**Universitatea Tehnică “Gheorghe Asachi” din Iași**

**Facultatea de Automatică și Calculatoare**

**Domeniul Calculatoare și Tehnologia Informației**

**Specializarea Tehnologia Informației**

**City Moving -** **aplicație pentru călătorit și gestionat transportul în comun**

LUCRARE DE LICENTA

**Coordonator stiintific,**

șef lucrări dr.ing. Cristian Aflori

**Absolvent,**

Enache Ștefan

Iasi 2023

**Cuprins**

Capitolul 1. Introducere...................................................................................................................1

Capitolul 2. Fundamentarea teoretică și documentarea bibliografică pentru tema propusă............3

2.1. Concepte...............................................................................................................................3

2.1.1. Sistem informatic........................................................................................................3

2.1.2. Sistem de gestiune................................................................................................................3

2.1.3. Cod QR........................................................................................................................4

2.2. Tehnologii.............................................................................................................................6

2.2.1. Android...............................................................................................................................6

2.2.2. Kotlin………...................................................................................................................7

2.2.3. Java cu frameworkul SpringBoot..........................................................................................8

2.2.4 Node.js cu React ……………………………………………………………………………8

2.2.5. MongoDB....................................................................................................................9

2.2.6. MySQL....................................................................................................................9

Capitolul 3. Proiectarea sistemului................................................................................................11

3.1. Funcțiile sistemului............................................................................................................11

3.1.1. Detalierea funcțiilor sistemului.................................................................................13

3.2. Diagrame UML..................................................................................................................21

3.2.1. Diagrame use-case.....................................................................................................21

3.2.2. Diagrama de secvență................................................................................................23

3.2.3. Diagrama de activități...............................................................................................24

Capitolul 4. Implementarea sistemului..........................................................................................25

4.1. Implementarea aplicației server..........................................................................................25

4.1.1. Aplicatia client………................................................................................................27

4.1.2. Funcționalitatea aplicației..........................................................................................29

4.2. Implementarea aplicațiilor server.......................................................................................31

4.2.1. Implementarea aplicatiei client....................................................................................34

Capitolul 5. Testarea sistemului și rezultate experimentale...........................................................38

5.1. Testarea aplicației client.....................................................................................................38

5.2. Testarea aplicațiilor server.................................................................................................41

Capitolul 6. Concluzii....................................................................................................................44

Bibliografie...................................................................................................................................46

Anexe............................................................................................................................................47

6.1. Exemplu de cod sursă din aplicația client..........................................................................47

6.2. Exemplu de cod sursă din aplicația server.........................................................................50

**City Moving -** **aplicație pentru călătorit și gestionat transportul în comun**

Enache Stefan

**Rezumat**

Proiectul reprezinta un sistem de cumparat bilete si vizualizat rute de transport in comun fiind o aplicatie web de tip client server, formata din 3 componete: aplicatia de tip server-client ce ruleaza in cadrul retelei locale, dar si aplicatia de control bilete ce ruleaza pe dispozitivele Android. Sistemul este conceput pentru ajutarea utilizatorilor sa isi achizitioneze biletele necesare calatoriei cu transportul public in Iasi cat si sa ofere detalii referitoare la rutele si traseele disponibile. Sistemul se axează pe maximizarea eficienței și productivității cumpărătorului, menținând în același timp o interfață ușor de înțeles și de utilizat.

Utilizatorii vor putea să își creeze propriul cont, să isi cumpere bilete si sa vizualizeze istoricul tranzactiilor. De asemenea, la fiecare bilet se poate obtine un cod QR pentru a I se verifica daca acesta are legitimatie de calatorie valida. Fiecare utilizator poate vizualiza rutele de tramvai sau de autobuz disponibile, cat si locatia sa pe harta.

Sistemul este menit să ofere o alternativă rapidă, ușoară și mult mai convenabilă decât cautarea manuală pe paginile web sau casele de bilete ale companiei de transport. Principala componentă a proiectului este aplicația client-server. Interfața cu utilizatorul a fost concepută cât mai intuitiv posibil iar astfel, curba de învățare este mică sau inexistentă. Aplicația nu utilizează intensiv resurse grafice astfel că, în afară de prezența unei conexiuni la internet, nu este nevoie de alte resurse.

Ideea de bază a funcționării sistemului este următoarea: utilizatorul doreste să călătorească cu transportul public și să beneficieze de facilitățile oferite de aplicație pentru cumpărarea rapidă și ușoară a biletelor de călătorie. În continuare, prin intermediul aplicației un utilizator, pentru a a accesa aplicatia este necesar sa se inregistreze ca mai apoi sa se logheze in aplicatie. Apoi are la dispozitie o gama variata de optiuni aferente contului sau: sa cumpere bilete, sa vizualizeze istoricul tranzactiilor, sa afle informatii despre mijloacele de transport, sa fie verificat daca are un bilet valid, etc. Odată cu acesul in aplicatie, datele referitoare la transport si statii se obtin, prin intermediul API - ului oferit de către platforma online Opendata API, valoarile aferente fiind procesate la nivelul backend-ului si apoi oferite pentru vizulizare de catre utilizator prin intermediul interfatei web.

Desigur că procesele care au loc în partea de backend a aplicației sunt mult mai complexe decât ce a fost enumerat mai sus. Unul dintre obiectivele proiectării sistemului a fost păstrarea unei interfețe cu utilizatorul cât mai simple și mai clare, astfel ca utilizatorul să nu fie derutat de funcționalitățile aplicației. Un alt lucru important a fost dezvoltarea unei aplicații server care sa permita gestionarea datelor utilizatorilor si validarea biletelor. Serverul administrează datele personale ale utilizatorilor și tranzacțiile efectuate de aceștia, verifică codurile QR generate pentru a confirma autenticitatea biletelor/abonamentelor. Acest servicu, cel de cumparat bilete, este o componentă esențială in cadrul aplicatiei și care oferă interfața de vizualizare necesară pentru funcționarea optimă a întregului sistem. Servicul dispune de o bază de date minimală care conține va stoca biletele pentru fiecare utilizator. După ce un client achizitioneaza un bilet/abonament folosind serviciul cumparare bilete si tranzactia a reusit, în aplicația mobilă, aplicația CityMoverScan are capacitatea de a verifica autenticitatea cumparat de catre utilizator. Această operatiune se va efectua de catre un angajat care are acces la aplicatie asupra utilizatorului.

**Capitolul 1. Introducere**

In ziua de astăzi, tehnologia ocupă o parte din ce în ce mai semnificativă din viețile noastre. Aproape orice persoană, indiferent de vârstă, poartă asupra sa cel puțin un „mini-computer”,mult mai puternic decât orice mijloc de calcul de acum două zeci de ani. Astfel, am ales realizarea unui sistem informatic si de gestiune pentru achiziționării bilete/abonamente ce are drept component pricipal o aplicație web. De asemenea, în mod tradițional, procesul de informare cu refere la trasee, statii existente si timpii de asteptare este unul anevoios, în sensul că, utilizatorul trebuie să se afle în fața unui ghiseu sau sa afle informatii de la personae specializate pentru a putea obtine informatiile necesaree. Cu ajutorul aplicației, acest lucru nu mai este necesar deoarece totul se afla la un click distanta.

În contextul aglomerat al orașelor mari, călătoriile cu mijloacele de transport în comun pot fi uneori o provocare. Timpul este prețios, iar fiecare secundă contează, de aceea am dezvoltat această aplicație pentru a rezolva problemele întâmpinate de călătorii urbani. Oricine a utilizat transportul public știe că achiziționarea biletelor poate fi uneori complicată, iar accesul la informații actualizate despre rute și orare poate fi dificil.

Proiectul vine cu o incercare de a rezolva aceste probleme, oferind o soluție completă și modernă pentru a călători mai ușor și mai eficient în oraș.Aplicatia se axează pe îmbunătățirea modului în care oamenii interacționează cu transportul public, eliminând obstacolele întâlnite în procesul de cumpărare a biletelor și de planificare a traseelor.

"CityMover" este o aplicație inovatoare, concepută pentru a facilita călătoriile cu transportul public în Iași. Cu o interfață prietenoasă și funcționalități avansate, această aplicație oferă utilizatorilor posibilitatea de achiziționare a biletelor, de exploatare a rutelor disponibile și pentru a obține informații în timp real despre transportul în comun din oraș. Având în vedere nevoile și cerințele specifice ale utilizatorilor, “CityMover" oferă un mod convenabil și eficient de a cumpăra bilete pentru a calatori in siguranta cu mijloacele de transport din cadrul Companiei de Transport Public, Iasi. Utilizatorii pot alege tipul de bilet dorit și pot efectua plata securizată, eliminând astfel necesitatea de a cumpăra biletele fizic sau de a aștepta în cozi lungi.

Utilizatorii pot crea conturi personale, efectua plăți securizate și primi biletele direct pe dispozitivele lor, în format electronic, inclusiv coduri QR pentru validare rapidă. Aplicația dispune și de o bază de date actualizată cu traseele de autobuz și tramvai disponibile pentru a ajuta utilizatorii să-și planifice călătoriile. De asemenea, "CityMover" permite vizualizarea istoricului tranzacțiilor, facilitând urmărirea cheltuielilor și gestionarea expirarii abonamentului.

"CityMover" oferă utilizatorilor posibilitatea de a vizualiza rutele și orarele mijloacelor de transport în comun în timp real. Această caracteristică ajută utilizatorii să-și planifice călătoriile în avans și să evite întârzierile neplăcute. Fiecare bilet cumpărat prin intermediul aplicației este asociat cu un cod QR unic, care servește drept dovadă a legitimării călătoriei. Acest cod QR poate fi scanat de către personalul de control, asigurând călătoriile în condiții de siguranță și evitând fraudarea biletelor. De asemenea, se pune la dispoziție o hartă interactivă, permițând utilizatorilor atat să-și localizeze rapid stația și să găsească cele mai apropiate rute disponibile, cat si sa vizualizate traseele disponibile.

Utilizarea aplicației "CityMover" este extrem de simplă și accesibilă tuturor utilizatorilor. Acestia trebuie să-și creeze un cont personal. După autentificare, aceștia vor avea acces la întreaga gamă de funcționalități, inclusiv achiziționarea de bilete, vizualizarea rutelor, generarea codului QR și multe altele.

Nu în ultimul rând, " CityMover " oferă o experiență de cumpărături și călătorii simplă, rapidă și fără complicații, punând la dispoziția utilizatorilor toate resursele necesare pentru a se bucura de transportul public în Iași într-un mod eficient și plăcut. Această aplicație își propune să revoluționeze modul în care oamenii călătoresc și interacționează cu transportul public, aducând beneficii semnificative pentru întreaga comunitate urbană. Prin intermediul caracteristicilor sale avansate și a interfeței prietenoase, “CityMover" face călătoriile în oraș mai rapide, mai ușoare și mai plăcute. Obiectivul meu este să creștem eficiența și confortul în transportul urban, oferind utilizatorilor acces la informații actualizate și posibilitatea de a călători cu încredere. Cu "CityMover", călătoriile în oraș devin o experiență plăcută și relaxantă, iar tehnologia modernă devine aliatul nostru pentru o mobilitate mai bună.

**Capitolul 2. Fundamentarea teoretică și documentarea bibliografică pentru tema propusă**

Tema proiectulul este reprezentată de realizarea unui sistem informatic si de gesiune ce imbina doua aplicatii diferite: MoveIt si 24Pay. Astfel, principalul scop al sistemului este de a oferi utilizatorilor o alternativă de a obtine informatii si de cumpărare a biletelor/abonamentelor flexibilă și rapidă. De aceea, aplicația a fost concepută cât mai simplu și intuitiv posibil.

Datorită faptului că sistemul este destinat utilizării pe platforme web, dimensiunea ecranului și rezoluția nu au fost factori ce au avut considerente majore de-a lungul procesului de proiectare a interfeței cu utilizatorul. Performanța nu reprezintă o problemă, întrucât request-urile catre serverul sistemului implică bucăți mici de date.

"CityMover" se bazează pe o fundamentare teoretică solidă și o documentare bibliografică riguroasă pentru a dezvolta o aplicație mobilă eficientă și inovatoare în domeniul transportului public. Înainte de a construi aplicația, am efectuat cercetări aprofundate și am analizat nevoiele aplicatiei si constrangerile aferente, precum și alte aplicații similare existente pe piață precum MoveIt, HereItIs, 24Pay. Acest proces a permis identificarea celor mai bune practici și tehnologii potrivite pentru a crea o aplicație robustă și utilizabilă.

Am realizat o analiză amănunțită a transportului public urban, cu accent pe specificul orașului Iași. De asemenea, am studiat caracteristicile infrastructurii, tipurile de mijloace de transport disponibile, rutele frecventate și problemele comune cu care se confruntă călătorii.

Această abordare a ajutat la înțelegerea nevoilor utilizatorilor și așteptărilor lor în ceea ce privește o aplicație destinate transportului.

Am analizat tehnologiile moderne, limbajele de programare, platformele de dezvoltare și am tinut cont de cunostintele dobandite in cadrul anilor de facultate pentru a alege cele mai fiabile si bune resure care se pliaza pe cerintele mele. S-a acordat atenție securității datelor și performanței aplicației. M-am informat despre tehnologia codului QR și modul în care acesta poate fi utilizat pentru validarea biletelor. Pentru acest pas am folosit o librarie React.

De asemenea, in cazul modului de stocare al datelor am cautat metodele eficiente de gestionare a bazelor de date pentru a stoca informații despre utilizatori, bilete si rute.

Interfața cu utilizatorul a fost concepută cu flexibilitate ca principala prioritate. Aplicația este prezentată și organizată într-o manieră care este și plăcută la vedere, și ușor de înțeles pentru ca utilizatorii să navigheze fără efort.

Aceste aspecte au fost luate in considerare și utilizate pentru a construi aplicația"CityMover" care să îndeplinească nevoile utilizatorilor, să ofere o soluție completă și eficientă pentru călătoriile cu transportul public în Iași, dar sa ating si obiectivele pe care mi l-am propus la inceputul proiectului. Pe parcursul dezvoltării, m-am confruntat cu multiple ajustări pentru a ma asigura de funcționalitatea aplicației, respectând cele mai bune practici din domeniu. Rezultatul final este o aplicație mobilă inovatoare, pregătită să îmbunătățească experiența călătoriilor în transportul public și să faciliteze accesul la informații esențiale pentru utilizatori.

**2.1. Concepte**

**2.1.1. Sistem informatic**

Un sistem informatic este un sistem care permite introducerea de date prin procedee manuale sau prin culegere automată de către sistem, stocarea acestora, prelucrarea lor și extragerea informației (rezultatelor) sub diverse forme. Componentele sistemului informatic sunt: calculatoarele, programele, rețelele de calculatoare și utilizatorii. Exemple: cartea de telefoane a unui anumit operator, repertoriul de legi inclusiv funcția lor activă și pasivă, bănci de date medicale, sistemele de colectare și analiză a datelor furnizate de un telescop și multe, multe altele.

În ceea ce privește structura lor, multe sisteme informatice folosesc într-o măsură mai mică sau mai mare redundanța pentru reține toate informațiile importante pentru aplicație, precum datele utilizatorilor, informațiile despre bilete, trasee, stații și alte detalii relevante pentru transportul public.

Aplicatia server este componentă centrală a sistemului, responsabilă cu gestionarea cererilor și răspunsurilor de la aplicația web. Serverul se ocupă de procesarea datelor, interacțiunea cu baza de date și gestionarea logicii de afaceri. Partea de securitate asigura protecția datelor și prevenirea accesului neautorizat. Am implementează măsuri de Securitate precum: criptarea parolelor și autentificare pentru a proteja datele utilizatorilor și informațiile.

**2.1.2. Sistem de gestiune**

Sistemul de gestionare al transportului reprezintă partea a aplicației care se ocupă de administrarea și organizarea resurselor transportului public, multe dintre aceste date fiind obtinute de la Opendata API. Aceasta include:

* Planificarea rutelor: Administrarea rutelor disponibile pentru tramvaie și

autobuze, cu detalii despre trasee, orare și stații.

* Gestionarea stațiilor: Informații despre stațiile de îmbarcare/debarcare, inclusiv locația lor pe hartă, facilitățile disponibile și alte detalii relevante.
* Monitorizarea transportului: Administrarea în timp real a vehiculelor în mișcare, pentru a oferi utilizatorilor informații precise despre sosiri și plecări, precum și starea traficului.
* Statistici și rapoarte: Generarea de rapoarte și statistici cu privire la utilizarea transportului public, gradul de ocupare, rute populare etc., pentru a ajuta la luarea deciziilor și îmbunătățirea serviciilor.

.

Prin integrarea unui sistem informatic și a unui sistem de gestionare a transportului, aplicația "CityMover" poate oferi utilizatorilor o soluție completă și inovatoare pentru a călători cu transportul public în Iași. Faptul ca permite utilizatorilor să cumpere bilete pentru transportul public si să afle informatii despre traseele disponibile reprezinta o usurare a vietii acestora.

În plus, gestionarea eficientă a resurselor transportului public poate contribui la îmbunătățirea experienței călătorilor și la optimizarea serviciilor de transport în zonă.

**2.1.3. Cod QR**

Codul QR, cunoscut și sub denumirea de "Quick Response Code," este un tip de cod de bare bidimensional, care a fost inițial proiectat pentru industria de automobile din Japonia. Însă, datorită capacității sale de stocare ridicată și citirii rapide, codurile QR s-au răspândit pe scară largă în diverse domenii, inclusiv publicitatea de consum, urmărirea comercială, biletele de transport, marketingul produselor și multe altele.

Un cod QR este alcătuit din module negre, adică puncte pătrate, aranjate într-o grilă pătrată pe un fundal alb. Acest model poate fi citit de către dispozitivele de imagini, cum ar fi camerele foto, care apoi procesează imaginea cu ajutorul corecției de erori Reed-Solomon2 pentru a interpreta datele din codul QR. Informațiile stocate într-un cod QR pot fi de diverse tipuri, incluzând text, URL-uri sau informații de contact.

Inițial utilizate pentru urmărirea pieselor de automobile, codurile QR au fost rapid adoptate în diverse industrii. În prezent, ele sunt folosite în publicitatea de consum, permițând comercianților să ofere acces rapid și ușor la paginile web ale produselor sau serviciilor lor. Prin intermediul smartphone-urilor, utilizatorii pot scana codurile QR pentru a accesa rapid site-uri web sau pentru a obține informații despre produsele dorite, eliminând astfel necesitatea de a introduce manual adresele web într-un browser.

Un alt domeniu în care codurile QR au devenit extrem de utile este urmărirea comercială. Acestea sunt utilizate pentru identificarea și etichetarea produselor în magazine, permitând o gestionare mai eficientă a stocului și o urmărire mai ușoară a produselor. Aceste cupoane pot fi apoi utilizate pentru a obține reduceri și oferte speciale oferite de către companii.

Un alt aspect interesant al codurilor QR este utilizarea lor pentru stocarea informațiilor personale. Astfel, utilizatorii pot integra coduri QR în reviste, pe cărți de vizită sau pe alte obiecte pentru a oferi informații despre ei sau despre produsele lor. Totuși, codurile QR sunt proiectate să fie sigure și să nu transporte date executabile, cu excepția tipului de date URL care poate conține cod JavaScript, dar acesta este executat doar într-un context de navigare regulat.

Cu toate acestea, codurile QR pot fi utilizate cu succes într-o varietate de scenarii, inclusiv pentru furnizarea de informații legate de conturi bancare sau carduri de credit, cu condiția să fie interpretate numai de anumite aplicații ale furnizorilor de plăți

În ceea ce privește dimensiunea datelor stocate într-un cod QR, aceasta depinde de tipul de date, versiunea și nivelul corecțiilor de erori. Un cod QR poate conține până la 7089 caractere numerice, 4296 caractere alfanumerice, 2953 octeți/date binare și 1817 caractere Kanji . Modelul de mesaje este poziționat într-un model zig-zag de la dreapta la stânga, iar în simboluri mai mari, acest aspect este complicat de prezența modelelor de aliniament și utilizarea codurilor intercalate Reed-Solomon 2, care sunt folosite pentru corectarea erorilor.

În concluzie, codurile QR au devenit o metodă eficientă și sigură pentru transmiterea și stocarea informațiilor, fiind utilizate cu succes într-o varietate de aplicații și domenii, de la publicitate și marketing la gestionarea plăților și a produselor.

[[1]](#footnote-1)

**2.2. Tehnologii**

**2.2.1. Android**

Android este un sistem de operare mobil dezvoltat de către Google, conceput inițial pentru smartphone-uri și tablete. Android se bazează pe nucleul Linux și utilizează o arhitectură open-source, ceea ce a permis dezvoltatorilor să creeze o varietate de aplicații și servicii pentru această platformă.

Unul dintre principalele avantaje ale Android este diversitatea dispozitivelor pe care rulează. Este disponibil pe o gamă largă de produse, inclusiv smartphone-uri, tablete, ceasuri inteligente, televizoare inteligente și chiar unele electrocasnice.

O caracteristică-cheie a Android este personalizarea. Interfața utilizatorului este flexibilă și permite personalizarea aspectului și funcționalității dispozitivului. Utilizatorii pot alege din diferite teme, widget-uri și aplicații pentru a-și adapta dispozitivul la preferințele lor.

Android are și o arhitectură deschisă pentru aplicații, ceea ce înseamnă că dezvoltatorii pot crea și publica aplicații pentru platformă fără a avea nevoie de aprobare prealabilă din partea Google. Un alt aspect important al Android este faptul că este un sistem de operare open-source. Acest lucru înseamnă că codul sursă al sistemului de operare este disponibil public și poate fi modificat și adaptat de către dezvoltatori.

Cu toate acestea, Android a fost, de asemenea, criticat pentru fragmentarea sa. Datorită faptului că este disponibil pe o gamă atât de largă de dispozitive, producătorii pot personaliza interfața și funcționalitatea Android-ului în diferite moduri. Aceasta poate duce la diferențe semnificative în experiența utilizatorului între diferite dispozitive și poate încetini procesul de actualizare a sistemului de operare pentru anumite modele.

În concluzie, Android este o tehnologie mobilă de referință la nivelul licenței, care a revoluționat piața dispozitivelor mobile și a adus o varietate de opțiuni și funcționalități în viața utilizatorilor. Cu o bază imensă de utilizatori și o comunitate activă de dezvoltatori.

**2.2.2. Kotlin**

Kotlin este un limbaj de programare modern, concis și sigur. A fost anunțat ca limbaj oficial pentru dezvoltarea aplicațiilor Android de către Google în 2017. Kotlin a devenit rapid unul dintre cele mai populare limbaje de programare pentru dezvoltarea aplicațiilor Android, înlocuind în mare măsură Java, limbajul tradițional utilizat pentru acest scop.

Kotlin are un cod mai concis și mai expresiv în comparație cu Java, ceea ce înseamnă că dezvoltatorii pot scrie mai puțin cod pentru a realiza aceleași funcționalități. Aceasta duce la o productivitate crescută și la o gestionare mai ușoară a proiectelor.

De asemenea. este un limbaj static tipizat, ceea ce înseamnă că erorile de tip pot fi depistate în timpul compilării, reducând numărul de erori la timpul de execuție. Acest aspect aduce stabilitate și fiabilitate în aplicațiile Android.

Kotlin aduce multe îmbunătățiri în sintaxă și funcționalități în comparație cu Java. Un exemplu notabil este folosirea operatorului de nulitate null-safe, care ajută la gestionarea mai eficientă a excepțiilor NPE (NullPointerException).

În general, Kotlin oferă dezvoltatorilor o alternativă modernă și eficientă pentru dezvoltarea aplicațiilor Android, aducând îmbunătățiri semnificative în productivitate, siguranță și ușurință în gestionarea codului.

**2.2.3. Java cu framework-ul SpringBoot**

Java este un limbaj de programare de nivel înalt, orientat pe obiecte și platformă independentă, cunoscut pentru fiabilitatea și flexibilitatea sa. A fost lansat inițial de către Sun Microsystems în 1995 și de atunci a devenit unul dintre cele mai populare limbaje de programare utilizate în dezvoltarea de aplicații. Java permite dezvoltarea aplicațiilor care pot fi executate pe diverse platforme, de la dispozitive mobile și calculatoare personale până la servere și dispozitive înglobate.

Spring Framework este unul dintre cele mai populare și influente cadre de dezvoltare pentru aplicații Java. A fost creat de Rod Johnson și a fost inițial lansat în 2003. De-a lungul anilor, Spring Framework a evoluat și a devenit un ecosistem complet și complex, furnizând soluții pentru o gamă largă de provocări în dezvoltarea de aplicații Java.

Unul dintre principalele motive pentru popularitatea Spring Framework este modul său elegant de gestionare a dependențelor și inversiunii controlului (IoC). Prin IoC, Spring preia controlul asupra creării și gestionării obiectelor din aplicație, reducând astfel cuplarea între componente și facilitând testarea unitară și refactorizarea.

Un alt aspect important al Spring Framework este modul său de gestionare a tranzacțiilor. Spring oferă suport pentru tranzacții declarative, ceea ce permite dezvoltatorilor să definească tranzacții folosind adnotări și să se concentreze pe logica de afaceri, fără a fi nevoie să se ocupe de aspectele tehnice ale tranzacțiilor.

Spring Boot este o extensie populară a Spring Framework, care facilitează crearea rapidă și ușoară a aplicațiilor Java gata pentru producție. Cu Spring Boot, dezvoltatorii pot configura rapid o aplicație cu ajutorul setărilor implicite și a convențiilor. Acest lucru permite dezvoltatorilor să se concentreze mai mult pe dezvoltarea funcționalităților aplicației decât pe configurare.

Principala caracteristică a Spring Boot este abordarea sa "convenție peste configurare". Acest lucru înseamnă că, prin folosirea unor setări implicite și convenții bine stabilite, dezvoltatorii pot începe să dezvolte o aplicație Spring Boot fără a fi nevoie să configureze manual o mulțime de detalii.

Un alt beneficiu important al Spring Boot este integrarea ușoară cu alte proiecte și tehnologii din ecosistemul Spring. Spring Boot vine cu suport integrat pentru modulele de bază ale Spring Framework, precum Spring Core, Spring MVC, Spring Security și multe altele. Astfel, dezvoltatorii pot folosi cu ușurință aceste module pentru a construi aplicații complexe și performante.

Un aspect cheie al Spring Boot este capacitatea sa de a încapsula toate dependențele necesare într-un singur fișier JAR executabil. Acest fișier poate fi apoi rulat pe orice platformă care are instalată mașina virtuală Java (JVM), ceea ce face distribuirea aplicației foarte ușoară și portabilă.

Un alt avantaj al Spring Boot este suportul său pentru încorporarea serverului web direct în aplicație. Aceasta înseamnă că dezvoltatorii pot rula aplicația lor fără a fi nevoie să instaleze și să configureze un server web separat, cum ar fi Tomcat sau Jetty. Serverul web este încorporat direct în fișierul JAR al aplicației, ceea ce face pornirea și oprirea aplicației foarte simplă.

În concluzie, Spring Boot este un framework puternic și versatil pentru dezvoltarea de aplicații Java moderne. Abordarea sa convențională și integrarea ușoară cu alte module Spring fac din Spring Boot o alegere populară pentru dezvoltatorii Java din întreaga lume.

**2.2.4. Node.js cu React**

Node.js este o platformă open-source pentru dezvoltarea aplicațiilor scalabile de tip server sau rețea. Aceasta a fost creată de Ryan Dahl și a fost lansată pentru prima dată în 2009. Principala caracteristică a Node.js este capacitatea sa de a permite dezvoltatorilor să creeze aplicații de rețea cu performanțe ridicate, folosind JavaScript atât pe partea server, cât și pe partea client.

Node.js utilizează un model de programare asincronă și neblocantă, care maximizează eficiența și randamentul aplicațiilor. Acest model este posibil datorită utilizării operațiilor I/O neblocante și evenimentelor asincrone, care permit aplicației să continue să ruleze și să gestioneze mai multe cereri concurente fără a fi blocată.

Tehnologia mentionata are și o bibliotecă internă pentru operațiuni I/O, socket și comunicații HTTP, ceea ce îi permite să funcționeze ca un server web fără a avea nevoie de alte software-uri adiționale, cum ar fi Apache 3. Această abordare face ca dezvoltarea aplicațiilor web în Node.js să fie simplă și eficientă.

Un alt aspect important al Node.js este capacitatea sa de a manipula mai multe conexiuni client concurente, permițând astfel scalabilitatea la nivel înalt. Node.js utilizează o buclă eveniment pentru gestionarea conexiunilor, eliminând astfel necesitatea utilizării mai multor fire de execuție. Acest lucru îi permite să poată gestiona cu ușurință milioane de conexiuni simultane, ceea ce face platforma ideală pentru aplicații în timp real, cum ar fi aplicații de chat sau jocuri multiplayer online.

React este o bibliotecă JavaScript pentru construirea interfețelor utilizator (UI) în aplicații web. A fost dezvoltată inițial de către Facebook și a fost lansată în 2013. React se concentrează pe construirea componentelor UI reutilizabile, care sunt actualizate eficient și reprezintă o interfață de utilizator declarativă.

Principala caracteristică a React este utilizarea Virtual DOM (Document Object Model). Acesta este o reprezentare a memoriei a DOM-ului real al aplicației. Atunci când datele se schimbă în aplicație, React compară Virtual DOM cu DOM-ul real și actualizează doar elementele care sunt diferite. Aceasta reduce la minimum numărul de manipulări directe ale DOM-ului și face actualizările mai eficiente, ceea ce conduce la o performanță mai bună a aplicației.

Un alt avantaj al React este arhitectura sa component-based. Dezvoltatorii pot construi interfețe complexe împărțind aplicația în componente reutilizabile și gestionând starea și logica fiecărui component separat. Aceasta face codul mai ușor de întreținut și permite dezvoltatorilor să creeze rapid aplicații scalabile și complexe.[[2]](#footnote-2)

Node.js și React sunt două tehnologii JavaScript puternice și foarte populare în dezvoltarea aplicațiilor web moderne. Node.js oferă o platformă scalabilă și eficientă pentru dezvoltarea backend-ului, în timp ce React simplifică construirea interfețelor de utilizator interactive și performante. Combinația dintre Node.js și React oferă dezvoltatorilor un mediu puternic și prietenos pentru construirea aplicațiilor web complexe și moderne.

**2.2.5. MongoDB**

MongoDB este o bază de date NoSQL3 orientată pe documente, ceea ce înseamnă că evită structura tabelelor specifice bazelor de date relaționale și, în schimb, stochează datele în documente JSON4. Aceasta face integrarea datelor mai ușoară și mai rapidă, permițând dezvoltatorilor să lucreze cu date flexibile și să le modeleze în funcție de nevoile aplicației.

Dezvoltată inițial de către compania 10gen (acum MongoDB Inc.) în 2007, MongoDB a devenit rapid populară și, în 2009, a adoptat modelul open-source. Datorită caracteristicilor sale avansate și performanței ridicate, MongoDB este folosită de companii renumite precum Craigslist, eBay, Foursquare, MTV, Cisco, MetLife și New York Times, fiind cel mai popular sistem de bază de date NoSQL.

Mongo oferă suport pentru căutarea după câmpuri, interogări și expresii regulate. Aceste interogări pot returna câmpuri specifice din documente și pot include funcții JavaScript definite de către utilizator. De asemenea, orice câmp din interiorul unui document poate fi indexat, ceea ce optimizează căutările și interogările. MongoDB oferă și posibilitatea de a crea indecși secundari pentru îmbunătățirea performanței.

Unul dintre avantajele majore ale MongoDB este disponibilitatea ridicată și debitul crescut oferite de seturile de replici. Fiecare replică poate acționa ca replică primară sau secundară, iar în cazul unei defecțiuni a replicii primare, setul de replici alege automat o replică secundară pentru a deveni primară. Astfel, se asigură redundanță și disponibilitatea datelor.

De asemenea, MongoDB poate fi scalată orizontal prin procesul de sharding5, care implică împărțirea datelor în seturi pe baza unei chei de partiționare și distribuirea acestora pe mai multe partiții. Aceasta poate fi folosit ca și un sistem de fișiere, profitând de avantajul oferit de către principiul echilibrării încărcării și replicarea datelor peste mai multe mașini cu scopul stocării fișierelor.

**[[3]](#footnote-3)2.2.6. MySql**

MySQL este un sistem de management al bazelor de date relaționale open-source și foarte popular, utilizat pe scară largă pentru stocarea și gestionarea datelor în diferite aplicații web și software. A fost dezvoltat inițial în anii 1990, iar mai târziu a fost achiziționat de Oracle Corporation, dar rămâne disponibil sub licență open-source.

Oferind un set complet de funcționalități pentru gestionarea bazelor de date, MySQL este compatibil cu mai multe platforme, inclusiv Windows, Linux și macOS, ceea ce îl face să fie una dintre cele mai preferate soluții pentru dezvoltatori. Ca bază de date relațională, MySQL utilizează un model tabular pentru stocarea datelor, în care datele sunt organizate în tabele cu rânduri și coloane. Fiecare tabelă conține înregistrări individuale numite rânduri și câmpuri numite coloane, iar interacțiunea cu acestea se realizează folosind limbajul de interogare structurată SQL (Structured Query Language).

Unul dintre avantajele majore ale MySQL este performanța ridicată, ceea ce îl face potrivit pentru aplicații cu volum mare de date și trafic intens. Optimizarea interogărilor și gestionarea eficientă a indexării asigură că interacțiunea cu baza de date este rapidă și eficientă.

MySQL oferă de asemenea suport pentru multiple conexiuni simultane, ceea ce permite accesul concurent la date pentru mai mulți utilizatori. Aceasta face ca MySQL să fie potrivit pentru aplicații web care au nevoie de acces rapid și constant la date din mai multe surse.

**Capitolul 3. Proiectarea sistemului**

Aplicația mea de gestionare și informații pentru transportul public are un obiectiv clar de a oferi o alternativă ușor de utilizat și intuitivă pentru utilizatorii de toate vârstele care doresc să aibă acces rapid și convenabil la informații legate de transportul public. Prin utilizarea arhitecturii bazate pe microservicii REST, ai optat pentru o abordare scalabilă și ușor de întreținut, ceea ce permite o dezvoltare mai flexibilă și modulară.

Arhitectura pe microservicii REST implică împărțirea aplicației în servicii mici și independente, fiecare având rolul său bine definit. În cazul meu, cele trei componente ale aplicației, respectiv aplicația client, aplicația server și aplicația Android, rulează independent pe domeniul localhost, asigurând o separare eficientă a responsabilităților.

Folosind baze de date SQL și NoSQL pentru fiecare serviciu, am abordat un aspect important în proiectarea aplicației: stocarea datelor specifice fiecărui serviciu în mod separat pentru a menține o organizare eficientă și o gestionare mai ușoară.

Securitatea joacă un rol vital în protejarea datelor și a aplicației în ansamblu. Prin intermediul serviciului de securitate, datele sensibile sunt criptate în baza de date, iar accesul în aplicație se realizează prin intermediul unui token cu durată limitată, oferind astfel un nivel suplimentar de protecție.

Pe măsură ce aplicația devine mai populară are posibilitatea de a se extinde, am prevăzut posibilitatea de a adăuga noi funcționalități pentru a satisface nevoile utilizatorilor. Printre aceste funcționalități se pot număra: un serviciu de suport pentru clienți, setări personalizate, posibilitatea de a cumpăra abonamente sau bilete în funcție de zonele orașului și chiar integrarea cu mai mulți furnizori de transport din diverse orașe și municipii.

De asemenea, am evidențiat posibilitatea de a oferi utilizatorilor posibilitatea de a oferi feedback cu privire la diferite aspecte ale aplicației sau ale serviciilor de transport. Acest aspect este esențial pentru a menține o abordare centrată pe utilizator și pentru a identifica constant noi modalități de îmbunătățire a experienței lor.

Aplicația de gestionare și informații pentru transportul public are un plan bine conturat și o arhitectură bine structurată pentru a oferi o soluție modernă, eficientă și scalabilă. Cu atenție la detaliile de securitate și cu posibilitatea de a se extinde prin abordarea scalabilă și focusul pe experiența utilizatorilor, aplicația poate aduce valoare adăugată pentru toți cei implicați în ecosistemul transportului public. Cu o abordare constantă de îmbunătățire și inovare, se poate crea o aplicație care să răspundă mereu nevoilor în schimbare ale utilizatorilor și să îmbunătățească calitatea experienței de călătorie în transportul public.

**3.1. Funcțiile sistemului**

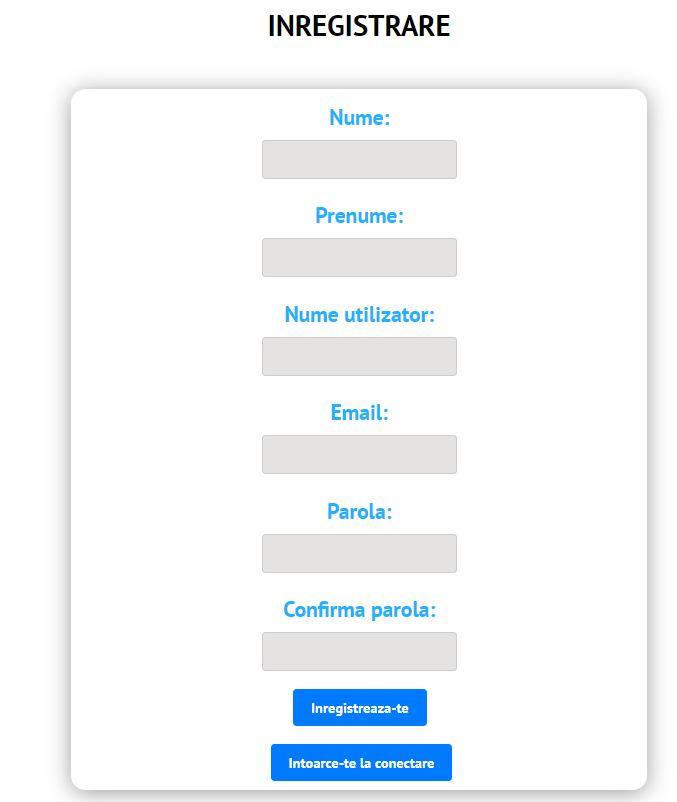
În continuare este prezentată o listă ce conține caracteristicile oferite de către sistemul dezvoltat. Întrucât principala componentă a sistemului este aplicația client-server, aceste caracteristici au fost formulate din prisma acestei componente:

* Înregistrarea/autentificarea utilizatorului
  + este accesibilă de fiecare dată când utilizatorul nu este autentificat
  + necesară pentru a putea folosi capacităție puse la dispoziție de către sistem
  + permite utilizatorilor să se înregistreze sau să se autentifice în cadrul sistemului
* Resetarea parolei
  + în cazul în care utilizarul și-a uitat parola, acesta poate primi una nouă prin intermediul email-ului
* Cumpararea biletelor
  + Este posibila cumpararea de bilete in cadrul aplicatiei si stocarea acestora in contul fiecarui utilizator
  + Posibilitatea de a alege tipul de bilet
  + Vizualizarea ultimelor 5 tranzactii
* Vizualizarea informatiilor despre rute si vehicule
  + Este posibila obtinerea informatiilor despre mijoacele de transport in comun puse la dispozitie de catre compania de transport
* Generarea codului QR
  + Este posibil generarea codului pentru un bilet, in vedera validarii acestuia
* Vizionarea traseului
  + Posibiliatea de a selecta un traseu si a-l vedea graphic pe harta

**3.1.1 Funcțiile sistemului**

Mai jos, sunt prezentate pe rând și în detaliu, funcționalitățile sistemului. De asemenea, descrierile sunt însoțite de imagini pentru o înțelegere mai ușoară a funcționalităților

*1. Înregistrarea unui utilizator*

****

*Figura 1.1: Înregistrarea utilizatorului*

Pentru a putea utiliza aplicatia, utilizatorul trebuie să aibă un cont personal în cadrul acestuia. Pentru realizarea acestui lucru utilizatorul trebuie inițial să își introduce datele personale, precum: nume, prenume, un nume de utilizator, adresa personală de e-mail și parola. Pentru mai multă siguranță, parola trebuie reintrodusă pentru a elimina situațiile în care utilizatorul a introdus din greșeală o parolă eronată. In cazul in care nu sunt introduce toate campurile, nu se va putea realiza inregistrarea contului, deoarece nu se permite trimiterea la server a unor date nule. După completarea tututor datelor, aplicația client va face o cerere către aplicația server a sistemului pentru crearea unui cont.

*2.Autentificarea utilizatorului*

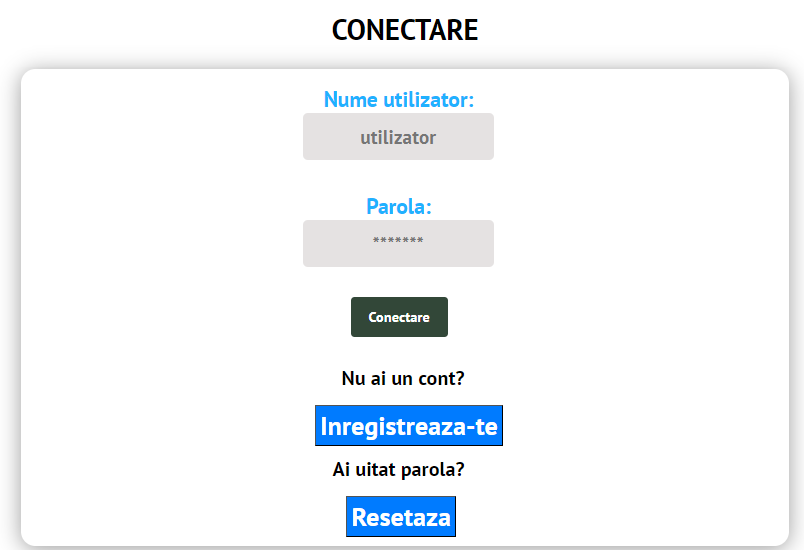
****

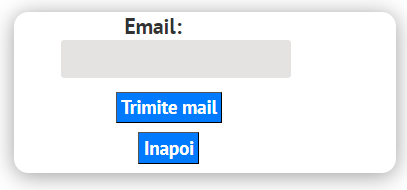
Figura 1.2: Autentificarea utilizatorului

Pasul următor implică introducerea datelor de conectare, iar la fel ca in cazul conectarii, daca campurile sunt goale sau completate partial, cererea nu este trimisa catre aplicatia server.

In cazul în care utilizatorul are deja un cont personal, el se poate autentifica în cadrul acestuia fără cel mai mic efort. Pentru realizarea acestui lucru acesta trebuie să își introducă numele de utilizator și parola.

*3.Recuperarea parolei*

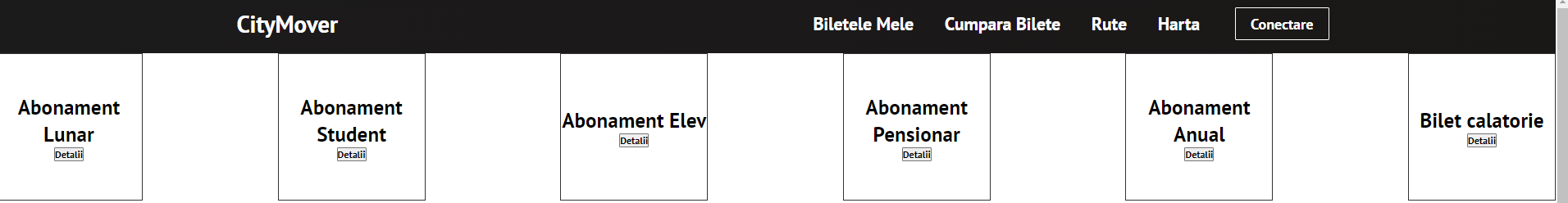
În cazul în care utilizatul și-a uitat parola, apăsând butonul din dreapta jos numit sugestiv „Reseteaza”, acesta va trebui sa își introducă adresa de e-mail. După ce utilizatorul a apăsat butonul „Trimite mail”, aplicația va face o cerere către server care va trimite un mail pe adresa furnizată, e-mail ce va conține un cod de confirmare si apoi va putea sa isi seteze o parolă aferentă contului utilizatorului.



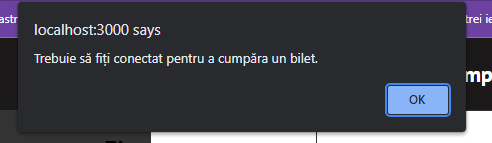
*Figura 1.2: Resetare a parolei utilizatorului*

*4. Achizitia de bilete sau abonamente*

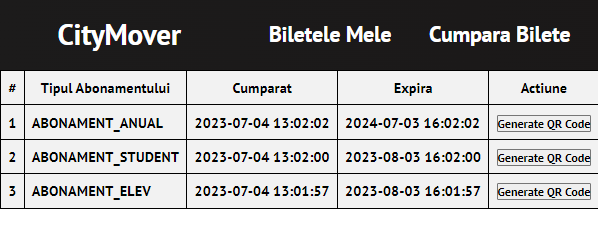
O data ce utilizatorul este conectat acesta are acces la aplicatie de unde isi poate cumpara bilete intrand in meniul de cumpart bilete. Acesta ii pune la dispozitie utilizatorului o gama variata de variata de optiuni in vederea nevoilor fiecarui client. Fiecare bilet este insotit de un pret si de detalii referitoare la valabilitatea acestuia.



In cazul in care un utilizator care nu esste logat, doreste sa achizitioneze un bilet, acesta este atentionat ca nu poate cumpara bilet daca nu este conectat si redirectionat pe pagina de conectare pentru a se conecta sau pentru a-si creea cont.

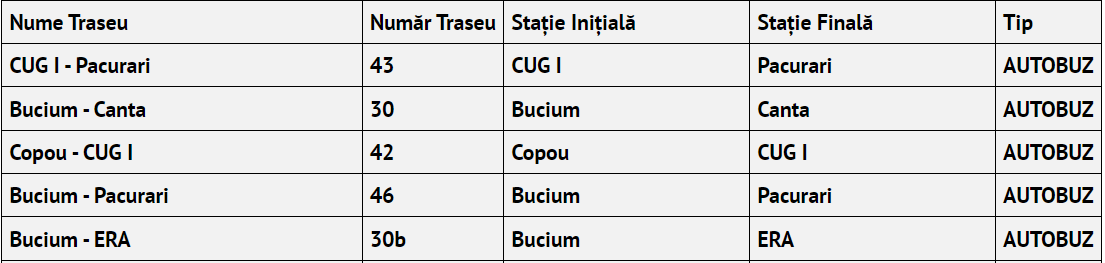


*5. Verificarea istoricului tranzactiilor unui utilizator*



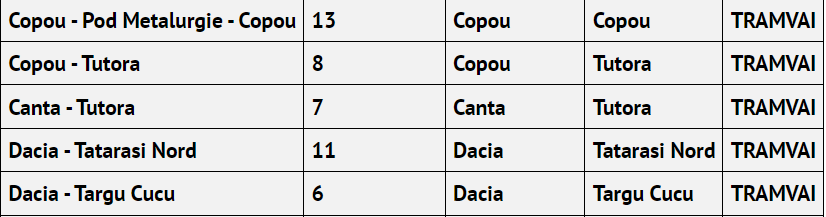
Aceasta functionalitate va afisa utilizatorului ultimele 5 tranzactii effectuate, acesta putand fi accesate in orice moment dupa conectarea in cont.

6. *Obtinerea datelor despre autobuze*



Aceasta functionalitate va putea fi accesata de catre un utilizator iar dupa ce selecteaza optiunea despre autobuze, ii vor fi afisate intr-un tabel datele cerute.

6. *Obtinerea datelor despre tramvaie*



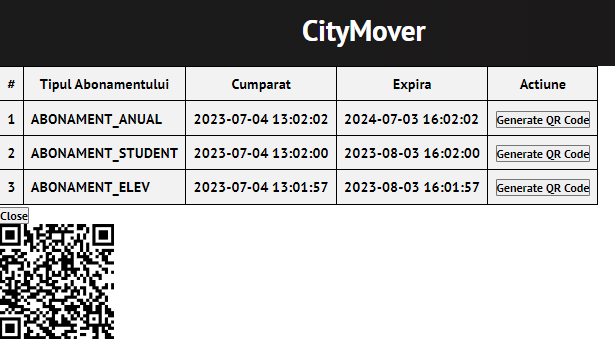
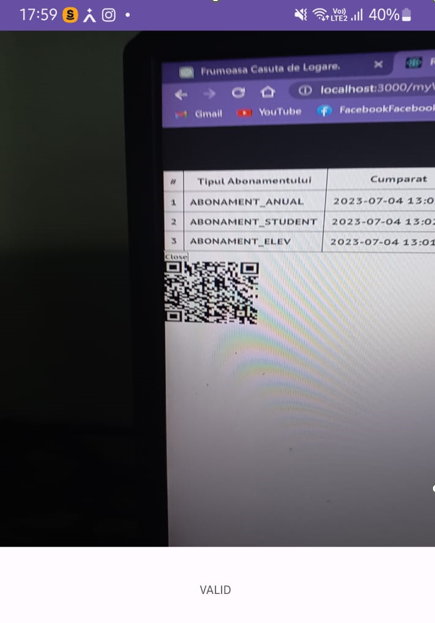
Aceasta functionalitate va putea fi accesata de catre un utilizator iar dupa ce selecteaza optiunea despre tramvaie, ii vor fi afisate intr-un tabel datele cerute.

7. Vizualizarea unui traseu dorit pe harta in functie de traseul ales.

8. Posibilitatea de a vedea o simulare in timp real a calatorilor, a traficului si statistici referiitoare la anumite rute

9.Validare bilet prin cod QR

Odată autentificat în contul personal, utilizatorul poate să genereze un cod QR pentru unul din biletele sau abonamentele sale. Acest pas a fost adaugat in vederea validarii unui bilet de catre un controlor pe durata utilizarii transportului public. În cazul în care codul QR nu este valid, aplicația va afișa un mesaj informativ de eroare. În caz contrar, aplicația va afișa un mesaj informativ de ca biletul este valid. De asemenea, utilizatorul poate inchide fereastra cu codul QR pentru un bilet si sa genereze pentru alt bilet.

10. Poate sa acceseze stiri sau noutati din pagina pornire.

/// de modificat stirile

**3.2. Diagrame UML**

**3.2.1. Diagrame use-case**

Mai jos sunt prezentate două diagrame de tip Use-Case, una pentru procesul de înregistrare/autentificare și una pentru momentul de după ce utilizatorul s-a autentificat cu succes în contul personal:

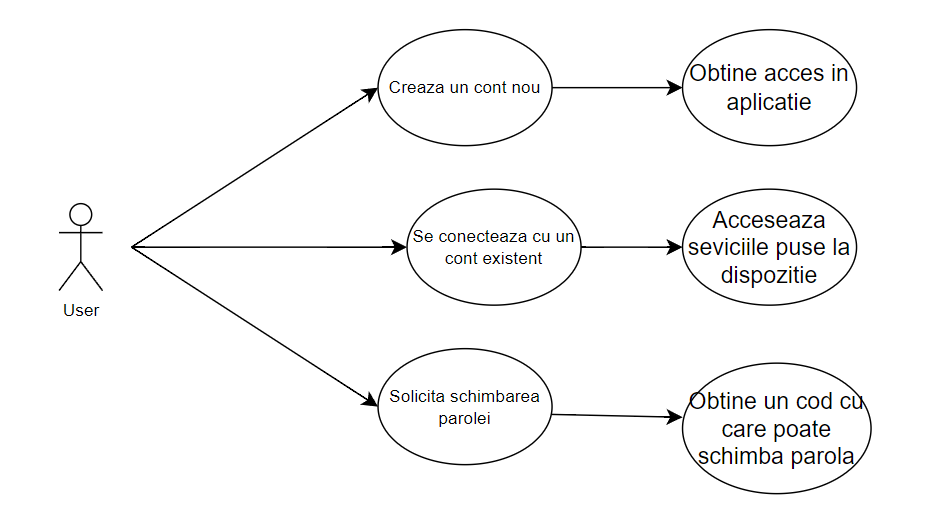


Figura 3.10: Diagrama use case aferentă procesului de înregistrare/autentificare în aplicație

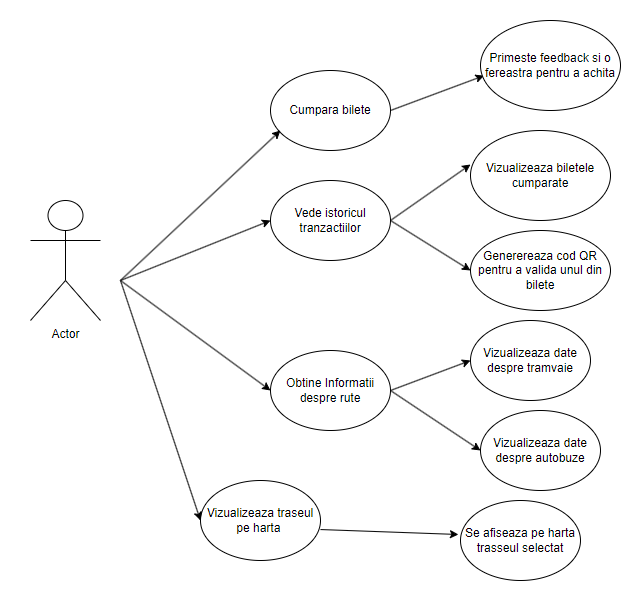


Figura 3.10: Diagrama use case aferntă momentului de post-înregistrare/autentificare

**3.2.3. Diagrama de secvență**

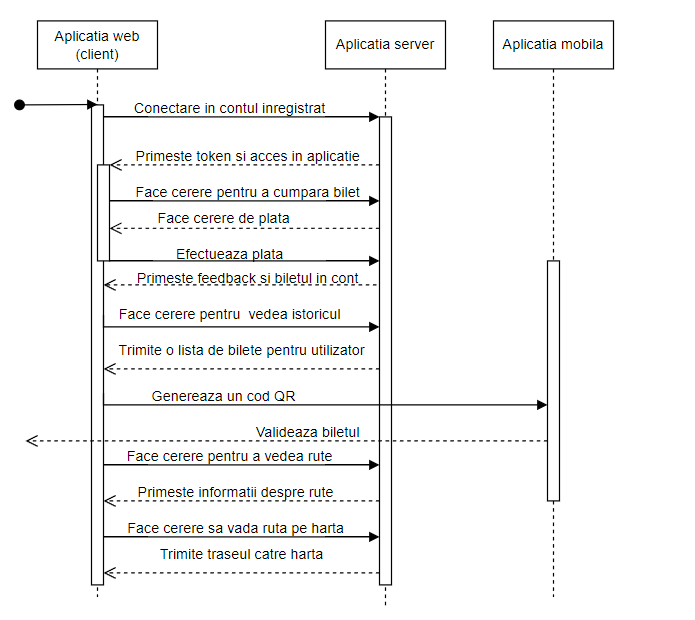


Figura 3.12: Diagrama de secvență

**Figura 3.12: Diagrama de activități**

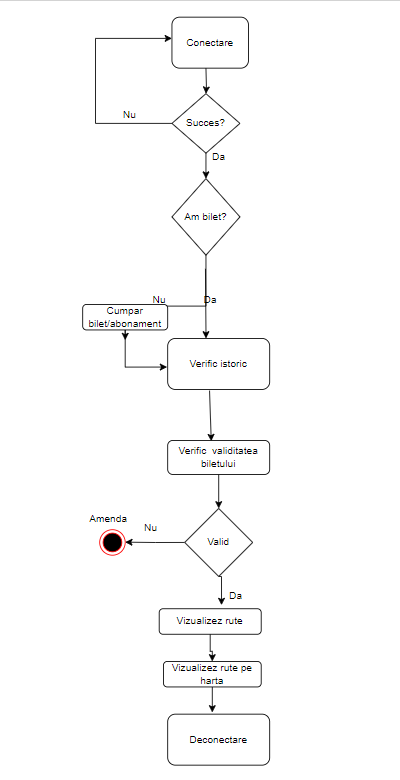
****

Figura 3.13: Diagrama de activități

**Capitolul 4. Implementarea sistemului**

**4.1. Implementarea aplicației server**

Aplicațiile web sunt aplicații software accesibile prin intermediul unui browser web. Ele rulează pe servere și oferă utilizatorilor o interfață grafică pentru a interacționa cu diverse funcționalități. Aplicațiile web au devenit extrem de populare și esențiale în era digitală de astăzi, datorită avantajelor lor, cum ar fi accesibilitatea globală, ușurința de actualizare și scalabilitatea.

O abordare tot mai utilizată în dezvoltarea aplicațiilor web este microserviciile, o arhitectură de proiectare software care împarte o aplicație mare în mai multe componente independente numite microservicii. Aceste microservicii sunt unități mici, autonome și specializate, care pot fi dezvoltate și implementate independent.

Microserviciile REST4 sunt o arhitectură comună folosită pentru a proiecta și implementa servicii web. REST4 este un stil arhitectural bazat pe standardele HTTP5 și se concentrează pe resurse și operații CRUD6 . Acesta se potrivește bine cu cerințele aplicațiilor web moderne și este foarte folosit în dezvoltarea de API-uri.

Fiecare cerere către un serviciu REST4 conține toate informațiile necesare pentru ca serverul să o înțeleagă. Serverul nu păstrează starea clienților între cereri. Aceasta face ca serviciile REST4 să fie scalabile și ușor de gestionat, deoarece nu necesită să mențină sesiuni pentru fiecare client.

Microserviciile REST4 sunt bazate pe resurse, care pot fi reprezentări ale entităților (de exemplu, utilizatori, produse, comenzi) din aplicație. Fiecare resursă este identificată printr-un URI8 , ceea ce permite clienților să acceseze și să manipuleze resursele utilizând URL-uri8.

REST4 utilizează adesea JSON9 sau XML10 ca format de date pentru cereri și răspunsuri. JSON9 este cel mai răspândit format datorită ușurinței de utilizare și a simplității sale.

Datorită structurii sale modulare, fiecare microserviciu poate fi scalat individual în funcție de nevoile aplicației, fără a afecta celelalte componente. Astfel, se poate asigura o scalabilitate mai flexibilă și eficientă a întregii aplicații.

Deoarece fiecare microserviciu este independent, actualizările și îmbunătățirile pot fi efectuate fără a afecta întreaga aplicație. În plus, în cazul unor probleme sau erori, acestea pot fi izolate și rezolvate mai rapid și mai eficient. Fiecare microserviciu poate fi dezvoltat folosind tehnologii specifice sau limbaje de programare diferite, adaptate scopului său.

Cu toate acestea, utilizarea microserviciilor poate aduce și provocări, cum ar fi complexitatea gestionării comunicării dintre servicii, asigurarea unei arhitecturi coerente și abordarea problemelor legate de tranzacții și consistență. O bună planificare, documentare și gestionare sunt esențiale pentru a beneficia în totalitate de potențialul oferit de microservicii și a evita capcanele asociate cu această arhitectură.

[[4]](#footnote-4)

Serviciul de utilizatori pe care l-am, implementat este responsabil pentru gestionarea informațiilor despre utilizatori, inclusiv înregistrarea, autentificarea și gestionarea drepturilor de acces. Acesta utiliza o baze de date SQL13, si anume distributia MySQL pentru a stoca informații despre utilizatori, inclusiv nume, prenume, nume de utilizator, adrese de email, parole criptate și alte detalii relevante. De asemenea, gestioneaza funcționalitătea de resetare a parolei.

Serviciul permite ca utilizatorii să se înregistreze în aplicație cu un cont nou, prin introducerea datelor personale. De asemenea, permite autentificarea utilizatorilor existenți prin intermediul numelui de utilizator si a parolei. Serviciul oferă posibilitatea utilizatorilor să își recupereze parola în cazul în care o uită. Aceasta poate fi realizată prin intermediul unui e-mail si a unui cod de confirmare.

Acest serviciu ruleaza in domeniul localhost pe portul 8091 si expune urmatoarele puncte finale, accesibile la nivelul clientului:

* */api/v1/service/users* – aceasta interogare are rolul de a oferi informatii referitoare despre utilizatorii care ssunt inregistrati in aplicatie;
* */api/v1/service/user/{username}* – aceasta interogare are rolul de a oferi informatii referitoare despre un utilizator, returand id acestuia;
* */api/v1/service/users/save* – acesta cerere are rolul de a salva in baza de date informatiile unui utilizator nou, aceasta metoda cere un corp de pesaj care sa contina informatiile cu care utilizatorul va fi salvat in baza de date
* */api/v1/service/users/role* – aceasta metoda are rolul de a insera unui utilizator un nou rol, aceast apel necesita ca parametru in URL, numele utilizatorul si in corpul cerereii de rolul pe care sa il adauge sub format JSON;

Securitatea aplicațiilor este o componentă critică pentru a proteja datele și resursele aplicației și pentru a asigura că utilizatorii pot utiliza aplicația într-un mod sigur și încredere

Servicul de securitate pe care l-am implementat are rolul de a asigura securitatea aplicației și a datelor utilizatorilor. Acesta gestioneaza autentificarea și autorizarea utilizatorilor, folosind tehnologii precum JWT, pentru a verifica identitatea utilizatorilor și pentru a le oferi acces la resursele protejate. De asemenea, poate implementa politici de securitate pentru protejarea aplicației de atacuri cibernetice, cum ar fi atacurile de tip XSS12 sau injecțiile SQL13.

Acest serviciu are in spate o baza de date SQL13 si anume distributia MySQL in care salvez tokens pentru fiecare utilizator care se conecteaza in aplicatie. Structura sub care se salveaza este id inregistrare, id-ul utilizatorului pentru care s-a generat token-ul si token-ul.

Acest serviciu ruleaza in domeniul localhost pe portul 8081 si chiar daca utilizatorul nu are acces direct catre punctele finale ale servicului, acesta expune urmatoarele metode:

* */api/v1/auth* */register* - această methoda servește la crearea unui nou utilizator în sistem odată ce corpul cererei a trecut de validare, un nou utilizator va fi creat în bază de date cu datele oferite
* */api/v1/auth* /*authenticate* – acest punct final necesita in corpul mesajului numele de utilizator si parola pentru a cripta folosind algoritmul de criptare SHA-256 și apoi transforma într-un șir de caractere folosind schema Base64 și genereaza un accessToken pe care il returneaza aplicației catre client, în corpul răspunsului pentru a fi folosit în interogările viitoare.
* */api/v1/auth* /*logout –* aceasta metodacauta in baza de date tokenul utilizatorului si il sterge devenind invalid pentru a face cereri in aplicatie cu acesta

Serviciul de bilete este o componentă cheie în cadrul aplicației mele de transport si genstionare al transportului public. Acesta se ocupă de gestionarea vânzării, istoricul tranzactiilor pentru un utilzator. Acesta stocheaza informații despre evenimente, tipurile de bilete disponibile, si prețurile. Aceasta foloseste in spate o baza de date SQL13 si anume distributia MySQL pentru a salva tranzactiile unui utilizator si detaliile despre biletele achizitionate.

Acest serviciu ruleaza in domeniul local host pe portul 8080 si expune urmatoarele rute care sunt accesibile la nivelul aplicatiei client:

* */api/v1/service/tickets –* Aceast apel in cadrul aplicatiei reurneaza toate biletele care sunt inregistrate in baza de data
* */api/v1/service/tickets/history/{user} –* Acest apel in cadrul aplicatiei returneaza informatii despre ultimele 5 achizitii ale utlizatorului si necesita in URL id-ul utilizatorului
* */api/v1/service*/*tickets/buy* – acest apel repezinta actiunea de cumparare a unui bilet de catre un utilizator. Aceasta cerere necesita un obiect de tipul “Bilet” si va returna eroare in cazul in care tranzactia nu se poate realiza

Serviciul de harti oferă funcționalități de afișare și gestionare a hărților în aplicație, cat si detalii despre rutele si autovehiculele disponibile. Acesta utiliza API-ul Opendata API pentru a obține date cartografice, detalii despre rutele si autovehiculele din reteau de transport public local și pentru a le afișa în interfața utilizatorului. Acest serviciu permite căutarea și vizualizarea locațiilor relevante și este integrat cu aplicatia. Acest servicu foloeste o baza de date NoSQL si anume distributia MongoDB pentru a stoca sub forma de documente datele referitoare la vehicule si rutele disponibile intr-un anumit oras.

Acest serviciu ruleaza in domeniul local host pe portul 8083 si ofera urmatoarele rute care sunt accesibile la nivelul aplicatiei client:

* */api/v1/service/harti/autovehicule/tramvaie –* cererea in acesta resura ofer a utilizatorului date referitoate la tramvaiele disponibile in cadrul companiei locale de transport public
* */api/v1/service/harti/autovehicule/autobuze –* cererea in acesta resura ofera utilizatorului date referitoate la tramvaiele disponibile in cadrul companiei locale de transport public
* */api/v1/service/harti/autovehicule –* acest punct final al servicului primeste o lista de date despre autovehicule disponibile la nivelul transportului public, pe care o parseaza si o pune intr- baza de date

Serviciul de simulare este un serviciu mai avansat, care preia datele de la acelasi API, Opendata, proceseaza informatiile primate si oferă o simulăre pentru un anumit scenariu, în funcție de nevoile aplicației. De exemplu, în cadrul unei aplicației de gestionare a traficului, serviciul de simulare simuleaza traseul pe care circula autobuzele sau tramvaiele, aparitia calatorilor, preluarea si transportul acestora la destinatii. În general, acest serviciu ar putea utiliza algoritmi mai compleși, dar am ales o varianta simpla.

Datele sunt primate separate, dar prin intermediul bazei de date NoSQL, MongoDB, stochez datele folosind colectii, procesez datele din aceste colectii intermediare pentru ca la finalul procesarii sa obtin o colectie care contine: numele traseului, numarul si lista de statii prin care trece traseul. Aceste date imi sunt necesare atat pentru a putea simula transportul in comun, cat si aparitia calatorilor.

**4.1.1. Implementarea aplicatiei client**

Node.js reprezintă un mediu JavaScript orientat către partea de server, bazat pe evenimente. Funcționează prin rularea codului JavaScript utilizând motorul V8 dezvoltat de Google, cunoscut pentru performanța sa în browserul Chrome. Node.js oferă un mediu de execuție pe partea de server, care compilează și rulează codul JavaScript la viteze foarte mari, datorită faptului că V8 compilează codul în cod mașină nativ, în loc să-l interpreteze ca bytecode.

Un avantaj major al Node.js este capacitatea sa de a rula codul asincron, datorită modelului excelent de evenimente al limbajului JavaScript. Astfel, operațiile intensive de tip I/O sunt executate asincron, permițând o scalabilitate ridicată la un număr mare de clienți. Deoarece aplicațiile bazate pe fire de execuție pot deveni dificil de gestionat și pot consuma o cantitate mare de memorie, Node.js folosește Event Loop (bucla eveniment) pentru a gestiona operațiile asincrone. Acest Event Loop gestionează operațiile asincrone și execută funcțiile de apel invers atunci când acestea se încheie, întorcând rezultatele către aplicație.

Utilizând Node.js, programatorii pot dezvolta aplicații complexe care pot scala la milioane de utilizatori. Datorită abilității de a gestiona operațiile asincrone și de a delega sarcinile buclei eveniment, dezvoltatorii pot concentra atenția pe funcționalitățile aplicației în loc să se preocupe de aspectele tehnice legate de gestionarea firelor de execuție.

Prin urmare, Node.js se dovedește a fi o unealtă puternică pentru programatori, oferind posibilitatea dezvoltării de aplicații eficiente, rapide și scalabile, ideale pentru gestionarea unui număr mare de utilizatori. Datorită popularității și cunoștințelor extinse de JavaScript în comunitatea de dezvoltatori, adoptarea Node.js se face mai ușor, fără a fi nevoie de învățarea unui limbaj nou pentru a dezvolta aplicații bazate pe această platformă.

React este o bibliotecă JavaScript open-source pentru construirea interfețelor utilizator (UI). A fost dezvoltată de către Facebook și este una dintre cele mai populare tehnologii folosite pentru dezvoltarea aplicațiilor web moderne. React folosește un model de programare declarativă, ceea ce înseamnă că dezvoltatorii se concentrează pe descrierea modului în care ar trebui să arate interfața în diferite stări, iar React se ocupă de actualizarea eficientă a interfeței în funcție de schimbările de stare.

Un concept central în React sunt componentele. Componentele sunt blocuri reutilizabile de cod care pot fi gândite ca "piese Lego" ale interfeței utilizator. Ele pot fi definite ca funcții sau clase, și fiecare componentă poate avea propriile sale proprietăți și stare (state).

Există două tipuri principale de componente în React:

* Componente funcționale: Sunt componente definite ca funcții JavaScript. Primesc proprietăți (props) ca argumente și returnează un element React, care descrie modul în care ar trebui să arate componenta în funcție de acele proprietăți.
* Componente bazate pe clasă: Sunt componente definite ca clase JavaScript. Acestea extind clasa React.Component și pot avea o stare internă. Metoda “render()” este utilizată pentru a returna elementul React reprezentând componenta.

In cadrul aplicatiei mele, pentru a fi usor de utilizat si pentru a obtine informatiile necesare utilizatorului am folosit o serie de componente React.

**Componenta “Acasa”**: aceasta este pagina principală a aplicație si conține informații de prezentare, elemente de navigare către celelalte pagini și este loc în care utilizatorii ajung după autentificare reusita.

**Componenta “Conectare”**: aceasta componenta este utilizata pentru autentificarea utilizatorilor. Aici isi vor introduce credențialele lor (numele de utilizator și parola) pentru a avea acces la funcționalitățile aplicației. De asemenea, aceasta contine si butoane cu rute catre alte doua componente, cea de recuperare parola si cea de inregistrare cont.

**Componenta “Recuperare”:** această componentă permite utilizatorilor să solicite o resetare a parolei în cazul în care au uitat-o. Conține un formular unde utilizatorii își introduc adresa de e-mail pentru a primi un link de resetare a parolei.

**Componenta “Navbar”**: aceasta componenta este bara de navigare a aplicației. Aici sunt incluse linkuri către celalalte pagini, numele aplicatiei si butoanele de autentificare/deconectare.

**Componenta “Inregistrare”:** această componentă permite utilizatorilor să se înregistreze pentru a crea un cont nou în aplicație. Conține un formular cu câmpuri pentru completarea datelor personale și de autentificare.  
 **Componenta “Portofel”:** aceasta componenta afișează portofelul utilizatorului, adică informații tranzacțiile recente, istoricul plăților și poate genera un cod QR pentru un bilet pentru a I se scana validitatea.

**Componenta “Bilete”:** această componentă afișează biletele sau tichetele de evenimente cumpărate de utilizator. Poate arăta detalii despre biletele puse la dispozitie utlizatorului.

1. Kanji reprezintă caractere chinezești logografice adoptate și folosite în sistemul modern de scriere japonez.

   2 În teoria codurilor, codurile Reed-Solomon sunt coduri non-binare, ciclice, corectoare de erori inventate de către Irving S. Reed și Gustave Solomon. [↑](#footnote-ref-1)
2. 3 Apache este o aplicație de tip server web ce a jucat un rol important în creșterea și dezvoltarea World Wide Web. [↑](#footnote-ref-2)
3. Bazele de date Not Only SQL sunt acelea care oferă un mecanism de stocare și recuperare a datelor care este modelat diferit față de relațiile tablare folosite în bazele de date relaționale.

   4 JavaScript Object Notation este un format standard deschis care utilizează text lizibil pentru a transmite obiecte de date formate din perechi atribut-valoare.

   5 Sharding este procesul prin care o bază de date sau un motor de căutare este partiționat orizontal. [↑](#footnote-ref-3)
4. REST - Representational State Transfer

   5 HTTP - Hypertext Transfer Protocol

   6 CRUD - Create, Read, Update, Delete

   7 API - Application Programming Interface

   8 URI - Uniform Resource Identifier

   9 JSON - JavaScript Object Notation

   10 XML - eXtensible Markup Language

   11 JWT - JSON Web Tokens

   12 XSS - Cross-Site Scripting

   13 SQL - Structured Query Language [↑](#footnote-ref-4)