1. Introducere

1.1 Descrierea problemei

În societatea modernă zona urbană devine una dintre provocările majore din punct de vedere al mobilității. Numărul mare de vehicule de pe străzi si traficul acestora duce la crearea unor blocaje permanente pe căile rutiere. Infrastructura devine tot mai suprasolicitată ca urmare a creșterii populației in marile orașe, acest lucru duce și la apariția unor consecințe nefavorabile pentru societatea, cum ar fi nivelul excesiv de gaze poluante. Serviciile de Car-sharing existente în orașele mari oferă o alternativă convenabilă cetățenilor, utilizatorii acestor servicii pot închiria un autoturism doar pentru perioada necesară deplasării. În felul acesta, dispare nevoie de achiziționare și întreținere a unui autoturism personal.

Chiar dacă în multe orașe ale lumii soluțiile existente de car-sharing s-au dovedit a fi benefice nu în toate cazurile au fost acceptate pentru a se stabili pe piață. Uneori infrastructura nu este destul de dezvoltată alteori prețurile impuse sunt ridicate prea mult, astfel incât utilizatorii nu mai văd un beneficiu în serviciul dat.

Prin crearea propriei soluții îmi propun rezolvarea problemelor majore al aplicatiilor existente deja pe piața globală prin utilizarea unor algoritmi și tehnologii moderne.

1.2 Descrierea obiectivelor lucrării

Proiectul își propunse să analizeze mai întâi problemele întâlnite de soluțiile deja existente pe piață și să implenteze o soluție inovatoare care va aborda într-un mod diferit provocările ce n-au lăsat alte companii să-și dezvolte potențialul.

Principalul obiectiv este crearea unei aplicații mobile intuitive, care să permită utilizatorilor parcurgerea a cât mai puțini pași pentru închirierea unui autoturism. Cred că unul dintre puntele forte al astfel de aplicații ce ar putea atrage clienții este simplitatea interfeței, deoarece orice om și-ar dori să petreacă cât mai puțin timp în interfața aplicației până a începe cursa efectivă.

Un alt obiectiv pe care trebuie să-l atingă proiectul este crearea unui sistem de tarifare prin care să putem sarisface nevoile diverse ale utilizatorilor. Tarifurile vor fi separate în câteva opțiuni: pe minut, pe ora, pe zi. Ideea unor astfel variante de tarifare este simplă, cu cât mai mare este perioada de timp pentru care închiriezi automobilul cu atât mai mică va fi plata lei/sec. Printr-o astfel de metodă de tarifare putem jongla între diversile nevoi ale clienților, putem satisface atât clienții care au nevoie de o cursă scurtă pentru a ajunge în câteva minute din punctul A în punctul B, cât și pe cei care au nevoie de un autovehicul pentru un timp mai îndelungat cum ar fi câteva ore sau chiar câteva zile.

De asemenea aplicația își propune să contribuie la reducerea impactului negativ asupra mediului înconjurător, promovând autovehiculele electrice și oferind utilizatorilor informații despre stațiile de încărcare disponibile.

Un obiectiv ce consider că ar fi apreciat de către utilizatori este implementarea unei sistem protejat de autentificare și în general protecția datelor consumatorului. Pentru logarea în

siguranță va fi implementat Firebase Authentication care permite menținerea în siguranță a tuturor datelor utilizatorilor.

1.3 Posibile avanteja

Prin aplicația propusă încerc să aduc o serie de înbunătățiri soluțiilor deja existente pe piață de Car-sharing, prin integrarea unor funcționalități moderne îmi propun să ofer o experiență cât mai plăcută și eficientă pentru utilizatorii noștri eliminând problemele frecvente ale serviciile actuale.

În mod normal pentru a închiria un autoturism prin serviciul de car-sharing este necesar ca clientul să se deplaseze către un autovehicul disponibil și să-l deblocheze prin intermediul aplicației de pe telefon. În soluția dată propun implementarea unui nou serviciu, există cazuri în care un utilizator vrea să-ți planifice o cursă dar descoperă că nu are niciun vehicul prin apropiere pe care l-ar putea rezerva sau cazuri în care utilizatorul are anumite preferinte pentru un vehicul, pentru aceste situații dar și altele asemănătoare se va potrivi noul serviciu al aplicației și anume livrarea unui vehicul. Utilizatorul poate selecta modelul de masină dorit sau aplica un set de filtre conform cărora va fi ales un autoturism. În urma selecției de către utilizator, va fi inițializata a doua etapă din proces si anume livrarea. Autoturismul este condus de către un angajat am companiei până la adresa utilizatorului unde va parca mașina în siguranță. În urma încheierii acestui proces clientul va primi o notificare referitor la disponibilitatea vehiculul la locația aleasă. În acest fel creștem nivelul flexibilității serviciilor oferite, fiind un punct forte care poate atrage mai mulți clienți față de concurenți.

1.4 Importanță și actualitate

Astăzi, suprasolicitarea infrastructurii urbane și emisiile de CO2 in cantități enorme au devenit unele dinte probelemele cele mai discutate. Mobilitatea urbană a devenit unul dintre subiectele aprinse ce pune autoritățile în situația de a cauta alternative care pot fi propuse cetățenilor în locul deținerii private ale autoturismelor. În contextul dat, car-sharing-ul este o soluție avantajoasă care poate diminua probleme apărute în societate în acest sens. Acest serviciu oferă posibilitatea utilizatorilor săi de a folosi un autoturism doar atunci când au nevoie, eliberându-i de costurile și responsabilitățile ce apar odată cu deținerea unei mașini personale. La nivel global tot mai multe orașe implementează politici pentru descurajarea utilizării mașinilor personale, promovând în schimb alte tipuri de mobilitate. Acest lucru oferă car-sharing-ului o plajă de dezvoltare majoră unde companiile din acest domeniu își pot pune la vedere avantajele.

2. Car Sharing – Problema abordată de lucrare

2.1 Contextul general al mobilității urbane

Mobilitatea urbană reprezintă în prezent una dintre cele mai mari provocări ale societății moderne. Odată cu creșterea rapidă a populației din mediul urban și cu extinderea continuă a centrelor urbane, ne confruntăm cu o presiune din ce în ce mai mare asupra infrastructurii de transport. Tot mai multe persoane aleg să dețină un autoturism personal, lucru care duce

la un număr foarte mare de vehicule în circulație pe străzi, în special în orele de vârf. Această tendință are efecte directe asupra calității vieții în orașe, generând trafic intens, blocaje frecvente, poluare atmosferică accentuată, lipsa locurilor de parcare și pierderi de timp semnificative pentru cetățeni. În plus, consumul crescut de combustibil fosil contribuie la emisiile ridicate de gaze cu efect de seră, agravând problemele de mediu.

Pentru a răspunde acestor provocări, autoritățile locale, alături de companiile din domeniul tehnologic și de transport, au început să caute soluții alternative care să permită cetățenilor să se deplaseze eficient și ecologic în orașe. Una dintre aceste soluții este reprezentată de serviciile de tip car sharing, care oferă un model diferit de utilizare a autovehiculelor, bazat pe partajare și utilizare la cerere. În esență, car sharing-ul presupune accesul la o flotă de autoturisme comune, disponibile pentru închiriere pe termen scurt, fără a fi necesară deținerea permanentă a unui vehicul personal.

2.2 Conceptul de car sharing

Serviciile de car sharing oferă multiple beneficii atât pentru utilizatori, cât și pentru societate în ansamblu. Utilizatorii au avantajul de a plăti doar pentru timpul efectiv de utilizare al vehiculului, fără a fi împovărați de costurile asociate deținerii unei mașini, cum ar fi întreținerea, reviziile, impozitele, asigurările sau combustibilul. În același timp, aceștia beneficiază de flexibilitate, având posibilitatea de a alege vehiculul dorit dintr-o anumită locație și de a-l folosi doar pentru perioada în care au nevoie.

La nivel social, car sharing-ul contribuie la reducerea numărului total de mașini din trafic, ceea ce se traduce printr-o scădere a nivelului de congestionare rutieră și a poluării. De asemenea, printr-o utilizare mai eficientă a resurselor, se promovează un comportament responsabil în rândul utilizatorilor, încurajându-se utilizarea în comun a mijloacelor de transport. Car sharing-ul este, astfel, o soluție sustenabilă care se aliniază cu principiile mobilității inteligente și ecologice.

2.3 Problemele serviciilor de car sharing existente

Cu toate acestea, aplicațiile și platformele de car sharing existente pe piață, precum Spark, FreeNow sau Bolt Drive, nu sunt lipsite de deficiențe. În primul rând, multe dintre aceste aplicații au interfețe prea complexe sau greoaie, ceea ce creează dificultăți în utilizare, în special pentru persoanele mai puțin familiarizate cu tehnologia. În loc să simplifice procesul de închiriere, aceste interfețe pot deveni o barieră în calea utilizatorilor.

Un alt aspect problematic este lipsa flexibilității în ceea ce privește sistemul de tarifare. Multe aplicații impun tarife fixe, care nu se adaptează la nevoile diferite ale utilizatorilor. Astfel, cei care doresc să folosească un vehicul pentru doar câteva minute sunt obligați să plătească un tarif pentru o oră întreagă, în timp ce alții care vor să închirieze o mașină pentru o zi întreagă nu beneficiază de reduceri semnificative.

În plus, actualizarea poziției vehiculelor pe hartă este adesea întârziată sau inexactă, ceea ce duce la confuzii și întârzieri în procesul de rezervare. Un alt inconvenient major este disponibilitatea limitată a vehiculelor, mai ales în zonele periferice ale orașelor, unde utilizatorii nu găsesc mereu un vehicul la îndemână. De asemenea, lipsa unui sistem

centralizat de gestionare a curselor și a istoricului utilizatorilor afectează experiența generală, limitând transparența și controlul asupra datelor personale și activităților anterioare.

2.4 Analiza pieței actuale

În prezent, piața car sharing-ului este dominată de câteva aplicații cunoscute, fiecare cu specificul său. De exemplu, Spark este o aplicație care oferă exclusiv vehicule electrice și operează în orașe ca Sofia, Vilnius, Plovdiv și Kaunas. Pe de altă parte, FreeNow oferă o gamă mai variată de vehicule (electrice, hibride, termice) și este disponibilă în orașe mari precum Berlin, Paris, Milano, Roma sau Madrid. Aceste aplicații au reușit să capteze atenția utilizatorilor din anumite regiuni, însă nu sunt disponibile în toate orașele sau nu sunt adaptate specificului local.

Pe piața din România, serviciile de car sharing nu sunt încă la un nivel avansat, ceea ce oferă un potențial de dezvoltare semnificativ. Lipsa unei aplicații special concepute pentru piața românească, cu o interfață intuitivă, tarifare flexibilă și funcționalități moderne, creează o oportunitate importantă pentru lansarea unei soluții noi.

2.5 Nevoia unei soluții noi

Având în vedere problemele menționate anterior, devine evidentă nevoia unei aplicații noi care să răspundă mai bine nevoilor utilizatorilor și să aducă inovații relevante în domeniu. O astfel de aplicație ar trebui să ofere o interfață ușor de utilizat, cu un proces de închiriere simplificat și rapid, care să reducă numărul de pași necesari de la deschiderea aplicației până la pornirea cursei.

Totodată, sistemul de tarifare ar trebui să fie flexibil, cu opțiuni de plată per minut, oră sau zi, astfel încât utilizatorii să poată alege varianta cea mai convenabilă pentru nevoile lor. Reducerile progresive pentru perioade mai lungi de utilizare ar reprezenta un avantaj semnificativ, care ar încuraja folosirea constantă a aplicației.

Un element de noutate propus de aplicația dezvoltată este serviciul de livrare a vehiculului la adresa utilizatorului. În cazul în care un client nu găsește un vehicul în apropiere sau are preferințe pentru un anumit model, aplicația permite selectarea unui vehicul și solicitarea livrării acestuia la locația aleasă. Această funcționalitate adaugă un nivel de confort suplimentar și crește atractivitatea aplicației în fața competitorilor.

2.6 Descrierea problemelor concrete abordate

Lucrarea de față abordează în mod direct o serie de probleme specifice identificate în aplicațiile existente. În primul rând, ea își propune să compenseze lipsa unei aplicații car sharing moderne și localizate pentru piața românească. În al doilea rând, urmărește implementarea unui sistem de tarifare flexibil, adaptat nevoilor utilizatorilor și orientat către eficiență financiară.

Un alt obiectiv important este integrarea serviciului de livrare personalizată a vehiculului, o funcționalitate inovatoare care oferă un avantaj competitiv major. În plus, aplicația include

un sistem de notificări și de gestionare a istoricului curselor, ceea ce contribuie la transparență și la o experiență îmbunătățită a utilizatorului.

În ceea ce privește siguranța datelor personale, aplicația implementează un sistem de autentificare bazat pe Firebase, care garantează protecția informațiilor și a conturilor utilizatorilor. Integrarea între o bază de date relațională (MySQL) și Firebase permite gestionarea eficientă a datelor statice și dinamice, inclusiv locația în timp real a vehiculelor.

2.7 Obiectivele concrete ale aplicației dezvoltate

Printre obiectivele aplicației dezvoltate se numără crearea unei interfețe grafice intuitive, capabilă să ghideze utilizatorii rapid și eficient în procesul de închiriere. În plus, aplicația integrează hărți interactive prin Google Maps, oferind informații în timp real despre locația vehiculelor disponibile.

Autentificarea utilizatorilor se realizează prin Firebase Authentication, ceea ce garantează un proces sigur și eficient. Sistemul de tarifare permite selectarea tipului de plată dorit (pe minut, oră, zi), iar toate informațiile despre curse, vehicule, documente și tranzacții sunt salvate în baza de date MySQL.

Utilizatorii au acces la istoricul curselor și tranzacțiilor, iar aplicația oferă posibilitatea încărcării soldului în portofelul digital. Astfel, se creează un ecosistem complet, care susține nevoile utilizatorilor de la înregistrare până la finalizarea unei curse. Mai jos urmează câteva screenshot-uri ce demonstrează unele dintre abilitățile aplicației.

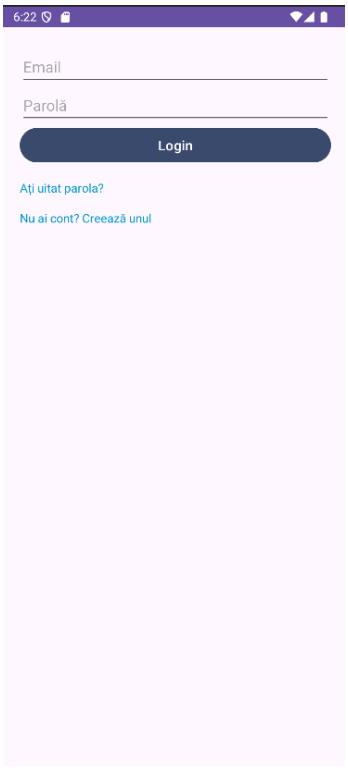

Figura 2.1: *Ecran de logare*

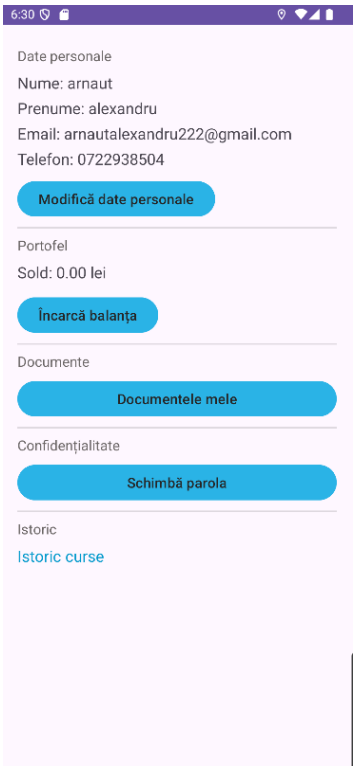

Figura 2.2: *Profilul utilizatorului*

3. Tehnologii și Metode Utilizate

3.1 Android Studio și Java pentru dezvoltarea aplicației mobile

Pentru implementarea aplicației mobile, a fost utilizat mediul de dezvoltare integrat Android Studio, care reprezintă soluția oficială oferită de Google pentru dezvoltarea aplicațiilor Android. Android Studio oferă unelte complete pentru proiectarea, codarea, testarea și depanarea aplicațiilor, având integrat un sistem puternic de construire a proiectului bazat pe Gradle. Alegerea acestei platforme a fost determinată de suportul extins pentru limbajul Java, dar și de posibilitatea de a integra rapid biblioteci externe și servicii cloud precum Firebase sau API-uri de localizare. Codul sursă al aplicației a fost scris în întregime în Java, un limbaj de programare orientat pe obiecte, robust și foarte utilizat în dezvoltarea de aplicații Android. Utilizarea Java a permis definirea clară a claselor și metodelor responsabile pentru fiecare funcționalitate, de la autentificare și gestionarea sesiunii utilizatorului, până la afișarea vehiculelor pe hartă, calculul prețului cursei și stocarea datelor în baza de date.

3.2 Firebase Authentication pentru gestionarea conturilor de utilizator

Una dintre cele mai importante componente ale aplicației este sistemul de autentificare, care trebuie să asigure atât securitatea datelor, cât și o experiență fluidă pentru utilizator. Pentru această funcționalitate a fost ales serviciul Firebase Authentication, oferit de Google, care permite autentificarea rapidă a utilizatorilor prin email și parolă, dar și gestionarea conturilor într-un mod securizat. Firebase Authentication oferă protecție împotriva atacurilor de tip phishing, controlează validarea sesiunii și gestionează autentificarea cu tokenuri de acces securizate. Prin intermediul acestui serviciu, aplicația asigură integritatea datelor utilizatorilor și accesul restricționat doar la informațiile proprii. Utilizarea Firebase simplifică procesul de înregistrare și logare, asigurând o integrare ușoară cu restul componentelor aplicației mobile.

3.3 Google Maps API și localizarea vehiculului

Pentru a permite afișarea vehiculelor disponibile pe hartă și urmărirea în timp real a locației utilizatorului în timpul cursei, aplicația a fost integrată cu Google Maps API. Această bibliotecă permite încorporarea unei hărți interactive în interfața aplicației, oferind multiple funcționalități precum marker-e personalizate, rutare, poziționare GPS și obținerea coordonatelor geografice exacte. Integrarea API-ului a fost realizată prin intermediul unei chei unice generate din Google Cloud Console și asociate cu contul Firebase. Harta permite afișarea în timp real a poziției curente a utilizatorului și a vehiculelor disponibile în apropiere. Locația este actualizată periodic atât pe telefon, cât și în baza de date, astfel încât sistemul backend să poată oferi sugestii relevante în funcție de proximitate. Google Maps este esențial în contextul aplicațiilor de mobilitate, iar utilizarea acestui API aduce un plus major de interactivitate și claritate vizuală.

3.4 MySQL pentru stocarea datelor persistente

Pentru gestionarea datelor care trebuie păstrate permanent, cum ar fi informațiile despre vehicule, curse, carduri bancare, tranzacții sau istoricul utilizatorului, a fost utilizat un sistem de baze de date relaționale, respectiv MySQL. Această alegere a fost motivată de fiabilitatea

și maturitatea MySQL ca sistem de gestiune a bazelor de date, dar și de suportul foarte bun oferit de Spring Boot pentru interacțiunea cu tabele relaționale. Structura bazei de date a fost proiectată pornind de la cerințele aplicației: fiecare entitate (de exemplu vehicul, cursă, tranzacție) este reprezentată printr-un tabel dedicat, cu relații bine definite între acestea. Tabelele conțin câmpuri relevante, indexări pentru căutare rapidă și chei primare pentru identificarea unică a fiecărui element. Accesul la bază se face prin intermediul unor repository-uri definite în backend, care permit inserarea, citirea, actualizarea și ștergerea datelor în mod controlat.

3.5 Firebase Realtime Database pentru date dinamice

Pe lângă MySQL, aplicația utilizează și serviciul Firebase Realtime Database, pentru acele date care trebuie actualizate în timp real, cum ar fi locația vehiculului sau statusul acestuia (disponibil, rezervat, în cursă). Firebase Realtime Database oferă un mecanism asincron prin care datele sunt transmise între dispozitive fără întârzieri semnificative, fiind extrem de utilă în aplicații mobile care necesită actualizări rapide și precise. De exemplu, în timpul unei curse active, aplicația transmite periodic către Firebase poziția curentă a utilizatorului, care este ulterior afișată pe hartă, iar serverul poate lua decizii în funcție de această locație. Utilizarea Firebase Realtime Database alături de MySQL permite separarea logică între datele volatile și cele persistente, maximizând performanța aplicației și optimizând consumul de resurse.

3.6 Spring Boot pentru implementarea backend-ului

Pentru a asigura comunicarea dintre aplicația Android și baza de date, a fost dezvoltat un backend folosind framework-ul Spring Boot. Acest framework Java permite crearea rapidă de aplicații web RESTful, oferind o arhitectură scalabilă și modulară. Backend-ul gestionează solicitările trimise de aplicația mobilă, validează datele și interacționează cu baza de date MySQL pentru salvarea sau extragerea informațiilor. Fiecare resursă (vehicul, cursă, card, tranzacție etc.) are asociat un controller dedicat care răspunde la cererile HTTP (GET, POST, PUT, DELETE). Utilizarea Spring Boot a permis aplicarea principiului separării responsabilităților, asigurând astfel un cod curat, testabil și ușor de extins. Pe lângă funcționalitățile de bază, backend-ul implementează și logica de calcul al prețului în funcție de durata cursei, verificarea soldului disponibil și actualizarea acestuia la finalul cursei, precum și gestionarea documentelor personale ale utilizatorilor.

3.7 Algoritmi și metode utilizate în logica aplicației

Deși aplicația nu rezolvă o problemă clasică de optimizare, logica sa internă presupune folosirea unor metode și algoritmi practici, adaptați scopului propus. Un exemplu relevant este algoritmul de calcul al prețului în funcție de durata reală a cursei. Acesta pornește de la tariful ales inițial de utilizator (pe minut, oră sau zi) și verifică la fiecare secundă dacă durata efectivă a cursei a fost depășită. În caz afirmativ, prețul se actualizează corespunzător, menținând tariful unitar. Un alt exemplu este mecanismul de verificare a soldului din portofelul digital. În timpul cursei, aplicația compară soldul disponibil cu prețul acumulat și, dacă soldul scade sub o anumită valoare critică (de exemplu 0.2 lei), cursa este încheiată automat pentru a evita pierderile. În plus, este implementat un sistem de notificări

automate care avertizează utilizatorul dacă soldul scade sub 5 lei, oferindu-i opțiunea de a încărca imediat contul printr-o tranzacție.

Pentru încărcarea soldului, aplicația utilizează un flux logic care include selectarea cardului salvat, introducerea sumei și transmiterea unei cereri către backend, unde este înregistrată tranzacția și actualizat soldul în Firebase. În același timp, backend-ul criptează datele cardului și le stochează securizat în MySQL, iar decriptarea acestora se face doar la afișare, pentru a păstra conformitatea cu bunele practici de securitate.

3.8 Metodologia de dezvoltare utilizată

În ceea ce privește metodologia de dezvoltare, proiectul a fost realizat într-un mod iterativ, pornind de la o versiune minim viabilă (MVP), urmată de adăugarea treptată a funcționalităților suplimentare. Acest model de dezvoltare incrementală a permis testarea fiecărei componente separat, precum și validarea comportamentului aplicației în diverse scenarii. Inițial s-a construit structura principală a aplicației mobile, cu sistemul de autentificare și afișarea vehiculelor pe hartă, urmate de dezvoltarea componentelor backend pentru salvarea datelor. Ulterior au fost adăugate funcționalitățile de tarifare, portofel digital, încărcare sold, istoric al curselor și al tranzacțiilor, precum și componenta de fotografiere a vehiculului înainte de cursă.

Această abordare modulară a asigurat o bună organizare a codului, posibilitatea testării unitare și o integrare ușoară a componentelor. De asemenea, folosirea unor biblioteci externe bine documentate a contribuit la accelerarea procesului de dezvoltare, reducând semnificativ timpul necesar pentru implementarea unor funcții precum încărcarea imaginilor, prelucrarea locației sau gestionarea notificărilor. S-a urmărit pe tot parcursul proiectului o abordare orientată pe utilizator, punând accent pe ușurința în utilizare, simplitatea interfeței și feedback-ul vizual rapid.

3.9 Alte tehnologii suport utilizate

Pe lângă tehnologiile principale utilizate în dezvoltarea aplicației de car sharing, au fost integrate și o serie de biblioteci, unelte și componente software auxiliare care au contribuit în mod esențial la completarea funcționalităților, optimizarea performanței și garantarea unei experiențe cât mai bune pentru utilizator. Aceste tehnologii, deși nu sunt neapărat în prim-planul arhitecturii, joacă un rol important în realizarea unui produs software complet și stabil.

Una dintre bibliotecile cele mai utilizate a fost Retrofit, un client HTTP tip REST dezvoltat special pentru Android. Retrofit permite trimiterea cererilor HTTP către server într-un mod simplificat, oferind o abstracție elegantă față de nivelul de rețea. Prin definirea interfețelor de tip API și utilizarea adnotărilor, se pot trimite cereri GET, POST, PUT sau DELETE către backend fără a scrie cod repetitiv. Avantajul major constă în suportul nativ pentru operare asincronă, ceea ce înseamnă că aplicația poate trimite sau primi date fără a bloca interfața utilizatorului. Retrofit este de asemenea compatibil cu biblioteci de conversie precum Gson sau Moshi, care sunt folosite pentru serializarea și deserializarea automată a datelor JSON.

Pentru schimbul de date între aplicația mobilă și serverul backend a fost folosită biblioteca Gson, dezvoltată de Google. Gson oferă mecanisme rapide și eficiente pentru conversia obiectelor Java în format JSON și invers. Această funcționalitate este esențială în contextul în care aplicația Android comunică permanent cu serverul pentru a transmite informații despre utilizatori, vehicule, curse, tranzacții și solduri. Prin utilizarea Gson, complexitatea codului de comunicare este redusă, iar datele sunt structurate într-un mod ușor de procesat la ambele capete.

Pentru procesarea cererilor de plată și stocarea informațiilor despre carduri s-au utilizat funcționalitățile criptografice puse la dispoziție de librăriile Java de criptare simetrică, în special algoritmul AES (Advanced Encryption Standard). Acest algoritm a fost implementat pentru a cripta informațiile sensibile legate de cardurile bancare înainte de a fi trimise către backend. La nivelul serverului, aceste date sunt stocate într-o formă criptată în baza de date MySQL, iar la afișare sunt decriptate local pentru a proteja securitatea utilizatorului. Implementarea criptării AES presupune generarea unei chei de criptare sigură, stocată local într-un mod protejat sau derivată dintr-un element de autentificare.

În ceea ce privește stocarea și trimiterea de imagini (fotografii ale vehiculului înainte de cursă, capturi de documente precum permisul auto), s-a adoptat o metodă de codificare în format Base64, care permite transformarea fișierelor binare în șiruri de caractere ce pot fi transmise prin JSON. Aceste șiruri sunt apoi decodificate și salvate ca BLOB-uri (Binary Large Objects) în baza de date MySQL. Această metodă, deși consumă mai mult spațiu decât formatele binare native, oferă o compatibilitate ridicată cu sistemele de transport de date JSON și permite integrarea facilă în modelul deja existent.

Pentru afișarea listelor și a elementelor interactive din aplicație, s-a utilizat extensiv componenta RecyclerView, parte din biblioteca AndroidX. RecyclerView oferă o soluție performantă și flexibilă pentru afișarea colecțiilor mari de date, cum ar fi istoricul tranzacțiilor, lista cardurilor salvate sau curselor efectuate. Această componentă permite reciclarea eficientă a elementelor UI pentru a reduce consumul de memorie și a crește performanța, fiind esențială într-o aplicație care gestionează date dinamice și multiple interacțiuni cu utilizatorul.

Totodată, pentru trimiterea notificărilor către utilizatori, aplicația este pregătită pentru integrarea cu Firebase Cloud Messaging (FCM), un serviciu care permite transmiterea de notificări push pe baza unor evenimente definite în aplicație. Deși integrarea completă a FCM este planificată pentru o versiune viitoare, arhitectura aplicației a fost concepută astfel încât să permită cu ușurință adăugarea acestei funcționalități, oferind astfel un canal direct de comunicare cu utilizatorii pentru oferte, alerte sau informații critice legate de contul lor.

De asemenea, pentru validarea câmpurilor de introducere (input validation), s-au utilizat metode Java locale, dar și clase helper care asigură că datele introduse de utilizatori respectă formatele cerute – de exemplu, verificarea structurii unei adrese de email, a unui număr de telefon sau a formatului unui număr de card bancar. Astfel, se evită trimiterea către server a unor date invalide, reducând numărul de erori și îmbunătățind experiența generală.

Nu în ultimul rând, pentru gestionarea permisiunilor Android (acces la locație, cameră, stocare), s-au utilizat metodele moderne din AndroidX și sistemul de permisiuni runtime, care permite solicitarea acestor drepturi doar în momentul în care sunt efectiv necesare, respectând bunele practici impuse de versiunile recente ale Android.

Prin combinarea acestor tehnologii suport cu cele fundamentale prezentate anterior, aplicația reușește să livreze o experiență completă, sigură și performantă, fiind pregătită pentru extindere și adaptare la cerințele reale ale pieței și ale utilizatorilor.

4. Arhitectura soluției

4.1 Introducere generală

Arhitectura soluției dezvoltate este rezultatul unei planificări riguroase care a avut ca scop crearea unei aplicații mobile modulare, ușor de utilizat și scalabile, capabilă să deservească un număr ridicat de utilizatori simultan și să răspundă cerințelor tot mai complexe ale serviciilor de tip car sharing. Aplicația a fost concepută pe baza principiilor arhitecturale moderne, urmărind o separare clară a responsabilităților între componente, astfel încât fiecare modul să poată fi dezvoltat, testat și menținut independent.

4.2 Descrierea generală a arhitecturii

Soluția este organizată în trei niveluri majore de abstractizare: nivelul de interfață cu utilizatorul, nivelul logicii aplicației și nivelul de stocare a datelor. Interacțiunea dintre aceste niveluri este asigurată prin intermediul unor interfețe bine definite, care permit schimbul de date și comenzi între module. Arhitectura modulară permite o dezvoltare incrementală a aplicației, fiecare funcționalitate fiind encapsulată într-un modul propriu, capabil să comunice cu celelalte prin metode bine stabilite.

4.3 Componentele principale ale aplicației

Din punct de vedere logic, aplicația este alcătuită din mai multe componente funcționale, fiecare responsabilă de un set de activități specifice. Prima componentă este modulul de autentificare și gestionare a contului de utilizator. Acesta se ocupă cu procesul de creare a contului, autentificare, validare a identității și menținerea sesiunii active.

Un alt modul important este cel de listare și selecție a vehiculului. Acesta este responsabil cu afișarea vehiculelor disponibile în proximitate, filtrarea acestora după criterii precum distanța, tipul vehiculului, disponibilitate sau preferințele utilizatorului, precum și inițierea rezervării.

Modulul de derulare a cursei reprezintă un nucleu funcțional al aplicației. El gestionează temporizarea cursei, actualizarea poziției utilizatorului și a vehiculului pe hartă, calcularea dinamică a prețului în funcție de durata efectivă a cursei și monitorizarea soldului disponibil. Acest modul este interconectat cu modulul de portofel digital, care permite gestionarea soldului, a cardurilor bancare asociate și a tranzacțiilor realizate de utilizator.

Modulul de istoric și analiză oferă utilizatorului posibilitatea de a consulta cursele anterioare, detaliile fiecărei curse (durată, preț total, vehicul folosit), precum și istoricul financiar, incluzând încărcările de sold și plățile efectuate. Acest modul contribuie la transparența serviciului oferit și la fidelizarea utilizatorilor prin oferirea unui control complet asupra activității desfășurate în aplicație.

4.4 Diagrama de componente

(diagramă de componente va fi inserată aici, de tip UML sau logic conceptual)

Diagrama de componente oferă o privire de ansamblu asupra modului în care principalele module ale aplicației interacționează între ele. Fiecare componentă este ilustrată ca un bloc independent, cu conexiuni directe către componentele de care depinde sau cu care colaborează. Se pot distinge clar interfețele de comunicare, iar nivelul de izolare al componentelor facilitează testarea modulară și mentenanța eficientă.

4.5 Fluxul de utilizare: Inițierea și finalizarea unei curse

Pentru a ilustra logica aplicației, se poate analiza fluxul de utilizare specific unei funcționalități cheie: inițierea și finalizarea unei curse. Utilizatorul pornește de la ecranul principal, unde este afișată harta în timp real cu toate vehiculele disponibile din zona sa. După selectarea unui vehicul, utilizatorul accesează un ecran cu detalii precum model, autonomie, tarif, disponibilitate. Confirmarea rezervării duce la pornirea cronometrului și a sistemului de urmărire a locației.

Pe durata cursei, aplicația actualizează periodic poziția utilizatorului și prețul acumulat. Oricând, utilizatorul poate accesa meniul lateral pentru a consulta detalii despre cursă, a reîncărca soldul sau a verifica informațiile despre vehicul. Cursa se finalizează prin accesarea butonului de oprire, care declanșează oprirea cronometrului, calculul final al prețului și înregistrarea cursei în baza de date.

Fluxul este conceput pentru a fi simplu, intuitiv și eficient, necesitând un număr redus de acțiuni din partea utilizatorului. Diagrama de flux a acestei funcționalități va fi inserată pentru a reprezenta vizual etapele menționate mai sus.

4.6 Organizarea interfeței și comunicarea între module

Aplicația are o interfață centrată pe utilizator, minimalistă și ghidată de principiile UX/UI moderne. Fiecare ecran este dedicat unei singure sarcini, iar navigarea între ecrane se face prin intermediul unui meniu lateral și al unor butoane de acțiune plasate intuitiv. Modulele comunică prin mesaje și evenimente interne, iar datele se transferă între ecrane prin obiecte serializabile.

Separarea funcțională a componentelor permite reutilizarea codului în mai multe părți ale aplicației, precum și integrarea facilă a unor funcții suplimentare în viitor. Această arhitectură modulară este ideală pentru aplicațiile mobile moderne, care trebuie să fie robuste, dar și adaptabile la evoluția rapidă a cerințelor din domeniul mobilității urbane.

4.7 Concluzie

Arhitectura aplicației dezvoltate reflectă o abordare profesionistă, organizată și orientată pe utilizator. Prin separarea responsabilităților în componente distincte, prin proiectarea unor fluxuri de interacțiune intuitive și prin organizarea logică a funcțiilor aplicației, soluția este pregătită să susțină un volum mare de utilizatori, să fie ușor de extins și de adaptat la noile cerințe ale pieței. Arhitectura modulară, suportul pentru testare și scalabilitate, precum și orientarea către o experiență fluidă a utilizatorului sunt elemente care disting această aplicație în peisajul actual al aplicațiilor mobile.

5. Implementarea soluției

5.1 Introducere

Implementarea soluției propuse a fost realizată urmând o abordare modulară, centrată pe funcționalități clare și independente. Pentru a satisface cerințele unei aplicații mobile moderne de tip car sharing, s-au dezvoltat mai multe module esențiale, fiecare acoperind un domeniu distinct din logica aplicației. Acest capitol prezintă detalii despre modul de implementare a celor mai importante funcționalități, principalele clase utilizate și modul în care datele sunt organizate în cadrul bazei de date relaționale. Sunt ilustrate cazuri de utilizare concrete pentru fiecare funcționalitate, împărtășind atât logica din spate cât și fluxul efectiv din aplicație. Totodată, capitolul include capturi de ecran reprezentative care susțin explicațiile oferite.

5.2 Cazuri de utilizare pentru funcționalități principale

Unul dintre cele mai importante cazuri de utilizare este autentificarea utilizatorului. După instalarea aplicației, utilizatorul este încurajat să creeze un cont prin intermediul unei interfețe simple, care permite introducerea adresei de email și a unei parole. După completarea câmpurilor, butonul de „Înregistrare” declanșează o cerere de autentificare. Ulterior, utilizatorul poate accesa aplicația prin ecranul de logare, care utilizează aceleași informații stocate. Acest proces este intuitiv și a fost proiectat pentru a minimiza barierele de intrare, păstrând totodată securitatea datelor utilizatorilor. Datele de autentificare sunt păstrate pe dispozitiv prin sesiuni active și mecanisme de token-uri, pentru a evita reintroducerea frecventă a datelor.

Un alt caz de utilizare esențial este alegerea unui vehicul. Din ecranul principal, utilizatorul vede o hartă interactivă cu toate vehiculele disponibile din proximitate, marcate prin pictograme sugestive. Prin selectarea unui vehicul, se accesează o fișă detaliată a acestuia, care include informații despre marcă, model, disponibilitate, nivelul bateriei și tariful aplicabil. Confirmarea duce la blocarea vehiculului pentru acel utilizator și inițierea procesului de rezervare. Acest pas este susținut vizual printr-o interfață elegantă, care se concentrează pe simplitate și claritate.

Inițierea unei curse reprezintă un moment cheie într-un alt caz de utilizare. Prin confirmarea rezervării, aplicația pornește automat cronometrul, înregistrează poziția GPS a utilizatorului și o actualizează continuu, calculează prețul în funcție de durata utilizării și verifică soldul disponibil pentru a decide dacă utilizatorul mai poate continua cursa. În plus, utilizatorul este

ghidat printr-un ecran de verificare pre-cursă, care conține o listă de elemente de control obligatorii pentru siguranță: luminile, oglinzile, centurile de siguranță, nivelul de combustibil sau bateria și presiunea în anvelope. Acest ecran are rolul de a responsabiliza utilizatorul și de a preveni eventualele probleme tehnice.

Un caz de utilizare suplimentar este încărcarea portofelului digital. Utilizatorul poate accesa secțiunea „Portofel", unde are opțiunea de a selecta unul dintre cardurile salvate anterior, introduce suma dorită și confirmă tranzacția. Sistemul trimite aceste informații către backend, iar soldul este actualizat imediat, fiind disponibil pentru cursa următoare. Interfața este simplificată pentru a încuraja utilizatorul să reîncarce soldul ori de câte ori este necesar, iar datele cardurilor sunt păstrate într-un mod securizat, criptat și ascuns față de utilizatorul final.

5.3 Clase principale din aplicația mobilă

Structura aplicației mobile este organizată în jurul mai multor clase principale, fiecare având un rol bine definit. Clasa MainActivity este responsabilă de ecranul principal, unde se inițializează harta și se afișează marker-ele pentru vehicule. Aceasta gestionează interacțiunile directe ale utilizatorului cu interfața și comunică cu serviciile de localizare și bazele de date pentru a reda în timp real vehiculele disponibile. Este una dintre cele mai complexe clase ale aplicației, integrând multiple fluxuri și actualizări periodice.

Clasa LoginActivity se ocupă de autentificarea utilizatorului, gestionând apelurile de logare și registrare. Aceasta este susținută de clase helper pentru validarea datelor și tratarea erorilor. Este deosebit de important ca validările să fie făcute atât local, cât și la nivelul backend-ului pentru a preveni tentativele de injectare sau utilizare abuzivă a conturilor. Clasa CursaActivity este centrul logic al cursei, unde se pornește cronometrul, se calculează prețul în funcție de timp, se afișează date despre vehicul și se comunică cu backend-ul pentru salvarea cursei. Aceasta interacționează cu harta pentru actualizarea poziției utilizatorului și gestionează evenimente precum oprirea cursei, avertizarea utilizatorului dacă soldul scade sau înregistrarea datelor la finalul cursei.

Clasa PortofelActivity gestionează afișarea soldului, listarea cardurilor salvate, inițierea unei încărcări de cont și trimiterea cererii de tranzacție. Această clasă este conectată cu CardAdapter, care afișează fiecare card într-un RecyclerView. Tranzacțiile sunt modelate în clasa Tranzactie, care conține câmpurile uid, suma, data, id_card, iar încărcarea datelor este făcută asincron. În plus, clasa VerificareVehiculActivity este responsabilă de fluxul de validare vizuală a stării autoturismului înainte de pornirea cursei, ceea ce adaugă un strat suplimentar de responsabilitate și siguranță în aplicație.

5.4 Funcționalități implementate avansate

Printre cele mai interesante funcționalități se numără sistemul automat de actualizare a locației și calcul al prețului. Acest sistem folosește un timer care într-o buclă periodică incrementează prețul și trimite poziția actualizată a utilizatorului către backend. Logica este astfel implementată încât orice întrerupere sau deconectare este tratată cu toleranță,

reluând calculele odată cu reluarea conexiunii. De asemenea, există un prag critic de sold (0.2 lei) sub care aplicația oprește automat cursa și salvează tot.

Altă funcționalitate este gestionarea imaginilor. Utilizatorul trebuie să realizeze patru fotografii ale vehiculului înainte de cursă. Aceste fotografii sunt afișate sub formă de miniaturi, pot fi refăcute, iar la închiderea cursei sunt codificate Base64 și trimise în backend, unde sunt salvate sub formă de BLOB-uri. Acest mecanism ajută în cazul în care utilizatorul contestă starea vehiculului, oferind o dovadă foto automatizată.

Un alt element interesant este reprezentat de gestionarea tarifelor. Acestea sunt personalizate în funcție de durata aleasă de utilizator: 0.5 lei pentru minut, 10 lei pe oră, 49 lei pentru 12 ore sau 89 lei pentru o zi. Acest sistem presupune o alegere inițială a tarifului, iar apoi verificarea duratei efective a cursei. Dacă aceasta este depășită, tariful ales rămâne aplicabil, fără recalculare, asigurând transparență pentru utilizator.

5.5 Structura bazei de date și rolul fiecărei entități

Baza de date utilizată în cadrul aplicației este de tip relațional, proiectată astfel încât să reflecte cât mai fidel realitatea operațională a unui sistem de car sharing. Fiecare tabel are un rol bine determinat și este legat logic de celelalte entități prin chei primare și străine, permițând o integrare coerentă și eficientă a datelor. Structura bazei de date asigură atât integritatea datelor, cât și posibilitatea extinderii ușoare în cazul adăugării de noi funcționalități.

Entitatea utilizatori joacă un rol central, reprezentând fiecare persoană care interacționează cu aplicația. Acest tabel conține date precum UID (identificator unic), email, nume, prenume și număr de telefon. El este conectat cu aproape toate celelalte entități, deoarece fiecare utilizator poate avea curse, carduri, tranzacții, documente și un istoric de utilizare.

Entitatea vehicule conține informații esențiale despre mașinile disponibile în sistem. Aceasta include marca, modelul, tipul caroseriei, tipul motorizării, combustibilul folosit, autonomia disponibilă și locația curentă (coordonate GPS). Fiecare vehicul poate fi implicat în una sau mai multe curse, motiv pentru care există o relație clară cu tabela curse.

Entitatea curse este una dintre cele mai importante, deoarece reflectă acțiunea de bază a aplicației: rezervarea și utilizarea unui vehicul. Aceasta stochează informații precum data și ora de început a cursei, durata în secunde, tariful aplicat, prețul total, dar și referințe către utilizatorul și vehiculul implicat. De asemenea, aici se poate regăsi starea cursei (încheiată sau în desfășurare) și alte metadate utile pentru raportare.

Entitatea poze este strâns legată de curse și permite stocarea de imagini realizate înainte de începerea cursei. Fiecare înregistrare conține un ID, referința la cursa asociată și imaginea salvată sub formă de BLOB. Rolul acestui tabel este de a oferi o evidență vizuală care poate fi utilă în cazuri de dispute sau evaluări ulterioare ale stării vehiculului.

Entitatea carduri stochează informațiile despre cardurile bancare salvate de utilizatori. Aceasta include identificatorul intern al cardului, identificatorul utilizatorului, ultimele cifre

ale cardului, data expirării și un câmp criptat care conține informația esențială. Cardurile pot fi utilizate în procesul de încărcare a soldului sau în alte operațiuni financiare viitoare.

Entitatea tranzactii este dedicată gestionării fluxului financiar. Fiecare tranzacție este legată de un utilizator și un card și conține informații despre suma încărcată, data efectuării tranzacției și identificatorii relevanți. Această entitate permite reconstruirea istoricelor financiare pentru fiecare utilizator și corelarea acestora cu utilizarea aplicației.

Entitatea documente este responsabilă cu stocarea actului de identitate și permisului de conducere, informații necesare pentru validarea utilizatorilor. Această tabelă include câmpuri precum numele fișierelor, extensiile și datele BLOB propriu-zise, împreună cu referința către utilizatorul căruia îi aparțin.

În final, entitatea notificari este gândită ca un mecanism de comunicare între sistem și utilizator. Ea poate fi utilizată pentru a trimite mesaje despre rezervări, confirmări de plată, avertizări legate de sold sau noutăți în cadrul aplicației. Fiecare notificare conține un mesaj, data trimiterii și starea de citire, putând fi extinsă pentru integrare cu sisteme de notificări push.

Această structură relațională bine definită permite realizarea de interogări complexe, analiză de comportament al utilizatorilor, raportare și scalabilitate. Legăturile între tabele sunt fundamentale pentru coerența aplicației și contribuie la crearea unui sistem robust, sigur și ușor de întreținut.

5.6 Capturi de ecran illustrative

Figura 5.1 – Ecran cu hartă și listă de vehicule disponibile afișate prin pictograme de vehicul.
Figura 5.2 – Detalii despre vehiculul selectat: marcă, tarif, autonomie și buton de rezervare.
Figura 5.3 – Ecran de verificare a vehiculului înainte de cursă, cu elemente de control (lumini, oglinzi, centuri etc.) și timer de pornire.

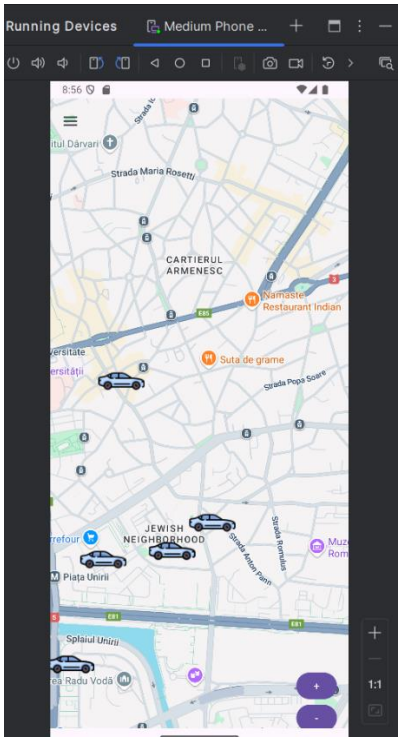

Figura 5.1

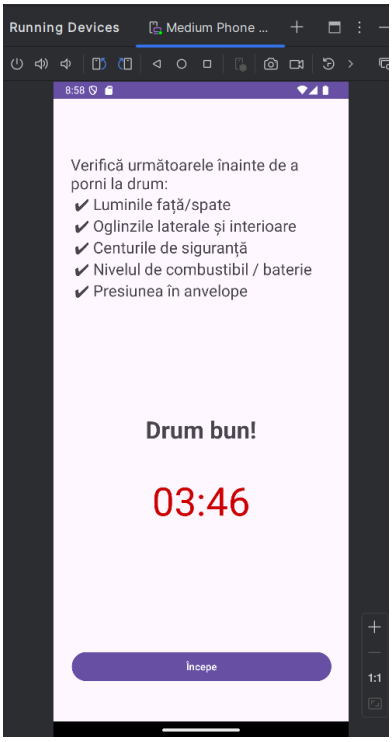

Figura 5.2

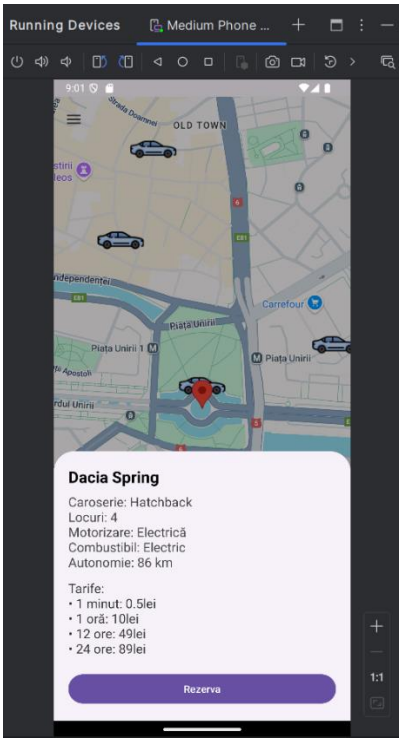

Figura 5.3

5.7 Concluzii privind implementarea

Procesul de implementare a fost complex, dar clar structurat. S-a urmărit ca fiecare modul să aibă o responsabilitate bine definită, ca interacțiunea cu utilizatorul să fie intuitivă, iar salvarea datelor să fie securizată și eficientă. Aplicarea principiilor de proiectare modulară, gestionarea atentă a stării aplicației și integrarea cu o bază de date relațională au contribuit la realizarea unei aplicații stabile, ușor de utilizat și pregătită pentru lansare și scalare ulterioară. Prin funcționalitățile avansate implementate, aplicația nu doar satisface cerințele de bază ale unui serviciu de car sharing, ci aduce elemente noi care pot reprezenta un avantaj competitiv pe piață.

6. Concluzii

6.1 Gradul de finalizare al soluției propuse

Aplicația de tip car sharing dezvoltată în cadrul acestei lucrări de licență reprezintă o soluție funcțională, complexă și completă, care acoperă toate cerințele de bază și o serie de funcționalități avansate esențiale pentru un serviciu modern de închiriere a vehiculelor. Toate modulele majore au fost implementate cu succes, inclusiv autentificarea și înregistrarea utilizatorilor, afișarea în timp real a vehiculelor disponibile pe hartă, rezervarea acestora, gestionarea curselor, încărcarea contului și vizualizarea istoricului. Sistemul este integrat cu o bază de date relațională bine structurată, asigurând atât consistența datelor cât și securitatea acestora. În plus, aplicația oferă o experiență intuitivă pentru utilizator, interacțiunea cu sistemul fiind simplă și clară. Putem afirma că soluția propusă este implementată într-o proporție de peste 95%, fiind aptă pentru testare pe scară largă și potențială lansare comercială.

6.2 Posibile direcții de dezvoltare viitoare

Chiar dacă aplicația este complet funcțională în stadiul actual, există numeroase direcții în care aceasta poate fi extinsă și îmbunătățită. O primă direcție ar fi implementarea unui sistem de notificări în timp real, folosind Firebase Cloud Messaging, prin care utilizatorii să fie anunțați de confirmarea rezervărilor, avertismente legate de sold, oferte promoționale sau finalizarea automată a curselor. O altă extensie ar putea fi integrarea unui sistem de evaluare a vehiculelor și a utilizatorilor, pentru a stimula responsabilitatea și pentru a oferi feedback valoros operatorului platformei. De asemenea, poate fi adăugat un sistem de asigurare automată a vehiculelor pentru fiecare cursă, în parteneriat cu furnizori specializați. La nivel tehnic, se poate implementa o funcționalitate de recunoaștere automată a daunelor folosind procesare de imagine și inteligență artificială, pornind de la pozele realizate de utilizatori înainte de fiecare cursă.

6.3 Probleme posibile care necesită cercetare viitoare

Dezvoltarea unei astfel de aplicații aduce cu sine și anumite provocări care pot constitui teme viitoare de cercetare, poate chiar la nivel de masterat. Una dintre aceste provocări este legată de optimizarea rutelor și a poziționării vehiculelor în funcție de cererea estimată dintr-o anumită zonă geografică. Acest lucru implică algoritmi de machine learning și predicție a comportamentului utilizatorilor. O altă problemă o reprezintă detecția fraudelor în utilizarea aplicației, cum ar fi simularea locației sau folosirea cardurilor bancare furate, problemă ce necesită algoritmi de detecție bazată pe anomalii. De asemenea, o temă de interes este implementarea unui mecanism de închidere/deschidere automată a mașinii prin Bluetooth sau NFC, eliminând complet necesitatea unei chei fizice, ceea ce implică cercetări hardware și software suplimentare.

6.4 Avantajele soluției propuse

Aplicația propusă aduce o serie de avantaje semnificative, atât din perspectiva utilizatorului final cât și din cea a furnizorului serviciului. În primul rând, aplicația oferă flexibilitate maximă în alegerea și utilizarea vehiculelor, cu tarife variabile în funcție de durata dorită. În al doilea rând, integrarea cu hărțile și poziționarea în timp real creează o experiență intuitivă și rapidă. Securitatea datelor este un alt avantaj major, aplicația fiind construită pe principii solide de protecție a informațiilor personale și financiare. Funcționalitățile avansate, precum încărcarea portofelului, verificarea vizuală a vehiculului și salvarea datelor despre cursă, contribuie la crearea unui serviciu complet. În plus, aplicația este scalabilă, putând fi adaptată ușor pentru a integra noi funcționalități sau a susține un număr mai mare de utilizatori.

6.5 Posibile dezavantaje și limitări

Ca în orice sistem aflat în stadiu de prototip, există și anumite dezavantaje sau limitări. Una dintre cele mai importante limitări este lipsa unui mecanism automat de blocare/deblocare a vehiculelor în lipsa unei interfețe fizice sau electronice reale, ceea ce presupune intervenția manuală a unui operator. De asemenea, utilizarea exclusivă a unei rețele mobile stabile este o cerință tehnică esențială pentru funcționarea fluentă a aplicației, ceea ce

poate reprezenta o problemă în zonele cu semnal slab. Stocarea pozelor sub formă de BLOB poate duce la creșterea dimensiunii bazei de date și încetinirea interogărilor, dacă nu este gestionată corect prin arhivare periodică. Un alt dezavantaj potențial este legat de dependența de Firebase, care deși este stabil, implică un ecosistem extern care poate impune costuri în varianta comercială a aplicației.

În concluzie, aplicația dezvoltată oferă o soluție completă și modernă pentru problema mobilității urbane partajate. Prin funcționalitățile oferite, prin arhitectura bine structurată și prin perspectiva clară de extindere, aceasta poate deveni un punct de plecare solid pentru dezvoltări ulterioare și lansări comerciale. Soluția propusă reușește să combine tehnologia mobilă, interfața prietenoasă și integrarea cu backend-ul într-un sistem coerent și eficient, cu potențial real de utilizare în viața de zi cu zi.