



Universitatea Tehnică „Gheorghe Asachi” din IAȘI
Facultatea de Automatică și Calculatoare
Domeniul: *Calculatoare și Tehnologia Informației*
Specializarea: *Tehnologia Informației*



LUCRARE DE DIPLOMĂ

Coordonator științific:
Ș.I. dr. Ing. Cristian Aflori

Absolvent:
Dumea Cristian

Iași, 2024



Universitatea Tehnică „Gheorghe Asachi” din IAȘI
Facultatea de Automatică și Calculatoare
Domeniul: *Calculatoare și Tehnologia Informației*
Specializarea: *Tehnologia Informației*



PLATFORMĂ WEB PENTRU SUPORTUL TEHNIC AL CALCULATOARELOR

LUCRARE DE DIPLOMĂ

Coordonator științific:
Ș.I. dr. Ing. Cristian Aflori

Absolvent:
Dumea Cristian

Iași, 2024

**DECLARAȚIE DE ASUMARE A AUTENTICITĂȚII
PROIECTULUI DE DIPLOMĂ**

Subsemnatul _____,

legitimă cu _____ seria _____ nr. _____, CNP _____

autorul lucrării _____

_____ elaborată în vederea susținerii examenului de finalizare a studiilor de licență, programul de studii _____ organizat de către Facultatea de Automatică și Calculatoare din cadrul Universității Tehnice „Gheorghe Asachi” din Iași, sesiunea _____ a anului universitar _____, luând în considerare conținutul Art. 34 din Codul de etică universitară al Universității Tehnice „Gheorghe Asachi” din Iași (Manualul Procedurilor, UTI.POM.02 - Funcționarea Comisiei de etică universitară), declar pe proprie răspundere, că această lucrare este rezultatul propriei activități intelectuale, nu conține porțiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislației române (legea 8/1996) și a convențiilor internaționale privind drepturile de autor.

Data

Semnătura

Curpins

1. Introducere.....	1
1.1 Context.....	1
1.2 Motivație.....	2
1.3. Aplicații similare.....	3
1.3.1 Compari.ro.....	3
1.3.2 Stack Overflow.....	4
2. Tehnologii utilizate în dezvoltarea aplicației.....	5
2.1 Mediu de dezvoltare.....	5
2.2 Backend.....	7
2.2.1 Spring Boot.....	7
2.2.2 Stocarea datelor.....	8
2.3 Frontend.....	9
2.3.1 HTML.....	9
2.3.2 CSS.....	10
2.3.3 Angular Material.....	10
2.3.4 Angular.....	11
3. Proiectarea Aplicației.....	12
3.1 Funcțiile aplicației.....	12
3.2 Backend.....	14
3.3 Frontend.....	16
4. Implementarea aplicației.....	18
4.1 Structura bazei de date.....	18
4.2 Securizarea aplicației.....	20
4.3 Rutarea.....	21
4.4 Accesarea datelor.....	22
4.4.1 Legătura dintre API și baza de date.....	22
4.4.2 Legătura dintre API și UI.....	22
4.4.3 Partajarea datelor între componentele.....	23
4.5 Serviciul de mail.....	24
4.6 Web Scraper.....	24
5. Testarea sistemului și rezultate experimentale.....	26
5.1 Componentele interfeței.....	26
5.1.1 Înregistrare și autentificare.....	26
5.1.2 Perspectiva utilizatorului.....	27
5.2.3 Perspectiva Tehnician.....	29
6. Direcții viitoare.....	32
7. Concluzie.....	33
Bibliografie.....	34

Rezumat

În concordanță cu nivelul ridicat al folosirii tehnologiei, am ales să dezvolt o aplicație care are ca scop eficientizarea procesului de reparație al calculatoarelor. Această aplicație este adresată atât persoanelor care este la început de drum cu tehnologia, cât și persoanelor care doresc să afle mai mult despre tehnologie. Utilizatorii au posibilitatea de a vizualiza o listă cu tehnicieni specializați pe anumite servicii din care pot alege unul pentru a face o programare în vederea problemelor pe care le-a întâmpinat la calculatorul personal. Aceștia au la dispoziție un chat și un forum, unde își pot rezolva problemelor fără a mai fi nevoiți să facă o programare la unul din tehnicieni.

În acest context, am ales să dezvolt această aplicație în unul din cele mai populare limbaje de programare, și anume limbajul Java cu framework-ul Spring Boot, iar pentru partea de frontend am ales să folosesc framework-ul Angular. Aceste două framework-uri formează un bun context în crearea unei aplicații web ușor de dezvoltat, bazat pe o arhitectură destul de întâlnită numită MVC.

Lucrarea este împărțită pe 7 capitole, primele 3 capitole prezintă aspecte tehnice pe care le-am folosit în vederea dezvoltării aplicației, cât și exemple de aplicații similare ce au ca mediu de lucru alte domenii. Următoarele 4 capitole se bazează pe descrierea structurii aplicației, cât și pe modul de implementare abordat.

1. Introducere

1.1 Context

Privind contextul actual în care tehnologia face parte din viața de zi cu zi al fiecărui om, aceasta este într-o continuă dezvoltare încercând să satisfacă cât mai mult nevoile omenești. Cu câțiva zeci de ani în urmă, pe când tehnologia abia își făcea apariția, doar producători aveau acces la utilizarea unui calculator, deoarece era un context nou pentru toată lumea, iar costurile de producție erau foarte mari, însă, cu trecerea timpului, datorită dorințelor companiilor de a face cât mai mult profit și a satisface cât mai mult nevoile personale, aceștia lansează cât mai des versiuni de componente cât mai sofisticate, care au nevoie de mai multă întreținere, la intervale cât mai scurte de timp.

Pe lângă aceste aspecte, în România, în anul 2017, s-a înregistrat că în procent de 65.6% fiecare gospodărie dispune de cel puțin un calculator personal. Putem presupune că în zilele noastre acest procent poate ajunge și la 85%-90%, iar gama de componente este foarte diversificată impunându-l pe utilizator la probleme atunci când apare o defecțiune. Soluția pentru această problemă nu este de a înlocui componenta, deoarece componentele fiind atât de evaluate dispun și de niște costuri considerabile pe care nu toată lumea și le poate permite. O soluție fiabilă ar fi găsirea problemei pe internet și încercarea individuală de rezolvare, aceasta poate fi destul de grea, deoarece pentru o componentă sunt diferite versiuni, iar acestea se rezolvă într-un mod diferit. Cea mai bună soluție ar fi prezentarea problemei unui cadru specializat pentru înlăturarea sau fixarea problemei.

1.2 Motivație

Pentru a eficientiza timpul persoanelor care caută să-și rezolve problemele de calculator, am decis să creez o aplicație web ce dispune de mai multe funcționalități ce îi sar în ajutorul utilizatorului. Această aplicație reduce timpul atribuit cât și energia consumată de persoanele care caută un suport tehnic, deoarece totul se face programat și sunt evitate cozile, pe care majoritatea nu le suportă.

Prin intermediul acestei platforme, utilizatorii nu mai sunt nevoiți să sune pentru o programare sau să caute fizic un loc disponibil, ci au la dispoziție o listă cu disponibilitatea fiecărui tehnician și serviciile pe care acesta le prestează. Aplicația este dedicată tuturor persoanelor indiferent de nivelul de cunoștințe, întrucât este disponibilă și o pagină de Forum unde utilizatorii se pot ajuta reciproc, nefiind nevoie să mai piardă timpul făcând rezervare pentru chestii banale.

Totodată, această aplicație aduce un beneficiu semnificativ timpului de căutare pentru piesele care trebuie înlocuite, deoarece aceasta dispune de o caracteristică care caută produsele de pe mai multe site-uri, ajutându-l pe utilizator să își facă o privire de ansamblu asupra prețurilor.

Aplicația sare și în ajutorul persoanelor specializate care sunt la început de drum, deoarece tendința persoanelor este de a își găsi un tehnician la care va apela pe viitor chiar dacă alegerea pe care o fac nu este una din cele mai bune, deoarece există posibilitatea ca distanța de lucru să fie foarte mare, iar prețurile să fie mai mari față de media pieței.

Așadar, printr-o gestiune eficientă a datelor, utilizatorii dispun de o gamă largă de servicii care îi vin în ajutor, iar tehnicienii dispun de o organizare cât mai centralizată.

1.3. Aplicații similare

1.3.1 Compari.ro

O aplicație similară cu platforma pe care am dezvoltat-o este platforma Compari, aceasta fiind accesată prin linkul următor: <https://www.compari.ro/>. Aria de lucru a platformei Compari este foarte largă de la domeniul electrocasnice la îmbrăcăminte, mobilier, construcție. Funcționalitatea ei principală este de a căuta produse pe diverse site-uri și compararea prețurilor acestora. Aplicația este disponibilă în 9 țări europene fiind înființată în anul 2004.

Asemănările semnificative dintre cele două aplicații este că ambele lucrează doar cu site-uri de încredere, prezentând cea mai bună opțiune ca preț și disponibilitate.

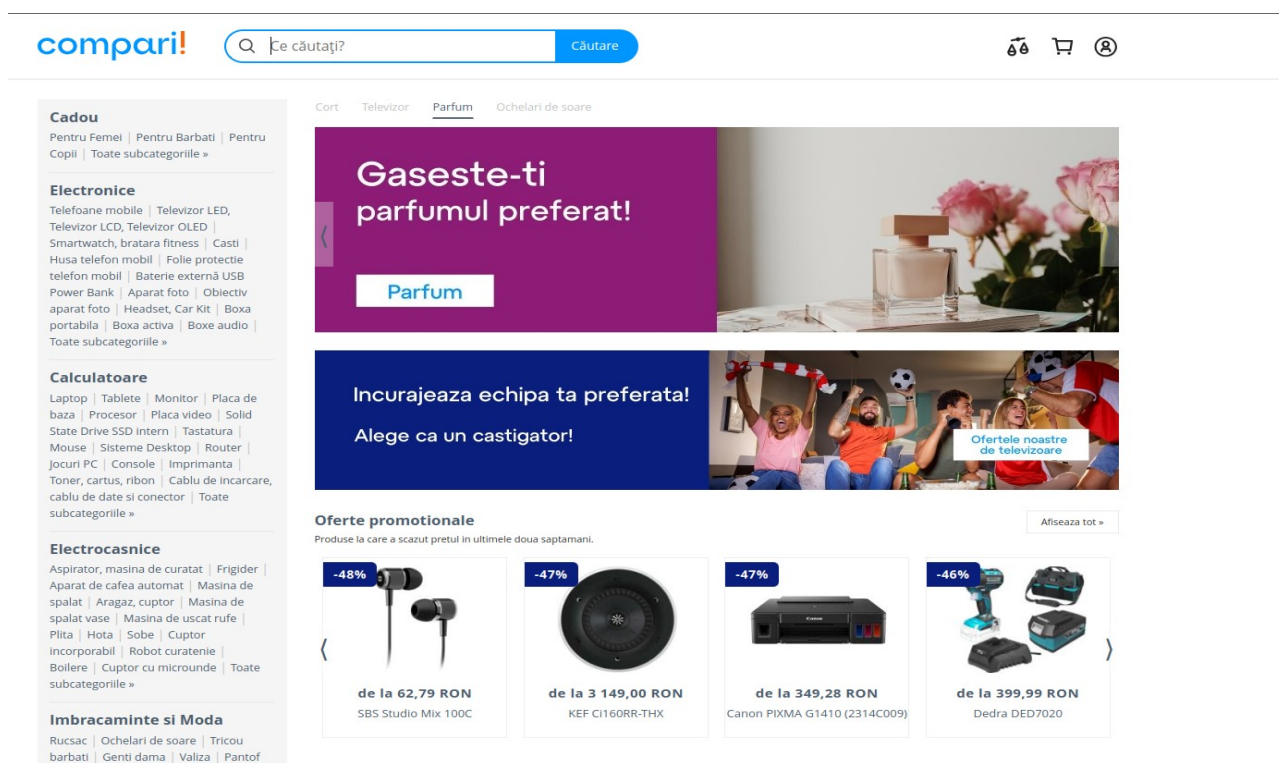


Fig 1.3.1 – Pagină principală Compari

1.3.2 Stack Overflow

O aplicație menită să se ajute utilizatorii reciproc este Stack Overflow, aceasta se poate accesa prin linkul următor: <https://stackoverflow.com/>. Aceasta este o aplicație internațională utilizată adesea de programatori pentru a rezolva problemele de cod și configurație ale proiectelor proprii. Platforma aceasta a fost înființată de către Jeff Atwood și Joel Spolsky în anul 2008, având o medie de 5.6k întrebări pe zi și 22 milioane de întrebări de când a fost înființată.

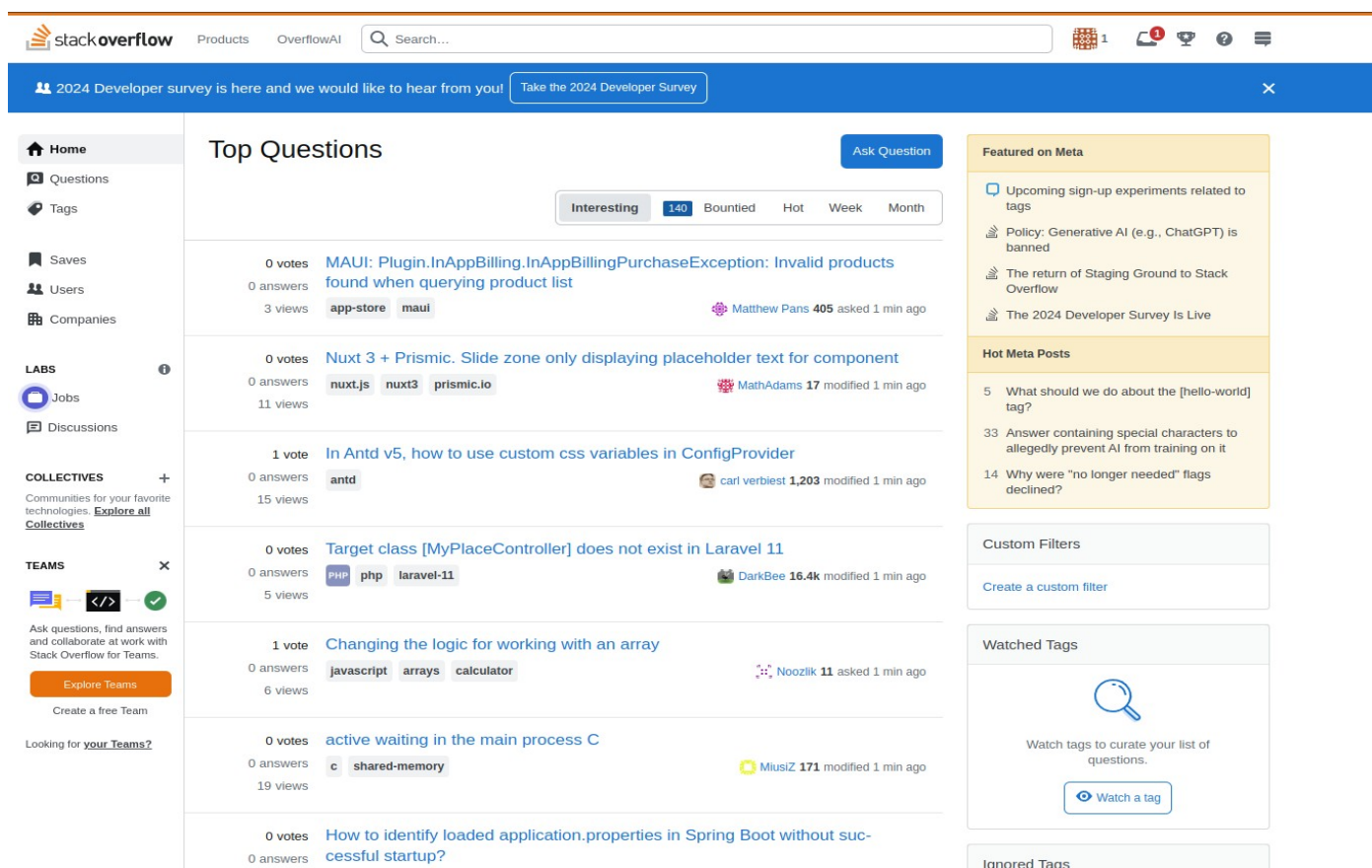


Fig 1.3.2 – Pagină principală Stack Overflow

2. Tehnologii utilizate în dezvoltarea aplicației

Pentru dezvoltarea acestei platforme am folosit unele din cele mai populare limbaje de programare și framework-uri, și anume, Java și Spring Boot pentru partea de Backend și Angular, Angular Material cu limbajul TypeScript pentru partea de Frontend. Pentru gestionarea datelor am folosit ca și bază de date MariaDb.

2.1 Mediu de dezvoltare



Fig 2.1.1 – Logo IntelliJ

IntelliJ IDEA este un IDE (Integrated Development Environment) oferit de compania JetBrains ce poate rula pe Windows, Mac și Linux. Acesta este conceput în principal pentru dezvoltarea de aplicații în limbajul Java și Kotlin însă dispune și de o gamă foarte largă de plugin-uri ce pot fi instalate pentru suportul altor limbaje de programare sau framework-uri. Acest IDE dispune de 3 versiuni: Ultimate, Community și Edu. Versiunea Community este cea gratuită, disponibilă oricărui utilizator însă conține mai puține funcționalități față de cea Ultimate.

Caracteristicile pe care le-am folosit pentru realizarea aplicației cu ajutorul acestui mediu de dezvoltare sunt:

- completarea inteligentă a codului, IDE-ul sugerând anumite cuvinte sau expresii doar prin tastarea primelor caractere.
- extensia Git-ului care face sistemul de versionare cât mai eficient, astfel când aplicația era într-un stadiu stabil putem urca foarte simplu proiectul pe git ca măsură de backup.

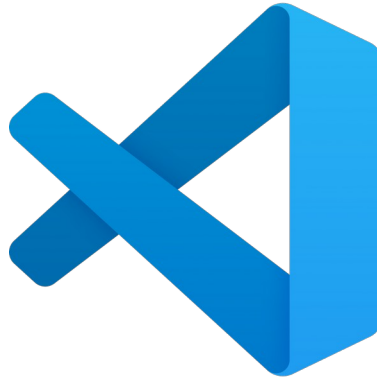


Fig 2.1.2 – Logo Visual Studio Code

Visual Studio Code este un editor text folosit de majoritatea programatorilor dezvoltat de Microsoft ce poate rula pe Windows, Mac și Linux. Acesta dispune de o paletă largă de plugin-uri suport pentru majoritatea limbajelor de programare. Acest program este gratuit și dispune de o singură versiune care se actualizează constant.

Principalele caracteristici ale acestui program le constituie:

- rularea oricărui limbaj de programare cu ajutorul extensiei respective
- aranjarea automată a codului pentru un aspect cât mai plăcut
- sistemul de versionare Git care este prestabilit

2.2 Backend

Partea de Backend este creată folosind cel mai întâlnit mod de dezvoltare, și anume arhitectura client-server ce conține 2 părți: partea clientului și partea server-ului. Conexiunea dintre cele 2 părți este realizată prin intermediul internetului însă, dacă ambele părți se află pe aceeași mașină de lucru, conexiunea la internet devine opțională.

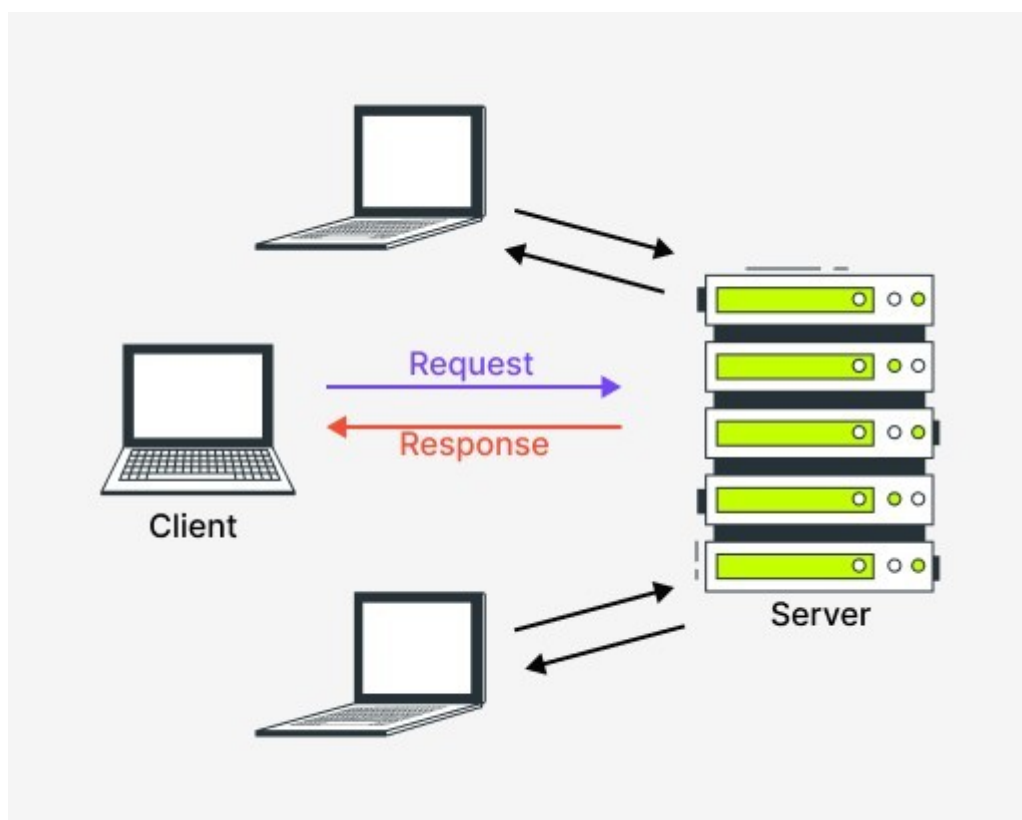


Fig 2.2 – Arhitectura client-server

2.2.1 Spring Boot



Fig 2.2.1 - Logo Spring Boot

Spring boot este un framework gratuit care are la bază limbajul de programare Java. Acesta este dezvoltat de către echipa Pivotal și este folosit pentru accelerarea creării unei aplicații web. Acesta este auto-configurată, dar permite și dezvoltatorilor să adauge sau să schimbe ulterior dependențele.

Avantaje:

- Permite conectarea cu ușurință la o varietate de baze de date.
- Simplitatea integrării în ecosistemul spring precum Spring Data, Spring Security.
- Reduce timpul de dezvoltare și îmbunătățește eficiența generală a echipei de dezvoltare.

Dezavantaje:

- Mărimea mare a fișierelor.
- Nu este potrivit pentru proiecte la scala largă.

2.2.2 Stocarea datelor



Fig 2.2.2 – Logo MariaDB

MariaDB este o baza de date relațională gratuită dezvoltată de Michael ”Monty” Widenius, destul de utilizată la nivel global. Aceasta dispune de o varietate de versiuni, ultima fiind 11.4 LTS disponibilă din data de 24.12.2023.

Avantaje:

- Oferă posibilitatea de a cripta datele la nivel de stocare
- Procesare paralelă a datelor

Dezavantaje:

- Datorită diversității versiunilor, pot apărea probleme de compatibilitate
- Documentație limitată

2.3 Frontend

Această parte a aplicației reprezintă interacțiunea directă cu utilizatorul, astfel încât se dorește a fi cât mai bine aranjată, să fie ușor de utilizat și nu în ultimul rând să fie ușor de înțeles pentru persoanele care accesează pentru prima dată aplicația. Pentru dezvoltarea acestei platforme am folosit framework-ul Angular cu ajutorul elementelor HTML și a limbajului TypeScript, iar pentru stilizare am folosit CSS și Angular Material.

2.3.1 HTML



Fig 2.3.1 – Logo HTML 5

HTML (Hypertext Markup Language) este un limbaj ce definește o structură a conținutului unei pagini web. Această structură este reprezentată de tag-uri pentru diferite elemente precum: text, imagini, paragraf, link-uri, butoane. Cea mai recentă versiune a limbajului este HTML 5, care aduce cu sine elemente organizatorii precum `<header>` partea de sus a paginii, `<section>` conținutul principal al paginii, `<footer>` partea de jos a paginii.

2.3.2 CSS



Fig 2.3.2 – Logo CSS

CSS (Cascading Style Sheets) este folosit în stilizarea și aranjarea unei pagini web. Acest limbaj este dependent de limbajul HTML întrucât fără elemente HTML, codul CSS nu are nici un efect. Acest cod CSS poate fi scris în 3 locuri diferite: în fișier separat cu extensia .css, în interiorul fișierului HTML cu extensia `<style>` sau în interiorul tagului unde se dorește stilizarea elementului.

2.3.3 Angular Material



Fig 2.3.3 – Logo Angular Material

Angular Material este o librărie pe care dezvoltatorii o folosesc în aplicațiile lor, dezvoltate în Angular, pentru a crește rapiditatea de dezvoltare oferind o interfață elegantă și ușor de folosit. Această librărie oferă elemente precum: Cards, Inputs, Tables, List care sunt disponibile pe site-ul oficial al documentației Angular Material ce conține exemple conținând și codul TS, CSS și HTML.

2.3.4 Angular



Fig 2.3.4 – Logo Angular

Angular este un framework gratuit al limbajului de programare JavaScript ce este scris în limbajul de programare TypeScript. Acesta este dezvoltat de compania Google și are ca scop principal dezvoltarea de SPA(Single Page Application).

Avantaje:

- Arhitectura MVC permite diferențierea logicii aplicației de partea de interfață
- Simplitatea integrării dezvoltatorilor noi în proiecte deja existente
- Legare bidirecțională a datelor.

Dezavantaje:

- Curbă de învățare destul de abruptă.
- Dimensiunea considerabilă care duce la încetinirea aplicației.

3. Proiectarea Aplicației

Structura aplicației este una tip RESTful în care interfața este afișată prin intermediul unui browser, cererile din interfață sunt trimise către serverul de Backend, iar acesta prin operațiile CRUD gestionează datele pe care le solicită utilizatorul. Cererile dintre partea de backend și frontend sunt de tipul HTTP: POST, GET, PUT, GET, DELETE. Fluxul de activități este cel din imaginea de mai jos:

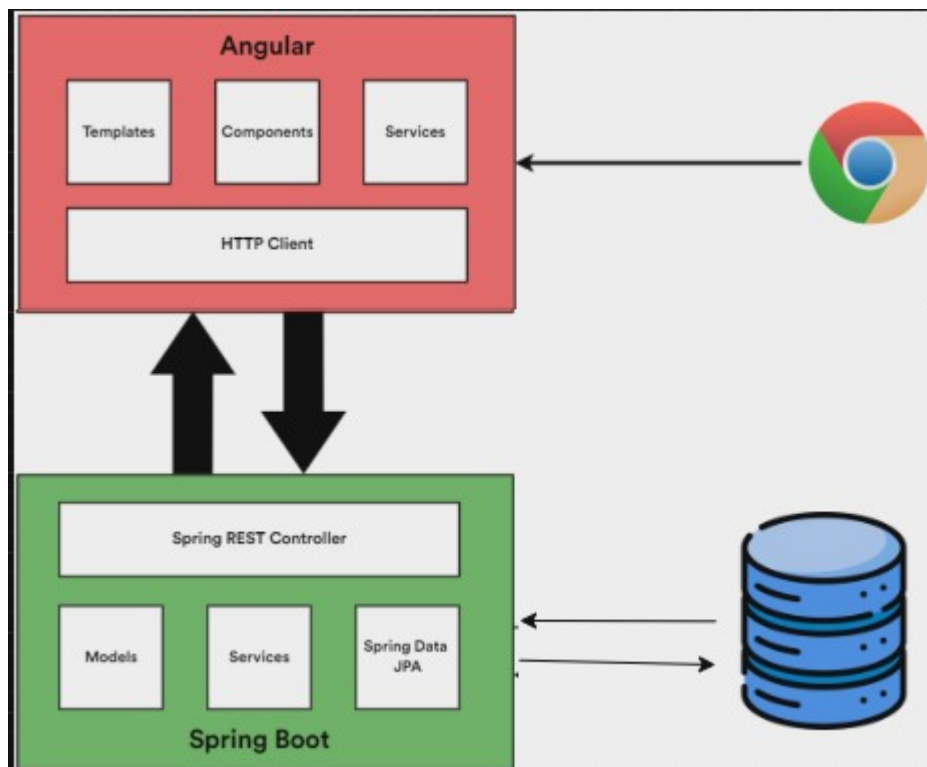


Fig 3 – Fluxul de activități

3.1 Funcțiile aplicației

Aplicația dispune de diverse funcționalități care îi vin în ajutorul utilizatorului, având o structură cât mai simplă pentru a putea fi înțeleasă de fiecare utilizator. Aceste funcționalități sunt prezentate în diagrama de mai jos:

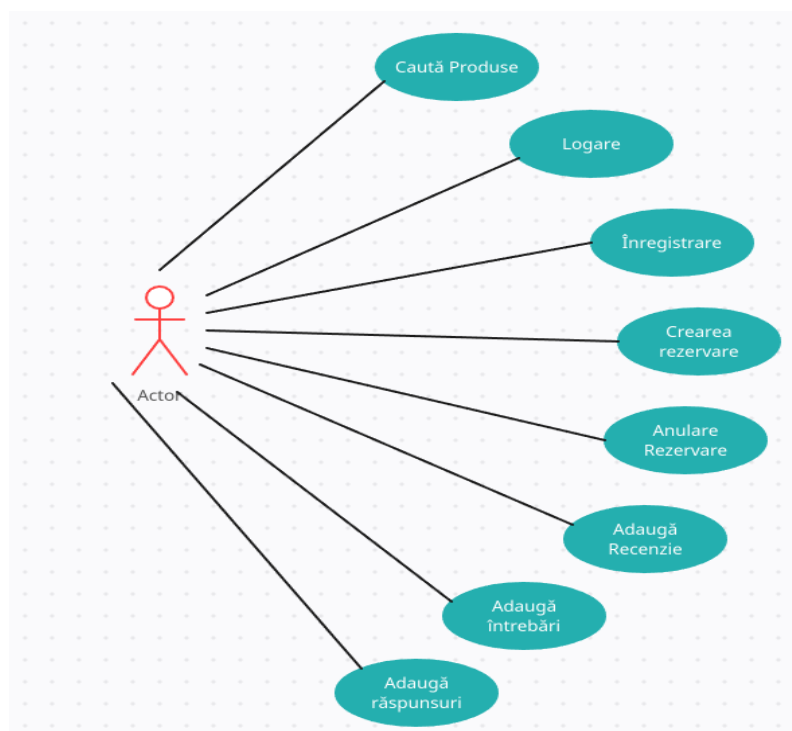


Fig 3.1 - Diagrama de cazuri

Din perspectiva utilizatorului, acesta își poate crea cont, să facă o rezervare la un tehnician, evaluează un tehnician, interacționează pe forum cu alți utilizatori, amână rezervări, își creează propria listă de produse favorite, are acces la chat. Din perspectiva tehnicianului, acesta are aceleași funcționalități ca ale utilizatorului adăugându-se personalizarea de servicii și finalizarea unei rezervări.

3.2 Backend

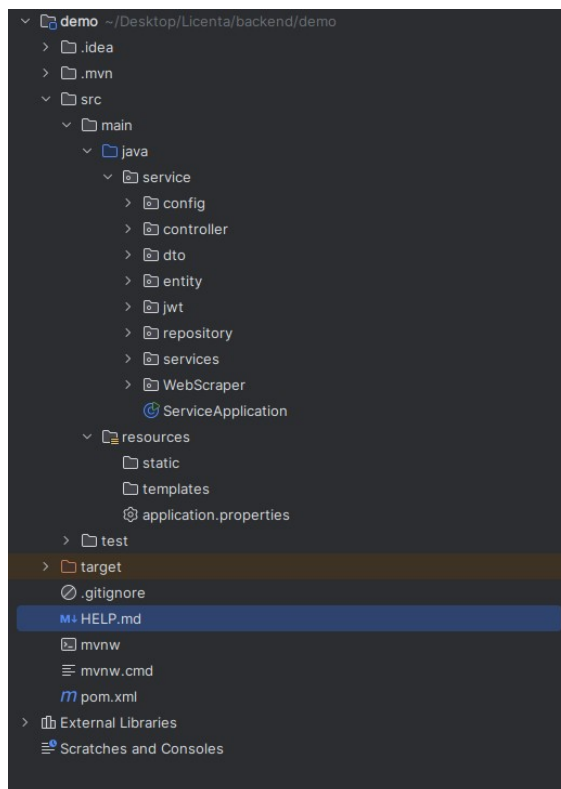


Fig 3.2.1 – Structura fișierelor din aplicația Spring Boot

Principalul director al unei aplicații Spring Boot este src. Acesta conține una din cele mai populare stiluri arhitecturale și anume structura ”Layered” cu următoarele directoare:

- În directorul config, după cum sugerează și numele vom găsi fișiere ce ajută la configurarea aplicației. În cazul aplicației dezvoltate de mine aici se găsesc fișiere ce configurează securitatea aplicației, în special pentru partea de înregistrare și autentificare a utilizatorului.
- La nivelul directorului controller se află toate controlerele care se vor ocupa cu cererile primite de API.
- Directorul entity conține tipurile de date ce provin nemodificate din baza de date. Se ocupă cu maparea tabelor din baza de date pentru procesarea datelor.

- În directorul repository se află interfețe ce se ocupă de operațiile CRUD(Create, Read, Update, Delete).
- Pachetele din cadrul directorului service se ocupă cu separarea logicii aplicației cu cea a controlerului.
- În directorul dto (Data Transfer Object) se află fișiere ce conțin pachete publice ce au rolul de a realiza tipuri de date potrivite pentru cererile API.
- Directorul jwt se ocupă cu securitatea aplicației, având rolul de autorizare al utilizatorului.
- În directorul WebScrapper se găsește logica de preluare al datelor de pe diferite site-uri. Acesta este împărțit într-o interfață comună și 3 clase care implementează interfața.

Pe lângă directoarele menționate mai sus, aplicația de tipul Spring Boot generează fișiere care se pot configura și de către dezvoltator. Un fișier important din acest director este .yaml sau .properties unde sunt configurate caracteristici ale aplicației precum: conectarea la baza de date sau noțiuni legate de securitate.

Nu în ultimul rând, un fișier foarte important este pom.xml. Acesta are rolul de a configura aplicația Spring Boot după nevoile utilizatorului. Aici sunt predefinite câteva dependențe de către aplicație, însă dezvoltatorul are dreptul de adăugare sau ștergere a diferitelor dependențe precum cele legate de baza de date, de securitate sau tool-uri ajutătoare în scrierea codului precum Lombok.

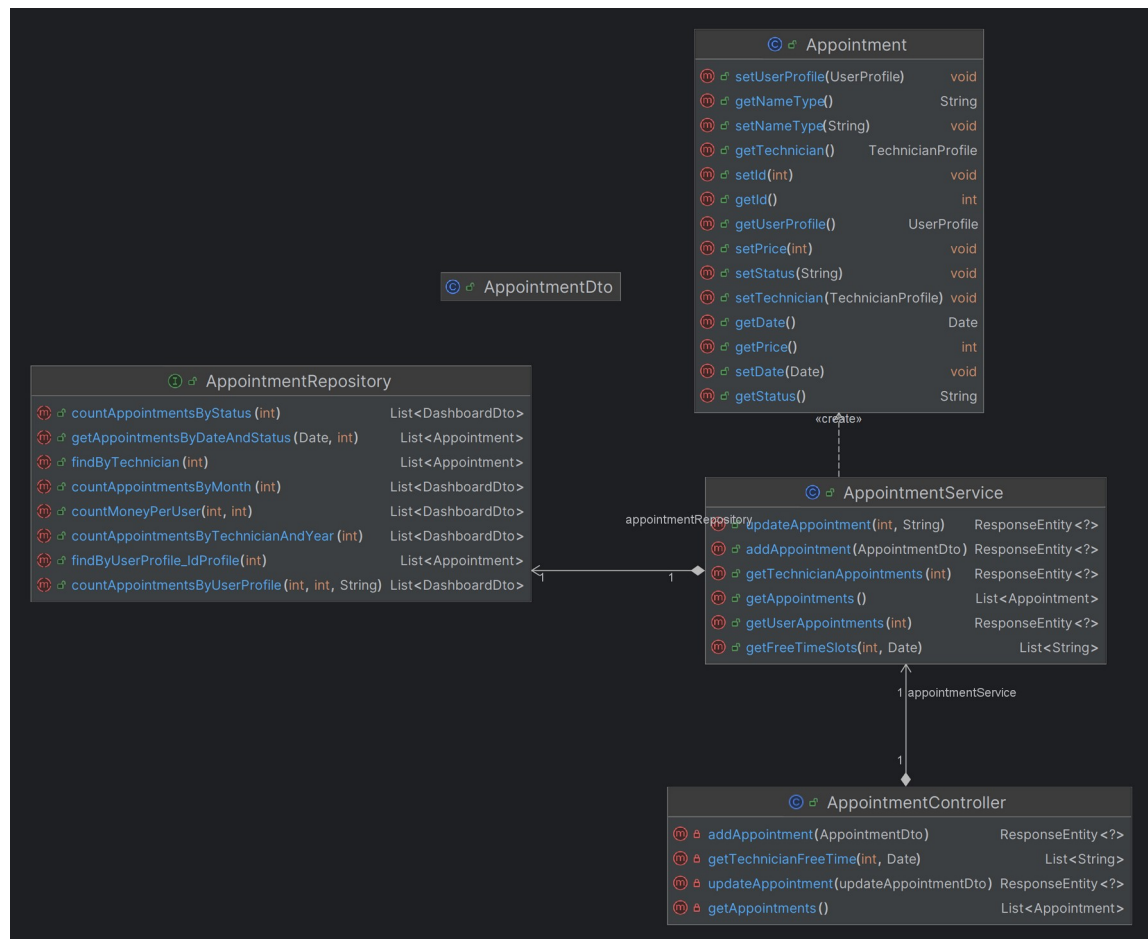


Fig 3.2.2 – Exemplu structură ierarhică

3.3 Frontend

```

Frontend
├── node_modules
├── service
├── .angular
├── .vscode
├── node_modules
├── src
│   ├── app
│   │   ├── components
│   │   │   ├── appointment
│   │   │   ├── chat
│   │   │   ├── compari
│   │   │   ├── dashboard
│   │   │   ├── forum
│   │   │   ├── home
│   │   │   ├── jwt-dialog
│   │   │   ├── notification
│   │   │   ├── profile
│   │   │   ├── review
│   │   │   ├── sign-in
│   │   │   └── sign-up
│   │   ├── model
│   │   └── service
│   ├── app-routing.module.ts
│   ├── app.component.css
│   ├── app.component.html
│   ├── app.component.spec.ts
│   ├── app.component.ts
│   └── app.module.ts
│   ├── assets
│   │   ├── favicon.ico
│   │   └── index.html
│   ├── main.ts
│   └── styles.css
  
```

Fig 3.3 – Structura fișierelor din aplicația Angular

În structura proiectului de Frontend regăsim:

- Directorul model care are aceeași structură ca al directorului dto din partea de backend și aceleași funcționalități.
- În service găsim mai multe servicii responsabile pentru comunicarea cu API-ul creat de noi.
- La nivelul directorului components se găsesc paginile propriu-zise ale aplicației, acestea fiind împărțite în 3 fișiere diferite.

Nu în ultimul rând, putem observa componenta principală denumită app. Acesta reprezintă pagina de start de care se leagă direct sau indirect celelalte componente. Acesta conține un fișier cu rolul de a importa anumite dependențe din proiect, precum cele din stilizarea cu Angular Material, dar și rolul de a realiza rutarea între componente.

4. Implementarea aplicației

4.1 Structura bazei de date

După cum am menționat în secțiunile precedente, în vederea dezvoltării aplicației am folosit o bază de date de tipul MariaDB. Aceasta este creată local, pe computerul ce găzduiește aplicația prin intermediul programului Docker ce oferă posibilitatea de a crea, porni sau configura anumite servicii. Pentru crearea tabelelor, crearea constrângerilor, relațiile dintre tabele și interogări am folosit aplicația DbBeaver. Structura tabelelor este un ierarhică prezentată în figura următoare:

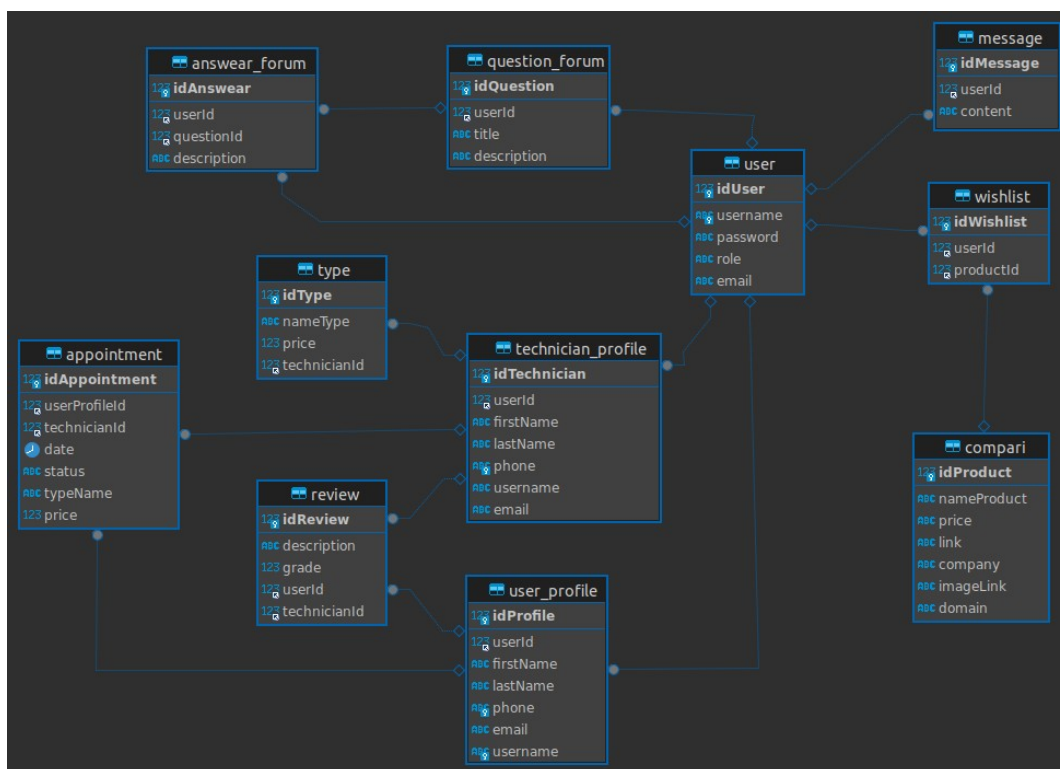


Fig 4.1 – Structură baza de date

Tabelele prezente în aplicație sunt următoarele:

- User este tabela ce stochează datele utilizatorilor simpli cât și pe cele ale tehnicienilor având câmpurile: id, username, password, role, email. Aceasta este folosită la logare.

- UserProfile reprezintă tabela ce stochează datele personale ale utilizatorului precum nume, prenume, email, parolă, număr de telefon, username. Această tabelă are ca și cheie străină id-ul din tabela principală User. Această tabelă este utilizată la înregistrarea unui utilizator nou, dar și la crearea de rezervări.
- TechnicianProfile este asemănătoare cu cea UserProfile, numai că aici sunt stocate datele tehnicienilor.
- În tabela Type sunt definite serviciile pe care un tehnician le prestează și prețul acestora. El are ca și cheie străină id-ul unui tehnician din tabela TechnicianProfile, întrucât fiecare tehnician are propriile servicii.
- Review reprezintă tabela ce stochează date despre recenziile scrise de clienți. Acesta conține un câmp unde este descrisă experiența utilizatorului și un câmp ce reprezintă nota acordată. Acesta dispune de 2 chei străine, una cu id-ul din tabela UserProfile și una cu id-ul din tabela TechnicianProfile.
- Appointment este o tabelă ce stochează datele despre rezervările ce sunt create de către utilizatorii obișnuiți. Astfel că, această tabelă conține 2 chei străine, pe cea a utilizatorului, cât și pe cea al tehnicianului. Pe lângă aceste, tabela mai conține date despre data programării, serviciul, prețul și statusul programării.
- În tabela Compari sunt stocate datele preluate de către Web Scrapper. Această tabelă nu depinde de alte tabele și conține următoarele date: numele produsului, prețul acestuia, link-ul, url-ul imaginii, domeniul și compania care îl vinde.
- Wishlist tabela ce conține date ale produselor favorite ale unui utilizator. Acesta are 2 chei străine, una cu id-ul produsului din tabela Compari și una cu id-ul unui utilizator din tabela User.
- Tabela Message se ocupă cu stocarea datelor chatului. Ea este alcătuită dintr-o cheie externă având referință la tabela User și un câmp cu conținutul mesajului trimis.

- Question_forum reprezintă tabela unde sunt stocate întrebările utilizatorilor legate de problemele pe care le întâmpină. Aceasta are referință la tabela User, având câmpurile titlu și descrierea problemei.
- Answer_forum tabela are referință la tabela Question_forum, întrucât aici sunt salvate răspunsurile utilizatorilor pentru întrebările altor utilizatori.

4.2 Securizarea aplicației

În vederea securizării aplicației m-am folosit de un framework pus la dispoziție de către Spring Boot numit Spring Security ce oferă suport de autentificare, autorizare și protecție împotriva atacurilor comune. Autentificarea este modul prin care se verifică identitatea celui care încearcă accesarea anumitor date este validă. Modul de autentificare folosit de mine, este unul foarte comun, format din nume și parolă, stocate în baza de date într-o formă securizată de 32 caractere generate cu ajutorul pachetului PasswordEncoder. Acest pachet transformă parolele prin algoritmul SHA256, într-un singur sens, adică nu pot fi decriptate cu vreun alt algoritm. Astfel, la autentificare se realizează compararea parolei din baza de date, care a fost deja criptată la înregistrare, cu parola furnizată de utilizator la momentul autentificării.

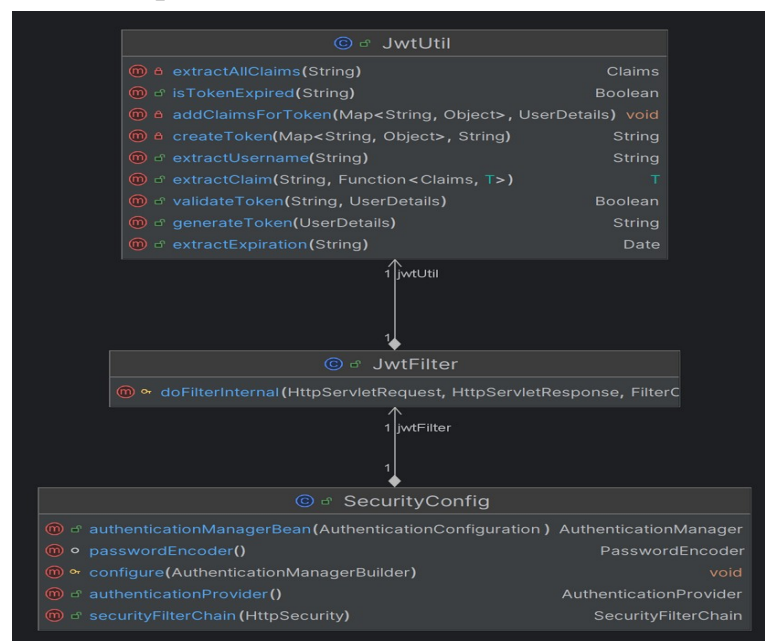


Fig 4.2 – Diagrama de clasă a securității

4.3 Rutarea

Rutarea reprezintă posibilitatea de a afișa sau a ascunde părți ale aplicației în funcție de utilizatorul ce folosește aplicația. În Angular pentru a gestiona această rutare se folosește Angular Router ce permite navigarea în SPA (Single Page Application) pe baza URL-urilor.

Cazurile pe care le-am luat în considerare sunt: utilizatorul nu este autentificat, acesta având acces doar la pagina de home, dashboard, login și register. Accesul este destul de limitat atunci când utilizatorul nu este autentificat, oferindu-i doar date statistice despre site. Cazul în care utilizatorul este autentificat îi permite utilizatorului să acceseze diverse servicii pe care aplicația le oferă cum ar fi: crearea de rezervare, accesarea profilului personal, adăugarea de recenzii, acces la forum și acces la chat. URL-urile sunt identice chiar dacă se autentifică un utilizator obișnuit sau un tehnician, dar interfețele diferă în funcție de rolul fiecăruia.

În momentul în care utilizatorul dorește să acceseze un URL ce nu face parte din rularea definită, acesta va fi redirecționat către pagina principală. Același lucru se va întâmpla și dacă utilizatorul nu este autentificat și dorește să acceseze URL-urile definite.

```
const routes: Routes = [  
  { path: 'login', component: SignInComponent },  
  { path: 'register', component: SignUpComponent },  
  { path: 'profile', component: ProfileComponent },  
  { path: 'appointment', component: AppointmentComponent },  
  { path: 'forum', component: ForumComponent },  
  { path: 'forum/:id', component: ForumComponent },  
  { path: 'compari', component: CompariComponent },  
  { path: 'home', component: HomeComponent },  
  { path: 'review', component: ReviewComponent },  
  { path: 'dashboard', component: DashboardComponent },  
  { path: '**', redirectTo: '/home' },  
];
```

Fig 4.3 - Rutarea aplicației

4.4 Accesarea datelor

4.4.1 Legătura dintre API și baza de date

În vederea accesării datelor din baza de date de către aplicația Spring Boot am folosit o interfață integrată numită JpaRepository. Această interfață extinde o tabelă adnotată ca "Entity" și are anumite metode predefinite pentru a face mai ușoară gestiunea datelor. Printre metodele pe care le oferă JpaRepository se numără: metoda findAll() ce întoarce toate datele din tabela respectivă, metoda findBy() care are rolul de a extrage date în funcție de unu sau mai mulți parametrii. Pentru partea de inserarea există funcția save(), iar dacă datele pe care le dorim să le introducem se regăsesc deja în tabelă, acesta face actualizare. Această interfață permite dezvoltatorului să își creeze propriile interogări adăugând adnotarea Query deasupra funcției.

```
public interface ReviewRepository extends JpaRepository<Review, Integer> {  
    List<Review> findByTechnicianProfile_IdTechnician(int idTechnician); 1 usage  
  
    Float findAvgGradeByTechnicianId(int TechnicianId); 3 usages  
  
    @Query("SELECT new service.dto.DashboardDto(AVG(r.grade), t.firstName, t.lastName) " +  
        "FROM Review r JOIN r.technicianProfile t " +  
        "GROUP BY t.idTechnician " +  
        "ORDER BY AVG(r.grade)")  
    List<DashboardDto> findAvgGradeTechnicians();  
  
    @Query("SELECT COUNT(r.grade) FROM Review r WHERE r.technicianProfile.idTechnician = :TechnicianId") 2 usages  
    Float findCountGradeByTechnicianId(int TechnicianId);  
}
```

Fig 4.4.1 – Interfață ce folosește Jpa Repository

4.4.2 Legătura dintre API și UI

Pentru realizarea conexiunii dintr API și interfața pe care o folosesc utilizatorii am folosit clasa HttpClient. Acesta este folosită pentru a trimite cereri și pentru a primi răspunsuri de la API. Pe partea de Spring Boot am implementat diferite metode numite și controlere ce dispun de adnotări specifice precum GetMapping, PostMapping, PutMapping și DeleteMapping ce pot avea diferiți parametrii.

Fluxul de date este început de aplicația Angular cu ajutorul clase HttpClient se apelează una din cele 4 verbe HTTP, get, post, put sau delete, cu câmpurile pe care le așteptăm ca răspuns în urma cererii. În partea de Spring Boot definire parametrilor trebuie adnotate specific cu RequestBody sau PathVariable. O descriere mai amplă al acestui flux este prezentat în figura următoare:

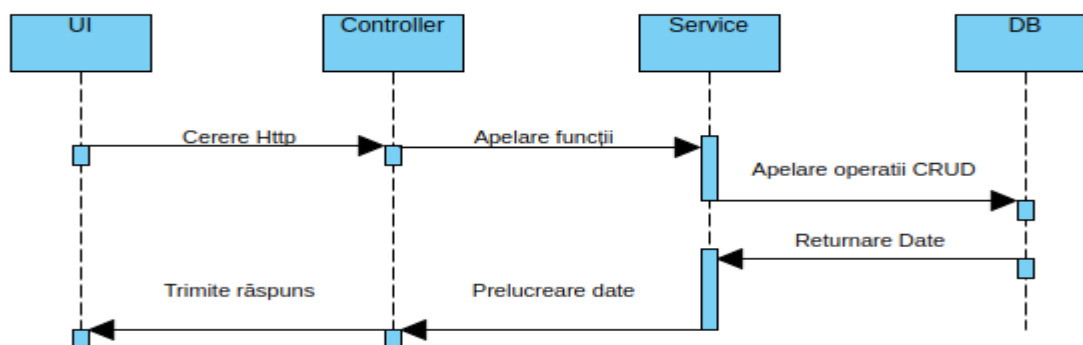


Fig 4.4.2 – Flux date API și UI

4.4.3 Partajarea datelor între componentele

În vederea partajării datelor între componente am folosit unealta Local Storage ce se găsește în toate aplicațiile web. Astfel că, atunci când am nevoie de o anumită dată pe care trebuie să o folosesc în alte componente, o stochez în Local Storage prin metoda `setItem`, primul parametrul fiind numele datei, iar al doilea reprezintă valoarea acestuia. Accesarea datelor din local storage se realizează foarte ușor prin metoda `getItem`, fiind accesibil din orice punct al aplicației.

```
public setTokenStorage( token: any, username: any): void {  
    localStorage.setItem('token', token);  
    localStorage.setItem('username', username);  
}
```

Fig 4.4.3 – Utilizarea Local Storage

4.5 Serviciul de mail

Utilizatorii sunt nevoiți să ofere la înregistrare date despre mail-ul propriu. Acest lucru este necesar deoarece atunci când utilizatorul creează o rezervare sau starea rezervării este modificată, acesta este ținut la curent într-un mod eficient, primind un mail cu starea rezervării, nefiind nevoie să verifice constat starea rezervării din aplicație.

În implementarea serviciului de mail am folosit un pachet pus la dispoziție de Spring Boot prin dependența Spring Boot Starter Mail. Dependența am adăugat-o în fișierul pom.xml, dar după aceasta în fișierul application.properties am setat proprietățile, și anume, host-ul, username-ul mail-ului și parola. Pentru a eficientiza trimiterea de mail-uri am folosit câte un thread la fiecare trimitere pentru a nu bloca aplicația așteptând finalizarea mesajului trimis.

4.6 Web Scrapper

Web scrapping-ul este procesul de extragere al datelor de pe site-uri web. Acesta este de 2 tipuri: manual în care dezvoltatorul extrage datele manual de pe site-uri și cel automat în care dezvoltatorul implementează un bot numit și crawler web care se ocupă de extragere. În aplicația dezvoltată de mine, am implementat un bot care extrage date de pe site-urile Cel.ro, Emag și Evomag pentru a crea un API pentru pagina compari. Bot-ul extrage date precum: numele produsului, prețul acestuia, url-ul imaginii, link-ul către produs prin selectori css și le stochează în baza de date. El este programat să se apeleze în fiecare zi la ora 00:00 pentru a fi la curent cu produsele populare. Având în vedere ca pentru fiecare site, botul folosește câte un thread pentru fiecare categorie de produs, acesta atinge o performanță de extragere a 4000 de produse într-un timp de 20 secunde.

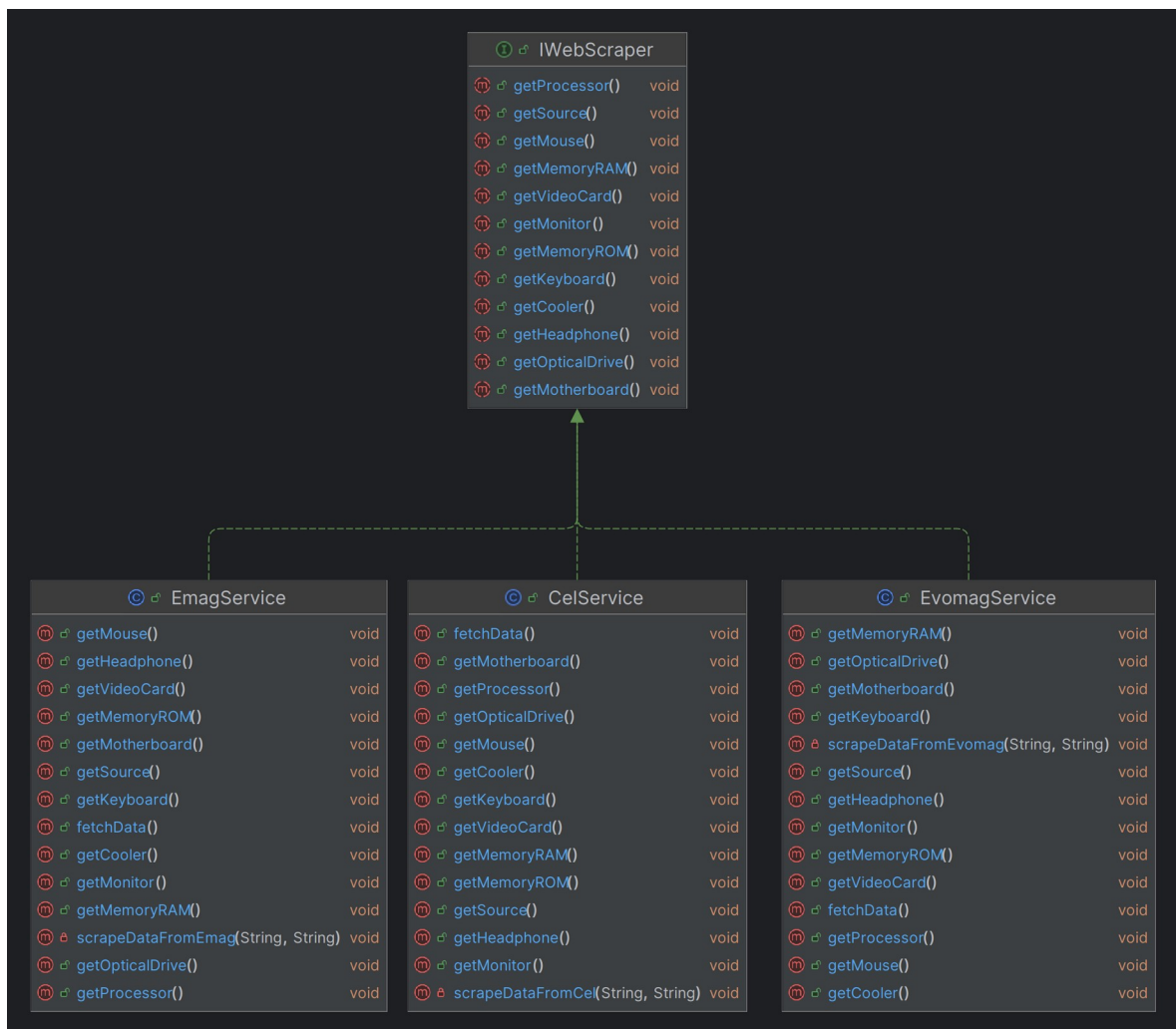


Fig 4.6 – Structura WebScraper-ului

5. Testarea sistemului și rezultate experimentale

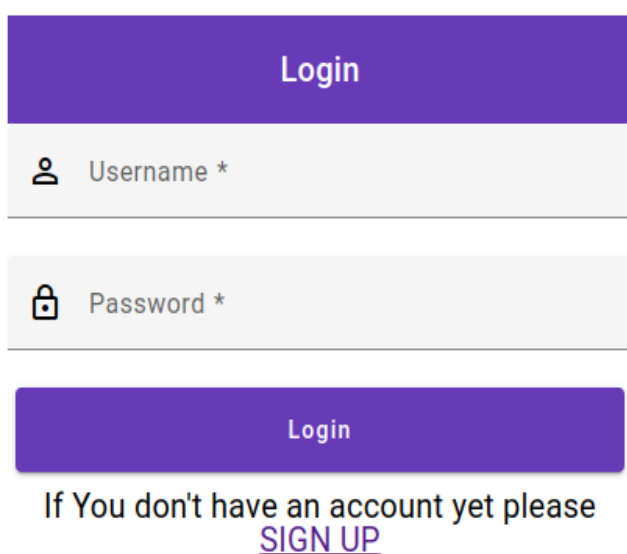
Pe partea de testare, am folosit o testare manuală încercând diverse acțiuni care ar duce la blocarea aplicației. Spring Boot dispune și de un fișier special creat pentru teste unitare, pe diferite controlere sau teste de integrare pe toată aplicația, însă nu face parte din obiectivul aplicației dezvoltate de mine.

5.1 Componentele interfeței

5.1.1 Înregistrare și autentificare

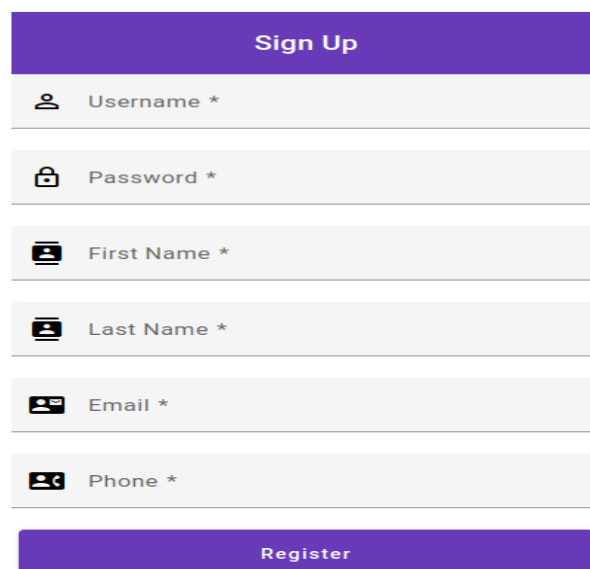
În vederea autentificării, utilizatorul trebuie să introducă credențialele sub forma username-ului și a parolei. Procesul de autentificare constă în următorul proces: interfața trimite datele introduse către backend, acesta criptează parola și verifică token-ul generat. În caz afirmativ, token-ul este trimis înapoi în interfață unde îi permite utilizatorului să acceseze diferite servicii ale aplicației. În caz contrar se va trimite o eroare interfeței.

În momentul în care utilizatorul dorește să se înregistreze, acestuia i se prezintă un formular simplu cu diferite câmpuri care au constrângeri, și anume, parola trebuie să aibă lungimea minimă de 6 caractere, emailul trebuie să fie unul valid, username obligatoriu.



The login form consists of a purple header with the text "Login". Below the header are two input fields: "Username *" with a person icon and "Password *" with a lock icon. At the bottom is a purple button labeled "Login". Below the button is a text link that says "If You don't have an account yet please [SIGN UP](#)".

Fig 5.1.1.1 - Formular de autentificare



The sign up form consists of a purple header with the text "Sign Up". Below the header are six input fields: "Username *" with a person icon, "Password *" with a lock icon, "First Name *" with a person icon, "Last Name *" with a person icon, "Email *" with an email icon, and "Phone *" with a phone icon. At the bottom is a purple button labeled "Register".

Fig 5.1.1.2 – Formular de înregistrare

5.1.2 Perspectiva utilizatorului

Utilizatorul are acces la componenta ce reprezintă profilul acestuia unde are posibilitatea de a vizualiza programările, lista cu produse favorite și date statistice despre numărul de programări și totalul cheltuit. Totodată la secțiunea profil poate anula programările și șterge date din lista de produse favorite.

La nivelul componentei programări, utilizatorului îi sunt prezentate lista cu tehnicieni, cu detalii despre ei, iar în momentul când utilizatorul selectează un tehnician, îi sunt prezentate serviciile pe care le prestează și prețurile acestora, după selectarea serviciului, utilizatorul trebuie să aleagă data când vrea să facă programarea.

Schedule an Appointment

Please follow the steps below to schedule an appointment with a technician:

Technicians Services Summary

Select a Technician

Select a technician from the list below:

Name: Andrei Marian Email: andreimarian22@gmail.com Phone: 0789989892 Grade: 6.75	Select
Name: Dani Martisca Email: daniel.martisca3@gmail.com Phone: 0754786521 Grade: 5.5	Select
Name: Filip Dimisca Email: filipdimis1999@gmail.com Phone: 0799876212 Grade: 7.3333335	Select

Fig 5.1.2.1 – Listă tehnicieni

Schedule an Appointment

Please follow the steps below to schedule an appointment with a technician:

Technicians **Services** Summary

Select a Service

Select a service from the list below:

Service: Curatare laptop Price: 20 lei	Select
Service: Instalare SO Price: 70 lei	Select
Service: Schimbare Placa Video Price: 125 lei	Select

Fig 5.1.2.2 – Listă servicii

Pentru evitarea situațiilor de programare a unei probleme care se rezolvă fără programare, aplicația îi pune la dispoziție un chat și un forum utilizatorului. Chat-ul este unul comun pentru toți utilizatorii indiferent de rol, iar în el sunt ultimele 20 de mesaje trimise de utilizatori. Forum-ul este menit să păstreze întrebările și răspunsurile în cazul în

care alți utilizatori întâmpină aceleași probleme. Utilizatorul are acces la aceste servicii doar atunci când este autentificat.

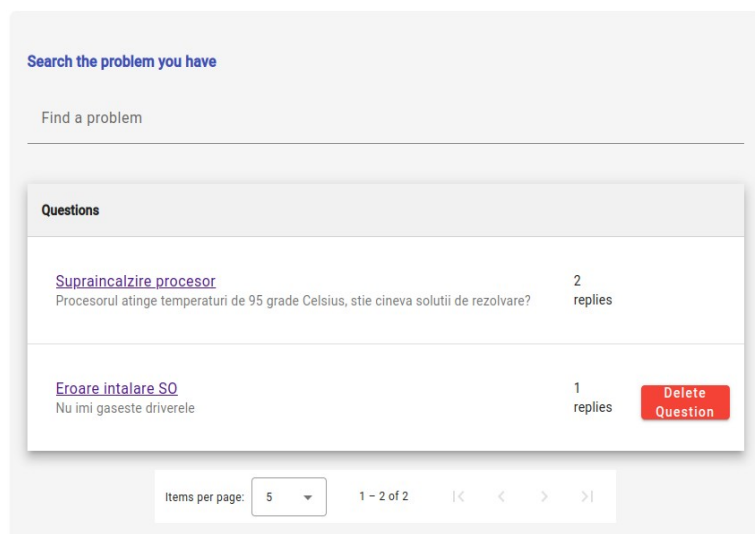


Fig 5.1.2.3 – Pagina de întrebări



Fig 5.1.2.4 – Pagina de răspunsuri

Utilizatorii își pot exprima opiniile în legătură cu modul de lucru al tehnicienilor având posibilitatea de evaluare al acestora. Această recenzie poate face referire atât la relația pe care a avut-o clientul, cât și la calitatea lucrărilor pe care tehnicianul le-a efectuat. Tot în acest formular, clientul are posibilitatea de a alege o notă ce se află în intervalul 1 și 10. Pe baza notelor, aplicația calculează media tuturor recenziilor și o afișează în dreptul fiecărui tehnician.

Aplicația îi mai pune la dispoziție utilizatorului și o pagină de comparați în care utilizatorul caută produse de pe diverse site-uri. Acest lucru îl ajută pe utilizator cu o privire de ansamblu al prețurilor de pe piață al produselor. Utilizatorul trebuie doar să selecteze un domeniu din cele predefinite de aplicație, un parametru sau un brand al produsului și un interval de prețuri.

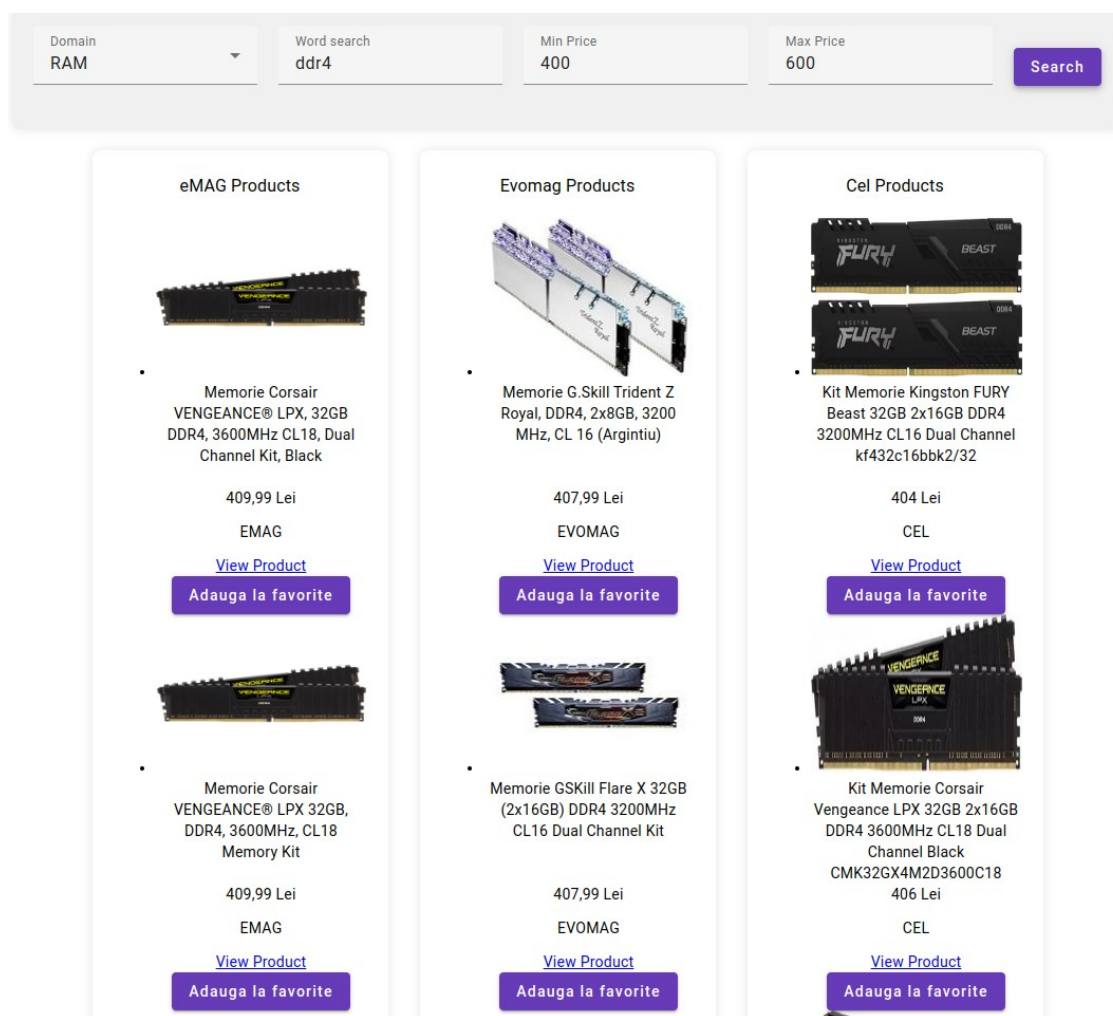


Fig 5.1.2.5 – Pagina Compari

De asemenea, aplicația are la dispoziție o pagină ce conține statistici ale tehnicienilor precum numărul de programări al fiecărui tehnician împărțite pe diverși ani, media recenziilor tehnicienilor, numărul total de programări pe fiecare an. Această caracteristică îl va ajuta pe utilizator să-și facă o privire de ansamblu al aplicației, în cazul în care este un utilizator care accesează pentru prima dată platforma.

5.2.3 Perspectiva Tehnician

În contextul în care utilizatorul are rolul de tehnician, acesta are acces la aceleași pagini ca un utilizator obișnuit dar cu anumite privilegii. Utilizatorul de tip tehnician are în pagina profilului un buton prin care își poate modifica serviciile pe care le prestează, le pot edita, șterge sau adăuga unele noi. Aceste modificări nu influențează rezervările deja existente.

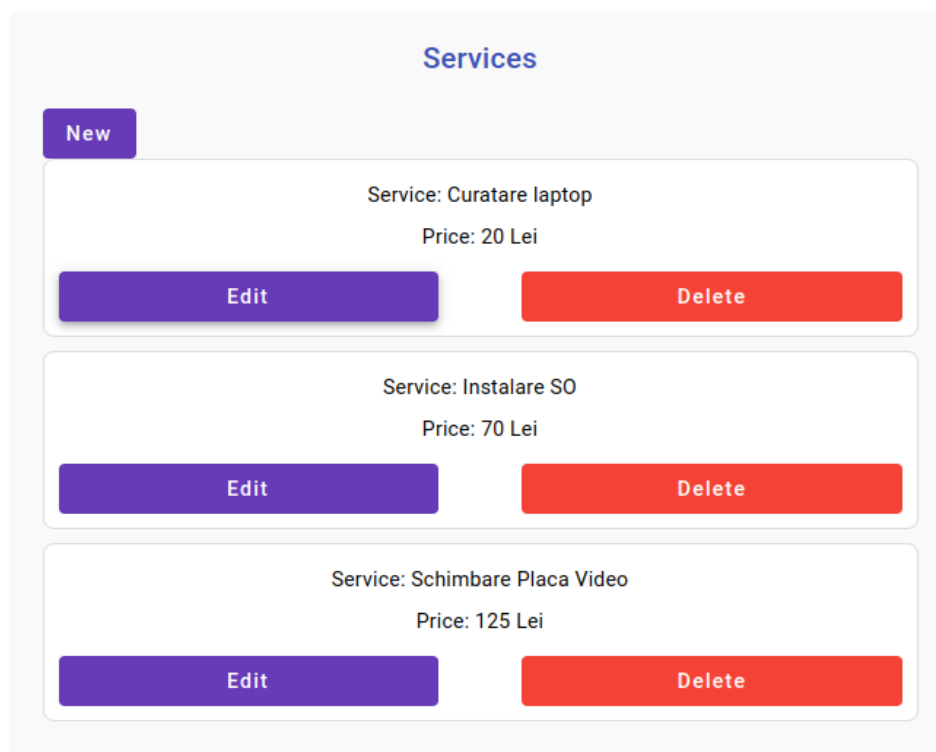


Fig 5.2.3.1 – Gestionare servicii

O altă funcționalitate pe care o poate accesa acest tip de utilizator este cea a gestionării programărilor aferente profilului său, astfel că, prin accesarea componentei Profile, acesta are la dispoziție o tabelă cu toate programările atribuite lui.

Tehnicianul are posibilitatea de a trece starea programării din așteptare, starea inițială, în stări precum finalizat, anulat sau nu se poate rezolva. Acest proces se realizează prin accesarea unor butoane dintr-o coloană specială numită Actions. Semnul de x reprezintă statusul de anulare, cel de al doilea semnifică faptul că programarea este finalizată, iar cel de al 3-lea semn trece programarea în statusul de nu se poate repara.

La fiecare schimbare a statusului unei programări, clientul va primi pe adresa de email pe care a introdus-o în momentul înregistrării, un mail ce specifică statusul inițial și cel în care s-a schimbat. În corpul aceluși mail sunt specificate detaliile despre programare pentru o identificare mai bună.







Service	Price	Data	User	Phone	Email	Status	Actions		
Instalare SO	60 lei	27/06/2024 09:00:00	cristi cristi	0795949441	cristiandumea96@gmail.com	Canceled			
Instalare SO	70 lei	25/06/2024 10:00:00	Marian Mirt	0790067907	marian.134@gmail.com	Pending			
schimbare pasta	30 lei	21/06/2024 18:00:00	cristi cristi	0795949441	cristiandumea96@gmail.com	Done			
Curatare laptop	20 lei	21/06/2024 13:00:00	cristi cristi	0795949441	cristiandumea96@gmail.com	Pending			
schimbare pasta	30 lei	21/06/2024 11:00:00	cristi cristi	0795949441	cristiandumea96@gmail.com	Done			

Fig 5.2.3.2 – Tablou de gestionare a rezervărilor

6. Direcții viitoare

Următoarele funcționalități ce ar putea fi integrate în aplicația dezvoltată constau în crearea unui tablou de comandă pentru un eventual administrator al aplicației, acesta având drepturi depline pe toată platforma. Astfel că, dacă se dorește să se creeze un utilizator tehnician, administratorul ar avea datoria să verifice informațiile oferite și abilitățile viitorului angajat.

De asemenea, dat fiind contextul actual în care PlayStation devin tot mai populare și de la un an la altul numărul lor crescând semnificativ, ar fi utilă extinderea domeniului de lucru, adăugând servicii specifice. Prin lărgirea domeniului, aplicația își micșorează riscul de a își pierde relevanța odată cu dispariția calculatoarelor într-un viitor mai mult sau mai puțin îndepărtat.

Din punct de vedere al utilizatorului, pe viitor ar fi util afilierea aplicației cu mai multe site-uri de unde poate să extragă date pentru comparații, dar și un serviciu prin care utilizatorul poate să comande direct prin intermediul platformei componentele necesare. Acest lucru l-ar ajuta pe utilizator pentru a economisi timp de căutare, dar și garanția că piesele comandate sunt cumpărate de pe un site de încredere.

7. Concluzie

În concluzie, am reușit să dezvolt o aplicație ce rezolvă problema oamenilor în a găsi un suport tehnic pentru problemele de calculatoare. Totodată, cred că aplicația oferă un avantaj semnificativ persoanelor care sunt la început de drum cu tehnologia și doresc să intre în era tehnologiei, dar și a persoanelor care doresc să învețe câte ceva nou din experiențele altor persoane.

Dezvoltarea acestei aplicații m-a ajutat totodată pe mine ca dezvoltator în descoperirea unor noi tehnici de programare sau în consolidarea celor pe care deja le cunoșteam, dar cel mai important lucrul pe care l-am învățat este aceea de a reuși să ofer un flux intuitiv unei aplicații. Astfel, aplicația este destinată tuturor utilizatorilor, neavând nevoie de cunoștințe sau de o instruire în prealabil.

Bibliografie

- [1] „Compari”: <https://www.compari.ro/>
- [2] „Stack Overflow”: <https://stackoverflow.co/>
- [3] „IntelliJ”: <https://www.jetbrains.com/help/idea/discover-intellij-idea.html>
- [4] „codegym”: <https://codegym.cc/ro/quests/lectures/ro.questservlets.level14.lecture00>
- [5] „SpringBoot”: <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>
- [6] „HTML”: <https://www.dcpvhpm.org/E-Content/BCA/BCA-II/Web%20Technology/the-complete-reference-html-css-fifth-edition.pdf/> Autor: Thomas A. Powell
- [7] „CSS”: <https://www.freecodecamp.org/news/how-to-use-css-overview-in-chrome-developer-tools/> Autor: CESS
- [8] „Angular”: <https://angular.io/guide/architecture>
- [9] „Angular Material”: <https://material.angular.io/>
- [10] „JavaGuides”: https://www.javaguides.net/2021/07/spring-boot-tutorial-for-beginners.html?utm_content=cmp-true/ Autor: Ramesh Fadatar
- [11] „cotidianul”: <https://www.cotidianul.ro/cati-romani-au-calculator-acasa/> Autor: I.R.
- [12] „WebScrapping”: <https://www.scrapingbee.com/java-webscraping-book/>
Autor: Kevin Sahin
- [13] „Promise”: <https://www.infragistics.com/community/blogs/b/infragistics/posts/angular-observable-vs-angular-promise/> Autor: Desislava Dincheva
- [14] „MVC Arhitecture”: <https://towardsdatascience.com/everything-you-need-to-know-about-mvc-architecture-3c827930b4c1> Autor: Zanfina Svirca
- [15] „RESTful”: <https://www.techtarget.com/searchapparchitecture/definition/RESTful-API> Autor: Stephen J. Bigelow
- [16] „MariaDB”: <https://mariadb.org/about/>
- [17] „Postman”: <https://www.geeksforgeeks.org/introduction-postman-api-development/>
Autor: Patrikshit Hooda
- [18] „ng2-charts”: <https://www.npmjs.com/package/ng2-charts>
- [19] „Visual Studio Code”: <https://www.educative.io/answers/what-is-visual-studio-code>
Autor: Anusheh Zohair Mustafeez

Listă figuri

1. Fig 1.3.1 – Pagină principală Compari
2. Fig 1.3.2 – Pagină principală Stack Overflow
3. Fig 2.1.1 – Logo IntelliJ
4. Fig 2.1.2 – Logo Visual Studio Code
5. Fig 2.2 – Arhitectura client-server
6. Fig 2.2.1 – Logo Spring Boot
7. Fig 2.2.2 – Logo MariaDB
8. Fig 2.3.1 – Logo HTML5
9. Fig 2.3.2 – Logo CSS
10. Fig 2.3.3 – Logo Angular Material
11. Fig 2.3.4 – Logo Angular
12. Fig 3 – Fluxul de activități
13. Fig 3.1 – Diagrama de cazuri
14. Fig 3.2.1 – Structura fișierelor din aplicația Spring Boot
15. Fig 3.2.2 – Exemplu structură ierarhică
16. Fig 3.3 – Structura fișierelor din aplicația Angular
17. Fig 4.1 – Structură baza de date
18. Fig 4.2 – Diagrama de clasă a securității
19. Fig 4.3 – Rutarea aplicației
20. Fig 4.4.1 – Interfață ce folosește Jpa Repository
21. Fig 4.4.2 – Flux date API și UI
22. Fig 4.4.3 – Utilizarea Local Storage
23. Fig 4.6 – Structura WebScraper-ului
24. Fig 5.1.1.1 – Formular de autentificare
25. Fig 5.1.1.2 – Formular de înregistrare
26. Fig 5.1.2.1 – Listă tehnicieni
27. Fig 5.1.2.2 – Listă servicii
28. Fig 5.1.2.3 – Pagina de întrebări
29. Fig 5.1.2.4 – Pagina de răspunsuri
30. Fig 5.1.2.5 – Pagina Compari
31. Fig 5.2.3.1 – Gestionare servicii
32. Fig 2.2.3.2 – Tablou de gestionare a rezervărilor