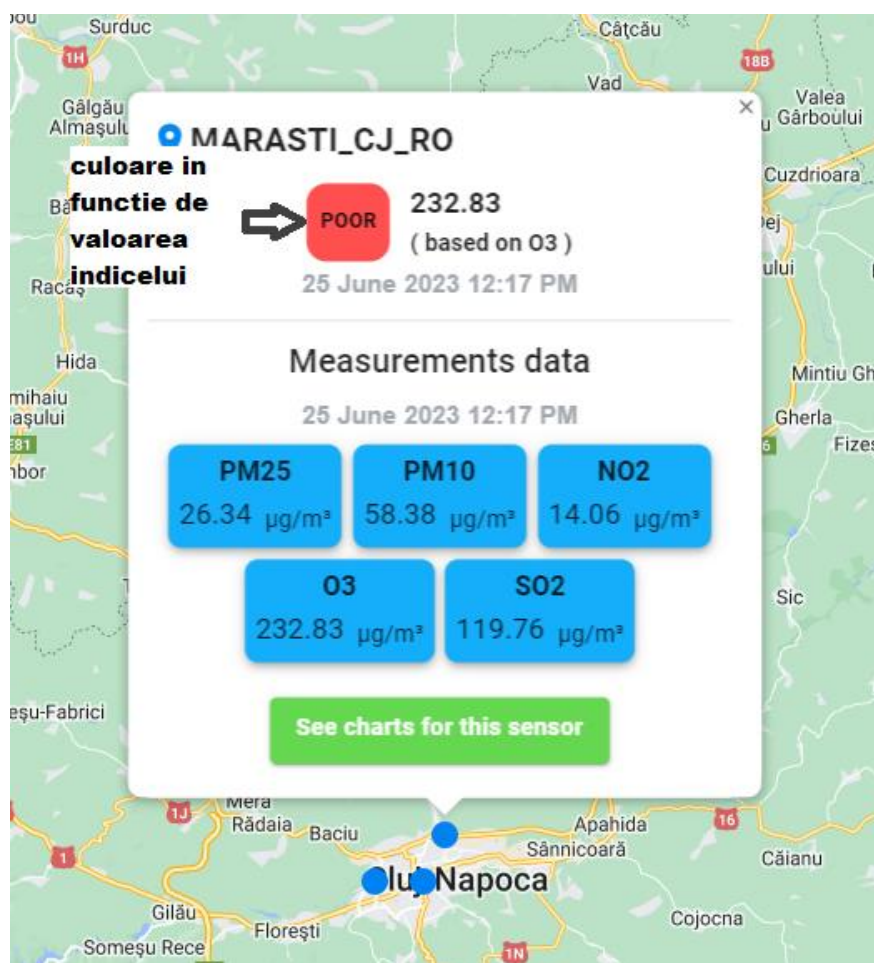


Figura 5.10 Calculare indice, afisare ultimele masuratori ale poluantilor gazosi

Pentru a vedea diagramele cu ultimele 12 ore de măsurători, se va apăsa butonul “See charts for this sensor”. Figura 5.11 reprezintă valorile ultimelor 12 ore pentru umiditate.

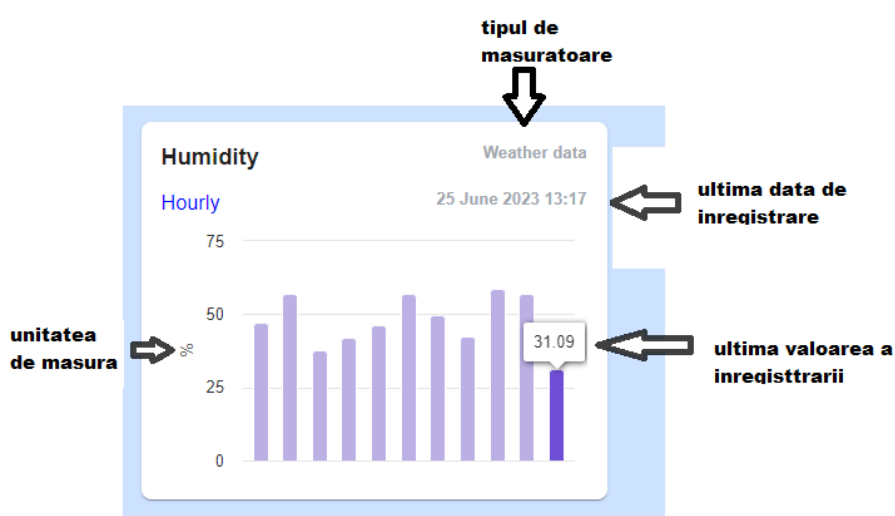


Figura 5.11 Diagrama ultimelor 12 valori O3

6 Concluzii

6.1 Obiective propuse

Obiectivele funcționale propuse inițial sunt în foarte mare parte îndeplinite cu succes în urma dezvoltării aplicației descrise în aceasta lucrare. Aplicația este de tip Web, folosind Spring Boot pe partea de server și React.js pe partea de client.

Funcționalitățile descriu un flux de utilizare eficient, ușor de pus în practică, acestea fiind testate pe parcursul dezvoltării.

Persoana dornică de a verifica indicele de calitate a aerului respirat va avea la dispoziție posibilitatea de a-și crea un cont folosind o adresă de email, de a-și recupera contul prin resetarea parolei, de a se autentifica și vedea senzorii plasați pe harta. Dacă utilizatorul nu are cont, nu va putea fi capabil să vadă datele puse la dispoziție de aplicație.

Pentru a naviga mai repede în locația dorită, va introduce locația în câmpul de căutare și va vedea senzorii din acea zonă. Va verifica indicele de calitate a aerului înregistrat în ultima oră, va primi un mesaj de sănătate pe baza valorii indicelui, metoda de calcul și un tabel cu valorile recomandate de autorități. Mai mult, poate afla starea atmosferei și va putea trage o concluzie asupra condiției meteorologice. Datele sunt înregistrate în baza de date pentru a fi arhivate ultimele 12 ore de înregistrare în diagramele pentru fiecare măsurătoare a senzorului.

Pentru a înțelege ce funcționalități prezintă aplicația, meniul de navigare din stânga este foarte intuitiv, acesta ajutând la navigarea paginilor dorite.

Tabelul de senzori este foarte ușor de vizualizat, acesta are filtre de căutare și opțiuni de paginare, ordonare crescătoare în funcție de nume și de data plasării senzorilor. Astfel, se câștigă timp în procesul de vizualizare a datelor senzorilor și a modificării unor specificații dacă se dorește.

Există și opțiunea de a șterge un senzor, cu o confirmare implementată prin dialogul ce apare după apăsarea butonului de ștergere. În cazul în care utilizatorul se răzgândește, acesta poate anula procesul de eliminare a senzorului.

Se va putea adăuga un nou senzor în pagina de adăugare. Odată ce utilizatorul primește mesaj de confirmare acesta este imediat plasat pe harta.

Gestionarea contului personal se va face în pagina specifică, cu posibilitatea de modificare a numelui și prenumelui. La final, utilizatorul se poate deconecta din cont.

Pentru a asigura o navigare ușoară și plăcută, dimensiunile componentelor se redimensionează pe orice fel de ecrane ale dispozitivelor (computer, tableta, telefon etc).

Suma tuturor funcționalităților aplicației reprezintă cu certitudine faptul că în aceasta lucrare este dezvoltat un sistem care optimizează procesul de monitorizare a calității aerului, punând la dispoziția utilizatorilor o interfață interactivă.

6.2 Direcții de dezvoltare

Aerul exterior expirat este un element esențial în existența noastră și calitatea lui reprezintă o importanță majoră. Cu trecerea timpului, acesta devine din ce în ce mai poluant, deci este vital să se ia măsuri.

Partea bună este că tehnologia perioadei în care trăim evoluează exponențial, fiind supusă frecvent îmbunătățirilor. Aceasta înseamnă că sistemele de informare pot fi supuse progreselor în domeniul dezvoltării.

Implementarea și logica aplicației dezvoltate pe parcursul acestei lucrări sunt ușor de modificat, ușor extensibile, permițând personalizări fără probleme datorită arhitecturii flexibile.

Câteva dintre îmbunătățirile care pot fi dezvoltate sunt:

- Autentificarea utilizatorilor pe baza de roluri: administrator și vizualizator
- În funcție de rol:
 - administratorii (admin) vor putea actualiza datele senzorilor, vor putea adăuga noi senzori sau vor putea șterge senzori doriți;
 - vizualizator (viewer) vor avea posibilitatea doar de vizualizare a datelor disponibile în aplicație;
- Integrarea unui modul hardware (LoRa), configurat cu senzori care înregistrează și transmit valorile poluanților gazoși, comunicând wireless pe distanțe mari;
- Implementarea de teste unitare pentru o testare 100% sigură a funcționalităților aplicației;
- Testarea funcționalităților pe diferite sisteme de operare;
- Aplicație mobilă.

7 Bibliografie

- [1] „European Environment Agency Website,” [Interactiv]. Available: <https://www.eea.europa.eu/en>.
- [2] „WHO global air quality guidelines: particulate matter (PM2.5 and PM10), ozone, nitrogen dioxide, sulfur dioxide and carbon monoxide,” World Health Organization, 2021.
- [3] „National Institute of environmental Health Sciences - Air Pollution and Your Health,” [Interactiv]. Available: <https://www.niehs.nih.gov/>.
- [4] N. L. K. Cromar, „Risk communication of ambient air pollution in the WHO European Region Review of air quality indexes and lessons learned,” 17 February 2023.
- [5] „Firebase,” [Interactiv]. Available: <https://firebase.google.com/>.
- [6] L. Shklar și R. Rosen, Web Application Architecture: Principles, Protocols, and Practices.
- [7] „What is Java,” [Interactiv]. Available: https://www.java.com/en/download/help/whatis_java.html.
- [8] „Spring Boot Framework,” [Interactiv]. Available: <https://spring.io/projects/spring-boot>.
- [9] „React,” [Interactiv]. Available: <https://react.dev/>.
- [10] „Google Maps Platform,” [Interactiv]. Available: <https://mapsplatform.google.com/>.
- [11] „React UI,” [Interactiv]. Available: <https://reach.tech/>.
- [12] „Material-Table Component,” [Interactiv]. Available: <https://material-table.com/#/>.
- [13] „Highcharts React,” [Interactiv]. Available: <https://www.highcharts.com/>.
- [14] „Firebase Token,” [Interactiv]. Available: <https://firebase.google.com/docs/reference/admin/java/reference/com/google/firebase/auth/FirebaseToken>.
- [15] „World Health Organization - Health topics - Air Pollution,” [Interactiv]. Available: https://www.who.int/health-topics/air-pollution#tab=tab_1.

-
- [16] „Evaluation of Carcinogenic Risks to Humans,” în *Outdoor Air Pollution*, IARC Monographs, 2015.
- [17] „California AIR RESOURCES BOARD, Inhalable Particulate Matter and Health (PM_{2.5} and PM₁₀),” [Interactiv]. Available: <https://ww2.arb.ca.gov/resources/inhalable-particulate-matter-and-health#:~:text=Particles%20are%20defined%20by%20their,5>.
- [18] U. Ackermann-Liebrich, „Respiratory and Cardiovascular Effects of NO₂ in Epidemiological Studies,” în *Encyclopedia of Environmental Health*, 2011, p. 844.
- [19] S. Gad, „Sulfur Dioxide,” în *Encyclopedia of Toxicology (Third Edition)*, 2014.
- [20] „World Health Organization, global air quality guaidelines,” 2021. [Interactiv]. Available: <https://apps.who.int/iris/bitstream/handle/10665/345329/9789240034228-eng.pdf>.
- [21] A. Cohen, „Outdoor air pollution and lung cancer,” 1 August 2000 .
- [22] „State of Global Air,” [Interactiv]. Available: <https://www.stateofglobalair.org/>.
- [23] I. HEI, „State of Global Air,” 2020.
- [24] „Airly,” [Interactiv]. Available: <https://airly.org/map/en/>.
- [25] „Rețeaua Națională de Monitorizare a Calității Aerului,” [Interactiv]. Available: https://www.calitateaer.ro/public/home-page/?__locale=ro.
- [26] „Risk communication of ambient air pollution in the WHO European Region, Air pollution forecasting”.
- [27] „Das Umweltbundesamt für mensch und umwelt [Agenția Federală pentru Mediu în scopul protejării oamenilor și mediului],” [Interactiv]. Available: <https://www.umweltbundesamt.de/>.
- [28] „Health Policy Watch - European Union Pushes for Stronger Air Pollution Rules,” [Interactiv]. Available: <https://healthpolicy-watch.news/eu-pushes-for-stronger-air-pollution-rules/#:~:text=The%20new%20rules%20aim%20to%20limit%20PM2.5%20and%20NO, potentially%20saving%20over%20150%2C000%20lives..>
- [29] „Conversia dintre μm^3 si ppb,” [Interactiv]. Available: <https://www.breeze-technologies.de/blog/air-pollution-how-to-convert-between-mgm3-%C2%B5gm3-ppm-ppb/>.
- [30] „EEA - European Air Quality Index - Methodology,” [Interactiv].

Available: [https://www.eea.europa.eu/themes/air/air-quality-index#:~:text=The%20European%20Air%20Quality%20Index,-The%20European%20Air&text=The%20Index%20is%20based%20on,ozone%20\(O3\)%3B](https://www.eea.europa.eu/themes/air/air-quality-index#:~:text=The%20European%20Air%20Quality%20Index,-The%20European%20Air&text=The%20Index%20is%20based%20on,ozone%20(O3)%3B).

[31] „codeCademy - What is CRUD,” [Interactiv]. Available: <https://www.codecademy.com/article/what-is-crud>.