



Academia de Studii Economice din București  
Facultatea de Cibernetică, Statistică și Informatică Economică  
Specializarea Informatică Economică

## LUCRARE DE LICENȚĂ

-

Aplicație mobilă pentru park-sharing

Coordonator științific:

Conf. univ. dr. Vespan Dragoș

Absolvent:

Șandru Dumitru-Dragoș

București, 2021

## Cuprins

1. Introducere.....	3
2. Conceptul de „park-sharing” .....	5
2.1. Economia colaborativă.....	6
2.2. Analiza pieței de park-sharing .....	8
2.3. Concurența.....	9
3. Descrierea sistemului informatic .....	11
4. Tehnologii folosite .....	20
4.1. Sistemul de operare Android .....	20
4.2. Bază de date non-relațională.....	22
4.3. API-uri și kituri de dezvoltare software .....	25
4.4. Interfața grafică cu utilizatorul .....	27
5. Concluzii .....	31
Bibliografie.....	32
Anexa 1.....	34
Anexa 2.....	35

## 1. Introducere

Aplicația mobilă de *park-sharing* pe care am intitulat-o „EveryPark” își propune să reducă poluarea și timpul destinat căutării unui loc de parcare liber în orașele aglomerate. Astfel, un utilizator al platformei va putea:

1. Să pună locul de parcare personal la dispoziția altor utilizatori ai platformei atunci când nu îl folosește (spre exemplu atunci când este plecat în vacanță sau la serviciu);
2. Să utilizeze locurile de parcare ale altor utilizatori atunci când are nevoie;

În marile orașe ale României traficul este îngreunat de numărul mult prea mare de mașini, iar lipsa locurilor de parcare, în special în zonele centrale, nu face decât să adauge un motiv în plus de frustrare pentru șoferi. Sistemul informatic prezentat în această lucrare oferă o soluție modernă la lipsa locurilor de parcare.

Al doilea capitol al lucrării constituie o cartografiere a situației traficului și a locurilor de parcare publice din orașele României, urmând apoi prezentarea avantajelor și posibilele greutăți care pot apărea în implantarea unui sistem informatic de *park-sharing* la scară largă. Vor fi introduse conceptele de bază asociate economiei colaborative (*sharing economy*) și se vor analiza posibilitățile de creștere a businessului în spațiul românesc. De asemenea, vor fi prezentate soluțiile deja existente pe piață și se vor compara avantajele lor competitive.

Capitolul al treilea este destinat descrierii sistemului informatic și a interacțiunii dintre componentele sale. Prin intermediul diagramelor UML se vor prezenta cazurile specifice de utilizare ale aplicației mobile, principalele cerințe care trebuie îndeplinite de sistemul informatic, precum și arhitectura sistemului. Se vor preciza, de asemenea, punctele în care sistemul are deficiențe și modul în care poate fi optimizat.

Capitolul al patrulea este destinat tehnologiilor utilizate în realizarea sistemului informatic, de la utilizările specifice ale limbajului de programare Java în cadrul sistemului de operare Android, până la avantajele utilizării unei baze de date tip NoSQL precum Firebase. API-urile și kiturile de dezvoltare software (SDK) precum Google Maps SDK, Google Places API și Firebase Android SDK oferă funcționalități vitale și implementare facilă în sistem.

În secțiunea de concluzii se vor reitera principalele obiective ale sistemului informatic realizat, iar autorul își va expune pe scurt opiniile cu privire la lecțiile învățate în timpul scrierii lucrării de licență.

Bibliografia este constituită din repoarte ale unor firme de consultanță recunoscute pentru a evidenția sustenabilitatea economică a sistemelor care implementează principiile economiei colaborative. De asemenea, au fost citate articole din presa internă, acolo unde erau necesare date statistice locale, consultând în prealabil surse multiple pentru a verifica veridicitatea datelor. La partea tehnică, a fost folosită documentația oficială Android sau a serviciilor puse la dispoziția publicului de către Google, pe lângă alte articole considerate relevante.

**Codul sursă** al aplicației EveryPark poate fi consultat integral la:

<https://github.com/dragosppp/EveryPark>

**Notății:** Cuvintele preluate din limba engleză și care au traduceri imperfecte în limba română, precum și numele unor funcții sau variabile, apar scrise cu italice (*crowdsourcing*, *peer to peer*, *myVar*, etc.)

## 2. Conceptul de „park-sharing”

*Park-sharing* reprezintă un concept relativ recent apărut care s-ar traduce în română prin „folosirea în comun a unui loc de parcare” și presupune eficientizarea utilizării spațiilor de parcare existente. Deoarece un loc de parcare privat rămâne liber în momentul deplasărilor proprietarului, acesta ar putea fi valorificat nu doar în favoarea comunității de șoferi care l-ar putea folosi, dar și a proprietarului care poate folosi la rândul său alte locuri de parcare din rețeaua colaborativă.

Eficientizarea utilizării spațiilor existente de parcare contribuie la reducerea timpului de căutare a unui loc liber în zonele aglomerate din orașe și la diminuarea nevoii construirii unor noi locuri. Din punct de vedere urbanistic, parcarile au reprezentat un compromis, ele nefiind plăcute din punct de vedere vizual și nici nu au vreo utilitate pentru cei care nu sunt posesori de vehicul personal. Datorită popularității deținerii de mașini personale și a convenienței deplasării cu acestea, parcarile au devenit un lucru necesar. Există numeroase moduri în care acel spațiu betonat ar putea fi folosit: spațiu de comerț, spațiu locuibil, spații verzi sau chiar piste de biciclete în cazul parcarilor stradale.

Shared parking este o formă de management a locurilor de parcare care facilitează în special interacțiunea de tip P2P (*peer to peer*) și mai puțin metodele tradiționale de gestiune a locurilor de parcare care sunt de tip B2P (*business to peer*) în cazul parcarilor cu plată la oră sau în cazul subcontractării anuale a unui loc de parcare de la administrația publică locală. În cazul implementării unei forme de plată între utilizatori ar exista avantajul liberalizării prețurilor. Astfel, prețul per oră de utilizare într-o rețea de tip park-sharing ar fi dictat de mecanismul cererii și ofertei, ci nu impus de către o instituție publică sau firmă privată așa cum se întâmplă în prezent.

Platforme de park-sharing sunt în general create de administrațiile locale din marile orașe cu scopul de a ușura viața șoferilor care pot rezerva online un loc de parcare deținut de primărie fără a mai fi nevoie de instalarea unor taxatoare. Datele colectate de o astfel de aplicație ajută la cartografierea locațiilor după gradul de ocupare și identificarea tiparelor de ocupare pe ore și zilele săptămânii. Aceste informații sunt esențiale pentru o mai bună adaptare a infrastructurii urbane la nevoile comunității. Marea problemă a acestor platforme finanțate din bani publici este că ele funcționează doar la nivel local, de oraș. Ele au o taxă fixă în funcție de zonă și de cele mai multe ori se ocupă doar de locurile de parcare care aparțin domeniului public, fără să

integreze și operatorii privați de parări pentru o mai bună coeziune a serviciilor. Tocmai de aceea, platformele private vin în soluționarea acestor hibe, gândind totul la nivel național sau chiar global.



Figura 2.1. Park-sharing românesc

Ideea de a folosi la comun puținele locuri de parcare disponibile nu este în sine una nouă. În orașele sufocate de mașini și în care investițiile în infrastructură rutieră sunt puține sau chiar inexistente, șoferii români au dezvoltat metode informale de a se folosi cât mai bine de locurile existente. În momentul parării într-un loc marcat ca fiind subcontractat de altă persoană de la administrația publică locală, șoferul aflat în ilegalitate lasă în geam o hârtie cu numărul personal de telefon, având obligația, conform cutumei autohtone, să-l elibereze imediat ce proprietarul de drept îi solicită acest lucru. O platformă de park-sharing joacă rolul hârtiei puse în geam, cu diferența evidentă că este mai versatilă. Aplicația este accesibilă de pe orice telefon, este sigură din punct de vedere al protecției datelor personale și, dacă platforma permite, deținătorii de locuri de parcare pot monetiza închirierea.

## 2.1. Economia colaborativă

Conceptul de park-sharing este o subcategorie a unui model economic emergent numit *conomie colaborativă* sau *conomie în comun*<sup>1</sup> și presupune o mai bună gestionare a bunurilor și serviciilor existente prin intermediul tehnologiei informației. Principalele argumente în

---

<sup>1</sup> În engleză se folosesc, de asemenea, o serie de termeni, precum: *mesh/sharing/peer-to-peer economy*; *collaborative consumption*

favoarea economiei colaborative sunt rentabilitatea și sustenabilitatea. Spre exemplu, multe obiecte proprietate privată precum un autovehicul stau până la 95% din timp nefolosite (Knowledge@Wharton, 2015), proprietarul asumându-și devalorizarea bunului pe care doar rareori îl folosește. Tocmai de aceea, devine atractiv din punct de vedere financiar folosirea în comun și plata unui produs fix pe perioada în care este utilizat.

Platformele de *sharing* vizează diferite categorii de bunuri: autovehicule (Uber, BlaBlaCar, FlixBus, Spark), trotinete electrice (Lime, Splash), locuințe (CouchSurfing, AirBnb), spații de birouri (WeWork, Hubble), spații de depozitare (Omni), cărți (Bookster) sau revânzări (OLX, Ebay, Craigslist). Orice bun care se pretează utilizărilor multiple poate face obiectul unei închirieri, împrumut, revânzări sau schimb. Modelul de business diferă de la o companie la alta. În timp ce unele firme precum AirBnb intermediază doar tranzacțiile dintre proprietarul de facto și cel care închiriază, fără să dețină bunurile, altele precum Bookster și Lime dețin bunurilor pe care le oferă spre închiriere. Alte platforme facilitează prestarea de servicii între utilizatori: curierat (Glovo, Tazz), finanțări sau *crowdsourcing* (KickStarer, Patreon), divertisment *on-demand* (YouTube, Twitch, Spotify), cunoștințe tehnice sau aptitudini<sup>2</sup> (Fiverr, UpWork).

Interesul pentru acest nou sistem socio-economic se prezintă la un nivel ridicat. Un studiu realizat de PwC pe un eșantion de 1000 de cetățeni din Statele Unite a arătat că în proporție de peste 83% din aceștia consideră că economia colaborativă are costuri reduse și contribuie la un stil de viață mai practic, iar peste 76% consideră că: ajută la menținerea comunității unite, este benefică pentru conservarea mediului și evită risipa (PriceWaterhouseCoopers, 2015a). Un alt raport al PwC estimează că veniturile totale ale companiilor din această arie economică vor crește de la 81 miliarde de dolari în 2020 până la 335 miliarde în 2025 (PriceWaterhouseCoopers, 2015b).

Argumentul ecologist este unul semnificativ. Spre exemplu, producția unui singur autovehicul eliberează în atmosferă în medie 5 tone de dioxid de carbon, adică peste 12% din totalul emisiilor de CO<sub>2</sub> eliberate în cursul unui ciclu mediu de utilizare de 15 ani (European Federation for Transport and Environment, 2018, p.40). Astfel, printr-o producție mai redusă de autovehicule ca urmare a transportului colaborativ se evită emisia de gaze cu efect de seră. Consumul colaborativ este sustenabil din punct de vedere ecologic, dar fără a prejudicia sau a

---

<sup>2</sup> Din engl. *freelancing*

priva consumatorului final. Un argument similar se poate aduce și în cazul locurilor de parcare, după cum se va putea observa în subcapitolul următor.

Un alt avantaj este reprezentat de comoditatea utilizării acestor servicii moderne. Odată cu răspândirea accesului la internet și la dispozitivele mobile inteligente, a devenit justificabilă economic crearea unor sisteme informatice care să gestioneze un volum mare de date și să încurajeze interacțiunea continuă între utilizatori. În mod comun, sistemul informatic veghează asupra respectării unor reguli de interacțiune și girează tranzacțiile, jucând rolul de mediator. Economia colaborativă poate fi considerată drept un fenomen tehnologic (Hamari, J, 2015).

## **2.2. Analiza pieței de park-sharing**

Sistemul informatic de park-sharing care face obiectul acestei lucrări se încadrează în sfera generală a economiei colaborative prezentate anterior și își propune să rezolve o problemă actuală din spațiul românesc: lipsa locurilor de parcare. Astfel, șoferii ar avea posibilitatea să împartă locurile de parcare deja existente într-un mod rapid, sigur și predictibil.

Problema locurilor de parcare și a poluării din orașe este veche și cunoscută. Transportul cu mașina personală este unul comod, deși infrastructura rutieră are multe neajunsuri. Se înmatriculează un număr mare de mașini raportat la populație, marea lor majoritate fiind mașini depășite tehnologic, la mâna a doua. La nivel de țară, România dispune de 1,2 milioane de locuri de parcare în spațiul public pentru cele 8,7 milioane de mașini înmatriculate până în anul 2019 (Wall-Street.ro, 2020).

Cazul Bucureștiului este unul deosebit. La o populație de 1,8 mil. locuitori sunt înmatriculate 1,4 mil. autoturisme, fără a fi luate în considerare cele care fac naveta zilnic din județele învecinate (Digi24, 2020). Locuri de parcare pe domeniul public sunt în număr de aproximativ 350.000, ceea ce înseamnă că 4 mașini concurează în medie pentru un loc de parcare. În funcție de oră și ziua săptămânii, găsirea unui loc în zona centrală a Bucureștiului poate dura și 20 minute, iar de cele mai multe ori soluția de compromis găsită de șoferi reprezintă parcare ilegală pe trotuar sau în dreptul trecerilor de pietoni. Soluțiile de fluidizare a traficului precum parcarile de tip park&ride de la intrarea în oraș s-au dovedind ineficiente, chiar și atunci când primăriile le oferă gratuit (cum e și cazul parcarii Străulești). Șoferii preferă din comoditate să conducă cât mai aproape de locația de destinație.

Parcarile private din zona centrală rămân și ele mai mult goale datorită faptului că prețul pe oră se ridică până la 15lei. Construcția de noi parcuri este, de asemenea, problematică, deoarece construcția unui singur loc de parcare începe de la 10.000 euro în orașe (Ivanov C., 2019).



Soluția ideală ar trebui să satisfacă două criterii: comoditatea șoferului și dorința acestuia de a-și minimiza cheltuielile. Un sistem informatic modern poate satisface aceste cerințe.

### 2.3. Concurența

Probabil cea mai matură platformă din punct de vedere a funcționalității și a numărului zilnic de utilizatori este JustPark. JustPark activează doar în Regatul Unit al Marii Britanii și pe parcursul celor 15 ani de activitate a reușit să formeze o comunitate de 5 milioane de utilizatori, din care 45.000 sunt proprietari de locuri de parcare, inclusiv instituțiile publice care administrează domeniul public. Cu ajutorul platformei, proprietarii au primit însumat peste 5,3 mil. lire sterline<sup>3</sup>.

Aplicația mobilă a primit numeroase premii în domeniul inovației. Design-ul este unul modern, simplu și intuitiv. Există numeroase filtre pentru găsirea unui loc de parcare, există o secțiune de informații și comentarii aferentă fiecărui loc de parcare și posibilitatea de a plăti cu cardul bancar. Prețurile sunt afișate la vedere și calculate în funcție de numărul de minute ocupate.

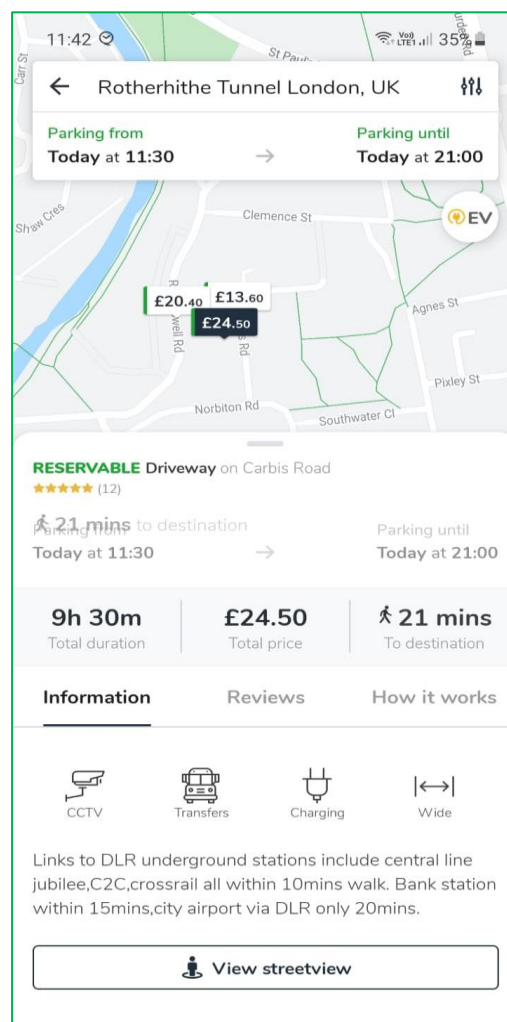


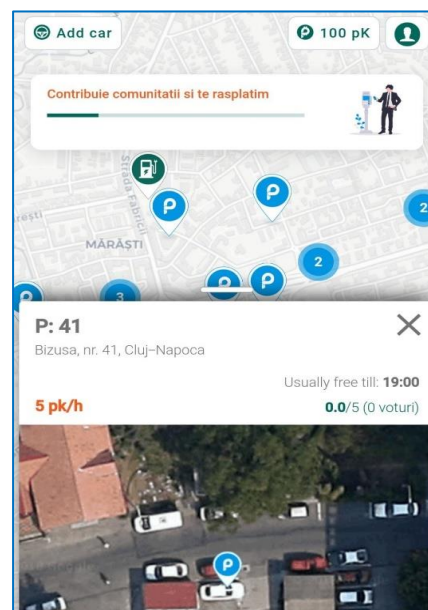
Figura 2.2. JustPark GUI

Un startup românesc, YeParking, înființat în 2018 încearcă să se impună pe piața locală, având o aplicație cu peste 5000 de descărcări în Play Store și câteva sute de locuri de parcare disponibile în orașe precum Cluj și București. Aplicația este încă la început de drum, lipsindu-

<sup>3</sup> Date preluate de pe site-ul oficial al JustPark: <https://www.justpark.com/company/overview>

i multe funcționalități importante, iar design-ul are un aspect învechit cu culori șterse și nuanțe de gri care evocă monotonie. Aplicația YeParking folosește harta OpenStreetMap, în timp ce EveryPark și JustPark utilizează mult mai familiara hartă Google Maps.

În *Tabelul 2.1* am prezentat funcționalitățile existente momentan la fiecare aplicație în parte. În raport cu YeParking, EveryPark vine cu un design modern și își propune să implementeze în viitor nu numai din funcționalitățile prezente la concurență, dar să și dezvolte propriile soluții care să o diferențieze.



*Figura 2.3. YeParking GUI*

	EveryPark	YeParking	JustPark
UI modern	✓	✗	✓
Notificări SMS/email	✗	✓	✓
Raportare probleme	✗	✓	✓
Recenzii parcări	✗	✓	✓
Stații de reîncărcare EV	✗	✓	✓
Mesagerie între utilizatori	✗	✗	✓
Opțiuni filtrare locuri parcare	✗	✗	✓
API Hartă	Google Maps	Open Street Map	Google Maps
Plată parcare	✗	Prin tokens	Direct cu cardul bancar
Sisteme de operare	Android	Android, iOS	Android, iOS
Tarifare parcări private	✗	✓	✓

*Tabelul 2.1. Funcționalități prezente la EveryPark vs competiție*

Ceea ce scoate în evidență EveryPark este designul minimalist, elegant și intuitiv. Bineînțeles, funcționalitățile se prezintă la un nivel minimal. Dezvoltarea efectivă a aplicației se află în urma competitorilor din piață care dispun de bugete de dezvoltare semnificative. Principale lipsuri în momentul de față sunt reprezentate de lipsa unui sistem de plăți între utilizatori și mecanisme de notificare ale utilizatorilor ca urmare a diferitelor schimbări de stare (notificări SMS, resetare parolă, etc.). Funcționalitatea care se distinge în cazul aplicației britanice, JustPark, este filtrarea locurilor de parcare libere în funcție de facilități: priză electrică, iluminare publică, cameră de filmat a parcării, îngrădire, precum și posibilitatea de a rezerva în prealabil un loc de parcare.

În cazul YeParking, utilizatorii tranzacționează puncte care pot fi cumpărate cu cardul sau obținute prin închirierea locului de parcare. În general, prețul de închiriere pornește de la 0,1lei/oră. La un calcul simplu, pentru o închiriere de câte 8 ore în zilele lucrătoare, tariful lunar s-ar ridica undeva la 16 lei, un preț mic dacă se iau în calcul că amenda pentru parcare în spații interzise începe de la 200 lei.

Potențialul unei astfel de platforme digitale este uriaș însă necesită investiții în soluții tehnice adaptate cerințelor pieței și marketing corespunzător pentru a atragerea unui număr cât mai mare de utilizatori. Convingerea autorităților locale și a proprietarilor de parări private să migreze către platformă în defavoarea altor sisteme de gestiune/plată reprezintă, de asemenea, o provocare.

### **3. Descrierea sistemului informatic**

Fenomenul park-sharing este nou și de mică amploare în România. Aplicațiile existente aparțin fie administrațiilor locale și se limitează la teritoriul unui anumit oraș, fie există inițiative private precum YeParking care nu au reușit încă să se impună pe piață și să formeze o comunitate de utilizatori. O soluție viabilă de park-sharing care să funcționeze predictibil la nivel național după modelul JustPark din UK, momentan nu există. Sistemul informatic propus reprezintă un model la scară mică de punere în practică a conceptului de park-sharing, evidențiind totodată cerințele ce ar trebui respectate de un astfel de sistem.

Principal obiectiv al unui astfel de sistem ar trebui să fie adoptarea aplicației de cât mai mulți utilizatori care să își pună la dispoziție locurile de parcare personale atunci când nu le folosesc și care la rândul lor să folosească cât mai des aplicația pentru a rezerva locuri de parcare. Pentru a reuși acest lucru este esențială gestionarea rapidă și sigură a unui volum mare de date precum datele utilizatorilor și ale locurilor de parcare. Introducerea datelor se face de către utilizatori prin intermediul interfeței grafice din aplicația Android cu respectarea unor validatorilor care să asigure corectitudinea datelor. Astfel, un utilizator poate avea funcție duală de a rezerva și de a pune spre rezervare un loc de parcare, iar platformă trebuie să acomodeze acest scenariu.

Pentru reprezentarea diagramelor am folosit aplicația CASE (*Computer-aided software engineering*), Visual Paradigm Community, utilizând limbajul standardizat de modelare, UML (*Unified Modeling Language*).

- **Diagrama cazurilor de utilizare**

Diagrama cazurilor de utilizare reprezintă o reprezentare grafică a diverselor moduri în care utilizatorul poate interacționa cu sistemul informatic. Diagrama este o abstractizare a principalelor cazuri de utilizare care stau la îndemâna actorilor din sistem, în cazul de față al utilizatorilor. Avantajul acestui tip de reprezentare este faptul că este accesibil și celor care nu au cunoștințe tehnice, spre deosebire de niște diagrame mai specializate cum ar fi cele de clase sau de componente. Astfel, se poate vizualiza în mod facil ce se așteaptă de fapt de la sistem.

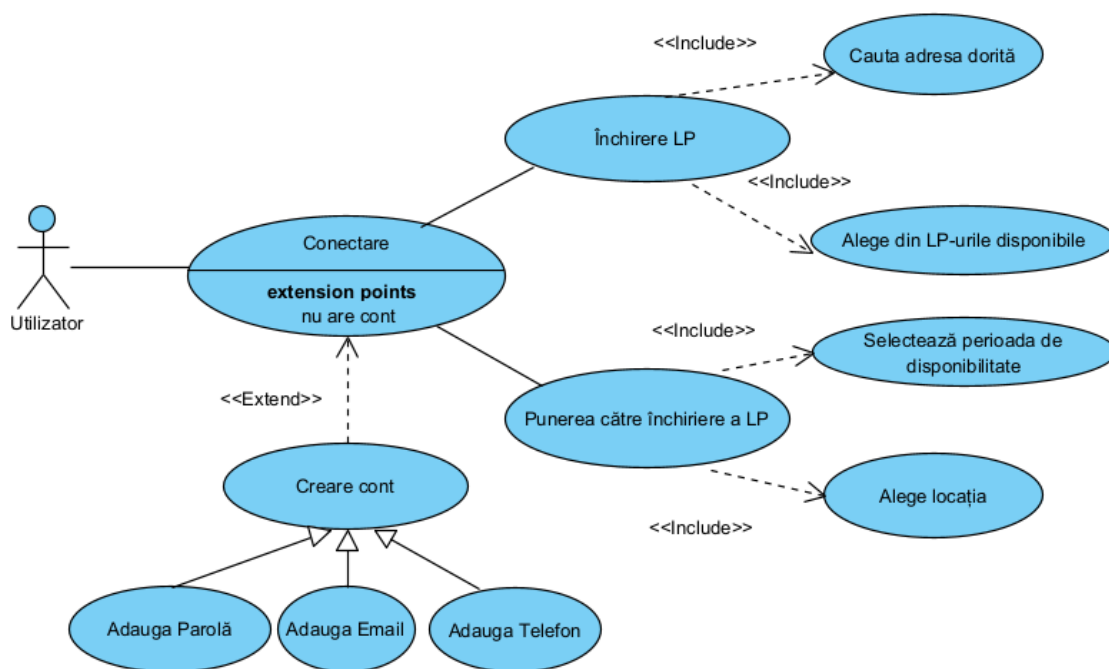


Figura 3.1. Diagrama cazurilor generale de utilizare

Se poate observa cum utilizatorul, odată ce intră în aplicație, este nevoit să se autentifice. În cazul în care nu are deja înregistrat un cont, trebuie să își creeze unul prin introducerea unui email, parolă și număr de telefon. După autentificare, el poate să își pună locul de parcare spre închiriere și/sau să închirieze pe o perioadă de maxim 2 săptămâni un loc de parcare, fiecare activitate având o serie de etape subordonate.

Descrierea textuală a cazurilor de utilizare se află în *Tabelul 3.1*, unde apar detalii suplimentare precum precondițiile și post-condițiile utilizării.

Element al cazului de utilizare	Descriere
Cod	CU-G
Stare	Schiță
Scop	Prezentarea cazurile generale de utilizare ale interfeței grafice(GUI)
Nume	Diagramă generală aplicație mobilă

Actor principal	Utilizator (client)
Descriere	Utilizatorul folosește aplicația pentru a rezerva și/sau a închiria un loc de parcare
Precondiții	Utilizatorul dispune de un telefon Android cu acces la internet pentru a se conecta în aplicație.
Post-condiții	Utilizatorul găsește un loc de parcare și/sau închiriază un loc de parcare .
Declanșator	Conectarea în aplicație prin contul de utilizator.
Flux de bază	1) Conectare <ul style="list-style-type: none"> <li>Rezervare loc parcare <ul style="list-style-type: none"> <li>Alege locul de parcare în funcție de locația dorită și perioada de ocupare preconizată</li> </ul> </li> <li>Închiriere loc parcare <ul style="list-style-type: none"> <li>Alege locația exactă a locului de parcare</li> <li>Specifică perioada disponibilă</li> </ul> </li> <li>Deconectare</li> </ul>
Fluxuri alternative	Creare cont în cazul în care accesează aplicația pentru prima oară.
Relații	-
Frecvența utilizării	Frecvent
Reguli ale afacerii	Utilizatorii au minim 18 ani. Datele personale introduse de utilizatori sunt adevărate și respectă legislația în vigoare în legătură cu protecția datelor.

*Tabel 3.2. Descrierea textuală (tip șablon) a cazului generale de utilizare*

- **Diagrama de activitate**

Diagrama de activitate este folosită pentru a ilustra desfășurarea etapelor unui caz de utilizare. Diferitele etape pot avea loc secvențial sau în paralel. Fiind o diagramă comportamentală la fel ca diagrama cazurilor de utilizare, diagrama de activitate se axează pe cauzele ce conduc la un anumit eveniment și urmările acestuia. De la punctul de start (precondiția), până la punctul terminal (post-condiția) se urmărește dinamica evenimentelor și circuitul fluxului de control prin nodurile decizionale.

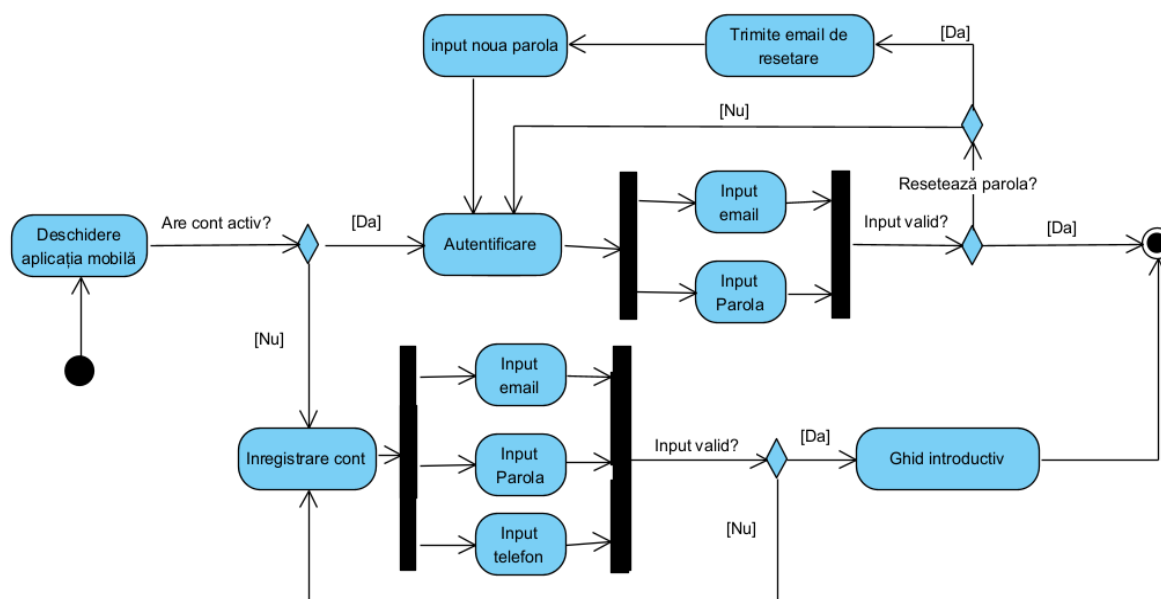


Figura 3.2. Diagrama activității de conectare în aplicația mobilă.

În diagrama precedentă, precondiția intrării în activitate este aceea ca utilizatorul să aibă instalată aplicația pe dispozitivul mobil. Rezultatul activității este autentificarea în sistemul informatic și redirecționarea către activitatea principală din interfața grafică. Primul nod decizional condiționează fluxul de control, bifurcându-l într-o etapă de autentificare pentru utilizatorii care au deja un cont activ și o etapă de înregistrare pentru utilizatorii care nu au cont activ. Diversele inputuri ale utilizatorului precum adresa de email sau parola contului sunt încadrate între un nod de bifurcație și unul de îmbinare marcate cu linii verticale îngroșate pentru a reprezenta un paralelism formal, adică pași ce trebuie îndepliniți cu necesitate pentru a trece la etapa următoare.

Datele de înregistrare în aplicație sunt salvate în memoria persistentă a dispozitivului. De aceea, utilizatorul trebuie să introducă emailul și parola:

- După înregistrarea contului
- La reinstalarea aplicației sau înlocuirea dispozitivului mobil
- Utilizatorul se deconectează manual prin apăsarea butonului „Sign out” și apoi dorește să se re-conecteze

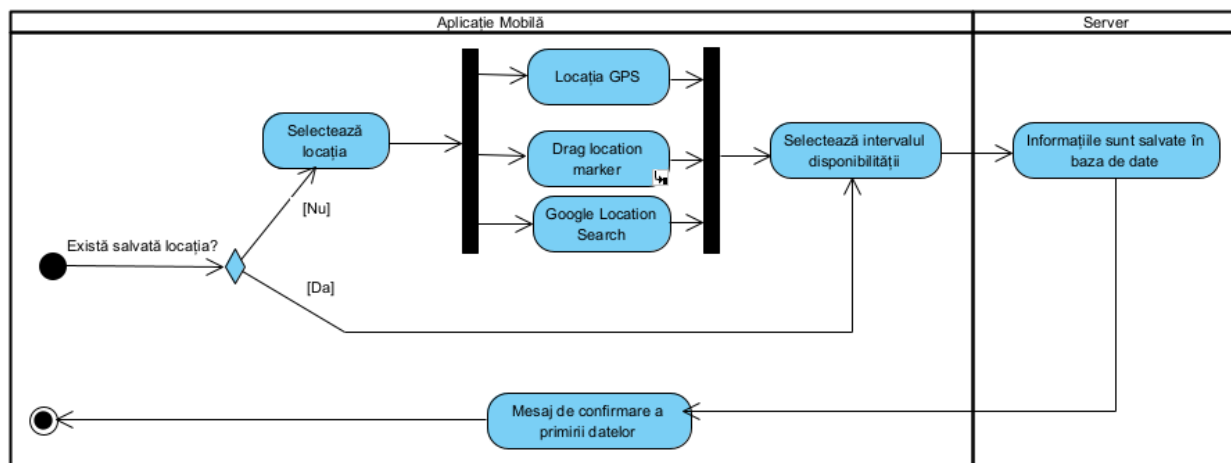


Figura 3.3. Diagrama activității de punere către închiriere a locului de parcare

În Figura 3.3 este schematizată desfășurarea etapelor prin care un utilizator ajunge să își pună spre închiriere locul personal de parcare. Dacă utilizatorul folosește această funcționalitate pentru prima oară, el trebuie să selecteze locația exactă. El poate face acest lucru folosind următoarele funcționalități care îi stau la dispoziție: bara de căutare a adresei, folosind *drag-and-drop* pe cursorul de pe hartă sau prin intermediul localizării GPS de pe dispozitiv. După selectarea intervalului de disponibilitate, datele sunt încărcate în baza de date, iar utilizatorului i se confirmă înregistrarea dacă nu apar erori.

Activitatea de închiriere este mai simplă. Aplicația solicită bazei de date o listă cu locurile de parcare pe care le afișează interactiv pe hartă în activitatea principală. Utilizatorul selectează locul de parcare dorit dintr-un anumit perimetru, apoi confirmă tranzacția. În aplicație va apărea un mesaj de confirmare în cazul în care închirierea a fost înregistrată cu succes în baza de date.

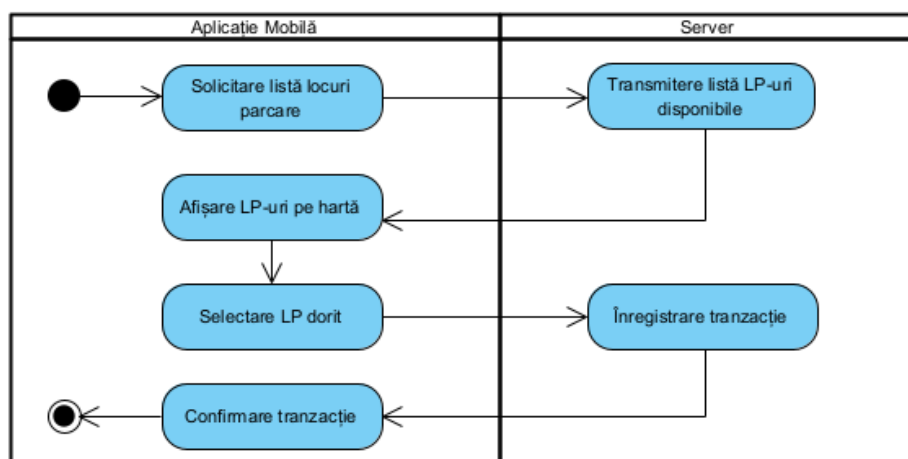
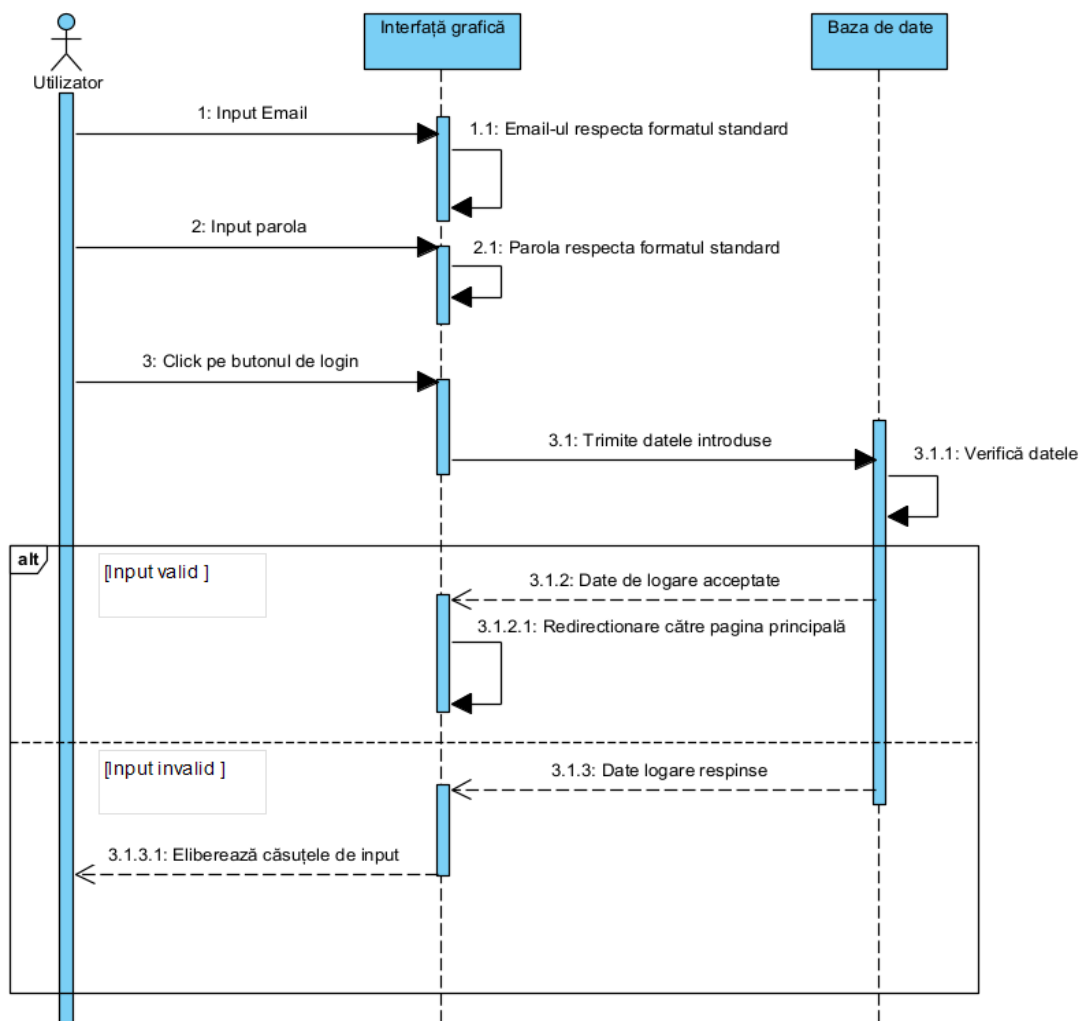


Figura 3.4. Diagrama activității de închiriere

- **Diagrama de interacțiune**

Diagramele de utilizare corespund unui singur caz de utilizare și ilustrează transferul de date care are loc. Transferul de date din fluxul de control este schematizat în ordine cronologică. Deși deservește în linii mari același scop ca o diagramă de activitate, diagrama de interacțiuni se pretează pentru un nivel ridicat de detaliere și este mai ușor de urmărit datorită faptului că limitează folosirea nodurilor decizionale, după cum se poate observa în *figura 3.5* de mai jos. Utilizatorul interacționează cu interfața grafică a aplicației mobile, care la rândul ei comunică cu baza de date în funcție de cerințele utilizatorului.



*Figura 3.5. Diagrama de interacțiune a procesului de autentificare*

- **Diagrama de clase**

Diagrama de clase este o reprezentare grafică a relației dintre clase și structura lor. Fiecare clasă deține un nume, attribute și metode. Diagrama de clase se pretează pentru modelarea orientată obiect, cum este și în cazul de față, codul sursă al aplicației mobile Android fiind scris în mare parte în limbajul Java.



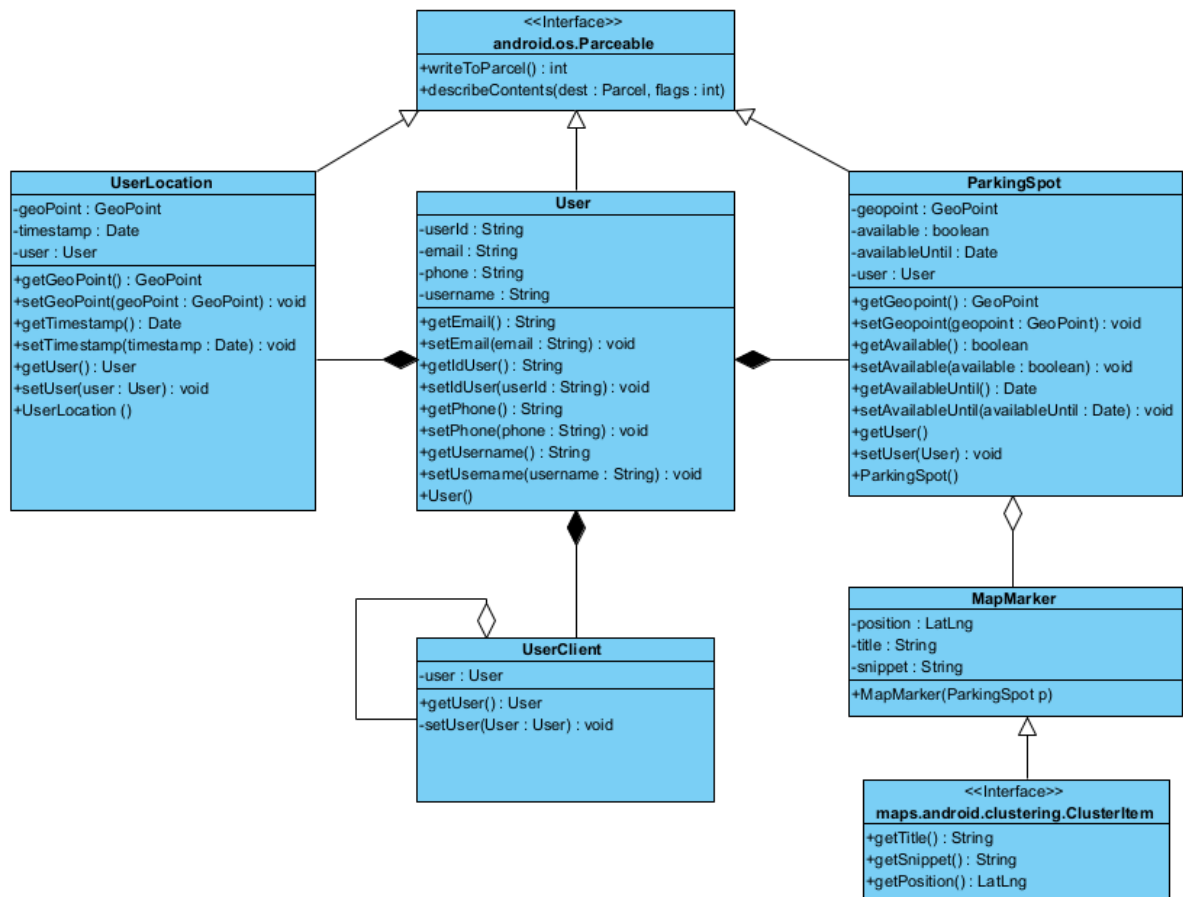


Figura 3.6. Diagrama de clase

Clasa *User* deține attributele și metodele specifice unui utilizator al EveryPark, precum ID-ul și parola. Odată autentificat cu succes în aplicație, o instanță de tip *singleton* numită *UserClient* este inițializată cu datele utilizatorului. Acest lucru este import, deoarece previne crearea de instanțe duplicate și evită interogarea multiplă bazei de date pentru același set de date. *UserLocation* are ca obiect manipularea coordonatelor GPS ale utilizatorului de aplicație. Deși attributele clasei *UserLocation* ar fi putut fi incluse la fel de bine în clasa *User*, relația de dependență (linie dreaptă cu romb negru la capăt) dintre cele două clase distincte respectă principiul *Single Responsibility*.

Orice loc de parcare, *ParkingSpot*, este proprietatea unui utilizator EveryPark. Variabila *available* este de tip boolean și are valoarea *true* când locul este disponibil spre închiriere sau *false* când nu este. În cazul în care locul de parcare este disponibil, variabila *availableUntil* indică ora și data maximă de disponibilitate.

O listă cu locurile de parcare disponibile este preluată din baza de date imediat după autentificarea în aplicație. Pentru a putea fi dispuse corect pe harta Google Maps sunt create

obiecte de tip *MapMarker* care conțin date despre locație, intervalul de disponibilitate și numărul de ore rămase până la expirare. *MapMarker* implementează interfața *ClusterItem* pentru a putea grupa ulterior obiectele în clustere. Afișarea efectivă și gruparea iconițelor în clustere este realizată de o clasă utilitară numită *MapMarkerRender*.



Figura 3.7. Afișarea locurilor de parcare disponibile folosind *MapMarker*

Clasele *UserLocation*, *User* și *ParkingSpot* implementează interfața *Parcelable* pentru a putea transfera obiecte complexe de la o activitate la alta folosind *Bundle*. *Bundle* permite transferul primitivelor, variabilelor de tip *String* și a obiectelor care implementează clasa *Serializable* sau *Parcelable*. *Parcelable* reprezintă o formă optimizată de serializare a datelor specifică Android și este inspirată din Java *Serializable*.

- **Diagrama de componente**

Diagrama de componente reprezintă o schematizare abstractă a funcționalităților sistemului informatic. Fiecare componentă îndeplinește o anumită sarcină. Ele sunt ordonate în funcție de fluxul de date, dinspre inputul primar al utilizatorului către starea finală a aplicației. O astfel de diagramă structurală arată și componentele externe care sunt implementate în sistem, precum API-uri externe, baze de date, micro-servicii, etc.

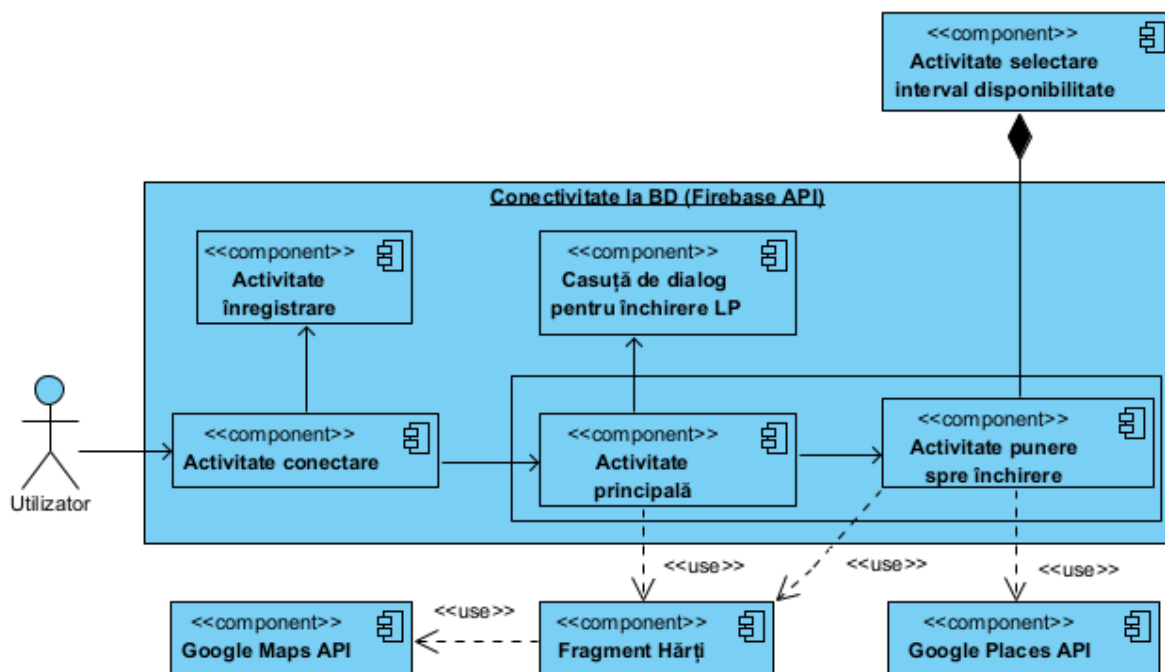


Figura 3.8. Diagrama de componente a aplicației mobile

În diagrama anterioară se poate observa faptul că aplicația mobilă conține cinci activități, dintre care patru au acces la baza de date Firebase Firestore prin intermediul unui API. Între activitatea de punere spre închiriere și activitatea de selectare a intervalului de disponibilitate este o relație de compoziție, deoarece prima nu își poate îndeplini obiectivul fără să primească rezultatul celei de-a doua.

Atât activitatea principală și cea de punere spre închiriere se folosesc de un fragment care afișează hărțile furnizate de Google Maps API. Ulterior, în fragment sunt adăugate *MapMarkers* și iconița corespunzătoare locației utilizatorului. Google Places API îi permite utilizatorului să găsească rapid orice locație de pe glob, și funcționează similar unui căutări pe Google Search. Rezultatul oferit este o adresă, și nu o pagină web.

Căsuța de dialog pentru închiriere din cadrul activității principale funcționează similar unei activități, cu avantajul că nu necesită părăsirea activității principale, fiind mult mai intuitiv de folosit de către utilizator. Componentă de tip *BottomSheetDialog* poate fi folosită cu succes pentru a crea meniuri sau pentru a înlocui vechile căsuțe de dialog tip *pop-up* din Android.

## 4. Tehnologii folosite

### 4.1. Sistemul de operare Android

Android este un sistem de operare care are la bază *Linux kernel* și este orientat către dispozitivele mobile, în particular telefoane inteligente cu ecran tactil. Începând cu anul 2005, Google oferă diferitele versiuni de Android sub licență *open-source* pentru a încuraja adopția acestuia și un ciclu de dezvoltare rapid și l-a adaptat pentru a accepta o gamă largă de dispozitive: ceasuri inteligente, televizoare, consolele mașinilor și dispozitive IoT de tot felul. Această strategie s-a dovedit a fi una câștigătoare. Android domină „lupta” sistemelor de operare pentru dispozitivele mobile cu o cotă de piață de aproximativ 75% în anul 2021<sup>4</sup>, la mare distanță de rivalul iOS. Numărul de dispozitive folosite activ care încorporează sistemul de operare Android se ridică la peste 3 miliarde<sup>5</sup>.

Versatilitatea Android și gradul ridicat de adopție se datorează în principal arhitecturii de sistemului care încurajează dezvoltarea colaborativă și inovația. În figura de mai jos sunt prezentate macro-componentele platformei de dezvoltare Android. Pornind de jos în sus, nivelul de abstractizare crește, la fel și nivelul de specializare al componentelor.

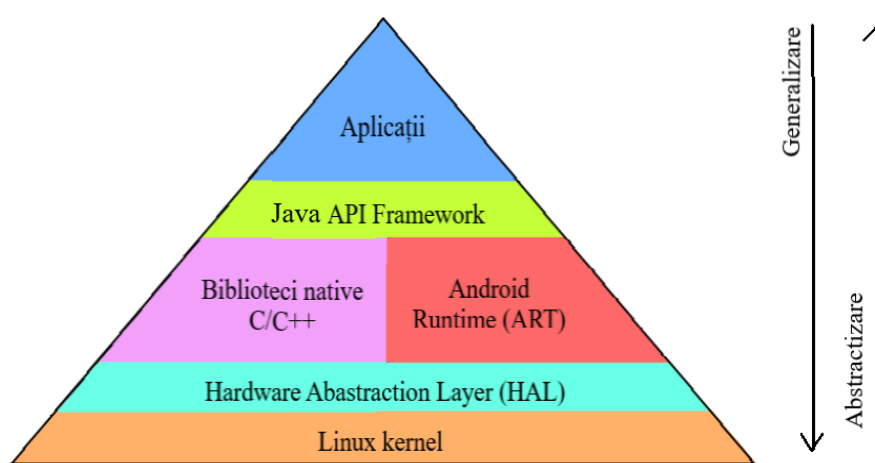


Figura 4.1. Stiva software Android

Sistemul de operare Android se bazează pe *Linux kernel*. Acesta facilitează interacțiunea dintre componentele hardware și software. Fiind de asemenea un produs al unei inițiative *open-source*, atât marii dezvoltatorii dispozitive mobile cât și programatorii amatori au putut experimenta și crea propriile drivere și implementări personalizate. Un alt avantaj al utilizării

<sup>4</sup> <https://www.businessofapps.com/data/android-statistics/>

<sup>5</sup> <https://www.theverge.com/2021/5/18/22440813/android-devices-active-number-smartphones-google-2021>

Linux este din punct de vedere al securității. Orice propunere de modificare adusă kernel-ului este riguros testată de comunitate formată din peste 4000 de colaboratori înainte de fi făcută publică.

Nivelul de abstractizare HAL reprezintă o serie de interfețe care expun componentele hardware (display, procesor, modem LTE, memorie, camera) de la diverși producători, la framework-ul Java. Folosind HAL, nu mai este necesar modificarea kernel-ului pentru a putea inter-opera diferitele componente hardware. În raport cu kernel-ul, HAL are un grad mult mai mare de specificitate, optimizând utilizarea hardware-ului și asigurând portabilitatea aplicațiilor sau, altfel spus, independența față de dispozitiv a aplicațiilor Android.

*Android Runtime*(ART) este un mediu de execuție care permite crearea de instanțe unice pentru fiecare aplicație în parte. Odată cu instalarea unei aplicații, ART convertește întregul cod sursă în cod-mașină executabil (*ahead-of-time compilation*) pentru o utilizare minimă a resurselor dispozitivului la rulare. Alte beneficii includ *garbage collector*, suport pentru *debugging* și monitorizare a performanței. Multe dintre operațiile care au loc la *runtime* pot fi gestionate prin intermediul unor librării Java din Java API framework.

O parte din componentele platformei Android, în special HAL și ART, se bazează pe biblioteci C și CPP. Funcționalitățile acestor biblioteci native sunt expuse programatorilor de către Java framework. Mai mult, există posibilitatea de a folosi în mod direct cod C/C++ care să fie compilat cu *CMake* sau *ndk-build* prin intermediul NDK (*Native Development Kit*). Librăriile native pot fi apelate din codul sursă Java folosind JNI (*Java Native Interface*) Avantajul NDK este că permite programatorilor să reutilizeze cod de C++ și deschide calea către optimizarea unor procese care folosesc intens procesorul.

Java API Framework oferă componente modulare și reutilizabile care ușurează dezvoltarea aplicațiilor Android. Prin intermediul acestui framework pot fi gestionate nivelele inferioare ale piramidei din *Figura 4.1*. Cu cât gradul de generalitate este mai ridicat cu atât posibilitățile de a le manipulare folosind framework-ul devin mai reduse. API framework gestionează controalele grafice, resursele grafice (fișiere *layout*), notificările, activitățile și datele preluate din alte aplicații sau surse externe.

Aplicațiile preinstalate oferă utilizatorului Android funcționalități elementare precum mesagerie, telefonie, acces la internet, tastatură, galerie foto, etc. Ulterior aceasta pot fi înlocuite prin intermediul unui serviciu dedicat de distribuție al aplicațiilor mobile precum Google Play sau instalând aplicații livrate în format .apk (*Android Application Package*). Din

motive de securitate singura aplicație preinstalată care nu poate fi modificată este cea pentru setările de sistem<sup>6</sup>.

#### **4.2. Bază de date non-relațională**

Pentru a gestiona toate datele utilizatorilor, a acțiunilor efectuate de aceștia și a locurilor de parcare am optat pentru utilizarea bazei de date Firestore Cloud, parte a platformei de dezvoltare Google Firestore. Principalul avantaj al unei baze de date NoSQL în general și a Firestore în particular este scalabilitatea. Fiind o parte integrată a Google Cloud Platform, toate operațiunile de scriere și citire din bază de date sunt optimizate pentru a oferi maximul de performanță. Spre deosebire de bazele de date relaționale, performanța de execuție a unui interogări pe o colecție este independentă de numărul de documente din acea colecție.

Bazele de date în timp real se pretează aplicațiilor care operează cu un volum mare de date care sunt modificate frecvent. Această particularitate s-a dovedit utilă încă din perioadă de dezvoltare a aplicației, când modificările aduse bazei de date puteau fi urmărite direct în consola Firestore fără a mai fi nevoie de interogarea manuală așa cum s-ar fi făcut în SQL prin „Select \* from”.

Mai mult, bazele de date NoSql sunt mult mai prietenoase cu dezvoltatorii începători, deoarece nu implică menținerea unor relații strânse între colecții și se evită utilizarea unor operații complexe precum cele de *Join*.

---

<sup>6</sup> Unele companii livrează dispozitivele mobile produse cu versiuni de Android modificate care conțin aplicații de mică utilitate pentru utilizator, dar care din motive comerciale nu pot fi dezinstalate. Aceste aplicații sunt cunoscute sub denumirea generală de „bloatware”.

park-69074	Parking Spots	7h9oc0tlzHZ2cWlvkeeOgVU3mHQ2
+ Start collection	+ Add document	+ Start collection
Parking Spots >	7h9oc0tlzHZ2cWlvkee	+ Add field
User Location	9zh0th40mcQB0iHYNy	available: true
Users	Ci3ndzB1WGj2S6MZ8E	availableUntil: July 11, 2021 at 4:00:00 PM UTC+3
	EPASY1vIHmJUUVLkvd	geoPoint: [45.8076531493041° N, 21.320660188794136° E]
	FRH7HWq6FML4o211RK	user
	F1FdWjkUtDgLBptadc	telefon: "0700 000 000"
	HgucJuK866o4XiFJGF	email: "f@f.com"
	ILjJM58uBuCEW4wzDGI	userId: "7h9oc0tlzHZ2cWlvkeeOgVU3mHQ2"
	LlKs7ywxIErelgYtgja	username: "f"
	PtuZIGDz3anacjWBjC	
	TxSs7WJC0EXPAFL6q2	
	UF33FLMdVQqjjkYfD3	

Figura 4.2. Gestionarea utilizatorilor în Firebase, colecția „Parking Spots”

În Figura 4.2 se poate observa faptul că baza de date conține 3 colecții de date. O colecție poate fi imaginată ca o tabelă sau o alăturare de mai multe tabele dintr-o bază de date relațională. Informațiile despre un loc de parcare se regăsesc într-un document cu ID unic în colecția de documente „Parking Spots”. Un loc de parcare posedă o variabilă booleană de disponibilitate (*available*) care are valoare *true* dacă locul de parcare este disponibil pentru închiriere, sau *false* dacă nu este. Variabila *availableUntil* indică data până la care locul este disponibil, iar variabila de tip *GeoPoint* indică locația exactă. *User* este o variabilă de tip *map* care indică proprietarul locului de parcare și conține date ce se regăsesc și în colecția „Users”.

O altă funcționalitate a Firebase se numește Cloud Functions și permitea crearea de acțiuni de manipulare automată a bazei de date. În cazul de față s-a dorit crearea unei funcții care la fiecare 2 minute să parcurgă colecția „Parking Spots”, iar în cazul în care dată curentă depășește data din variabila *availableUntil*, variabila *available* este setată pe *false*. Funcția folosită a fost scrisă în limbajul JavaScript, iar codul poate fi consultat în Anexa 1.

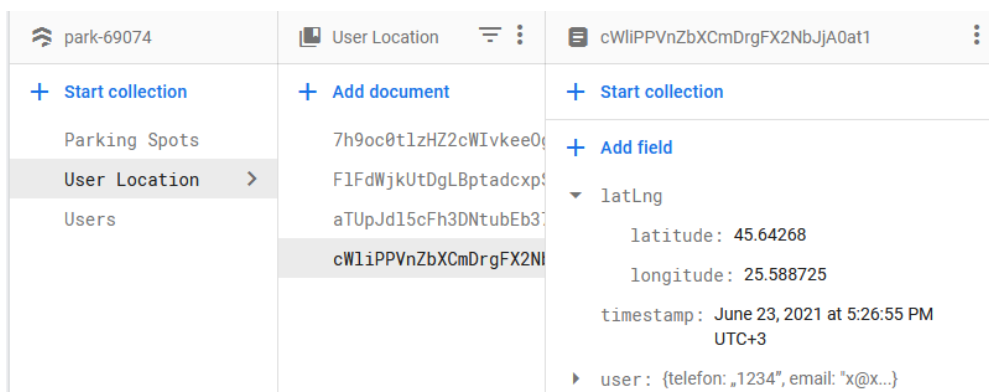


Figura 4.3. Gestionarea utilizatorilor în Firebase, colecția „User Location”

În colecția „User Location” este salvată ultima locație a utilizatorului și momentul înregistrării ei. Aceste informații sunt utile în cazul în care semnalul GPS este slab sau în cazul în care utilizatorul iese și se reconectează la aplicație, locația lui fiind afișată instant până la momentul unei noi actualizări a locației.

Firebase oferă un modul de autentificare și gestionare a datelor de utilizator numit Autetification. El facilitează stocarea în siguranță a datelor prin metode dezvoltate și testate de programatorii experimentați de la Google. Un dezvoltator de aplicații web sau mobile nu mai trebuie să-se preocupe de adaptori pentru servicii de autentificare sau să încropească mecanisme de criptare a datelor insuficient testate și predispuse abuzului. Astfel, tot ce rămâne de făcut este implementarea funcționalităților oferite prin intermediul *Firebase Android SDK*, precum autentificare, gestionarea datelor în Firestore Cloud, statistici sau declanșatoare de evenimente.

Mai jos este o imagine din consolă cu datele utilizatorilor care s-au conectat în aplicația Android folosind adresa de email și parola. De asemenea, se poate opta pentru autentificarea cu numărul de telefon, Facebook, Gmail, Microsoft Mail și multe altele. Fiecărui utilizator îi corespunde un ID unic generat automat. Codul de înregistrare a utilizatorilor în baza de date poate fi consultat în Anexa 2.



# Authentication

Users

Sign-in method

Templates

Usage

Search by email address, phone number, or user UID






Identifier	Providers	Created	Signed In	User UID <span>↑</span>
f@f.com		Apr 14, 2021	May 20, 20...	7h9oc0tlzHZ2cWlvi
ddd@d.com		Apr 2, 2021	Apr 2, 2021	Bwl5fPxqGSZLdSN
a@a.com		Apr 5, 2021	May 9, 2021	FIFdWjkUtDgLBpta
d@d.com		Apr 1, 2021	Apr 14, 2021	KFCM6fq5tbf4aSTF
b@b.com		Apr 7, 2021	Apr 7, 2021	aTUpJdl5cFh3DNtu

Figura 4.4. Gestionarea utilizatorilor în Firebase

### 4.3. API-uri și kituri de dezvoltare software

Pe lângă API-ul de conectare la baza de date non-relațională care face parte din kitul de dezvoltare *Firebase Android SDK*, aplicația se folosește de alte două API-uri pentru a-și spori funcționalitatea.

*Google Places API* este foarte versatil în cazurile în care este nevoie de informații cu privire la o locație anume. În activitatea de punere spre închiriere („ParkSharingActivity”) în partea de sus a ecranului se află un câmp de tipă *EditText* care odată apăsăat deschide o activitate prin care utilizatorul poate căuta o anumită locație. Pentru fiecare tastă apăsată sunt oferite sugestii de locații posibile. Astfel, cel care dorește să își pună locul de parcare spre închiriere îl poate găsi ușor pe hartă folosind căutarea locației sau folosind funcția GPS a telefonului.

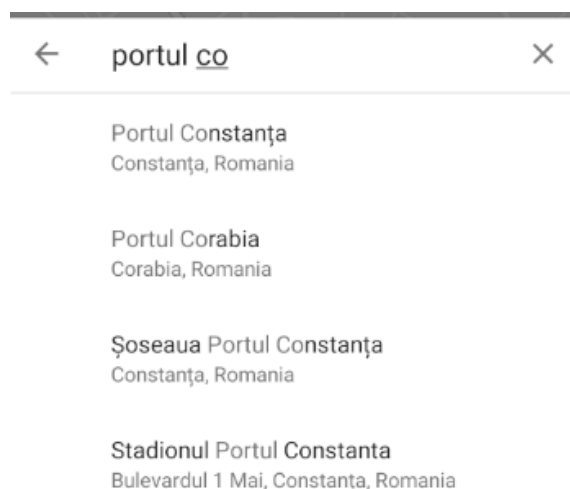


Figura 4.3. Activitatea de căutare a locațiilor

Fundamental pentru funcționarea aplicației este *Maps Android SDK* care permite utilizarea și manipularea hărților în interiorul fragmentului „MapsFragment”. „MapsFragment” este folosit atât activitatea de principală în care sunt afișate locurile de parcare disponibile („MainActivity”), cât și în activitate de punere spre închiriere a locurilor de parcare („ParkSharingActivity”). Designul hărților este unul modern și familiar utilizatorilor datorită faptului ca sunt folosite de majoritatea site-urilor și aplicațiilor.

Framework-ul dedicat componentelor grafice, Material Design, oferă posibilitatea folosirii unor șabloane grafice moderne care respectă recomandările Google. Componenta tip *snackbar* a fost folosită în activitatea de înregistrare pentru a comunica utilizatorilor eventualele erori apărute la conectare, iar componenta *BottomSheetDialog* este folosită în activitatea principală pentru afișarea adresei corespunzătoare fiecărui loc de parcare de pe hartă. În figura de jos se poate observa fereastra de dialog care apare ca urmare a apăsării pe iconița corespunzătoare unui loc de parcare.

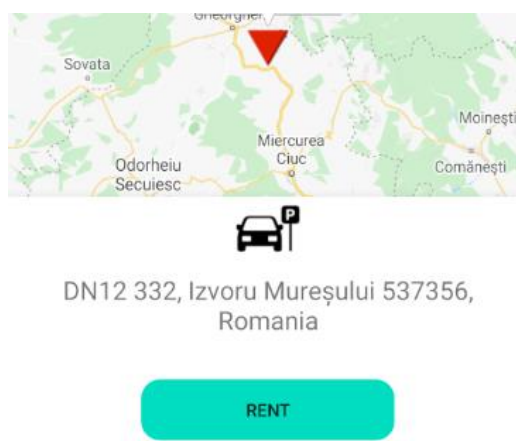


Figura 4.4. *BottomSheetDialog* în activitatea principală

Am folosit o dependență externă de pe GitHub în cadrul activității de selectare a intervalului de disponibilitate („DatePickerActivity”). Astfel, am putut folosi un *Timepicker* personalizat<sup>7</sup> cu ajutorul căruia pot fi alese doar date viitoare și o granularitate maximă de 15 minute.

---

<sup>7</sup> Codul sursă se găsește pe GitHub la: <https://github.com/florent37/SingleDateAndTimePicker>

Until when would you like to share the parking space?

02  
03 00  
Today 04 15 AM  
Wed 30 Jun 05 30 PM  
Thu 1 Jul 06 45  
Fri 2 Jul 07  
Sat 3 Jul 08

SAVE

Figura 4.5. TimePicker personalizat din activitatea DatePickerActivity

#### 4.4. Interfața grafică cu utilizatorul

În acest subcapitol va fi prezentat interfața grafică cu utilizatorul a aplicației Android pentru fiecare activitate în parte. Termenul „activitate” în Android corespunde unui set de elemente grafice care sunt afișate la un moment dat pe ecran și cu care se poate interacționa.

- **Activitatea de înregistrare a noilor utilizatori**

Park

Email Registration

Email

Telephone

Password

Confirm Password

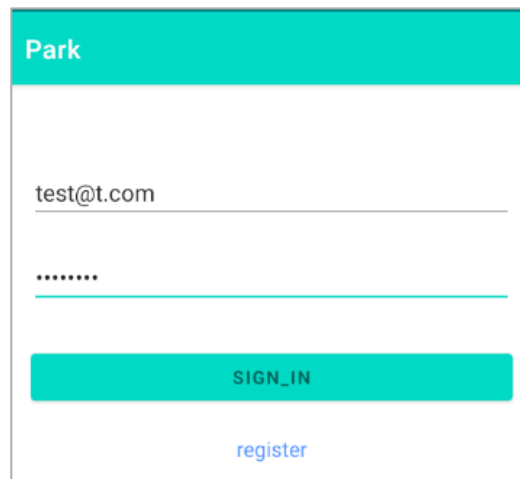
☐ I accept the [terms and conditions](#)

REGISTER

Figura 4.6. Activitatea de înregistrare

Activitatea de înregistrare este minimalistă din punct de vedere vizual. Utilizatorul trebuie să introducă adresa de email, numărul telefonul și parola dorită. În cazul în care nu respectă cerințele minime de corectitudine a inputului sau nu acceptă termenii și condițiile va fi atenționat prin intermediul unui *Snackbar*. Odată înregistrat, utilizatorul este redirecționat spre activitatea de conectare.

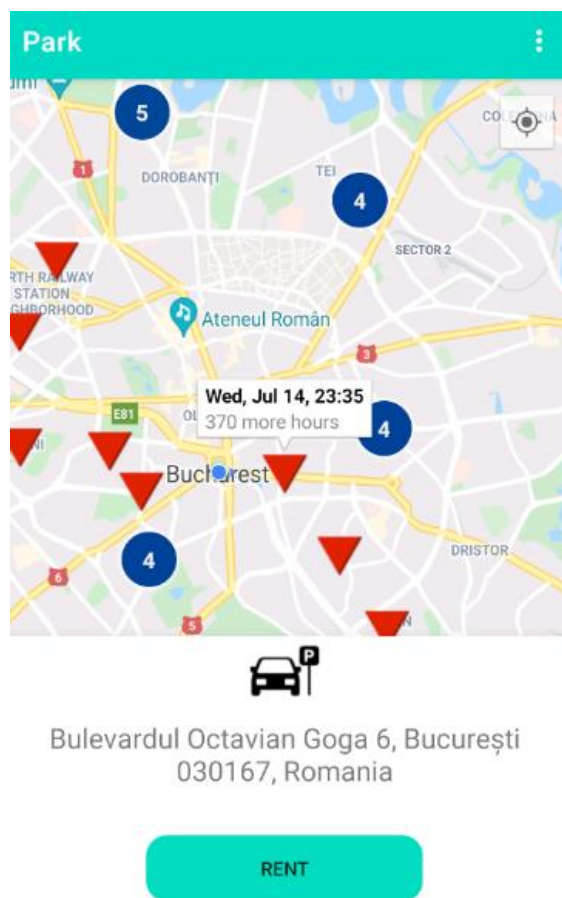
- **Activitatea de conectare**



*Figura 4.7. Activitatea de conectare*

În această activitate utilizatorul se poate conecta în aplicație folosind emailul și parola înregistrate. În cazul în care nu dispune deja de un cont poate opta spre crearea unui, apăsând pe link-ul către activitatea de înregistrare.

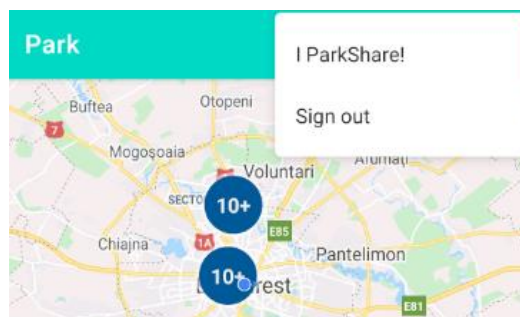
- **Activitatea principală**



*Figura 4.8. Activitatea de conectare + fereastră de dialog deschisă*

Utilizatorul interacționează cu activitatea principală imediat după conectare. În această activitate el poate vizualiza locurile de parcare disponibile spre închiriere. Apăsând pe iconița corespunzătoare locului de parcare, fereastra de dialog cu opțiunea de închiriere apare în partea jumătătea inferioară a ecranului.

Meniul tip *drop-down* din partea dreaptă-sus oferă opțiunea de deconectarea („Sign out”) și opțiunea de punere spre închiriere a locului de parcare personal („I ParkShare!”).



*Figura 4.9. Meniul activității principale*

- Activitatea de punere a locului de parcare spre închiriere

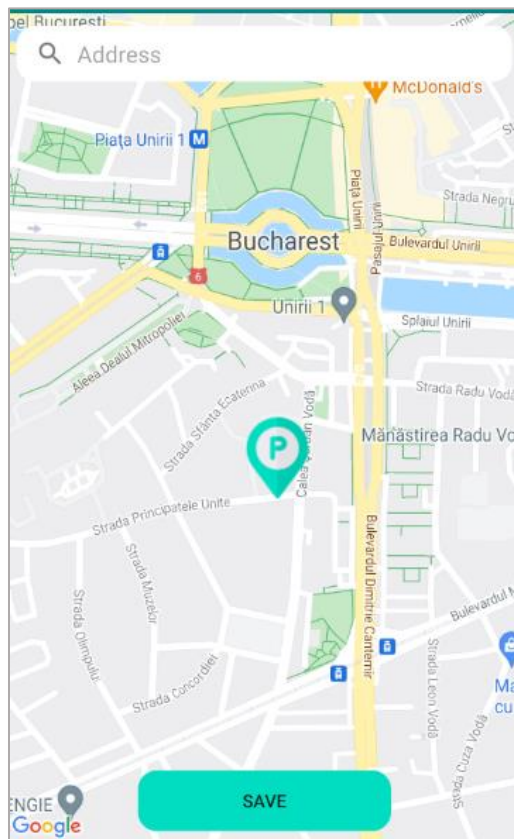


Figura 4.10. Selectarea locației unui loc de parcare

Această activitate permite utilizatorului punerea locului de parcare spre închiriere într-un mod intuitiv. Iconița corespunzătoare locului de parcare este poziționată în primă fază pe locația reală a utilizatorului. El poate folosi funcția de căutare pentru a găsi o locație aproximativă. Mecanismul de funcționare al activității de căutare inteligentă a adreselor este explicat în subcapitolul anterior, precum și modul în care este utilizat fragmentul de hartă. Folosind funcția de mărire a hărții și *drag-and-drop* se poate selecta cu precizie locația locului de parcare.

Odată apăsat butonul „SAVE” este apelată activitatea de selectare a intervalului de disponibilitate (Figura 4.5), iar ulterior se revine la activitatea principală.

## 5. Concluzii

Companiile din zona economiei colaborative s-au bucurat de interes crescut din partea publicului și există așteptări rezonabile că trendul se va menține și chiar accentua în anii următori. Adopția platformele de *sharing* este mult mai pronunțată în Europa occidentală și America de Nord, în timp ce în România rămâne un trend emergent și prea puțin valorificat în piață. Acest fapt este valabil și în cazul specific al aplicațiilor de *park-sharing*. În timp ce în țări precum UK sunt folosite la scară largă și sunt prezente în piață de mai bine de 10 ani, în România abia în ultimii 2 ani a apărut o inițiativă privată cu activitate modestă.

Pe fondul unei supra-aglomerări cu mașini a marilor orașe, penuria de locuri de parcare și costul prohibitiv de construire a unor noi parcuri, este sustenabil din punct de vedere economic optimizarea utilizării locurilor de parcare deja existente.

## Bibliografie

1. Android Developers. (2020). *Get started with the NDK*.  
<https://developer.android.com/ndk/guides>
2. Android Developers. (2021). *Platform Architecture*.  
<https://developer.android.com/guide/platform>
3. Android Open Source Project. (2020). *Kernel*.  
<https://source.android.com/devices/architecture/kernel>
4. Chege, J. (2021). *Implementing Bottom Sheet Dialogs using Android Studio*.  
Section.Io. <https://www.section.io/engineering-education/bottom-sheet-dialogs-using-android-studio/>
5. Digi24. (2020, January 25). *Studiu: Avem prea multe mașini. În București sunt 1,4 milioane de autoturisme înmatriculate*. Digi24.  
<https://www.digi24.ro/stiri/economie/studiu-avem-prea-multe-masini-in-bucuresti-sunt-14-milioane-de-autoturisme-inmatriculate-la-18-milioane-de-locuitori-1250050>
6. European Federation for Transport and Environment. (2018). *CO2 emissions from cars: the facts*, p. 40 .  
[https://www.transportenvironment.org/sites/te/files/publications/2018\\_04\\_CO2\\_emissions\\_cars\\_The\\_facts\\_report\\_final\\_0\\_0.pdf](https://www.transportenvironment.org/sites/te/files/publications/2018_04_CO2_emissions_cars_The_facts_report_final_0_0.pdf)
7. GeeksforGeeks. (2017). *Unified modeling language* .  
<https://www.geeksforgeeks.org/unified-modeling-language-uml-activity-diagrams/>
8. Hamari, J., Sjöklint, M., & Ukkonen, A. (2015). *The sharing economy: Why people participate in collaborative consumption*. Journal of the Association for Information Science and Technology, 67(9), 2047–2059. <https://doi.org/10.1002/asi.23552>
9. Ivanov C (2019). *Primăria Capitalei plătește 14.500 de euro/loc de parcare pentru construirea parcării park&ride de 500 de locuri de la capătul Șoselei Pantelimon*. HotNewsRo.  
[https://www.stiri-administratie\\_locala-23305246-foto-primaria-capitalei-plateste-14-500-euro-loc-parcare-pentru-construirea-parcarii-parkride-500-locuri-capatul-soselei-pantelimon.htm](https://www.stiri-administratie_locala-23305246-foto-primaria-capitalei-plateste-14-500-euro-loc-parcare-pentru-construirea-parcarii-parkride-500-locuri-capatul-soselei-pantelimon.htm)
10. Knowledge@Wharton. (2015). *How Green is the Sharing Economy?* Wharton School of the University of Pennsylvania. <https://knowledge.wharton.upenn.edu/article/how-green-is-the-sharing-economy/>
11. PriceWaterhouseCoopers (PWC). (2015a). *The sharing economy - Consumer intelligence series*. Pwc.com



<https://www.pwc.com/us/en/technology/publications/assets/pwc-consumer-intelligence-series-the-sharing-economy.pdf>

12. PriceWaterhouseCoopers (PWC). (2015b). *Sharing or paring? Growth of the sharing economy*. Pwc.com <https://www.pwc.com/hu/en/kiadvanyok/assets/pdf/sharing-economy-en.pdf>
13. Schaefer, L. (2021). *NoSQL vs SQL databases*. MongoDB. <https://www.mongodb.com/nosql-explained/nosql-vs-sql>
14. Stevenson, D. (2020). *The top 10 things to know about Firestore when choosing a database for your app*. Medium.com. <https://medium.com/firebase-developers/the-top-10-things-to-know-about-firestore-when-choosing-a-database-for-your-app-a3b71b80d979>
15. Techopedia. (2011). *Hardware abstraction layer (HAL)*. <https://www.techopedia.com/definition/4288/hardware-abstraction-layer-hal>
16. Wall-Street.ro, (2020, September 30). *Analiză: Cât de rău stă România la locurile de parcare și ce primărie le monetizează cel mai bine*. Wall-Street.Ro. <https://www.wall-street.ro/articol/Auto/261412/analiza-cate-locuri-de-parcare-publice-sunt-pentru-cele-8-milioane-de-masini-inmatriculate-in-romania.html#gref>

**Notă:** toate link-urile au fost accesate cu succes la data de 28.06.2021

## Anexa 1

Funcție în *cloud* pentru actualizarea valorii variabilei *available* atunci când perioada de disponibilitate este depășită. Apelul funcției are loc la fiecare 2 minute (această valoare este aleasă arbitrar). Actualizările au loc pe valorile variabilelor din documentele colecției „Parking Spots”.

Modulele de NodeJS, *firebase-functions* și *firebase-admin*, sunt necesare.





```
const functions = require('firebase-functions');
const admin = require('firebase-admin');
admin.initializeApp();

exports.scheduledFunction = functions.pubsub
  .schedule('every 2 minutes')
  .onRun(async () => {
    const currentTime = new Date();

    const data = await admin.firestore()
      .collection("Parking Spots")
      .where("available", "==", true)
      .where("availableUntil", "<", currentTime)
      .get();

    return Promise.all(data.docs.map(
      (item) => item.ref.set({ available: false }, { merge: true })));
  });
```

În consola de log-uri a Firebase Functions se poate observa că funcția este apelată în mod corect cu o frecvență de 2 minute.

4:48:01.434 PM		scheduledFunction	Function execution started
4:48:01.704 PM		scheduledFunction	Function execution took 270 ms, finished with status: 'ok'
4:50:00.359 PM		scheduledFunction	Function execution started
4:50:00.635 PM		scheduledFunction	Function execution took 277 ms, finished with status: 'ok'

## Anexa 2

Scrierea și citirea din baza de date Firebase se realizează foarte ușor prin intermediul kitului de dezvoltare, *Firebase Android SDK*. Mai jos am adăugat o funcție din cadrul activității de înregistrare a noilor utilizatori. Pe lângă parolă, fiecare utilizator posedă o adresă de mail, un nume derivat din adresa email și un ID generat automat de modulul *Authentication* al Firebase. Datele fiecărui utilizator corespund unei document unic din colecția „Users”.

Utilizatorul este avertizat printr-un snackbar (componentă grafică din *Material Design*) despre eventualele erori de înregistrare. Dacă înregistrarea în sistem are loc cu succes, utilizatorul este redirecționat către activitatea de conectarea unde trebuie să reintroducă parola aleasă.

```
public void registerNewEmail(final String email, String password){

    FirebaseAuth.getInstance().createUserWithEmailAndPassword(email, password)
        .addOnCompleteListener(new OnCompleteListener<AuthResult>() {

        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if (task.isSuccessful()){
                User user = new User();
                user.setEmail(email);
                user.setUsername(email.substring(0, email.indexOf("@")));
                user.setUserId(FirebaseAuth.getInstance().getUid());

                DocumentReference newUserRef = FirebaseFirestore
                    .getInstance()
                    .collection(getString(R.string.collection_users))
                    .document(FirebaseAuth.getInstance().getUid());

                newUserRef.set(user).addOnCompleteListener(new OnCompleteListener<Void>() {

                @Override
                public void onComplete(@NonNull Task<Void> task) {
                    if(task.isSuccessful()){
                        redirectLoginScreen();
                    }else{
                        View parentLayout = findViewById(android.R.id.content);
                        Snackbar.make(parentLayout, "Something went wrong.",
                            Snackbar.LENGTH_SHORT).show();
                    }
                }
                });
            }else {
                View parentLayout = findViewById(android.R.id.content);
                Snackbar.make(parentLayout, "Something is wrong.", Snackbar.LENGTH_SHORT).show();
            }
        }
    });
}
```