

UNIVERSITATEA NAȚIONALĂ DE ȘTIINȚĂ ȘI TEHNOLOGIE
POLITEHNICA BUCUREȘTI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL CALCULATOARE



PROIECT DE DIPLOMĂ

Catalog electronic cuprinzând oferta de tehnologii protejate prin DPI

David Mihalenco

Coordonator științific:

Prof. dr. ing. Ciprian Mihai Dobre

BUCUREȘTI

2024

NATIONAL UNIVERSITY OF SCIENCE AND TECHNOLOGY
POLITEHNICA BUCHAREST
FACULTY OF AUTOMATIC CONTROL AND COMPUTERS
COMPUTER SCIENCE DEPARTMENT



DIPLOMA PROJECT

Electronic catalog including the offer of technologies protected by IPR

David Mihalcenco

Thesis advisor:

Prof. dr. ing. Ciprian Mihai Dobre

BUCHAREST

2024

CUPRINS

Sinopsis	4
Abstract.....	4
Mulțumiri	5
1 Introducere	6
1.1 Context.....	6
1.2 Problema.....	6
1.3 Obiective	6
1.4 Structura lucrării	7
2 Analiza și specificarea cerințelor.....	8
2.1 Potențiali utilizatori.....	8
2.1.1 Utilizatori simpli	8
2.1.2 Utilizatori privilegiați (Cercetători)	8
2.2 Specificarea cerințelor	8
2.2.1 Cerințele funcționale	8
2.2.2 Cerințe non-funcționale.....	10
3 Studiu de piață / Abordări existente.....	11
3.1 Abordări existente pe piață	11
3.1.1 Platforme existente pe piață	11
3.2 State of the Art.....	15
3.2.1 Interfețe Intuitive.....	16
3.2.2 Suport și Asistență	16
3.2.3 Securitatea Avansată	17
3.3 Tehnologii folosite în lucrare	18
3.3.1 Motivatia alegerii acestor tehnologii.....	18
3.3.2 Tehnologii alternative	19
4 Solutia propusă	19
4.1 Backend.....	19
4.1.1 PostgreSQL.....	19
4.1.2 Spring Framework.....	21
4.1.3 KeyCloak.....	22

4.2	FrontEnd.....	24
4.2.1	HTML.....	24
4.2.2	CSS.....	24
4.2.3	JavaScript	25
4.3	Arhitectura Aplicației	25
4.3.1	Arhitectura Frontend	26
4.3.2	Arhitectura Backend	27
4.4	Adăugare Infrastructura/Oferta de tehnologie	29
4.5	Modificare/Ștergere Infrastructura	29
4.6	Vizualizare Toate infrastructurile/ofertele si căutarea lor.	30
4.7	Detalii Infrastructura/Oferta.....	31
5	Detalii Implementare	32
5.1	Implementare Login/Logout	32
5.2	Adăugarea unei infrastructuri/oferte	36
5.3	Căutarea unei infrastructuri/oferte	38
6	Studiu De caz / evaluarea rezultatelor	39
6.1	Corectitudinea soluției.....	39
6.2	Evaluarea performanțelor.....	40
6.3	Compararea cu alte platforme.....	42
6.4	Validarea cu clienții.....	42
7	Concluzii	43
8	Bibliografie	44
9	Anexe	45
9.1	Anexa 1 – Reprezentarea Bazei de date	45
9.2	Anexa 2 – Autorizare Flow preluata din [2]	46
9.3	Anexa 3 – Deployment Diagram	47
9.4	Anexa 4 – Pagina de contact.....	48
9.5	Anexa 5 – Pagina de adăugare a unei infrastructuri.....	49
9.6	Anexa 6 – Pagina de adăugare a unei oferte	50
9.7	Anexa 7 – Diagrama Activitate Adăugare Oferta/Infrastructură.....	51
9.8	Anexa 8 – Modificare Infrastructura.....	52
9.9	Anexa 9 - Profil Utilizator	53

9.10	Anexa 10 – Câmpurile definite pentru infrastructura HTML.....	54
9.11	Anexa 11 – Cererea POST pentru adăugare infrastructura	55
9.12	Anexa 12 – Metoda save din infrastructureService.....	56
9.13	Anexa 13 – Căutare Infrastructuri.....	57
9.14	Anexa 14 – Câmpuri infrastructura în baza de date	58
9.15	Anexa 15 – Performanța pe alte pagini ale platformei.....	59
9.16	Anexa 16 – Feedback pagina 1.....	60
9.17	Anexa 17 – Feedback pagina 2.....	61

SINOPSIS

Aceasta lucrare prezinta dezvoltarea si implementarea unei platforme pentru vizualizarea, adăugarea infrastructurilor si ofertelor de tehnologii din cadrul Universității Naționale de Știința si Tehnologie Politehnica București. Platforma își propune sa ofere utilizatorului o platforma intuitiva si interactiva prin care acesta poate sa explore și vizualizeze in detaliu ofertele de tehnologie și infrastructurile.

Obiectivele principale ale platformei sunt: identificarea cerințelor, proiectarea unei interfețe user-friendly și dezvoltarea unui sistem care sa îndeplinească toate cerințele propuse.

Rezultatele obținute indica faptul ca platforma dezvoltata permite utilizatorilor sa vizualizeze in mod eficient ofertele si infrastructurile, filtrarea acestora ajuta in mod deosebit utilizatorii sa găsească într-un timp cat mai scurt ceea ce doresc.

Concluziile subliniază importanta unei interfețe intuitive și a unui sistem de gestionare a datelor bine structurat. Recomandările pentru viitor includ adăugarea de noi tehnologii și continuarea dezvoltării interfeței pentru a face cat mai plăcută navigarea in platforma.

ABSTRACT

This paper presents the development and implementation of a platform for visualizing, adding infrastructures and technology offers within the National University of Science and Technology Politehnica Bucharest. The platform aims to offer the user an intuitive and interactive platform through which he can explore and visualize in detail the technology offers and the infrastructures.

The main objectives of the platform: identifying the requirements, designing a user-friendly interface and developing a system that will meet all the proposed requirements.

The results obtained indicate that the developed platform allows users to effectively visualize the offers and infrastructures, their filtering particularly helps users to find what they want in the shortest possible time.

The conclusions emphasize the importance of an intuitive interface and a well-structured data management system. Recommendations for the future include adding new technologies and continuing to develop the interface to make browsing the platform as pleasant as possible.

MULȚUMIRI

Prof. Dr. Ing. Ciprian Mihai Dobre pentru dedicația s-a, sfaturile si ajutorul pe tot parcursul dezvoltării acestei platforme.

1 INTRODUCERE

1.1 Context

În era digitalizării și a inovațiilor tehnologice rapide, gestionarea eficientă a resurselor și infrastructurilor devine crucială pentru dezvoltarea economică și tehnologică. În cadrul Universității Naționale de Știință și Tehnologie Politehnice București există o mulțime de infrastructuri și oferte tehnologice. Acest catalog este conceput pentru a centraliza și organiza informațiile privind la ofertele de tehnologii și infrastructuri disponibile în universitatea Politehnica, toate fiind protejate prin drepturi de proprietate intelectuală (DPI).

Motivația principală pentru care am dezvoltat această platformă constă în faptul de a promova, și a face cunoscut inovațiile care sunt dezvoltate în Universitate, vrem să aducem motivație și altora să contribuie la dezvoltarea tehnologiilor, chiar să își realizeze propriile idei, platforma le va veni în ajutor pentru a face cunoscută munca lor, și de a găsi alți oameni interesați în acest domeniu. Considerăm important ca noile generații să poată avea un acces ușor la toate informațiile necesare. Aceste informații pot ajuta la dezvoltarea noilor tehnologii, la formarea de echipe pentru dezvoltarea unor proiecte mai mari și la comunicarea eficientă dintre utilizator și cercetător.

1.2 Problema

De-a lungul anilor, eforturile inginerilor din cadrul universității au condus la dezvoltarea unor tehnologii avansate care pot aduce beneficii semnificative societății. Cu toate acestea, gestionarea și promovarea lor rămân provocări pentru universitate.

Cu o creștere constantă a activității de cercetare și dezvoltare, identificarea și documentarea tehnologiilor și infrastructurilor devine tot mai dificilă. Informațiile sunt adesea dispersate pe mai multe platforme și baze de date, ceea ce face dificilă identificarea și accesarea acestora.

În cadrul comunității academice și a industriei există o nevoie de a promova și de a evidenția inovațiile universității pentru a stimula interesul și investițiile în cercetările viitoare.

1.3 Obiective

Obiectivele principale pe care am încercat să le rezolv în acest proiect sunt:

- Centralizarea ofertelor de tehnologie și infrastructurilor din cadrul Universității.
- Ușurarea căutării tehnologiilor și infrastructurilor.
- Crearea unei platforme în care fiecare utilizator va putea adăuga infrastructuri și tehnologii.
- Oferirea detaliilor despre fiecare tehnologie/infrastructură, cât și ușurarea contactării persoanei responsabile.

Crearea unei platforme cu o interfață user-friendly, astfel încât utilizatorii să poată naviga și interacționa cu aceasta fără dificultăți, pentru a facilita accesul rapid la informații și funcționalități relevante a stat la baza creării acestei platforme.

1.4 Structura lucrării

Un paragraf în care fiecare dintre secțiunile următoare este prezentată în 1-2 fraze, punând accentul pe elementele cele mai semnificative din fiecare secțiune.

Capitol 2. Analiza si specificarea cerințelor

În acest capitol am discutat despre cerințele funcționale și cerințele nefuncționale care au fost planuite pentru integrarea pe platforma. Am discutat despre grupurile de utilizatori și posibilitățile lor pe platforma.

Capitol 3. Studiu de piața / abordări existente

În acest capitol am identificat alte platforme asemănătoare care au ca scop același lucru ca și platforma mea. Am discutat despre punctele forte și cele slabe ale platformelor. Am descris tehnologiile folosite în dezvoltarea platformei și alte tehnologii care aș fi putut să le folosesc.

Capitol 4. Soluția propusă

În acest capitol am descris componentele principale ale platformei cum ar fi baza de date, limbajele folosite pentru backend și frontend. Am vorbit despre arhitectura de backend și frontend a platformei, atât și despre funcționalitățile oferite de platforma.

Capitol 5. Detalii Implementare

În acest capitol am descrie mai în amănunt câteva funcționalități ale platformei, acestea includ, adăugare unei oferte și infrastructuri, adăugarea, logarea unui user în platforma, cat și căutarea unei infrastructuri/oferte.

Capitol 6. Evaluarea rezultatelor

În acest capitol am prezentat un chestionar care a fost completat de 20 utilizatori care s-au folosit de platforma mea. Am vorbit despre corectitudinea implementării platformei am discutat despre funcționalitățile care au fost propuse și implementate cât și am comparat performanța platformei mele cu alte două platforme.

Capitol 7. Concluzii

În acest capitol am făcut un sumar al proiectului, am discutat despre ce am reușit să fac, și despre ce ar fi necesar de îmbunătățit în platformă.

2 ANALIZA ȘI SPECIFICAREA CERINȚELOR

În acest capitol se vor stabili utilizatorii potențiali a platformei și voi prezenta funcționalitățile disponibile pe platforma.

Cerințele sistemului au fost modificate de-a lungul dezvoltării platformei, însă au existat cerințe care sunt esențiale pentru o astfel de platforma, care stau la baza ideii catalogului:

- Interfața platformei să fie intuitivă, ușor de navigat între pagini, denumiri relevante la funcționalitatea pe pagina respectivă.
- Posibilitatea de a vizualiza infrastructurile și ofertele tehnologice în două biblioteci diferite pentru a nu crea un haos.
- Căutarea eficientă a infrastructurii și ofertelor.
- Adăugarea și modificarea infrastructurilor și ofertelor pe platforma.
- Crearea unei pagini pentru vizualizarea aparte a unei oferte sau infrastructuri.

2.1 Potențiali utilizatori

Această platformă v-a putea fi accesată de toți utilizatorii interesați de tehnologiile și infrastructurile din cadrul Universității. Însă pentru a beneficia de toate funcționalitățile platformei am identificat câteva grupuri de utilizatori:

2.1.1 Utilizatori simpli

Acest tip de utilizatori sunt cei care accesează platforma doar pentru a beneficia de informațiile oferite despre tehnologiile și infrastructurile prezente pe platforma. Acești utilizatori beneficiază de toate funcționalitățile care permit căutarea și vizualizarea ofertelor.

2.1.2 Utilizatori privilegiați (Cercetători)

Acest tip de utilizatori sunt cei care nu doar accesează platforma pentru a vizualiza ofertele și infrastructurile, dar și contribuie la adăugarea de noi oferte. Acești utilizatori au un cont privat, beneficiază de toate funcționalitățile disponibile pe platforma.

2.2 Specificarea cerințelor

În procesul de dezvoltare a unui proiect este foarte important un feedback continuu din parte unui utilizator. Cerințele au fost stabilite cu îndrumătorul meu în acest proiect Ciprian Mihai Dobre, cu care am avut discuții privind cerințele funcționale ale proiectului.

2.2.1 Cerințele funcționale

- **Înregistrarea pe platforma:**

Înregistrarea pe platforma stă la baza adăugării unei oferte sau infrastructuri noi pe platforma. Crearea unui cont privat în care utilizatorii interesați de o infrastructură sau oferta vor putea vedea informații relevante despre cercetătorul care a înregistrat-o.

- **Login/Logout:**

Utilizarea platformei nu este redusa de autentificarea pe platforma. Însă pentru a avea un cont privat aceasta este obligatorie.

- **Crearea unui cont de cercetător/sau cererea drepturilor.**

Pe platforma nu orice utilizator autentificat are drepturi de a adaugă infrastructuri si oferte. Pentru a avea aceste drepturi implicit, utilizatorul își v-a crea un cont cu o adresa de e-mail UPB adică @upb.ro si v-a trece verificarea e-mailului, la crearea unui astfel de cont v-a primi automat un mail. Însă daca utilizatorul are nu are un astfel de e-mail atunci v-a putea cere aceste drepturi.

- **Crearea paginilor de vizualizare a infrastructurilor/ofertelor tehnologice**

La accesarea platformei utilizatorul v-a putea naviga la paginile de vizualizare a ofertelor tehnologice si infrastructurilor, unde sunt afișate toate cele disponibile.

- **Crearea unui mecanism de căutare**

Utilizatorul v-a putea caută infrastructurile si ofertele după nume sau după cuvinte cheie.

- **Pagina de vizualizare a unei oferte sau infrastructuri**

Utilizatorul v-a putea accesa o oferta sau infrastructura si v-a putea vedea detalii despre ele. O oferta v-a avea următoarele date: nume, imagine, descriere, beneficiile aduse de oferta, colaborările, telefon, e-mail, in ce context se utilizează, statutul, cuvinte cheie.

O infrastructura v-a avea următoarele date: nume, imagine, descriere, beneficii aduse de infrastructura , e-mail, telefon, specificațiile tehnice, locația pe harta a infrastructurii, cuvinte cheie.

- **Adăugarea infrastructurilor si ofertelor**

Crearea unei pagini in care un cercetător v-a introduce toate datele, si v-a putea publica pe platforma.

- **Modificarea infrastructurii sau ofertei**

Posibilitatea de a modifica infrastructurile si ofertele adăugate de un cercetător, el v-a putea modifica doar daca el a publicat.

- **Ștergerea infrastructurii sau ofertei**

Posibilitatea de a șterge infrastructurile si ofertele adăugate de către cercetător.

- **Vizualizarea ofertelor si infrastructurilor mele**

Fiecare cercetător v-a putea vizualiza ofertele si infrastructurile sale, unde v-a putea modifica si șterge.

In urma implementării acestor cerințe, după câteva discuții s-au mai arătat importante încă câteva aspecte importante, funcționalități care platforma le necesita:

- **Vizualizarea contului propriu**

Fiecare utilizator autentificat v-a putea accesa contul sau privat in care v-a avea informații despre el, aici își v-a putea seta si o poza de profil.

- **Creare unei pagini de Contact**

Adăugare unei pagini in care utilizatorul v-a avea toate informațiile de contact a platformei, si un ghid despre adăugarea ofertelor sau infrastructurilor pe platforma. Aici utilizatorul v-a putea accesa e-mailul platformei și v-a putea cere drepturi de cercetător.

- **Vizualizare oferte noi si infrastructuri noi**

Posibilitatea de a urmări ofertele noi adăugate și infrastructurile noi adăugate pe pagina principala a platformei.

2.2.2 Cerințe non-funcționale

Cerințele non-funcționale sunt niște constrângeri ale sistemului.

- **Accesibilitatea**

Utilizatorii vor putea ușor sa acceseze si sa utilizeze platforma. Platforma v-a avea o interfață intuitiva.

- **Securitatea**

Conturile utilizatorilor trebuie sa fie protejate de accesul neautorizat, adică alta persoana sa nu dăuneze asupra muncii depuse de un cercetător. Pentru a putea face anumite acțiuni v-a fi nevoie de autorizarea in platforma, un utilizator neautentificat nu v-a putea face acțiunile precum adăugare de oferte și infrastructuri.

- **Performanța platformei**

Sistemul trebuie sa răspundă rapid la cererile utilizatorului, viteza de încărcare a datelor sa fie una sporita.

Toate aceste cerințe duc la atragerea utilizatorilor noi pe platforma și la menținerea celor vechi pe ea. Ele trebuie luate în considerare și nu ignorate. Mentenanța platformei consta in primirea de feedback continuu din parte utilizatorilor si îmbunătățirea platformei, adăugarea de noi funcționalități necesare.

3 STUDIU DE PIAȚĂ / ABORDĂRI EXISTENTE

În acest capitol voi analiza alte platforme existente pe piață, voi descrie punctele forte și slabe ale acestor platforme. Voi vorbi despre starea curentă a domeniului, contextul, și unde ne găsim în dezvoltarea acestor platforme.

3.1 Abordări existente pe piață

Pe piața actuală există câteva abordări în ceea ce privește centralizarea ofertelor tehnologice și infrastructurilor. În general, abordările existente pe piață reflectă diversitatea nevoilor și preferințelor utilizatorului, încercând să răspundă cât mai bine acestora.

Dacă vorbim de starea curentă a acestui domeniu, nu prea există concurenți pe piață, nu există o diversitate de platforme care reprezintă un catalog centralizat al tuturor ofertelor și infrastructurilor. De obicei instituția are pe platforma sa un capitol unde se pot găsi tehnologiile și infrastructurile prezente pe teritoriul său sau tehnologiile dezvoltate de ei.

În cele ce urmează voi prezenta platformele deja existente, voi evidenția punctele forte și părțile slabe ale lor.

3.1.1 Platforme existente pe piață

infratech.crescdi.pub.ro¹

Această platformă include infrastructurile și ofertele tehnologice din UNSTPB, aplicația are o interfață destul de ușor de înțeles, clar de navigat.

Fluxul platformei:

- Pagina principală a platformei, accesul direct la toate infrastructurile și ofertele disponibile, o pagină de contact, și o bară de căutare care afișează atât infrastructurile cât și ofertele. A se vedea Figura 1.



Figura 1. Pagina principală Infratech.crescdi.pub.ro

¹ <https://infratech.crescdi.pub.ro/> accesat la data de 06.07.2024

- Platforma permite căutarea și navigarea fără restricții chiar dacă utilizatorul nu are un cont.
- Opțiunea de a vedea în detaliu o infrastructură sau oferta cu toate detaliile despre ea. Atât cât și un calendar, în care se poate observa când o infrastructură este disponibilă.
- Opțiunea de adăugare a unei infrastructuri sau oferte, în care utilizatorul completează câmpurile deja definite de către platformă. Pentru a adăuga o infrastructură sau oferta utilizatorul trebuie să-și creeze un cont.
- Existența opțiunii de a crea un cont pe platformă. A se vedea Figura 2.

Figura 2. Pagina de adăugare a unei infrastructuri/oferte

Puncte forte ale platformei :

- Interfața prietenoasă, ușor de navigat, conținutul paginii este aranjat bine.
- Prezența unei pagini de contact, unde utilizatorul are posibilitatea de a contacta ușor administrația platformei, există câmpuri deja definite în care utilizatorul introduce datele necesare pentru contactare. A se vedea Figura 3.

Figura 3. Pagina de contact infratech

- Opțiunea de creare a unui cont, care permite adăugarea de infrastructuri și oferte ceea ce nu permite oricărui utilizator să adauge pe platforma. Asta duce la controlarea procesului de adăugare, nu permite adăugării unor infrastructuri sau oferte tehnologice inadecvate de către un utilizator necunoscut.
- Prezenta unui calendar în care utilizatorul poate monitoriza când o infrastructura este disponibilă. Și opțiunea de a rezerva o infrastructura.
- Opțiunea de a vedea detalii despre o infrastructura sau oferta, unde avem informații despre fiecare infrastructura și oferta. Aici putem găsi datele de contact, alte pagini relevante, și informațiile necesare despre product. A se vedea Figura 4.

Infrastructurile de cercetare din cadrul Universității Naționale de Știință și Tehnologie **POLITEHNICA** București se pot înscrie de către coordonatorii acestora specificând următoarele aspecte: 1. Descriere infrastructură de cercetare; 2. Accesul la infrastructura de cercetare; 3. Acknowledgement și Publicare; 4. Pregătire experimente; 5. Echipamente; 6. Costuri experimente; 7. Link EERTIS; 8. Contact.

Tehnologiile dezvoltate în cadrul Universității Naționale de Știință și Tehnologie **POLITEHNICA** București se pot înscrie de către coordonatorii acestora specificând următoarele aspecte: 1. Titlu tehnologie; 2. Rezumat tehnologie; 3. Cuvinte cheie; 4. Contextul și conceptul de bază al tehnologiei; 5. Avantaje și elemente competitive; 6. Domenii de utilizare și oportunități de piață; 7. Statutul / stadiul protecției Proprietății Intelectuale; 8. Colaborări solicitate; 9. Contact

Figura 4. Datele disponibile despre infrastructura/tehnologie Infratech

Puncte slabe ale platformei:

- La crearea unui cont, utilizatorul v-a fi nevoit să aștepte aprobarea de către administratorul platformei. Ceea ce poate aduce pe viitor o problema majoră cu un flux de utilizatori mare, acest proces ar fi trebuit automatizat.
- Lipsa unui buton de autentificare în bara de navigare, autentificarea apare doar atunci când utilizatorul accesează anumite pagini.

EERTIS.EU²

EERTIS - Engage in European Research and Technology Infrastructure System este o platformă online gratuită care face legătura între furnizorii de infrastructură de cercetare și tehnologie și potențialii utilizatori.

Pe această platformă există două tipuri de conturi: Conturi administrative cu drepturi și responsabilități specifice. Conturi de utilizator fără drepturi administrative, dar cu posibilitatea de a naviga pe platformă și de a avea acces la informațiile publicate.

² <https://eertis.eu/> accesat la data de 06.07.2024

Care sunt drepturile și responsabilitățile unui administrator? Informația extrasă din EERTIS.EU About us³

- Poate adăuga și edita informații despre organizație.
- Poate defini structura organizației adăugând sub organizații, unități de cercetare, laboratoare, echipamente de sine stătătoare și listați serviciile furnizate direct de organizație și numiți administratori pentru acestea.
- Poate șterge/analiza contul organizației.

Fluxul platformei:

- Pagina principală a platformei pe care găsim o bară de căutare pe platforma unde putem găsi după un nume orice infrastructura, tehnologie, sau oferta. Aici avem și posibilitatea de a ne autentifica sau crea un cont. Statistici despre numărul de infrastructuri, oferte, tehnologii și echipamente pe care le putem găsi pe platforma.
- Pagina unde putem vedea toate organizațiile care au postat ceva pe platforma, infrastructurile, echipamentul și serviciile care sunt disponibile, aici ele se pot sorta după diferite criterii: Țara, status legal și tip. A se vedea Figura 5.

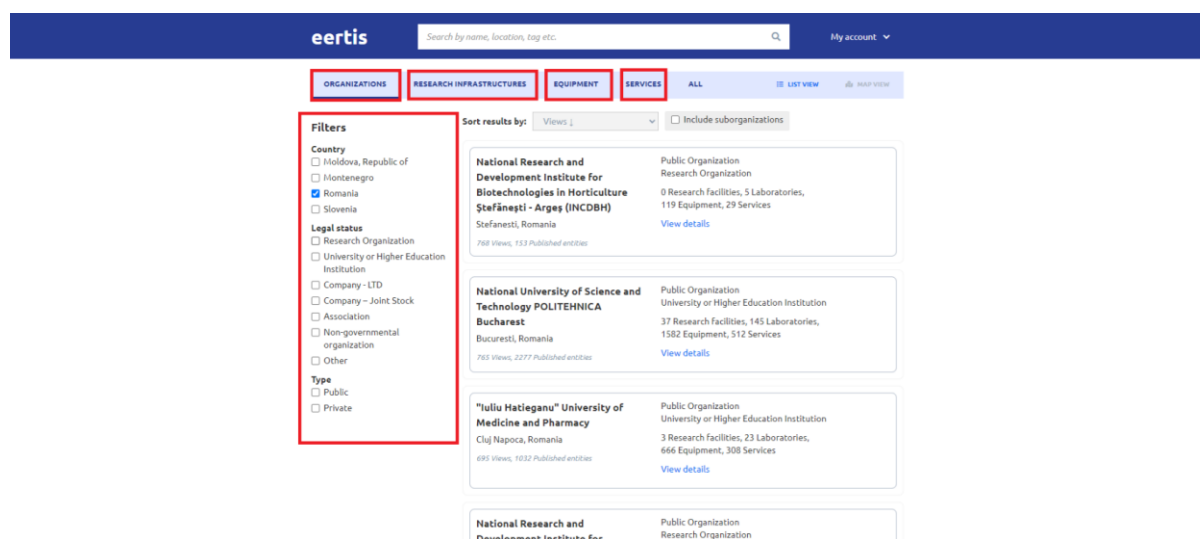


Figura 5. Tipurile de entități pe platforma eertis.eu, filtrarea lor.

- Pagina de vizualizare detalii, aici putem găsi informații despre o anumită entitate care ne interesează.

Puncte forte ale platformei:

- Gama largă de infrastructuri și oferte tehnologice, putem găsi foarte multe informații despre ce ne interesează.

³ <https://eertis.eu/about-us> accesat la data de 06.07.2024

- Prezenta unei locații pe harta, putem vedea unde se afla o infrastructura daca ne interesează sa o vizitam. A se vedea Figura 6.

Contact

Name: Technologies - Plant Protection - Virology in Horticulture

Contact information

Phone: +40740313695

Address: Fitotronului 50 Stefanesti 117715
Arges Romania

Direct link to EERTIS page

<https://eertis.eu/erlb-2300-000e-0198>

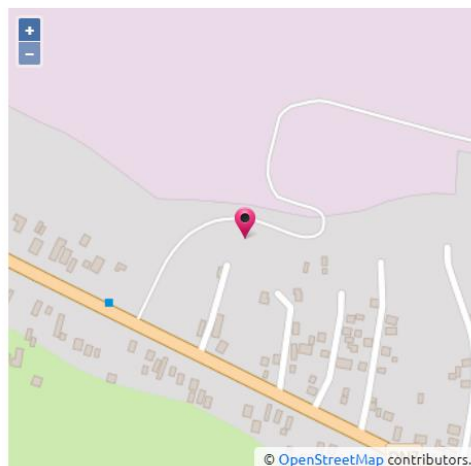


Figura 6. Locația pe harta a unei infrastructuri eertis.eu

- Prezenta unui algoritm de sortare laborios, care ajuta la căutarea informațiilor mai vaste, adică infrastructuri, oferte, tehnologii, din niște regiuni anumite.
- O gama larga de detalii despre o anumita entitate.

Puncte slabe ale platformei:

- Nu putem accesa direct infrastructuri si oferte din pagina principala a platformei, aceasta făcându-se ori prin căutarea in bara de căutare, sau de pe butonul explore, unde se vor afișa toate la un loc.
- Pentru a înțelege cum sa adaugi ceva pe platforma v-a trebui sa citești mai multe întrebări frecvente adică FAQ's, sa găsești întrebarea care te interesează din zecile de întrebări, și după sa creezi un cont.

3.2 State of the Art

"State of the art" este o expresie folosită pentru a descrie nivelul cel mai înalt de dezvoltare, performanță sau calitate într-un anumit domeniu la un moment dat.

În contextul unei platforme care reprezintă un catalog al infrastructurilor și ofertelor tehnologice, "state of the art" ar implica implementarea celor mai avansate caracteristici și tehnologii disponibile pentru a oferi o experiență completă, intuitivă și personalizată utilizatorilor.

3.2.1 Interfețe Intuitive

O interfață ușor de utilizat și intuitivă este crucială pentru a permite utilizatorilor să navigheze și să compare rapid diferitele opțiuni disponibile.

EERTIS.eu

Aceasta platforma oferă utilizatorului accesul ușor, la toate paginile disponibile, utilizatorul are accesul la vizualizarea datelor despre infrastructuri și oferte. Poate accesa, caută ușor informațiile necesare. Pe pagina principală a platformei poate găsi statistici despre platforma, cât și se poate contacta cu administratorii platformei.

Crearea contului este accesibilă și clară, platforma îți oferă alegeri, pentru a îți personaliza contul și pentru a face folosirea platformei cât mai familiară. Toate entitățile sunt sortate în diferite categorii, pot fi accesate, filtrate pentru a găsi informația necesară.

Platforma oferă și o ilustrație vizuală a locației pe harta a infrastructurilor, ceea ce ajută utilizatorul să înțeleagă exact unde se afla infrastructura, și cum poate ajunge la ea.

INFRATECH

Bara de navigare a acestei platforme oferă accesul la multe dintre posibilitățile platformei ceea ce ajută utilizatorul să acceseze rapid informația necesară. Informația despre infrastructuri și oferte este bine poziționată în pagină. Căutarea poate fi făcută direct din bara de navigare, unde utilizatorul va fi trimis pe o pagină cu toate infrastructurile și tehnologiile care coincid cu căutarea.

Calendarul care este disponibil la fiecare infrastructură ajută utilizatorii să-și planifice următoarele vizite sau rezervări. Rezervarea se face ușor datorită posibilității de a selecta zilele direct din calendar.

3.2.2 Suport și Asistență

EERTIS.eu

Pe această platformă utilizatorul are acces la cele mai frecvente întrebări, are acces la diferite ghiduri pentru utilizator și la asistență prin email. A se vedea Figura 7.

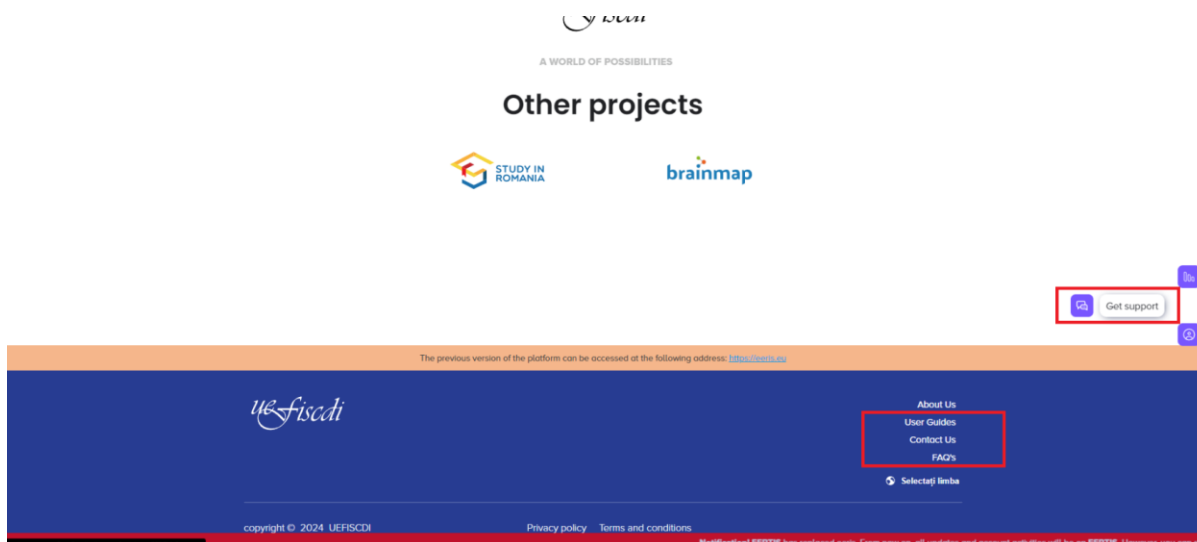


Figura 7. Asistență eertis.eu

INFRATECH

Pe platforma exista o pagina de contact, cat si ghiduri de adăugare a unei infrastructuri sau oferte pe platforma. Pe pagina de contact avem așa informații ca: email, număr de telefon, și adresa poștala la care poate fi contactata platforma. A se vedea Figura 8.

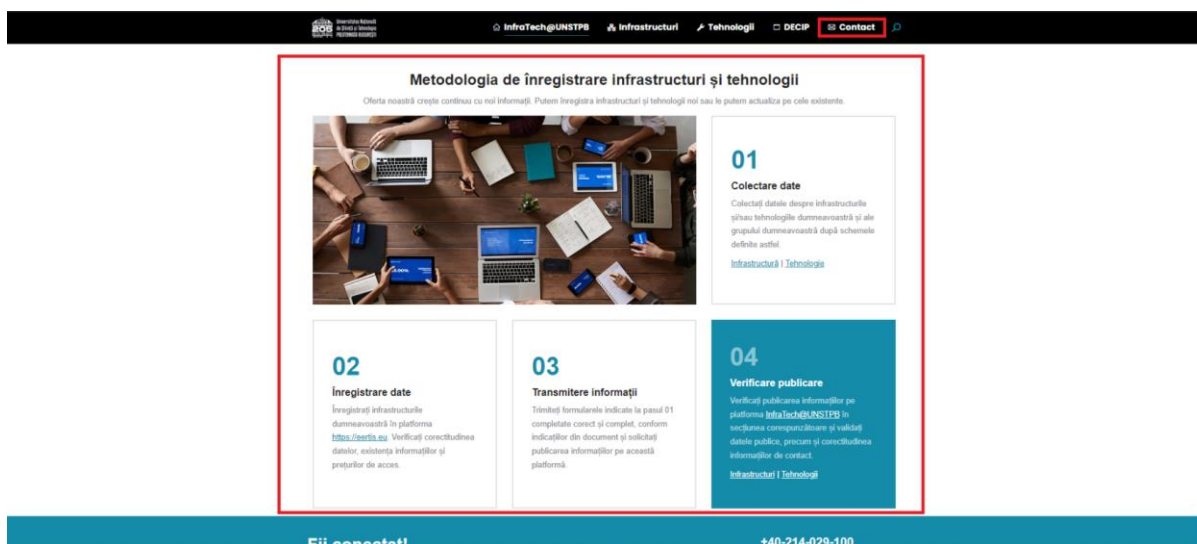


Figura 8. Asistență infratech

3.2.3 Securitatea Avansată

Pe ambele platforme este necesara creare unui cont pentru a adaugă date pe platforma. Aceasta nu permite altor utilizatori răufăcători, sa modifice, sau sa șteargă infrastructurile sau ofertele adăugate de alt utilizator.

Pe platforma infratech, după înregistrare utilizatorul v-a fi nevoit sa aștepte confirmarea unui administrator ceea ce permite administratorului sa controleze manual procesul de

înregistrare a utilizatorilor, și acceptarea celor care îndeplinesc anumite criterii. Ceea ce reduce intrarea în sistem a unor utilizatori suspicioși.

Pe platforma eertis, utilizatorul v-a fi nevoit să dețină un email valid, să confirme că el a creat acest cont. Aceasta implementare poate contribui la reducerea riscului de creare a unor conturi automate în masă sau botneturilor deși nu este o soluție completă și absolută pentru oprirea acestora.

3.3 Tehnologii folosite în lucrare

Baza de date: Pentru baza de date am folosit PostgreSQL.

Frontend și Backend Development: Pentru partea de frontend, am utilizat tehnologiile precum HTML, CSS și JavaScript pentru a crea interfețele platformei interactive și intuitive. Pentru partea de Backend am utilizat Java cu Spring Framework pentru a gestiona logica și interacțiunea cu baza de date.

API-uri RESTful: Utilizarea de API-uri RESTful pentru a permite comunicarea între frontend și backend.

Autentificare și autorizare: Am utilizat keycloak care este o soluție simplă și puternică pentru gestionarea autentificării.

3.3.1 Motivatia alegerii acestor tehnologii

HTML, CSS și JavaScript sunt limbaje flexibile, care permit crearea interfețelor interactive. CSS oferă posibilitatea de a stiliza elemente HTML, iar JavaScript permite crearea de interacțiuni mai complexe și dinamice în cadrul unei pagini web. Aceste sunt suportate de o varietate mare de platforme și dispozitive, ceea ce asigură compatibilitatea cu o gamă largă de browsere și dispozitive. Există o mulțime de resurse, tutoriale și instrumente disponibile pentru dezvoltarea cu HTML, CSS, JavaScript, ceea ce ușurează căutarea informațiilor, și ușurința rezolvării problemelor care pot apărea. Mai multe informații sursa [4].

Folosirea Spring Framework a fost făcută datorită flexibilității și modularității care oferă această unui proiect. Acest Framework oferă o abordare modulară și bazată pe componente, permițând personalizarea configurației platformei conform cerințelor utilizatorilor. Spring Framework este însoțit de module și proiecte suplimentare cum ar fi Spring Boot, Spring Security, Spring Data. Mai multe informații se pot găsi în sursa [5].

Am folosit PostgreSQL datorită scalabilității și extensibilității acestuia, aceasta permite creșterea capacității și performanței bazei de date. Folosirea acestuia a fost motivată și de ușurința integrării acestuia în proiectul meu datorită folosirii SpringFramework. Spring oferă suport nativ pentru integrare PostgreSQL, prin intermediul modulului Spring Data JPA.

Keycloak oferă funcționalități avansate pentru autentificare, inclusiv suport pentru autentificarea bazată pe utilizator. De asemenea oferă un control asupra resurselor care pot fi accesate prin intermediul politicilor de autorizare. Keycloak conține o varietate de

mecanisme, Two-Factor Autentification, creare de cont, gestionarea conturilor utilizatorilor, crearea de roluri, funcții precum înregistrare, autentificare si recuperare parola.

3.3.2 Tehnologii alternative

Alternative JavaScript ar fi TypeScript care este un superset al JavaScript acesta adaugă tipuri statice și alte caracteristici orientate pe obiect. El îmbunătățește scalabilitatea codului, facilitează detectarea de erori la compile-time. Însă JavaScript este mai simplu, iar TypeScript necesita configurarea unui sistem de compilare. Informații preluate din [1]

Pentru partea de frontend a platformei exista multe alternative: JakartaEE, Micronaut, Quarkus, Dropwizard etc.

JakartaEE este o platforma enterprise pentru dezvoltarea aplicațiilor Java, aceasta oferă specificații pentru dezvoltarea aplicațiilor web si enterprise, însă aceasta este mai complex si greoi decât Spring.

Micronaut este un framework modern pentru dezvoltarea aplicațiilor si microserviciilor. Este foarte rapid datorita unui model de injecție de dependenta compile-time, însă comunitatea si ecosistemul sunt mai mici comparativ cu Spring.

Pentru partea de autentificare/înregistrare exista Okta care este un serviciu cloud de management al identității si accesului care oferă autentificare, gestionarea utilizatorilor, integrarea cu diverse aplicații si API-uri, însă acesta oferă mai putina flexibilitate pentru personalizare fata de KeyCloak.

Sau as fi putut implementa singur o logica a autentificării, însă aceasta necesita mai mult timp, pot apărea mai multe erori daca nu este implementata corect, KeyCloak oferă toate funcționalitățile implicit ceea ce face ușor și rapid implementarea lui.

4 SOLUTIA PROPUȘĂ

In acest capitol voi descrie soluția propusa cat si tehnologiile utilizate, legăturile dintre ele si arhitectura proiectului.

4.1 Backend

4.1.1 PostgreSQL

PostgreSQL este un sistem de gestionare a bazelor de date relaționale de tip open-source, este renumit pentru stabilitatea sa, extensibilitate si conformitatea cu standardele SQL. Este utilizat pe scara larga in diverse aplicații, atât in proiectele mici cat si in aplicații Enterprise complexe. In proiectul meu, PostgreSQL joaca un rol crucial in gestionarea datelor. Iată cum am utilizat principalele caracteristici ale PostgreSQL in aplicația mea:

1. Modelarea datelor: Am creat un model de date rațional pentru a reprezenta infrastructurile si ofertele de tehnologie. Am definit tabele, relații si constrângeri pentru a asigura integritatea si consistenta datelor. Anexa 1 – Reprezentarea Bazei de

date. Am creat tabele pentru infrastructura, uşer şi oferta. Aceste tabele includ toate informaţiile necesare pentru a putea lucra cu aceste entităţi. Si doua tabele pentru imaginile infrastructurilor si ofertelor. Aceste tabele sunt conectate datorita id-ului userului, cu ajutorul acestui id pot extrage informaţiile necesare despre infrastructurile si ofertele cu care un user poate interacţiona adică modifica sau şterge.

2. Interogarea bazei de date: Am folosit PostgreSQL pentru a realiza interogări SQL necesare pentru funcţionalităţile de căutare si filtrare a catalogului. Câteva exemple de interogare a bazei de date. A se vedea Figura 9.

```

1 usage
@Query("SELECT i FROM Infrastructure i WHERE i.user_id = :user_id AND i.private_status = true")
List<Infrastructure> findByUser_idOwner(Integer user_id);

2 usages
@Query("SELECT i FROM Infrastructure i WHERE i.user_id = :user_id")
List<Infrastructure> findByUser_id(Integer user_id);

no usages
@Query("SELECT i FROM Infrastructure i WHERE i.infrastructure_name = :name")
List<Infrastructure> findByName(String name);

1 usage
@Query("SELECT COUNT(i) FROM Infrastructure i WHERE i.private_status = true")
Long countByPrivate_status();

1 usage
@Query("SELECT i FROM Infrastructure i WHERE i.user_id = :user_id AND i.private_status = true")
List<Infrastructure> findByUser_idDetails(Integer user_id);

```

Figura 9. Exemple interogare baza de date

3. Integrarea cu Spring Data JPA: Am utilizat Spring Data JPA pentru a facilita interacţiunea cu baza de date PostgreSQL, simplificând operaţiunile CRUD si gestionarea entităţilor.

```

public interface UserRepository extends JpaRepository<User, Integer> {
    11 usages
    Optional<User> findByEmail(String email);
}

```

Figura 10. Folosirea JPA

Am folosit Flyway care este un instrument puternic si uşor de utilizat pentru gestionarea migraţiilor schemelor de baze de date. Acesta este des folosit in proiectele Spring pentru a automatizez crearea si actualizarea tabelelor din baza de date. In proiectul meu Flyway este folosit pentru am genera tabelele daca acestea nu sunt încă create in baza de date. Asta îmi permite automatizarea creării bazei mele de date, si uşurata modificării acesteia. Un exemplu de creare a unui table in baza mea de date A se vedea Anexa 14 – Câmpuri infrastructura în baza de date

Intellij IDEA permite generarea automata a scriptului SQL pentru crearea tabelor. Acesta permite in interfața lui conectarea la baza de date si la vizualizarea acesteia. Ceea ce permite ușor si intuitiv schimbarea unor tabele si generarea noilor scripturi SQL pentru folosirea Flyway.

4.1.2 Spring Framework

Spring Framework este unul dintre cele mai populare si puternice cadre de dezvoltare pentru aplicațiile Java, oferind un ecosistem vast si flexibil pentru construirea aplicațiilor robuste, scalabile și ușor de întreținut. Caracteristici Cheie ale Spring Framework pe care le-am folosit in aplicația mea:

1. Data Access Integration: Spring oferă suport pentru accesul la date, inclusive integrarea JPA, și soluții de date NoSQL.
2. Web Frameworks: Spring MVC si Spring WebFlux sunt cadre de lucru pentru dezvoltarea de aplicații web, ceea ce reprezinta si platforma mea. In aplicația mea am folosit Spring MVC care este orientat către aplicațiile sincrone. Un exemplu de utilizare in aplicația mea. A se vedea Figura 12.

```
@Controller
public class AddOfferPageController {

    @GetMapping("/addOffer")
    public String getAddInfraPage(Model model){
        Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
        if(authentication != null && authentication.isAuthenticated() &&
            !authentication.getPrincipal().equals("anonymousUser")){
            model.addAttribute("isLoggedIn", authentication.isAuthenticated());
        }
        return "addOffer";
    }
}
```

Figura 12. Exemplu Spring MVC

3. Spring Boot: Spring Boot simplifica configurarea si lansarea aplicațiilor Spring prin convenții si setări implicite, eliminând necesitatea configurațiilor XML complexe. Exemplu in aplicația mea. A se vedea Figura 13.

```
@SpringBootApplication
public class LicentaApplication {

    public static void main(String[] args) {
        SpringApplication.run(LicentaApplication.class, args);
    }

    @Bean
    public FreeMarkerConfigurationFactoryBean getFreeMarkerConfig() {
        FreeMarkerConfigurationFactoryBean bean = new FreeMarkerConfigurationFactoryBean();
        bean.setTemplateLoaderPath("classpath:/templates/");

        return bean;
    }
}
```

Figura 13. Exemplu Spring Boot

4. Security: Spring Security oferă un cadru de lucru pentru securitatea aplicațiilor, inclusiv autentificarea, autorizarea și protecție. În Exemplu de mai jos se poate vedea crearea unor restricții pentru adăugarea de oferte și infrastructuri pe platforma. Pentru a avea drepturi de adăugare un utilizator v-a fi nevoit să aibă un rol care se numește „Privilegiat”. Exemplu de folosire Figura 14.

```

    , BearerTokenAuthenticationFilter.class)
    .authorizeHttpRequests(authorizeHttpRequests -> authorizeHttpRequests
        .requestMatchers(🔒"/login").permitAll()
        .requestMatchers(🔒"/addInfrastructure").hasRole("privilegiat")
        .requestMatchers(🔒"/addOffer").hasRole("privilegiat")
        .requestMatchers(🔒"/auth").permitAll()
        .anyRequest().permitAll())

```

Figura 14. Exemplu Spring Security

Datorita modularității care li oferă Spring, modificarea claselor a interfețelor si altor componente deja create de către dezvoltator devine foarte accesibila. A se vedea Figura 15.

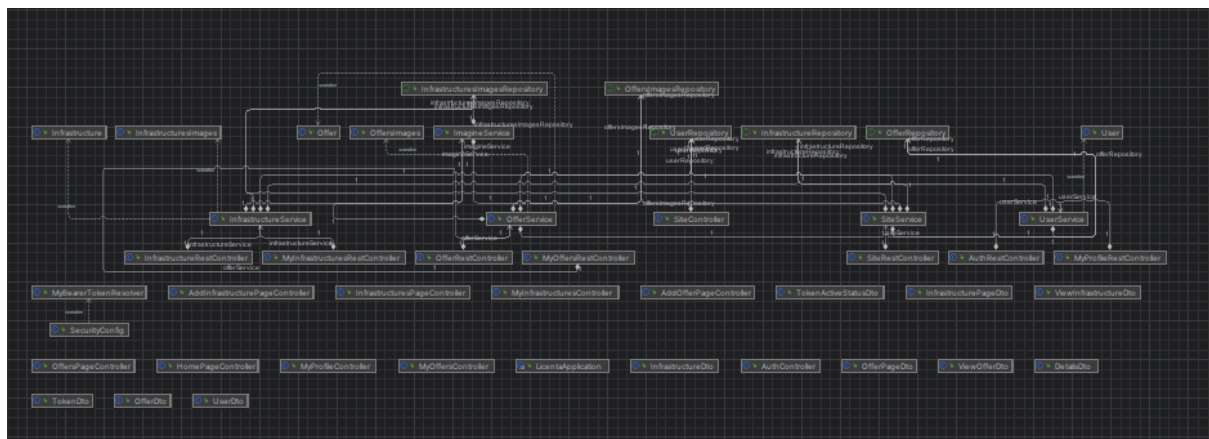


Figura 15. Diagrama Claselor

4.1.3 KeyCloak

Keycloak este o platforma open-source pentru gestionarea identității și accesului, care oferă servicii precum: Autentificare, autorizare, Securitate pentru aplicațiile web și serviciile API. Am utilizat KeyCloak pentru ușurința sa de implementare și de beneficiile care le aduce acesta. KeyCloak implementează specificațiile OpenID Connect și OAuth 2.0, oferind suport pentru autentificare bazată pe token-uri de autorizare, oferă și o interfață de administrație pentru gestionarea utilizatorilor, rolurilor și a permisiunilor. A se vedea Figura 16. Un flow de autentificare poate fi văzut în Anexa 2 – Autorizare Flow.

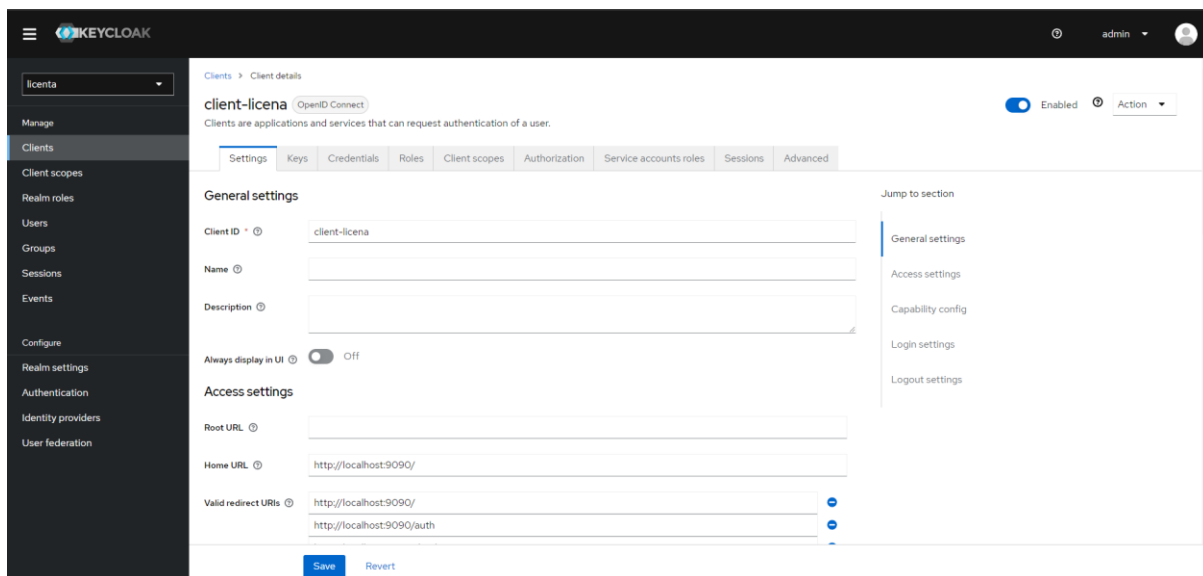


Figura 16. Interfața KeyCloack

Datorita posibilității de a asigura roluri utilizatorilor, ceea ce este oferit de KeyCloack, a făcut ușor implementarea posibilităților de adăugare Infrastructura si Oferta, astfel asignând un rol unui utilizator „Privilegiat” acesta primește permisiunile necesare de adăugare. Asignarea unui rol poate fi făcută direct din interfața KeyCloack, dar și din aplicație. A se vedea Figura 17.

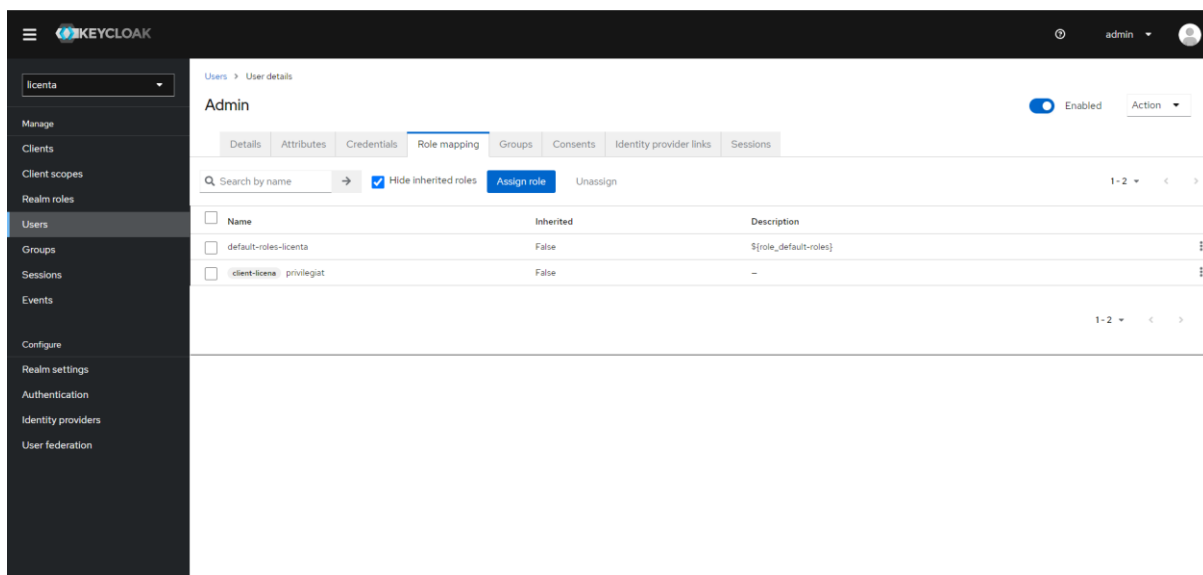


Figura 17. Utilizator cu rol „Privilegiat”

KeyCloack oferă și posibilitatea de a verifica e-mailul unui utilizator. Atunci când un utilizator își creează un cont pe platforma aceasta va fi nevoit să introducă un e-mail valid. Aceasta a fost adăugată din necesitatea de a asigura rolul privilegiat la utilizatorii care fac parte din UNSTBP automat. Adică dacă utilizatorul va avea un e-mail @upb.ro acesta va avea posibilitatea de adăugare automat. E-mailul primit atunci când un utilizator se va înregistra pe platforma InfraOff se poate vedea în Figura 18.

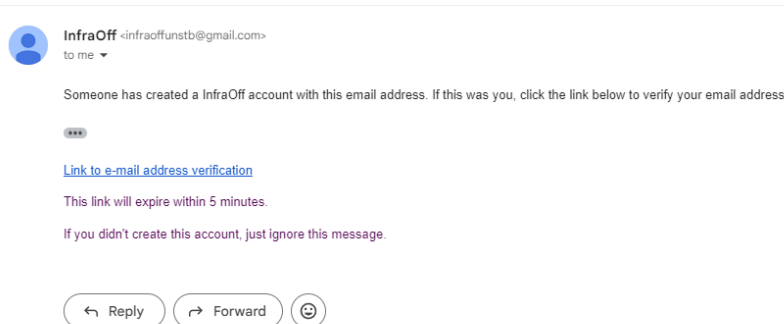


Figura 18. E-mail primit la înregistrare

4.2 FrontEnd

Pentru implementarea interfeței utilizatorului in aplicația mea am folosit HTML(HyperText Markup Language), CSS(Cascading Style Sheets) și JavaScript.

4.2.1 HTML

Html este limbajul standard folosit pentru crearea si structurarea conținutului unei pagini. Am utilizat HTML pentru a defini structura Logica a paginii. De asemenea, am folosit etichetele HTML pentru a marca diferite elemente cum ar fi titluri, imagini, legături. A se vedea Figura 19.

```
<li><a href="/" class="active">Acasa</a></li>
<li><a href="/offers">Oferte</a></li>
<li><a href="/contact">Contact</a></li>
<li><a href="/infrastructures">Infrastructuri</a></li>
```

Figura 19. Exemplu legături HTML

Prin intermediul 'href', am creat legături către diferite pagini si resurse din aplicația mea Spring, utilizând endpointurile definite in aplicație.

4.2.2 CSS

Css este utilizat in aplicația mea pentru stilizarea si formatarea paginilor web create cu HTML. Am folosit pentru a adauga stiluri, culori, fonturi si aspecte vizuale la elemente HTML. A se vedea Figura 20.

```
.blur-bg {
  width: 100%;
  height: 100%;
  background: url(../images/about-content-bg.jpg) no-repeat center center fixed;
  background-size: cover;
  filter: blur(8px) brightness(80%);
  position: absolute;
  left: 0;
  top: 0;
  z-index: 1;
}
```

Figura 20. Stiluri aplicate cu CSS

4.2.3 JavaScript

JavaScript este un limbaj de programare utilizat pentru a face paginile web interactive si dinamice. In proiectul meu am utilizat JavaScript pentru a adaugă funcționalități interactive, cum ar fi animații, manipularea evenimentelor si comunicarea cu serverul.

Comunicarea cu serverul in JavaScript am făcut-o cu ajutorul ajax. Aceasta permite actualizarea parțiala a conținutului unei pagini, fără a fi necesara o încărcare completă a acesteia. A se vedea 21.

```
$.ajax({
  url: 'http://localhost:9090/modifyInfra',
  type: 'PUT',
  data: formData,
  processData: false, // Important
  contentType: false, // Important
  success: function(response) : void {
    window.location.href = "http://localhost:9090/myInfrastructures";
  }
});
```

Figura 21. Trimiterea unei cereri PUT către server

4.3 Arhitectura Aplicației

Arhitectura platformei mele consta din câteva componente:

- Backend: am folosit Java si Spring Framework. Aici se primesc datele introduse de utilizator, se salvează, modifica sau șterg din baza de date, se trimit datele înapoi către utilizator.
- Frontend: realizat cu HTML, CSS si JavaScript. Partea cu care utilizatorul interacționează direct, se face legătura cu backend-ul, se extrag datele si se afișează datele către utilizator.
- Serviciul de autentificare realizat cu KeyCloak: aceasta este logica de autentificare si autorizare.
- Baza de date: implementata cu PostgreSQL, aici se stochează toate datele adăugate de către utilizatori.

Interacțiunea dintre componentele aplicației:

1. Front-end Client (Browser):

- Utilizatorul deschide aplicația într-un browser.
- Cand utilizatorul dorește sa se autentifice in cont este redirectionat către pagina de autentificare a Keycloak
- După autentificare, clientul primește un token JWT.

2. Serverul aplicației (Spring Boot):
 - Primește cererile HTTP de la client.
 - Verifica token-ul JWT primit de la client folosind Keycloak pentru a valida identitatea utilizatorului și permisiunile lui.
 - Accesează și manipulează datele în baza de date PostgreSQL folosind JDBC și Spring Data JPA
 - Returnează răspunsuri către client cu datele solicitate.
3. Keycloak Server:
 - Gestionează autentificarea și autorizarea utilizatorilor.
 - Furnizează un formular de autentificare și înregistrare.
 - Returnează un token JWT către client după autentificare, care poate fi folosit pentru a accesa resursele pe serverul aplicației.
4. Database Server (PostgreSQL):
 - Primește și execută interogările SQL trimise de către server pentru a accesa și modifica datele.
 - Returnează rezultatele interogărilor către serverul aplicației pentru a le procesa și trimite înapoi către client.

A se vedea Anexa 3 – Deployment Diagram.

4.3.1 Arhitectura Frontend

Pentru partea de Frontend am optat pentru o arhitectura bazată pe componente. O arhitectura bazată pe componente oferă o structură și o organizare clară a componentelor frontend. Aplicația este organizată în jurul componentelor, fiecare componentă are un scop și o responsabilitate bine definite. Componentele fiind decuplate una față de alta modificările aduse unei componente nu afectează și alte componente. Asta face ca atunci când o componentă nu lucrează corect, restul aplicației să funcționeze cum este așteptat.

Structura componentelor în proiectul meu:

- Folderul templates: Aici se afla toate fișierele HTML, fișierele fac legătura între toate componentele Frontend. Adică atunci când se va apela un endpoint, un @Controller mă va redirecționa pe o pagină HTML, de aici se vor executa toate scripturile necesare paginii deschise, se vor pune stilurile care sunt definite în fișierele CSS.
- Folderul js: Aici se afla fișierele JavaScript. În mare parte scripturile le-am folosit pentru a cere informații despre anumite entități cum ar fi: Utilizator, Oferta, Imagini, Infrastructura.
- Folderul css: Aici se găsesc toate fișierele CSS, toate stilurile sunt definite în interiorul acestor fișiere.
- Folderul images: Imaginile care sunt folosite static pe platforma, ele nu se modifică. Cele care sunt asociate infrastructurilor și ofertelor se afla în baza de date.

4.3.2 Arhitectura Backend

Spring oferă o Arhitectura modulara, modularitatea este promovata prin organizarea pachetelor si a componentelor in cadrul aplicației. Arhitectura Spring Figura 22.

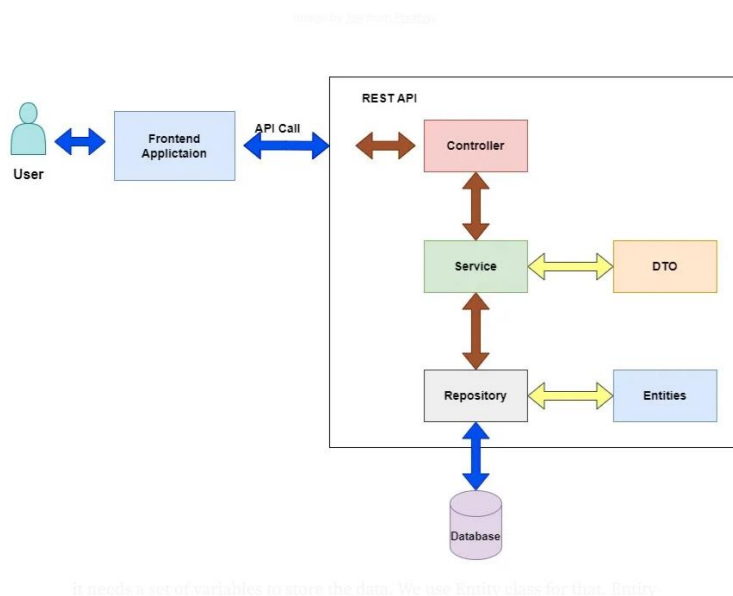


Figura 22. Arhitectura Spring(Preluata din [3])

Proiectul meu este organizat pe baza arhitecturii Model-View-Controller (MVC), împărțind aplicația in straturi distincte pentru o mai buna separare a responsabilităților:

- Controlere : Acestea se ocupa de gestionarea solicitărilor HTTP de la client, apelarea serviciilor si returnarea răspunsurilor. In figura 23 voi arata un exemplu de controller folosit in aplicația mea.

```
@Controller
public class HomeController {

    @GetMapping("/{path}")
    public String getHomePage(Model model){
        Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
        if(authentication != null && authentication.isAuthenticated() &&
            !authentication.getPrincipal().equals("anonymousUser")){
            model.addAttribute( attributeName: "isLoggedIn", attributeValue: "");
        }
        return "index";
    }
}
```

Figura 23. HomeController

Acest controller gestionează cererile HTTP pentru pagina de start a aplicației. Adnotația @Controller indica faptul ca acesta este un Controller, si este responsabil pentru gestionarea cererilor Web. Metoda getHomePage adnotata cu @GetMapping("/{path}"), gestionează cererile GET către rădăcina aplicației. Model model este utilizat pentru adăugarea atributelor in cazul meu verific daca avem un utilizator logat.

- Servicii: Conțin logica de business a aplicației. Acestea preiau cereri de la RestControllers, procesează datele și apelează repository-urile pentru accesul la baza de date. Se poate vedea în Figura 24 apelarea unui serviciu de către RestController.



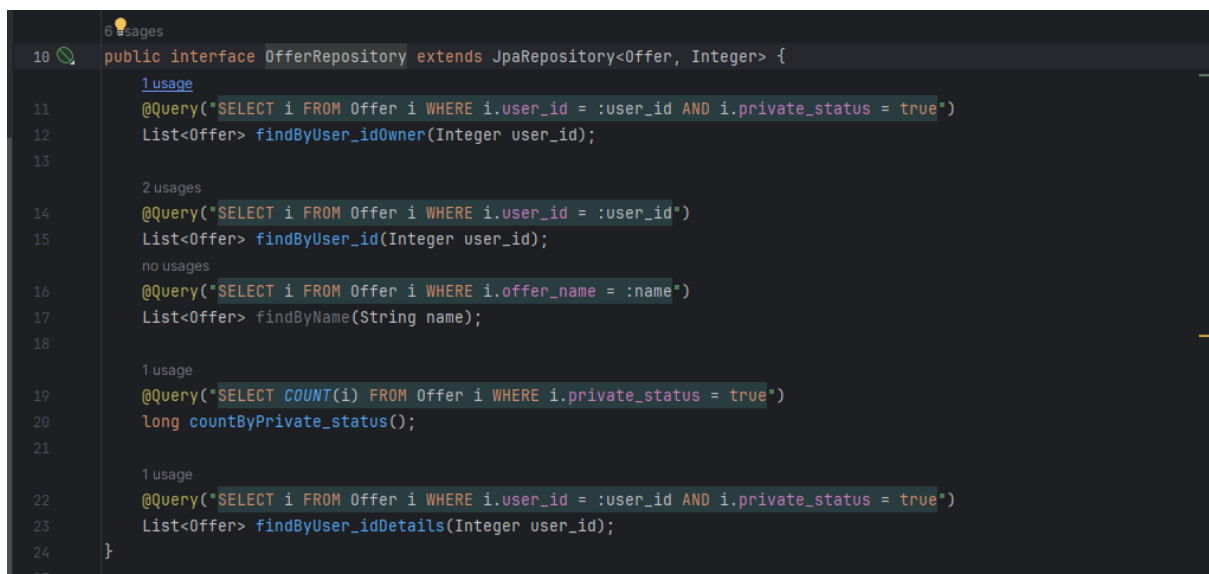
```

1 package com.licenta.Licenta.controllers;
2
3 > import ...
4
5
6
7
8
9
10
11 @RestController
12 public class SiteRestController {
13     6 usages
14     private final SiteService siteService;
15
16     15 > public SiteRestController(SiteService siteService) { this.siteService = siteService; }
17
18
19     @GetMapping("/api/getDetails")
20     20 > public DetailsDto getDetails() { return siteService.getDetails(); }
21
22
23
24     @GetMapping("/api/viewUserOwner/{id}")
25     25 > public UserDto getUserOwner(@PathVariable("id") Integer id) { return siteService.getViewUserOwner(id); }
26
27
28

```

Figura 24. Apelarea serviciilor

- Repository-uri: Gestionează operațiunile de acces la baza de date, cum ar fi interogările și manipularea datelor în baza mea de date PostgreSQL, utilizând Spring Data JPA. Un exemplu avem în Figura 25.



```

10 6 usages
11 public interface OfferRepository extends JpaRepository<Offer, Integer> {
12     1 usage
13     @Query("SELECT i FROM Offer i WHERE i.user_id = :user_id AND i.private_status = true")
14     List<Offer> findByUser_idOwner(Integer user_id);
15
16     2 usages
17     @Query("SELECT i FROM Offer i WHERE i.user_id = :user_id")
18     List<Offer> findByUser_id(Integer user_id);
19
20     no usages
21     @Query("SELECT i FROM Offer i WHERE i.offer_name = :name")
22     List<Offer> findByName(String name);
23
24     1 usage
25     @Query("SELECT COUNT(i) FROM Offer i WHERE i.private_status = true")
26     long countByPrivate_status();
27
28     1 usage
29     @Query("SELECT i FROM Offer i WHERE i.user_id = :user_id AND i.private_status = true")
30     List<Offer> findByUser_idDetails(Integer user_id);
31
32 }

```

Figura 25. Repository exemplu

- Entități : Modelează structura datelor stocate în baza de date. Fiecare entitate corespunde unei tabele din PostgreSQL și este mapată folosind JPA. În aplicația mea există entități pentru: Infrastructura, Oferta și User atât și pentru tabelele care conțin imaginile infrastructurilor cât și al ofertelor.

- DTO : Este un pachet care stochează date de comunicare între Frontend și Backend. A se vedea Figura 26.

```

5      12 usages
6      @Builder
7      public class DetailsDto {
8          @JsonProperty
9          public String infrastructures_count;
10         @JsonProperty
11         public String offers_count;
12     }
13

```

Figura 26. DetailsDto

4.4 Adăugare Infrastructura/Oferă de tehnologie

Pentru adăugarea acestora soluția mea propusă a fost de a crea câmpurile necesare pentru a descrie complet o infrastructură și ofertă. Pentru a adăuga ceva pe platforma utilizatorul va fi nevoit să fie autentificat cu un cont creat cu o adresă de e-mail @upb.ro sau să primească drepturi făcând o cerere pe e-mail în pagina de contact. Pagina de contact a platformei poate fi văzută în Anexa 4 – Pagina de contact. La adăugarea unei infrastructuri utilizatorul va introduce toate datele necesare cât și o locație care o poate alege direct pe harta sau poate scrie coordonatele singur. Pagina de adăugare a unei infrastructuri poate fi văzută în Anexa 5 – Pagina de adăugare a unei infrastructuri. Pe platforma mea un utilizator va putea face publică sau privată o infrastructură sau ofertă adăugată. Pagina de adăugare a unei oferte poate fi văzută în Anexa 6 – Pagina de adăugare a unei oferte. Pentru a înțelege mai bine cum se adăugă o infrastructură și o ofertă pe platformă am creat o digramă de activitate care poate fi văzută aici Anexa 7 – Diagrama Activitate Adăugare Ofertă/Infrastructură.

4.5 Modificare/Ștergere Infrastructura

Pentru Modificarea și Ștergerea unei Oferte sau Infrastructuri soluția mea propusă a fost de a crea două pagini unde utilizatorul va putea accesa toate ofertele și infrastructurile sale Figura 27.



Figura 27. Infrastructurile și ofertele mele

Utilizatorul la accesarea acestora v-a putea vedea toate infrastructurile si ofertele adăugate de el, unde v-a avea câteva opțiuni pe care le poate accesa: Detalii Infrastructură/Ofertă, Modifica Infrastructura/Oferta si Șterge Infrastructura/Oferta Figura 28.

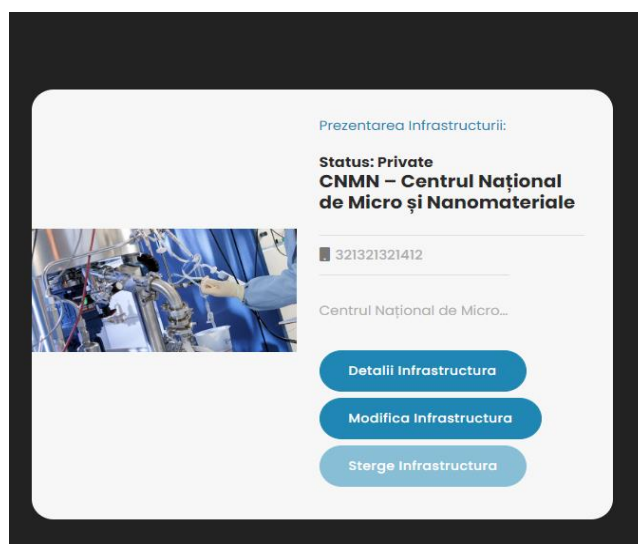


Figura 28. Infrastructură

Utilizatorul v-a putea vedea daca aceasta este privata sau nu, si v-a putea face modificările unei infrastructuri sau oferte accesând opțiunea de modifica. Toate detaliile care deja utilizatorul le-a introdus vor rămâne disponibile, acesta poate doar sa le modifice sau sa le rescrie total, a se vedea Anexa 8 – Modificare Infrastructura.

4.6 Vizualizare Toate infrastructurile/ofertele si căutarea lor.

Soluția mea propusa pentru aceasta a fost de a crea doua pagini unde utilizatorul v-a putea vedea toate ofertele si infrastructurile care sunt publice pe platforma, Figura 29.



Figura 29. Oferte/Infrastructuri

Accesând aceste pagini utilizatorul v-a putea vedea toate ofertele/infrastructurile disponibile cat si câteva detalii despre ele: Denumirea, descrierea, data publicării, telefon de contact. Se poate vedea in Figura 30.

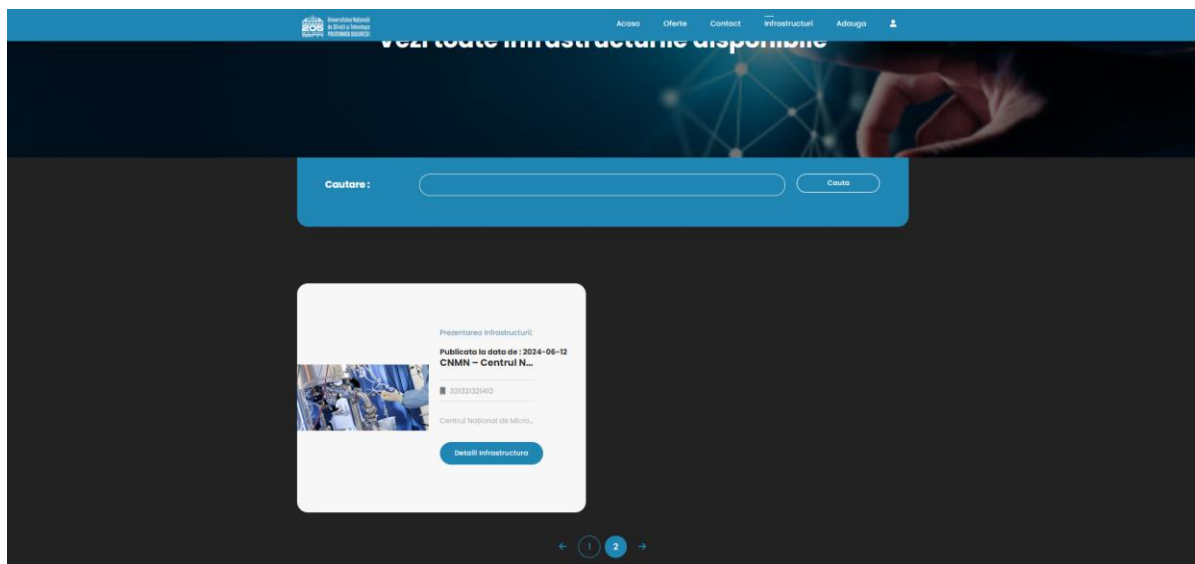


Figura 30. Pagina Infrastructuri

Pe aceasta pagina utilizatorul v-a putea si caută infrastructura sau oferta care dorește. Căutarea acestora se poate face după: Nume, cuvinte cheie și e-mail al utilizatorului care a adăugat.

4.7 Detalii Infrastructura/Oferta

Soluția mea propusa a fost ca atunci cand un utilizatorul v-a accesa pagina de vizualizare a tuturor ofertelor si infrastructurilor acesta v-a avea si un buton la fiecare se poate vedea in Figura 30. La accesarea acestuia, utilizatorul v-a fi redirectionat pe pagina cu detalii. Figura 31

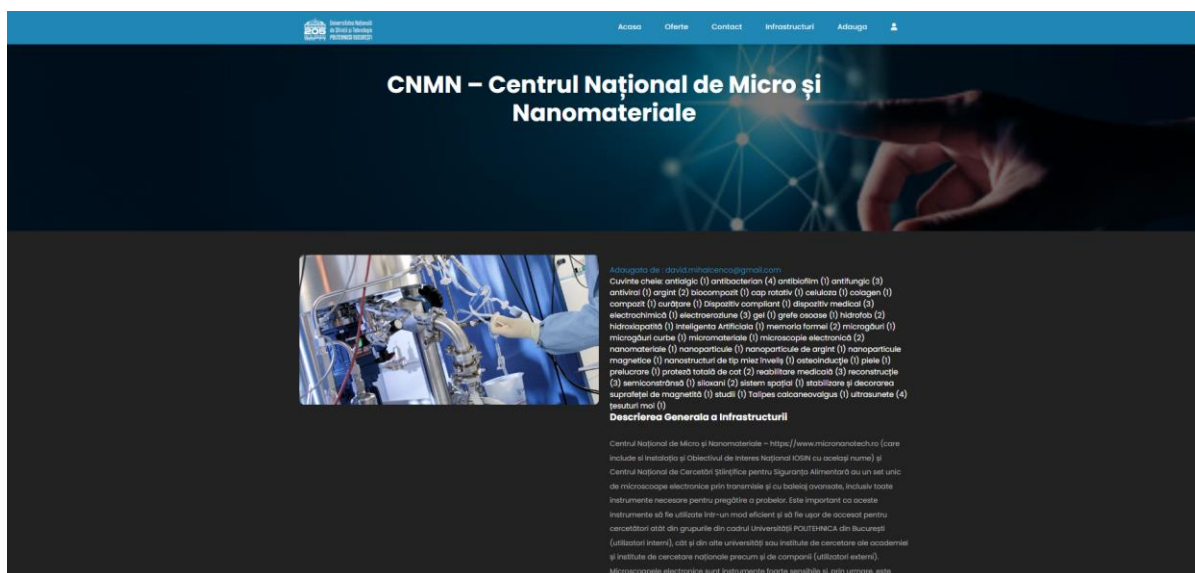


Figura 31. Detalii Infrastructura

Aici el v-a vedea toate informațiile adăugate de către un cercetător cat si o harta la infrastructuri cu locația exacta a acesteia. Pe aceasta pagina utilizatorul v-a putea accesa si utilizatorul care a adăugat aceasta oferta sau infrastructura. Pe platforma mea fiecare utilizator v-a avea un cont in care vor avea câteva informații despre el. Pe aceasta pagina

utilizatorul v-a putea vedea cate infrastructuri si oferte au fost adăugate de către el si v-a putea accesa ofertele si infrastructurile altor utilizatori daca a accesat pagina altui utilizator. V-a putea sa-si adauge o poza de profil, si v-a putea da click de e-mail pentru a începe o conversație cu utilizatorul accesat. Anexa 9 - Profil Utilizator

5 DETALII IMPLEMENTARE

5.1 Implementare Login/Logout

Cum am menționat anterior pentru autentificarea si înregistrarea unui utilizator am utilizat KeyCloak. In aceasta secțiune voi menționa cum se fac acestea si cum se salvează utilizatorii in baza de date si cum i se asignează rol pe partea de backend.

Pentru logarea in aplicație am creat o metoda numita login care gestionează cererea GET către endpoint-ul /login. A se vedea Figura 32. Aceasta returnează un obiect de tip RedirectView care redirecționează utilizatorul către URL. URL-ul practic este endpoint-ul de autentificare al KeyCloak pentru realm-ul meu care are numele de „licenta”. Acest URL are si câțiva parametri:

- `http://localhost:8080/realms/licenta/protocol/openid-connect/auth`: Acesta este endpoint-ul de autentificare al Keycloak pentru realm-ul "licenta".
- `client_id=client-licena`: Parametrul `client_id` specifică ID-ul clientului înregistrat în Keycloak.
- `response_type=code`: Specifică faptul că se folosește codul de autorizare (Authorization Code Flow).
- `scope=openid`: Definirea scope-ului cerut, în acest caz openid, care este minimul necesar pentru autentificare OpenID Connect.
- `redirect_uri=http://localhost:9090/auth`: URL-ul la care utilizatorul va fi redirecționat după autentificare.
- `state=auth`: Un parametru de stare pentru a preveni atacurile CSRF și pentru a menține starea între cereri.

```
@GetMapping("/login")
public RedirectView login() {
    return new RedirectView("http://localhost:8080/realms/licenta/protocol/openid-connect/auth?client_id=client-licena&res");
}
```

Figura 32. Login

Aceasta metoda este utilizata pentru a iniția un flux de autentificare OpenID Connect, redirecționând utilizatorul către serverul de autentificare Keycloak. După ce utilizatorul se va autentifica, acesta va fi redirecționat înapoi la URL-ul specificat in `redirect_uri` adică: `redirect_uri=http://localhost:9090/auth&state=auth` cu un cod de autorizare, pe care îl voi folosi pentru a obține un access si refresh token.

Metoda care gestionează cererea /auth adică `redirect_uri` de mai sus se numește auth. Aceasta metoda se ocupa de procesarea codului de autorizare primit de la serverul de

autentificare Keycloak, cu ajutorul lui se primește un acces si un refresh token, si apoi se salvează acestea in cookie HTTP.

Pentru a primi token-urile creez o cerere de tip MultiValueMap, care continue parametrii cererii POST către endpoint-ul de token al Keycloak. A se vedea Figura 33. Parametrii include:

- client_id: ID-ul clientului.
- client_secret: Secretul clientului.
- grant_type: Tipul grant-ului, în acest caz authorization_code.
- code: Codul de autorizare primit ca parametru în metodă.
- redirect_uri: URL-ul de redirecționare, care trebuie să fie același ca cel folosit la inițierea autentificării.
- scope: Scope-ul cerut, în acest caz openid.

```
public RedirectView auth(@RequestParam String code, HttpServletResponse rsp){
    MultiValueMap<String, String> params = new LinkedMultiValueMap<>();
    params.add("client_id", "client-licena");
    params.add("client_secret", "aKgAUuXq76vwEZ9NDkXCLkI96nkITVjC");
    params.add("grant_type", "authorization_code");
    params.add("code", code);
    params.add("redirect_uri", "http://localhost:9090/auth");
    params.add("scope", "openid");
```

Figura 33. Parametrii pentru cererea token-ului

După se face cererea pentru token către endpoint-ul Keycloak, si se obțin token-ul de access si refresh pentru care mi-am create un Dto numit TokenDto. Creez cookies pentru access_token si refresh_token. Le adaugă in răspunsul HTTP pentru a le trimite Clientului. A se vedea Figura 34.

```
HttpEntity<MultiValueMap<String, String>> entity = new HttpEntity<>(params, new HttpHeaders());
ResponseEntity<TokenDto> accessTokenDto = new RestTemplate().exchange(url: "http://localhost:8080/realms/licenta/protocol

String accessToken = accessTokenDto.getBody().accessToken;
String refreshToken = accessTokenDto.getBody().refreshToken;
Cookie accessTokenCookie = new Cookie(name: "access_token", accessToken);
accessTokenCookie.setHttpOnly(true);
accessTokenCookie.setPath("/");
accessTokenCookie.setSecure(true);
Cookie refreshTokenCookie = new Cookie(name: "refresh_token", refreshToken);
refreshTokenCookie.setHttpOnly(true);
refreshTokenCookie.setPath("/");
refreshTokenCookie.setSecure(true);

rsp.addCookie(accessTokenCookie);
rsp.addCookie(refreshTokenCookie);

userService.saveUserJWT(accessTokenDto.getBody().accessToken);

return new RedirectView(url: "/");
}
```

Figura 34. Salvarea token in cookies

În ceea ce privește actualizarea acestor tokenuri, am metoda `defaultSecurityFilterChain`, în această metodă există un filter de tip `OncePerRequestFilter` care se execută o singură dată pe cerere. A se vedea Figura 35. Logica filtrului:

- Se obțin cookies `access_token` și `refresh_token`.
- Dacă `access_token` există dar este invalid, și `refresh_token` există, atunci se va obține un nou `access_token` care va fi setat în cookies
- Se apelează `filterChain.doFilter` pentru a continua procesarea cererii prin lanțul de filtre.

```
@Bean
public SecurityFilterChain defaultSecurityFilterChain(HttpSecurity http) throws Exception {
    http
        .addFilterBefore((OncePerRequestFilter) (request, response, filterChain) -> {
            Cookie access_token_cookie = WebUtils.getCookie(request, name: "access_token");
            Cookie refresh_token_cookie = WebUtils.getCookie(request, name: "refresh_token");
            if (access_token_cookie != null && !isValid(access_token_cookie.getValue()) && refresh_token_cookie != null) {
                try {
                    getAndSetNewAccessToken(response, access_token_cookie.getValue(), refresh_token_cookie.getValue());
                } catch (Exception e) {
                    Cookie accessTokenCookie = new Cookie(name: "access_token", value: null);
                    accessTokenCookie.setHttpOnly(true);
                    accessTokenCookie.setPath("/");
                    accessTokenCookie.setSecure(true);
                    accessTokenCookie.setMaxAge(0);

                    Cookie refreshTokenCookie = new Cookie(name: "refresh_token", value: null);
                    refreshTokenCookie.setHttpOnly(true);
                    refreshTokenCookie.setPath("/");
                    refreshTokenCookie.setSecure(true);
                    refreshTokenCookie.setMaxAge(0);

                    response.addCookie(accessTokenCookie);
                    response.addCookie(refreshTokenCookie);
                }
            }
        }, filterChain.doFilter(request, response);
}
```

Figura 35. SecurityFilter

Metoda care este apelată pentru generarea unui nou `access_token` cu ajutorul `refresh_token` poartă numele de `getAndSetNewAccessToken`. A se vedea Figura 36. Pentru a primi un `access_token` nou setez parametrii cererii pentru KeyCloak aceasta conține:

- `Client_id` : ID-ul clientului.
- `Client_secret`: Secretul clientului.
- `Grant_type`: Tipul grantului, în acest caz `refresh_token`
- `Refresh_token` : Token-ul de refresh primit ca parametru din metoda anterioară.

```
MultiValueMap<String, String> params = new LinkedMultiValueMap<>();
params.add("client_id", "client-licena");
params.add("client_secret", "aKgAUuXq76vWEZ9NDkXCLkI96nkITVjC");
params.add("grant_type", "refresh_token");
params.add("refresh_token", refresh_token);
```

Figura 36. Parametrii cererii pentru a genera `access_token` din `refresh_token`

Utilizez `RestTemplate` pentru a trimite cererea POST la endpoint-ul de token al serverului OAuth. Se vor primi un nou `access_token` și `refresh_token` care la fel sunt setate la Cookies.

A se vedea Figura 37.

```
HttpEntity<MultiValueMap<String, String>> entity = new HttpEntity<>(params, new HttpHeaders());
ResponseEntity<TokenDto> response = null;
response = new RestTemplate().exchange(url: "http://localhost:8080/realms/licenta/protocol/openid-connect/token", HttpM

if (response.getStatusCode() == HttpStatus.OK) {
    TokenDto tokenResponse = response.getBody();
    if (tokenResponse != null) {
        Cookie accessTokenCookie = new Cookie( name: "access_token", tokenResponse.accessToken);
        accessTokenCookie.setHttpOnly(true);
        accessTokenCookie.setPath("/");

        Cookie refreshTokenCookie = new Cookie( name: "refresh_token", tokenResponse.refreshToken);
        refreshTokenCookie.setHttpOnly(true);
        refreshTokenCookie.setPath("/");

        httpServletResponse.addCookie(accessTokenCookie);
        httpServletResponse.addCookie(refreshTokenCookie);
    }
}
```

Figura 37. Setarea access_token nou si refresh_token

Pentru logout la fel ca la și login se face o cerere către Keycloak doar că pentru alt endpoint, acesta fiind numit “/dauth” , practic această metodă îmi setează pe null access_token si refresh_token. A se vedea Figura 38.

```
@GetMapping("/dauth")
public RedirectView deauth(HttpServletResponse rsp){
    Cookie accessTokenCookie = new Cookie( name: "access_token", value: null);
    accessTokenCookie.setHttpOnly(true);
    accessTokenCookie.setPath("/");
    accessTokenCookie.setSecure(true);
    accessTokenCookie.setMaxAge(0);

    Cookie refreshTokenCookie = new Cookie( name: "refresh_token", value: null);
    refreshTokenCookie.setHttpOnly(true);
    refreshTokenCookie.setPath("/");
    refreshTokenCookie.setSecure(true);
    refreshTokenCookie.setMaxAge(0);

    rsp.addCookie(accessTokenCookie);
    rsp.addCookie(refreshTokenCookie);

    return new RedirectView( url: "/");
}
```

Figura 38. Dauth metod

Pentru salvarea utilizatorilor in baza de date PostgreSQL se folosesc o metoda din UserService cu numele SaveUserJwt, apelata din metoda auth, ca parametru primește un JWT(JSON Web Token) care este access_token-ul utilizatorului.

In aceasta funcție se parsează JWT pentru a extrage anumite informații despre utilizator in cazul meu e-mail. Fac verificarea daca acest email nu este null, si apelez metoda saveUser, in care se face verificarea daca acest user nu exista deja in baza mea de date, daca acesta nu exista se salvează datele despre el. A se vedea Figura 39

```

public void saveUser(String email) {
    Optional<User> optionalUser = userRepository.findByEmail(email);
    if (optionalUser.isPresent())
        return;

    userRepository.save(new User(user_id: null, email, email, image: null));
}

```

Figura 39. Salvare user

Pentru a asigura un rol utilizatorului se face verificarea e-mail-ului, dacă se termina cu @upb.ro și acesta este verificat atunci i se va asigura rolul de privilegiat. Adică v-a putea adăuga oferte și infrastructuri pe platforma.

Pentru asignarea rolului se face o cerere pentru a primi un access_token folosind client credentials grant type. A se vedea Figura 40.

```

assert email != null;
if (email.endsWith("@upb.ro") && jwt.getJWTClaimsSet().getBooleanClaim( name: "email_verified")) {
    MultiValueMap<String, String> params = new LinkedMultiValueMap<>();
    params.add("client_id", "client-licenta");
    params.add("client_secret", "aKgAUuXq76vWEZ9NDkXCLkI9onkITVjC");
    params.add("grant_type", "client_credentials");

    HttpEntity<MultiValueMap<String, String>> entity = new HttpEntity<>(params, new HttpHeaders());
    ResponseEntity<TokenDto> accessTokenDto = new RestTemplate().exchange( url: "http://localhost:8080/realms/licenta/protocol

```

Figura 40. Cerere access_token

După ce acesta a fost obținut se trimite o cerere POST la endpoint-ul Keycloak pentru a asigura rolul privilegiat utilizatorului. A se vedea Figura 41.

```

ResponseEntity<TokenDto> accessTokenDto = new RestTemplate().exchange( url: "http://localhost:8080/realms/licenta/protocol
if (accessTokenDto.getBody().accessToken != null) {
    String x = "{\n" +
        "    {\n" +
        "        \"id\": \"690f8143-8546-401b-8e1f-d37c65d818f6\", \n" +
        "        \"name\": \"privilegiat\", \n" +
        "        \"description\": \"\", \n" +
        "        \"composite\": false, \n" +
        "        \"clientRole\": true, \n" +
        "        \"containerId\": \"bf4e06c3-2ced-45b4-8533-c130200a6c0c\" \n" +
        "    } \n" +
        "}";

    HttpHeaders httpHeaders = new HttpHeaders();
    httpHeaders.add( headerName: "Authorization", headerValue: "Bearer " + accessTokenDto.getBody().accessToken);
    httpHeaders.add( headerName: "Content-Type", headerValue: "application/json");

    HttpEntity<String> entity1 = new HttpEntity<>(x, httpHeaders);
    ResponseEntity<String> response = new RestTemplate().exchange( url: "http://localhost:8080/admin/realms/licenta/user

```

Figura 41. Asignarea rolului privilegiat

5.2 Adăugarea unei infrastructuri/oferte

Cum am vorbit anterior pe platforma mea la adăugarea unei oferte sau infrastructuri utilizatorul v-a completa câteva câmpuri deja predefinite, unde v-a pune detaliile despre infrastructura sau oferta adăugată. Mai jos voi vorbi despre adăugarea unei infrastructuri întrucât adăugarea de infrastructura și oferta este aproximativ același lucru, însă la infrastructura există și o hartă unde utilizatorul v-a alege și locația acesteia.

Pentru adăugare unei infrastructuri mi-am făcut o pagina HTML unde sunt definite câteva câmpuri : Name, Lat, Lng, Key_words, Description, Benefits, Access-info, The-specification, Number, Email, image. Aceste câmpuri vor fi completate de utilizator. A se vedea Anexa 10 – Câmpurile definite pentru infrastructura HTML. După completarea acestora a fost definit si un buton “Adaugă Infrastructura”, acest buton are si un id “submit-button-infra”.

Pentru locația de pe harta utilizatorul are doua posibilități, de a o introduce singur in câmpurile definite, sau sa dea click pe harta si se v-a extrage locația adică longitudinea si latitudinea. Pentru harta am folosit google maps for developers. Pentru a face harta interactiva se folosește JavaScript in care am un Listener peste harta. Cand utilizatorul v-a da click pe harta scriptul meu JavaScript v-a extrage lat si lng si v-a pune un marker pe harta. A se vedea Figura 42. Iar cu ajutorul funcției ViewMapInfra acestea se vor salva si in câmpurile definite pe platforma. A se vedea Figura 43

```
map.addListener({callback: "click", (mapsMouseEvent) : void => {  
  
    const latLng = mapsMouseEvent.latLng;  
    const lat = latLng.lat();  
    const lng = latLng.lng();  
  
    ViewMapInfra(lat, lng);  
  
    if (marker) {  
        marker.setPosition(latLng);  
    } else {  
        marker = new google.maps.Marker({  
            position: latLng,  
            map: map,  
        });  
    }  
});
```

Figura 42. Extragerea Lat si Lng

```
function ViewMapInfra(lat,lng) : void {  
    const latInput : Element = document.querySelector(selectors: '.Lat');  
    const lngInput : Element = document.querySelector(selectors: '.Lng');  
    latInput.value = lat;  
    lngInput.value = lng;  
}
```

Figura 43 Salvarea in câmpurile definite a Lat si Lng

După ce toate datele au fost completate si utilizatorul v-a da click pe buton am un script in JavaScript care v-a trimite toate datele inserate de pe platforma in forma de InfrastructureDto către server. Acestea sunt trimise in forma de formData acesta este un obiect din JavaScript utilizat pentru a construi seturi de perechi. Trimiterea pozei si a restul datelor se face aparte pentru a putea extrage poza mai ușor si de a introduce in tabelul cu imagini infrastructuri. Pentru a face un api call am folosit ajax, am făcut o cerere POST întrucât eu vreau sa adaug pe aceasta in baza de date, endpoint-ul care îmi va recepționa acest call este „/addInfra”. A se vedea Anexa 11 – Cererea POST pentru adăugare infrastructura

Metoda addInfraDataBase care este într-un rest controller îmi apelează o metoda din infrastructureService care răspunde de adăugarea infrastructurii in baza de date. Aceasta

primește ca parametrii imaginea si InfrastructureDto care conține toate informațiile despre infrastructura. A se vedea Figura 44

```
@PostMapping(value = "/addInfra", consumes = MediaType.MULTIPART_FORM_DATA_VALUE)
public void addInfraDataBase(@RequestPart("InfrastructureDto") InfrastructureDto infrastructureDto,
                             @RequestPart("image") MultipartFile image){
    infrastructureService.save(infrastructureDto, image);
}
```

Figura 44. addInfraDataBase

Save este metoda care salvează in baza de date. In general ea se ocupa de crearea unui Obiect nou de tip Infrastructura si un obiect nou de tip Infrastructure Image, in care se setează datele venite ca parametru. A se vedea Anexa 12 – Metoda save din infrastructureService

Pentru a ști data cand aceasta s-a adăugat folosesc LocalDate.now() care îmi întoarce data curenta, la fel aceasta este salvata in infrastructura pentru a fi afisata pe platforma.

Imaginea este salvata cu id generat automat, însă are si câmpul care este id-ul infrastructurii pentru a ști care imagine corespunde infrastructurii. Imaginea este salvata in forma de Base64. Care apoi este transformata înapoi in imaginea la afișarea s-a.

Pentru a afla cine a adăugat aceasta, folosesc Spring Security si JSON Web Tokens pentru a afla ce utilizator este autentificat. Mai exact, extrag adresa de e-mail a utilizatorului logat din token-ul JWT, caut in baza de date utilizatorul și obțin ID-ul utilizatorului. Figura 45

```
Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
String userEmail = ((Jwt)authentication.getCredentials()).getClaim("email");
Optional<User> optionalUser = userRepository.findByEmail(userEmail);
Integer userId = optionalUser.get().getUser_id();
```

Figura 45. Extragerea ID-ului pentru utilizatorul autentificat

5.3 Căutarea unei infrastructuri/oferte

Pentru căutarea acestora am creat o bara de căutare atât la infrastructura cat si la oferta. A se vedea Figura 46.



Figura 46. Bara de cautare

Cand utilizatorul v-a introduce ceva ce cauta si v-a apasă pe caută sau enter. Se v-a face un api call către endpoint-ul `/api/searchInfrastructures`. Metoda din RestController cu numele `searchInfrastructures` v-a recepționa acest call, si v-a apela o metoda in `infrastructureService` ca parametru aceasta v-a primi string-ul introdus de utilizator. A se vedea Figura 47.


```

@GetMapping("/api/searchInfrastructures")
public List<ViewInfrastructureDto> searchInfrastructures(@RequestParam(value = "name", required = false) String name){
    return infrastructureService.searchInfrastructures(name);
}

```

Figura 47. Metoda SearchInfrastructures

Căutarea unei infrastructuri/oferte poate fi făcută după nume, cuvinte cheie și e-mail al utilizatorului care a adăugat-o.

Metoda de căutare a infrastructurii lucrează astfel:

- Se verifica string-ul care a venit ca parametru, adică ce a introdus utilizatorul, dacă acesta este un sir gol atunci ii voi returna toate infrastructurile.
- Dacă nu este un sir gol atunci fac căutarea și filtrarea infrastructurilor.
 - InfrastructureRepository.findAll(): Obțin toate infrastructurile din repository
 - .stream(): Creez un stream pentru procesarea ulterioară
 - .filter(): Filtrez infrastructura pe baza criteriilor nume, cuvinte cheie, e-mail
 - .filter(o -> o.isPrivate_Status()) : Filtrez infrastructurile care au private_status setat la true
 - .map(): Transform fiecare infrastructura filtrată într-un obiect de tip ViewInfrastructureDto, pentru a extrage ușor datele în JavaScript la afișarea acestora.
 - .collect(): Colectez rezultatele care au îndeplinit toate criteriile.

Pentru filtrarea am adus tot string-ul dat de utilizatorul la litere mici, pentru a ușura căutarea, chiar dacă utilizatorul v-a scrie o literă mare infrastructura să apară la căutare.

Filtrarea se face și după un substring adică dacă utilizatorul nu v-a ști denumirea completă aceasta oricum v-a apărea.

Pentru partea de search al ofertelor este aceeași idee ca și la infrastructuri doar că se apelează alte metode, însă acestea fac același lucru. String-ul însă v-a veni de pe pagina de oferte.

Codul scris pentru metoda de search se poate vedea în Anexa 13 – Căutare Infrastructuri

6 STUDIUL DE CAZ / EVALUAREA REZULTATELOR

Acest capitol v-a răspunde la două întrebări, platforma merge corect și cât de bine merge aceasta. Voi discuta despre cerințele care au fost propuse și dacă au fost implementate și voi compara performanța platformei cu alte soluții existente.

6.1 Corectitudinea soluției

Pentru a analiza ce soluții au fost propuse și dacă au fost implementate, voi realiza un tabel în care voi compara și arată ce cerințe am implementat în platforma și dacă ele sunt și pe alte platforme despre care am vorbit în capitolul Platforme existente pe piață. A se vedea Tabel 1.

Tabel 1. Comparație funcționalități platforme

Cerință \ Platformă	InfraOff	InfraTech	EERTIS
Login/Logout	✓	✓	✓
Register	✓	✓	✓
Adăugare Infrastructura	✓	✓	✓
Adăugare Oferta	✓	✓	✓
Vizualizare Oferta	✓	✓	✓
Vizualizare Infrastructura	✓	✓	✓
Vizualizare Infrastructura pe Harta	✓	X	✓
Modificare Infrastructura/Oferta	✓	✓	✓
Ștergere Infrastructura/Oferta	✓	✓	✓
Vizualizare Cont propriu/cercetător	✓	X	X
Vizualizare Calendar Infrastructura	X	✓	X
Pagina Contact	✓	✓	✓
Mecanism de Căutare	✓	✓	✓
Sortare după Tara	X	X	✓

În Tabel 1, am realizat o comparație între funcționalitățile de pe platforma mea și cele mai importante de pe alte platforme existente ca EERTIS și INFRA TECH, după cum se poate observa funcționalitățile propuse în capitolul Analiza și specificarea cerințelor au fost toate realizate cu succes, însă platformele mai au și alte funcționalități care nu există pe platforma mea. Ca de exemplu platforma EERTIS permite sortarea infrastructurilor și ofertelor după Țara din care provin, pe platforma mea aceasta nu ar avea sens întrucât se prezintă infrastructurile și ofertele de tehnologie din incinta UNSTPB.

6.2 Evaluarea performanțelor

Pentru a evalua și compara performanța platformei mele și a altor platforme am folosit Google LightHouse care este un instrument open-source dezvoltat de Google acesta este utilizat pentru a analiza și optimiza performanța pe fiecare pagină a platformei. Mai jos voi arăta performanța pe paginile principale ale aplicației.

Caracteristicile Principale ale acestuia include:

1. Performanță: Evaluează viteza și eficiența site-ului web, oferind scoruri pentru timpul de încărcare, interactivitate și stabilitatea vizuală.
2. Accesibilitate: Verifică site-ul dacă este ușor de utilizat pentru toate categoriile de utilizatori, evaluând aspect precum contrastul culorilor, structura documentului.

3. Best Practices: Analizează conformitatea cu cele mai bune practici de dezvoltare web, cum ar fi utilizarea de resurse de securitate și evitarea utilizării API-urilor depreciate.
4. SEO (Search Engine Optimization): Evaluează optimizarea pentru motoarele de căutare.

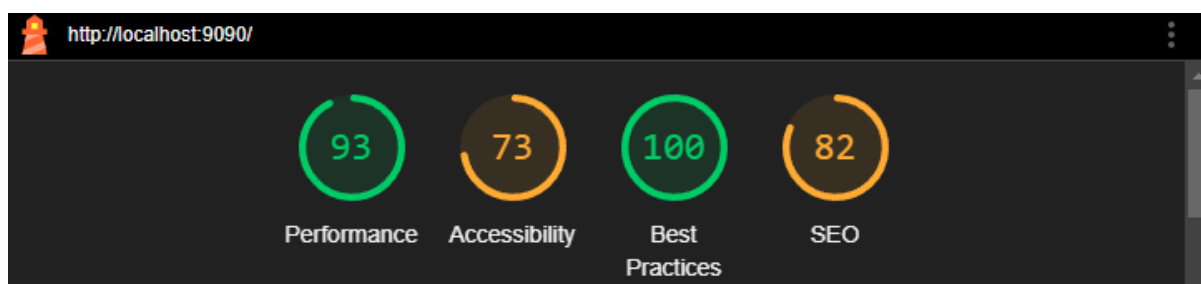


Figura 48. Analiza cu Google LightHouse InfraOff Pagina Principala

În ceea ce privește performanța platformei pe Pagina Principala am obținut un scor de 93 ceea ce este foarte bun.

Metricile pe care le folosește pentru a calcula performanța se pot vedea în Figura 49.

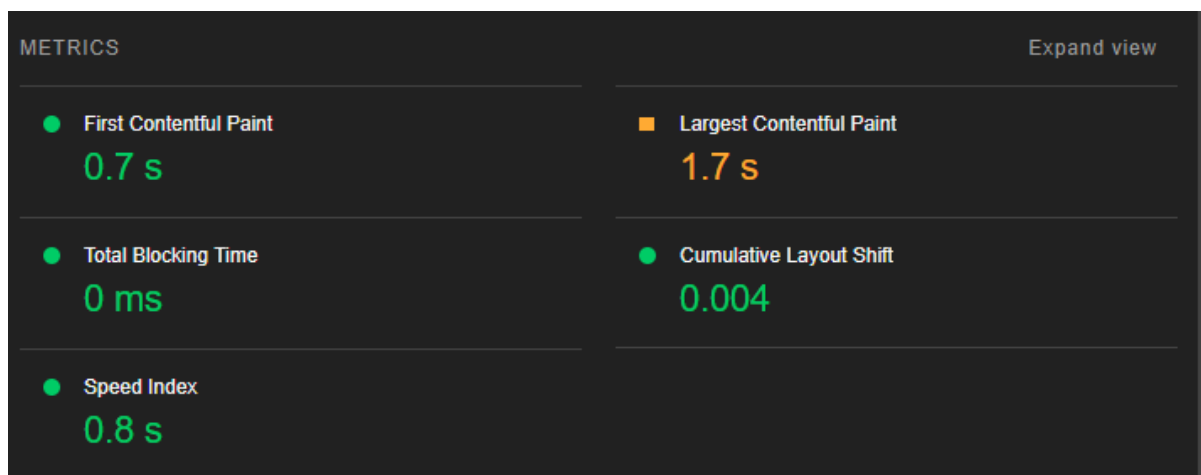


Figura 49. Metrici

Toate aceste metrici indică faptul că pagina mea are o performanță excelentă:

1. First Contentful Paint de 0.7 secunde arată că primul conținut vizibil apare rapid.
2. Largest Contentful Paint de 1.7 secunde indică o încărcare rapidă a celui mai mare element vizual.
3. Total Blocking Time de 0 ms sugerează că pagina este foarte responsivă.
4. Cumulative Layout Shift de 0.004 demonstrează stabilitate vizuală remarcabilă.
5. Speed Index de 0.8 secunde confirmă o experiență de încărcare vizuală rapidă.

În Anexa 15 – Performanța pe alte pagini ale platformei se poate observa că pe paginile principale ale platformei performanța este una foarte bună cu un scor peste 90. Aceasta indică faptul că platforma are un timp de răspuns foarte mic.

6.3 Compararea cu alte platforme

Folosind Google LightHouse am analizat si cele doua platforme EERTIS si InfraTech pentru a putea face o comparație între platforme.

În Tabelul 2 voi face o medie între performanțele pe diferite pagini ale platformelor si voi prezenta rezultatele.

Tabel 2. Mediile metricilor

Platformă \ Metrică	InfraOff	InfraTech	EERTIS
Performanta	91	90	73
Accesibilitate	82	79	78
Best Practices	96	100	75
SEO	82	85	82

- Performanta: InfraOff si InfraTech au scoruri excelente în performanta, ambele fiind peste 90. Acest lucru arata ca platformele se încarcă rapid și oferă o experiența buna utilizatorilor din punct de vedere a performantei. EERTIS are un scor mai scăzut de 73.
- Accesibilitate: Toate platformele au scoruri rezonabile în accesibilitate.
- Best Practices: InfraTech are un scor perfect de 100, ceea ce indica ca respecta toate cele mai bune practici recomandate. InfraOff are un scor excelent de 96, iar EERTIS are un scor mai scăzut, ceea ce arata ce exista loc pentru îmbunătățiri.
- SEO: Toate platformele sunt relativ bine optimizate din punct de vedere SEO, însă exista loc pentru îmbunătățiri.

6.4 Validarea cu clienții

Platforma infraOff a fost validata cu domnul coordonator de la care am primit feedback si idei noi de implementare pe tot parcursul dezvoltării platformei. Însă pentru obținerea si altor feedback-uri legate de platforma, am dat colegiilor, prietenilor să o folosească pentru a obține acest feedback și a testa mai bine platforma. Am oferit niște detalii minime pentru ce face platforma mea și am încercat sa observ daca platforma este intuitiva. Pentru răspunsurile colegilor am utilizat un chestionar în care aceștia au notat de la 1 la 5 câteva întrebări. Se pot vedea răspunsurile în anexele Anexa 16 – Feedback pagina 1 si Anexa 17 – Feedback pagina 2.

În ceea ce privește experiența generala pe platforma am primit un feedback pozitiv cu note de 4 si 5 ceea ce arata ca utilizatorii au avut o experiența plăcută pe platforma.

La capitolul interfața platformei utilizatorii au notato la fel de bine însă am avut niște remarcări în ceea ce privește pagina de descriere a infrastructurii, și unele elemente în prezentarea tuturor infrastructurilor si ofertelor.

Calitatea si funcționalitățile platformei au la fel note destul de mari, însă in timpul folosirii platformei utilizatorii au întâmpinat un bug, care l-am fixat.

Acest formular m-a ajutat mult sa înțeleg calitatea platformei și m-a ajutat sa testez platforma, sa înțeleg cat de intuitiva este si ce schimbări ar trebui făcute.

Însă in urma acestui sondaj se poate concluziona ca platforma corespunde cerințelor utilizatorilor si ca funcționalitățile platformei sunt la un nivel înalt.

7 CONCLUZII

In cadrul platformei InfraOff, am încercat sa adaug funcționalitățile necesare pentru a satisface ideea principala a platformei si sa creez o interfața plăcută si intuitiva pentru utilizatori. Rezultatele pozitive primite in feedback arata ca ceea ce mi-am propus a fost îndeplinit. Anexa 16 – Feedback pagina 1 și Anexa 17 – Feedback pagina 2.

Toate cerințele funcționale si nefuncționale au fost implementate. Platforma are un timp de răspuns foarte mic asta se poate vedea si in feedback-ul primit si in datele primite de către Google Lighthouse Figura 48.

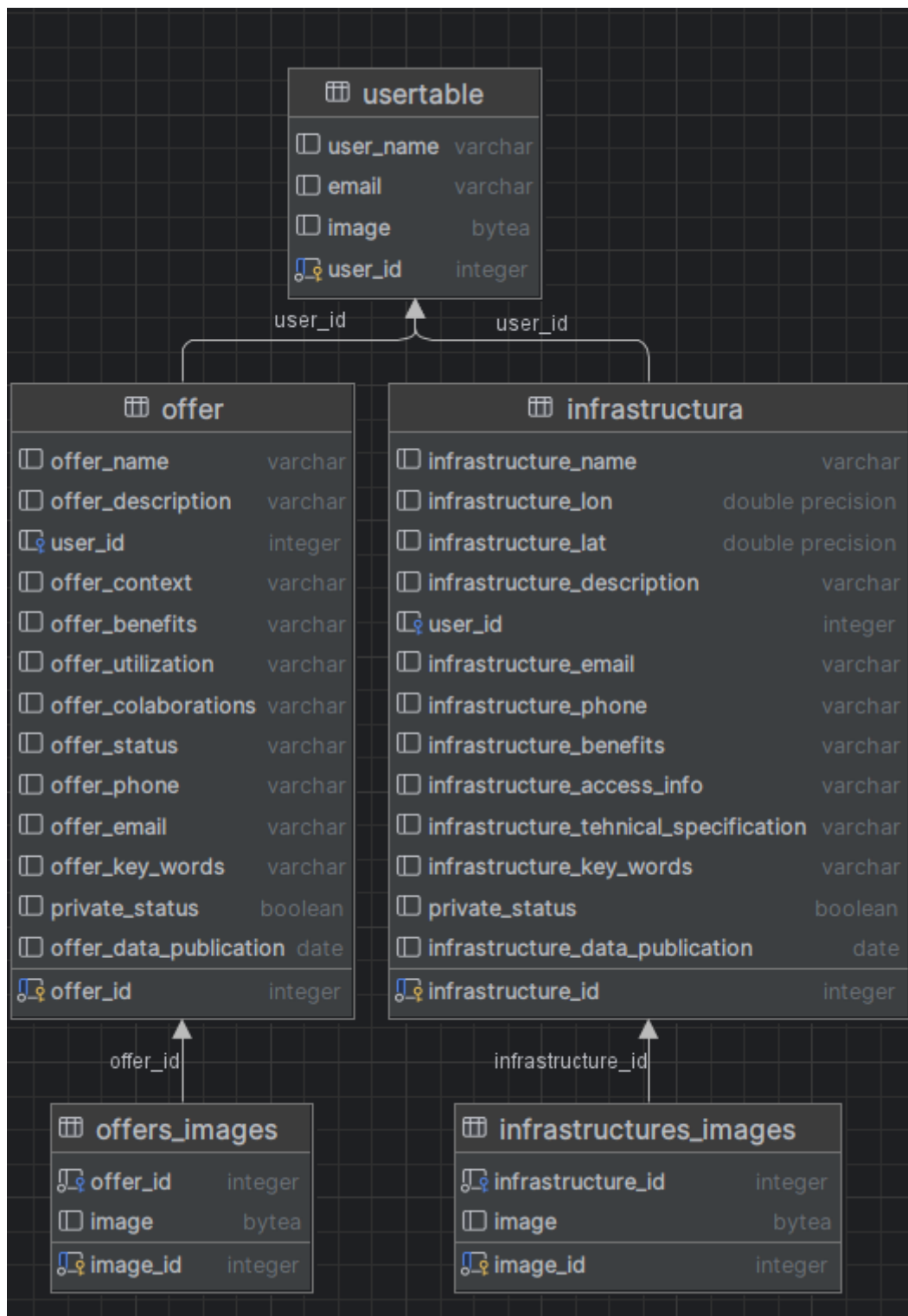
Însă platforma are si unele limitări in ceea ce privește informația despre o infrastructura si oferta, adăugarea de câmpuri noi pentru descrierea acestora v-a fi posibila doar de dezvoltator, însă structura care o oferă Spring face acest proces unul foarte ușor. Pe parcursul dezvoltării baza de date a fost modificate de foarte multe ori după primirea unui feedback din partea coordonatorului acest proces a fost destul de ușor.

8 BIBLIOGRAFIE

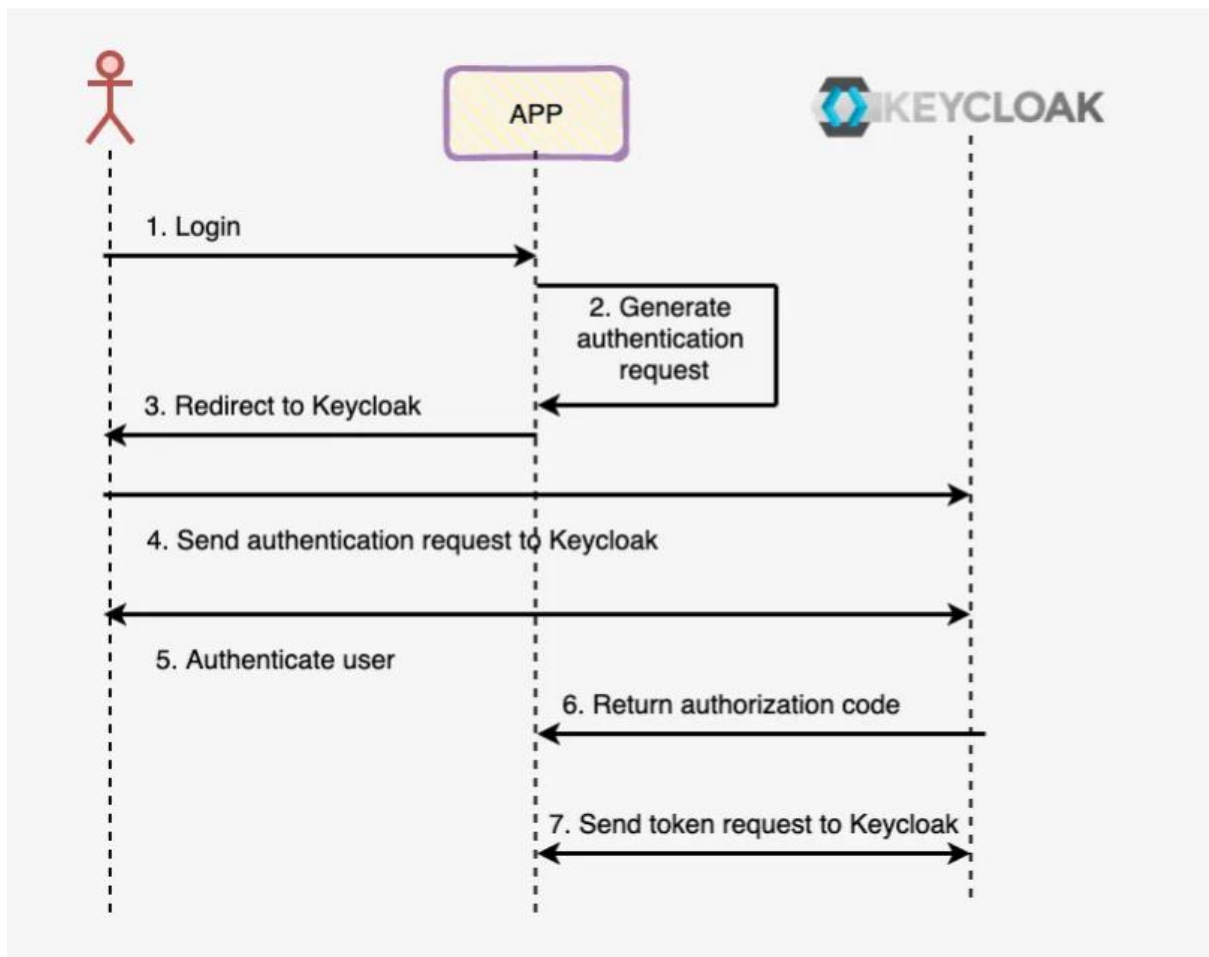
- [1] G. A. M. T. M. Bierman, „Understanding TypeScript,” în *In: Jones, R. (eds) ECOOP 2014 – Object-Oriented Programming. ECOOP 2014. Lecture Notes in Computer Science, vol 8586. Springer, Berlin, Heidelberg., 2014.*
- [2] A. ILERI, „Introduction to Keycloak,” 9 2021. [Interactiv]. Available: <https://abdulsamet-ileri.medium.com/introduction-to-keycloak-227c3902754a> [Accesat 6.10.2024]. [Accesat 10 6 2024].
- [3] P. Sudusinghe, „The arhitecture of the Spring Boot REST applications,” 10 2023. [Interactiv]. Available: <https://medium.com/@piyumisudusinghe/the-architecture-of-the-spring-boot-rest-applications-3643e905c5f8>. [Accesat 10 6 2024].
- [4] M. McGrath, HTML, CSS & JavaScript in easy steps, In Easy Steps Limited, 2020.
- [5] R. H. J. D. K. S. C. H. R. R. T. .. & W. P. Johnson, „The spring framework-reference documentation. interface, 21, 27,” 2004.

9 ANEXE

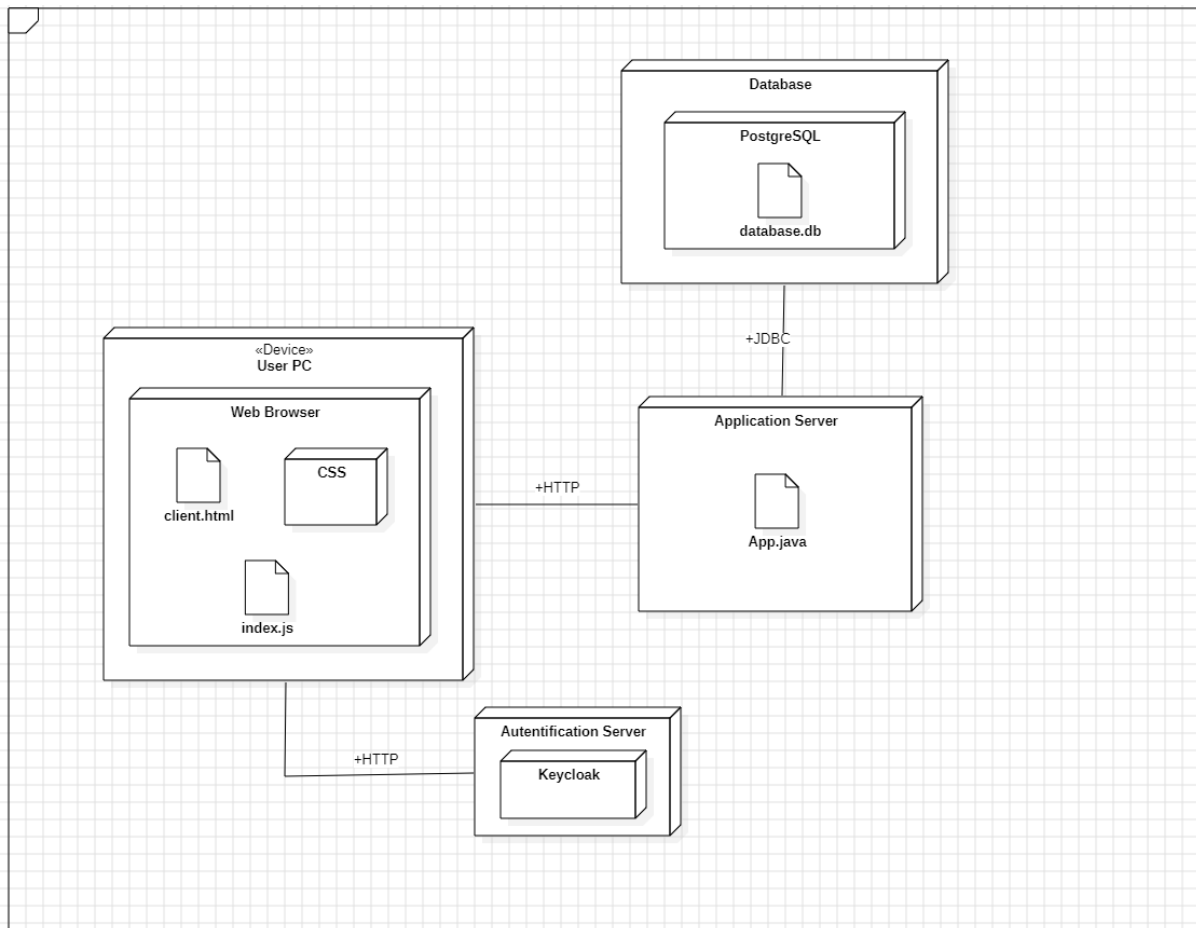
9.1 Anexa 1 – Reprezentarea Bazei de date



9.2 Anexa 2 – Autorizare Flow preluata din [2]



9.3 Anexa 3 – Deployment Diagram



9.4 Anexa 4 – Pagina de contact

Contact

Daca vreti drepturi pentru a adauga pe site-ul nostru oferte sau infrastructuri atunci ne puteti contacta

Suna
0732 576 574

Contact prin email
infraoffunstp@gmail.com

Viziteaza-ne
Splaiul Independenței 313, București 060042

Ghid Pas cu Pas Adaugare Infrastructura sau Oferta

- 1 Autentificare/Inregistrare**
Te autentifici cu contul tau sau iti creezi unul nou, daca ai un cont upb atunci vei putea automat sa adaugi pe site-ul nostru, daca nu atunci ne poti contact pentru a cere permisiuni.
- 2 Navigheaza la Adaugare**
Acceseaza pe site Adauga, si alege optiunea care doresti, infrastructura sau oferta.
- 3 Adaugare**
Dupa ce ai ales ce vrei sa adaugi completeaza formularul (nu uita ca poti sa modifichi oricand oferta sau infrastructura) si apasa butonul Adauga.
- 4 Navigheaza la Infrastructurile mele sau Ofertele mele**
Aici poti vedea ce infrastructura sau oferta adaugata.
- 5 Modificare/Stergere**
Poti sterge sau modifica infrastructura, vei vedea toate datele disponibile care deja le-ai adaugat anterior.

Copyright © Universitatea Nationala de Stiinta si Tehnologie POLITEHNICA Bucuresti

9.5 Anexa 5 – Pagina de adăugare a unei infrastructuri.

Adauga Infrastructura Cu Ajutorul Formularului De Mai Jos

Numele infrastructurii

Lot

Imp

Categorie chela

Descrierea infrastructurii

Beneficiarii care se adauga infrastructurii

Accesul la infrastructura

Specificatiile tehnice ale infrastructurii

Numar de contact

E-mail de contact

Publica infrastructura

Privat

Choose File No file chosen

Adauga infrastructura

9.6 Anexa 6 – Pagina de adăugare a unei oferte

Adauga Oferta Cu Ajutorul Formularului De Mai Jos

Numele Ofertei

Costul chiriei

Descrierea Ofertei

Contextul și conceptul de bază al tehnologiei

Avantaje și Elemente Competitive

Domenii de utilizare și Oportunități de piață

Colaboranți Solicitați

Statutul / stadiul protecției Proprietății Intellectuale

Număr de contact

E-mail de contact

Fotografia Ofertei

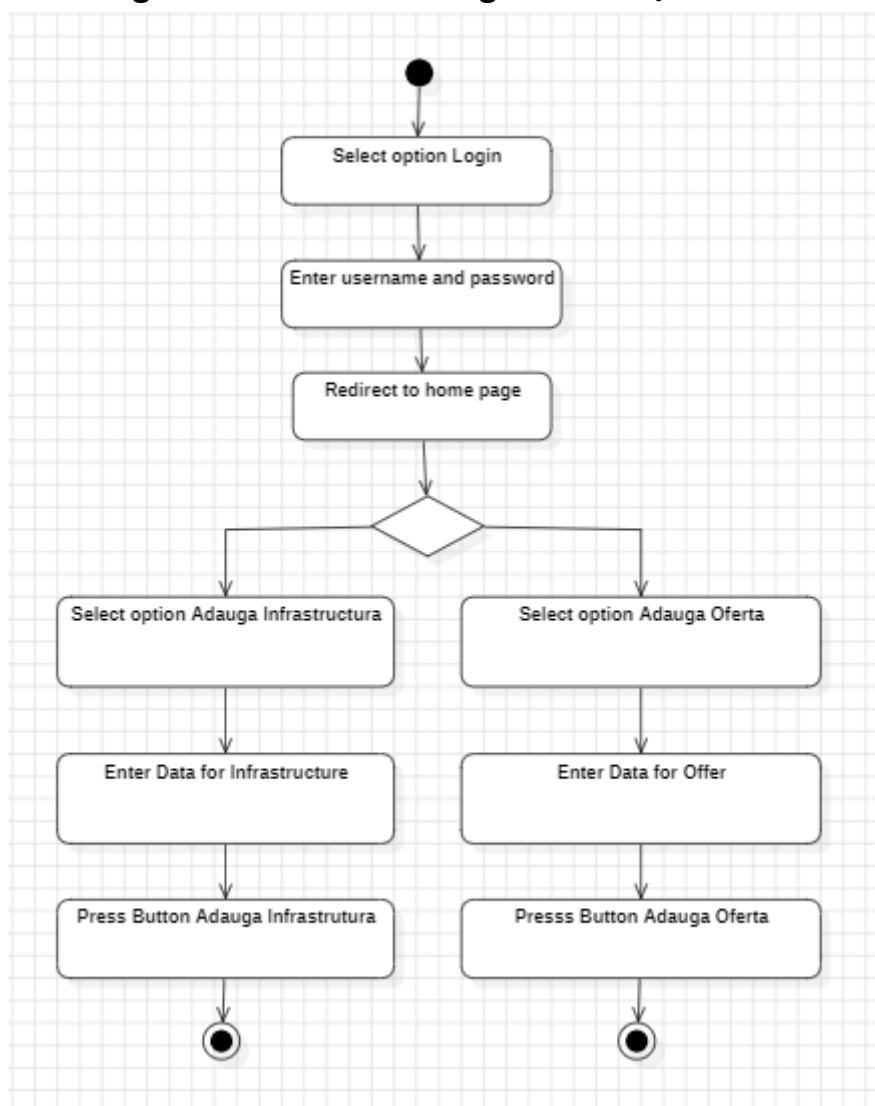
Choose File

No file chosen

Privat

Adauga Oferta


9.7 Anexa 7 – Diagrama Activitate Adăugare Oferta/Infrastructură




9.8 Anexa 8 – Modificare Infrastructura

[illegible]

9.9 Anexa 9 - Profil Utilizator



AcasaOferteContactInfrastructuriAdauga

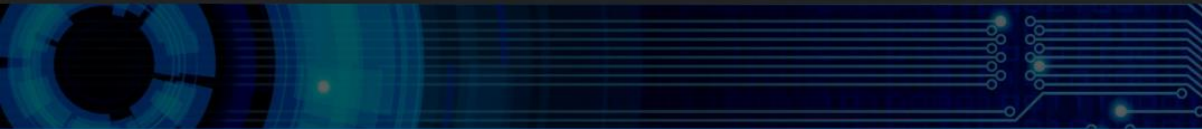


david.mihalcenco@gmail.com
Infrastructuri pe Site : 6
Oferte pe Site : 1
[About](#) [Infrastructuri](#) [Oferte](#)

Alege Fotografia

Modifica Fotografia

Email: david.mihalcenco@gmail.com



Copyright © Universitatea Nationala de Stiinta si Tehnologie POLITEHNICA Bucuresti

9.10 Anexa 10 – Câmpurile definite pentru infrastructura HTML

```
</fieldset>
<label for="Name" class="form-label">Numele Infrastructurii</label>
<input type="text" name="Name" class="Name" autocomplete="on" required>
</fieldset>
</div>
<div class="col-lg-12">
  <fieldset>
    <label for="Lat" class="form-label">Lat</label>
    <input type="text" name="Lat" class="Lat" autocomplete="on" required>
  </fieldset>
</div>
<div class="col-lg-12">
  <fieldset>
    <label for="Lng" class="form-label">Lng</label>
    <input type="text" name="Lng" class="Lng" autocomplete="on" required>
  </fieldset>
</div>
<div class="col-lg-12">
  <fieldset>
    <label for="Key_words" class="form-label">Cuvinte cheie</label>
    <input type="text" name="Key_words" class="Key_words" autocomplete="on" required>
  </fieldset>
</div>
<div class="col-lg-24">
  <fieldset>
    <label for="Description" class="form-label">Descrierea Infrastructurii</label>
    <textarea type="text" name="Description" class="Description" autocomplete="on" required rows="5"></textarea>
  </fieldset>
</div>
<div class="col-lg-24">
  <fieldset>
    <label for="Benefits" class="form-label">Beneficiile care le aduce Infrastructura</label>
    <textarea type="text" name="Benefits" class="Benefits" autocomplete="on" required rows="5"></textarea>
  </fieldset>
</div>
```


9.11 Anexa 11 – Cererea POST pentru adăugare infrastructura

```
const formData : FormData = new FormData();
formData.append( name: 'image', image);
formData.append( name: 'InfrastructureDto', new Blob( blobParts: [JSON.stringify( value: {
    'infrastructure_name': name,
    'infrastructure_key_words': key_words,
    'infrastructure_description': description,
    'user_id' : 1,
    'infrastructure_benefits' : benefits,
    'infrastructure_email' : email,
    'infrastructure_phone' : number,
    'infrastructure_tehcnical_specification' : specification,
    'infrastructure_access_info' : access,
    'private_status' : option,
    'infrastructure_data_publication': 1,
    'infrastructure_lon': lng,
    'infrastructure_lat': lat,
  } ]), options: {type: 'application/json'}}));

$.ajax({
  url: 'http://localhost:9090/addInfra',
  type: 'POST',
  data: formData,
  processData: false, // Important
  contentType: false, // Important
  success: function(response) : void {
    window.location.href = "http://localhost:9090/infrastructures";
  }
});
```

9.12 Anexa 12 – Metoda save din infrastructureService

```
public void save(InfrastructureDto infrastructureDto, MultipartFile image){

    LocalDate currentDate = LocalDate.now();

    Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
    String userEmail = ((Jwt)authentication.getCredentials()).getClaim("email");
    Optional<User> optionalUser = userRepository.findByEmail(userEmail);
    Integer userId = optionalUser.get().getUser_id();

    Infrastructure infrastructure = new Infrastructure();
    InfrastructuresImages infrastructuresImages = new InfrastructuresImages();

    infrastructure.setInfrastructure_name(infrastructureDto.infrastructure_name);
    infrastructure.setInfrastructure_key_words(infrastructureDto.infrastructure_key_words);
    infrastructure.setInfrastructure_description(infrastructureDto.infrastructure_description);
    infrastructure.setUser_id(userId);
    infrastructure.setInfrastructure_email(infrastructureDto.infrastructure_email);
    infrastructure.setInfrastructure_phone(infrastructureDto.infrastructure_phone);
    infrastructure.setInfrastructure_benefits(infrastructureDto.infrastructure_benefits);
    infrastructure.setInfrastructure_access_info(infrastructureDto.infrastructure_access_info);
    infrastructure.setInfrastructure_tehcnical_specification(infrastructureDto.infrastructure_tehcnical_specification);
    infrastructure.setPrivate_status(infrastructureDto.private_status);
    infrastructure.setInfrastructure_data_publication(currentDate);
    infrastructure.setInfrastructure_lat(infrastructureDto.infrastructure_lat);
    infrastructure.setInfrastructure_lon(infrastructureDto.infrastructure_lon);

    Infrastructure savedInfrastructure = infrastructureRepository.save(infrastructure);

    Integer infrastructureID = savedInfrastructure.getInfrastructure_id();
    infrastructuresImages.setInfrastructure_id(infrastructureID);
    infrastructuresImages.setData(image.getBytes());

    infrastructuresImagesRepository.save(infrastructuresImages);
}
```

9.13 Anexa 13 – Căutare Infrastructuri

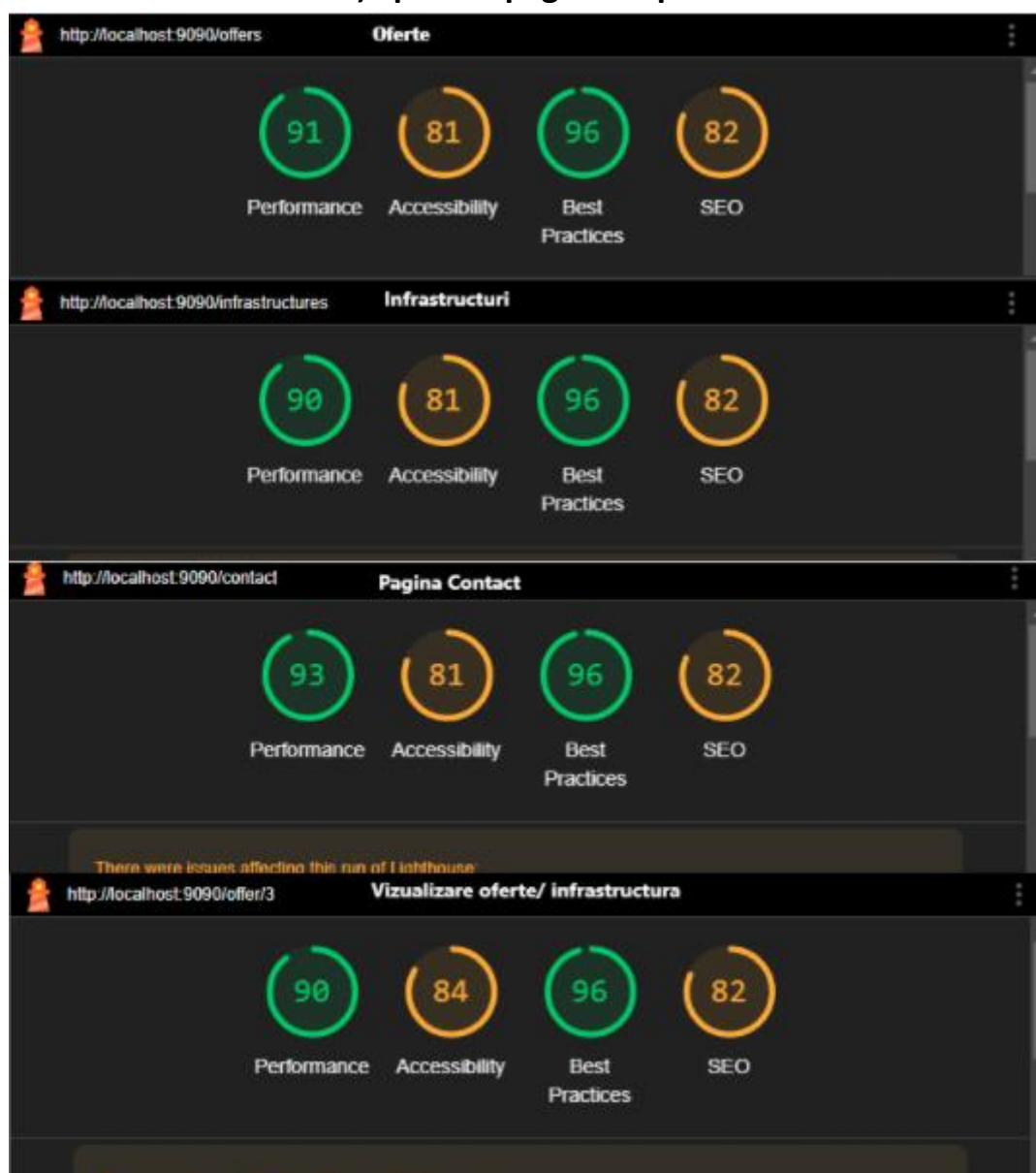
```
public List<ViewInfrastructureDto> searchInfrastructures(String name){
    if(!Objects.equals(name, b: "")) {
        return infrastructureRepository.findAll() List<Infrastructure>
            .stream() Stream<Infrastructure>
            .filter(o -> o.getInfrastructure_name().toLowerCase().contains(name.toLowerCase()) ||
                o.getInfrastructure_key_words().toLowerCase().contains(name.toLowerCase()) ||
                userRepository.findById(o.getUser_id()).get().getEmail().contains(name.toLowerCase()))
            .filter(o -> o.isPrivate_status())
            .map(o -> {
                byte[] imageData = new byte[0];
                try {
                    imageData = infrastructuresImagesRepository.findById(o.getInfrastructure_id()) Optional<In
                        .map(InfrastructuresImages::getData) Optional<byte[]>
                        .orElse(new byte[0]);
                } catch (Exception e) {
                    e.printStackTrace();
                }
                int maxLength = 100;
                String description = o.getInfrastructure_description();
                if (description.length() > maxLength) {
                    description = description.substring(0, maxLength) + "...";
                }
                return ViewInfrastructureDto.builder()
                    .infrastructure_name(o.getInfrastructure_name())
                    .infrastructure_description(description)
                    .infrastructure_phone(o.getInfrastructure_phone())
                    .infrastructure_image(imageData)
                    .infrastructure_id(o.getInfrastructure_id())
                    .infrastructure_data_publication(o.getInfrastructure_data_publication())
                    .build();
            }) Stream<ViewInfrastructureDto>
            .collect(Collectors.toList());
    } else {
        return getAllInfrastructures();
    }
}
```

9.14 Anexa 14 – Câmpuri infrastructura în baza de date

```
create table infrastructura
(
    infrastructure_name          varchar,
    infrastructure_id            integer generated always as identity
        constraint infrastructura_pk
            primary key,
    infrastructure_lon           double precision,
    infrastructure_lat           double precision,
    infrastructure_description   varchar,
    user_id                     integer
        constraint infrastructura_user_user_id_fk
            references usertable,
    infrastructure_email         varchar,
    infrastructure_phone         varchar,
    infrastructure_benefits      varchar,
    infrastructure_access_info   varchar,
    infrastructure_tehcnical_specification varchar,
    infrastructure_key_words     varchar,
    private_status               boolean,
    infrastructure_data_publication date default CURRENT_DATE
);

alter table infrastructure
    owner to "user";
```

9.15 Anexa 15 – Performanța pe alte pagini ale platformei

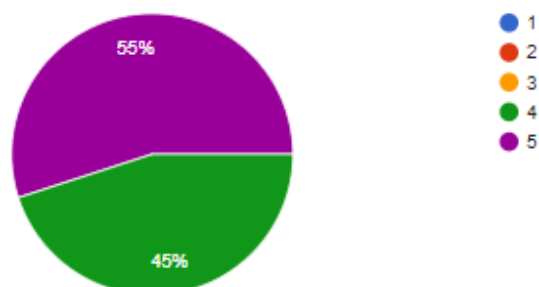


9.16 Anexa 16 – Feedback pagina 1

Cum ați evalua experiența generală cu platforma de la 1 la 5?

 Copy

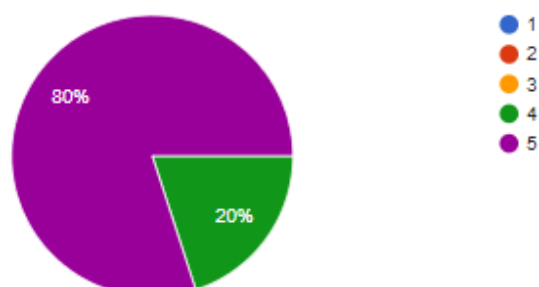
20 responses




Cât de rapidă a fost platforma în utilizare? (1 fiind foarte lentă, 5 foarte rapidă)

 Copy

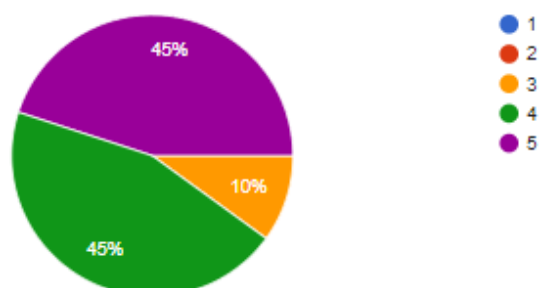
20 responses



Evaluati de la 1 la 5 interfața platformei

 Copy

20 responses

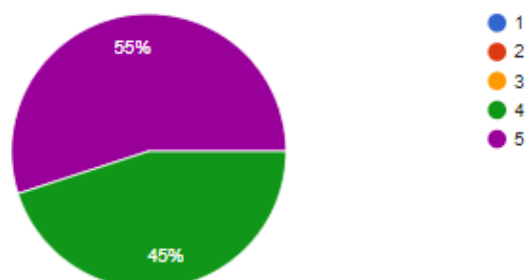


9.17 Anexa 17 – Feedback pagina 2

Cât de ușor a fost să navigați și să utilizați funcționalitățile platformei? (1 fiind foarte dificil, 5 foarte ușor)

 Copy

20 responses



Cum ați evalua calitatea și funcționalitățile oferite de platforma? (1 fiind slabă, 5 foarte bună)

 Copy

20 responses

