Departamentul Automatică și Informatică Industrială
Facultatea de Automatică și Calculatoare
Universitatea POLITEHNICA din București

LUCRARE DE DIPLOMĂ

Aplicație de recomandare a itinerariului pentru vizitarea atracțiilor turistice

Coordonatori științifici
Profesor Dr. Ing. Moisescu Mihnea Alexandru
As. Drd. Ing. Iliuță Miruna-Elena

Absolvent
Iacob Ioana-Roxana

2023

Cuprins

Capitolul 1

Introducere

1.1. Context general și obiectivele lucrării

Activitatea de a călători este printre cele mai des întâlnite dorințe ale oricărei persoane atunci când vine vorba de a-și petrece timpul liber. Tinerii în special își doresc, pe lângă oportunitățile financiare, să aibă posibilitatea de a călători cât mai mult. Această activitate dezvoltă simțul cultural și poate fi relaxantă și distractivă, dar în același timp poate fi și obositoare, iar procesul de planificare a călătoriei poate deveni unul complex care consumă mult timp. Pe lângă timpul alocat pentru alegerea destinațiilor și rezervarea biletelor de avion sau a cazării pentru locația destinație, persoana care dorește să plece într-o vacanță trebuie să stabilească itinerariul și bugetul călătoriei.

Există două mari tipuri de turiști: cei care își planifică călătoria din timp și în detaliu, care știu exact ce obiective turistice vor vizita, și cei spontani, care nu au niciun plan. În cazul primei categorii menționate se pierde foarte mult timp pentru a cerceta locul care urmează a fi vizitat și pentru a organiza un itinerariu pe zile, dar este o variantă care asigură îndeplinirea scopului. În cel de al doilea caz, călătorul nu consumă mult timp în ceea ce privește planificarea, dar problema poate să apară în momentul în care ajunge la destinație deoarece neavând nimic programat poate să devină dezorientat și astfel să nu descopere complet experiențele din acel oraș ce ar putea fi făcute în zilele alocate vacanței. De asemenea, în ambele variante poate să apară greșeala umană, itinerariul să nu fie cel mai eficient în ceea ce privește ordinea în care obiectivele turistice sunt vizitate. Aceste probleme pot fi rezolvate prin automatizarea procesului de planificare a itinerariului.

Prezenta lucrare propune implementarea unei soluții care să îl ajute pe utilizator să își planifice concediile și vacanțele cât mai ușor și rapid, o aplicație web ce generează recomandări de itinerarii pe baza datelor de intrare (locația destinație, numărul de zile alocate vacanței, obiectivele turistice prioritare etc.). Pentru obținerea itinerariului de vizitare eficient al traseului utilizatorului se vor lua în considerare doar obiectivele turistice din locația destinație, de asemenea, în aplicație se vor putea genera hărți, prin care se evidențiază traseul pentru fiecare zi, și statistici a celor mai populare obiective turistice dintr-un oraș, a orașelor cu cele mai multe atracții turistice, și a obiectivelor turistice bazate pe recenziile utilizatorilor. Pentru

implementarea funcționalităților se vor integra tehnologii avansate și algoritmul de tip Greedy, bazându-se pe euristici.

În plus se vor lua în considerare și funcționalitățile minimale de dezvoltare a unei aplicații interne, administrative destinată echipelor ce asigură ca funcționalitatea de bază din aplicația web principală să funcționeze în mod corespunzător. Membrii echipelor având ca scop actualizarea și gestionarea informațiilor despre atracțiile turistice, utilizatorii înregistrați în sistem, diverse statistice și monitorizarea de recenzii ale aplicației, aceste date fiind utilizate în scopul de a asigura buna funcționare a aplicației.

1.2. Descrierea domeniului

Turismul face parte din cele mai importante industrii care stă la baza economiei la nivel global. În cadrul acestuia se regăsesc și alte sectoare precum cel al transporturilor, al cazărilor, al restaurantelor, al obiectivelor turistice și al organizatorilor de călătorii. Multe țări, precum Thailanda, Egipt, Grecia etc., sunt dependente financiar de activitatea turistică ce se desfășoară acolo.

În anul 2020, din cauza pandemiei declanșate de coronavirus, industria turismului a înregistrat pierderi extrem de mari, o scădere de 70% a sosirilor de turiști internaționali față de anul 2019, rezultând în pierderea slujbelor a sute de mii de oameni și având un impact direct asupra industriei transporturilor de persoane, hotelieră și culturală. După o perioadă de 2 ani de izolare, oamenii își doresc să viziteze diverse locuri și să experimenteze noi culturi. În 2022 s-a remarcat o creștere de 60% a sosirilor de turiști, iar în prezent se află pe o pantă ascendentă, dar încă nu va putea depăși nivelurile pre-pandemice [8].

Avansul tehnologic și-a lăsat amprenta și asupra acestui domeniu, dacă în trecut resursele pentru planificarea unei călătorii erau destul de limite, existând doar ghiduri turistice și agenții de turism menite să ajute persoana care dorește să călătorească. În momentul actual, datorită existenței Internetului, oamenii au acces la informațiile dorite mult mai rapid și ușor de oriunde s-ar afla, putând-și planifica călătoriile de unii singuri fără a mai interveni persoane intermediare în proces, printre acțiunile ce le pot face aceștia pentru organizarea unei călătorii se identifică: informarea în legătură cu diverse destinații, rezervarea biletelor de avion și a cazării, căutarea de atracții turistice, restaurante și evenimente culturale, și vizualizarea recenziilor lăsate de alți călători pentru locațiile dorite.

Modalitatea clasică de orientare este și ea digitalizată, hărțile fizice devenind doar simple accesorii ce sunt folosite de turiști pentru a le face fotografiile mai interesante. Datorită flexibilității și accesibilității pe care le oferă hărțile online, persoanele care au nevoie de ghidare, cu doar câteva click-uri poate primi indicațiile necesare, privind rutele, atracțiile turistice și alte puncte de interes din locația destinație. De asemenea reprezintă o variantă mult mai bună când vine vorba de direcționare, întrucât acestea oferă informații în timp real despre trafic, călătorul putând astfel să își aleagă ruta în mod eficient, în plus funcționalități, precum evidențierea traseului, ghidaj vocal și realitatea augmentată fac experiența utilizatorului mult mai plăcută.

1.3. Structura lucrării

Lucrarea cuprinde 6 capitole mari, acestea fiind descrise după cum urmează:

- **Capitolul 1** este o introducere generală care are scopul de a clarifica subiectul temei studiate, descriind domeniul de cercetare și scopul lucrării. Tot aici este explicată și structura lucrării

- **Capitolul 2** abordează într-un mod detaliat problema specifică domeniului ales, analizând atât soluțiile existente pe piață cât și cerințele funcționale și non-funcționale ale soluției propuse

- **Capitolul 3** este dedicat studiului și prezentării tehnologiilor ce au fost folosite pentru implementarea rezolvării problemei, cât și a analizării algoritmilor de căutare euristică relevanți în rezolvarea problemei

- **Capitolul 4** prezintă considerentele tehnice privind realizarea soluției, inclusiv arhitectura aplicației, detaliile de implementare și testarea aplicației, ilustrând procesul tehnic de dezvoltare

- **Capitolul 5** oferă un studiu de caz relevant pentru a ilustra aplicarea practică a soluției propuse și a demonstra beneficiile și aplicabilitatea acesteia într-un context real

- **Capitolul 6** sumarizează ceea ce se prezintă în lucrarea studiată, fiind dedicat concluziilor, analizează succesul soluției propuse și propune direcții viitoare de dezvoltare pentru a îmbunătăți și extinde lucrarea actuală

Capitolul 2

Prezentarea problemei abordate

2.1. Formularea problemei

Planificarea unui itinerariu, în ceea ce privește o vacanță, un concediu, poate deveni destul de complicată și obositoare, mai ales în cazul persoanelor care nu au suficientă răbdare sau timp să aloce pentru această activitate. Poate să dureze mult timp, deoarece necesită o cercetare detaliată asupra atracțiilor turistice din orașul dorit a fi vizitat: să verifice pentru fiecare în parte orele de deschidere și închidere, poziționarea lor, recenziile în legătură cu acestea pentru a stabili dacă trebuie să-l prioritizeze sau nu.

Fiecare atracție turistică are propria pagină web, unele sunt mai bine dezvoltate și organizează informația astfel încât să fie ușor accesibile de către oricine, iar în alte cazuri nu se respectă standardele care țin de crearea unei aplicații web [11], precum: paginile să se încarce în timp util, aspectul paginii să fie cât mai plăcut și conținutul să fie relevant și ușor de accesat altfel este foarte greu să găsești ceea ce ai avea nevoie, acest lucru fiind o problemă pentru cei care se ocupă de organizarea călătoriei. O altă problemă identificată este că nu toate site-urile suportă și alte limbi, ce sunt cunoscute internațional, cum este limba engleză, acestea utilizând doar limba maternă a țării respective, făcând navigarea printre pagini și înțelegerea conținutului mult mai greu sau chiar imposibil de realizat. Existența mai multor site-uri, ce încercă să centralizeze obiectivele turistice din diferite țări, orașe, îngreunează, de asemenea, procesul de planificare deoarece în cazul în care conținutul nu este actualizat apare fenomenul de inconsistență a datelor: programe și adrese diferite, având ca efect apariția sentimentului de nesiguranță asupra persoanei care vrea să planifice o călătorie.

Aplicația este destinată tuturor persoanelor care au în plan să călătorească, atât în scop recreațional cât și în interes de serviciu. În plus poate fi o alternativă bună pentru studenții care aleg să-și continue studiile în alt oraș și nu sunt familiarizați cu locul, cu ajutorul acestei aplicații aceștia pot să descopere orașul și să profite din plin de reducerile și beneficiile de a fi student.

Utilizatorii țintă în acest context sunt persoanele care dețin cunoștințe de bază în ceea ce privește utilizarea calculatoarelor și a Internetului. Prin urmare, aplicația dezvoltată ar trebui să fie ușor de accesat și utilizat de orice persoană.

2.2. Analiza soluțiilor similare din domeniu

În prezent, există multiple aplicații ce îl pot ajuta pe utilizator în organizarea călătoriilor. În cele ce urmează vor fi descrise cele mai populare dintre ele:

Google Maps este cel mai folosit serviciu de hărți și navigare la nivel global. Acest serviciu oferă atât o interfață grafică pentru utilizatori cât și o suită de API-uri puse la dispoziția dezvoltatorilor. Aplicația este complexă și permite, pe lângă generarea de trasee pentru locațiile introduse de utilizator, cât și urmărirea parcurgerii acestuia în timp real [13][9]. De asemenea, aplicația cuprinde următoarele funcționalități adiționale, o parte din ele fiind observabile și în Figura 1:

- salvarea și partajarea locațiilor: utilizatorii pot salva locațiile preferate, precum adresa de acasă sau de muncă, pentru a le putea accesa ușor și rapid și pot să partajeze locații cu alte persoane pentru a le ghid;
- explorarea și salvarea de obiective turistice, restaurante, lucruri de făcut în apropiere hoteluri etc.: utilizatorii pot să caute diverse locuri și să vizualizeze informații despre acestea, precum programul de funcționare, recenzii, detalii de contact, imagini. Există posibilitatea și de a salva locațiile în diferite categorii predefinite (Planuri de călătorie, Preferate, Vreau să merg acolo) sau de acrea o listă proprie;
- planificarea călătoriilor cu transportul în comun în funcție de orele de plecare: această funcționalitate permite utilizatorilor să vadă programul actualizat pentru diferite mijloace de transport (autobuz, tren, metrou), precum și traseul acestuia;
- imagini din satelit, perspective cu vedere 360°, vederea în 3D a marilor orașe, fotografii cu străzile: utilizatorul are acces la diferite moduri de vizualizare a hărților, străzilor pentru a identifica mai ușor locația, poziția unde se află, sau unde trebuie sa ajună;
- descărcarea hărților pentru utilizarea offline: în cazul în care nu există conexiune la internet, utilizatorii pot încă să vizualizeze hărțile dacă le-au descărcat.

Printre cele mai importante tehnologii folosite în dezvoltarea acestei platforme se menționează: Cloud Computing pentru a stoca și procesa multitudinea de date de care Google are nevoie, datele fiind obținute din imagini provenite de la satelit sau cu ajutorul unor camere capabile să capteze imagini panoramice de 360°. De asemenea, utilizează algoritmi de învățare profundă și inteligența artificiala pentru sugerarea de rute, prezicerea traficului etc. [9].

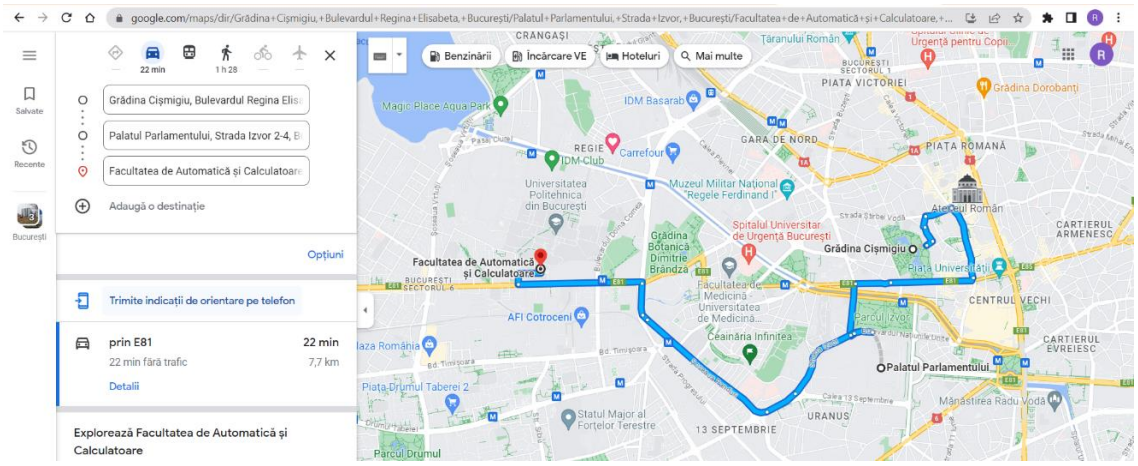


Figura 1: Interfața grafică Google Maps [13]

Sygic Maps este o aplicație de navigație similară cu Google Maps, având multiple funcționalități care să îl ajute pe utilizator să își planifice vacanța. Această aplicație, spre deosebire de cea descrisă anterior, nu se limitează la un număr de 10 locații pentru generarea traseului, utilizatorul putând introduce mai multe puncte de vizitare, să le aranjeze în funcție de preferințe și să salveze rutele personalizate pentru a le putea reutiliza sau pentru a le accesa rapid atunci când are nevoie de ele. Traseele, de asemenea, pot fi urmărite în timp real, deplasările fiind făcute cu mașina sau la pas, neavând funcționalitatea de a urmări și rutele pentru transportul public [24].

O altă aplicație este **TripIt**, un organizator digital al călătoriilor, aceasta înglobează toate informațiile necesare pentru a merge într-o vacanță: transport, cazare, activități, program și rute. Pe baza acestor detalii oferite se creează itinerariul organizat după preferințele utilizatorului, ce poate fi salvat, partajat și utilizat în condiții în care accesul la Internet nu este disponibil. De asemenea, există opțiunea de scanare a e-mail-urilor, ce caută informațiile relevante scopului, cum ar fi rezervările de avion sau hotel, și care le adaugă automat în itinerariu, fără a mai fi nevoie de introducerea manuală a acestor date de către persoana care dorește beneficieze de acest serviciu, acesta poate să fie notificat prin e-amil în cazul în care una dintre rezervări este anulată și să i se sugereze alte variante alternative [26].

TripAdvisor este una din cele mai cunoscute și folosite platforme de către turiști, fiind o sursă de inspirație pentru a decide cum să planifici o călătorie. Utilizatorii au acces la recenziile și opiniile altor persoane ce au vizitat diferite locații sau au încercat diverse experiențe, iar astfel pot să selecteze lucrurile mai importante și care merită a fi văzute din destinația aleasă. Alte funcționalități ale acestei platforme sunt: realizarea de rezervări pentru hoteluri, zboruri de avion și restaurante direct din aplicație și filtrarea cazărilor după preț, utilizatorii putând compara multiple prețuri pentru a găsi hotelul cel mai avantajos pentru bugetul pe care și-l permite [25].

Similar cu **TripAdvisor** este aplicația **GetYourGuide**, aceasta concentrându-se pe partea de rezervarea de tururi, atracții turistice și experiențe. Prin intermediul acestei platforme, călătorii îți

pot planifica și rezerva experiențe, de unii singuri, înainte ca vacanța să aibă loc sau chiar în timpul călătoriei, în timp real.

Aplicațiile prezentate anterior au capacitatea de a sugera o rută de parcurgere doar pentru o selecție de obiective turistice a căror ordine a fost deja stabilită. În Tabel 1 s-a făcut o scurtă comparație a acestora, luând în considerație doar funcționalitățile definitorii ce includ obiectivele turistice.

	Google Maps	Sygic Maps	TripIt	TripAdvisor
Servicii gratuite	X		X	X
Sugestii de obiective turistice	X	X		X
Planificarea itinerariu		X	X	
Recenzii despre obiectivele turistice	X	X		X
Generare de rute	X	X	X	
Generarea de rute optime ce nu necesită intervenția umană				

Tabel 1: Compararea funcționalităților aplicațiilor existente pe piață

2.3. Cerințe funcționale și non-funcționale ale aplicației

Pentru îmbunătățirea experienței utilizatorului s-au dezvoltat două aplicații: cea de bază și cea internă, accesul făcându-se pe bază de rol: utilizator simplu, membru din echipa de administrație a aplicației și din echipa de cercetare a pieței. Aplicația de bază este destinată utilizatorilor care au rolul de utilizator simplu pentru a putea accesa funcționalitatea de generare de sugestii de itinerarii, iar cea internă este destinată membrilor din echipa de administrație și cea de cercetare a pieței pentru a gestiona, monitoriza și actualiza informațiile despre obiectivele turistice, utilizatori și recenzii, cât și de a vizualiza statistici legate de acestea.

În scopul rezolvării problemei, următoarele funcționalități și caracteristici sunt necesare a fi dezvoltate:

- autentificarea și înregistrarea utilizatorilor
- itinerariul generat trebuie să ia în calcul doar rutele în ceea ce privește deplasarea cu mașina, acest lucru realizându-se prin implementarea unui algoritm de sugestie ce folosește algoritmul de tip Greedy
- itinerariul afișat trebuie să conțină: perioada de timp a călătoriei setate de turist, numele obiectivelor turistice, durata de vizitare a obiectivelor, ora aproximativă la care turistul ajunge la obiectivul turistic
- programul de deschidere și închidere este specific pentru fiecare obiectiv turistic, nu toate obiectivele au același program

- orice utilizator înregistrat în sistem trebuie să aibă acces la funcționalitățile aplicației, în funcție de rolul utilizatorilor:

Pentru rolul de utilizator, aplicația va permite:
- vizualizarea de obiective turistice, și căutarea acestora
- vizualizarea de statistici a celor mai populare destinații și a celor mai vizitate atracții turistice dintr-un oraș, țară având în vedere recenziile utilizatorilor
- generarea de itinerarii personalizate și salvarea acestora în aplicație și în format .pdf
- vizualizarea hărții cu traseul trasat pentru fiecare zi din itinerariu
- scrierea de recenzii atât pentru obiective turistice cât și pentru aplicație

Pentru rolul de membru din echipa de administrație, și cel de cercetare a pieței și colectare de date, aplicația va permite:
- accesul doar pentru funcționalitățile interne, de analiză și cercetare, nu și accesarea funcționalității de planificare a călătoriilor

Pentru rolul de membru din echipa de cercetare și colectare de date, aplicația va permite:
- adăugarea, ștergerea și editarea unui obiectiv turistic
- filtrarea obiectivelor turistice după anumite criterii: oraș, țară, categorie etc. și gruparea acestora
- descărcarea de rapoarte
- vizualizarea de statistici a celor mai populare destinații și a celor mai vizitate atracții turistice dintr-un oraș, țară având în vedere recenziile utilizatorilor
- centralizarea recenziilor pentru fiecare obiectiv turistic

Pentru rolul de membru din echipa de administrație, aplicația va permite:
- vizualizarea de statistici pentru urmărirea progresului de dezvoltare a aplicației, în funcție de recenziile primite de la utilizatori
- centralizarea utilizatorilor înregistrați în aplicație
- filtrarea și gruparea utilizatorilor
- centralizarea recenziilor referitoare la utilizarea și la funcționalitățile aplicației

Cerințele non-funcționale ce trebuie îndeplinite de aplicație sunt:

- meniul de navigare trebuie să fie mereu vizibil și accesibil indiferent de rolul utilizatorului
- utilizarea limbii engleze ca fiind modul implicit de traducere al interfeței
- timpul maxim de generare a itinerariului să nu depășească un minut, acesta fiind stabilit de numărul obiectivelor turistice ale destinației pentru care se sugerează itinerariul. În cazul în care timpul de așteptare este depășit se va afișa un mesaj specific și se va reîncerca să se genereze un itinerariu
- în urma salvării itinerariului, utilizatorul îl poate vizualiza instant în aplicație

- în urma operațiilor de adăugare, editare și ștergere de obiective turistice, recenzii și utilizatori, modificările se vor putea observa instant atât în cadrul aplicației interne, cât și în cea de bază
- în momentul în care itinerariul va fi disponibil: se va evidenția obiectivele turistice în funcție de scorul acestora:
    - verde pentru atracțiile cu scorul mai mare decât 4.50
    - mov închis pentru atracțiile cu scorul mai mare de 3.50
    - mov închis pentru restul atracțiilor

Capitolul 3

Stadiul actual in domeniu si selectarea soluției tehnice

Acest capitol este esențial pentru scopul realizării lucrării, deoarece sunt prezentate tehnologiile ce au fost selectate pentru a rezolva problema studiată, ținând cont de tendințele dominante actuale și de caracteristicile specifice ale acestora. De asemenea, pune bazele pentru etapele ulterioare de proiectare și implementare.

3.1. Stadiul actual al tehnologiilor utilizate pentru dezvoltarea soluției

În urma unor cercetări și comparații a tehnologiilor existente s-a decis să se utilizeze următoarele pentru dezvoltarea aplicației:

3.1.1. Tehnologii Java

Java este un limbaj de programare ce se bazează pe principiile de orientare de obiecte, folosit adesea pentru dezvoltarea de aplicații deoarece permite crearea paginilor web dinamice cu interfețe primitoare și interactive pentru utilizator.

Spring Framework

Așa cum este prezentat în [23], Spring este o platformă Java cu licență liberă și cel mai popular framework de dezvoltare de aplicații Java Enterprise Edition (JEE) de înaltă performanță. Un framework reprezintă o colecție de biblioteci, de scripturi predefinite pentru a ușura dezvoltarea de aplicații și programe software. Spring structurează conexiunile între programe și entități, facilitând munca dezvoltatorului prin simplificarea proceselor complexe. Conceptele fundamentale ale acestui framework sunt injectarea de dependențe și inversarea controlului.

Inversarea controlului este utilizată pentru a împărți funcționalitatea programului în module independente și a le face extensibile. Prin injectarea de dependențe se poate implementa

inversarea controlului, în loc ca obiectul să-și creeze propriile dependențe, dependințele sunt injectate în obiect în momentul creării acestuia.

Spring Boot

Spring Boot reprezintă un proiect bazat pe Spring Framework prin care se simplifică procesul de configurare și lansare a aplicațiilor Spring, de asemenea este foarte popular în ceea ce privește dezvoltarea de API-uri REST [21]. Cele mai importante concepte sunt:

- autoconfigurarea: clasele sunt configurate în funcție de cerințele și dependențele necesare proiectului
- autonomia: nu este nevoie de o configurație foarte complicată pentru crearea unei aplicații
- server încorporat: nu este necesară folosirea altei aplicații pentru a se ocupa de pornirea aplicație pe un server. Serverul Tomcat este un exemplu de server încorporat din Spring Boot, folosit și în cadrul aplicației prezentate în această lucrare
- conceptul de „starter”: existența unor pachete ce înglobează mai multe dependențe
- Spring Initializr: un instrument ce generează structura de proiect Spring Boot, având dependențele necesare deja incluse

Java Servlet API, REST API / RESTful API

Java Servlet API conține 2 pachete, javax.servlet și javax.servlet.http, ce asigură multiple clase și interfețe pentru a putea realiza găzduirea de servere web, cel mai cunoscut fiind Apache Tomcat [15].

API-ul, interfața de programare a aplicațiilor, reprezintă o colecție de reguli, standarde și protocoale, folosită pentru definirea modului în care se poate realiza comunicația între dispozitive sau aplicații. REST, este un set de principii arhitecturale ce utilizează protocolul HTTP pentru a putea manipula eficient datele din cadrul unei aplicații web; acest acronim vine de la *Representational State Transfer*. Operațiile de manipulare sunt crearea, vizualizarea, actualizarea și ștergerea datelor din baza de date [20]. Cele mai importante principii sunt:

- arhitectura client-server: se referă la separarea celor două componente: interfața cu utilizatorul și baza de date. Clientul trimite cereri către server, prin protocolul HTTP, cererile sunt gestionate și procesate, iar apoi se generează un răspuns, ca de exemplu o pagină HTML, care se trimite înapoi către client, reprezentare în Figura 2
- datele care sunt frecvent utilizate sunt stocate în cashe, putând fi accesate mai rapid de către server, fără a fi nevoie ca clientul să trimită din nou cererile, astfel încât performanța rețelei se îmbunătățește
- sistem stratificat: existența mai multor straturi pentru a realiza conexiunea între client și server, reprezentat în Figura 2 prin Proxy
- cererile trebuie să conțină informații suficiente și necesare pentru a putea fi procesată de server

- partajarea resurselor dintre client și server se face prin stabilirea unor standarde care realizează conexiunea. Resursele sunt identificate prin intermediul URL-urilor, iar acestea pot fi manipulate cu ajutorul metodelor HTTP: GET, PUT, POST, DELETE

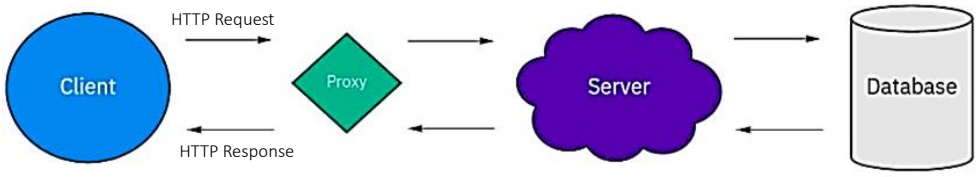


Figura 2: Comunicarea client – server [6]

3.1.2. React

JavaSript este un limbaj de programare, fiind foarte popular în dezvoltarea de aplicații web dinamice și interactive.

React este o bibliotecă JavaScript care este utilizată pentru a construi interfețe dinamice și interactive cu utilizatorul și care este folosită în dezvoltarea de aplicații web. Principalul avantaj al acesteia față de alte framework-uri îl reprezintă capacitatea de a crea componente, blocuri de cod, ce pot fi reutilizabile și mai ușor de întreținut, această abordare modulară, reușind să reducă în mod semnificativ durata de dezvoltare a aplicației. Un alt avantaj este că React generează o structură arborescentă a elementelor echivalentă cu DOM (Document Object Model), numită DOM virtual, pentru a ține evidența asupra modificărilor aduse în paginile web – când se actualizează starea unei componente, se creează o copie a DOM-ului cu acea modificare, apoi se compară cu DOM-ul real și se fac doar modificările necesare, nefiind cazul de reîncărcarea completă a acestora, având ca rezultat îmbunătățirea performanțelor în ceea ce privește viteza de încărcare și procesare a aplicației, [4] [19]. Multe aplicații cunoscute folosesc React pentru a crea interfețe intuitive și interactive pentru a îmbunătăți experiența utilizatorului, dintre care amintim Fecebook, Netflix sau Airbnb .

Bootstrap și **Material UI** sunt cele mai populare framework-uri de interfețe cu utilizatorul utilizate în dezvoltarea de aplicații web cu scopul de stiliza paginile web, aducând o experiență vizuală plăcută a utilizatorului în ceea ce privește interfața grafică. Bootstrap este un framework ce se ocupă de dezvoltarea de aplicații web interactive și oferă stiluri CSS predefinite și componente JavaScript pentru crearea rapidă de interfețe atractive și funcționale, ca de exemplu butoane, bara de navigare, bara de căutare, tabele etc. Material UI este un framework pentru React, ce se bazează pe principiile Google Material Design, un set de instrucțiuni pentru o bună proiectare a site-ului web, și care oferă o mulțime de componente React predefinite, receptive și stilizate conform regulilor de estetică Material Design: butoane, casete de dialog, formulare etc. Bootstrap, în ceea ce privește proprietatea de răspundere a unei componente, este mult mai rapid și se comportă conform funcționalității, dar din punct de vedere al aspectului, Material UI este mult mai estetic, având diferite animații și stiluri de design, [7] .

3.1.3. PostgreSQL

Multe companii, precum Uber, Skype și Instagram, folosesc în dezvoltarea de produse și soluții baza de date relațională PostgreSQL întrucât are licență liberă, suportă multiple limbaje de programare, cum ar fi Java, Python. C/C++ plus altele, și de asemenea, spre deosebire de alte sisteme de gestionare a bazelor de date relaționale, acesta poate să combine atât interogări relaționale cât și non-relaționale. PostgreSQL este folosit pentru stocarea unui volum mare de date și oferă stabilitate și performanțe ridicate deoarece are implementat un optimizator de interogări avansat și acceptă paralelism, indexare și alte caracteristici. [18].

3.1.4. Docker

Docker este o platformă cu licență liberă ce folosește containere, pachete ce conțin resursele necesare pentru ca diferite componente – biblioteci, dependențe, cod etc. - să se pot rula, pentru dezvoltarea și gestionarea aplicațiilor și bazelor de date. Docker-ul permite configurarea unui server de baze de date, astfel încât conexiunea la baza de date are loc doar dacă containerul este activ. Pentru ca acest lucru să poată să funcționeze este nevoie de o imagine Docker specifică pentru serverul de baze de date dorit a fi utilizat și de un fișier `docker-compose.yml` prin care se configurează conexiunea, specificându-se imaginea, portul folosit pentru a realiza conectarea ți alte setări relevante [12].

3.1.5. Servicii de obținere a distanțelor dintre diferite puncte definite prin coordonate geografice

Google Maps API, așa cum este prezentat și în documentația oficială [14], oferă o colecție vastă de API-uri ce pot fi folosite de dezvoltatori pentru crearea propriilor aplicații. Prin intermediul API-urilor se pot accesa datele de la anumite servicii, cum ar fi generarea de trasee între diferite locații, căutări de locații, extragerea de informații legată de o locație, accesarea hărților etc., utilizarea acestor servicii se face contra cost în funcție de numărul de cereri necesare funcționalităților din aplicație.

Pentru generarea de rute se folosește serviciul Distance Matrix API, întrucât acest API returnează în răspunsul său atât distanța dintre două sau mai multe locații, cât și durata de timp în care s-ar parcurge acea distanță. Aceste informații nu se limitează doar la un singur mijloc de transport, modul implicit fiind deplasarea cu mașina, ci în funcție de ce este nevoie se specifică în cerere mijlocul de transport dorit a fi folosit: mers, bicicletă, mașină, tramvai/metrou sau autobuz. Rezultatul primit va conține doar cele mai eficiente valori, cele mai optimizate rute.

În cerere se specifică locația de origine și locația sau locațiile destinație pentru a se putea calcula distanța și timpul dintre cele două, modalitatea de transport: driving – deplasarea cu mașina, walking – deplasarea pe jos sau transit – deplasarea cu un mijloc de transport și cheia API. Cheia se folosește pentru activarea serviciului, pentru obținerea cheii este nevoie de crearea unui cont pe Google Maps Platform, și a unui proiect pentru care se generează o chei unică.

Open Source Routing Machine, la fel ca Distance Matrix API, este un serviciu folosit pentru calcularea de rute între diverse locații geografice. Însă în comparație cu serviciul oferit de Google Maps, acesta este open-source, având atât o versiune gratuită cât și una ce necesită plătită, prima variantă fiind mul mai limitată, putând returna doar distanțele parcurse cu mașina și obține performanțe scăzute în ceea ce privește realizarea de cereri și numărul mare de locații geografice pentru care se încearcă calcularea distanțelor [17].

3.2. Algoritmi de căutare euristică

Algoritmii de căutare euristică sunt folosiți în rezolvarea problemelor reale de optimizare și în luarea de decizii, având scopul de a obține rezultate bune, rezonabile într-un timp eficient printr-o explorare a spațiului stărilor, decât să găsească soluția optimă, perfectă. Aceștia sunt utilizați atunci când problema este complexă și nu există un model matematic care poate să o rezolve în mod direct, sau modelul nu poate fi implementat din cauza complexității sale. Algoritmii euristici sunt prezenți într-o gamă vastă domenii ce se confruntă cu diverse probleme, cum ar fi cele de planificare, optimizare și de învățare profundă. Cu toate că nu oferă cele mai bune soluții, aceștia devin utili în situațiile în care resursele și timpul sunt limitate [1][10].

Problema comis-voiajorului este întâlnită în multiple aplicații din viața de zi cu zi, ca de exemplu curierat, rețele de transport public etc., fiind cea mai cunoscută problemă de optimizare, unde algoritmii euristici sunt esențiali în găsirea unei soluții. Scopul problemei este de a găsi cea mai scurtă rută posibilă astfel încât să se parcurgă toate locațiile, vizitând fiecare locație o singură dată, și opțional să se revină la locația de unde s-a pornit inițial. Complexitatea problemei crește atunci când există un număr mai mare de locații ce trebuie vizitate, dar și în cazul unui set de date mai mic găsirea unei soluții optime poate fi o sarcină dificilă, fiind considerată o problemă de tipul NP-dură.

Algoritmul Greddy reprezintă o alternativă rapidă în rezolvarea acestei probleme prin obținerea unei soluții rezonabile, apropiată de cea optimă și suficientă. Ca și principiu de funcționare, se va selecta, pornind la locația de origine, cea mai bună opțiune disponibilă, adică cea mai apropiată locație nevizitată, la fiecare pas, fără a lua în considerare și perspectivele viitoare în ceea ce privește ordinea vizitării atracțiilor. Pentru a obține rezultate optime se recomandă realizarea unei combinații între algoritmul Greddy cu alte tehnici și metode ce folosește o euristică mai complexă, [2] [3] [5].

Capitolul 4

Considerente legate de implementarea soluției tehnice

În acest capitol se detaliază aspectele tehnice ale aplicației, ce au fost folosite pentru implementarea soluției problemei, aceasta fiind analizată în capitolele anterioare. Ne propunem descrierea fiecărei etape elaborate în cadrul acestui proces, pornind de la stabilirea cerințelor necesare până la faza de testare a aplicației, conform ciclului de viață a unui sistem software.

4.1. Arhitectura aplicației

4.1.1. Diagrama de clase

Relațiile dintre entitățile sistemului sunt reprezentate în diagrama de clase, Figura 3. Un utilizator principal generează itinerariul pentru călătoria pe care urmează să o facă, acesta fiind organizat în funcție de obiectivele turistice dorite a fi vizitate, plus cele care mai intră în completarea perioadei alocate, plus se pot observa interacțiunea echipelor interne în ceea ce privește aplicația.

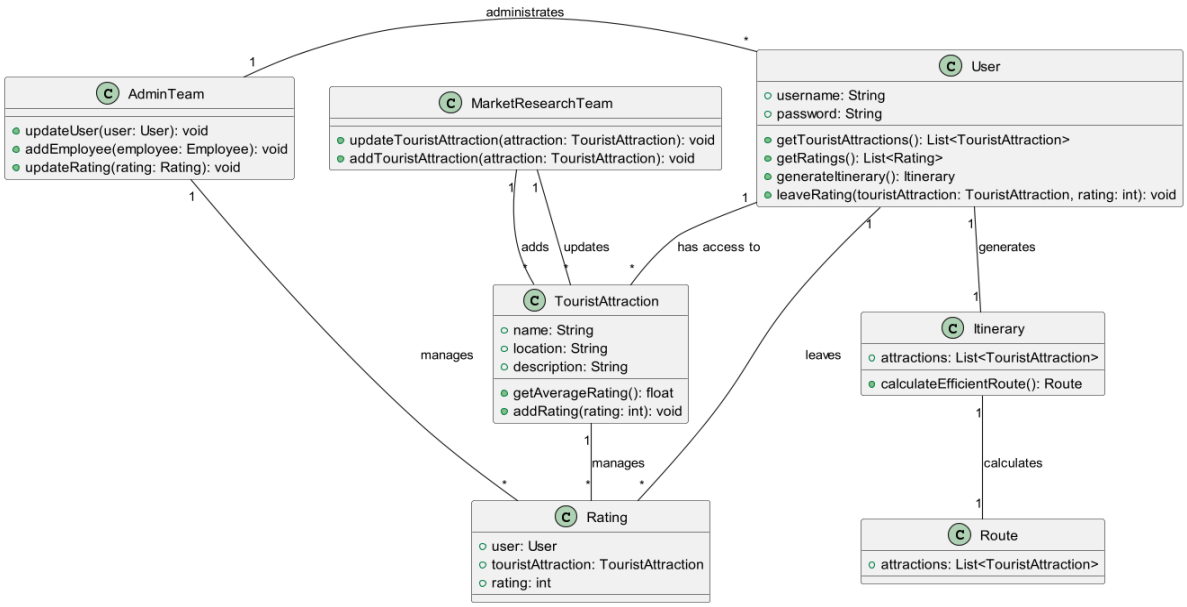


Figura 3: Diagrama de clase

4.1.2. Diagrama cazurilor de utilizare

Diagrama cazurilor de utilizare, Figura 4.1 și Figura 4.2, evidențiază funcționalitățile și comportamentul aplicației din perspectiva utilizatorilor. Utilizatorii aplicației, numiți actori, sunt persoanele care doresc să își planifice călătoria și membrii echipei care se ocupă de gestionarea obiectivelor, utilizatorilor și recenziilor.

În continuare se vor descrie scenariile aferente fiecărui caz de utilizare pentru evidențierea funcționalităților de bază ale aplicației ce au fost menționate anterior.

Primul caz este comun pentru toate tipurile de utilizator, pașii fiind aceiași în mare parte.

Actorul principal poate fi orice utilizator, indiferent de rol: *USER, RESEARCH_MEMBER, ADMIN_MEMBER.*

Cazul nr. 1 - Funcționalitatea de *Login* în aplicație

- Descriere: utilizatorul se autentifică în aplicație pe baza unui email și a unei parole pentru a fi redirecționat către pagina web principală
- Precondiții: utilizatorul trebuie să se afle pe pagina dedicată autentificării în aplicație.
- Postcondiții: utilizatorul se autentifică și este redirecționat corect către pagina principală de "Home" a aplicației
- Scenariu de bază:
    1. Utilizatorul introducere în câmpurile destinate credențialele de autentificare: email-ul și parola
    2. Utilizatorul apasă butonul *Login*
    3. Utilizatorul este redirecționat către fereastra principală
- Scenarii alternative:
    - În cazul în care utilizatorul nu este înregistrat în sistem, acesta trebuie să se înregistreze, apăsând link-ul numit *Don't have an account? Register here.*
    - Utilizatorul va fi redirecționat către pagina dedicată înregistrării în sistem; după introducerea datelor necesare în câmpurile destinate acestora și apăsarea butonului de înregistrare *Register*, va apărea o notificare în care se specifică că înregistrarea a fost făcută cu succes și utilizatorul va fi redirecționat către pagina de *Login*

    Observații:
    - toate câmpurile trebuie completate, altfel se afișează mesaje de eroare specifice
    - pentru membrii echipei interne va apărea un card, unde se va alege echipa din care face parte: de administrație sau de cercetare
- Excepții:
    - Credențialele introduse sunt invalide: email-ul și/sau parola nu sunt conform celor existente în baza de date, în acest caz se vor relua pașii din scenariu de bază

- Credențialele introduse nu sunt înregistrate în sistem, se va intra pe scenariu alternativ

Cazul nr. 2 - Funcționalitatea de *Logout*
- Descriere: deconectarea utilizatorului din aplicație
- Precondiții: utilizatorul să fie autentificat în aplicație
- Postcondiții: utilizatorul să nu mai are acces la funcționalități
- Scenariu de bază:
    1. Indiferent pe ce pagină se află utilizatorul, acesta apasă pe butonul dedicat contului de utilizator (*Profile*), iar un mic mediu cu 3 opțiuni este vizibil
    2. Utilizatorul selectează opțiunea de deconectare (*Logout*), și este redirecționat către pagina de *Login*

Cazurile descrise în continuare se aplică doar pentru persoanele care își doresc să își planifice călătoriile, Figura 4.1.

Actor principal: *USER.*

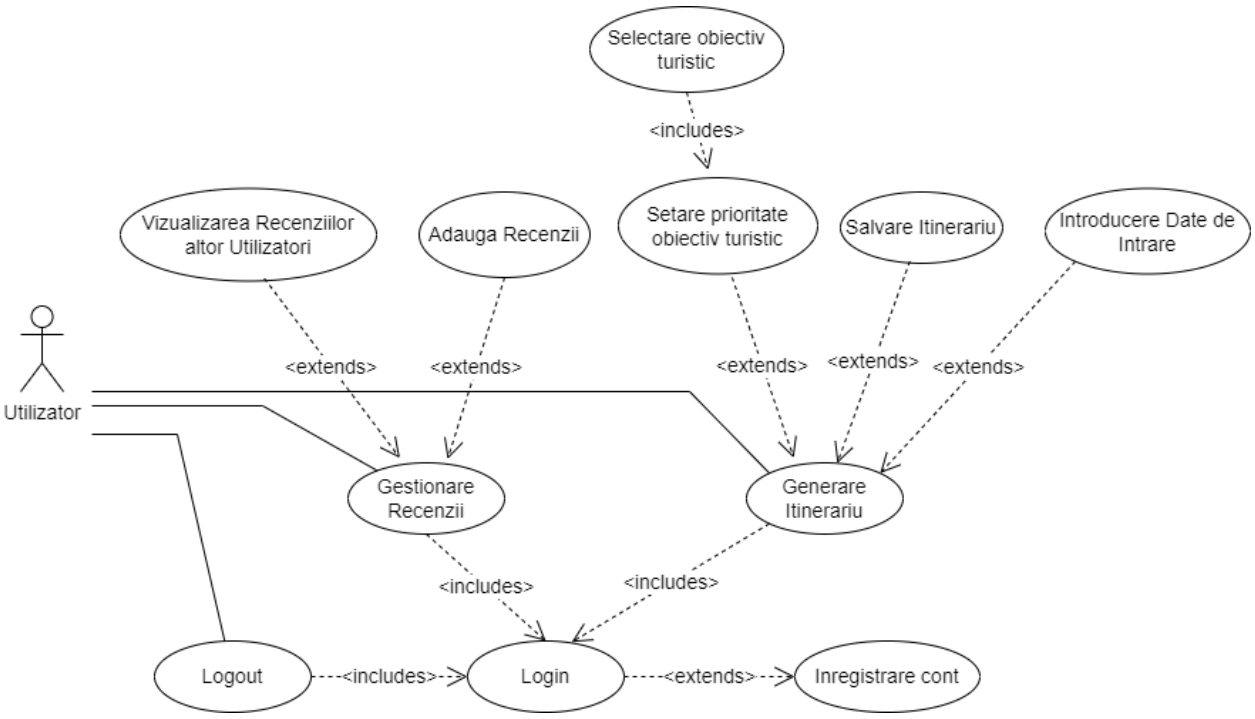


Figura 4.1: Diagrama cazurilor de utilizare pentru aplicația de bază

Cazul nr. 3 - Vizualizarea de obiective turistice și a recenziilor acestora, și adăugarea de recenzii

- Descrierea: toate obiectivele sunt listate și accesibile, putând să se vizualizeze detaliile despre ele, inclusiv recenziile altor utilizatori în ceea ce le privește. De asemenea, utilizatorul poate să lase o recenzie pentru un obiectiv turistic dacă dorește
- Precondiții: utilizatorul trebuie să fie autentificat în aplicație și să se afle pe pagina intitulată *Inspiration* a aplicației
- Postcondiții: obiectivele turistice sunt vizibile, iar la selectarea unui obiectiv se pot vizualiza detaliile și recenziile despre acesta
- Scenariu de bază:
    1. Utilizatorul navighează către pagina "Inspiration"
    2. Întreaga listă de obiective este vizibilă
    3. Utilizatorul selectează un obiectiv turistic
    4. Detaliile despre obiectiv sunt afișate pe ecran, la fel și recenziile

    5.1. Utilizatorul iese din modul de vizualizare detalii, fără a scrie o recenzie și se revine la pasul 2

    5.2.1. Utilizatorul lasă o recenzie pentru atracția turistică selectată, introducând în câmpul specific o impresie și selectând o evaluare (de la 1 la 5 stele) despre aceasta

    5.2.2. Utilizatorul apasă butonul *Submit Review,* iar o notificare cu mesajul: recenzia a fost adăugată cu succes (*Review Add Successfully*) va fi afișată pe ecran

    5.2.3. Utilizatorul își poate vedea recenzia scrisă în lista celor existente.

    5.2.4. Utilizatorul iese din modul vizualizare, și se revine la pasul 2
- Scenarii alternative:
    - Utilizatorul se răzgândește și nu mai duce la final procesul de adăugare a unei recenzii, iese din modul de vizualizare.
- Excepții:
    - Eroare de sistem: se va afișa un mesaj specific (*Something went wrong)* dacă recenzia nu a fost adăugată și se revine la etapa 5.2.1.

Cazul nr. 4 - Vizualizarea de grafice

- Descriere: graficele pentru: cele mai populare destinații, cele mai vizitate obiective turistice dintru-un oraș și cele mai vizitate obiective turistice în general sunt accesibile pentru a fi vizualizate
- Precondiții: utilizatorul trebuie să fie autentificat în aplicație și să se afle pe pagina intitulată "Inspiration" a aplicației
- Postcondiții: graficele pot fi vizualizate
- Scenariu de bază:
    1. Vizualizarea celor 3 link-uri specifice graficelor în pagina *Inspiration*: *the most popular destinations, the most popular attractions in a destination, the most popular attractions based on traveler reviews*

2. Utilizatorul apasă pe unul dintre link-urile
3. Graficul specific alegerii este deschis
4. La ieșirea din modul vizualizare grafic, se văd toate elementele din pagina Inspiration și se poate reia oricând se dorește la pasul 2

Cazul nr. 5 - Generare de itinerariu

- Descriere: itinerariul vacanței este planificat astfel încât obiectivele turistice cele mai vizitate și cele dorite a fi vizitate de utilizator sunt împărțite pe zile și ordonate pentru a obține un traseu eficient
- Precondiții: utilizatorul trebuie să fie autentificat în aplicație și să se afle pe pagina intitulată *Plan The Trip* a aplicației
- Postcondiții: itinerariul se poate salva după ce este generat
- Scenariu de bază:
    1. Utilizatorul introduce informațiile despre călătorie în câmpurile dedicate: țara, orașul, perioada alocată, ora de început și de sfârșit itinerariu ce se va lua în considerare pentru fiecare zi, și opțional locația de origine, de unde se va pleca inițial
    2. Utilizatorul apasă butonul *Next* care îl va redirecționa către pasul următor
    3. Lista de obiective turistice din orașul selectat este vizibilă
    4. Utilizatorul selectează obiectivele turistice pe care le dorește să le viziteze
    5. Utilizatorul apasă butonul *Next* care îl va redirecționa către pasul următor
    6. Itinerariul este generat, ținând cont de preferințele utilizatorului
    7. Utilizatorul apasă pe butonul *Save* pentru a salva itinerariul, iar o notificare cu un mesaj specific este afișat
- Scenarii alternative:

  Varianta 1:
    1. Utilizatorul nu selectează niciun obiectiv turistic
    2. Itinerariul este generat având în vedere cele mai populare obiective turistice din acel oraș

  Varianta 2:
    1. Utilizatorul selectează mai multe obiective turistice decât ar permite timpul pentru a le putea viziteze pe toate
    2. Itinerariul final va conține obiectivele selectate, filtrate după rating, astfel încât să se viziteze în perioada alocată doar cele mai populare obiective
- Excepții:
    - Eroare de sistem: se va afișa un mesaj specific (*Something went wrong)* dacă itinerariul nu se poate încărca și se revine la etapa 1

Cazurile descrise în continuare se aplică doar pentru membrii echipei, cei care se ocupă de gestionarea obiectivelor, a recenziilor și a utilizatorilor, Figura 4.2.

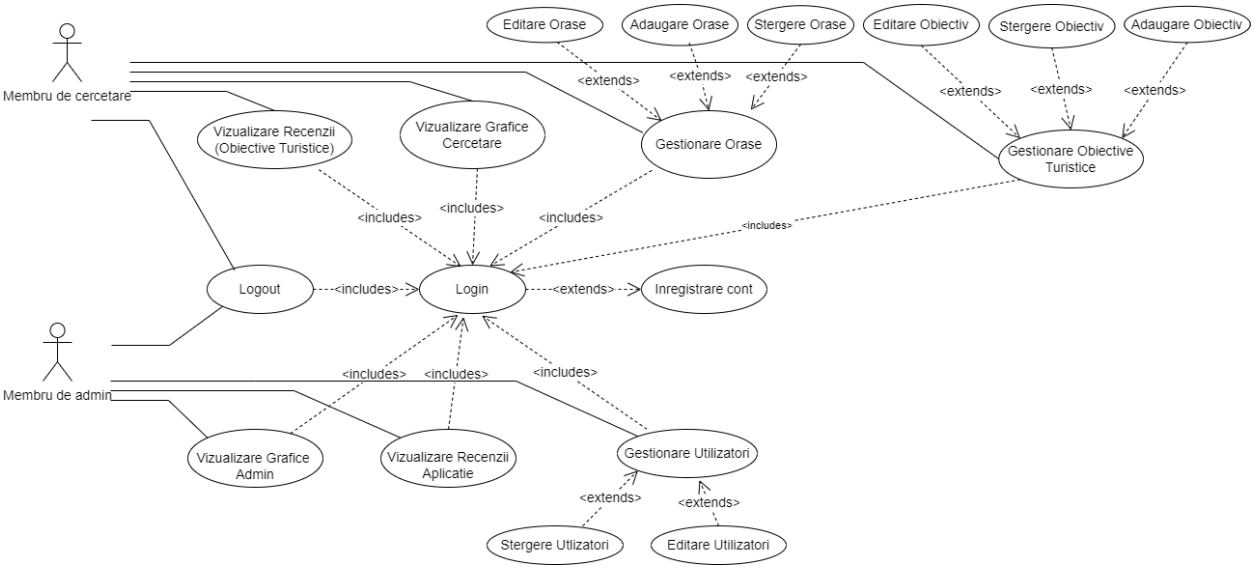


Figura 4.2: Diagrama cazurilor de utilizare pentru aplicația internă

Pentru cazurile cu numărul 6, 7 și 8 actorul principal este utilizatorul cu rolul *RESERACH_MEMBER*

Cazul nr. 6 - Gestionarea obiectivelor turistice (similar este și pentru gestionarea orașelor)

- Descriere: membrii echipe de cercetare pot să vizualizeze lista de obiective turistice, să adauge obiective noi, să actualizeze informațiile despre un obiectiv și să șteargă unul
- Precondiții: utilizatorul trebuie să fie autentificat în aplicația internă și să se afle pe pagina *Sights* aplicației
- Postcondiții: efectuarea operaților CRUD (acronim ce provine de la *Create, Read, Update, Delete*) s-a făcut cu succes, schimbările vor fi aplicate și în versiunea principală a aplicației
- Scenariu de bază:
    1. Întreaga listă de obiective turistice este afișată
        - Pentru adăugare de atracții turistice:
            2. Utilizatorul apasă pe iconița de adăugare obiectiv
            3. nouă pagină, ce conține câmpurile necesare adăugării de obiectiv, se deschide
            4. Utilizatorul completează câmpurile cu informațiile specifice: nume, adresă, descriere etc.
            5. Utilizatorul apasă pe butonul *Save* pentru a salva noul obiectiv
            6. Un mesaj de confirmare a acțiunii se afișează pe ecran
            7. Obiectivul adăugat va fi afișat în listă cu restul obiectivelor
        - Pentru actualizarea de informații:

2. Utilizatorul selectează un obiectiv și apasă pe iconița de editare
3. nouă pagină, similară cu pagina pentru adăugare de obiectiv se deschide, câmpurile fiind deja completate cu datele existente ale obiectivului selectat
4. Utilizatorul face modificările necesare, apasă pe butonul *Save*, iar un mesaj de confirmare se va afișa

- Pentru ștergerea unui obiectiv:
  2. Utilizatorul apasă pe iconița de ștergere obiectiv
  3. Un pop-up de confirmare a ștergerii obiectivului se deschide
  4. Utilizatorul apasă pe butonul de confirmă acțiunea de ștergere
  5. Obiectivul de mai este vizibil în lista de obiective din sistem

- Scenarii alternative:
  - Utilizatorul abandonează oricare dintre acțiuni, operațiunea încetează, neafectându-se datele din sistem
- Excepții:
  - Pentru adăugare de atracții turistice: există un deja un obiectiv turistic înregistrat în sistem cu același nume

Cazul nr. 7 - Vizualizarea de recenzii în ceea ce privește obiectivele turistice
- Descriere: ținerea în evidență a recenziilor utilizatorilor în legătură cu atracțiile turistice
- Precondiții: utilizatorul trebuie să fie autentificat în aplicația internă și să se afle pe pagina *Dashboard* a aplicației
- Scenariu de bază:
  - Utilizatorul are acces la toate recenziile: părerile utilizatorilor și ratingul acestora, folositor pentru generarea de grafice

Cazul nr. 8 - Vizualizare de grafice destinate membrilor din echipa de cercetare de atracții turistice (este asemănător cu cazul nr. 4, este vorba despre aceleași grafice)
- Precondiții: utilizatorul trebuie să fie autentificat în aplicația internă și să se afle pe pagina de *Analytics* la care are acces persoana care se ocupă de cercetare
- Scenariu de bază: toate graficele apar pe aceiași pagină

Pentru cazurile cu numărul 9 (Gestionarea utilizatorilor), 10 (Vizualizarea de recenzii în ceea ce privește modul de funcționare a aplicației) și 11 (Vizualizare de grafice) actorul principal este *ADMIN_MEMBER*. Acestea funcționează similar cu cazurile 6, 7 și 8 dar sunt specifice echipei administrative.

4.1.3. Diagrama secvențială

Interacțiunea dintre obiecte, din perspectiva modului de transmitere a mesajelor de tipul cerere-răspuns, este evidențiată de diagrama secvențială. Interacțiunile și schimbul de mesaje pentru a putea avea loc o interacțiune sunt prezentate într-o manieră cronologică. În Figura 5 este

prezentată o astfel de diagramă, urmărind interacțiunile dintre utilizatorul cu rolul de *USER*, care pentru prima dată are contact cu această aplicație, sistemul de autentificare și baza de date.

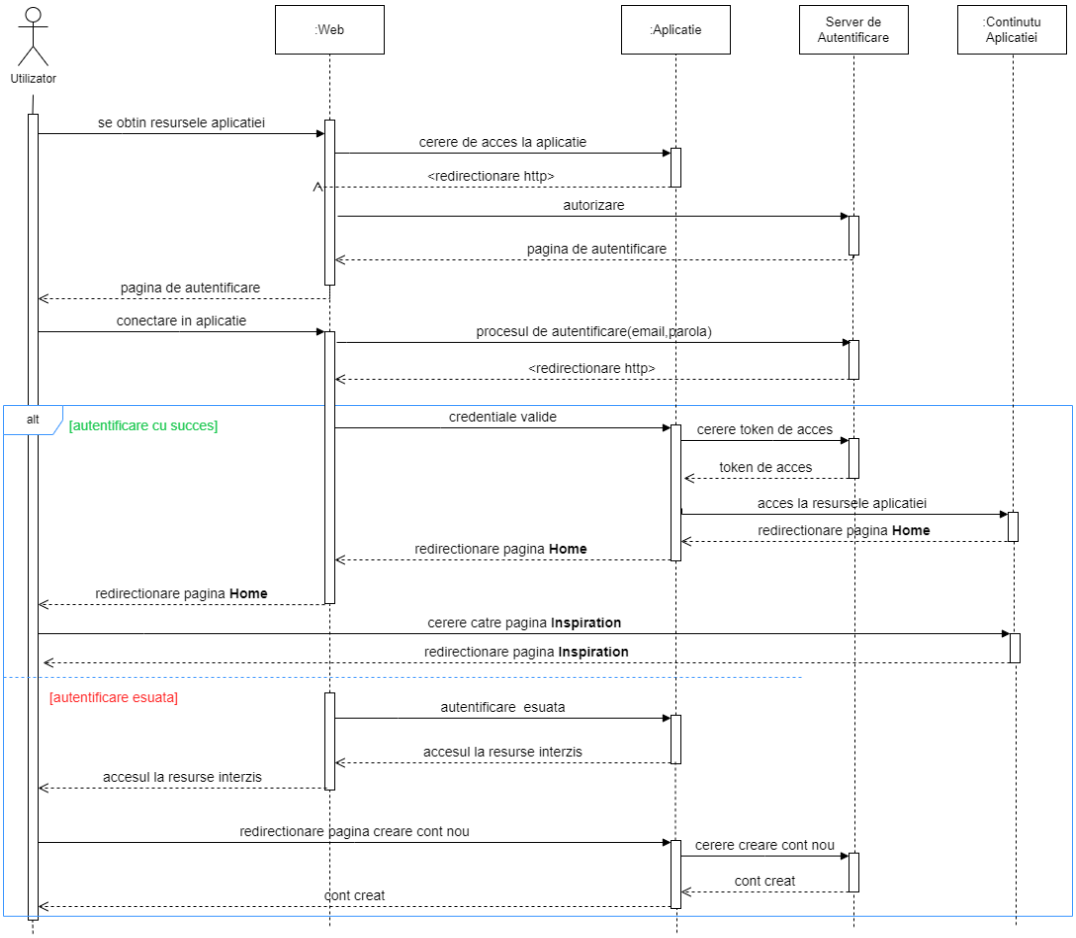


Figura 5: Diagrama secvențială

Desfășurarea operațiunii de autentificare în sistem a fost selectată pentru a fi explicată, în cazul celorlalte interacțiuni procedeul este similar.

Utilizatorul inițiază interacțiunea de a se conecta în aplicație, folosind email-ul și parola. Se trimite o cerere de autentificare către server pentru a verifica dacă credențialele sunt valide și înregistrate în sistem. Dacă autentificare a eșuat, nu se poate conecta la sistem, sistemul de autentificare trimite un răspuns prin care anunță utilizatorul că cererea nu a avut loc cu succes, afișându-i un mesaj de eroare specific prin care se sugerează crearea unui cont sau verificarea corectitudinii credențialelor. Utilizatorul alege să își creeze cont, furnizează informațiile obligatorii pentru acesta, iar o cerere este trimisă pentru a solicita creare unui cont. Aceasta este procesată cu succes și un mesaj de confirmare se întoarce către utilizator. Utilizatorul încercă din nou autentificarea, dar de data asta reușind să se conecteze și să fie redirecționat către pagina principală, cererea făcându-se cu succes. După utilizatorul decide să facă o cerere prin care să se deconecteze din aplicație, cererea este procesată și aprobată, iar ca și răspuns, utilizatorul este deconectat.

4.1.4. Structura bazei de date

Baza de date utilizată pentru stocarea datelor este una relațională, fiecare instanță a fiecărui tabel are o cheie primară doar a ei, cu ajutorul căreia se identifică când se dorește accesarea acesteia. Ca și structură, există 9 tabele, numite și entități, între care există relațiile: unu la unu (1→1), unul la mai mulți (1→∞) și mai mulți la mai muți (∞→∞), a se observa Figura 6 și Tabel 2.

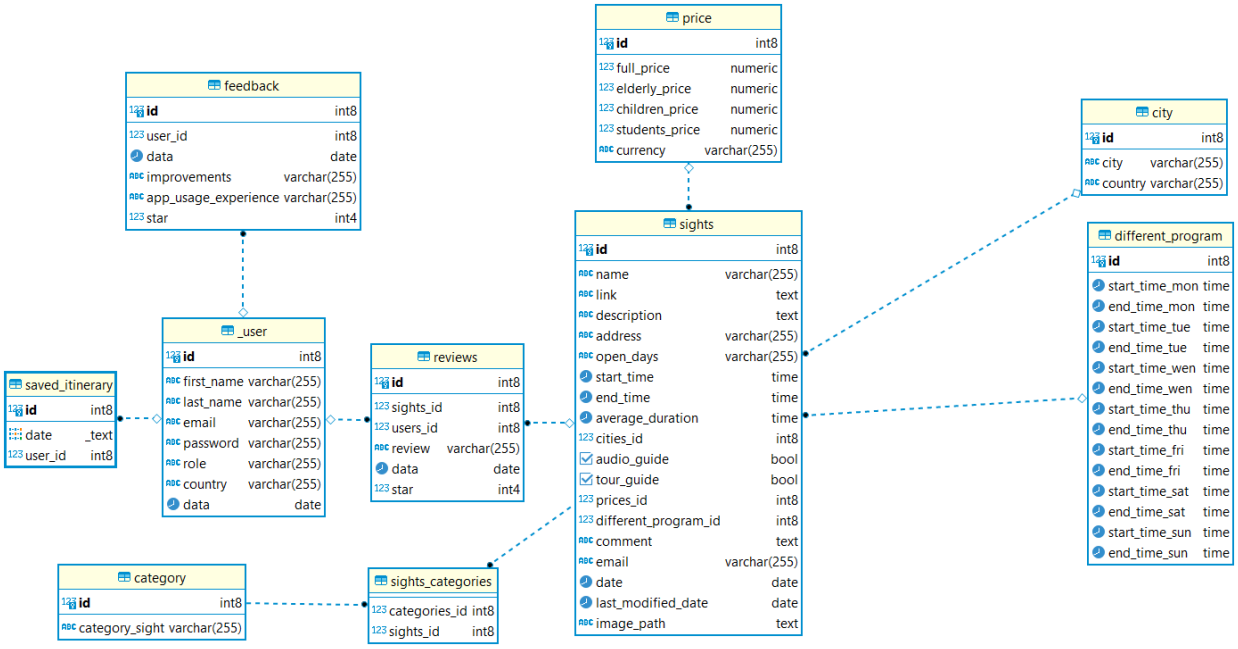


Figura 6: Structura bazei de date

Entitate	Relație	Descriere
Sights – City	1→∞	Sights = obiectivele turistice City = orașe Different_Program = programul obiectivelor turistice atunci când nu este constant Price = prețul biletelor pentru permiterea vizitării obiectivelor Review = recenziile în legătură cu obiectivele turistice User = utilizatorii, atât membrii echipei cât și persoanele care vor să își planifice călătoria Feedback = recenziile în legătură cu aplicația în sine Saved_Itinerary = salvarea itinerariilor generate
Sights – Different_Program	1→1	
Sights – Price	1→1	
Sights – Review	1→∞	
Sights – Category	∞→∞	
User – Review	1→∞	
User – Feedback	1→∞	
User – Saved_Itinerary	1→∞	

Tabel 2: Relațiile dintre entități

Acest tip de reprezentare a fost ales datorită menținerii consistenței datelor și stocării eficiente a datelor; un alt factor important fiind și asigurarea rapidității în execuția operațiilor.

Inițial, proiectul nu s-a bazat pe folosirea unui instrument comun, cum sunt bazele de date, pentru operația de strângere a datelor, s-a dorit ca procesul de colectare să se facă în timp real, în momentul când există nevoie de ele, prin extragerea de informații de pe site-ul celor de la Google Maps, utilizând instrumente specifice pentru a putea automatiza procesul și cereri API, termenul din engleză fiind cel de *web scraping*. Motivul principal care a condus la renunțarea acestei idei a fost calitatea nesatisfăcătoare a datelor. Datele obținute prin *web scraping* erau incomplete, inconsistente și chiar și duplicate, iar astfel apărea riscul folosirii unor date inexacte pentru generarea itinerariului. Un alt motiv fiind și implementare a algoritmului, neputând extrage eficient datele necesare.

4.2. Implementarea aplicației

În cadrul acestui proiect s-a dezvoltat o arhitectură bazată pe conceptul de client-server, unde partea de client reprezintă utilizatorul de servicii și este asociat cu termenul de *front-end*, iar partea de server este furnizorul de servicii, asociat cu termenul de *back-end*. Între cele două componente există o legătură strânsă, iar astfel experiența de utilizare devine coerentă și se realizează fără probleme. Interacțiunea dintre *back-end* și *front-end* se realizează prin intermediul API-urilor ce permit schimbul de date dintre client și server.

În cele ce urmează, se va detalia fiecare componentă, ce tehnologii, componente au fost folosite și cum au fost implementate pentru dezvoltarea aplicației.

Componenta server

Back-end-ul este cel care furnizează datele și logica esențială, fiind responsabil de procesarea cererilor primite de la client. Limbajul de programare folosit în dezvoltarea aplicației este Java împreună cu framework-ul Spring, ce permite gestionarea și dezvoltarea de API-uri și de servicii, precum sunt cele de gestionare a entităților și de autentificare a utilizatorilor.

Structura proiectului poate fi observată în Figura 7, care sunt clasele și cum sunt ele organizate în pachete în funcție de scopul acestora pentru dezvoltarea aplicației.

Fișierul *docker-compose.yml* este utilizat pentru definirea și configurarea unei infrastructuri de tip container asociată bazei de date. În acest fișier s-a specificat serviciul de bază de date folosit, acesta fiind PostgreSQL, volumul de date, porturile folosite și alte componente relevante. Containerul Docker, creat pentru baza de date, trebuie să fie pornit de fiecare dată când se dorește activarea acesteia pentru a o putea accesa prin intermediul DBeaver, prin care se poate interacționa mult mai mai ușor cu serviciul de baze de date dorit.

În ceea ce privește conectarea aplicației la baza de date, s-a utilizat dependința DataSource din Spring Boot. Pentru aceasta s-a configurat fișierul application.properties cu următoarele proprietăți: url-ul bazei de date, numele și parola utilizatorului cu rolul de administrator al bazei de date și driver-ul specific tipului de bază de date pentru a se putea realiza conexiunea cu baza de date.

Crearea și gestionarea entităților s-a realizat prin intermediul standardului JPA, acronimul provenind de la *Java Persistence API*, ce facilitează interacțiunea dintre aplicația Java și baza de date, separând astfel logica de manipulare a datelor și limbajul SQL specific tipului de bază de

date. Cu ajutorul adnotărilor speciale, „@Entities”, „@Table”, clasele din **pachetul entieies** sunt mapate direct la tabelele din baza de date, denumită *travelDB*. Fiecare entitate conține câmpuri, evidențiate prin adnotările „@Id”, „@Column”, ce corespund coloanelor din tabela asociată, de exemplu entitatea City are câmpurile: id, city, country. Adnotările sunt utilizate pentru ca *framework*-ul Spring să poată recunoaște clasele, metodele și variabilele ce le folosesc pentru a căpăta diverse caracteristici ce sunt necesare implementării unor funcționalități mai complicate și să le execute atunci când este nevoie.

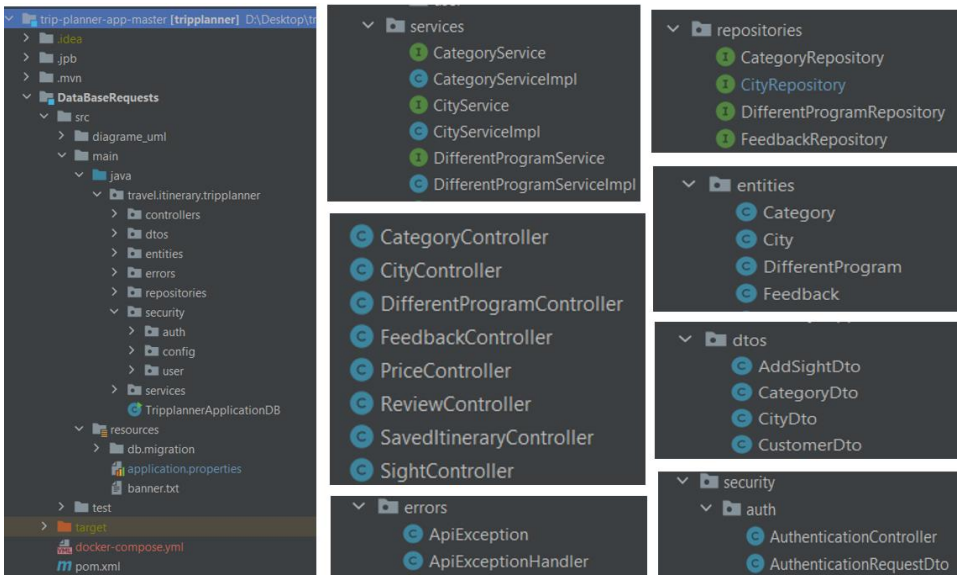


Figura 7: Structura proiectului

Pe parcursul dezvoltării aplicației, baza de date a suferit multiple schimbări până la varianta finală: adăugare, modificare, ștergere de tabele și coloane. Pentru ca consistența și integritatea schemei bazei de date să fie menținută de-a lungul ciclului de viață a aplicație, s-a folosit Flyway, o unealtă open-source de gestionare a migrărilor bazei de date. Astfel fiecare modificare făcută asupra unei clase Java sau direct bazei de date este actualizată în mod automat și controlat în ambele sensuri, de la aplicație la baza de date, sau invers. Fiecare migrare se identifică printr-un număr unic de versiune și include diverse instrucțiuni SQL ce reprezintă modificările ce au fost făcute. La fiecare rulare a programului se aplică automat scripturile de migrare în ordinea în care au fost create și se verifică dacă coincid cu starea curentă a bazei de date, în cazul în care există discrepanțe între acestea, Flyway le detectează și informează utilizatorul prin intermediul unor mesaje de eroare specifice că se necesită o ajustare a actualizare a scripturilor pentru a remedia diferențele. În Figura 8 se poate observa schema de funcționare a celor prezentate anterior.

Gestionarea bazei de date și a operațiilor de manipulare a acesteia se realizează prin implementarea mai multor componente ce au un rol bine definit, acestea fiind: *controller, service, repository*.

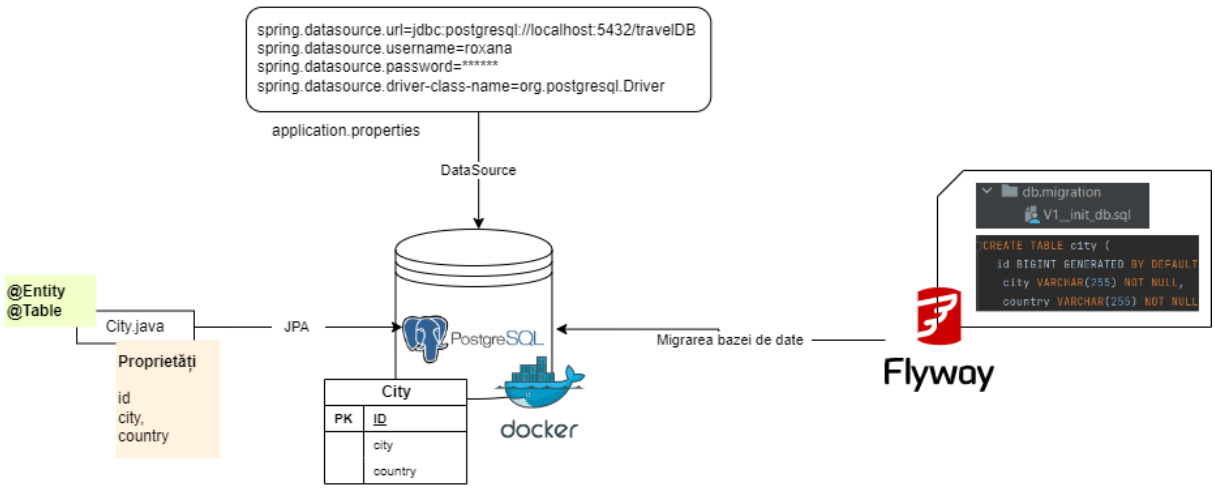


Figura 8: Schema de interacțiune dintre back-end și baza de date [22]

Pachetul controllers cuprinde clasele ce au rolul de a gestiona cererile HTTP în interacțiunea cu utilizatorul. Acestea sunt adnotate cu „@RestController” și funcționează în modul următor: clientul inițiază o cerere, după primirea cererii HTTP se validează datele de intrare și apoi se apelează metodele din clasele de Service prin care se execută operațiile solicitate de client, cele de citire, creare, editare și ștergere, și întoarce un răspuns cu rezultatele obținute către acesta. Un exemplu de astfel clasă se poate vedea în Figura 9. Adnotările „@RequestMapping” și „@GetMapping” sunt utilizate pentru a mapa ruta URL către metoda respectivă, există o astfel de adnotare pentru fiecare metodă HTTP.

Pachetul dtos conține clasele de obiecte ce sunt folosite pentru transferul de date între diferite componente ale aplicației. Acestea definesc structura datelor, doar atributele ce sunt necesare pentru efectuarea operației de manipulare, ce trebuie trimisă între client și server, fiind un mod eficient de a realiza acest lucru, spre deosebire de a transmite întreg obiectul, ceea ce este mai costisitor. DTO este un acronim ce provine de la termenul din engleză *Data Transfer Object.*

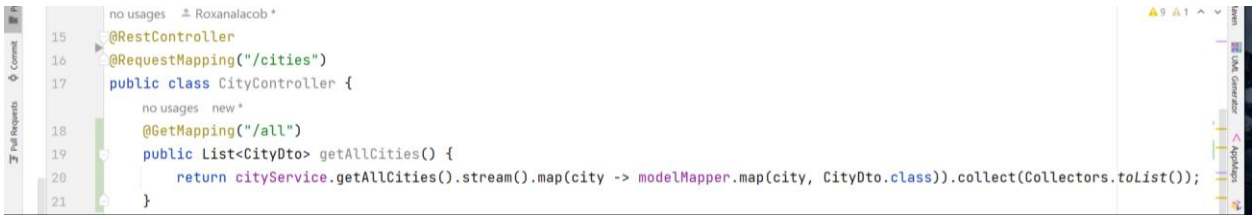


Figura 9: Clasa controller

Pachetul repositories conține doar interfețe, fiecare extinde interfața JPA Repository din *framework*-ul de Spring Boot și sunt utilizate pentru efectuarea de operații de manipulare a datelor și interacționarea cu baza de date. Un exemplu fiind ilustrat în Figura 10.

```
7 usages  Roxanalacob
public interface CityRepository extends JpaRepository<City, Long> {
    1 usage  Roxanalacob
    @Query("SELECT COUNT(DISTINCT c.country) FROM City c")
    long countDistinctCountry();
}
```

Figura 10: Interfața de repository

În **pachetul services** se găsesc clasele cu adnotarea „@Service” responsabile de implementarea logicii necesare pentru manipularea și gestionarea cererilor primite de la clasele de Controller, prin utilizarea de repository-uri și interfețe, Figura 11.

```
no usages  Roxanalacob *
12  @Service
13  public class CityServiceImpl implements  CityService {
        8 usages
14      private final CityRepository cityRepository;
        1 usage  new *
15      @Override
16      public List<City> getAllCities() { return cityRepository.findAll(); }
```

Figura 11: Clasa service

Acest mod de a grupa clasele ajută la separarea responsabilității și la modularizarea codului și este abordare bine definită pentru crearea unei arhitecturi scalabile și ușor de întreținut. În Figura 12 este prezentă interacțiunea dintre componentele menționate anterior.

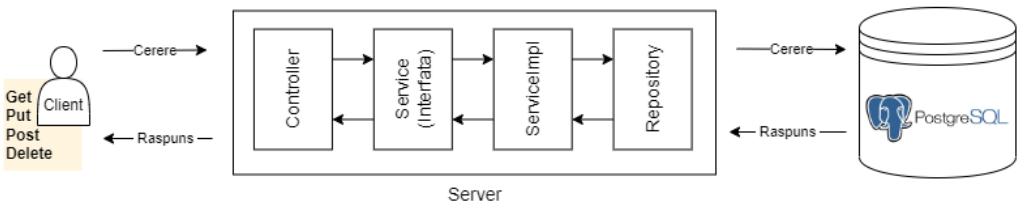


Figura 12: Interacțiunea dintre componentele: controller, repository, service

De menționat este că serverul Apache Tomcat se pornește automat la execuția programului, și este configurat pe portul 8080.

În ceea ce privește procesul de autentificare al utilizatorului, există 3 pachete destinate acestui lucru, toate fiind grupate într-unul singur numit **pachetul security**. În **pachetul user** se găsesc toate componentele de mai sus corespunzătoare utilizatorului, la fel și în **pachetul auth**, dar specific pentru partea de manipulare și de răspuns pentru autentificare. **Pachetul config** conține clasele ce se ocupă de validarea credențialelor utilizatorului și a token-ului, securitate, dacă se poate face autentificarea sau nu.

Procesul de autentificare începe atunci clientul trimite o cerere HTTP către server, aceasta este interceptat de *JwtAuthFilter*, filtru de autentificare JWT, fiind primul lucru care se execută, ce are rolul de a verifica și valida token-ul JWT, executându-se o singură dată pe cerere. Etapele procesului de validare sunt următoarele: filtrul efectuează un apel folosind metoda

userDetailsService din clasa *ApplicationConfig* prin care se încearcă să se obțină detaliile utilizatorului din token, cum ar fi e-mail-ul, folosit pentru a verifica și autoriza autentificarea acestuia în aplicație. Dacă e-mail-ul nu poate să fie extras, atunci se trimite un răspuns cu statusul 403 către client, care înseamnă accesul interzis. În caz contrar, e-mail-ul va fi folosit pentru a identifica și prelua informațiile despre utilizatorul respectiv din baza de date, unde în cazul în care nu există se va trimite un răspuns cu 403, iar dacă există se încearcă autentificarea acestuia.

Mecanismul de validare de token încearcă să apeleze un serviciu, clasa JwtService, ce folosește ca și parametrii: utilizatorul și token-ul, și unde verifică dacă token-ul este expirat sau că este pentru utilizatorul specificat. În aceste cazuri se va returna un răspuns 403, altfel token-ul este valid și se poate trece mai departe.

După preluarea detaliilor despre utilizator și validarea token-ului, acesta s-a autentificat cu succes și se va putea apela sau actualiza contextul de securitate curent din aplicație cu detaliile și privilegiile acestuia, prin intermediul clasei *SecurityContextHolder* din cadrul framework-ului de Sping Security. De fiecare dată când se verifică dacă utilizatorul este autentificat și răspunsul pentru această solicitare este pozitiv, contextul de securitate este actualizat, iar cererea va fi transmisă automat către DispatcherServlet, ce o va direcționa direct către componentele *controller* din cadru aplicației Spring. De la controller un răspuns cu statusul 200 va fi returnat către client. În Figura 13 este reprezentată o schemă ce ilustrează etapele pentru autentificarea utilizatorilor.

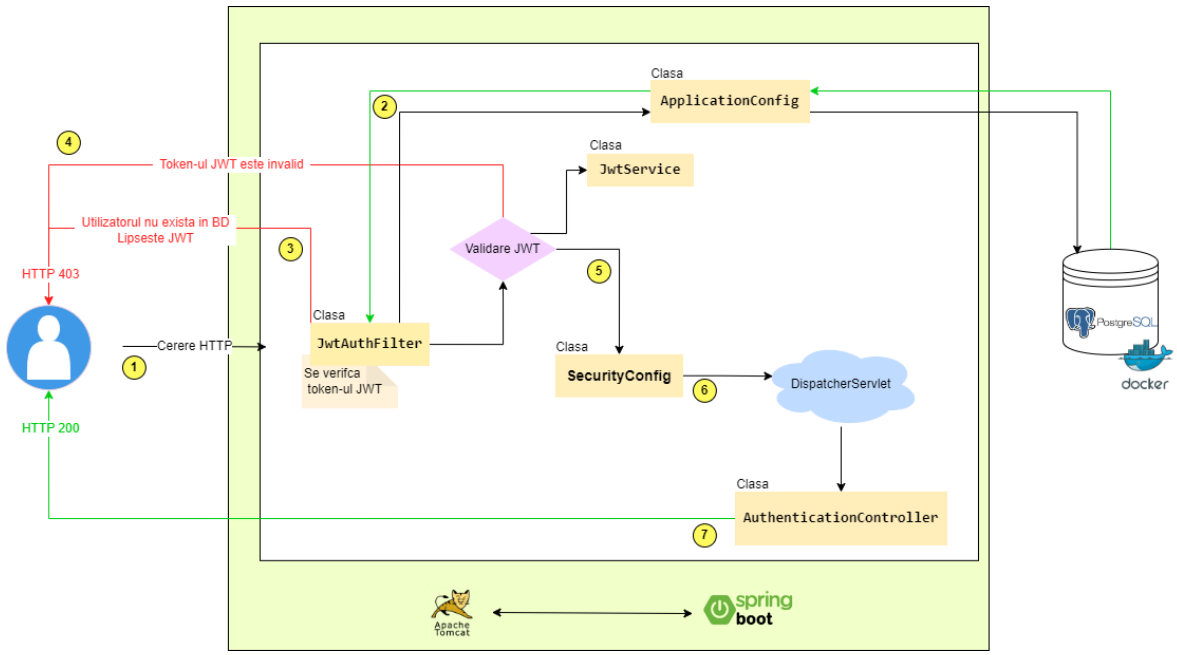


Figura 13: Schema de funcționare a procesului de autentificare [16]

Componenta client

Scopul acestei componente este de a crea o experiență completă pentru utilizator, fiind partea interactivă, dinamică și vizibilă a aplicației, ceea ce îl convinge pe client să o utilizeze.

Tehnologii precum JavaScript, HTML și CSS s-au folosit pentru dezvoltarea *front-end*-ului, împreună cu librăria React ce folosește și framework-urile Bootstrap și Material UI pentru a crea și stiliza diverse componente cu scopul de a face interfața cu utilizatorul mult mai plăcută și ușor de utilizat.

Interfața grafică a fost concepută astfel încât să se urmărească cu atenție principiile de design UX/UI.

- utilizarea unui design cât mai simplu, minimalist și clar cu spațiu între elemente pentru a facilita navigarea
- meniurile și butoanele sunt plasate în locuri ușor de accesat și intuitive pentru ca utilizatorului să nu îi fie greu să le găsească
- folosirea acelorași stiluri, culori și modele de interacțiune în toată aplicația, menținând astfel consistența în aspectul și comportamentul elementelor din interfață

Comunicarea cu serverul aplicației, componenta de *back-end*, se realizează prin intermediul librăriei JavaScript axios, permițând gestionarea cererilor, răspunsurilor și a excepțiilor. Acesta oferă suport pentru toate metodele HTTP, un exemplu de utilizare al unei astfel de cereri este prezentat în Figura 14.

```
42        try {
43          const config = {
44            headers: {
45              Authorization: `Bearer ${token}`
46            }
47          };
48          const response = await axios.get(`http://localhost:8080/users/id/${userId}`, config);
49          setUserData(response.data);
50          console.log(response.data);
51        } catch (error) {
52          console.log(error);
53        }
```

Figura 14: Utilizarea librăriei Javascript: axios

Algoritmul de generare de itinerarii

Acest algoritm JavaScript este folosit pentru generarea itinerariului bazat pe atracțiile turistice disponibile în perioada alocată vacanței, distanța dintre ele, durata de vizitare și ratingul pentru fiecare obiectiv turistic.

Algoritmul conține 3 componente:

Datele de intrare:

- `attraction`: o listă de atracții turistice, în care sunt stocate informațiile despre obiectivele turistice: numele atracției, ratingul, durata de vizitare, orarul de funcționare, coordonatele de localizare
- `distanceMatrix` și `durationMatrix`: în aceste matrici se salvează distanțele dintre fiecare atracție și timpul de deplasare de la o atracție la alta
- `startTime`, `endTime`, `firstDay`, `endDay`: variabile folosite pentru determinarea perioadei de timp alocată vacanței și intervalul de timp ce limitează numărul de obiective vizitate într-o zi

- `startingAttraction`: reprezintă indexul atracției de la care se pleacă

Algoritmul principal: pentru planificarea itinerariului se utilizează o abordare bazată pe o euristică de tip Greedy, cea mai bună alegere la fiecare pas. Acesta funcționează în următorul mod:

- se parcurg zilele disponibile
- pentru fiecare zi se alege prima atracție pe baza disponibilității și a orarului de funcționare
- apoi se caută următoarea atracție care să fie vizitată în aceeași zi, folosind ratingul, distanța și timpul de vizitare; se alege următoarea atracția cu cel mai mare rating care poate fi vizitată în timpul rămas, Figura 15
- a est proces se repetă pentru fiecare zi până când nu mai există suficient timp pentru a vizita alte atracții sau toate locațiile au fost vizitate

```
289  for (let i = 0; i < attractions.length; i++) {
290    if (visited[i]) continue; // Skip visited attractions
291    let distance = distanceMatrix[currentAttraction][i]; // Distance to next attraction
292    let duration = durationMatrix[currentAttraction][i]; // Duration to next attraction
293    let visitTime = attractions[i][2]; // Visit time of the next attraction
294
295    // Check if we have enough time to travel to, visit the attraction, and return
296    if (availableTime >= distance + duration + visitTime) {
297      let score = attractions[i][1] / (distance + duration + visitTime); // Calculate score
298      let openingTime = attractions[i][3][currentDate.getDay()][0];
299      let closingTime = attractions[i][3][currentDate.getDay()][1];
300
301      // Check if the attraction is closed on the current day
302      if (openingTime === null || closingTime === null) {
303        continue; // Skip the attraction and continue to the next one
304      }
305
306      // Check if the attraction is open during the available time
307      if (toMinutes(openingTime) <= toMinutes(startTime) && toMinutes(closingTime) >= toMinutes(endTime)) {
308        // The attraction is open for the entire available time, so it's a valid option
309        if (score > maxScore) {
310          // Update next attraction and max score
311          nextAttraction = i;
312          maxScore = score;
313        }
314      } else {
315        // Calculate the time range in which we can visit the attraction
316        let validStartTime = toMinutes(startTime) > toMinutes(openingTime) ? toMinutes(startTime) : toMinutes(openi
317        let validEndTime = toMinutes(endTime) < toMinutes(closingTime) ? toMinutes(endTime) : toMinutes(closingTime
318
319        // Calculate the duration within the valid time range
320        let validDuration = validEndTime - validStartTime;
321
322        // Check if we have enough time to visit the attraction within the valid time range
323        if (validDuration >= visitTime) {
324          // Calculate the score based on the duration within the valid time range
325          score = attractions[i][1] / (distance + duration + validDuration);
326
327          if (score > maxScore) {
328            // Update next attraction and max score
329            nextAttraction = i;
330            maxScore = score;
331          }
332        }
333      }
334    }
```

Figura 15: Secvență de cod din algoritmul de generare a itinerariului pentru verificarea constrângerilor

Metodele auxiliare, folosite pentru a completa codul:

- `toMinutes`: convertește un șir de caractere în timpul de date Time, în formatul hh:mm
- `addMinutes`: folosit pentru calcularea timpului de placare și cel de ajungere de la un obiectiv turistic la altul

Popularea matricelor de distanță și de durată

Distanțele și timpii de parcurgere de la un obiectiv turistic la altul se realizează prin trimiterea unei cereri HTTP către un serviciu extern, OSRM, Figura 16. Pentru calcularea rutei dintre două locații se utilizează coordonatele geografice, longitudinea și latitudinea locației de origine și de destinație. Acesta va returna un răspuns cu distanța în km și durata în minute pentru modul de deplasare cu mașina, acesta fiind modul implicit.

```
145  // Populate distance and duration matrices
146  async function populateMatrices() {
147
148    async function calculateDistanceAndDuration(originLat, originLng, destLat, destLng) {
149      const apiUrl = `https://router.project-osrm.org/route/v1/${travelMode}/${originLng},${originLat};${destLng},${destLat}
150      const response = await fetch(apiUrl);
151      const data = await response.json();
152
153      const distance = data.routes[0].distance / 1000; // Convert distance to kilometers
154      const duration = Math.ceil(data.routes[0].duration / 60); // Convert duration to minutes
155
156      return { distance, duration };
157    }
```

Figura 16: Cerere OSRM pentru extragerea distanțelor

În implementarea inițială s-a optat să se folosească serviciile de la Google Maps pentru extragerea distanței și a duratei, fiind o variantă mult mai eficientă, dar din cauza costului ridicat al utilizării platformei s-a decis abordarea unei alte variante, cea prezentată anterior.

În ceea ce privește performanța algoritmului de generare, aceasta depinde de numărul de obiective turistice din orașul destinație, cu cât sunt mai multe cu atât se execută mai multe cereri API, pentru extragerea fiecărei distanțe dintre fiecare obiectiv, fiind o operație foarte costisitoare. Pentru demonstrarea funcționalităților s-a ales un număr limitat de obiective pentru fiecare oraș înregistrat, între 13-18 obiective turistice, și s-a setat un timp de maxim un minut pentru ca sugestia de itinerariu să fie disponibilă utilizatorului, în cazul în care timpul este depășit se va afișa un mesaj sugestiv și se va reîncerca generarea itinerariului.

4.3. Testarea aplicației

Pentru testarea aplicației s-a optat pentru folosirea a două tipuri de testare: testare manuală pentru verificarea funcționalităților aplicației și componentelor ce aparțin de interfața grafică, și testarea API pentru a verifica schimbul de date dintre sistem și API. În scopul simplificării procesului de testare s-a creat un nou tip de utilizator, cel cu rolul de *SUPERUSER*, care are toate permisiunile, acesta putând să accese orice funcționalitate disponibilă din ambele aplicații: cea principală și cea internă.

Testarea funcționalităților și a componentelor

Testarea manuală este un tip de testare software, în care persoanele responsabile trebuie să simuleze toate scenariile posibile atât din perspectiva unui utilizator experimentat, cât și din cea a utilizatorului nou, indiferent de intențiile acestuia, scopul fiind de a identifica diferite probleme ale aplicației și de a le raporta pentru a putea fi rezolvate în timp util.

În cadrul testării aplicației dezvoltate s-a utilizat următoarea abordare:

- exploatarea aplicației, neexistând cazuri de testare predefinite, prin care s-au încercat într-o manieră ad-hoc diferite intrări, atât valide cât și invalide, și scenarii pentru diferite permisiuni în funcție de rolul utilizatorilor pentru a găsi defecte, această abordare poartă numele de testare exploratorie
- crearea de cazuri de testare bazate pe posibile scenarii ale utilizatorilor, cazurile de utilizare au fost prezentate în capitolul 4.1.2, și execuția manuală a acestora
- testarea interfeței cu utilizatorul, prin care s-a verificat dacă elementele vizuale ale aplicației, cum ar fi butoane, casete text, formulare etc. sunt afișate corect și funcționează conform cerințelor

În urma testării s-a constatat că toate cazurile prezentate în 4.1.2 funcționează corespunzător, iar excepțiile sunt tratate conform tipului de utilizator, acesta fiindu-i afișat o notificare corespunzătoare când este cazul.

Testarea schimbului de date

Testarea API este un tip de testare care verifică dacă API-ul funcționează în mod corect, îndeplinește cerințele și se comportă conform așteptărilor. Acest tip de testare s-a folosit pentru: validarea datelor și pentru testarea performanței, în ceea ce privește timpul de răspuns.

Postman este un instrument de testare special folosit pentru a testa, crea și documenta API-urile, cu ajutorul lui s-a realizat verificarea *endpoint-urilor.* Un endpoint reprezintă locația de unde se poate accesa o anumită resursă furnizată de un serviciu.

Pentru testare s-a creat colecția Postman, denumită *trip-planner*, ce conține fiecare tip de cerere necesar pentru endpoint-urile utilizate în aplicație: În Figura 17 (1) se poate observa structura colecției.

Endpoint-urile sunt restricționate, deci nu toți utilizatorii pot să le acceseze, cu excepția rolului de *SUPERUSER*, care are toate permisiunile. Mai întâi utilizatorul trebuie să se autentifice, în cazul în care nu o face sau credențialele sunt invalide, accesul este interzis pentru orice tip de cerere. După autentificarea cu succes se generează un token, Figura 17 (2), o cheie unică de acces, la fiecare cerere trimisă, pe baza căruia se pot realiza celelalte cereri.

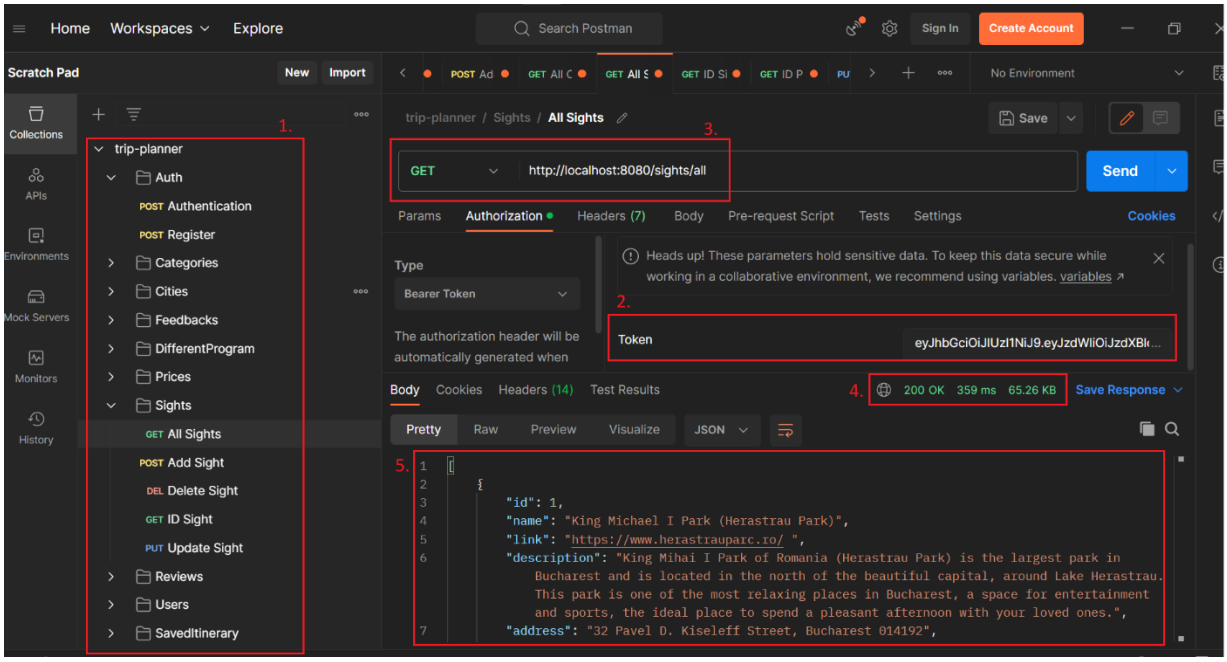


Figura 17: Testarea endpoint-urilor în Postman

Metodele folosite sunt cele de:

- POST: pentru crearea unei resurse noi
- GET: pentru obținerea datelor de la server
- PUT: pentru actualizarea unei resurse deja existente pe server
- DELETE: pentru a șterge o resursă de pe server

Pentru a se putea extrage obiectivele turistice de pe server se trimite următoarea cerere de GET: *http://localhost:8080/sights/all* Figura 17 (3), unde http://localhost:8080 specifică locația resurselor și reprezintă adresa URL, acronim ce vine de la *Uniform Resource Locator*, fiind partea comună pentru toate cererile, iar /sights/all este URI, din engleză *Uniform Resource Identifier,* prin care se identifică resursele solicitate. În cazul utilizatorului autentificat, cererea a fost procesată cu succes, având codul 200 Figura 17 (4), ceea ce indică acest lucru, răspunsul fiind afișat în spațiul dedicat Figura 17 (5). În cazul în care utilizatorul nu are permisiuni de acces la acest endpoint, cum este cazul utilizatorilor cu rolul de *ADMIN_MEMBER*, cererea răspunde cu codul 403, ce indică accesul refuzat, Figura 18.

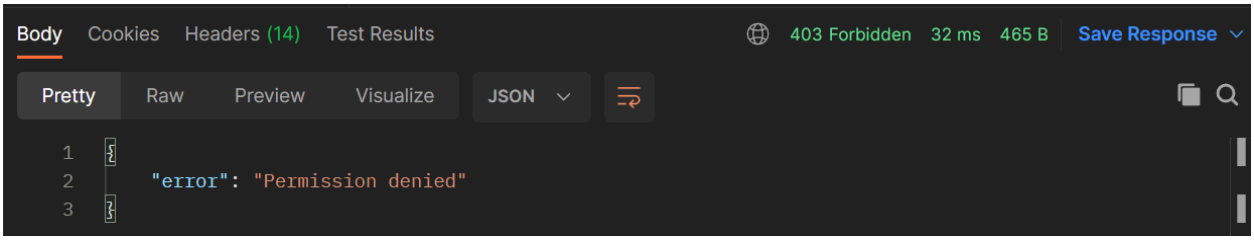


Figura 18: Mesaj de eroare

Capitolul 5

Studiu de caz

În acest capitol se prezintă și se analizează amănunțit câteva scenarii practice ce ilustrează aplicabilitatea și eficiența soluției propuse. Prin intermediul acestui studiu de caz, se dorește să se demonstrează modul în care soluția dezvoltată aduce beneficii, rezolvă problemele identificate și îndeplinește cerințele stabilite, oferind dovezi în ceea ce privește utilizările sale într-un context real.

Numele aplicație este compus din adjectivul „Nomadic” care sugerează dorința de a explora si de a calatorii in jurul lumii, îndemnând utilizatorul sa descopere locuri noi si îndepărtate. Iar „Monkey” s-a ales datorită existenței unei specii de maimuțe, numită vervet, cunoscută pentru abilitatea sa de a planifica traseul ce trebuie parcurs până la mâncare într-un mod optim.

La deschiderea aplicației web, utilizatorul este întâmpinat de o interfață primitoare, Figura 19, elementul principal fiind un *slider* cu imagini provenite din locații turistice. De pe pagina principală, un utilizator logat poate accesa paginile: de inspirație pentru călătorii, de planificare a călătoriei și profilul de utilizator. Dacă utilizatorul nu este logat, dar are cont pe platformă, acesta va fi redirecționat către pagina de login, Figura 20 a), iar dacă nu are încă cont, acesta va fi trimis către pagina de înregistrare, Figura 20 b). La înregistrare, utilizatorul trebuie să introducă datele sale personale, respectiv: numele, prenumele, email-ul și parola contului, iar necompletarea oricărui câmp va fi comunicată utilizatorului prin înroșirea marginii câmpului și comunicarea explicită a erorii, Figura 20 c) și cazurile nr. 1 și nr. 2 din prezentat în capitolul 4.

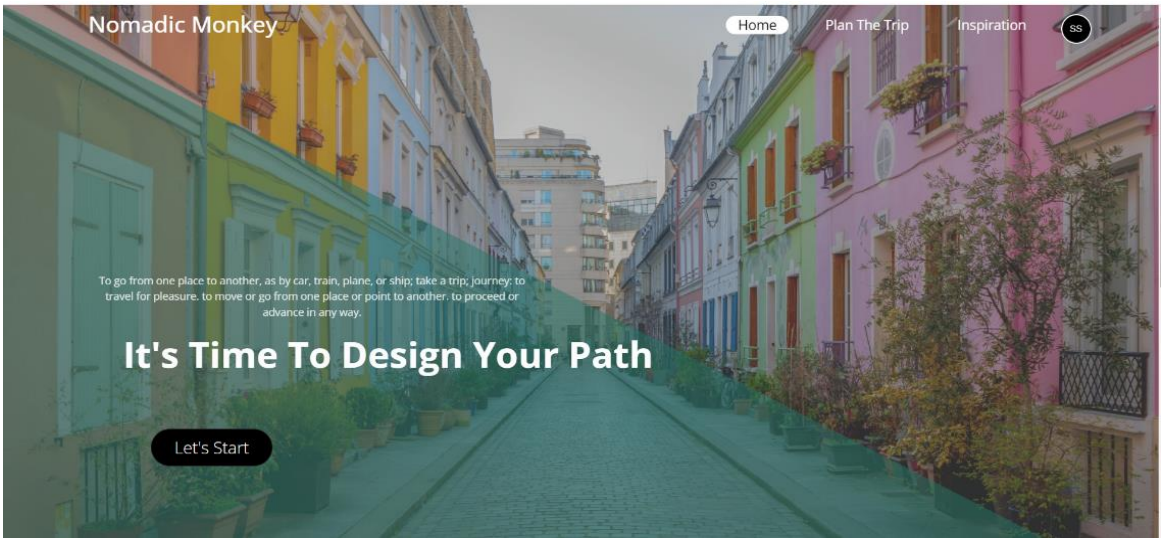
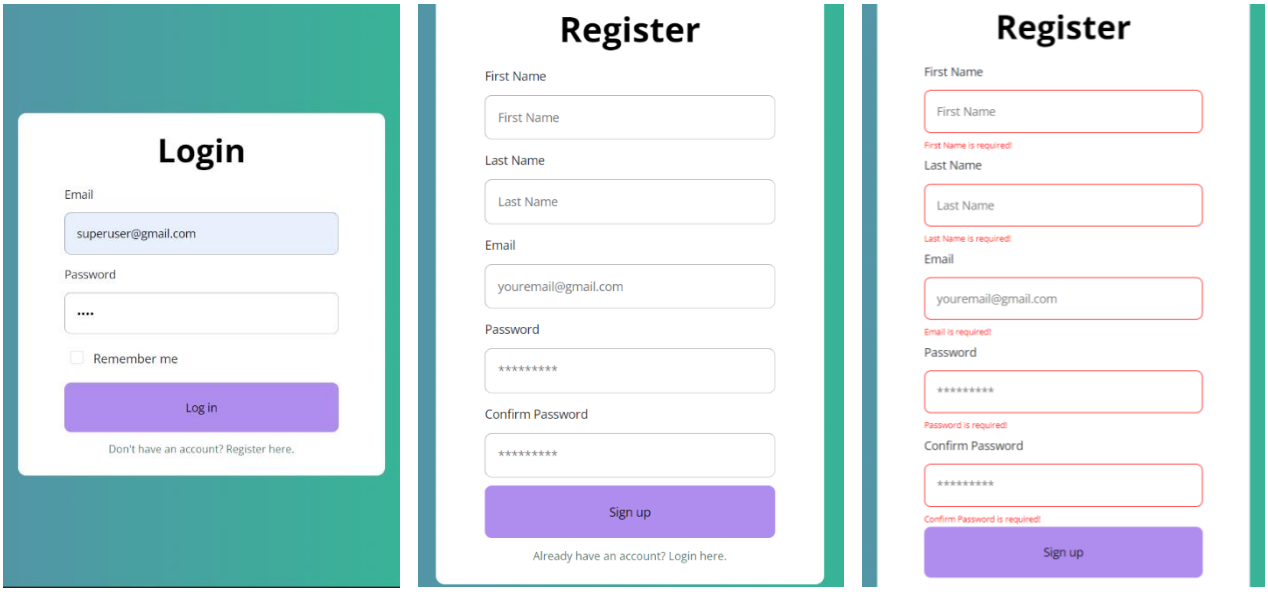


Figura 19: Pagina principală a aplicației de bază - Home



a) Login b) Înregistrare cont c) Tratarea erorilor

Figura 20: Conectarea și înregistrarea în aplicație

Pagina de inspirație pentru călătorie oferă utilizatului posibilitatea de a viziona posibilele obiective turistice pe care le poate vizita. Fiecare obiectiv turistic este reprezentat printr-o componentă card ce conține: o imagine reprezentativă, numele și locația, și ratingul, care ajută utilizatorul în deciziile sale, reflectând experiența celorlalți utilizatori, Figura 21 și cazurile nr. 3 și nr. 4 din capitolul 4.

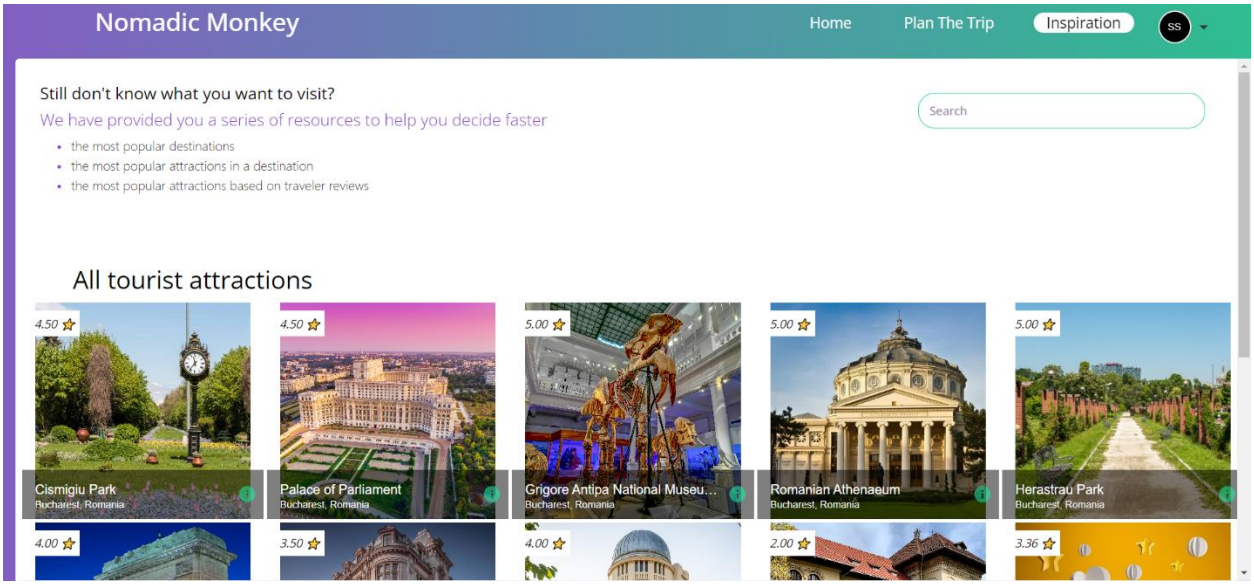


Figura 21: Pagina de inspirație pentru călătorii - Inspiration

Prin apăsarea iconiței versi de *info* pentru o atracție turistică de către utilizator, un pop-up situat pe centrul paginii se deschide, unde se pot vizualiza detaliile despre atracția turistică selectată și recenziile altor utilizatori ce au vizitat locul respectiv, și de asemenea utilizatorul poate lăsa o recenzie despre acesta, Figura 22 și cazul nr 5 din capitolul 4.

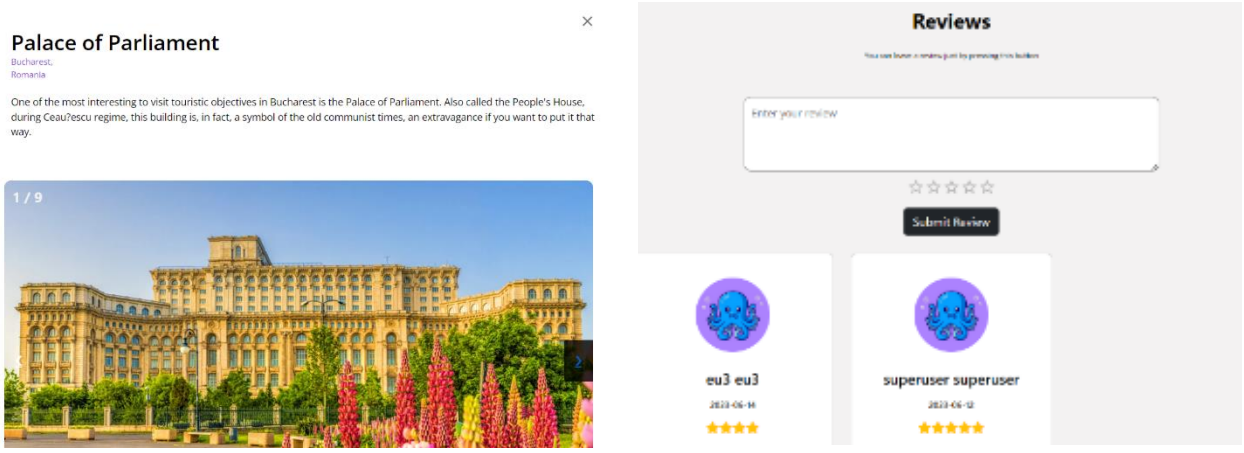


Figura 22: Pop-up-ul de vizualizare a detaliilor despre obiective și de recenzii

În cazul în care utilizatorul știe destinația pentru care vrea să planifice itinerariul, se poate naviga direct către pagina pentru a realiza acest lucru, Figura 23. Acesta trebuie să specifice obligatoriu țara și orașul pe care intenționează să o viziteze, împreună cu ziua și ora la care va ajunge, respectiv ziua și ora de întoarcere. De asemenea, utilizatorul are posibilitatea să specifice zona unde este cazat sau orice altă locație pentru a putea fi folosită ca loc de plecare în generarea itinerariului.

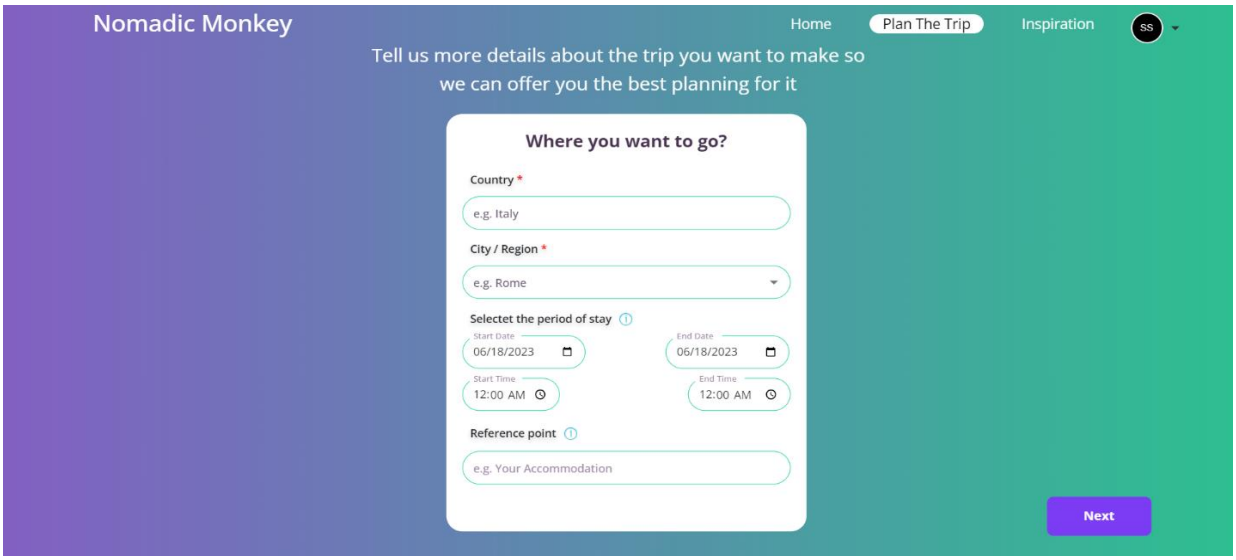


Figura 23: Introducerea datelor de intrare Plan The Trip

După ce au fost stabilite detaliile temporale și spațiale ale călătoriei, utilizatorul este redirecționat către pasul următor, unde are posibilitatea să selecteze obiectivele turistice pe care le dorește să le prioritizeze, Figura 24. Selectarea atracției se face prin apăsarea simplă a obiectivului, această acțiune marcându-se printr-un contur verde.

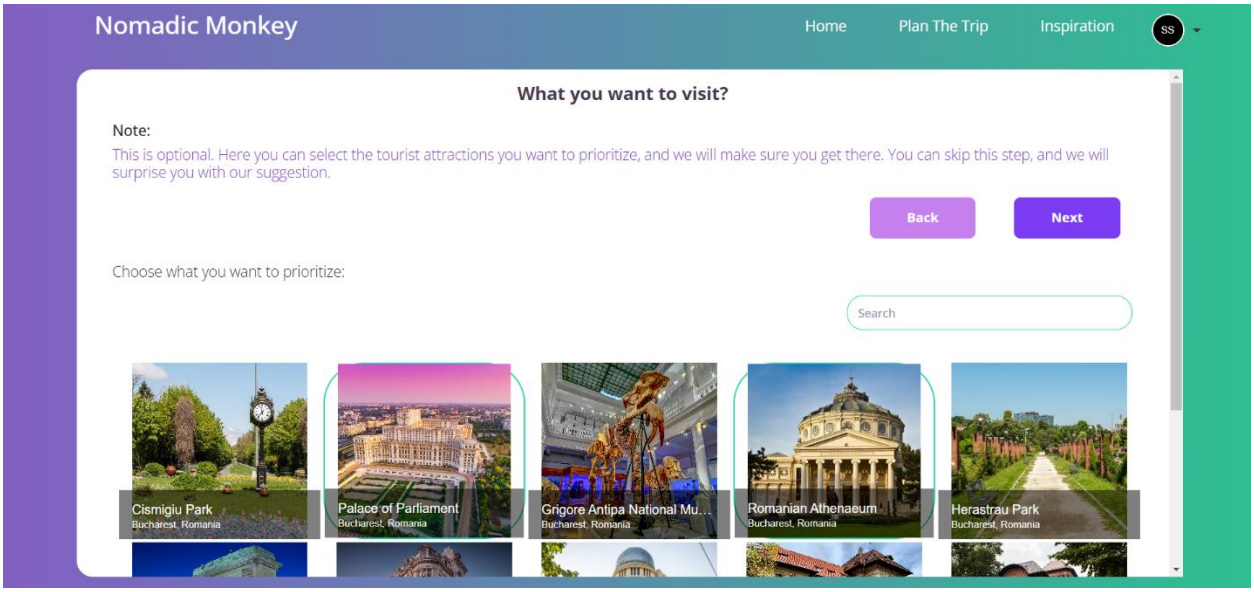


Figura 24: Pagina unde se selectează prioritatea atracțiilor turistice

După finalizarea acestei etape, se poate genera itinerariul. Timpul de încărcare a itinerariului depinde de numărul de obiective disponibile din locația destinație, cu cât sunt mai multe cu atât durează mai mult, dar nu trebuie să depășească mai mult de 1 minut.

O dată cu generarea itinerariului, acesta va fi afișat pe o grilă cu trei coloane și un număr variabil de rânduri, în funcție de numărul de zile al călătoriei. Fiecare zi a călătoriei este afișată separat pe un card, iar itinerariul din ziua respectivă conține componenta *timeline* din Material UI cu obiectivele planificate pentru ziua respectivă ordonate în funcție de ratingul și de disponibilitatea lor, și ora aproximativă de ajungere la obiectiv, ținând cont că deplasarea se face cu mașina. Culoarea bulinei ce delimitează obiectivele este setată în funcție de scorul acestora verde dacă scorul este mai mare decât 4.50 și mov închis dacă este mai mare decât 3.50 și mov deschis pentru restul valorilor. Aplicația oferă și o vizualizare practică a acestui itinerariu, având posibilitatea de a afișa traseul pe o hartă. Din această pagină utilizatorul poate să descarce itinerariul într-un fișier pdf, pentru a-l putea partaja ușor cu însoțitorii săi de călătorie, și să îl adauge la itinerariile sale favorite pentru a îl putea viziona mai târziu, Figura 25.

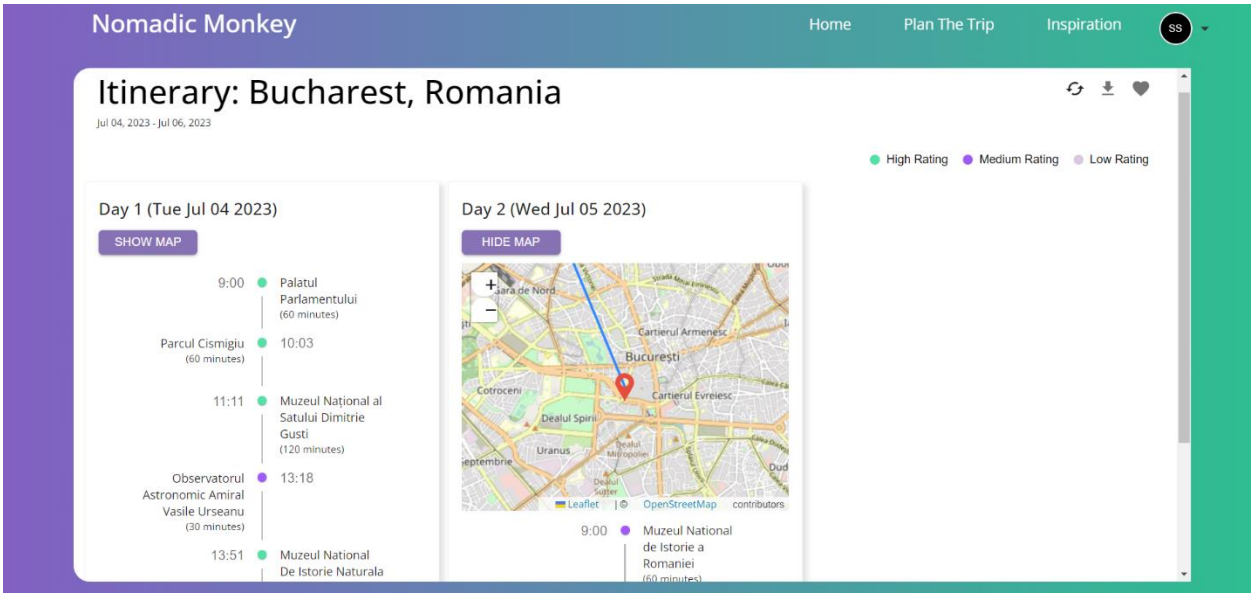


Figura 25: Itinerariul generat în funcție dedatele de intrare

Bazat pe felul cum a interacționat utilizatorul cu aplicația, acesta poate lăsa o recenzie, împreuna cu sugestiile de îmbunătățire a funcționalităților actuale sau chiar de adăugare a unor noi funcționalități, Figura 26.

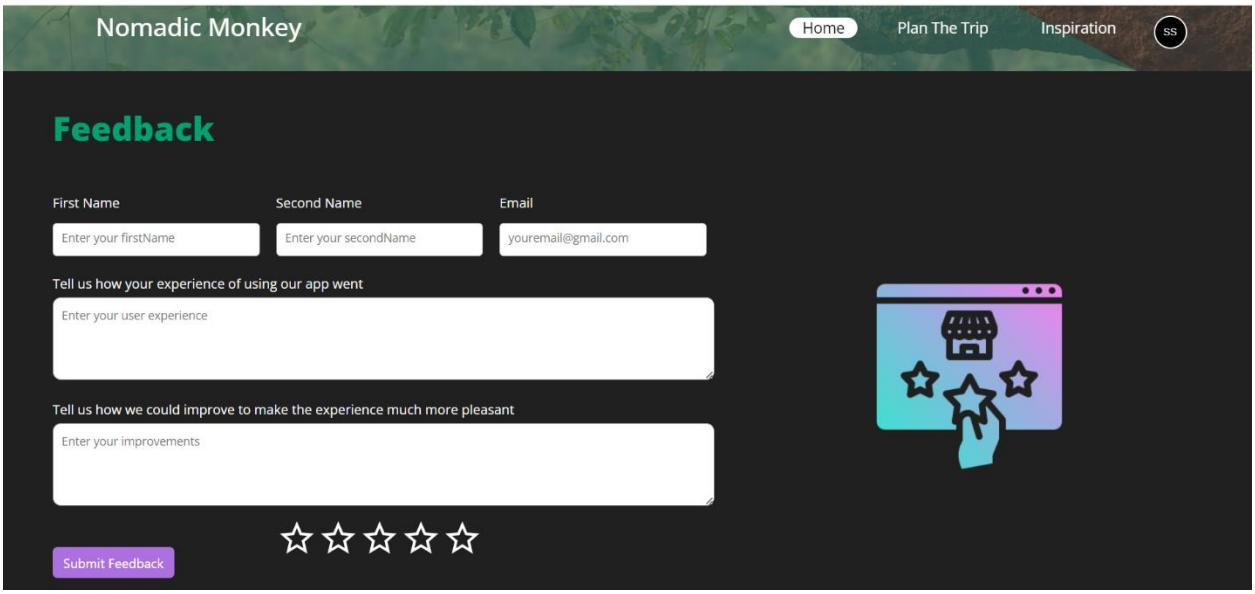


Figura 26: Pagina unde utilizatorul poate să lase o recenzie aplicației

În ceea ce privește aplicația internă, aceasta este mult mai simplă. Pagina principală afișează două opțiuni, fiecare fiind specifică tipului de utilizator: persoană administrativă și persoană de cercetare a pieței. Utilizatorii fiecărei echipe au rezervate anumite drepturi, iar încercarea accesării unei zone restricționate va fi revocată de către sistem și întâmpinată de un mesaj de eroare, Figura 27.

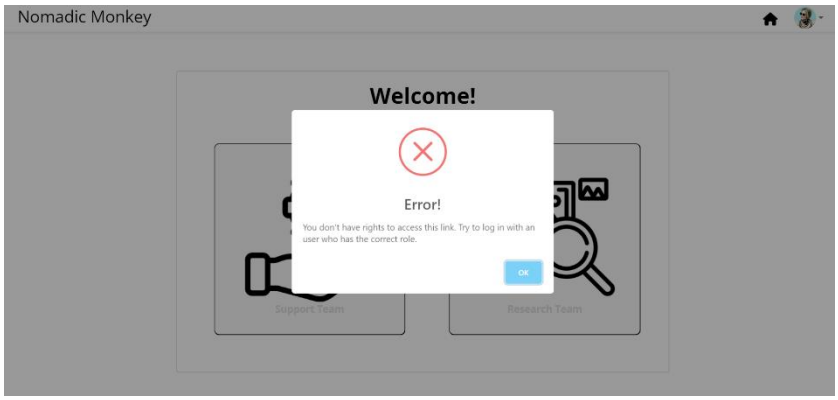


Figura 27: Mesaj de eroare în cazul utilizatorilor restricționați

Funcționalitățile puse la dispoziție echipei de cercetare a pieței constau în monitorizarea obiectivelor existente deja în baza de data cât a recenziilor acestora. De asemenea, sunt puse la dispoziție și numărul total de atracții turistice înscrise pe platformă împreuna cu numărul total de orașe și țări în care aceste sunt situate. Aceștia pot vizualiza si actualiza detaliile fiecărui obiectiv turistic, care cuprind: numele, locația, un link cu pagina web asociata si datele de contact, împreuna cu orele de deschidere si de închidere ale acestuia, și alte detalii, Figura 28 și cazurile nr. 6 și nr 7 din capitolul 4 .

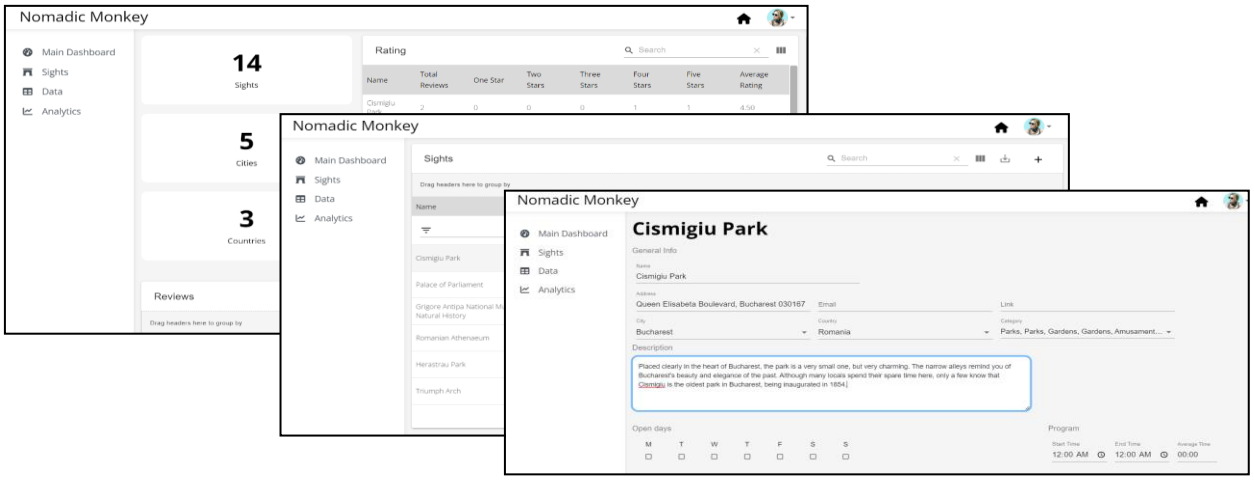


Figura 28: Funcționalitățile echipei de cercetare a pieței

Pentru a evita adăugarea datelor eronate care ar putea cauza erori în funcționarea aplicației, o serie de verificări sunt efectuate înainte de a salva, modifica sau șterge informațiile în baza de date. Succesul sau eșuarea acestui proces de validare va fi comunicat intuitiv utilizatorului așa cum este ilustrat în Figura 29.

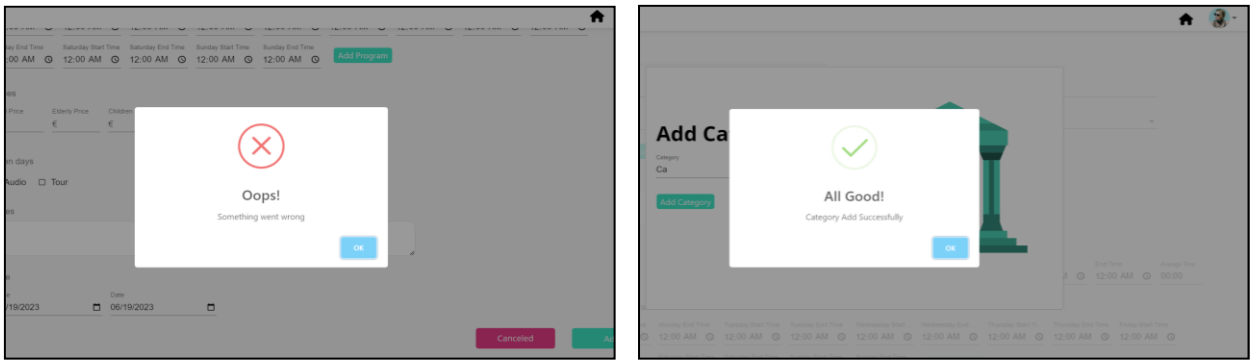


Figura 29: Mesaje de eroare și de succes în urma cererilor făcute de utilizator

Pentru echipa administrativă sunt puse la dispoziție informații despre starea platformei, incluzând numărul total de utilizatori și de ratinguri oferite aplicației, împreună cu rating mediu al acesteia. În această pagină, sunt de asemenea, oferite echipei administrative detalii despre numărul de angajați al celor două echipe. Pentru a analiza mai bine recenziile utilizatorilor, acestea pot fi vizualizate individual într-un tabel, în care fiecare rând conține email-ul utilizatorului, data scrierii recenziei, rating-ul, feedback-ul și îmbunătățirile sugerate. Acesta poate vizualiza și grafice pentru aceste lucruri, Figura 30.

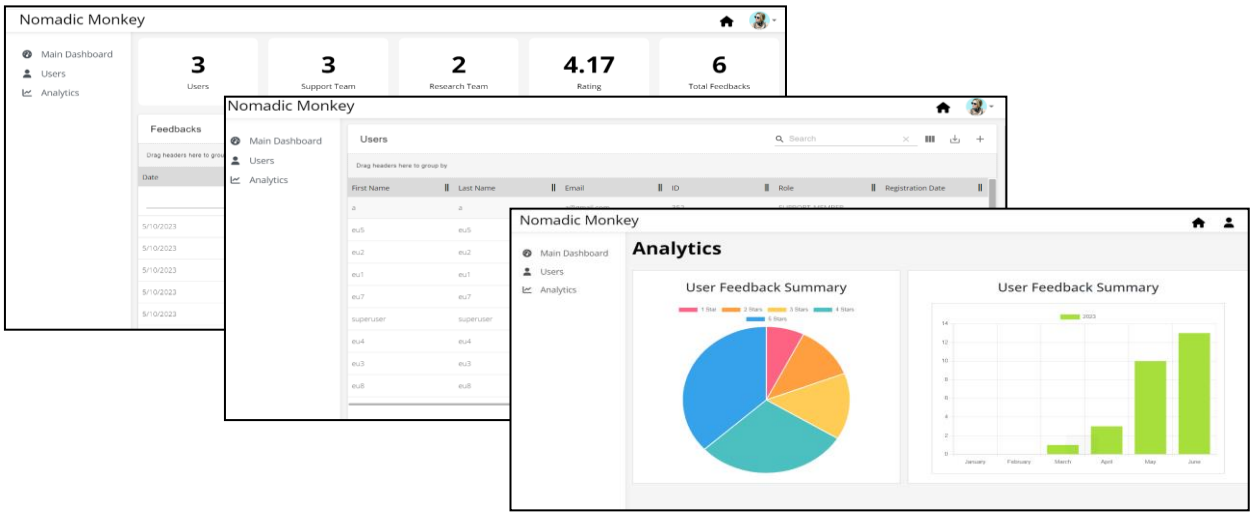


Figura 30: Funcționalitățile echipe de administrare

Capitolul 6

Concluzii și direcții viitoare de dezvoltare

6.1. Concluzii

În această lucrare au fost atinse obiectivele menționate la început, acestea fiind implementarea unui algoritm care să genereze un itinerariu personalizat pentru fiecare zi a vacanței, ținând cont de orarul de funcționare și de prioritatea atracțiilor turistice, și crearea unei aplicații web ușor de utilizat pentru ca utilizatorii să poată să beneficieze de această funcționalitate.

Problema de organizare a unui plan de călătorie, în ceea ce privește obiectivele turistice, este rezolvată de soluția propusă, ce folosește algoritmul de tip Greedy pentru selectarea celei mai bune atracții turistice la fiecare pas, criteriul de selecție bazându-se pe euristici, iar astfel se obține eficientizarea itinerariilor în funcție de multiple criterii. Itinerariile generate sunt adaptate în funcție de nevoile și preferințele fiecărei persoane, aceștia putând să selecteze atracțiile turistice pe care le consideră importante și să stabilească o locație de origine, care să reprezinte punctul de plecare pentru fiecare zi.

O problemă întâmpinată în timpul implementării a fost cauzată de expirarea perioadei de gratuitate a serviciului oferit de Google Maps, ce era necesar pentru extragerea distanțelor dintre obiective, ceea ce a avut ca rezultat pierderi semnificative de timp și efort suplimentar în cercetarea și dezvoltarea unei metode alternative pentru a asigura continuitatea funcționalității.

Implementarea soluției se îmbină cu o interfață grafică web, astfel încât, prin intermediul aplicație rezultate, utilizatorii să poată vizualiza itinerariile generate, inclusiv harta cu rutele trasate dintre obiectivele turistice, grafice în legătură cu locațiile și obiectivele populare și detaliile despre fiecare obiectiv aflat în sistem. În plus vine și cu o interfață grafică internă, pentru peroanele care se ocupă de cercetare și de gestionarea atracțiilor turistice și cele care monitorizează utilizatorii înregistrați și părerile acestora despre aplicația principală pentru a le avea în vederea dezvoltării altor versiuni ale aplicației.

Timpul utilizatorului ce ar fi fost alocat pentru planificare se reduce considerabil de mult, procesul automatizat luând aproximativ 5 minut, cât se introduc datele necesare și se sugerează itinerariul. Pe lângă acest beneficiu, aplicația rezolvă problema deplasărilor inutile și poate diminua cheltuielile legate de transport, astfel îmbunătățind experiența utilizatorilor.

În concluzie, aplicația dezvoltată aduce un beneficiu în domeniul în care activează, întrucât propune o soluție inteligentă și practică pentru persoanele care doresc să viziteze cât mai multe obiective turistice într-un timp limitat, fiind forte util mai ales în cazul călătoriilor scurte sau destinațiilor cu multe atracții turistice.

6.2. Direcții viitoare de dezvoltare

Tema studiată are un potențial vast în ceea ce privește direcțiile viitoare de dezvoltare ulterioară, cuprinzând următoarele aspecte pentru a crea o aplicație complexă:

- îmbunătățirea logicii algoritmului de generare a itinerarului, ținând cont și de orele de vârf pentru a evita vizitarea unui obiectiv, astfel încât să se evite aglomerările și cozile lungi
- calcularea rutelor în funcție de mai multe moduri de transport: deplasarea pe jos și tranzit
- includerea altor activități suplimentare relevante în planificarea itinerariului, cum ar fi restaurante, spectacole, festivaluri sau diferite evenimente
- funcționalitatea de actualizare în timp real pentru a oferi utilizatorilor flexibilitate maximă, permițând modificarea datelor și a priorităților în orice moment. Astfel putând să se asigure că itinerariile sunt actualizate și relevante pe tot parcursul vacanței și oferindu-le utilizatorilor libertatea de a-si personaliza experiența în mod dinamic
- integrarea sistemului de rezervări și recomandări pentru a oferi o experiență completă utilizatorilor. Aceștia ar avea posibilitatea dea rezerva în avans biletele pentru obiective turistice, restaurante și evenimente.

Bibliografia

1 B. Groza (2008), *Introducere în inteligenţă artificială – aplicaţii cu strategii de căutare neinformate şi informate*, Universitatea Politehnica Timişoara

2 F. Farooq (2010), *Optimal Path Searching through Specified Routes using different Algorithms*, Master Thesis Computer Engineering Applied Artificial Intelligence

3 C. Cerrone, R. Cerulli, B.Golden (2016), *Carousel Greedy: A Generalized Greedy Algorithm with Applications in Optimization*, Computers & Operations Research

4 K. Chinnathambi (2017), *Learning React*, United States of America

5 J. Bullinaria (2019), *Data Structures and Algorithms*, University of Birmingham

Birmingham

6 C.S. Cappellari (2021), *Fundamentals of REST API*, https://dev.to/cassiocappellari/fundamentals-of-rest-api

7 A. Rubanau, *Bootstrap vs. Material-UI*, https://flatlogic.com/blog/bootstrap-vs-material-ui Accesat: 2023

8 C. Macchiarelli (2022) , *Transport & Infrastructure* https://www.economicsobservatory.com/international-travel-and-tourism

9 J. Wallis (2022*), How Does Google Maps Work?*, *https://webo.digital/blog/the-tech-behind-google-maps/*

10 V. Kanade (2022), *What is Heuristics?*, https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-heuristics/

11 Best Practices for Exceptional Web Design and Usability, https://blog.hubspot.com/guidelines-for-exceptional-website-design-and-usability.aspx

12 Docker, https://www.oracle.com/ro/cloud/cloud-native/container-registry/what-is-docker/

13 Documentație Google Maps, https://developers.google.com/maps/documentation

14 Documentație Distance Matrix API, Google Cloud Platform. https://developers.google.com/maps/documentation/distance-matrix Accesat: 2023,

15 Java Web Application Technologies, https://www.upgrad.com/blog/top-java-web-application-technologies/ Accesat: 2023

16 JWT Authentication and Authorization with Spring Boot 3 and Spring Security 6 , https://medium.com/jwt-authentication-and-authorization-with-spring-boot

17 OSRM, https://forge.naos-cluster.tech/aquinetic/mapotempo/osrm-backend/

18 PostgreSQL, https://www.ibm.com/topics/postgresql

19 React, https://legacy.reactjs.org/docs/getting-started.html

20 REST API, https://www.ibm.com/topics/rest-apis

21 Spring Boot, https://spring.io/projects/spring-boot

22 Spring Boot using JDBC, Connection Pool, Flyway, JDBC Template, SQL and Docker, https://www.classcentral.com/spring-boot

23 Spring Framework, https://docs.spring.io/spring-framework/docs/

24 Sygic, https://travel.sygic.com/en

25 TripAdvisor, https://www.tripadvisor.com/

26 TripIt, https://www.tripit.com/web