Aplicație software pentru monitorizarea calității aerului exterior

PROIECT DE DIPLOMĂ

Autor: **Cristina-Elena Caia-Hoanăș**

Conducător științific: **As.Drd.ing. Gabriel HARJA**

|  |  |
| --- | --- |
| DECAN  **Prof. dr. ing. Liviu MICLEA** | Vizat,  DIRECTOR DEPARTAMENT AUTOMATICĂ  **Prof. dr. ing. Honoriu VĂLEAN** |

Autor: **Cristina-Elena CAIA-HOANĂȘ**

Aplicație software pentru monitorizarea calității aerului exterior

1. **Enunțul temei:** *Dezvoltarea unei aplicații web care oferă utilizatorului posibilitatea de monitorizare a calității aerului prin evaluarea indexului acestuia în diverse locații.. În ansamblu, se va realiza o perspectiva globala 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, departamentul Automatică și Informatică Aplicată*
4. **Data emiterii temei:** 6 octombrie 2022
5. **Data predării:** iulie 2023

Semnătura autorului

Semnătura conducătorului științific

**Declaraţie 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 software pentru monitorizarea calității aerului 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 Prenume NUME

(semnătura)

**SINTEZA**

proiectului de diplomă cu titlul:

Aplicație software pentru monitorizarea calității aerului exterior

Autor: **Cristina-Elena CAIA-HOANĂȘ**

Conducător științific: **As.Drd.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 aplicatiei in Spring Boot pentru partea de server, structurat in 3 componente: Controller, Service si Firebase;

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

Manipularea datelor este efectuata în baza de date Firebase Database. Procesul de securizare a datelor, autentificarea utilizatorilor si 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 si valorile măsurătorilor acestora in tabele si diagrame, gestionează procesul de autentificare a utilizatorilor.

Funcționalitățile aplicației oferă utilizatorilor o navigare plăcută prin implementarea diferitelor scurtături: căutarea locațiilor pe harta, 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 in partea de frontend. S-au testat diferite scenarii prin introducerea de date incorecte sau manipularea eronată a componentelor pentru a se asigura ca dezvoltarea este corectă si 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 si ușor de utilizat care reda indicele de calitate cu scopul de a influența utilizatorii sa conștientizezi importanța aerului din mediul in care trăiesc și efectele dăunătoare în cazul expunerii la aerul poluant.

6. Surse de documentare:

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

Semnătura autorului

Semnătura conducătorului științific

Cuprins

[1 Introducere 8](#_Toc139294780)

[1.1 Context general 8](#_Toc139294781)

[1.2 Obiective 9](#_Toc139294782)

[1.2.1 Obiectivul principal 9](#_Toc139294783)

[1.2.2 Obiective secundare 9](#_Toc139294784)

[1.2.3 Obiective funcționale 9](#_Toc139294785)

[1.2.4 Obiective non-funcționale 11](#_Toc139294786)

[1.3 Specificații 12](#_Toc139294787)

[2 Studiu bibliografic 14](#_Toc139294788)

[3 Analiză și proiectare 21](#_Toc139294789)

[3.1 Metoda de calcul pentru Indicele de Calitate a Aerului 21](#_Toc139294790)

[3.2 Arhitectura software 22](#_Toc139294791)

[3.2.1 Structura aplicației 22](#_Toc139294792)

[3.2.2 MCV 23](#_Toc139294793)

[3.3 Concepte teoretice 24](#_Toc139294794)

[3.3.1 Baza de date 24](#_Toc139294795)

[3.3.2 Partea server 24](#_Toc139294796)

[3.3.3 Partea client 24](#_Toc139294797)

[3.3.4 REST 25](#_Toc139294798)

[3.4 Resurse software 25](#_Toc139294799)

[3.4.1 Firebase 25](#_Toc139294800)

[3.4.2 Spring Boot 26](#_Toc139294801)

[3.4.1 React JS 29](#_Toc139294802)

[3.4.1.1 React Google Maps API 30](#_Toc139294803)

[3.4.1.2 Material UI si Material-Table 31](#_Toc139294804)

[3.4.1.3 Formik 32](#_Toc139294805)

[3.4.1.4 Highcharts React 33](#_Toc139294806)

[3.4.1.5 Reach UI 34](#_Toc139294807)

[3.4.2 Firebase Authentication 34](#_Toc139294808)

[4 Implementare 36](#_Toc139294809)

[4.1 Configurarea Firebase si Spring Boot 36](#_Toc139294810)

[4.1.1 Crearea proiect Firebase 36](#_Toc139294811)

[4.1.2 Crearea proiect Spring Boot 36](#_Toc139294812)

[4.1.3 Cheia privata Firebase 37](#_Toc139294813)

[4.1.4 Conexiunea Spring Boot la Firebase Database 37](#_Toc139294814)

[4.2 Structura bazei de date Firestore Database 38](#_Toc139294815)

[4.3 Structura Spring Boot 39](#_Toc139294816)

[4.3.1 Spring Boot Service 39](#_Toc139294817)

[4.3.2 Securizarea datelor Spring Boot cu Cloud Firebase 44](#_Toc139294818)

[4.3.3 Controller 44](#_Toc139294819)

[4.3.4 Utilizatorii 45](#_Toc139294820)

[4.3.5 CORS 45](#_Toc139294821)

[4.3.6 Generarea de senzor și măsurători. 45](#_Toc139294822)

[4.4 Structura React JS 46](#_Toc139294823)

[4.4.1 AXIOS 46](#_Toc139294824)

[4.4.2 Firebase Authentication cu React JS și utilizatorii in Firebase Database 47](#_Toc139294825)

[4.4.3 Harta Google Maps cu senzorii si campul de cautare pentru locatii 48](#_Toc139294826)

[4.4.4 Indicele de calitate a aerului si masuratorile 49](#_Toc139294827)

[4.4.5 Sidebar – meniul din lateralul paginii si dimensiunea paginilor adaptivă 49](#_Toc139294828)

[4.4.6 Tabelul senzorilor și modificarea datelor acestora. Adaugarea unui senzor si stergerea 50](#_Toc139294829)

[4.4.7 Gestionarea rutarilor 50](#_Toc139294830)

[5 Testare și rezultate 52](#_Toc139294831)

[5.1 Mod de utilizare 52](#_Toc139294832)

[5.1.1 Set-up backend si frontend 52](#_Toc139294833)

[5.2 Testare 52](#_Toc139294834)

[5.2.1 Testare Postman pentru serverul Spring Boot 52](#_Toc139294835)

[5.2.2 Testare manuala pentru functionalitatile aplicatiei; Rezultate 53](#_Toc139294836)

[**5.2.2.1** **Testare înregistrare, logare si recuperare cont** 53](#_Toc139294837)

[6 Concluzii 58](#_Toc139294838)

[6.1 Obiective propuse 58](#_Toc139294839)

[6.2 Direcții de dezvoltare 59](#_Toc139294840)

[7 Bibliografie 60](#_Toc139294841)

# Introducere

## Context general

In 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, PM2.5 și PM10 (particule cu diametrul aerodinamic mai mic decât 2.5, respectiv 10 µ/m³ - microni pe metru cub), O3 (ozonul), NO2 (dioxidul de azot) si 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 chimica etc. [3]

Poluarea aerului din zonele naturale, rurale și urbane devine o preocupare majoră pentru cercetători, iar Organizația Mondiala 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 si afecțiuni respiratorii acute.

Studiile EPA (United States Environmental Protection Agency) confirmă ca poluarea aerului are de asemenea un impact dăunător asupra climei și ecosistemului la scară mondiala. 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 si informarea populației asupra calității aerului exterior este soluția propusa în aceasta lucrare, reprezentând un prim pas esențial in a ajuta omenirea să perceapă efectele adverse are poluării aerului și să fie conștienți de importanța și valoarea aerului curat pe care l-ar putea consuma. Scopul de lungă durata 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, aceste informații fiind afișate pe interfața grafică a aplicației web, disponibilă utilizatorilor.

## Obiective

### 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 marcare 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 Mondiala a Sănătății (WHO) și un mesaj de informare în funcție de valoare, populația iși 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.

### Obiective secundare

Un obiectiv secundar este ca utilizatorii să dispună și de starea atmosferei: temperatură, umiditate si presiune atmosferică, aceasta fiind accesibilă in 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 sa îi fie vizibilă zona dorita 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 intr-un mod clar si 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 campuri.

### 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 si au un rol important în orientarea procesului de proiectare și implementare.

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

Tabel .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 in caz de uitare a acesteia |
| Redirecționarea pe pagina de înregistrare, conectare sau resetare parola in cazul in 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 cat mai exacte pentru redirecționarea utilizatorului in acel loc pe harta vi vizualizarea senzorilor din acea zona |
| Vizualizare senzorii pe hartă |
| Indicele de calitate a aerului si 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 si 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 si starea atmosferei |
| Tabel cu senzorii din aplicație cu posibilitatea de ordonare a lor după nume sau data plasării lor in locație, paginare si 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 si pe alte dispozitive electronice: telefoane mobile, tablete. |

### Obiective non-funcționale

Obiectivele non-funcționale reprezintă aspecte legate de experiența utilizatorului, scalabilitatea, securitatea si performanta aplicației. Acestea asigura calitatea si eficienta, influențând aspecte precum capacitatea de gestionare a datelor, timpul de răspuns, protecția împotriva amenințărilor cibernetice, gestionarea erorilor si nu in 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 si rapidă, timpii de răspuns fiind scurți, încărcarea paginilor fiind una rapidă. |
| Scalabilitate | Capabilitatea aplicației de adaptare ușoara a creșterii volumului de cererilor simultane din partea numărului mare de utilizatori si in acest timp, performanta 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 si 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 explicatii | Datele prezentate trebuie să fie explicate printr-un sistem oferit de aplicație, impreuna cu suport pentru a ajuta utilizatorii să înțeleagă și să utilizeze eficient aplicația. |

## 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 si flexibil, pentru o manipulare mai eficientă și scalabilitate mai ușoară a datelor nestructurate. [4]

Cartea [5] oferă o analiza detaliata a arhitecturii aplicațiilor web si evidențiază avantajele lor in cadrul procesului de dezvoltare software. Autorii oferă studii de caz si exemple practice pentru a ilustra beneficiile aplicațiilor web. De exemplu, in dezvoltarea unei aplicații de comerț electronic, beneficiile ar putea fi: accesibilitate globala, reducerea costurilor operaționale pentru un magazin fizic, personalizarea produselor pe placul utilizatorului si analizarea detaliata a comportamentului utilizatorilor, performanta produselor si rezultatelor campaniilor de market. Sunt explorate, de asemenea, si diverse aspecte ale acestor exemple precum gestionarea datelor, securitatea si 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 părtii de backend, cat si a celei de frontend, sunt acțiunile esențiale în crearea aplicației. Backend-ul reprezinta 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 [6] cu framework-ul Spring Boot [7], o combinație ce facilitează dezvoltarea rapidă si 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 [8].

Gestionarea si 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 harta Google Maps [9] din librăria React. Această componenta oferă o integrare convenabilă cu API-ul Google si o manipulare ușoară pentru utilizator.

Senzorii pot fi găsiți foarte ușor pe hartă cu ajutorul componentei Combobox , o fundație accesibila a sistemului de design bazat pe biblioteca React [10], 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 [11], Această componenta 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 campului „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 [12], care dispune de o integrare simpla cu animații și interactibilitate 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 parola și o adresa de email, vizibila de asemenea în console Firebase Authentication. Atunci cand utilizatorul este înregistrat si dorește sa se autentifice, metoda signInWithEmailAndPassword este utilizata 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 [13]. Aceasta conține informații de autorizare si autentificare pentru a valida identitatea utilizatorului și solicitările acestuia prin verificarea token-ului trimis de frontend la autentificarea userului.

# Studiu bibliografic

Când vorbim despre respirație, aproape toate viețuitoarele Pământului dispun de aceasta abilitate, însă nu toate beneficiază de același aer curat. Calitatea vieții si sănătatea viețuitoarelor poate fi amenințata de unele substanțe toxice din atmosfera emise de forte naturale si activități umane. Orice agent chimic, biologic sau fizic poate modifica caracteristicile naturale ale atmosferei. [14]

Ozonul (O3), dioxidul de azot (NO2), dioxidul de sulf (SO2) si particulele suspendate in aer (PM 2.5, PM10) sunt substanțele incluse in poluanții care prezinta o preocupare importanta pentru sănătatea publica.

PM2.5 si PM10 reprezintă o combinație complexa de solide si aerosoli, formata din mici picături de lichid, fragmente solide uscate si nuclee solide acoperite cu lichide. Particulele variază semnificativ in ceea ce privește dimensiunea, forma si compoziția chimica, cu posibilitatea de a conține ioni anorganici, compuși metalici, substanțe provenite din scoarța terestra. Sursele PM2.5 si PM10 sunt de multe ori diferite, compozițiile lor fiind distincte. Poluarea cu PM2.5 poate proveni din arderea de benzina, ulei, motorina sau lemn. In adiție, PM10 include si praful provenit din apropierea sănierilor de construcții, incendiile de pădure si arderea vegetației/deșeurilor, surse industriale, polenul si fragmentele de bacterii.

Expunerea la PM produce efecte adverse asupra sănătății, cauzând boli precum afecțiuni cronice ale inimii si a sistemului respirator. Conform unui articol publicat de către Agenția Internaționala pentru Cercetarea Cancerului, particulele suspendate in aerul exterior cauzează cancerul pulmonar. [15] Copii, persoanele astmice si 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, cat si la PM10. [16]

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

Căile respiratorii pot fi iritate de către o concentrație ridicata de NO2, copii astmatici fiind afectați in mod specific in urma expunerii pe termen lung la acest gaz deoarece simptomele de bronșita se accentuează. [17]

Cel mai întâlnit poluant din atmosfera este dioxidul de sulf (SO2), fiind prezent in concentrații ridicate zonele urbane si industriale. In prezenta umidității si altor poluanți, SO2 contribuie la formarea unor straturi de coroziune vizibile. Principala sursa umana a acestui gaz este arderea combustibililor fosili, iar inhalarea acestui rezultat in concentrații ridicate poate cauza stimularea nervilor din căile respiratorii, rezultând modificări patologice, inclusiv edem pulmonar – afecțiune datorata acumulării de lichid in plămâni. [18]

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

Organizația Mondiala a Sănătății (World Health Orzanization), cunoscuta si sub numele de Agenia Natiunilor Unite are ca scop, după cum numele sugerează, sa promoveze stilul de viată sănătos, sa mențină lumea in siguranța si sa fie de ajutor pentru cei vulnerabili, astfel încât lumea sa poată trai apogeul nivelului de sănătate, toate metodele adoptate fiind bazate pe pura știința.

Unul dintre cele mai importante subiecte abordate de către cercetătorii implicați in WHO este poluarea aerului. Ei afirma ca poluarea atmosferei rezulta din prezenta unei sau mai multe contaminări in atmosfera (fum, gaz, praf, ceata sau vapori), iar aceasta expusa in cantități si perioade de timp mari este o amenințare pentru viețuitoarele planetei. Principalele boli care vin in consecința expunerii oamenilor la poluarea aerului sunt accidente vasculare cerebrale, boli cardiatice, rezultate adverse ale sarcinii, diabet, tulburări cognitive si boli neurologice.

Organizația Mondiala a Sănătății recomanda un set de limite pentru poluanți si propune 3 etape de urmat pentru a reduce progresiv poluarea aerului, recomandate mai ales in zonele cu risc ridicat de poluare. In alta ordine de idei, aceste etape reprezintă niveluri de poluanți din atmosfera care sunt mai ridicate in comparație cu nivelurile recomandate de calitate a aerului, dar pe care autoritățile din zona 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 prezinta nivelurile recomandate de calitate a aerului (AQG - air quality guaidance level ) si țintele intermediare (Interim target) in 2 intervale medii pentru 365 zile, respectiv 24 ore. [19]

A picture containing text, screenshot, number, font

Description automatically generated

Figura 2.1, Nivelurile recomandate de către WHO pentru calitatea aerului, WHO [19]

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 publica este in pericol, dar poluanții din atmosfera au un impact negativ asupra ecosistemului si climei pământului la nivel global. In particulele fine (PM) se găsește carbon negru care este considerat unul dintre cei mai influenți contribuitori la încălzirea globala. Acesta încălzește atmosfera pământului absorbind lumina soarelui , accelerând astfel topirea gheții si 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 in timp aproape real. In plus, organizația a elaborat si implementat o strategie de conștientizare a oamenilor cu privire la riscul de contaminare a aerului prin monitorizarea indicelui de calitate. [20]

Majoritatea tarilor europene folosesc un indice comun propus de European Environment Agency. In plus, acesta oferă informații despre sub populațiile afectate, consecința care reprezintă simptomele probabile si in final recomandări specifice pentru a diminua expunerile si riscurile pentru sănătate. Un exemplu este platforma online bazata pe un sistem de informații geografice pentru monitorizarea in timp real a calității aerului in Europa. Aceasta este accesibila atât sub forma de aplicație web, cat si aplicație pentru telefoane mobile. Datele sunt raportate sub forma EAQI (European Air Quality Index), concept dezvoltat de către European Environment Agency. Unele dintre informațiile accesibile sunt locația stației pe hartă, indicele de calitate a aerului monitorizat pe ultimele câteva zile si 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 si Europa de Vest a fost un impuls pentru cercetători sa examineze efectele aerului poluat din exterior, fiind considerata drept o cauza. Deși consumul de tutun a avut un rol central in aceasta epidemie, rezultatele studiilor demonstrează ca agenții cancerigeni sunt eliberați in aerul exterior si inspirarea acestora poate duce la cancerul pulmonar. [21]

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

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

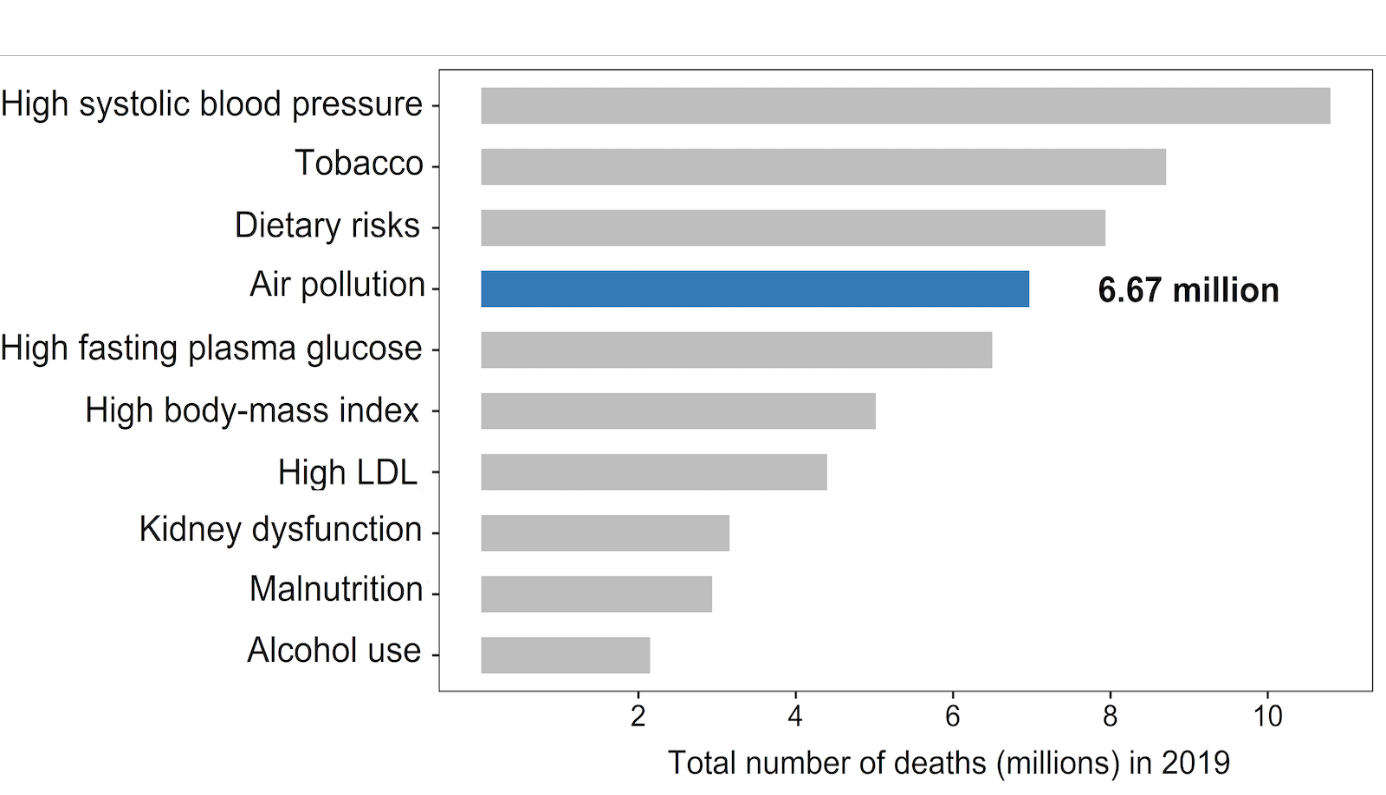


Figura 2.2 Clasamentul global al factorilor de risc în funcție de numărul total de decese din toate cauzele în anul 2019 – State of Global Air [22]

State of Global Air [22] este soluția HEI reprezentata de un website care produce o analiza corespunzătoare a nivelurilor si tendințelor privind calitatea si sănătatea aerului pentru fiecare tara din lume. Un exemplu de raport preluat dintr-un studiu de caz [23] este prezentat in figura de mai jos, unde in 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 in Asia, Africa si Orientul Mijlociu: India – cu 83.2, Nepal – cu 83.1, Niger 80.1, Quatar – cu 76.0 etc, iar cele mai mici expuneri (concentrații medii anuale ponderate sub 8 µg/m³ in funcție de populație) sunt surprinse in Australia, Brunei (Indonezia), Canada, Estonia, Finlanda, Islanda etc.

A map of the world

Description automatically generated with medium confidence

Figura .3 Concentrații ale poluantului PM2.5 în anul 2019 – State of Global Air [22]

Proiectul State of Global Air oferă o cuprinzătoare si comparativa analiza a principalelor factori de risc care contribuie la apariția bolilor si decesului prematur la nivel mondial. In concluzie, aceasta este o baza esențiala pentru stabilirea priorităților in îmbunătățirea sănătății oamenilor. Conștientizarea in ceea ce privește pericolul bolilor ce pot apărea in urma poluării aerului este importanta si vitala pentru planificare acțiunilor de reducere a poluării in moduri cat mai eficiente.

O alta platforma care are ca scop combaterea poluării aerului este Airly [24]. Sistemul consta in diferite forme de monitorizare ca senzori, o platforma web sau o aplicație mobila. Senzorii măsoară indexul de calitate a aerului folosind valorile particulelor suspendate in aer PM1, PM2.5 si PM10 de la cea mai apropiata stație din zona, iar sursele de poluare principale sunt traficul rutier, ferme si incendiile pădurilor. Normele acceptate de către Airly pentru PM2.5 si PM10 sunt cele recomandate de WHO, iar utilizatorul dispune de o scara cu 7 nivele pentru index, începând cu verde ( calitate excelenta) pana la maro (nivel îngrijorător de ridicat).

Populația României este informata in timp real de parametrii de calitate a aerului, monitorizații in mai mult de 100 de stații care cuprind toata suprafața tarii si care, împreună formează Rețeaua Națională de Monitorizare a Calității Aerului (RNMCA) [25]. Datele sunt actualizate in fiecare ora, iar poluanții care sunt luați in considerare pentru a calcula indicele de calitate sunt SO2, NO2, O3, PM2.5 si PM10. Indicele general este exprimat pe o scara de la 1 la 6, 1 reprezentând o calitate foarte buna, iar 6 una extrem de rea. Fiind conștiente de poluarea aerului cu ajutorul RNMCA, autoritățile locale pot lua masuri prompte in timp util pentru eliminarea sau minimizarea poluării sau in cazuri de urgenta, sa avertizeze si protejeze oamenii.

In Figura 2.4 se poate observa valoarea pentru PM2.5 înregistrata de către stație CJ-3 din județul Cluj, la fiecare interval de orar, reprezentata atât sub forma de grafic, dar si tabel.

O imagine care conține text, software, Pictogramă computer, Pagină web

Descriere generată automat

Figura 2.4, Senzor colectare a datelor poluanților și grafice corespunzătoare în judetul Cluj, Romania [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 in mare parte de variațiile zilnice ale calității aerului. [26]

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

Germania – Portalul de date privind calitatea aerului actuală

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

Portugalia – aplicația QualAr

Se redau informații zilnic despre calitatea aerului, dar si prognoze pentru locațiile selectate. Se furnizează avertismente privind sănătatea folosind un sistem care notifica utilizatorul in 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, asa cum sunt furnizate de la autorități, însoțite de mesaje privind sanatatea 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, cat si consecințele acestuia, iar multe studii arata ca este o sursa negativa de boli grave. De aceea, monitorizarea aerului exterior ii va determina pe oameni sa conștientizeze gravitatea situației si sa participe la masurilor care se iau la nivel global.

# 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 si PM2.5 - particulele in suspensie cu diametrul mai mic decât 10, respectiv 2.5 microni, NO2 - ozonul, SO2 - dioxidul de sulf si NO2 - dioxidul de azot. Înregistrarea valorilor poluanților si a stării atmosferice se va face la fiecare ora, dar exista posibilitatea ca unele valori sa lipsească de la oră la oră, de aceea este important sa se ia acest aspect in considerare.

NO2 și PM2.5 sunt poluanți gazoși emiși în special de industrii, 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 ca NO2 si PM2.5 sunt cele cei mai periculoși poluanți gazoși, reprezentând principalele cauze de decesele premature. [28]

Valorile pentru concentrațiile poluanților gazoși PM2.5 si PM10 sunt exprimate in µ/m³ (micrograme pe metru cub), pentru NO2, SO2 si O3 sunt exprimate in µ/m³, părți pe milion (ppm) sau părți pe miliard (ppb), iar indicele de calitate a aerului se exprima in µ/m³, unde ppb reprezintă volumul de poluant gazos din volume de aer.

In aceasta lucrare, senzorii înregistrează valorile măsurătorilor (poluanților gazoși) in părți pe miliard, de aceea este necesara conversia din ppb in µ/m³, iar aceasta depinde de masa moleculara a poluantului gazos si densitatea aerului.

Conversia pentru poluanți gazoși este deja cunoscuta si ușor accesibila, așadar aceasta este prezentata in Tabelul 3.1. [29]

Tabel 3.1 Conversia între ppb si µ/m³ pentru poluantii gazoși folosiți

|  |  |
| --- | --- |
| Poluant gazos |  |
| SO2 | 1 ppb = 2.62 µ/m³ |
| NO2 | 1 ppb = 1.88 µ/m³ |
| O3 | 1 ppb = 1.96 µ/m³ |

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 si pe “European Air Quality Index”, un website oficial al Uniunii Europene. [30]

.

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

Tabelul 3.1 reprezintă intervalele pentru fiecare poluant gazos împărțit in 6 nivele. “Foarte bine” reprezintă o calitate excelenta, 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 |

## Arhitectura software

### Structura aplicației

Arhitectura software in cadrul unei aplicatii web definește modul in care componentele si modelele interacționează intre ele, cum acestea sunt organizate si cum responsabilitățile sunt distribuite.

Stabilirea unei structuri arhitecturale solide pentru o aplicație web este de o importanță vitală în organizarea și structurarea codului, aceasta permițând adaptabilitatea si oportunitatea spre extindere a aplicației. De pe urma acestei implementari, se va putea reutiliza și modulariza codul, testa în mod eficient si 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 ependențe reciproce care asigură funcționarea corectă a aplicației.

Figura 3.1 prezintă schema arhitecturii aplicației bazată pe niveluri.

O imagine care conține text, captură de ecran, Font, diagramă

Descriere generată automat

Figura 3.1 Arhitectura de nivel inalt a aplicației web

### MCV

MCV (Figura 3.2) este un concept de proiectare folosit în dezvoltarea aplicațiilor web, in special in 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 imparti in 3 componente distincte:

1. Modelul (Model) gestionează datele si accesul la acestea, dar este responsabil si 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.

O imagine care conține text, captură de ecran, diagramă, Font

Descriere generată automat

Figura 3.2 Arhitectura MCV

## Concepte teoretice

### Baza de date

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

### Partea server

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

Backend-ul gestionează rutele și punctele finale (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.

### 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 implicand crearea structurii si aspectului aplicației, a componentelor vizuale, a stilizării si a animațiilor.

Partea client implementează funcționalități care rulează în browserul utilizatorului, cum ar fi validarea datelor, gestionarea interacțiunilor si manipularea stării aplicației. Frontend-ul interactionează in legatură stransă cu backend-ul, integrand 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 in 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.

### 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 fară stare ( intre cereri, serverul nu este nevoit sa pastreze nicio stare despre client), definește o interfață uniformă și permite stocarea în cache a răspunsurilor. REST facilitează dezvoltarea aplicațiilor web adaptabile, compatifile și ușor de administrat.

O imagine care conține calculator, text, Dispozitiv de ieșire, captură de ecran

Descriere generată automat

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 in server
* PUT: actualizează o resursa existentă
* DELETE: șterge o resursă specificată prin URL

## Resurse software

### 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 si sincronizarea datelor in timp real intre diferite dispozitive. Firestore dispune de asemenea de serviciu de autentificare si autorizare, utilizând metode simple pentru utilizatori. Firestore storage este un alt serviciu oferit reprezentat de un sistem de stocare scalabil si fiabil pentru fișiere.

În scopul realizării acestei aplicații, s-a utilizat Firestore Database, caracterizată printr-o structura flexibilă și complexa, cu o capacitate de stocare mare. Datele sunt organizate in 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.

O imagine care conține text, captură de ecran, diagramă, proiectare

Descriere generată automat

Figura 3.4 Modul de structurare a datelor în Firestore Database

### Spring Boot

Spring Boot este un framework de dezvoltare pentru aplicații Java, conceput pentru a simplifica și ca un rezultat, a accelera procesul de creare a aplicațiilor robuste și scalabile, deținut si 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 si alte beneficii oferite de SpringBoot (prezentate si in Figura 3.5):

* Livrarea unui set de funcționalități efectuate de catre actuator pentru monitorizarea si gestionarea aplicațiilor in timpul execuției.
* Permiterea integrarii cu alte framework-uri si biblioteci Java prin intermediul gestionării dependentelor și instrumentelor de construire a proiectului Maven.

O imagine care conține text, captură de ecran, diagramă, Font

Descriere generată automat

Figură .5 Funcționalitățile principale oferite de Spring Boot

Arhitectura parții de backend a acestei lucrări (Figura 3.6) este reprezentata de următoarele componente:

1. Clasa responsabila de implementarea logicii de lucrarii, componenta a modelului de design „Service Layer” este numita „Service” si adnotata cu „@Service” pentru a fi marcata ca fiind un bean de serviciu. Aici se descrie funcționalitatea aplicației, in principiu metodele de creare, citire, actualizare si stergere a informațiilor din baza de date.
2. FirebaseInitialize este responsabila de inițializarea bazei de date Firestore Database. Spring Boot recunoaște si aceasta componenta ca fiind un serviciu, de aceea se folosește adnotarea @Service.
3. Componenta Controller este responsabilă de gestionarea interacțiunii intre partea de frontend si 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.

O imagine care conține text, captură de ecran, diagramă, Font

Descriere generată automat

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 prezinta interacțiunea între o aplicație Java MVC și partea de client a unei aplicații web într-un browser web.

Clasa „Sensor” reprezintă o Entitate utilizata pentru a manipula si ilustra datele unui senzor, asociata unui document in 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 toti parametrii, respectiv niciun parametru pentru clasa Sensor si 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). 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 si folosit in aplicație pentru clasa „Sensor” se poate vizualiza in Figura.

O imagine care conține text, captură de ecran, Font, linie

Descriere generată automat

Figura 3.7 DTO (Data Transfer Object) în Spring Boot

O imagine care conține linie

Descriere generată automat

Figura 3.8 DTO integrat in aplicație

### React JS

React.js este o bibliotecă JavaScript open-source creată in mare partea 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 si individuale din cadrul interfetei. Acestea pot fi reutilizate în diverse părți ale aplicației și pot comunica între ele, imbunătățind astfel dezvoltarea și întreținerea aplicațiilor complexe.

Atunci când doar anumite date se schimba, React.js nu redă intrega pagina, 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.

O imagine care conține text, captură de ecran, diagramă, Chip de om

Descriere generată automat

Figura 3.9 DOM-ul virtual React

In exemplul din Figura 3.9, observam ca pentru a modifica datele din campul „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 asigura că DOM-ul real primește și redesenează doar nodurile actualizate.

1. JSX: Permite utilizarea sintaxei JSX, combinând JavaScript și HTML într-un mod expresiv și ușor de înțeles.
2. Integrare ușoară: Se integrează cu alte biblioteci sau framework-uri, oferind extensibilitate.

In 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 si 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 alea acestei biblioteci pentru afișarea unei harți interactive si gestionarea datelor afișate pe aceasta:

Pentru a permite încărcarea și inițializarea API-ului Google Maps, se va folosi funcția „useLoadScript” impreuna cu cheia API furnizată. Cheia se va obține prin crearea unui proiect in consola Google Cloud si 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 reprezentata 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 contine informatii despre indicele de calitate si ultimele măsurători, aceste 2 informatii fiind insoțite de data de actualizare.

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

Figura 3.10 reprezintă arhitectura configurării API-ului Google Maps

O imagine care conține diagramă, captură de ecran, linie, text

Descriere generată automat

Figura 3.10 Arhitectura de servicii pentru Google Maps

### 3.4.1.2 Material UI si Material-Table

Biblioteca MUI – Material UI oferă dezvoltatorilor de aplicații React componente si stilizarea predefinita 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 si implementarea MUI oferă flexibilitate, iar componentele pot fi personalizate in functie de nevoile aplicației (culori, dimensiuni, stiluri). Redimensionarea ecranelor este un alt beneficiu pe care MUI îl pune la dispozitie deoarece componentele se vor adapta la diferite dimensiuni de ecrane si 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 cautare intr-o fereastra), paginarea (datele pot fi afisate pe mai multe pagini pentru o vizualizare mai rapidă și ușoară) editarea și actualizarea datelor, selectarea rândurilor, acțiuni personalizate (ștergerea unui rând ce reprezinta un document in 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 in React JS, cu 2 coloane: „Name” si „Creation Date” este prezentat in Figura 3.11.

O imagine care conține text, captură de ecran, Font

Descriere generată automat

Figura 3.11 Randare Tabel in material-table

### 3.4.1.3 Formik

Pentru a crea, valida și manipula datele din formularele aplicației (Inregistrare 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 pasi: introducerea valorilor de intrare, urmarirea 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 intr-un form Formik si acestea nu sunt in 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 in Figura 3.12. Parola este obligatorie, minimul de caractere acceptat este 12, iar maximul 128.

O imagine care conține text, captură de ecran, Font

Descriere generată automat

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 odata 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. In Form-ul de actualizare a senzorului, sunt afișate valorile câmpurilor din fiecare document senzor din baza de date, iar in tabel sunt afișate numele si data plasării acestuia. La fiecare schimbare a datelor senzorilor, baza de date va fi actualizata, care mai departe, va schimba interfața afișata.

O imagine care conține diagramă, text, captură de ecran, Plan

Descriere generată automat

Figura 3.13 Formik si Material-Table

### 3.4.1.4 Highcharts React

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

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

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

O imagine care conține text, captură de ecran, Font, linie

Descriere generată automat

Figura 3.14 Structura de implementare Highcharts

### 3.4.1.5 Reach UI

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

Utilizând acesta funcționalitate, experiența utilizatorului se va îmbunătăți deoarece are posibilitatea de a găsi rapid locațiile dorite si de verifica daca in zona selectata exista senzori si care sunt valorile măsurătorilor ( indicele de calitate, valoarea poluanților gazoși, starea atmosferica)

### 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.

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

Utilizatorii care folosesc aplicația prezentata in aceasta lucrare vor avea posibilitatea de crearea a unui cont folosind email si parola. Se vor autentifica folosind aceste credențiale, iar in caz de uitare a parolei, vor putea recupera contul primind un email de resetare a parolei.

Firebase Authentication oferă următoarele principale functonaliăți:

Diverse metode de autentificare folosite de utilizatori: email si parola, autentificarea folosind conturi de rețele sociale (Google, Facebook, GitHub etc), folosind numărul de telefon (SMS) sau autentificarea anonima.

Funcții de gestionare a conturilor utilizatorilor, inclusiv crearea de noi conturi, actualizarea datelor utilizatorului, resetarea parolelor și ștergerea conturilor.

Identificarea utilizatorul autentificat prin procesul de gestionare automata a sesiunilor de autentificare și furnizarea unui token.

Integrarea ușoară cu alte servicii Firebase, precum Cloud Firestore sau Cloud Functions.

O imagine care conține text, diagramă, linie, Font

Descriere generată automat

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 Fireabse SDK.

Figura 3.15 reprezinta arhitectura Firebase Autentication, iar procesul de autorizare si accesare a datelor utilizatorului este prezentat asftel:

La autentificarea sau inregistrarea utilizatorului, credentialele vor fi trimise catre server intr-un mod securizat utilizand metodele Firebase SDK: createUserWithEmailAndPassword (inregistrare) sau signInWithEmailAndPassword (autentificare). Pe partea de server, Firebase valideaza aceste credentiale si trimite ca raspuns un token de autentificare, iar în acest fel vom accesa datele (email, nume, prenume) in partea de frontend. Avand token-ul, il putem trimite ca header la partea de backend API si pe baza lui datele utilizatorului vor putea fi accesate iar in cele din urma, aceste vor fi salvate in baza de date Firebase Database in documentele din colectia “users”.

# Implementare

## Configurarea Firebase si Spring Boot

### Crearea proiect Firebase

Stocarea datelor se va realiza in baza de date Firestore Database, serviciu de baza de date Google Cloud. Aceasta oferă un sistem de stocare si sincronizare a datelor intra-un format JSON organizat pe colecții si documente. Firestore Database oferă capacitați de stocare si sincronizare in timp real.

Primul pas in realizarea configurării bazei de date a aplicației, este autentificarea in contul google ( sau mai întâi crearea unui nou cont, daca 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 necesara doar definirea numelui proiectului si acceptarea termenilor si condițiilor). Odata creat proiectul, se va configura tipul de baza de date dorit („Firestore Database”) din secțiunea „Build”. Ultimii pași in acest proces sunt selectarea modului de producție „product mode” pentru a securiza datele definind reguli de acces si setarea locației proiectului.

O imagine care conține text, software, Pictogramă computer, Sistem de operare

Descriere generată automat

Figura 4.1 Creare proiect nou Firebase Database

### 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 si dependentelor dorite. Din secțiunea „Project”, se va selecta Maven, limbajul de programare Java si versiunea, apoi se vor completa câmpurile care definesc proiectul: Group, Artifact, Name etc si dependințele dorite: Spring Web, Lombok, Firebase Admin etc, care vor fi adăugate in fișierul pom.xml.

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

### Cheia privata Firebase

Următorul pas consta in autorizarea aplicației locale Spring Boot si 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”. Fisierul JSON generat conține date de identificare a proiectului (tip, id-ul, cheia privata si id-ul ei etc). Aceasta esta valoarea unica de autentificare utilizata in cererile de acces la baza de date. Pentru a face initia accesul cheie din codul de server, se va salva scriptul in folder principal al proiectului pentru a putea fi accesat ulterior.

### Conexiunea Spring Boot la Firebase Database

Se va crea mai departe clasa „FirebaseInitialize” in pachetul Java „service” pentru a defini si 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 conectare, se vor defini opțiunile firebase, cum ar fi credentialale si URL – ul bazei de date. Deoarece conexiunea este necesara la pornirea aplicatiei, se va folosi adnotarea @Service (Spring Boot va recunoaște aceasta clasa ca fiind un serviciu) si @PostConstruct. Pentru a trata posibilele erori din timpul conectării, se foloseste blocul „try-catch”.

O imagine care conține text, captură de ecran, software, Software multimedia

Descriere generată automat

Figura 4.2 Conexiunea Spring Boot la Firebase Database

## Structura bazei de date Firestore Database

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

**Identificarea entităților principale**

Aplicația dispune de parte de autentificare si autorizare si 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 in diferite locații pe harta, așadar „Sensor” este cea de-a 2-a entitate principala.

**Definirea colectiilor**

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

**Definirea documentelor din colecția „users”**

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

O imagine care conține text, Font, număr, linie

Descriere generată automat

Figura 4.3 Colecția "users" in Firebase Database

**Definirea documentelor din colecția „sensors” si 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 si „uuid”: string (acesta este si ID ul documentului).

Fiecare document senzor poate conține o subcolecție „measurements” reprezentând tipurile si valorile măsurătorilor înregistrate. Câmpurile pentru o măsurătoare sunt: „instantTime”: Timestamp Firebase, „type”: string, „unit”: string, „value”: double.

O imagine care conține text, captură de ecran, număr, software

Descriere generată automat

Figura 4.4 Subcolecția "measurements" in Firebase Database

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

## Structura Spring Boot

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

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

### Spring Boot Service

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

O imagine care conține text, captură de ecran, software, Software multimedia

Descriere generată automat

Figura 4.5 Diagrama claelor pentru entitatile aplicatiei

**Clasa SensorService** implementează logica pentru entitățile senzorilor, principalele metode fiind cele de CRUD.

Pentru a putea scrie, șterge, obține si 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)

O imagine care conține text, captură de ecran, software

Descriere generată automat

Figura 4.6 Crearea unui document senzor

Figura 4.6 reprezintă crearea unui document senzor in Firestore. Campul „uuid” reprezintă identificatorul unic generat pe baza valorii obținute din data si ora curenta, transformate in conținutul șirului de octeți.

Metoda primește ca parametru un senzor DTO, deci se va face conversia in senzor pentru a crea aceasta entitate in 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 in Firestore Database.

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

Tabel 4.1 CRUD in Spring Boot si coresponențele in 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> |

O imagine care conține text, captură de ecran, diagramă

Descriere generată automat

Figura 4.7 Diagrama claselor pentru logica din SensorService

**Data Transfer Object – SensorDTO**

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

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

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

O imagine care conține text, captură de ecran

Descriere generată automat

Figura 4.8 Obiectul Java GeoPointDTO

O imagine care conține text, captură de ecran, software, Software multimedia

Descriere generată automat

Figura 4.9 Diagrama implementării SensorDTO

**Clasa FirebaseService** implementează in 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 in contul existent si resetare parola. Pentru aceasta, se va implementa autentificarea si autorizarea.

Pentru crearea unui unei măsurătoare, metoda primește ca parametrii ID-ul senzorului si obiectul de tip măsurătoare. Mai intai, se va verifica daca ID-ul senzorului trimis ca parametru exista si daca senzorul corelat ID-ului are definit acest tip de măsurătoare. Se va scrie o noua măsurătoare in 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 – in funcție de existenta, iar acestea se vor filtra astfel încât sa rămână doar acele înregistrări din ultima ora (acelea a carui timestamp este maxim).

Valoarea indicelui de calitate a aerului este obținuta din valorile poluanților gazoși (PM2.5, PM10, NO2, SO2, O3) înregistrați in ultima ora ( folosind metoda getLastMeasurementsOfLastHour) . Mai intai se va verifica daca valorile minime necesare (PM2.5 si 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 si poluantul definitor (cel a carui valoare este maxima). Aceste informații se vor folosi in partea de front-end corespunzător.

O imagine care conține text, captură de ecran, diagramă, software

Descriere generată automat

Figura 4.10 Diagrama claselor pentru logica din FirebaseService

### Securizarea datelor Spring Boot cu Cloud Firebase

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

O imagine care conține text, captură de ecran, Font

Descriere generată automat

Figura 4.11 Metoda de validare a tokenului

Senzorii si datele acestora sunt securizate și pentru a fi livrate este nevoie de validarea tokenului. Metoda “validateToken” verifică datele utilizatorului și permisiunile acestuia. Daca 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”.

O imagine care conține text, captură de ecran, Font

Descriere generată automat

Figura 4.12 Validarea tokenului pentru afișarea senzorilor pe harta

### 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 si metodele HTTP (GET – getUser, POST – createMeasurementForSpecificSensor, PUT – updateSensor, DELETE – deleteSensorByUUID) si se va verifica token-ul pentru validarea cererilor utilizatorului.

### Utilizatorii

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

O imagine care conține text, Notiță adezivă, captură de ecran, Font

Descriere generată automat

Figura 4.13 Diagrama Flow-Chart pentru autentificarea userilor in aplicatie

### CORS

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

### Generarea de senzor și măsurători.

Metoda de generare a datelor va genera atât senzorul, cat si 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.

## Structura React JS

Partea de frontend dezvoltata in React JS este structurata după cum se observa in Figura, sub forma mai multor fișiere, organizate in mai multe foldere.

O imagine care conține text, captură de ecran, software, Software multimedia

Descriere generată automat

Figura 4.14 Structura frontend

### AXIOS

Cererile HTTP in mediul de browser se efectuează cu metodele si 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 |

### Firebase Authentication cu React JS și utilizatorii in Firebase Database

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

O imagine care conține text, captură de ecran, Font, software

Descriere generată automat

Figura 4.15 Configurare Firebase

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

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

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

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

O imagine care conține text, captură de ecran, afișaj, software

Descriere generată automat

Figura 4.16 Functia signInUserWithEmailAndPassword

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

### Harta Google Maps cu senzorii si campul de cautare pentru locatii

Pentru a afișa harta din biblioteca „react-google-maps/api” pe pagina “Map”, se defininește hook-ul “useLoadScript” format din cheia API. Se personalizeaza pinul reprezentat de componenta “Marker”, cat si “InfoWindow” (modalul ce se deschide la apasarea pe un pin). Pentru a vedea toti senzorii, se face o cerere “get” catre backend și acesta verifica mai întai tokenul de validare trimis prin header din (tokenul este valabil in “localStorage”. Daca utilizatorul este autentificat, primește datele si acestea se stochează intr-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.

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

### Indicele de calitate a aerului si masuratorile

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

Pentru definirea culorii si încadrarea indicelui într-un status, se implementeaza o componentă numita “SeverityPill”. Aceasta are ca referintă tabelul cu intervalele și în funcție de valoarea si măsurătoarea care definește valoarea, se va genera o culoare. Statusul este primit de la server atunci cand 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” si “column”), datele măsurătorilor, data in care s-a facut ultima măsurătoare, legenda, unitatea de măsură, tooltip (pentru a vizualiza măsurătorile pe ultimele ore) și mesaj in 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

O imagine care conține text, captură de ecran, software, Font

Descriere generată automat

Tabel 4.17 Codul pentru randarea diagramei cu datele de umiditate

### Sidebar – meniul din lateralul paginii si dimensiunea paginilor adaptivă

“Drawer” este o componenta din biblioteca “mui/material” și are dimensiunile adaptive ecranelor. Se foloșeste useState pentru interschimbarea mărimilor și o dimensiune exacta 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 încat aceasta să fie adaptivă la diferite dimensiuni. Principala componentă care permite dimensionarea conținutului este “Grid”.

O imagine care conține text, Font, captură de ecran, Grafică

Descriere generată automat

Figura 4.18 Dimensionare continut in pagina

### Tabelul senzorilor și modificarea datelor acestora. Adaugarea unui senzor si stergerea

Senzorii pot fi de asemenea vizualizati si intr-un tabel “MaterialTable” din biblioteca “material-table”. Acesta are 2 coloane: numele senzorului si data plasarii lui in locatie. La apasarea unui rand, se va redirectiona catre pagina de editare a datelor senzorului. Aici se pot modifica datele (nume si stare: active sau inactiv) prin apasarea butonului de “Update sensor” in formularul Formik. Tipurile de masuratori nu poti fi modificate, acestea reprezentand specificatiile unice ale senzorului.

Senzorul se poate sterge, iar a apasarea butonului pentru a face o cerere de delete catre server, va aparea un modal numit “Dialog” pentru confirmarea stergerii.

Adaugarea unui senzor se realizeaza printr-o metoda de tip post. Se va verifica mai intai tokenul si daca utilizatorul este conectat in cont, dupa care se va trimite obiectul senzor la server. Latitudinea si longitudinea sunt capuri obligatorii de definite in schema de validare Yup Formik. Daca senzorul este adaugat cu success, se va afisa pe ecran un mesaj corespunzator.

### Gestionarea rutarilor

Componentele din biblioteca “react-router” gestionează rutările (componenta cadefinitoare a modului de afișare si manipulare a diferitelor pagini ale aplicației) care depind de URL-ul din browser.

In componenta “App”, “Routes” continue o serie de componente “Route”, fiecare cu un “path” (URL-ul asociat) si o proprietate “element”, reprezentand componenta care trebuie afisata atunci cand URL-ul se portiveste cu ruta respectiva.

O imagine care conține text, captură de ecran, Font

Descriere generată automat

Figura 4.19 Rutarea paginilor aplicatiei

O imagine care conține text, captură de ecran, Notiță adezivă, Font

Descriere generată automat

Figura 4.20 Diagrama flow-chart pentru functionalitatile aplicatiei

# Testare și rezultate

## Mod de utilizare

### Set-up backend si frontend

**Backend – cerințe sistem - instalarea următoarelor**

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

**Frontend – cerinte 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 dependințele proiectului ( acestea vor fi actualizate in scriptul package.json)
4. Rularea aplicației folosind comanda „npm run start” sau „yarn start”

## Testare

### Testare Postman pentru serverul Spring Boot

Platforma Postman pune la dispoziție dezvoltarea si testarea API-urilor, avand un mediu simplu si 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, setam id-ul senzorului („sensorUUID”) in header și introducem adresa URL corespunzătoare, așa cum este specificat in clasa Controller.

O imagine care conține text, captură de ecran, Font

Descriere generată automat

Figura 5.1 Metoda GET aqi in Controller

Răspunsul acestei cereri este “200 OK”, cee ace inseamna ca 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.

O imagine care conține text, captură de ecran, număr

Descriere generată automat

Figura 5.2 Cerere de tip GET in Postman

### Testare manuala pentru functionalitatile aplicatiei; Rezultate

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

#### **Testare înregistrare, logare si recuperare cont**

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

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

O imagine care conține text, captură de ecran, software, hartă

Descriere generată automat

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 in Firebase Authentication, cat si in Firebase Database ( sub forma de document, in colecția “users” – vezi sectioned 4.2).

O imagine care conține text, software, Pictogramă computer, Pagină web

Descriere generată automat

Figura 5.4 Noul utilizator adaugat in firebase Authentication

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

O imagine care conține text, captură de ecran, Font, linie

Descriere generată automat

Figura 5.5 Procesul de recuperare cont prin resetarea parolei

Mai departe, utilizatorul se va putea loga ăn 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 gasește tabelul cu valorile recomandate de către Organizația Mondială a Sănătații și intervalele de feedback, cu mesajele de sănătate în funcție de valoarea indicelui de calitate și metoda de calcul a acestuia.

O imagine care conține text, captură de ecran, software, hartă

Descriere generată automat

Figura 5.6 Pagina principala a aplicatiei

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

O imagine care conține text, captură de ecran

Descriere generată automat

Figura 5.7 Permisiune interzisa in cazul in care utilizatorul nu s-a autentificat

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

O imagine care conține text, hartă, captură de ecran, Font

Descriere generată automat

Figura 5.8 Camp de cautare locatii pe harta

In tabelul cu senzori ( „Sensors -> List”), utilizatorul va putea identifica un senzor anume introducând in câmpul de căutare atât numele cat si data plasării ( „Creation Date”).

O imagine care conține text, captură de ecran, software, Font

Descriere generată automat

Figura 5.9 Filtrare senzori

Pentru a testa daca indicele de calitate este calculat corect, vom lua acest exemplu: Tipurile măsurătorilor, respectiv ultimele valori sunt: PM2.5 – 26.34 µ/m³, PM10 – 58.38 µ/m³ , NO2 – 14.06 µ/m³, O3 – 232.83 µ/m³ si SO2 – 119.76 µ/m³. Valoarea indicelui reprezintă valoarea maxima dintre poluanții gazoși enumerați mai sus si afișați in Figura 5.10.

O imagine care conține text, captură de ecran, diagramă, Font

Descriere generată automat

Figura 5.10 Calculare indice, afisare ultimele masuratori ale poluantilor gazosi

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

O imagine care conține text, captură de ecran, diagramă, Font

Descriere generată automat

Figura 5.11 Diagrama ultimelor 12 valori O3

# Concluzii

## Obiective propuse

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

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

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

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

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

Tabelul de senzori este foarte usor de vizualizat, acesta are filtre de căutare si opțiune de paginare, ordonare crescătoare in funcție de nume si de data plasării senzorilor. Astfel, se câștiga timp in procesul de vizualizare a datelor senzorilor si a modificării unor specificații daca se dorește.

Exista si opțiunea de a șterge un senzor, cu o confirmare implementata prin dialogul ce apare după apăsarea de ștergere. In cazul in care utilizatorul se răzgândește, acesta poate anula procesul de eliminare a senzorului.

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

Gestionarea contului personal se va face in pagina specifica, cu posibilitatea de modificare a numelui si prenumelui. La final, utilizatorul se poate deconecta din cont.

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

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

## Direcții de dezvoltare

Aerul exterior expirat este un element esențial in existenta noastră si calitatea lui reprezintă o importanta majora. Cu trecerea timpului, acesta devine din ce in ce mai poluant, deci este vital sa se ia masuri.

Partea buna este că tehnologia perioadei in care trăim evoluează exponențial, fiind supusa frecvent îmbunătățirilor. Aceasta inseamna ca sistemele de informare pot fi supuse progreselor in domeniul dezvoltării.

Implementarea si logica aplicației dezvoltate pe parcursul aceste lucrări sunt ușor de modificat, ușor extensibile, permițând personalizări fără probleme datorita arhitecturii flexibile.

Câteva dintre îmbunătățirile care pot fi dezvoltate sunt:

* Autentificarea utilizatorilor pe baza de roluri: administrator si vizualizator
* In 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 in 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% sigura a funcționalităților aplicației;
* Testarea funcționalităților pe diferite sisteme de operare;
* Aplicație mobila.

# 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] | „Firebase,” [Interactiv]. Available: https://firebase.google.com/. |
| [5] | L. Shklar și R. Rosen, Web Application Architecture: Principles, Protocols, and Practices. |
| [6] | „What is Java,” [Interactiv]. Available: https://www.java.com/en/download/help/whatis\_java.html. |
| [7] | „Spring Boot Framework,” [Interactiv]. Available: https://spring.io/projects/spring-boot. |
| [8] | „React,” [Interactiv]. Available: https://react.dev/. |
| [9] | „Google Maps Platform,” [Interactiv]. Available: https://mapsplatform.google.com/. |
| [10] | „React UI,” [Interactiv]. Available: https://reach.tech/. |
| [11] | „Material-Table Component,” [Interactiv]. Available: https://material-table.com/#/. |
| [12] | „Highcharts React,” [Interactiv]. Available: https://www.highcharts.com/. |
| [13] | „Firebase Token,” [Interactiv]. Available: https://firebase.google.com/docs/reference/admin/java/reference/com/google/firebase/auth/FirebaseToken. |
| [14] | „World Health Organization - Health topics - Air Pollution,” [Interactiv]. Available: https://www.who.int/health-topics/air-pollution#tab=tab\_1. |
| [15] | „Evaluation of Carcinogenic Risks to Humans,” în *Outdoor Air Pollution*, IARC Monographs, 2015. |
| [16] | „California AIR RESOURCES BOARD, Inhalable Particulate Matter and Health (PM2.5 and PM10),” [Interactiv]. Available: https://ww2.arb.ca.gov/resources/inhalable-particulate-matter-and-health#:~:text=Particles%20are%20defined%20by%20their,5). |
| [17] | U. Ackermann-Liebrich, „Respiratory and Cardiovascular Effects of NO2 in Epidemiological Studies,” în *Encyclopedia of Environmental Health*, 2011, p. 844. |
| [18] | S. Gad, „Sulfur Dioxide,” în *Encyclopedia of Toxicology (Third Edition)*, 2014. |
| [19] | „World Health Organization, global air quality guaidelines,” 2021. [Interactiv]. Available: https://apps.who.int/iris/bitstream/handle/10665/345329/9789240034228-eng.pdf. |
| [20] | N. L. Kevin Cromar, „Risk communication of ambient air pollution in the WHO European Region Review of air quality indexes and lessons learned,” 17 February 2023. |
| [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 µ/m³ 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. |