



**UNIVERSITATEA TEHNICĂ**  
DIN CLUJ-NAPOCA

**FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE**  
**DEPARTAMENTUL CALCULATOARE**

## **PLATFORMA DE E-COMMERCE “SMALL BUSINESSES” PENTRU MICII PRODUCĂTORI**

**LUCRARE DE LICENȚĂ**

Absolvent: **Rona DUMITRESCU**

Coordonator  
științific: **Conf. Univ. Dr. Paulina MITREA**

**2021**

---

## Cuprins

<b>Capitolul 1. Introducere .....</b>	<b>1</b>
1.1. Contextul proiectului .....	1
1.2. Structura lucrării .....	4
<b>Capitolul 2. Obiectivele Proiectului .....</b>	<b>5</b>
2.1. Scopul proiectului .....	5
2.2. Cerințele sistemului .....	5
2.2.1. Cerințe funcționale .....	5
2.2.2. Cerințe non-funcționale .....	7
<b>Capitolul 3. Studiu Bibliografic.....</b>	<b>9</b>
3.1. Platforme web vs. Aplicații mobile .....	9
3.2. Platforme similare.....	10
3.2.1. Waze .....	10
3.2.2. Bursa de legume .....	12
3.2.3. Oferte sector legumicol .....	13
3.3. Analiza comparativă .....	14
<b>Capitolul 4. Analiză și Fundamentare Teoretică.....</b>	<b>16</b>
4.1. Cazuri de utilizare. Actorii sistemului .....	16
4.1.1. Oaspete .....	17
4.1.2. Utilizator – Client .....	17
4.1.3. Utilizator – Comerciant .....	18
4.1.4. Administrator.....	18
4.2. Tehnologii utilizate .....	18
4.2.1. Mediul de dezvoltare - Android Studio .....	18
4.2.2. Gradle .....	19
4.2.3. Limbaj de programare – Kotlin .....	19
4.2.4. Limbajul de marcare – XML .....	20
4.2.5. Google Cloud Platform.....	21
4.2.6. Google Maps API .....	21
4.2.7. Firebase.....	21
4.2.8. Adobe Illustrator .....	22
<b>Capitolul 5. Proiectare de Detaliu și Implementare.....</b>	<b>23</b>

---

5.1.	Proiectare software .....	23
5.1.1.	Diagrama generală arhitecturală .....	23
5.1.2.	Diagrama de desfășurare .....	24
5.1.3.	Diagrama de pachete .....	24
5.1.4.	Diagrama de secvențe .....	25
5.1.5.	Diagrama de comunicare .....	27
5.1.6.	Diagrama de stări (taskuri) .....	28
5.1.7.	Diagrama flux de activități (flowchart) .....	28
5.1.8.	Flux de date alternativ .....	30
5.2.	Baza de date .....	30
5.2.1.	Diagrama bazei de date.....	30
5.2.2.	Descrierea tabelelor .....	31
5.3.	Proiectarea elementelor grafice ale interfeței utilizator .....	32
5.3.1.	Diagrama de fragmente și conexiuni între acestea .....	32
5.4.	Componentele principale ale sistemului .....	36
5.4.1.	Harta .....	36
5.4.2.	Navigabilitatea.....	36
5.4.3.	Autentificarea și Înscrierea .....	38
5.4.4.	Liste și Filtre .....	39
5.4.5.	Procesul de Vânzare-Cumpărare .....	39
5.4.6.	Conexiunea cu baza de date.....	40
<b>Capitolul 6.</b>	<b>Testare și Validare.....</b>	<b>42</b>
6.1.	Cazuri de test .....	42
6.1.1.	Testarea unitară.....	42
6.2.	Scenarii de test.....	43
6.2.1.	Testare manuală .....	43
6.2.2.	Înregistrare teste Espresso .....	44
<b>Capitolul 7.</b>	<b>Manual de Instalare și Utilizare.....</b>	<b>47</b>
7.1.	Instalarea platformei .....	47
7.2.	Manual de utilizare .....	48
7.2.1.	Utilizare Actor Oaspete .....	49
7.2.2.	Utilizare Actor Administrator.....	50
7.2.3.	Utilizare Actor Client .....	52
7.2.4.	Utilizare Actor Comerçant .....	53

---

<b>Capitolul 8. Concluzii .....</b>	<b>54</b>
8.1. Rezultate obținute și contribuții personale .....	54
8.2. Dezvoltări ulterioare .....	54
8.2.1. Îmbunătățiri funcționalități existente.....	55
8.2.2. Funcționalități noi.....	55
<b>Bibliografie .....</b>	<b>57</b>
<b>Anexa 1. Lista Figurilor .....</b>	<b>59</b>
<b>Anexa 2. Lista Tabelelor .....</b>	<b>60</b>
<b>Anexa 3. Glosar de termeni .....</b>	<b>61</b>
<b>Anexa 4. Secvențe de cod sursă relevant.....</b>	<b>64</b>

## Capitolul 1. Introducere

### 1.1. Contextul proiectului

Revoluția tehnologică marchează o nouă eră, cu un impact asupra tuturor aspectelor vieții pe care foarte puțini l-au prevăzut. Din punct de vedere social, medical, și, subiectul acestei lucrări, economic, tehnologia modelează traiul omului modern. Datorită acestui fapt, comerțul s-a extins din mediul real în cel virtual, sub formă de comerț electronic. [2]

E-commerce se definește ca fiind activitatea de vânzare-cumpărare de bunuri și servicii din mediul online. Deși la începutul anilor '90 existau mai multe firme care au desfășurat o activitate de pionierat în acest domeniu, publicul larg nu a început să utilizeze acest serviciu decât la sfârșitul anilor '90, după accesul în masă la internet și după îmbunătățirea protocoalelor de securitate pentru tranzacții sigure.

Dintre primele astfel de companii, merită se menționăm companiile Amazon și eBay, ambele înființate la jumătatea anilor '90. Amazon a început ca o librărie online, fără constrângeri legate de spațiu, apoi și-a extins gama de produse la muzică, electronice, mobilă, jucării etc. Evoluția lor și a altor companii se poate urmări în graficul următor:

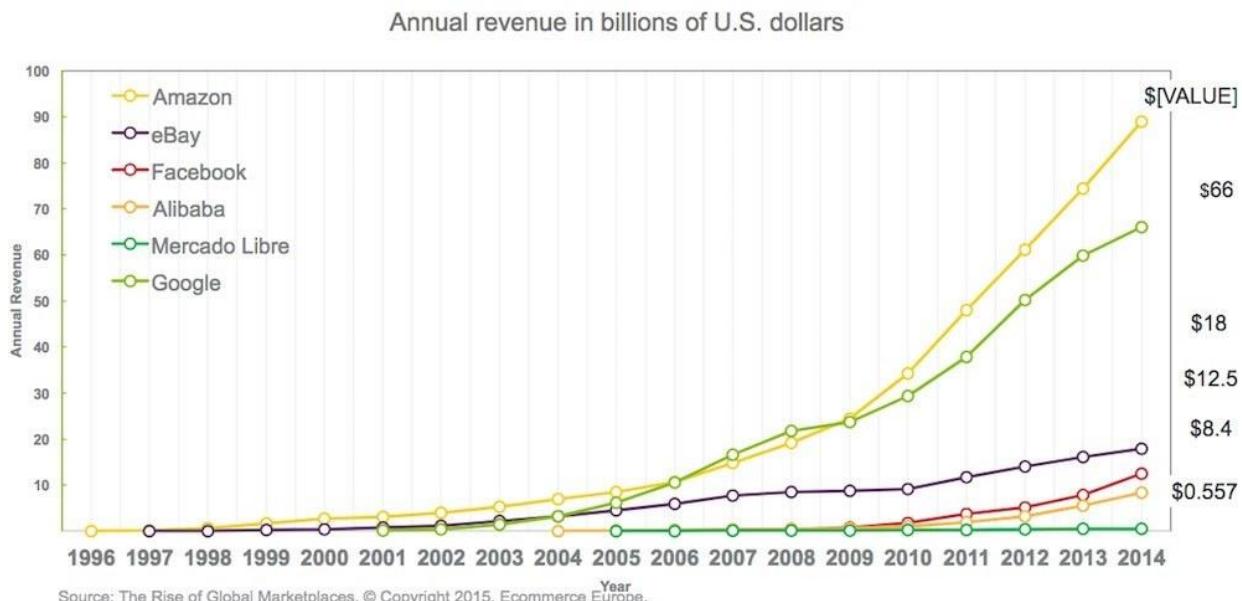


Figura 1.1 Creșterea vânzărilor în mediul online între 1996-2014 în SUA [17]

Comerțul de tip “mobile” s-a născut pentru prima dată în 1997, în Finlanda, la o inițiativă din partea companiei “Coca-Cola”. Comerțul de tip mobil înseamnă realizarea tranzacțiilor de vânzare-cumpărare în mediul online de pe dispozitive fără fir, precum telefonul mobil sau tableta. În ziua de astăzi, mobile e-commerce reprezintă 54% din toate acțiunile de comerț electronic din 2021. Pentru a înțelege dezvoltarea acestuia, s-a atașat următorul grafic:

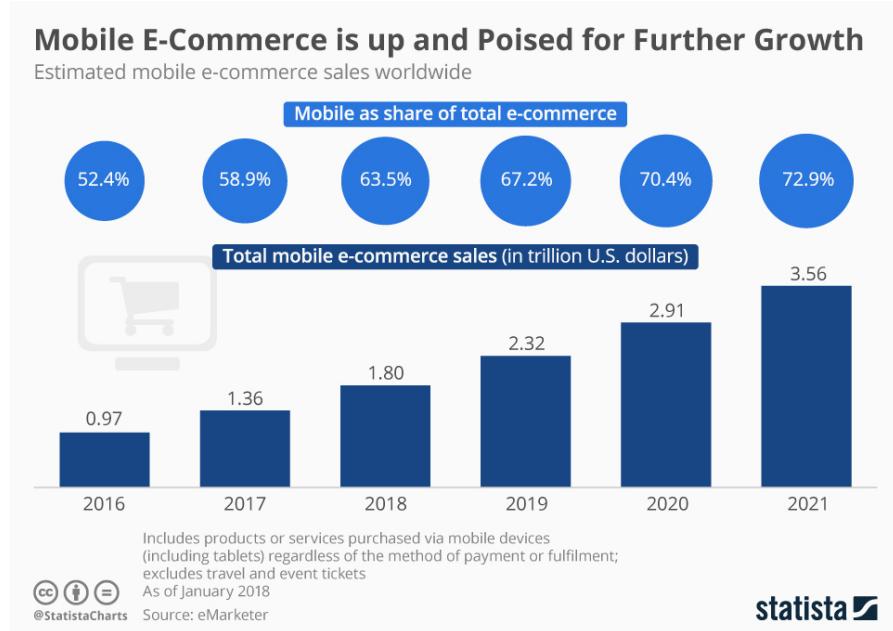


Figura 1.2 Creșterea comerțului electronic pe mobil între 2016-2021 [27]

Un alt sector de interes pentru această lucrare de diplomă este agricultura românească.

România este o țară cu un potențial de dezvoltare imens pe planul agricol și nu numai, deoarece se bucură de faptul că are toate tipurile de relief, de poziția geografică propice pentru o climă temperat-continențală și nu în ultimul rând, de resurse naturale variante. Toate aceste avantaje îi oferă posibilitatea de a avea produse concurente pe piață în următoarele domenii: panificație, carne, lactate, băuturi alcoolice și non-alcoolice, apicultură, conserve fructe-legume, produse morărit, semipreparate etc.

Din păcate, în ultimul timp, în presă au apărut titluri precum “Agricultura din România – impas dramatic”, “Agricultura română zdruncinată de pandemie și secată”, “Pierderi uriașe pentru fermieri” etc., titluri care trag un semnal de alarmă în ceea ce privește situația economică în care se află agricultura românească. Producțiile de dinainte de '89 au dus România pe culmile succesului financiar al sectorului agricol, însă modalitățile prin care s-a obținut acest lucru au fost și sunt de condamnat. Limitarea drepturilor cetățenilor, confiscarea bunurilor private și colectivizarea a toate sunt lucruri pe care este bine să le lăsăm în trecut.

În ce privește sănătatea, omul urban este în dezavantaj față de omul rural. Expus la mai mult stres, poluare, sedentarism și mâncare fast-food decât analogul său, duce o luptă mai serioasă pentru a se menține sănătos. În ce privește alimentația, în oraș, omul urban se poate aproviza cu produse românești din piețe, însă piețele nu pot îngloba întreagă gama de agricultori din țară.

Proiectul de licență se adresează nișei de comercianți care produc în gospodăriile lor mai mult decât pot consuma și, în momentul de față își scot produsele la vânzare pe mese, aproape de carosabil, iar noi vrem să schimbăm acest lucru. De asemenea, proiectul se adresează clientilor care doresc să mănânce sănătos, fie dintr-o poftă pe care o au când sunt la drum lung (de struguri, zmeură, afine etc.), fie dintr-o dorință de a se aproviziona regulat, direct de la producători de încredere din apropiere.

Scopul acestui proiect este de a realiza o platformă pentru comercializarea produselor naturale, crescute în grădinile micilor producători. Utilizatorii vor putea cumpăra alimente cu adevărat “bio” de la comercianții care, de obicei, au un stand cu produse de sezon în fața gospodăriei, contribuind la dezvoltarea economică a mediului rural și a domeniului agriculturii.

În articolul său „Promovarea produselor agroalimentare tradiționale românești în regiunea de dezvoltare nord-est”, Lucian Tanasa [14] atașează o hartă a produselor tradiționale românești la nivelul anului 2008, care ne poate ajuta să vizualizăm mai ușor specificul agroalimentare ale fiecărei regiuni din țară:

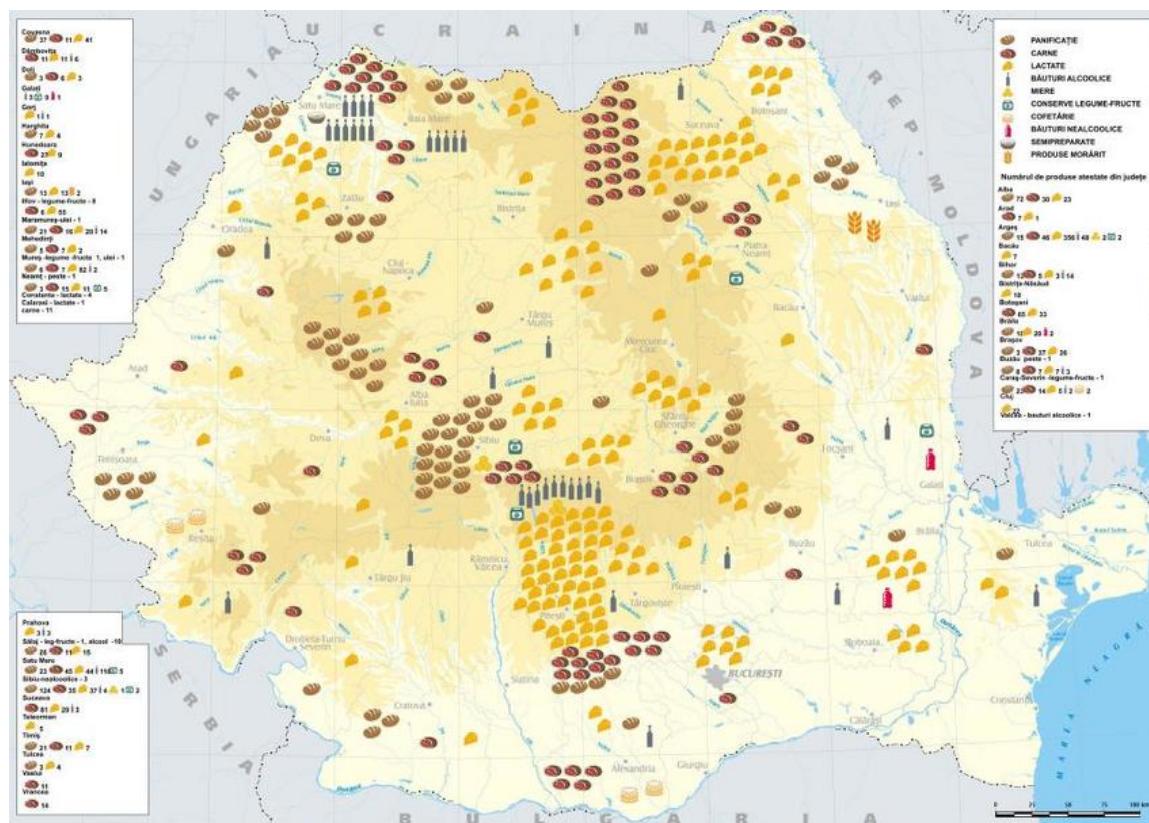


Figura 1.3 Harta produselor tradiționale din România la nivelul anului 2008 [4]

Nevoia de produse sănătoase a fost, este și mereu va rămâne indispensabilă pentru societate, de aceea, se vor căuta modalități prin care tehnologia să poată ajuta la împlinirea acestei nevoi primare.

### **1.2. Structura lucrării**

Descrierea conținutului și structurii lucrării vor fi detaliate în următoarele paragrafe.

În primul capitol, „Introducere”, este prezentată o scurtă descriere a contextului proiectului, care dorește să scoată în evidență evoluția în timp a e-commerce-ului, a mobile e-commerce-ului și situația comerțului produselor agroalimentare românești.

În următorul capitol, “Obiectivele proiectului”, sunt enumerate obiectivele propuse în urma aducerii la cunoștință a necesității unei astfel de platforme. Pe lângă obiective, sunt descrise și cerințele de sistem, clasificate în două categorii: funcționale și non-funcționale.

Capitolul al treilea, “Studiu bibliografic”, cuprinde descrierile platformelor similare, Waze, Bursa de legume și Ofertă Sector Legumicol, precum și un tabel comparativ al caracteristicilor și funcționalităților acestora și aplicația implementată. De asemenea, cuprinde și un studiu comparativ între platformele web și aplicațiile pentru telefonul mobil.

În capitolul al patrulea, “Analiză și fundamentare teoretică”, sunt prezentate în detaliu cazurile de utilizare și actorii sistemului (oaspete, utilizator-client, utilizator-comerciant și administrator), nu în ultimul rând, sunt prezentate pe larg tehnologiile utilizate, de la mediul de dezvoltare, limbaje, până la baza de date și alte componente.

În capitolul al cincilea, “Proiectare în Detaliu și Implementare”, sunt prezentate detaliile de implementare. Chiar dacă baza de date utilizată este nerelațională, s-a realizat și o diagramă corespunzătoare modelului conceptual al resursei de date prin prisma abordării relaționale. Ultima parte a acestui capitol cuprinde prezentarea pe larg a componentelor principale ale sistemului.

În capitolul al șaselea, “Testare și Validare”, se găsesc prezentate în detaliu câteva cazuri de test și scenarii de test.

În capitolul al șaptelea, “Manual de Instalare și Utilizare”, sunt cuprinse informații privind instalarea și utilizarea aplicației.

În ultimul capitol, “Concluzii”, au fost relatate contribuții personale pentru acest proiect și mai multe dezvoltări ulterioare, care ar îmbogăți și mai mult proiectul pe viitor.

## Capitolul 2. Obiectivele Proiectului

### 2.1. Scopul proiectului

Omul urban este în dezavantaj față de omul rural. Acesta este expus la mai mult stres, poluare, sedentarism și mâncare fast-food decât analogul sau și duce o luptă mai serioasă pentru a se menține sănătos. În ce privește alimentația, în oraș, omul urban se poate aproviziona cu produse românești din piețe, însă piețele nu pot îngloba întreagă gama de agricultori din țară.

Problema alimentelor cu gust de cauciuc din import, piețe supra-aglomerate, care nu pot îngloba toți comercianții din țară, inexistența unei platforme pentru comerțul fructelor și legumelor românești din împrejurimile orașelor afectează orășenii care își doresc un stil de viață sănătos și sătenii care cultivă mai mult decât pot consuma, iar urmarea, impactul, este consumul fructelor și legumelor fără gust, alterarea roadelor în grădină, dar și dezvoltarea slabă a economiei agriculturii românești. O soluție de succes ar fi crearea unei platforme de comerț pentru produsele naturale românești.

Pentru comercianții care produc în gospodăriile lor fructe și legume, precum și pentru clienții care își doresc produse agroalimentare cu gust, crescute în mod natural, fără acceleratori de creștere sau adaosuri, spre deosebire de produsele din această gamă din import, platforma este menită să asigure un mediu de vânzare-cumpărare pentru această nișă de produse, comercianți și clienți.

Proiectul de licență se adresează nișei de comercianți care produc în gospodăriile lor mai mult decât pot consuma și își scot produsele la vânzare pe mese, aproape de carosabil.

Scopul acestui proiect este de a realiza o platformă pentru comercializarea produselor naturale, crescute în grădinile micilor producători. Utilizatorii vor putea cumpăra alimente cu adevărat "bio" de la comercianții care, de obicei, au un stand cu produse de sezon în fața gospodăriei, contribuind la dezvoltarea economică a mediului rural și a domeniului agriculturii.

*Principalele obiective sunt:*

- Navigarea și localizarea utilizatorilor cu ajutorul unei hărți.
- Asigurarea unei modalități de comunicare online (via chat) și de plată online.
- Implementarea operațiilor de e-commerce – adăugare în coș, comandă, retur etc.
- Stimularea comerțului cu produse naturale românești.

### 2.2. Cerințele sistemului

#### 2.2.1. Cerințe funcționale

Cerințele funcționale sunt definite ca fiind totalitatea serviciilor și funcționalităților pe care sistemul trebuie să le presteze și a răspunsurilor și reacțiilor pe care este necesar să le ofere în diferite situații. Pe scurt, este vorba despre ceea ce se așteaptă utilizatorii ca sistemul să poată face. Acestea pot fi analizate mai îndeaproape în tabelul următor:

## Capitolul 2

Nr. CF	Descriere CF	Actor
CF 1	Înregistrare/Creare cont	Oaspete
CF 2.1	Navigare – prin “căutare destinație”	Oaspete + Utilizatori + Administrator
CF 2.2	Navigare – prin “near me”	Oaspete + Utilizatori + Administrator
CF 3	Vizualizare detalii produse și comercianți de la standuri	Oaspete + Utilizatori + Administrator
CF 4	Autentificare și delogare	Utilizatori
CF 5	Gestionare profil	Utilizatori
CF 6	Comunicare via chat	Utilizatori
CF 7	Oferire review-uri celorlalți utilizatori	Utilizatori
CF 8	Gestionare listă de produse și comercianți favoriți	Utilizatori
CF 9	Vizualizare istoric	Utilizatori
CF 10	Solicitare rezervare produse	Utilizatori
CF 11	Semnalare probleme administratori	Utilizatori
CF 12	Adăugare anunțuri	Utilizatori
CF 13	Gestionare detalii anunțuri	Utilizator - comerciant
CF 13.1	Gestionare produse și prețuri	Utilizator - comerciant
CF 13.2	Gestionare locații și program	Utilizator - comerciant
CF 14	Gestionare metode de plată	Utilizator - comerciant
CF 15	Gestionare utilizatorii din baza de date	Administrator
CF 16	Gestionare produse din baza de date	Administrator
CF 17	Rezolvare probleme semnalate	Administrator

Tabel 2.1 Tabel cu cerințele funcționale ale platformei de implementat

### 2.2.2. Cerințe non-funcționale

Totalitatea constrângerilor față de diversele caracteristici de funcționare și serviciile poartă numele de cerințe non-funcționale. Acestea se grupează în diverse categorii, dintre care le vom aminti pe cele mai relevante în următoarele secțiuni.

#### 2.2.2.1. Utilizabilitatea

Utilizabilitatea reprezintă măsura ușurinței cu care sistemul poate fi învățat și utilizat, a siguranței pe care o inspiră, a eficienței și eficacității, a atitudinii utilizatorilor în raport cu acesta. Printre atrbutele utilizabilității se numără ușurința de învățare a aplicației de către utilizatorii noi, rapiditatea cu care se familiarizează cu ea, eficiența cu care utilizatorii avansați se descurcă, predictibilitatea sistemului precum și multe altele.

Aplicația propusă are o interfață simplă, nu e încărcată, astfel încât utilizatorii să găsească cu ușurință informațiile și funcționalitățile de care au nevoie. Controalele sunt ușor de observat, cu imagini și descrieri cât mai sugestive, oferind productivitate sporită, odată ce utilizatorul e obișnuit cu aplicația.

#### 2.2.2.2. Disponibilitatea

Disponibilitatea este o caracteristică indispensabilă, deoarece atât timpul de răspuns, cât și viteza de transfer sunt importante pentru o experiență a utilizatorilor, mai ales atunci când aceștia sunt pe drum sau caută un produs necesar imediat. Această cerință este exprimată ca un procent din timpul de rulare al aplicației minus timpul în care aplicația nu poate fi accesată de utilizatori. O rată de disponibilitate dorită pentru sistemul nostru ar fi peste 99% în timpul zilei, obținută prin acoperirea posibilelor scenarii de eșec de la început și înăнд cont de faptul că aplicația folosește servicii Google și implicit, Firebase, care asigură o rată de disponibilitate ridicată.

#### 2.2.2.3. Portabilitatea

Sistemul dezvoltat este o aplicație mobile, deci cerința de portabilitate va fi îndeplinită inițial pentru orice dispozitiv Android. De asemenea, limbajul Kotlin permite o migrare facilă spre iOS, fiind gândit ca un limbaj multi-platformă. Într-o versiune ulterioară poate fi implementată și o versiune Web a aplicației, care ar asigura portabilitatea pe aproape orice device.

#### 2.2.2.4. Securitatea

Securitatea aplicațiilor mobile reprezintă totalitatea măsurilor luate pentru a mări siguranța protejării datelor, prin căutarea, rezolvarea și prevenirea vulnerabilităților de securitate.

Deoarece aplicația implică trimiterea de date, informații, tranzacții, unele date ale utilizatorilor, atât a comercianților, cât și a clientilor trebuie protejate. Astfel, fiecare utilizator trebuie să fie autentificat pentru a putea folosi aplicația. Acest lucru se face pe baza unui token, unic pe sesiune pentru un utilizator, existând și o perioadă de expirare a acestuia, pentru a nu putea fi folosit de către cei din exterior. Fără acest token, requesturile către server nu vor putea fi realizate. Mesajele din conversații vor fi încriptate end-to-end pentru a spori securizarea conversațiilor.

### *2.2.2.5. Scalabilitatea*

Scalabilitatea se referă la capacitatea de a depăși limitele de performanță prin adăugarea de resurse. Când performanța sistemului scade, hardware-ul suplimentar sau suplimentarea resurselor oferite de serviciile Google pe care aplicația le folosește ar putea rezolva problema.

## Capitolul 3. Studiu Bibliografic

### 3.1. Platforme web vs. Aplicații mobile

Încă din introducere am pus accentul pe dezvoltarea rapidă a aplicațiilor de comerț online pentru telefoane mobile prin diverse grafice și argumente, însă, înainte de a începe orice formă de implementare, se ridică întrebarea: De ce nu dezvoltăm o asemenea platformă pe web?

Deși pe un telefon mobil pot părea asemănătoare, ambele fiind la câteva click-uri distanță, dezvoltarea și utilizarea lor sunt diferite și studiile o arată. Pentru a căntări mai bine avantajele și dezavantajele celor două, s-a întocmit următorul tabel:

Platformă Web	Aplicație Mobilă
Nu sunt la fel de rapide ca aplicațiile mobile	Sunt mai rapide decât aplicațiile mobile
Conexiunea la internet este absolut necesară	Pot funcționa și fără conexiune la internet
Nu este nevoie să fie descărcate și instalate	Este necesar să fie descărcate
Sunt mai ușor de dezvoltat și de întreținut	Dezvoltarea și întreținerea este mai costisitoare decât în cazul platformelor web
Se vor actualiza automat	Apare problema actualizării periodice
Dat fiind faptul ca nu fac parte dintr-o bază de date precum AppStore, Google Play, sunt mai greu de descoperit	Sunt mai ușor de descoperit
Nu sunt la fel de sigure	Sunt mai sigure, datorită verificării și testării de către magazinele care le scot "pe piață"
Nu au acces la resursele sistemului.	Datorită accesului la resursele sistemului, disponă de mai multe funcționalități (necesare aplicației noastre)

Tabel 3.1 Avantajele și dezavantajele aplicațiilor mobile vs. web

Avantajul pentru care merită poate să luăm în considerare toate dezavantajele platformelor web în favoarea aplicațiilor mobile este faptul că înainte de toate, aplicațiile mobile sunt mult mai populare. Studiile arată, după cum se poate observa în următorul

grafic, că utilizatorii telefoanelor mobile petrec aproape 90% din timpul de navigare în aplicații și doar 10% în browsere.

## 90% OF TIME ON MOBILE IS SPENT IN APPS

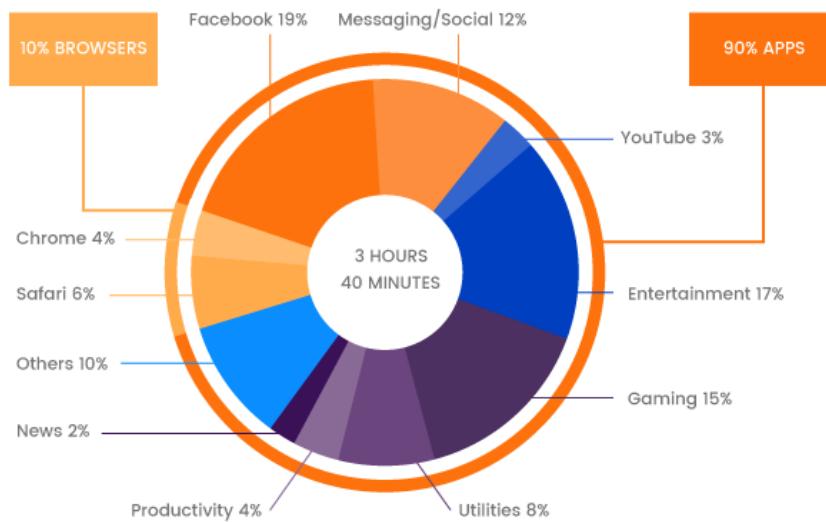


Figura 3.1 Timpul petrecut pe aplicațiile mobile vs. platformele web [26]

### 3.2. Platforme similare

Alternatiile pentru implementarea ideii prezentate, pe lângă dezvoltarea platformei propuse, sunt multiple, iar majoritatea se bazează pe aplicații similare cu cea prezentată.

O propunere ar fi contactarea unor platforme de navigație mai mari, precum Waze sau chiar Google Maps, în vederea dezvoltării unui feature nou, de localizare a comercianților mici, agricultorilor. Așa cum Google Maps afișează programul, site-ul și contactul diverselor supermarketuri, poate adăuga un nou feature/pin pentru fermieri, cu opțiunea de a administra un mod de vânzare-cumpărare a produselor online. Punctele tari ale acestor platforme sunt interfețele ușor de utilizat și poate cel mai important, numărul mare de utilizatori care le accesează, având în vedere că lucrarea se adresează unui grup de utilizatori la nivel național.

O altă opțiune de dezvoltare ar fi livrarea de la distanță prin Bolt sau Bla Bla Car. Premisa este aceeași, se contactează platformele care au servicii asemănătoare și se propune extinderea acestora printr-un serviciu nou – de comercializare a produselor alimentare produse în România. Atât acestor platforme, spre deosebire de cele menționate anterior, este transportul. Clientul poate beneficia de livrarea produselor acasă.

#### 3.2.1. Waze

“FreeMap Israel” a fost dezvoltată de către firma israeliană “Waze Mobile”, după care a fost cumpărată de către Google în anul 2013 și a devenit “Waze”. Aplicația este

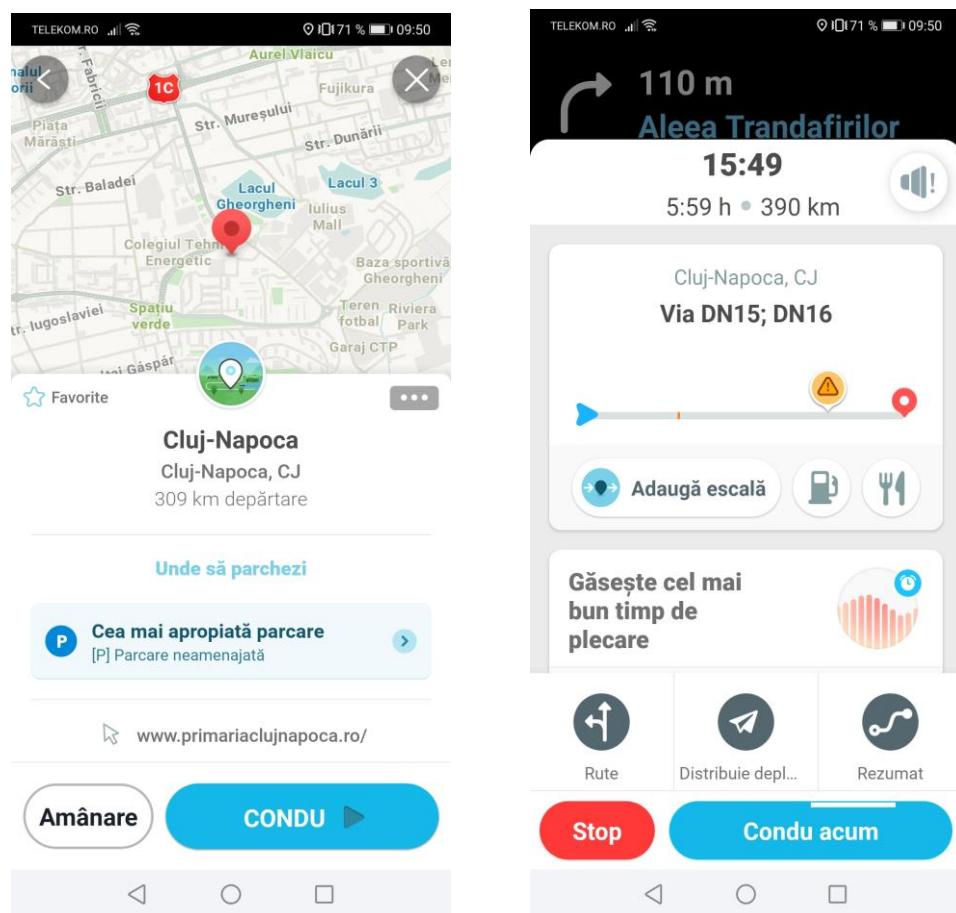
## Capitolul 3

disponibilă atât pe telefoanele mobile cu sisteme de operare Android și iOS, cât și pe calculatoare, atât timp cât au suport GPS.

Waze oferă informații de interes pentru utilizatori în timp real, precum: indicații precise în timpul călătoriei cu privire la drum, obstacole, trafic, radar, accidente etc. Acesta are și un serviciu de “carpooling”, care se referă la utilizarea în comun a automobilelor prin publicarea unui anunț pe o astfel de platformă de către șofer, precizând destinația și ora de plecare, dar și rezervarea unui loc de către un pasager care dorește să ajungă la aceeași destinație. Momentan, acest serviciu este indisponibil în România.

Sistemul de navigare este impecabil. La fiecare cursă este afișat timpul de sosire, durata călătoriei și numărul de kilometri până la destinație, cu posibilitatea de a alege diferite orașe intermediare sau diferite rute, în funcție de trafic. Pentru cei care utilizează aplicația mai des, Waze afișează istoricul și o listă cu locațiile favorite salvate anterior.

Deși Waze dispune de indicatori pe hartă pentru obiective turistice, instituții, parcuri și mai multe societăți comerciale, micii comercianți nu dispun de o identificare în cadrul aplicației. Prin aplicația implementată, autorul și-a propus să suplimească această lipsă, realizând o hartă navigabilă care informează utilizatorul despre acești comercianți locali.



## Capitolul 3

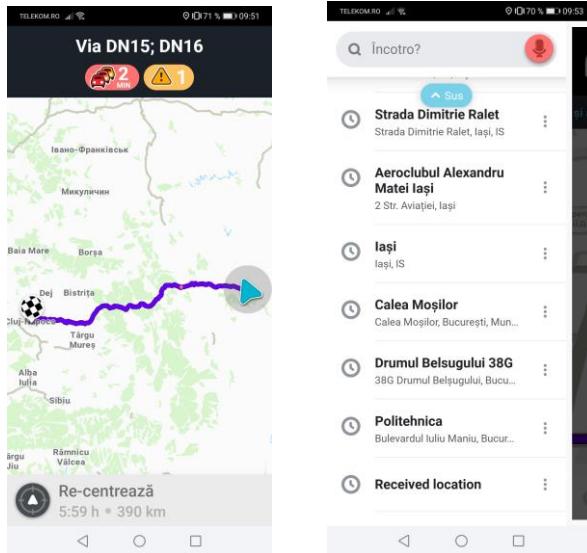


Figura 3.2 Capturi de ecran cu principalele funcționalități din aplicația Waze

### 3.2.2. Bursa de legume

“Bursa de legume” este un site online inițiat în zona Matca, Galați, pentru fermierii români, care produc predominant produse alimentare în cantități mari.

Așa cum este scris în ghidul website-ului, platforma “Bursa de legume” ține la curent cumpărătorii și vânzătorii cu prețurile en-gros din marile bazine legumicole din România: Matca, Brașov, București, Suceava în cadrul secțiunii Mercurial. [19] Bursa de legume oferă utilizatorilor posibilitatea de a posta anunțuri pentru a vinde o anumită cantitate de marfă și de a furniza câteva date de contact, pentru ca potențialii clienți să poată lua legătura cu ei. Anunțurile pot fi activate, dezactivate sau promovate premium în schimbul unei sume către platformă. [19]

Data/Ora	Produs	Pret (RON)	Judet	Localitate
25 iunie 2021	Dovelcel	min 0.5 - max 0.7	Galați	Matca
25 iunie 2021	Ardei lute	min 3.5 - max 4.0	Galați	Matca
25 iunie 2021	Castraveti	min 0.3 max 0.5	Galați	Matca
25 iunie 2021	Varata	min 3.0 - max 3.2	Galați	Matca

## Capitolul 3

The screenshot shows the homepage of the "Bursa de legume" platform. On the left, there's a green sidebar with a login form for "INSCRIERE MEMBRU" and "AUTENTIFICARE". Below it is a search bar with the placeholder "Cautare cheie". In the center, there's a search section titled "CERERE" and an offer section titled "OFERTA". The "CERERE" section shows a search dropdown for "Ordoneaza" and "Sorteaza", and a dropdown for "Tip anunt". Below these are two offers: "VÂNZARE FASOLE VERDE" (valid until July 20) and "VARZA ALBA" (valid until July 03). The right side features a red "MEMBRU PREMIUM" box with "AVANTAJE MEMBRU PREMIUM" (1. Afisare pe prima pagina, 2. Prioritate in cautari) and a "DEVINO MEMBRU PREMIUM" button. A sidebar on the right contains "INFORMATII UTILE" with articles like "BAZINUL LEGUMICOL MATCA | SPECIFIC, PROBLEME SI POTENTIAL DE DEZVOLTARE" and "CUM PROTEJAM CULTurile DE LEGUME DE TEMPERATURILE RIDICATE". At the bottom right is a link "VEZI TOATE ARTICOLELE >".

Figura 3.3 Platforma „Bursa de legume”

### 3.2.3. Oferte sector legumicol

Un alt website de interes pentru acest domeniu este site-ul “Oferte Sector Legumicol”, o platformă de comerț legumicol care oferă utilizatorilor posibilitatea de a posta anunțuri, bazate pe eligibilitatea din punct de vedere legală. Deși are un design minimalist, nu oferă suficiente informații pentru navigabilitate și un alt dezavantaj este ca actualizarea a diferitelor oferte și prețuri, poate fi anevoieasă pentru micii comercianți.

The screenshot shows the homepage of the "Oferte Sector Legumicol" website. The header includes a search bar with "www.mndr.ro/legume/" and navigation links for "Home", "Oferte producători", "Adaugă anunț", "Căută anunț", and "Termeni și condiții". The main content area has a dark green background with the text "OFERTE SECTOR LEGUMICOL". Below this is a section titled "Anunțuri" with a sub-section "IMPORTANT! Condiții pentru publicare oferte producători:" which lists requirements for producers. To the right is a sidebar for "Acces producători" with fields for "Nume utilizator" and "Parola", a "Tîne-mă minte" checkbox, and a "CREATORI" button. At the bottom is an "AUTENTIFICARE" section with links for "Creează un cont", "Ai uitat utilizatorul?", and "Ai uitat parola?".

## Capitolul 3

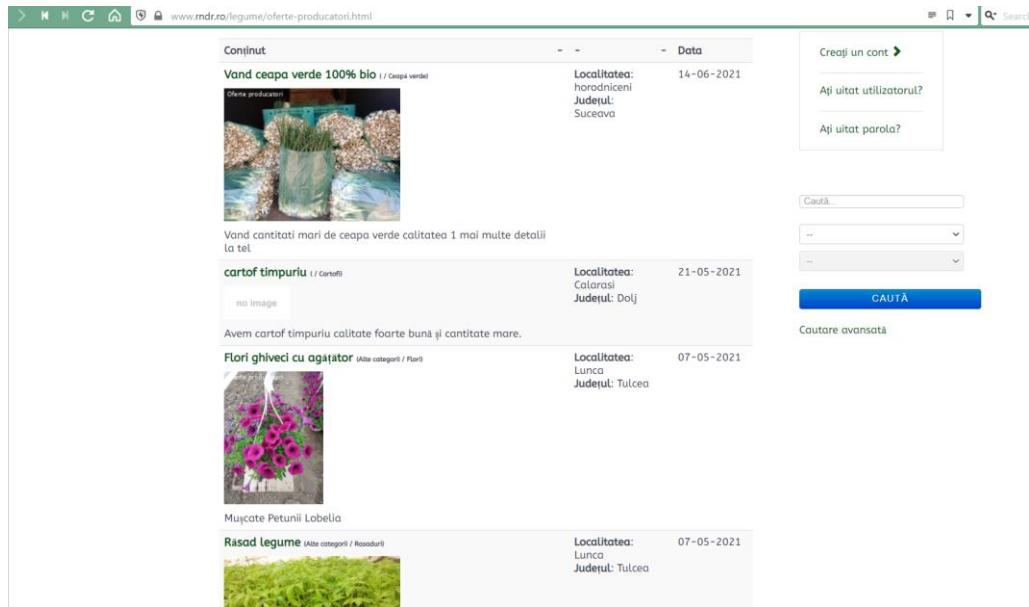


Figura 3.4 Platforma „Oferte Sector Legumic”

### 3.3. Analiza comparativă

Nevoia pronunțată pentru deschiderea unui astfel de market online a încercat să fie acoperită de diverse site-uri pilot, fără un impact la nivel național. Dintre aceste site-uri, se trec în revistă: “Agrihub” [16], care întrunește mai mult întreprinderi sub umbrela unui hub, „Rnrd” [24] sau „Oferte sector legumic”, care are un motor de cătare bazat mai mult pe produse decât pe locația acestora, și de asemenea, este unul dintre site-uri cu foarte puține anunțuri posteate din partea producătorilor/comerçanților. Ultimul site care are caracteristici similare cu cel propus în lucrarea prezentă și care se va trece în revistă este “Bursa de legume” [19]. O platformă cu un număr mare de utilizatori.

În următorul tabel, pe lângă faptul că platforma propusă este de tip mobile, iar, în afară de Waze, celelalte două menționate în paragraful anterior sunt de tip web, vor fi prezentate avantajele și dezavantajele descoperite comparând cele trei platforme cu platforma mobilă realizată conform obiectivelor proiectului și încărcată în GooglePlay sub denumirea “Made in Ro”.

Caracteristici	Made in Ro	Waze	Bursa de legume	Oferte sector legumic
Opțiune Autentificare	+	+	+	+
Bară de căutare	+	+	+	-
Gestionare anunțuri	+	-	+	+

### Capitolul 3

Listă Favorite	+	+	-	-
Vizualizare Istoric	+	+	-	-
Opțiune Adăugare Review	+	-	-	-
Semnalare probleme admin	+	+	-	-
Identificare pe hartă a producătorilor	+	-	+	+
Comunicare via Chat	+	-	-	-
Rubrică informații utile/știri	-	-	+	-
Livrare acasă	-	-	-	-
Comenzi vocale	-	+	-	-
Aplicație mobil	+	+	-	-
Preț min/max per produs	-	-	+	-

Tabel 3.2 Tabel cu avantajele și dezavantajele platformelor similare

## Capitolul 4. Analiză și Fundamentare Teoretică

### 4.1. Cazuri de utilizare. Actorii sistemului

Interacțiunea omului cu platforma se va putea realiza printr-unul dintre următoarele roluri: oaspete (fără cont personal în baza de date), utilizator – client (rolul pe care îl primește automat orice utilizator care își face cont în aplicație), utilizator – comerciant (rol pe care îl dobândește prin faptul că postează primul anunț de vânzare) și administrator, care are rolul de a gestiona neregularitățile semnalate, eventual șterge sau bloca temporar anumite conturi, fiind o persoană de contact.

Nume	Descriere	Responsabilități	Stakeholder
Vizitator	Nu are cont	informarea asupra comercianților, produselor, căutare, filtrare etc.	Rona Dumitrescu
Utilizator - Client	Are cont și badge de comerciant	comunica via chat cu ceilalți utilizatori, rezerva produse, adăuga în favorite, vizualiza istoric, oferi review-uri	Rona Dumitrescu
Utilizator-Comerciant	Are cont și badge de comerciant	postare anunțuri produse, prețuri adăugare locație și program	Rona Dumitrescu
Administrator	Administrează conturi și reclamațiile	Introduce produse noi în baza de date, rezolvă reclamațiile și poate bloca diverse conturi	Rona Dumitrescu

Tabel 4.1 Tabel cu actorii și responsabilitățile lor

Neavând o identitate înregistrată, oaspetele va putea căuta comercianți, produse etc., însă nu va putea beneficia de multe funcționalități care necesită un cont. Utilizatorii cu un cont în baza de date vor putea comunica via chat, efectua plăti, salva în favorite comercianți și produse, revedea istoricul comenziilor și mai ales, posta un anunț de vânzare prin selectarea locației, a produselor, a programului, a metodei de plată, etc.

Administratorul are rolul de a gestiona problemele semnalate pe parcurs, acestea privind utilizatorii cu o etică neadecvată, având opțiunea de a-i bloca sau cenzura pentru un anumit timp, dar și gestiona mesajele semnalate de către utilizatori via chat și administra diverse lucruri privitoare la comerț, adăugarea, ștergerea, modificarea produselor din baza de date etc.

Actorii care au un cont pe platformă și nu sunt administratori, deci utilizatorii, sunt împărțiți în două roluri: clienți și comercianți.

În următoarea figură sunt prezentate cazurile de utilizare și cei patru actori din cadrul aplicației implementate:

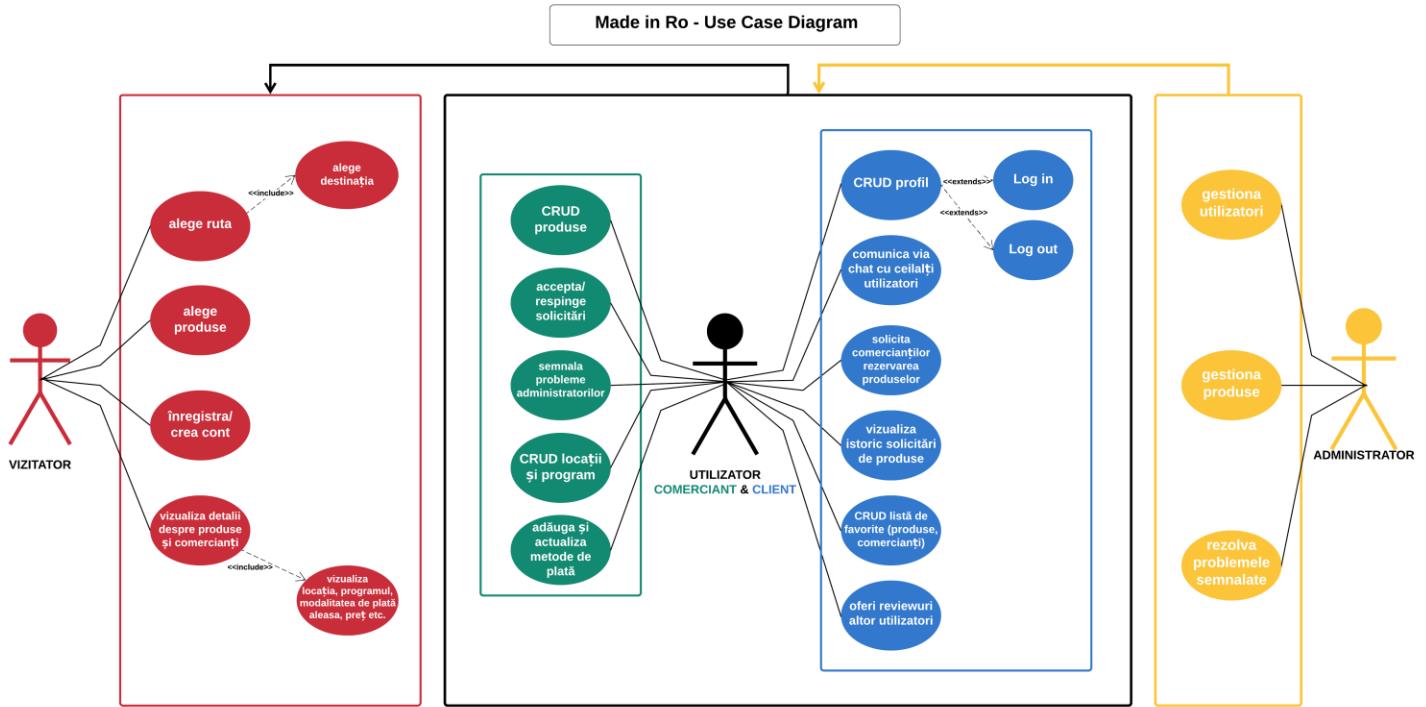


Figura 4.1 Diagrama Use Case

#### 4.1.1. Oaspete

Particularitatea Oaspelui este aceea că nu are cont, deci nu este înregistrat în baza de date. Operațiile pe care acesta le poate efectua în cadrul platformei sunt cele care nu necesită confirmarea unei identități, precum autentificarea, deconectarea, comunicarea via chat cu ceilalți utilizatori, nu poate păstra liste cu produse și producători favoriți și desigur, nu va putea posta anunțuri de vânzare din anonimitate.

În cadrul platformei, Oaspetele va putea căuta diferite produse alimentare prin cele două modalități, Search & Near Me. Va putea căuta diversi utilizatori/producători, vizualiza detaliile de la fiecare stand accesat și desigur, se va putea înscrie în aplicație gratuit.

#### 4.1.2. Utilizator – Client

Utilizatorul-client, spre deosebire de Oaspete, are cont și automat, acesta poate face toate operațiile pe care le poate face și oaspetele, precum: căutarea de produse și producători, vizualizarea detaliilor standurilor etc.

## Capitolul 4

Operăriile pe care acesta le poate desfășura în plus sunt: editarea sau ștergerea profilului, vizualizarea unui istoric și întocmirea, ștergerea sau editarea unei liste de produse și producători favoriți. Poate semnală probleme administratorilor, comunica via chat cu ceilalți utilizatori, oferi review-uri și poate adăuga produse în coș sau trimite solicitări de rezervare asupra acestora. Nu în ultimul rând, Utilizatorul Client se poate autentifica și deconecta din cont.

### 4.1.3. Utilizator – Comerçant

Utilizatorul-Comerçant va putea să se autentifice în cont, să își gestioneze anunțurile prin adăugarea/editarea/ștergerea produselor, locațiilor și a programului, prin gestionarea metodelor de plată și va putea să confirme sau să respingă solicitările/comenziile pe care le va primi. Acesta va putea să își vizualizeze reviewurile și să genereze anumite grafice de interes pentru anunțurile sale.

### 4.1.4. Administrator

În aceasta platformă, administratorul are același rol pe care l-ar avea și un administrator într-un magazin – de a rezolva diverse probleme semnalate – prin blocarea temporară sau permanentă a unor utilizatori sau prin semnalarea mai departe, către programatori, unele neajunsuri tehnice, de a gestiona clienții și comercianții (utilizatori), de a se asigura de ce produse sunt comercializate, asupra sa revenind rolul de adăugare/editare/ștergere produse din baza de date.

## 4.2. Tehnologii utilizate

### 4.2.1. Mediul de dezvoltare - Android Studio

Dezvoltarea aplicațiilor Android nu a fost întotdeauna la fel de convenabilă cum este în ziua de astăzi. Când Android a fost lansat în 2008, ceea ce au obținut dezvoltatorii prin intermediul unui kit de dezvoltare a fost puțin mai mult decât o mână de instrumente din linia de comandă și scripturi Ant build. Construirea de aplicații cu Vim, Ant și alte instrumente din linia de comandă nu a fost atât de rău dacă ești obișnuit cu astfel de lucruri, dar mulți dezvoltatori nu erau obișnuiți, iar lipsa capabilităților IDE, cum ar fi sugestia de cod, configurațiile proiectului și depanarea integrată a constituit un obstacol în calea intrării. [6]

Deși Android Development Tools a fost lansat în același an, 2008, a durat patru ani pentru ca Eclipse IDE și Android ADT să poată fi instalate împreună, deoarece, până în 2012, procesul de instalare al celor două în mod separat era anevoie.

Având la bază o colaborare între JetBrains și Google, Android Studio (AS) a fost lansat în anul 2013. Încă de atunci s-a anticipat că va fi opțiunea numărul unu pentru programatorii Android, eclipsându-l pe Eclipse.

Android Development Tools nu este administrat de o singură organizație, ci de un grup de companii numite “Open Handset Alliance”, care se angajează să ofere un sistem de operare mobil gratuit, complet și open source. Deși această abordare asigură controlul descentralizat al platformei, creează unele complexități. [15]

Dezavantajele sau principalele dificultăți cu care dezvoltatorii Android sunt nevoiți să se confrunte sunt:

- Fragmentarea. Nu toate dispozitivele sunt actualizate în același timp atunci când o nouă versiune este lansată, de aceea, este necesar ca noile dezvoltări să poată accepta și versiunile mai vechi ale sistemului.
- Dimensiunile multiple ale ecranelor. Acest lucru îi obligă pe dezvoltatori să acorde un timp mai mare pentru proiectarea pe mai multe dispozitive, succesul aplicațiilor putând fi influențat de modul în care arată.
- Resurse limitate. Spre deosebire de puterea pe care o are un calculator, UCP-ul și viteza unui telefon mobil sunt limitate.

Pe de alta parte, avantajele sunt incontestabile. Android va continua să își consolideze poziția dominantă pe piața mondială din mai multe motive. Arhitectura modulară Android permite o mare varietate de configurații și personalizări. Toate aplicațiile de bază care sunt livrate standard cu dispozitive Android sunt interschimbabile cu orice număr de aplicații terță parte. [5]. Nu în ultimul rând, un alt mare avantaj este faptul că dezvoltarea de aplicații Android este compatibilă cu multiple sisteme de operare și configurații.

### 4.2.2. Gradle

Utilizând sistemul open source Gradle, aplicațiile Android sunt construite pe baza unui script de construire, “built script”. Aceste fișiere sunt scrise într-o limbă bazată pe Groovy, o limbă dinamică, specifică de domeniu (DSL), pentru JVM.

Gradle este un API de ultimă generație care acceptă cu ușurință personalizările și este utilizat pe scară largă în lumea Java. Plug-in-ul Android pentru Gradle adaugă o gamă largă de caracteristici, inclusiv tipuri de versiuni, arome, configurații de semnare, proiecte de bibliotecă și multe altele. [8]

Gradle are trei etape: inițializare, configurare și execuție. Executarea unei construcții Gradle este, în forma sa cea mai simplă, doar executarea acțiunilor pe sarcinilor care depind de alte sarcini. Pentru a simplifica procesul de compilare, instrumentele de construire creează un model dinamic al fluxului de lucru sub forma unui grafic aciclic direcționat (DAG). Odată ce o sarcină a fost executată, aceasta nu va mai fi apelată. Sarcinile fără dependențe vor rula întotdeauna înainte de celelalte. [12]

### 4.2.3. Limbaj de programare – Kotlin

Inspirat de limbajele de programare Swift, Groovy, C#, Scala și altele, Kotlin a fost creat de profesioniști JetBrains, iar luna Mai a anului 2018, a fost anunțat în mod oficial de către Google ca fiind unul dintre limbajele de programare oficiale pentru mediul de dezvoltare Andoird. Google a mai menționat că este “concis, expresiv și proiectat să fie null-safe”. [11]

Kotlin este deplin interoperabil cu Java și poate rula pe JVM, de asemenea, este compatibil cu JDK 6, deci aplicațiile dezvoltate pot rula chiar și pe dispozitive mai vechi decât Android 4. [10]

În ceea ce privește limbajul de programare Kotlin, de multe ori discuția se construiește în jurul comparației dintre acesta și limbajul de programare Java. După ce s-au cântărit avantajele și dezavantajele celor două limbi, s-a hotărât ca implementarea

## Capitolul 4

acestei lucrări de diplomă să se efectueze cu ajutorul limbajului Kotlin. Câteva dintre diferențele studiate sunt ilustrate în următorul tabel:

Kotlin	Java
Este un limbaj mai sigur din punct de vedere al nulabilității	Are obiecte și variabile nule
Nu are NullPointerException – Acest lucru a dus la evitarea unei greșeli cunoscută sub numele de “Billion Dollar Mistake”)	Java se bazează pe NullPointerException
Nu mai este necesar simbolul “;” de la sfârșitul fiecărei instrucțiuni	Este necesar să punem
Are funcție de “smart cast”	Nu dispune de o astfel de funcție
A înlocuit operatorul ternar cu “if else”, ex Kotlin: “if(a>b) a else b”	Dispune de operatorul ternar ex Kotlin: “(a>b) ? a : b”
Nu este chiar atât de popular	Găsirea informațiilor necesare sau a mentorilor este mai ușoară datorită popularității limbajului
Kotlin se poate utiliza direct în “Gradle build scripts”	Nu oferă astfel de funcționalitate
Dispune de o funcție de “Lazy Loading”	Nu dispune de o astfel de funcție
Nu are nevoie de specificarea tipului de variabilă, utilizând “var” pentru variabile și “val” pentru constante	În Java trebuie specificat individual fiecare tip
Nu oferă conversii implicite	Oferă conversii implicite
Volumul de cod scris este mult mai redus comparat cu Java	Este un limbaj mai amplu în ce privește numărul de cuvinte necesare

Tabel 4.2 Diferențele dintre limbajele de programare Java și Kotlin

### 4.2.4. Limbajul de marcare – XML

Dezvoltarea în Android înseamnă automat limbajele de programare Kotlin sau Java și meta-limbajul XML. Acesta din urmă este un sistem dinamic de marcare, neavând taguri predefinite, este responsabil cu stocarea, structurarea eficientă a datelor și în special cu transportul acestora.

Respectând şablonul SAMF (Single Activity, Multiple Fragments), fiecare fragment implementat în Kotlin a avut un fișier corespondent în XML.

### 4.2.5. Google Cloud Platform

Pentru o platformă care și-a propus ca obiective non-funcționale flexibilitate și scalabilitate ridicată, utilizarea unui cloud este esențială. Cloud computing se referă la abstractizarea infrastructurii de calcul și a altor resurse asociate și oferirea lor ca serviciu, de obicei pe bază de plată pe utilizare, pe internet. Serviciul poate fi vizat pentru consumul uman sau pentru alte sisteme software. Utilizatorii au nevoie doar de un browser web pentru a accesa serviciile; sistemele software pot consuma servicii folosind o interfață de programare a aplicațiilor web (API). [9]

În multe privințe, cloud-ul este următorul strat de abstractizare în infrastructura computerelor, unde calculul, stocarea, analiza, rețea și multe altele sunt împins mai sus până la stiva de calcul. Această structură se concentrează asupra dezvoltatorului de la CPU și RAM și către API-uri pentru operațiuni de nivel superior, cum ar fi stocarea sau interogarea datelor. [16]

Deși există mai multe platforme de cloud cu servicii diver, Amazon, Microsoft, DigitalOcean etc., pentru această lucrare s-a ales Google Cloud Platform, datorită serviciului integrat Google Maps API.

### 4.2.6. Google Maps API

Google, inclusiv Google Cloud Platform, a adaptat o filosofie de dezvoltare bazată pe APIs, fiind principala interfață pentru dezvoltatori. De aceea, pentru utilizarea Google Maps API, una dintre principalele componente ale proiectului, autorul a fost nevoie să activeze acel API în proiect, printr-o cheie specifică generată de Google Cloud Platform – Maps API Services, atât pentru release, cât și pentru debug.

Google Maps a fost dezvoltat inițial de doi frați danezi, Lars și Jens Rasmussen. Aceștia au co-fondat compania Where 2 Technologies, o companie care a decis să creeze soluții de cartografiere. Compania a fost achiziționată de Google în octombrie 2004 și astfel, cei doi frați au creat Google Maps. După ce câțiva oameni și-au dat seama cum să pirateze Google Map pentru a-l integra în hărțile lor, Google a ajuns la concluzia că este nevoie pună al dispozitiv un API public.

API-ul în sine constă în principiu din fișiere JavaScript care conțin clase cu metode și proprietăți pe care le puteți utiliza pentru a spune hărții cum să se comporte. [13]

### 4.2.7. Firebase

Una dintre cerințele stabilite încă de la început pentru baza de date a acestei platforme a fost ca aceasta să fie în cloud, pentru a reduce dimensiunile aplicație pe dispozitivul utilizatorului. S-au luat în considerare mai multe sisteme, printre care Praser Microsoft Azure SQL, Firebase etc. și în final, s-a ales cea din urmă datorită avantajelor care vor fi prezentate în paragrafele următoare.

Firebase a fost înființat de Andrew Lee și James Tamplin din 2011, lansată oficial în aprilie 2012. În 2014, Google a preluat Firebase și a dezvoltat serviciile acestuia. Atunci când vine vorba de dezvoltarea aplicațiilor, Firebase gestionează cea mai mare parte a lucrărilor din partea serverului. În acest fel, ajută la menținerea unei stări de armonie între dezvoltator și client, asigurând o minimă întârziere a lucrului. [3]

## Capitolul 4

Firebase stochează date în format JSON, care nu utilizează interogări pentru inserarea, editarea sau ștergerea datelor. Acesta are multe servicii disponibile, însă cele esențiale pentru aplicația implementată sunt serviciile de cloud messaging, autentificare, baze de date în timp real, stocare și teste lab pentru Android. Autentificarea se poate realiza foarte simplu și prin intermediul Facebook, Google, Twitter sau GitHub, utilizând codul clientului. De asemenea, este un serviciu care poate activa autentificarea cu email și parola de login stocate în Firebase.

Serviciul de stocare a datelor în timp real presupune furnizarea unui API dezvoltatorului, pentru a permite sincronizarea datelor aplicațiilor client și stocarea acestora în cloud. [7]

### 4.2.8. *Adobe Illustrator*

Pentru un design reușit al interfeței utilizator, s-a utilizat și aplicația software Adobe Illustrator pentru acest proiect. Ilustrațiile, iconițele și celelalte componente grafice, s-au realizat fiecare pe o planșă de lucru separată în cadrul aceluiași fișier, având la bază diferite tooluri specifice de construire și tăiere a formelor geometrice.

Datorită faptului că aplicația va fi utilizată și de utilizatori nu foarte experimentați în domeniul tehnologiei, s-a urmărit realizarea unei grafici simple, intuitive. Toate elementele grafice pot fi ușor modificate datorită proprietăților vectoriale și pot fi exportate cu o gamă largă de extensii pe viitor.

## Capitolul 5. Proiectare de Detaliu și Implementare

### 5.1. Proiectare software

Prin proiectare software înțelegem activitatea prin care analizăm și stabilim modul de rezolvare al unei probleme. Acest lucru se realizează prin modularizarea și stratificarea problemei, prin stabilirea modului de comunicare între module, a funcționalităților fiecărui și a modului de testare și validare.

Pentru o proiectare software în detaliu, s-au realizat mai multe diagrame de tip UML, cu scopul de a realiza o documentație cât mai explicită a proiectului software de diplomă.

Diagramele UML sunt împărțite în două categorii: de structură și de comportament. Diagramele de structură au rolul de a evidenția diferențele obiectelor din sistem, iar diagramele de comportament descriu ce ar trebui să se întâmple în cadrul sistemului, interacțiunea dintre obiecte.

#### 5.1.1. Diagrama generală arhitecturală

În următoarea figură este ilustrată prima diagramă structurală, diagrama de arhitectură a sistemului, care cuprinde componentele principale hardware și software ale sistemului.

Diagrama arhitecturală generală a aplicației mobile este ilustrată în următoarea figură:

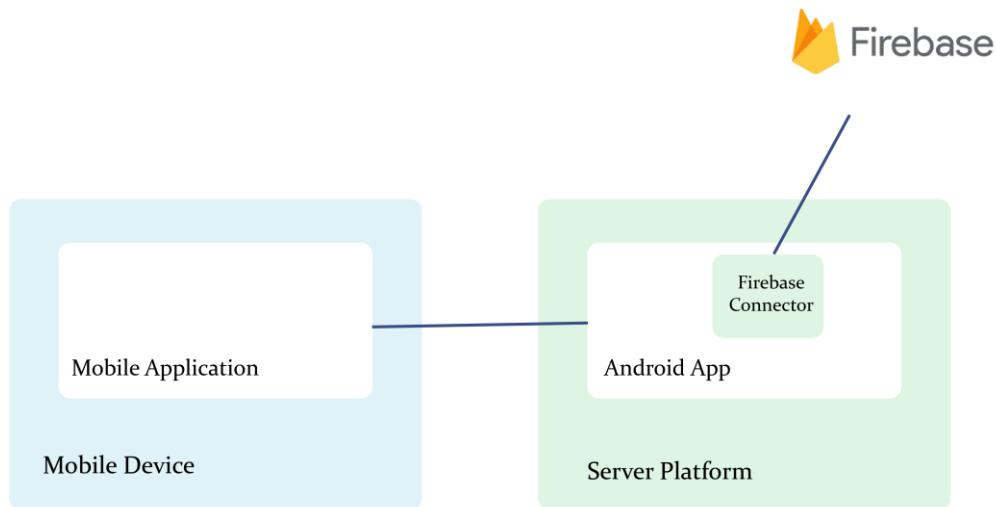


Figura 5.1 Diagrama generală arhitecturală a sistemului

### 5.1.2. Diagrama de desfășurare

Sistemul este alcătuit dintr-o bază de date, un server, o conexiune prin API cu Google Maps și implicit, interfață grafică prin care utilizatorul interacționează cu aplicația și conexiunea la internet care face posibil acest lucru.

Baza de date Firebase este o bază de date nerelațională, utilizată foarte des de către dezvoltatorii de aplicații Android în Android Studio. Pentru majoritatea aplicațiilor bazate pe navigație, precum este cea propusă, se utilizează Google Maps APIs, împreună cu toate funcționalitățile necesare pentru clădirea pilonilor de navigație în cadrul acestei aplicații de vânzare-cumpărare.

Următoarea diagramă de structură este ilustrată în figura de mai jos, diagrama de desfășurare a proiectului software este:



Figura 5.2 Diagrama de desfășurarea a aplicației

### 5.1.3. Diagrama de pachete

O altă diagramă de structură utilă pentru o mai bună înțelegere a aplicației ar fi diagrama de pachete, care ilustrează legătura dintre pachetele sistemului, legătură care constituie baza pentru controlul configurației și stocarea.

Obiectivul principal al acestei diagrame este de a prezenta relația dintre diferitele componente mari (pachete), care formează un sistem mai complex. [22]

Pachetele “Fragments” și “Adapters” sunt echivalente cu “View” din modelul Model-View-Controller. Un pachet pentru Adapters a fost necesar, deoarece s-au utilizat

mai multe RecyclerView-uri în diferite locuri în cadrul aplicației. Cele două, Fragments și Adapters, înglobează clasele pentru interfața grafică cu care interacționează utilizatorul și sunt legate între ele prin data binding.

Pachetul “Service” cuprinde toate clasele care se ocupă de operațiile CRUD ale aplicației, reprezentând nivelul de “business logic”. Mai departe, nivelul de acces la date este reprezentat de către pachetele “dao” și “repositories”, responsabile de accesul și operațiile cu baza de date, Firebase. Pachetul „models” , conține modelele cu care s-a lucrat în cadrul aplicației.

Următoare figură le prezintă în detaliu, atât pachetele, cât și conexiunile dintre acestea:

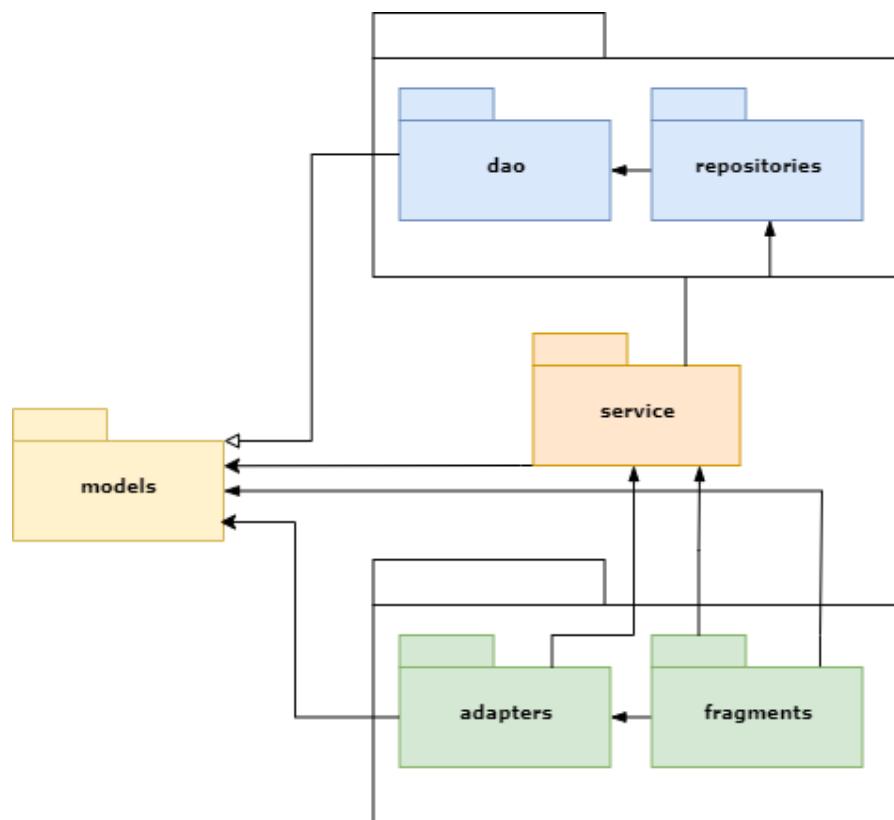


Figura 5.3 Diagrama de pachete

#### 5.1.4. Diagrama de secvențe

În următoarea figură se poate observa reprezentarea diagramei de secvențe la modul general, pentru majoritatea funcționalităților din aplicație disponibile pentru actorul: utilizator – client. Se prezintă procedeul de autentificare, comandare produse, oferire de review-uri și gestionare listă de favorite.

## Capitolul 5

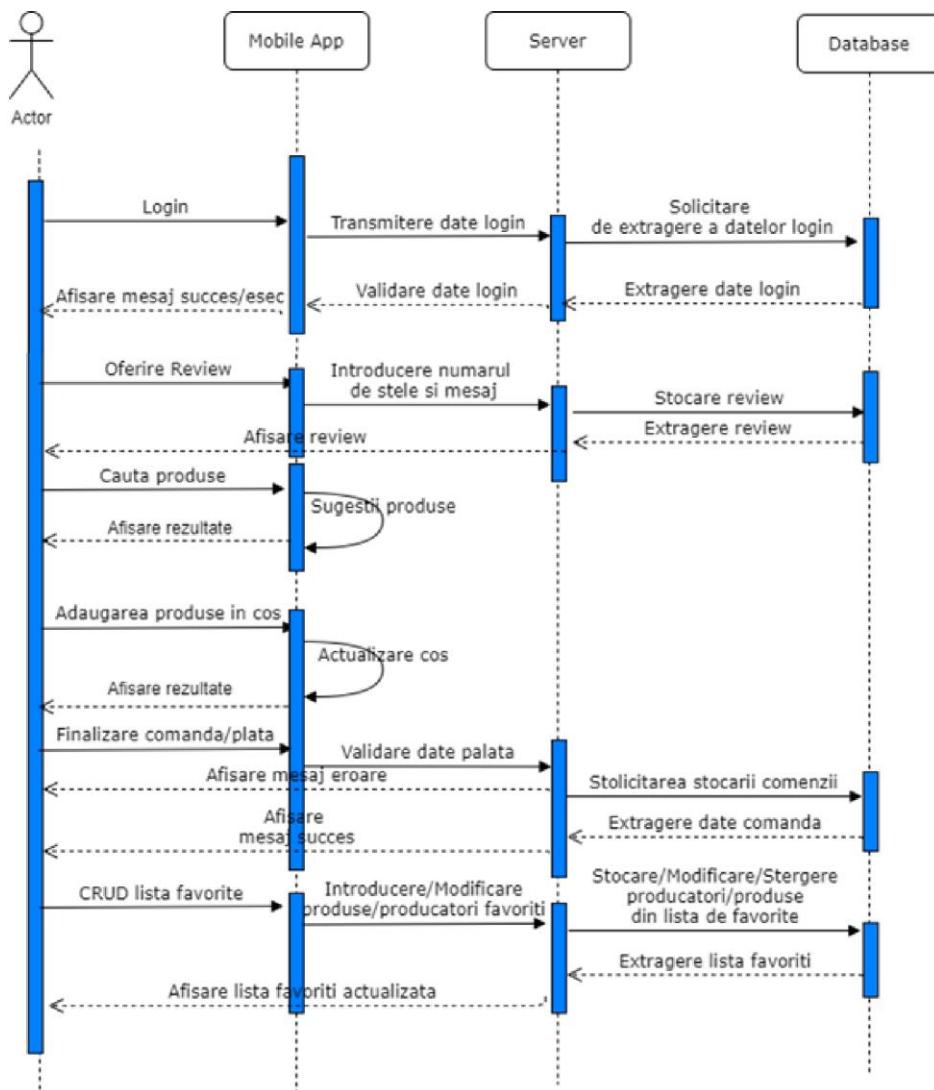


Figura 5.4 Diagrama de Secvențe (cazul general utilizator - client)

Descriere pe pași a diagramei de secvențe prezentate:

*Pas 1.*

Utilizatorul trebuie să se autentifice în contul său. Va introduce datele corespunzătoare, iar dacă acestea vor fi corecte, va intra în cont, altfel, se va afișa un mesaj de eroare.

*Pas 2.*

Pentru a căuta produse, utilizatorul introduce textul în bară de căutare și primește sugestii sub formă de listă sub bară de căutare.

*Pas 3.*

Utilizatorul selectează produsele dorite, adăugându-le în cos și își continuă cumpărăturile.

*Pas 4.*

Când dorește să finalizeze comanda, apasă butonul de finalizare, introduce datele de plată, iar după validarea plății, va primi un mesaj de plată efectuată sau nu.

*Pas 5.*

Pentru oferirea unui review, se selectează producătorul, se va selecta numărul de stele și se va introduce un mesaj care să reflecte experiența clientului cu acesta.

*Pas 6.*

Adăugarea sau alte modificări la lista de favorite se va face prin selectarea sau deselectarea emoticonului “inimă” vizând fiecare produs/producător în parte.

### 5.1.5. Diagrama de comunicare

Un alt tip de diagramă comportamentală este diagrama de comunicare. Aceasta ilustrează activitățile între obiecte prin schimburi de mesaje.

Diagrama de comunicare corespunde (adică ar putea fi convertită sau înlocuită) unei diagrame simple secvențiale, fără mecanisme de structurare, cum ar fi utilizările de interacțiune și fragmentele combinate. Se presupune, de asemenea, că depășirea mesajelor (adică ordinea receptiilor este diferită de ordinea de trimis a unui set dat de mesaje) nu va avea loc sau este irelevantă. [28]

Am întocmit diagrama de comunicare pentru cazul în care utilizatorul-client dorește să își comande câteva produse.

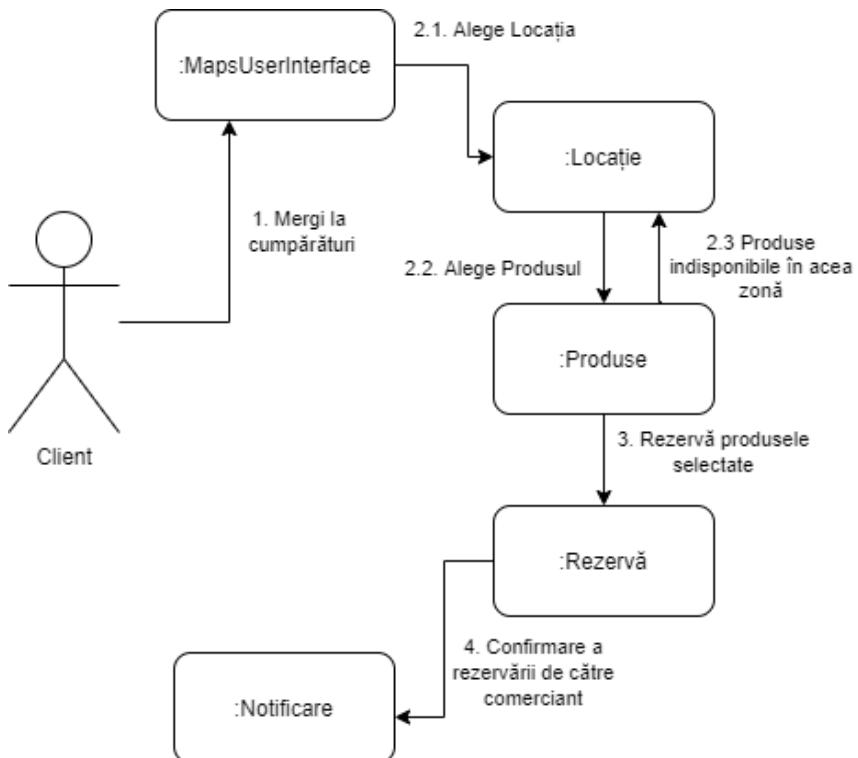


Figura 5.5 Diagrama de comunicare

### 5.1.6. Diagrama de stări (taskuri)

Diagrama de stări modelează natura dinamică a sistemului, timpul de viață și diferențele stării în care sistemul se află în timp.

Un astfel de model de diagramă de stări este ilustrat în următoarea figură:

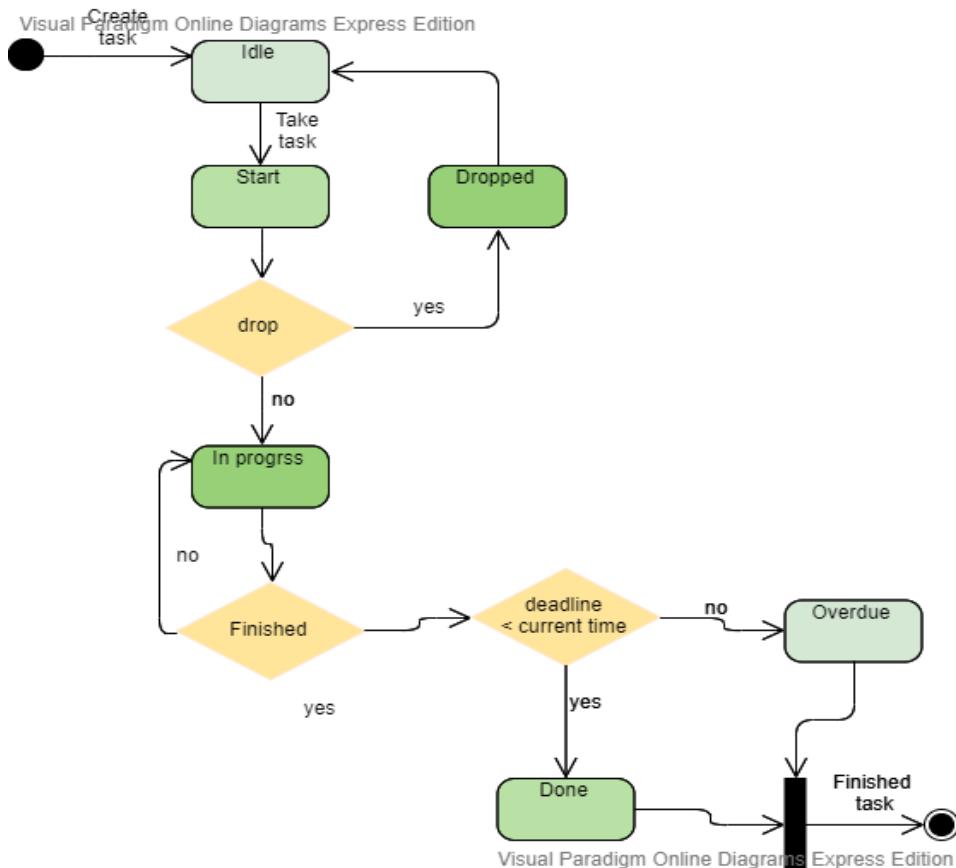


Figura 5.6 Digrama de tranziție stări (taskuri) [29]

### 5.1.7. Diagrama flux de activități (flowchart)

Proiectul de licență se adresează nișei de comercianți care produc în gospodăriile lor mai mult decât pot consuma și își scot produsele la vânzare pe mese, aproape de carosabil. De asemenea, proiectul se adresează clientilor care doresc să mănânce sănătos, românesc, fie dintr-o poftă pe care o au când sunt la drum lung (de struguri, zmeură, afine etc.), fie dintr-o dorință de a se aproviziona regulat, direct de la producători de încredere din apropiere.

Aceștia pot interacționa cu platforma printr-unul dintre următoarele patru moduri: oaspete, utilizator – client, utilizator - comerciant și administrator. Pentru cazul în care utilizatorul-comerciant dorește să interacționeze cu aplicația, am prezentat modul de interacțiune prin următoarea diagramă de flux de activități, specifică actorului cu rolul de utilizator-comerciant.

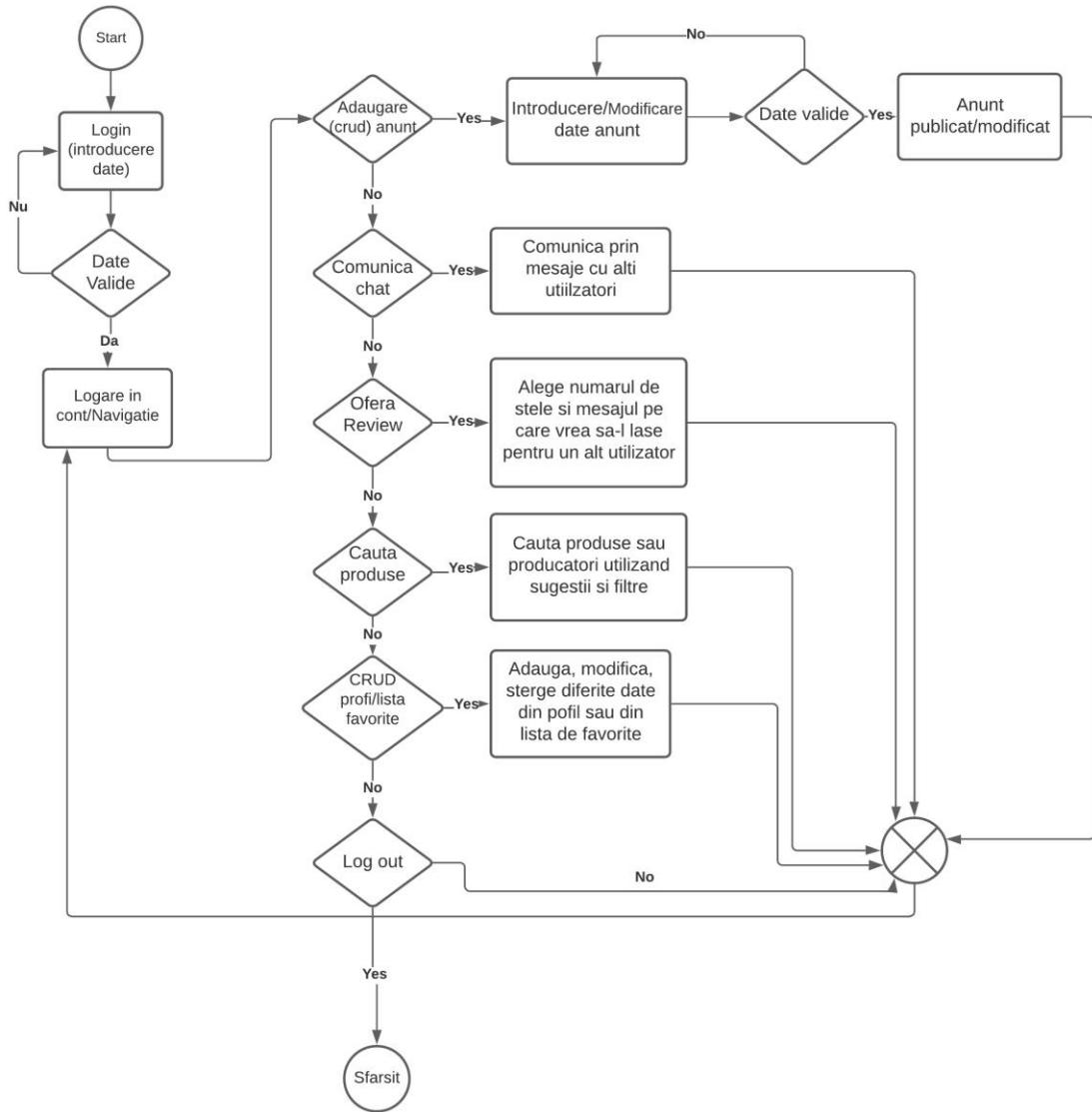


Figura 5.7 Diagrama fluxului de activități

#### **Start caz de utilizare general**

Diagrama evenimentelor pornește din starea de start, reprezentând faptul că utilizatorul a intrat pe platformă și dorește să interacționeze cu aceasta.

#### **Pas 1**

Primul pas îl reprezintă autentificarea în aplicație. Dacă datele introduse de utilizator sunt corecte, acesta este logat în pagina principală, adică în pagina de navigație. Dacă logarea nu este efectuată corect, vezi alternative flow 2.2.1 Login Error.

#### **Pas 2**

Următorul pas îl reprezintă desfășurarea acțiunii dorite în cadrul aplicației. Drept exemplu, am luat adăugarea unui anunț. Se vor selecta toate datele necesare pentru postarea unui anunț: locația, programul și produsele comercializate.

### **Pas 3**

Dacă datele sunt valide, anunțul va fi publicat în cadrul aplicației. Dacă datele nu sunt valide, vezi alternative flow 2.2.2 Date invalide adăugare anunț.

### **Sfârșit caz de utilizare general**

Sfârșitul evenimentelor desfășurate pe platformă este marcat de către delegarea utilizatorului și părăsirea aplicației.

#### *5.1.8. Flux de date alternativ*

##### *5.1.8.1. Eroare login*

**Step 1** Fluxul de date alternativ începe.

**Step 2** Dacă datele introduse la login nu sunt valide, se afișează un mesaj de eroare.

**Step 3** Fluxul de date alternativ ia sfârșit.

##### *5.1.8.2. Date invalide adăugare anunț*

**Step 1** Fluxul de date alternativ începe.

**Step 2** Dacă datele introduse la login nu sunt valide, se afișează un mesaj de eroare.

**Step 3** Fluxul de date alternativ ia sfârșit.

## **5.2. Baza de date**

Baza de date care a fost aleasă pentru această platformă este Firebase, de la Google, o bază de date nerelațională. În ciuda faptului că s-a ales o bază de date nerelațională, prima dată s-a pornit de la ideea că sistemul va avea o bază de date relațională, obligatoriu cu stocarea datelor în cloud și s-a realizat diagrama corespunzătoare relațională.

#### *5.2.1. Diagrama bazei de date*

Pentru o prezentare cât mai precisă a entităților și a relațiilor dintre acestea, s-a întocmit următoarea diagramă de bază de date relațională, indicând cheile primare și cheile străine:

## Capitolul 5

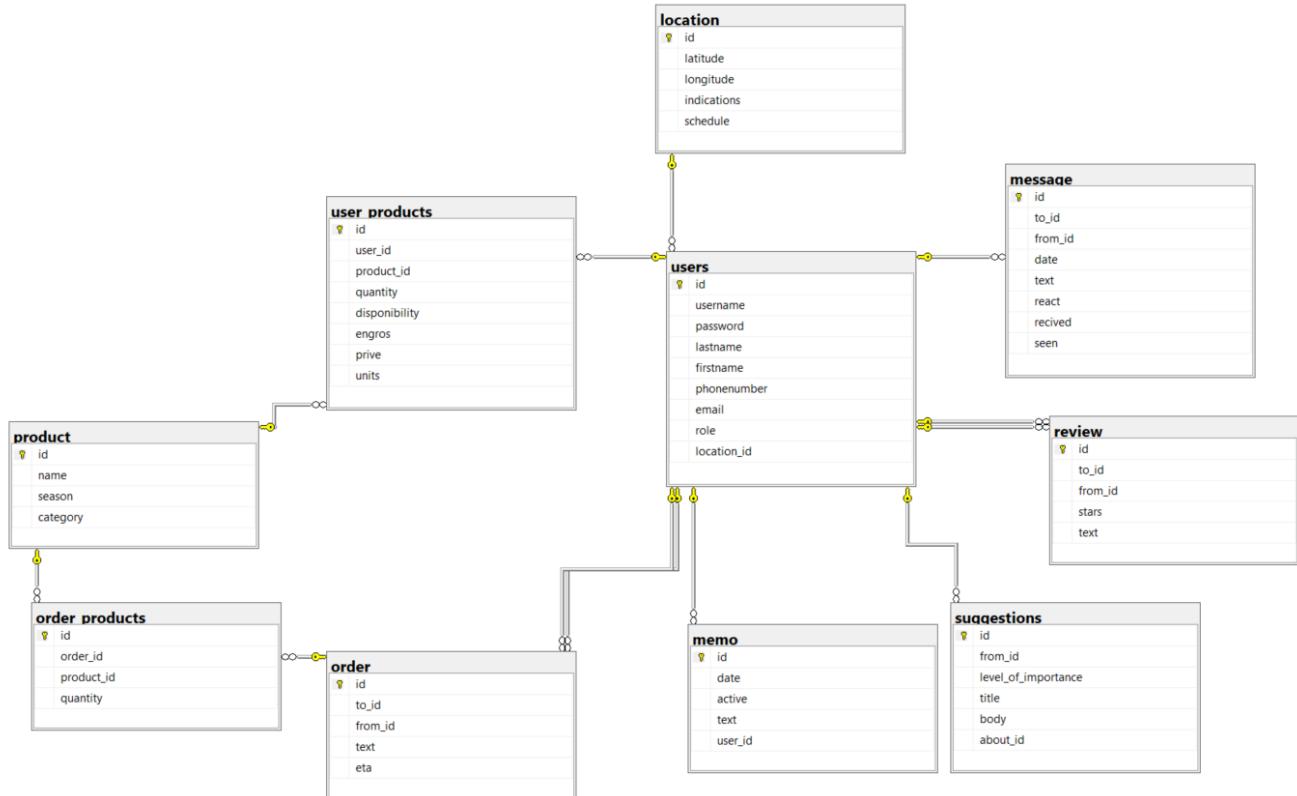


Figura 5.8 Diagrama relațională a bazei de date

### 5.2.2. Descrierea tabelelor

În cea de-a cincea figură a documentului se prezintă diagrama bazei de date a platformei mobile “Made in Ro”. Această bază de date cuprinde zece tabele, între care s-au utilizat toate tipurile de relații specifice bazelor de date relaționale, în afară de relația de recursivitate.

Tabela “users” cuprinde datele utilizatorilor care se înregistrează pe platforma implementată - utilizatorul-client, utilizatorul-comerçant și pentru administrator, care va urma să primească un cont cu care se va putea autentifica pentru a-și desfășura acțiunile specifice.

Pentru că utilizatorii pot comunica între ei, s-a adăugat și tabela de mesaje, “message”, care conține câteva date necesare pentru identificarea unui mesaj: destinatarul, expeditorul, data și ora, mesajul propriu-zis, statusul și eventuale reacții cu care destinatarul poate răspunde mesajului.

## Capitolul 5

În ce privește partea de vânzare-cumpărare, comerciantul poate alege mai multe locații în care să își comercializeze produsele, astfel, relația dintre tabelele “users” și “location” este de tipul unu la n. Pentru stabilirea unei locații, sunt necesare datele de identificare pe hartă, latitudinea și longitudinea, dar și câteva indicații pentru a ajuta clienții să identifice mai ușor locul în care se află comerciantul. De asemenea, fiecare locație dispune și de un program de lucru.

Produsele din baza de date se găsesc în tabelul “product” și sunt definite de câteva detalii semnificative: numele, sezonul în care acestea apar și pot fi consumate, precum și categoria din care fac parte: fructe, legume etc. Fiecare comerciant, înainte de posta anunțul cu standul său, este nevoie să selecteze produsele alimentare pe care acesta le comercializează din produsele din baza de date a aplicației. Astfel, între utilizatorii-comercianți și produse se stabilește o relație de n la n, deoarece un produs poate fi adăugat de către mai mulți comercianți în anunțul fiecărui dintre ei și un comerciant poate adăuga mai multe produse în anunțurile sale. De aceea, s-a introdus tabelul “user\_products”, pentru a transforma relația de tipul n la n, în două relații de tip unu la n. În momentul în care comerciantul alege produsele pentru anunțul său, este nevoie să introducă prețul, cantitatea și alte informații pentru fiecare.

Pentru a trimite o comandă, utilizatorul selectează produsele și cantitatea dorită, dacă dorește, poate preciza și timpul aproximativ în care va ajunge să ridice comanda și la final, i se va afișa prețul total. Tabelul “order” ține evidența tuturor cumpărăturilor. Și de această dată a fost nevoie de un tabel în plus, “order\_products”, pentru a rezolva situația în care tabelele sunt în relație de n la n, deoarece un produs poate face parte din mai multe comenzi și o comandă poate conține mai multe produse.

Fiecare utilizator poate lăsa un review. Informațiile create în urma acestui proces, se vor stoca în tabelul “review”, cu datele necesare identificării acestuia: utilizatorul care a lăsat feedback-ul, utilizatorul căruia i se adresează review-ul, numărul de stele și textul scris, pentru cei care doresc să își motiveze alegerea.

De asemenea, la constatarea unor probleme, se pot trimite sugestii către administratori în vederea rezolvării acestora. La fel ca în cazul celorlalor tabele, se păstrează în tabelul “suggestions” din baza de date, informațiile esențiale: persoana care a trimis sesizarea, nivelul de importanță al acesteia, titlul care sintetizează conținutul și mesajul propriu-zis.

### 5.3. Proiectarea elementelor grafice ale interfeței utilizator

#### 5.3.1. Diagrama de fragmente și conexiuni între acestea

Unele dintre obiectivele inițiale au fost ca navigabilitatea să fie intuitivă și funcționalitățile să fie ușor de identificat prin ilustrații specifice.

Pentru orice proiect în Android Studio, dezvoltatorul este nevoie să aleagă între două metode de implementare în ceea ce privește navigabilitatea în cadrul aplicației. Fie să implementeze activități dedicate pentru fiecare ecran prin care utilizatorii vor naviga, fie să implementeze o activitate care să indice primul fragment din cadrul unui graf de navigare compus din fragmente.

## Capitolul 5

Există mai multe dezavantaje legate de prima metodă, cea cu activități multiple. Este mai dificil să comunică date și informații între activități decât între fragmente, performanța aplicației nu este la fel de ridicată și de asemenea, este necesar să treacă de la o activitate la cealaltă cu ajutorul funcțiilor “show” și “hide”, uneori costisitoare pentru un număr mare de activități.

Datorită performanței ridicate specifice fragmentelor spre deosebire de crearea unor noi activități, a librăriei “Navigation component” puse la dispoziție, care poate să gestioneze tranzitii în cadrul grafurilor de navigare de o complexitate ridicată, a faptului că același context poate fi utilizat în oricare dintre fragmente și a accesului mai facil la elementele din interfața grafică a utilizatorului prin intermediul funcționalității “data binding”, am ales să utilizez principiul SAMF (Single Activity Multiple Fragments) pentru implementarea platformei.

Am început astfel implementarea proiectului de diplomă prin construirea graficului de navigabilitate. Am adăugat fragmentele necesare, reprezentând ecranele prin care vor naviga și vor interacționa diferenți actori, fiecare după rolul său și, în final, am ajuns la un total de douăzeci de fragmente, douăzeci de fragmente și o singură activitate (SAMF).

În timpul implementării am păstrat următoarea convenție în ce privește numele variabilelor sau a oricărui tip de element (widget, fragment etc.): denumirea variabilelor și a celorlalte elemente este în limba engleză, iar textul pe care utilizatorul îl citește este afișat în limba română.

În cadrul platformei, pentru afișarea listelor de utilizatori, produse, sugestii, anunțuri comerciale etc., de cele mai multe ori s-a utilizat widgetul “RecyclerView” în detrimentul lui “ListView”, predecesorul lui. Am luat această decizie datorită faptului că RecyclerView este mai economic din punctul de vedere al memoriei, nu este limitat de un tip de afișare al elementelor, cum era ListView de modul de afișare orizontal al elementelor, este mai flexibil și necesită un volum de cod semnificativ mai mic pentru a fi implementat.

Un alt element utilizat des în cadrul aplicației este “Floating Action Button”, prescurtat FAB, care literalmente ar însemna un buton de acțiune care plutește. Datorită formei rotunde de buton din viața cotidiană, a faptului că ocupă un spațiu mic pe ecran (dacă se respectă dimensiunile standard de 56dp x 56dp și 40dp x 40dp) și a faptului că poate fi adăugată o iconă sugestivă pentru acțiunea pe care acesta o îndeplinește, am ales să utilizez predominant, urmând sfaturile de bune practici ale unui website specializat în UI/UX: [18].

În ce privește diagrama de navigabilitate, a fragmentelor și a conexiunilor dintre acestea, nu toți actorii au acces la toate fragmentele. Fragmentele pentru actorul Oaspete sunt: permissionFragment, MapsFragment, LoginFragment și SignUpFragment. Grupul de fragmente pentru acțiunile specifice Administratorului (după ce intră în cont, adică după ce trece LoginFragment), este: AdminHomeFragment, UsersFragment, ProductsFragment și NewProductsFragment. În cazul Utilizatorului Comerçant, acestea sunt: SellerHomeFragment, AdsFragment, NewAdFragment, NewProductAdFragment și RequestsFragment, iar în cazul Utilizatorului Client, acestea sunt: MapsUserFragment, ChatFragment, IndividualConversationFragment, ShoppingCartFragment, CustomSettingsFragment, ProfileFragment și SuggestionsFragment. Acestea vor fi prezentate în detaliu în următorul subcapitol, “Componentele principale ale sistemului”.

## Capitolul 5

Cele douăzeci de fragmente și conexiunile dintre acestea sunt ilustrate în diagrama de mai jos:

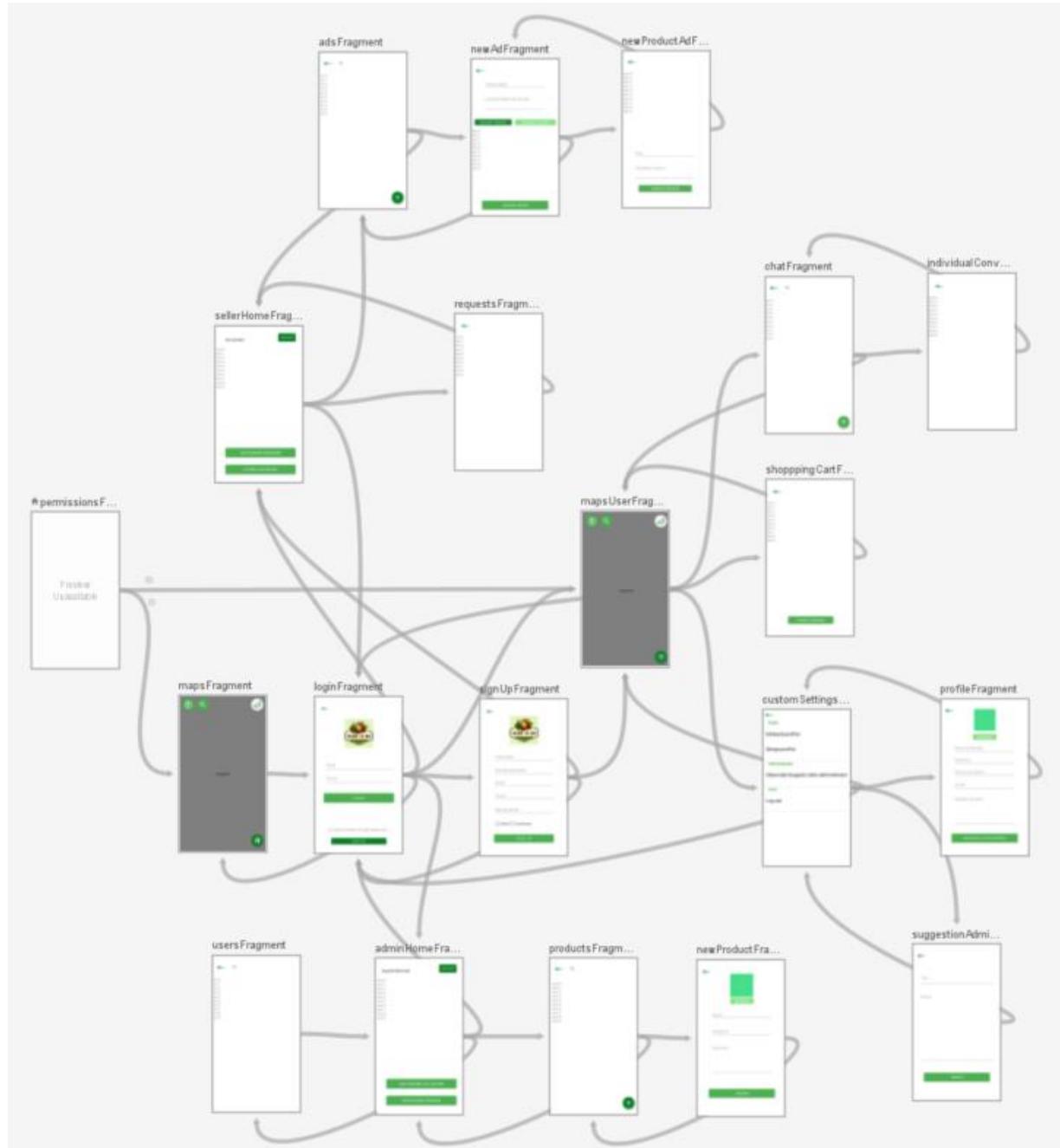


Figura 5.9 Diagrama de fragmente și conexiunea între acestea

Pentru un aspect plăcut, ordonat, în ton cu domeniul platformei, paleta de culori a aplicației este formată din trei nuanțe de verde, alese de pe un site de specialitate: [25], pentru a crea diverse accente în designul ecranelor.

## Capitolul 5

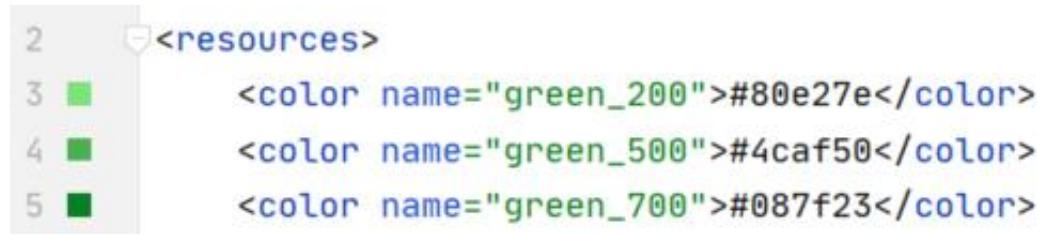


Figura 5.10 Paleta de culori utilizată în aplicația mobilă

Datorită faptului că platforma se adresează și unei game de utilizatori cu mai puțină experiență în domeniul tehnologiei, s-a acordat o deosebită importanță elementelor grafice ale interfeței utilizator, dar și a gradului de dificultate în ce privește navigabilitatea aplicației.

De aceea, pentru butoanele FAB, despre care am menționat anterior, s-au realizat iconițe speciale, care să indice tipul de acțiune pe care acestea le vor realiza dacă vor fi apăsate. Unele iconițe sunt realizate de la zero, altele sunt preluate și transformate astfel încât să răspundă unor cerințe specifice.

Mai jos este prezentat fișierul de lucru din aplicația “Adobe Illustrator”, în care am lucrat la elementele grafice ale interfeței utilizator:

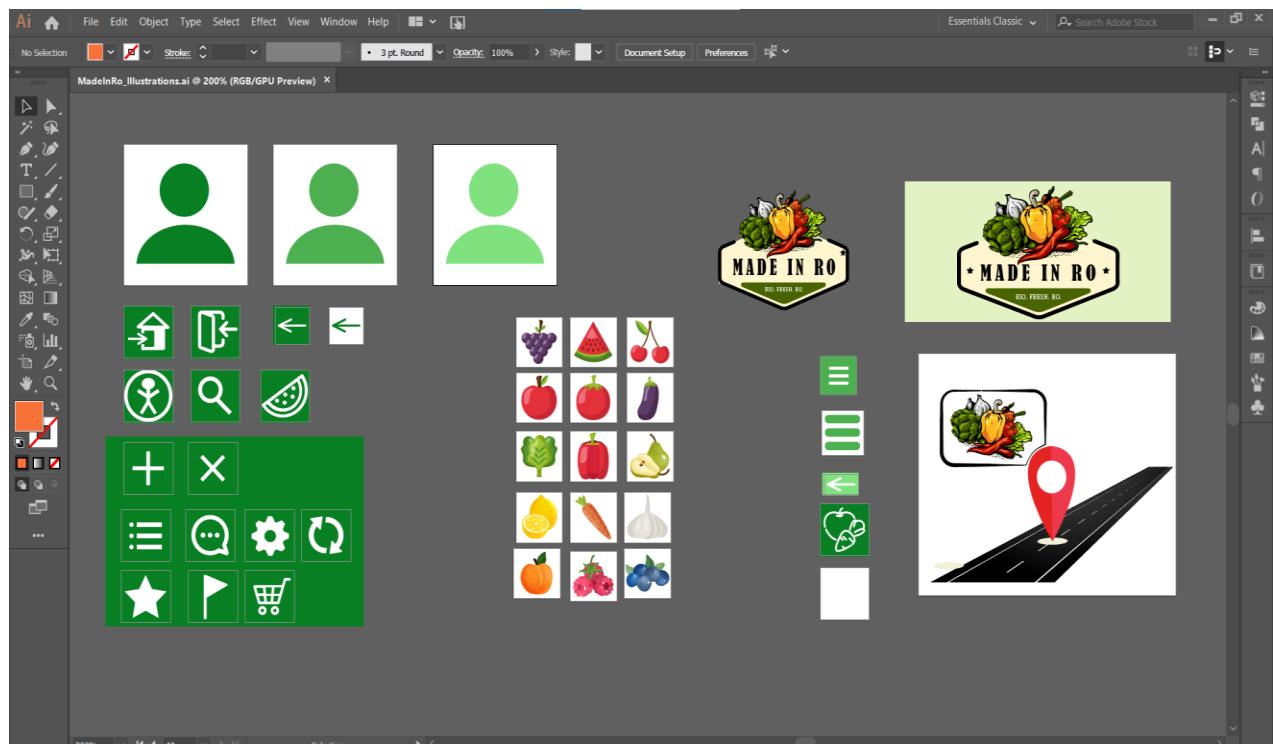


Figura 5.11 Fișierul de lucru din Adobe Illustrator

## 5.4. Componentele principale ale sistemului

### 5.4.1. Harta

Proiectul are la bază navigarea utilizatorilor în cadrul unei hărți furnizate de Google Maps. Pentru toate tipurile de actor este necesar să se ofere accesul la locație, lucru făcut cunoscut încă din prima interacțiune a utilizatorului cu aplicația. Accesul cerut este de tipul “ACCESS\_FINE\_LOCATION”, în detrimentul lui “ACCESS\_COARSE\_LOCATION”, care are o precizie mai scăzută în ce privește identificare utilizatorului, precizie de câteva blocuri.

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

Figura 5.12 Captură de ecran cu tipul de acces solicitat utilizatorilor

Harta a fost furnizată de către Google Cloud Platform prin intermediul unui Google Maps API key. Atât pentru versiunea debug, cât și pentru versiunea release, a fost nevoie de chei separate, care au fost obținute pe baza unui certificat “SHA-1”.

### 5.4.2. Navigabilitatea

Atât din postura de Client, cât și din cea de Oaspete, utilizatorul poate naviga în cadrul aplicației prin intermediul a două funcționalități: near me (1) și căutare locație (2).



Figura 5.13 Capturi de ecran - Opțiuni de navigabilitate

## Capitolul 5

Pentru a redimensiona corect camera și a face posibilă vizualizarea produselor din apropierea utilizatorului, este nevoie să ne folosim de nivelurile de zoom pe care le pune la dispoziție Google Maps. Acestea sunt ilustrate în următorul tabel:

Nivel de zoom Google Maps	Distanță (m/pixel la ecuator)	Scără
0	156 412	1:500 milioane
1	78 206	1:200 milioane
2	39 103	1:150 milioane
3	19 551	1:70 milioane
4	9 776	1:35 milioane
5	4 888	1:15 milioane
6	2 444	1:10 milioane
7	1 222	1:4 milioane
8	610,984	1:2 milioane
9	305,492	1:1 milioane
10	152,746	1:500 mii
11	76,373	1:250 mii
12	38,187	1:150 mii
13	19,093	1:70 mii
14	9,547	1:35 mii
15	4,773	1:15 mii
16	2,387	1:8 mii
17	1,193	1:4 mii
18	0,596	1:2 mii
19	0,298	1:1 mie
20	0,149	1:5 sute

Tabel 5.1 Nivelurile de Zoom Google Maps [23]

Pentru a obține un nivel de zoom corespunzător, care să cuprindă cercul desenat pe ecran, trebuie să îl calculăm cu ajutorul următoarei formule:  $\log_2(2000.0.i)$ , unde i reprezintă raza în metri a cercului, după cum se poate observa în următoarea figură:

## Capitolul 5

```
236 private fun createCircle(i: Int) {  
237     if(circle == null) {  
238         circle = map.addCircle(CircleOptions().apply { this: CircleOptions  
239             center(userLocation)  
240             radius( radius: i*1000.0)  
241         })  
242         map.animateCamera(CameraUpdateFactory.zoomTo((log2(x: 20000.0/i).toFloat())))  
243     } else {  
244         circle?.remove()  
245         circle = map.addCircle(CircleOptions().apply { this: CircleOptions  
246             center(userLocation)  
247             radius( radius: i * 1000.0)  
248         })  
249         map.animateCamera(CameraUpdateFactory.zoomTo((log2(x: 20000.0 / i).toFloat())))  
250     }  
251 }  
252 }
```

Figura 5.14 Funcția pentru trasarea cercului

### 5.4.3. Autentificarea și Înscrierea

O altă componentă importantă a aplicației este autentificarea. Înainte de a se autentifica, utilizatorul trebuie să se înregistreze în baza de date a platformei prin completarea unui formular de înscriere cu câteva date minime de identificare a acestuia. Mai multe date pot fi completate ulterior în secțiunea “Editează profilul”.

După înscriere, utilizatorul se poate autentifica introducând credențialele stabilite anterior, emailul și parola.

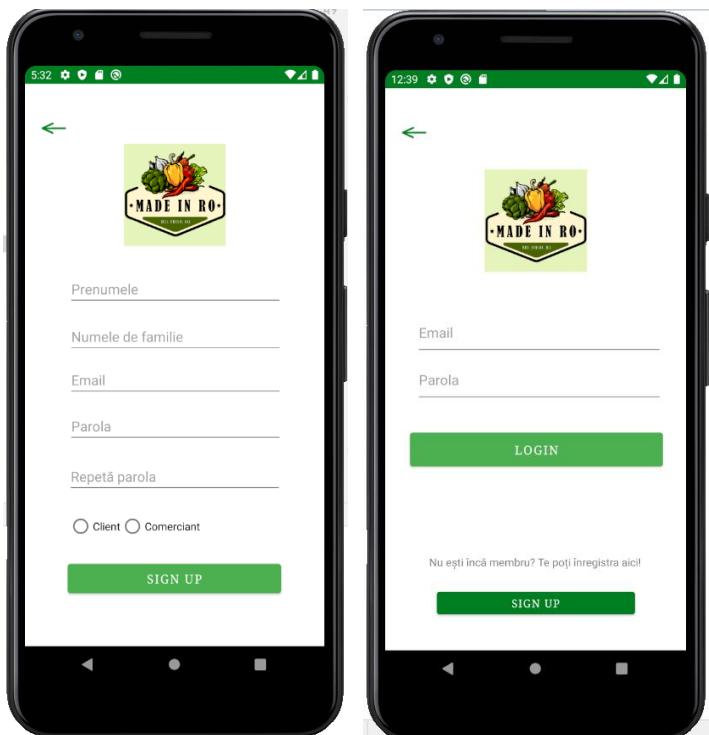


Figura 5.15 Capturi de ecran pentru înscriere și autentificare

## Capitolul 5

### 5.4.4. Liste și Filtre

Cum am menționat și în capitolul “Proiectarea elementelor grafice ale interfeței utilizator”, în cadrul acestui proiect s-au utilizat mai multe liste de tip RecyclerView. Fiecare dintre aceste liste are nevoie de o clasa Adapter, care să adapteze, după cum îi spune și numele, elementele listelor după anumite reguli, stabilite de programator. Pe lângă adaptare, pentru cele mai multe liste s-a implementat și o funcție de căutare, care permite utilizatorului să caute diverse informații care s-ar afla în lista respectivă.

Un exemplu de astfel de funcție îl constituie funcția pentru căutarea unui utilizator în lista de utilizatori pe care o gestionează administratorul, după cum se poate observa și în următoarea figură:



```
68     private fun filter(text: String) {
69         val filteredlist: ArrayList<User> = ArrayList()
70
71         for (item in DataExample.getUsersList()) {
72             if (item.lastName.lowercase(Locale.getDefault()).contains(text.lowercase(Locale.getDefault())) ||
73                 item.firstName.lowercase(Locale.getDefault()).contains(text.lowercase(Locale.getDefault()))) {
74                 filteredlist.add(item)
75             }
76         }
77         if (filteredlist.isEmpty()) {
78             Toast.makeText(context, text: "Nu s-a găsit în listă", Toast.LENGTH_SHORT).show()
79         } else {
80             usersAdapter.filterList(filteredlist)
81         }
82     }
```

Figura 5.16 Tip de funcție de căutare în cadrul listelor

### 5.4.5. Procesul de Vânzare-Cumpărare

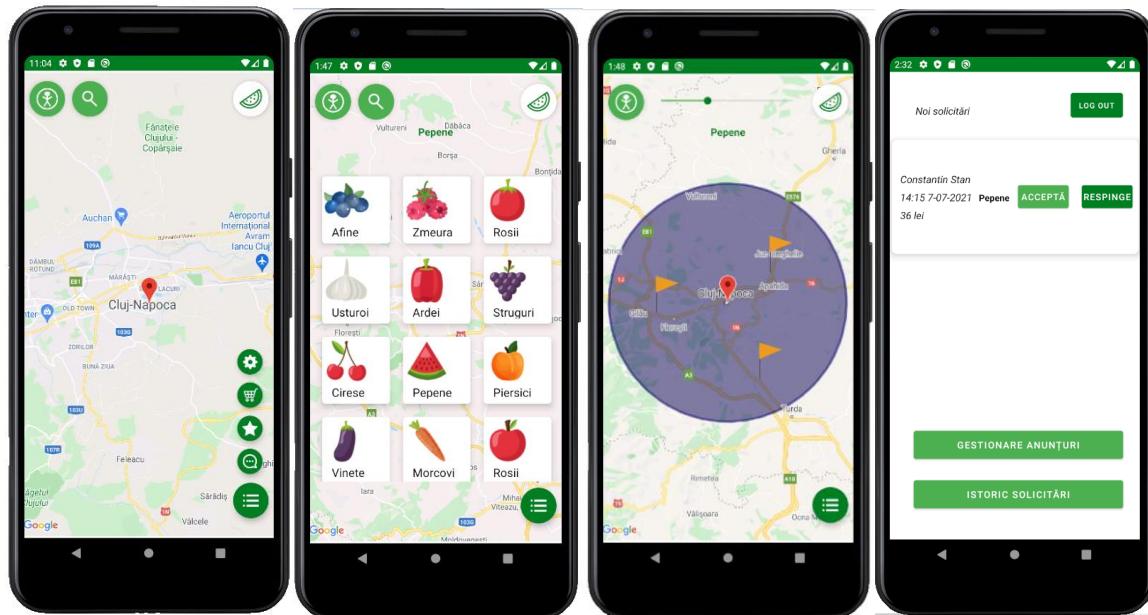


Figura 5.17 Capturi de ecran – proces vânzare-cumpărare

## Capitolul 5

Clientul se autentifică în cont. Pentru a cumpăra anumite produse, acesta apasă pe butonul din dreapta sus, pentru a vedea lista de produse din baza de date și a le selecta pe cele pe care le dorește. După ce le-a selectat, acestea vor apărea deasupra listei, scrise cu verde, iar după ce a selectat și zona în care dorește să le achiziționeze, standurile care conțin aceste produse vor apărea cu steaguri în zona selectată.

După ce acesta plasează comanda, ea va apărea în contul comerciantului sub formă de notificare, cu informațiile relevante. Aceasta poate să confirme sau să refuze cererea, în funcție de stocul său.

### 5.4.6. Conexiunea cu baza de date

Datorită instrumentului Firebase Assistant, de care dispune Android Studio, procesul pentru stabilirea conexiunii cu baza de date a devenit foarte simplu. Totul constă în adăugarea dependințelor în mod corect și a inițializării bazei de date prin linia de cod de la pasul numărul trei:

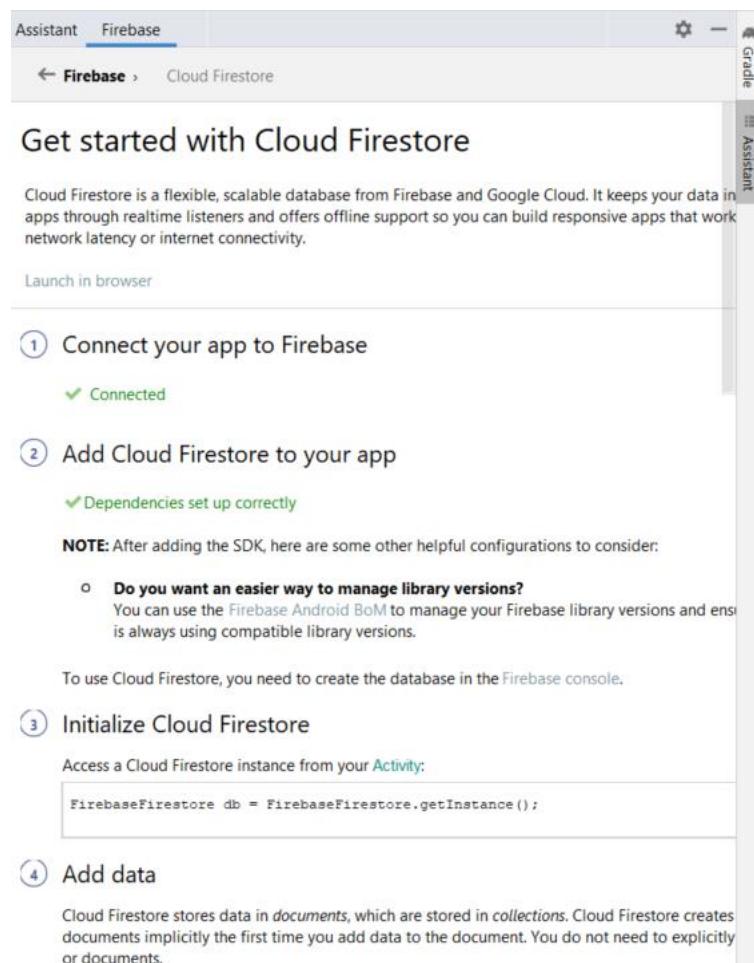
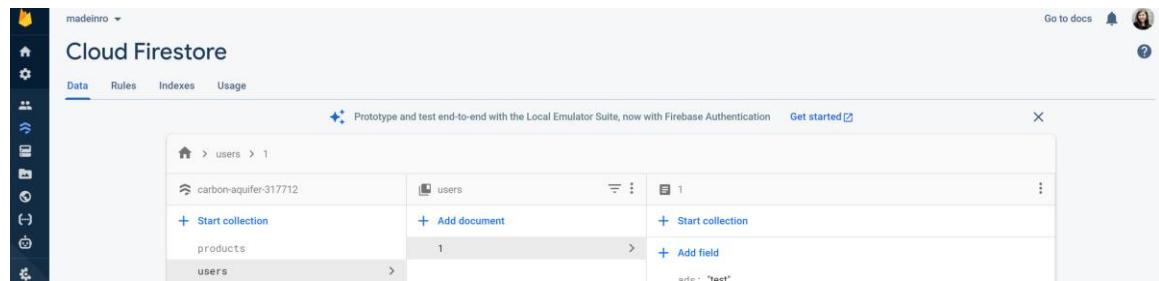


Figura 5.18 Firebase Assistant – Conexiunea în Cloud

După stabilirea conexiunii, se poate vizualiza în cloud:

## Capitolul 5



The screenshot shows the Firebase Cloud Firestore interface. The left sidebar has icons for Home, Settings, and other services. The main header says "Cloud Firestore" and "madeinro". The top navigation bar includes "Data", "Rules", "Indexes", "Usage", "Go to docs", a bell icon, and a user profile. A banner at the top right says "Prototype and test end-to-end with the Local Emulator Suite, now with Firebase Authentication" and "Get started". The main area shows a hierarchical view: "carbon-aquifer-317712" > "users" > "1". The "users" collection has an "Add document" button. The "1" document has an "Add field" button. The URL bar at the bottom shows "users/1/edit".

Figura 5.19 Captură de ecran - conexiune baza de date Firebase

## Capitolul 6. Testare și Validare

Testarea software-ului este o componentă importantă a unui proiect de dezvoltare a unei aplicații mobile care ajută la determinarea faptului dacă aplicația software îndeplinește cerințele de implementare sau nu.

Implementarea aplicației MadeInRo s-a făcut folosind paradigma Test Driven Development, care înseamnă că după planificarea funcționalităților și descrierea lor în detaliu, dar înainte de a începe implementarea, sunt create cazurile de testare pentru aplicație, teste care vor fi rulate în permanență la adăugarea unei noi funcționalități. Dacă o funcționalitate adăugată duce la erori de testare, atunci aceasta trebuie revizuită. Această abordare ne scutește de foarte multe erori care se pot complica în timp și ne ajută să le identificăm mult mai rapid, când implementarea este încă în curs. Alternativa, a scrie teste după ce o funcționalitate a fost implementată, este mult mai anevoieasă, unele probleme identificate târziu necesitând modificări și refactorizări uneori masive. Într-un mediu Test Driven erorile sunt identificate din timp și rezolvate pe loc.

Aplicația implementează Continuous Integration folosind pipeline-urile de testare puse la dispozitie de GitLab. În momentul în care o nouă versiune este încărcată în serviciul de versionare (Git), este rulat un nou pipeline de testare, în care toate cazurile de test sunt analizate. În cazul în care apar erori, deployment-ul eșuează și ultimele funcționalități adăugate trebuie analizate pentru a identifica problema. Cazurile de testare au fost implementate sub forma de teste JUnit.

Cazurile de testare și scenariile de testare sunt cele două aspecte importante ale testelor software folosite pentru a determina cerințele unui proiect și pentru a evalua posibilele rezultate și pentru a testa funcționalitatea aplicației.

### 6.1. Cazuri de test

Un caz de testare este un document detaliat care constă dintr-un set de variabile și condiții pentru a stabili dacă aplicația este compatibilă cu cerințele de implementare și funcționează așa cum a fost planificat inițial. Un caz de testare include o documentație detaliată care cuprinde totul, de la premisele, intrările și precondițiile la procedura de testare, rezultatele așteptate și condițiile ulterioare execuțării.

#### 6.1.1. Testarea unitară

Testele de tip Unit Test scrise pentru aplicație acoperă serviciile și logica de backend. Operațiile de CRUD și cererile mai complexe de date sunt validate la nivel de metode prin teste care au un anumit set de date de intrare și un set de date așteptat la ieșire.

Cazurile de test pentru aplicație conțin:

1. Cazuri de test pentru utilizatori: creare, modificare, ștergere, obținerea listei de utilizatori (pentru administratori), obținerea informațiilor de profil pentru un utilizator.
2. Cazuri de test pentru permisiunile necesare în aplicație: pentru a putea folosi aplicația este nevoie de anumite permisiuni, fără care funcționalitățile nu pot fi utilizate. Pentru a putea folosi harta este nevoie de locație, pentru a putea adăuga o poza de profil sau cu produsele este nevoie de acces la aparatul foto

sau la galeria foto. Toate aceste permișii sau absența lor reprezintă cazuri de test care sunt acoperite prin Unit Tests.

3. Cazuri de teste pentru autentificare, sesiune și roluri utilizatori: pentru a folosi toate funcționalitățile aplicației este nevoie de autentificarea utilizatorilor, care primesc un token care este validat la fiecare cerere. Utilizatorii au roluri diferite, putând fi cumpărători, comercianți sau administratori. Un utilizator nu poate accesa o funcționalitate care nu este în drepturile rolului său, primind un mesaj de avertizare în cazul în care încearcă să acceseze aceste funcții prin diferite mijloace. Toate aceste cazuri sunt acoperite prin testarea accesării serviciilor de către diferiți utilizatori, autentificați sau nu.
4. Cazuri de utilizare pentru hartă: accesarea hărții și a informațiilor de pe aceasta este funcționalitatea centrală a aplicației și cea mai intens utilizată de clienți și comercianți, și pentru aceasta este nevoie de cazuri extensive de test pentru toate funcționalitățile.

### 6.2. Scenarii de test

Scenariul de testare este un set colectiv de cazuri de testare, un scenariu care determină aspectele pozitive și negative ale proiectului, evaluând posibilele rezultate pentru a identifica eventualele deficiențe din program. Acesta este nivelul următor de testare software care implică o serie de etape încorporate pentru a ușura munca de cazuri de testare. Este o procedură de testare cu mai multe cazuri de testare care ajută la testarea programului pentru potențiale erori și pentru a asigura că funcționalitatea finală este cea așteptată. Spre deosebire de cazurile de testare, ele sunt mai puțin descriptive și sunt menite să ofere parcursul unui caz de utilizare, decât să intre în detaliu.

#### 6.2.1. Testare manuală

Testarea manuală s-a făcut de către un număr de utilizatori restrâns care a primit acces la versiunea de testare a aplicației. Fiecare utilizator a primit un număr de scenarii de testare pe care a trebuit să le verifice cu multiple date de intrare. De asemenea, utilizatorii au putut accesa aplicația în mod liber, pentru a identifica orice alte probleme care pot apărea în momentul utilizării acestieia.

Câteva din scenariile de testare sunt:

1. Căutarea unei destinații pe hartă cu un utilizator neautentificat. Căutarea de comercianți pe o arie în apropiere.
2. Crearea unui utilizator nou de tip client (consumator) și autentificarea acestuia în aplicație. Căutarea unui produs în apropiere.
3. Crearea unui utilizator nou de tip comerciant și autentificarea în aplicație. Adăugarea unui nou produs în listă.
4. Contactarea în chat a unui comerciant, trimiterea și primirea de mesaje.
5. Modificarea setărilor aplicației.
6. Generarea unui raport de vânzări pentru o anumită perioadă

După testarea fiecărui scenariu utilizatorii au fost rugați să completeze un formular de identificare a problemelor și un formular de feedback în ce privește experiența utilizării aplicației.

### 6.2.2. Înregistrare teste Espresso

Mediul de dezvoltare folosit pentru implementarea aplicației, Android Studio, pune la dispoziție un instrument puternic de testare și creare de test, numit Espresso. Acest instrument permite înregistrarea fiecărui pas dintr-un scenariu de testare al aplicației și adăugarea de aserțiuni și notificări pe variabile. Aceasta a fost unul din principalele metode folosite în testare pentru a asigura funcționarea.

Testele Espresso sunt simulate într-un emulator, similar cu utilizarea propriu-zisă a aplicației, iar fiecare acțiune efectuată în momentul înregistrării este trecută într-un log. La fiecare etapă pot fi adăugate condiții de validare. După înregistrarea unui test, acesta poate fi rulat oricând, după efectuarea unor modificări în implementare.

Pașii de urmat pentru înregistrarea unui test Espresso:

1. Pornirea în execuție a instrumentului de înregistrare Record Espresso Test:

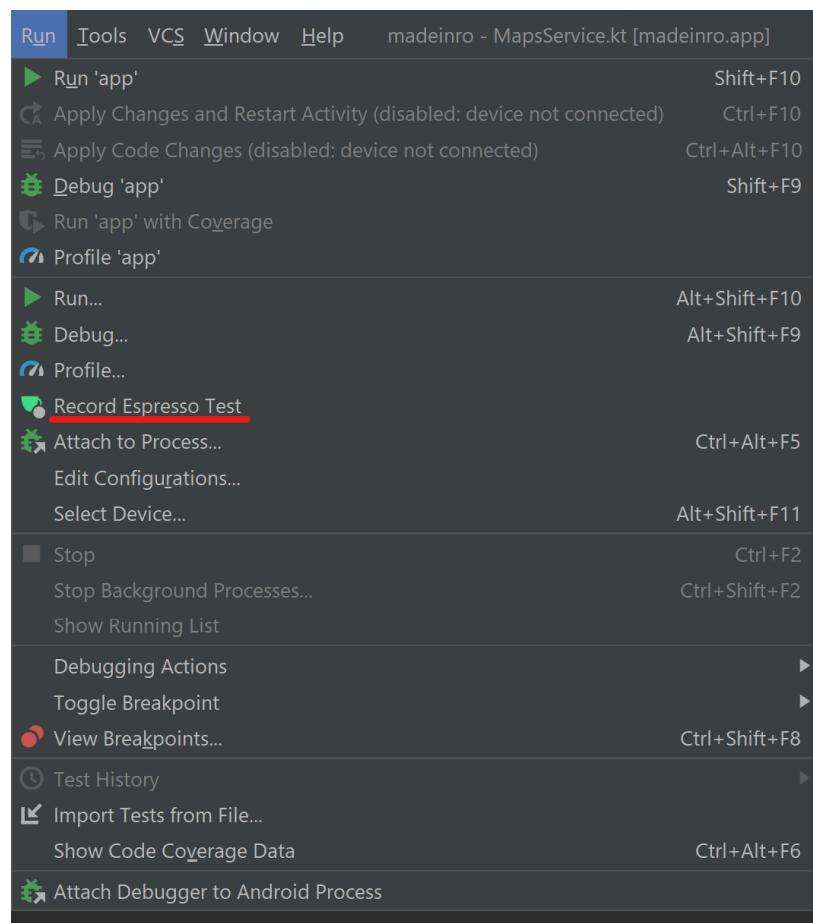


Figura 6.1 Captură de ecran pentru pornirea în execuție a Record Espresso Test

2. Înregistrarea pașilor de execuție

## Capitolul 6

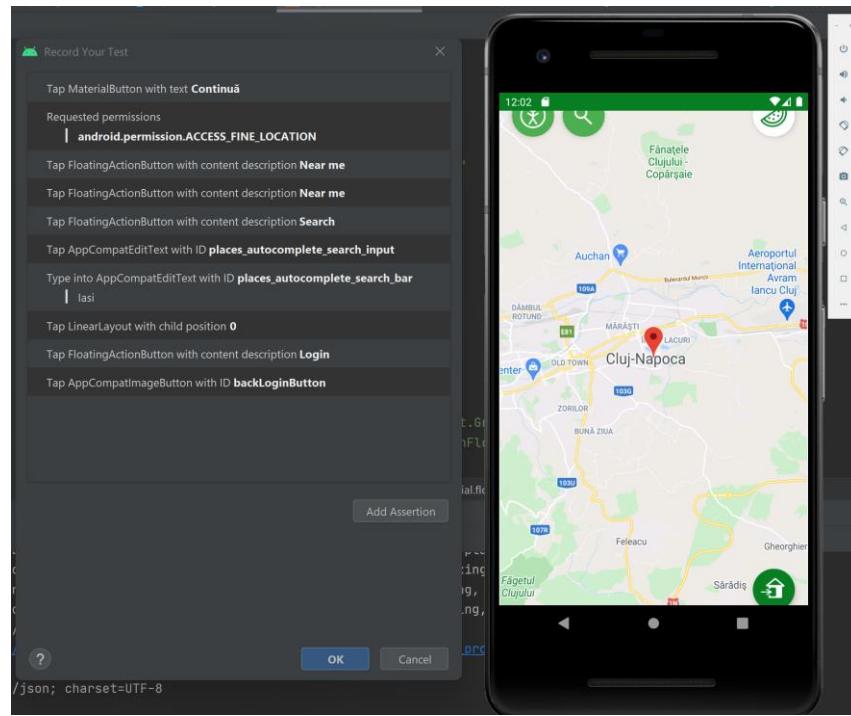


Figura 6.2 Captură de ecran pentru înregistrarea pașilor de execuție

### 3. Adăugarea aserțiunilor

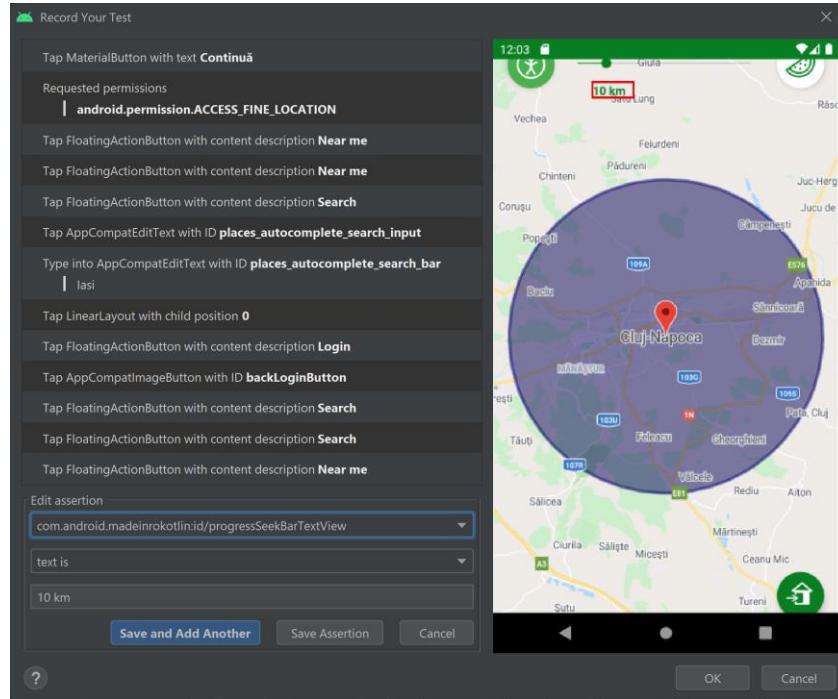


Figura 6.3 Captură de ecran pentru adăugarea aserțiunilor

## Capitolul 6

### 4. Salvarea înregistrării ca test

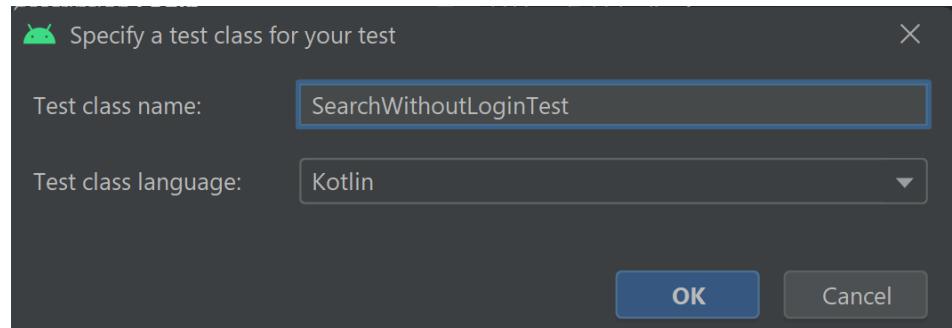


Figura 6.4 Captură de ecran pentru salvarea înregistrării ca test

### 5. Rularea testării în cazul unor modificări

Toate aceste metode de testare și validare implementate asigură că erorile de implementare sunt descoperite la timp și pot fi corectate rapid.

## Capitolul 7. Manual de Instalare și Utilizare

### 7.1. Instalarea platformei

Pentru instalarea aplicației mobile “Made in Ro” pe un dispozitiv electronic, mai multe condiții trebuie să fie îndeplinite. Prima condiție este ca dispozitivul să aibă sistem de operare Android cu versiunea cel puțin egală cu 10.0, să aibă sistem GPS, să aibă un spațiu de stocare liber de aproximativ 10-12MB și să fie conectat la internet.

Resursele necesare pentru dezvoltarea platformei de e-commerce sunt:

- Sistem de operare: Android (cel puțin o versiune egală cu 10.0)
- Sistem GPS
- Spațiu liber de aproximativ 10-12 MB
- Conexiune la internet

Pentru a instalarea aplicației, se poate accesa linkul care se găsește la referința [21] sau se poate căuta aplicația pe Google Play și instala de acolo:

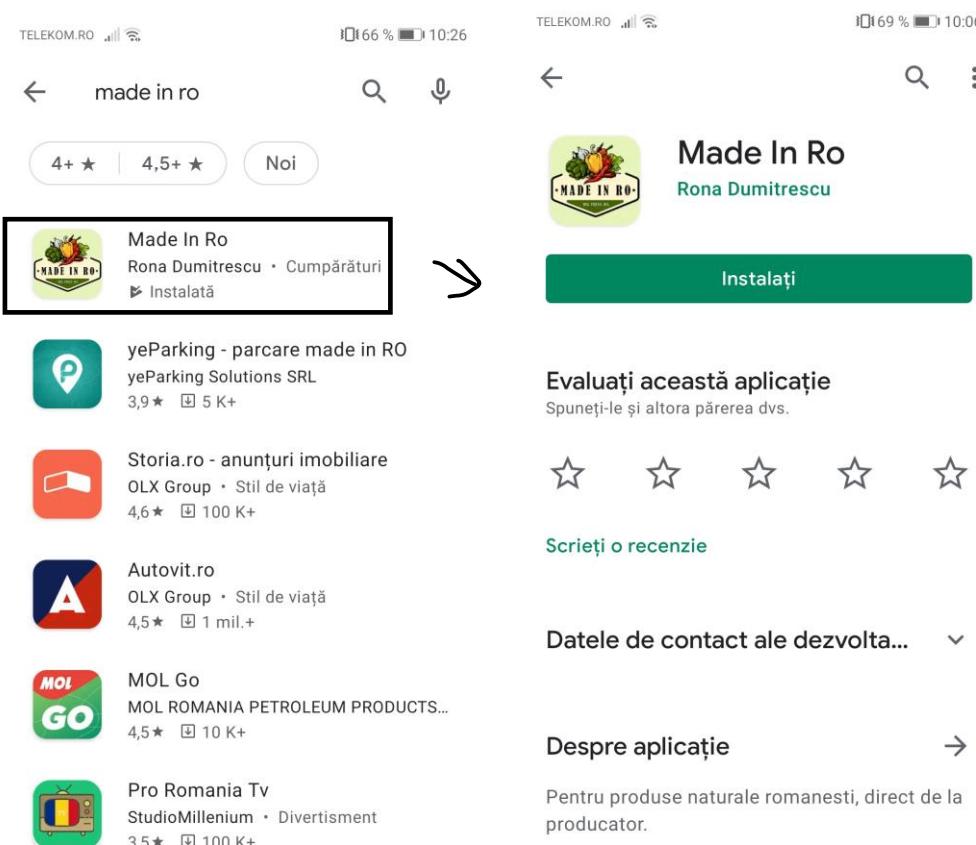


Figura 7.1 Instalarea aplicației din Google Play

## 7.2. Manual de utilizare

Indiferent de tipul de utilizator care dorește să acceseze aplicația, prima pagină cu care acesta va interacționa va fi pagina în care i se cere accesul la locație pe durata utilizării aplicației, deoarece proiectul are la bază diverse funcționalități de navigabilitate.

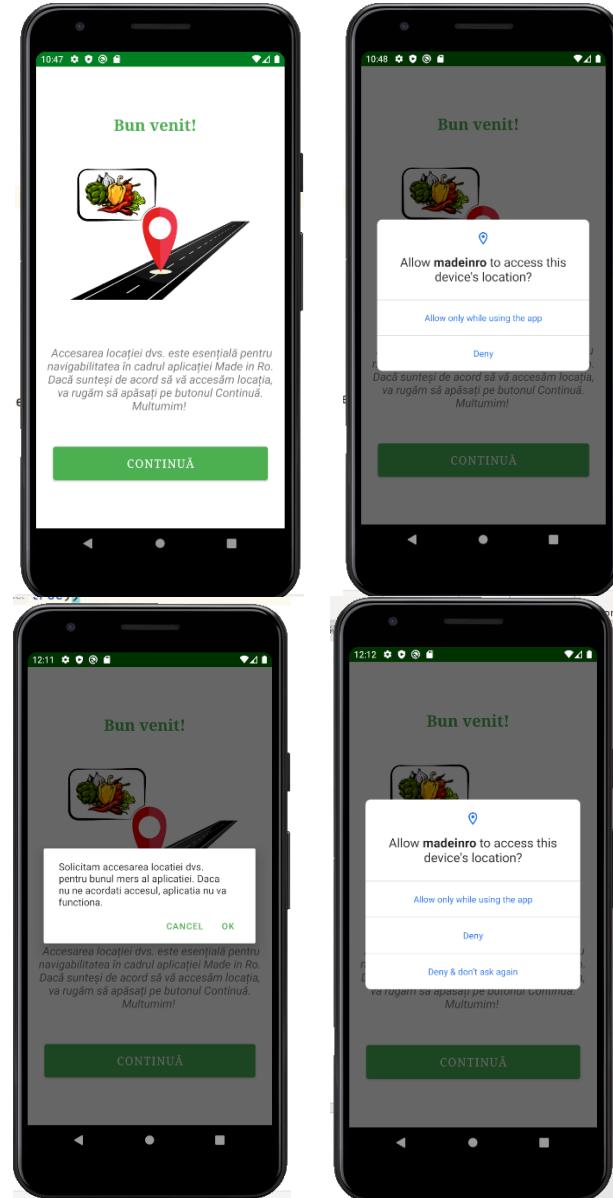


Figura 7.2 Prima pagină din aplicație – Solicitarea accesului la locație

Pentru a intra în aplicație, utilizatorul apasă pe butonul "Continuă". Dacă este de acord cu oferirea accesului la locația sa pe durata utilizării aplicației, acesta selectează prima opțiune și intră în aplicație. Dacă nu este de acord, selectează cea de-a doua opțiune și este notificat legat de faptul că nu va putea utiliza corespunzător aplicația dacă nu permite accesul. Dacă selectează ok, îi apare din nou o fereastră de dialog în care ar putea selecta în plus față de opțiunile de dinainte, opțiunea a treia "Deny & don't ask me again" și astfel, să iasă din aplicație.

### 7.2.1. Utilizare Actor Oaspete

Dacă este de acord cu oferirea accesului, intră în aplicație și nu i se va mai afișa acest ecran următoarea dată când va intra în aplicație, ci va apărea ecranul cu o hrată și patru butoane, în ordinea stanga-sus până în dreapta-jos, pe ecran apar: butonul “near mme”, butonul “search”, butonul “produse alimentare” și butonul de “login”. În următoarele capturi de ecran sunt demonstreate modurile de funcționarea ale acestora:

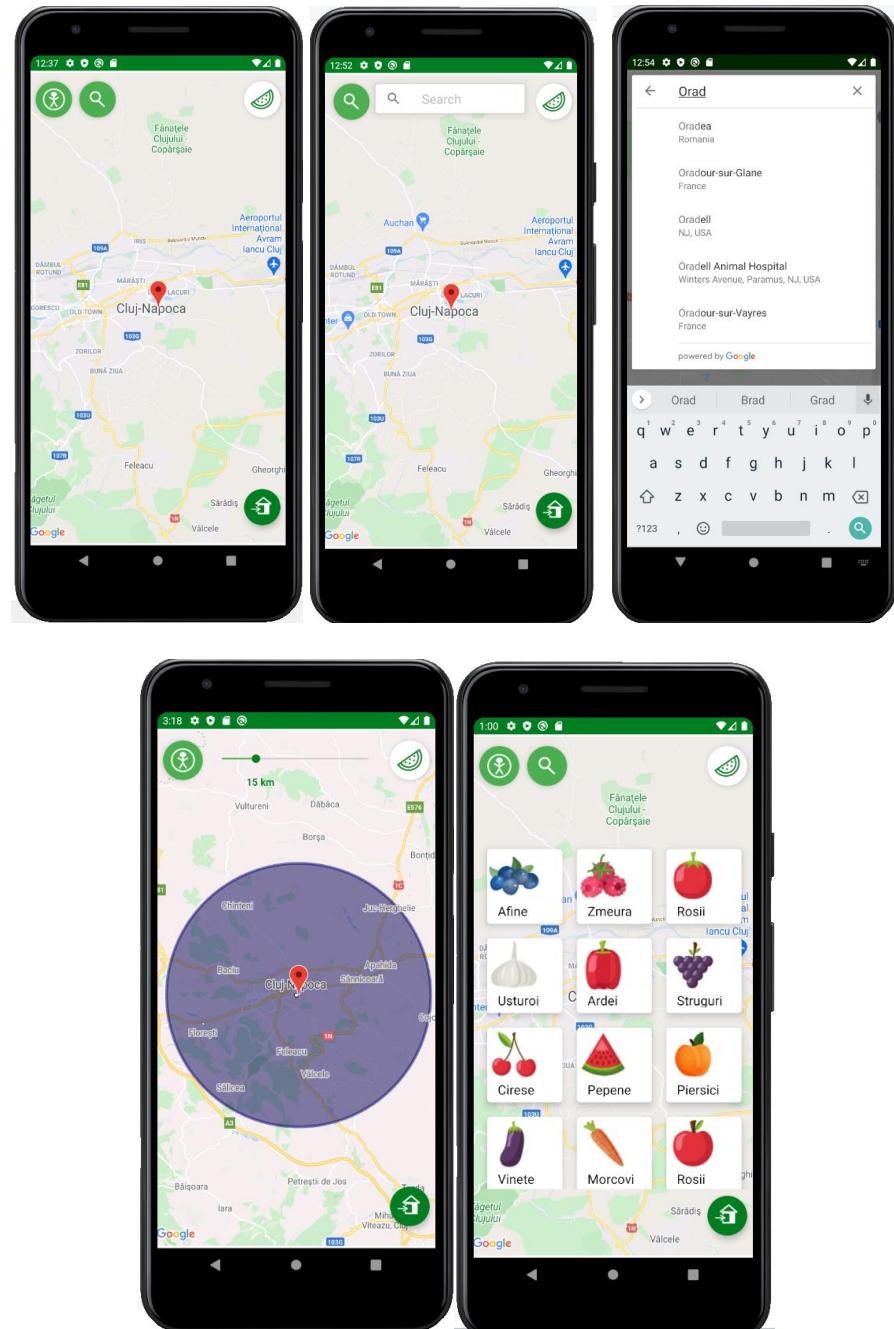


Figura 7.3 Capturi de ecran cu acțiunile din pagina principală a Oaspetelui

Dacă Oaspetele dorește să își facă un cont de Client sau de Comerçant, sau să intre în contul său, acesta poate apăsa pe butonul din dreapta jos, de login, pentru a naviga către următoarele două ecrane:

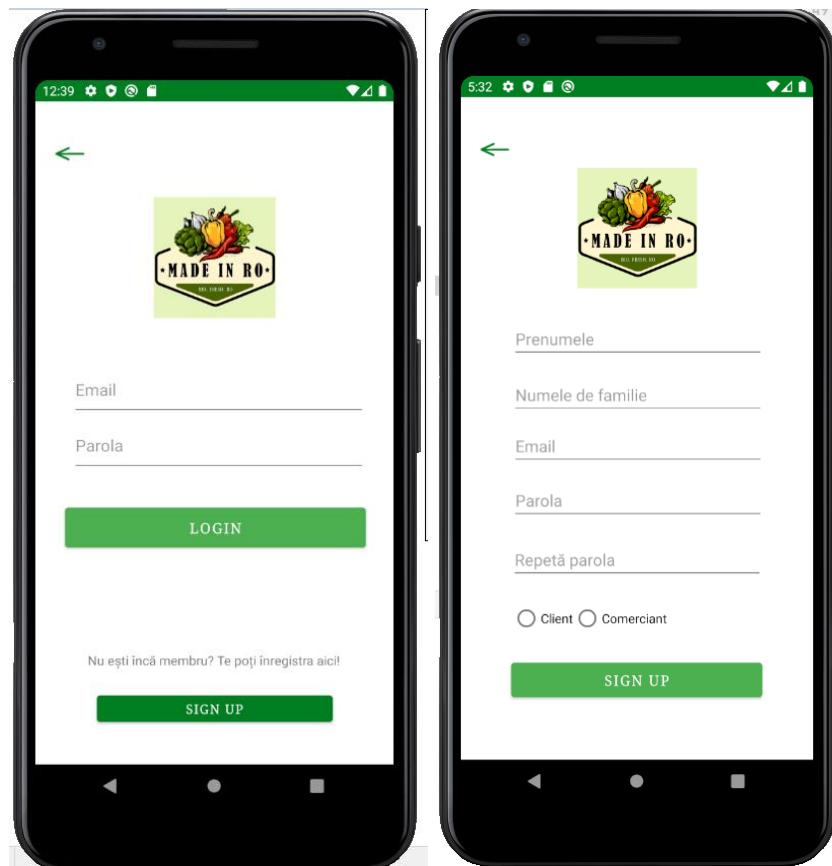


Figura 7.4 Capturi de ecran pentru autentificarea și înregistrarea în cont

Dacă informațiile introduse se vor dovedi a fi valide, acesta va intra în contul corespunzător.

### 7.2.2. Utilizare Actor Administrator

Credențialele pentru contul de administrator sunt furnizate de către dezvoltatori. După ce administratorul le introduce, acesta ajunge în pagina sa principală, din care poate realiza mai multe acțiuni. Acesta poate vizualiza sugestiile și observațiile trimise de utilizatori către el chiar în lista de observații din partea de sus a ecranului.

Administratorul poate gestiona utilizatori. Dacă cineva semnalează o problemă legată de un utilizator, în funcție de gravitatea situației, acesta poate lua măsuri precum ștergerea contului de utilizator (client sau comerciant). De asemenea, acesta poate gestiona lista de produse din baza de date a aplicației. În caz că este sesizat de absența unui produs, acesta îl poate adăuga. Ambele ecrane de gestiune au implementate funcția de căutare.

## Capitolul 7

Acțiunile și ecranele corespunzătoare pentru actorul administrator sunt prezentate în următoarele figuri:

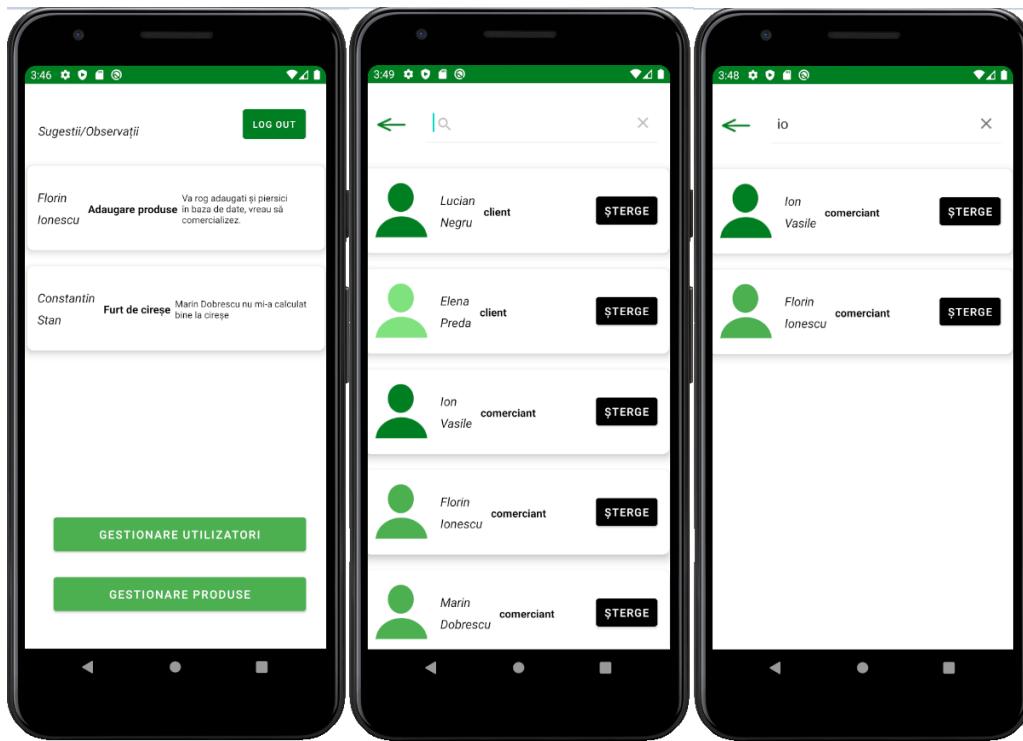


Figura 7.5 Capturi de ecran - Gestiunea utilizatorilor de către administrator

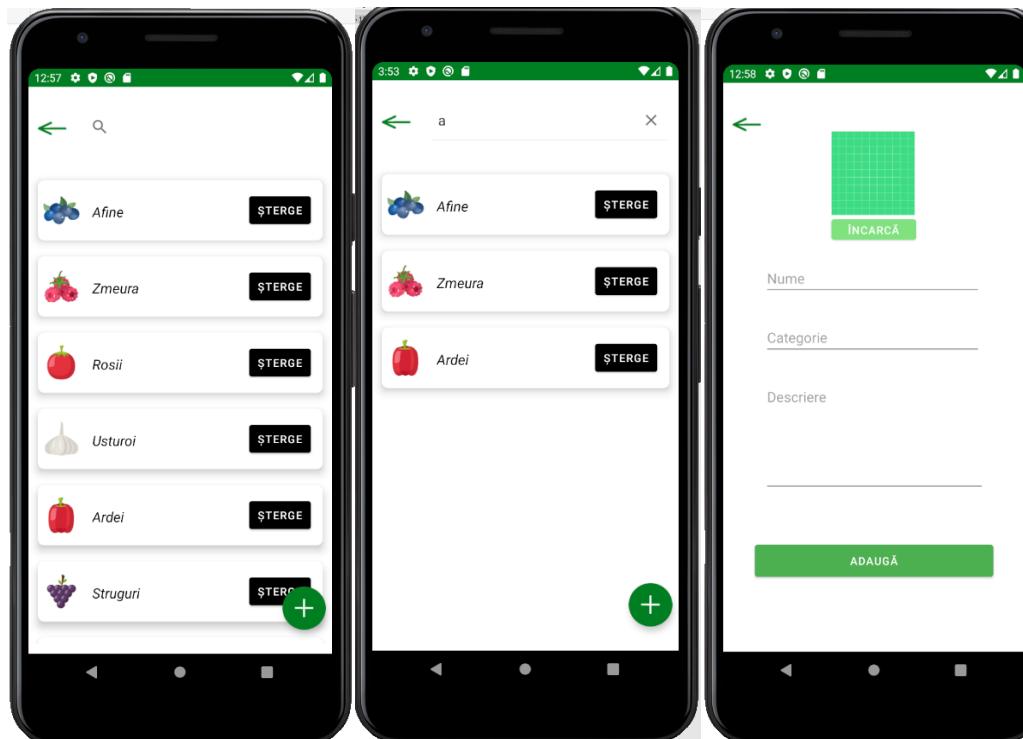


Figura 7.6 Capturi de ecran - Gestiunea produselor de către administrator

### 7.2.3. Utilizare Actor Client

Utilizatorul Client poate realiza exact aceleasi acțiuni de căutare, selecție și navigare ca actorul Oaspete, de aceea, nu au mai fost ilustrate și în acest subcapitol. Pe lângă procesul de vânzare cumpărare illustrat în secțiunea 5.4.5, clientul își poate șterge și edita profilul, poate trimite sugestii sau observații către administrator și se poate delega din cont, acțiuni prezentate în următoarele capturi de ecrane:

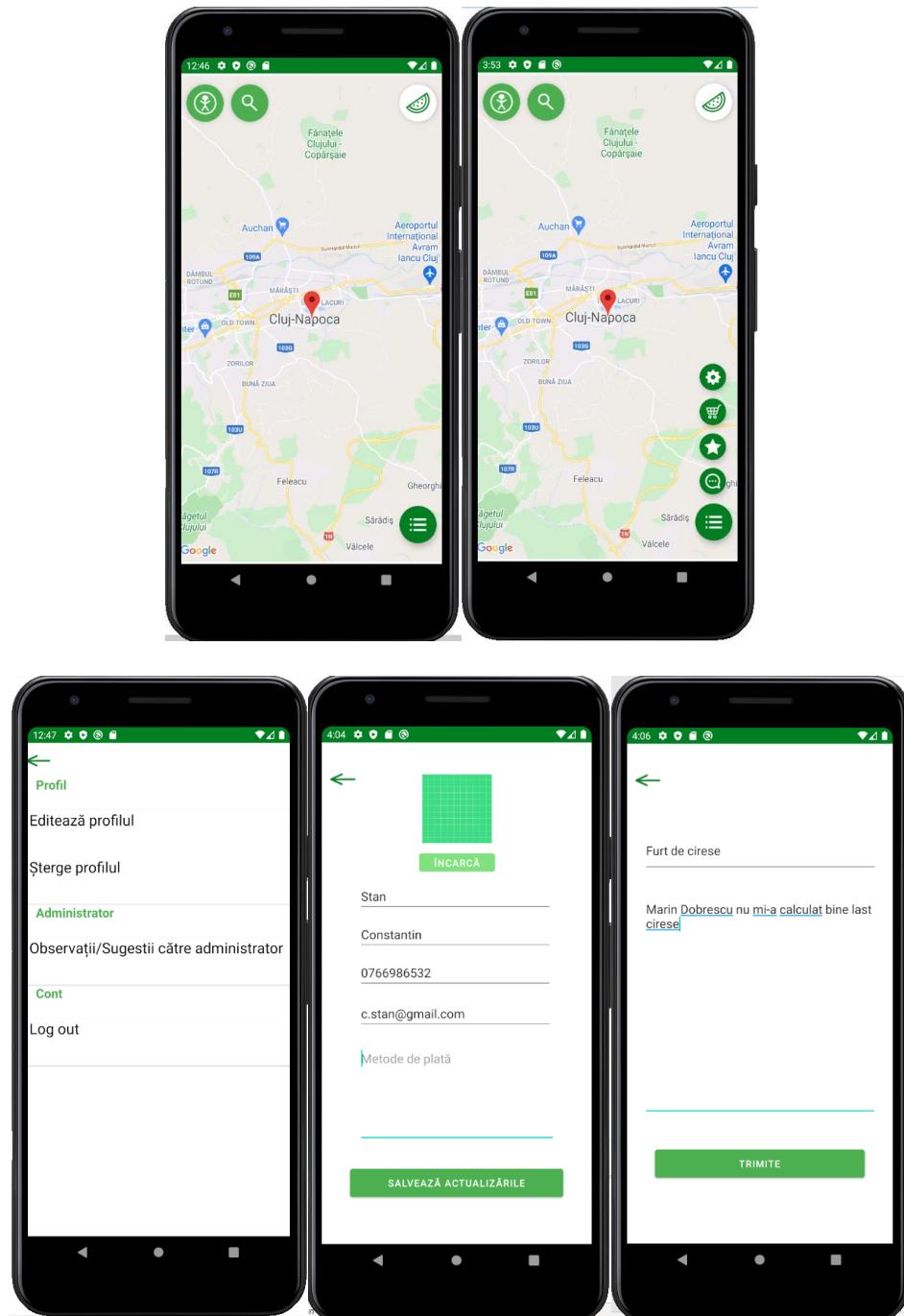


Figura 7.7 Capturi de ecran corespunzătoare actorului Client

### 7.2.4. Utilizare Actor Comerciant

După ce s-a autentificat cu credențialele corespunzătoare, Comerciantul intră în ecranul principal al contului său. În partea de sus a ecranului poate vizualiza solicitările de cumpărare a produselor sale venite de la diversi clienți. În funcție de stocul său, acesta poate să le gestioneze accepte sau să le respingă. Fie că va alege o acțiune sau cealaltă, acesta le va putea vizualiza intrând la istoric solicitări, cu statusul specific comenzi “acceptată” sau “respinsă”.

Dacă dorește, Comerciantul are posibilitatea de a adăuga mai multe anunțuri de comercializare selectând un nume, o locație, produsele pe care le comercializează și alte informații necesare. Acțiunile specifice acestui actor sunt prezentate în urmărele figuri:

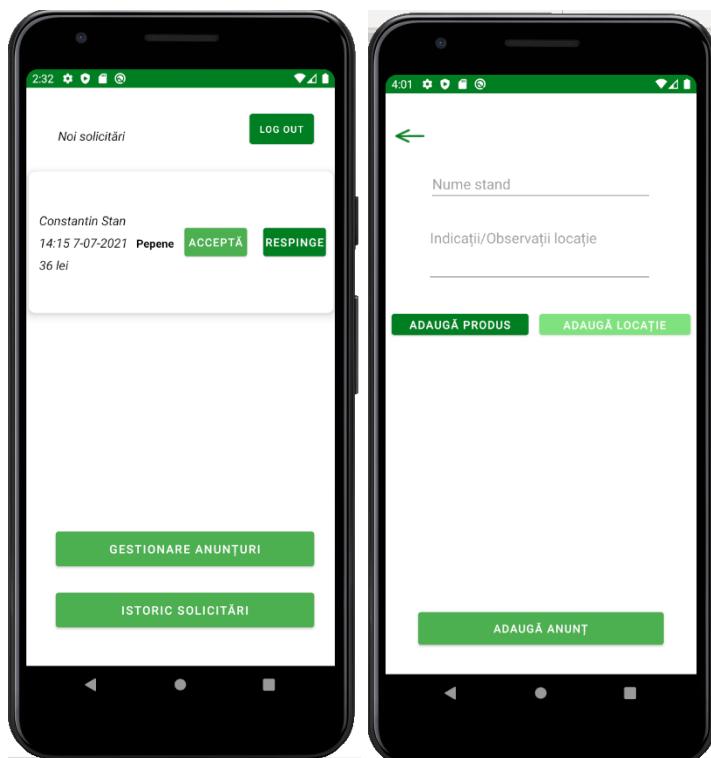


Figura 7.8 Capturi de ecran corespunzătoarea actorului Comerciant

## Capitolul 8. Concluzii

### 8.1. Rezultate obținute și contribuții personale

Scopul proiectului a fost de a construi o platformă destinată comercializării produselor naturale din grădinile micilor producători, pentru ca oamenii din zonă sau care ajung în acea zonă, să beneficieze de produse naturale cu adevărat “bio” de la producătorii care, de regulă, se află în fața gospodăriei cu câteva produse și un anunț specific atașat.

Realizarea aplicației a îndeplinit mai multe nevoi și cerințe detaliate în amănunt în cel de-al doilea capitol.

În primul rând, în aplicație am integrat hărțile furnizate de Google Maps prin intermediul celor două chei Google Maps API pentru versiunile de debug și release, cât și funcționalitățile puse la dispoziție, precum: desenarea pe hartă, animarea tranzițiilor, mărirea și micșorarea acesteia, adăugarea mai multor “pinuri” cu informații noi pe hartă etc.

O nevoie îndeplinită a fost aceea de a realiza o platformă de tip mobil pentru identificarea producătorilor din zonă sau care ajung într-o anume zonă, atât prin funcționalitatea “Near me”, care afișează producătorii cu un anumit tip de produse agroalimentare selectate de client pe o anumită rază și reprezentată grafic printr-un cerc, fie prin selectarea unei locații și afișarea oportunităților de cumpărare care se află pe drumul până la aceasta, trasate și identificate de către aplicație, în funcție de selecțiile clientului.

Alte cerințe realizate cu succes în cadrul aplicației mobile implementate au fost: implementarea unui chat pentru comunicarea între utilizatori și a unui sistem de vânzare-cumpărare de pe telefonul mobil în ce privește produsele din sectorul legumicol, care cuprinde gestionarea anunțurilor de către comercianți, adăugarea în coș a produselor dorite de către clienți, editarea produselor din coș, plasarea comenzi etc.

De asemenea, unul dintre obiectivele propuse a fost și realizarea unui design intuitiv în ceea ce privește interfața grafică a utilizatorului. Utilizatorii se pot bucura acum de o interfață simplă, eficientă, caracterizată printr-o paletă de nuanțe de verde, care să le ofere acestora senzația de prospețime și relaxare.

Un ultim obiectiv a fost de a pune aplicația în Google Play, obiectiv atins prin crearea unui cont personal pe Google Developers [20] și parcurgerea pașilor necesari pentru încărcarea fișierului “.apk” în baza de date a magazinului Google Play.

### 8.2. Dezvoltări ulterioare

Una dintre cele mai mari provocări cu care m-am confruntat în realizarea acestei lucrări de diplomă a fost aproximarea duratei de timp necesară realizării unui lucru, indiferent dacă acesta a fost legat de partea de implementare sau de partea de redactare. Cel mai probabil este o provocare de care mulți proiectanți se lovesc, indiferent de domeniu.

Înainte și de a scrie titlul lucrării, îmi propusesem să realizez o aplicație cu toate funcționalitățile utile acesteia la care am putut să mă gândesc în acel moment, însă, parcă am uitat să iau în considerare că pentru realizarea acestui proiect lucrează un singur om, nu o echipă de dezvoltatori și testeri. În timp ce implementam diverse funcționalități, se

nășteau noi idei de dezvoltare. De aceea, am împărțit secțiunea “Dezvoltări ulterioare” în două subsecțiuni – “Îmbunătățiri funcționalități existente” și „Funcționalități noi”, pentru a enumera toate îmbunătățirile notate de-a lungul procesului de dezvoltare, care consider că ar trebui aduse aplicației, și, de asemenea, funcționalități noi, care ar largi gama de utilizatori sănătoși.

### *8.2.1. Îmbunătățiri funcționalități existente*

Pe viitor, la autentificare, ar fi necesar să se adauge și opțiunea de autentificare prin intermediul conturilor de Facebook și de Google, motiv pentru care s-a și ales la început baza de date Firebase, care face implementarea acestei părți de cod foarte ușoară. De asemenea, alte funcționalități necesare aici ar mai fi și opțiunea de “am uitat parola”, prin care utilizatorul își poate reseta parola pe mail și verificarea identității în doi pași, prin intermediul adresei de e-mail și al numărului de telefon, din motive de securitate.

Îmbunătățirile pe care le văd necesare pentru chat sunt: adăugarea funcționalității pentru trimiterea de poze/videoclipuri, atașamente, gifs și stickers, editarea textului pentru a scrie “bold”, “italic” sau “underline” sau alte formatare ale textului, implementarea posibilității de a reacționa la mesajul primit și, de asemenea, redactarea unui mesaj prin intermediul comenzilor vocale.

Pentru partea de navigare, o funcționalitate utilă ar fi opțiunea de adăugare a locațiilor intermediare până la destinație, deplasarea poziției camerei hărții în același timp cu poziția utilizatorului și calcularea timpului rămas până va ajunge la destinație. De asemenea, pentru a putea utiliza și alte aplicații în timp ce utilizatorul își parcurge drumul utilizând această aplicație, o îmbunătățire de adus ar fi și accesarea locației prin solicitarea accesului la locația din background, caz în care, ieșirea din aplicație nu ar determina închiderea sa completă, ci rularea în background.

### *8.2.2. Funcționalități noi*

Datorită faptului că aplicația este destinată și utilizatorilor cu mai puțină experiență în domeniul tehnologiei, o îmbunătățire care ar putea fi adusă pentru a face utilizarea acesteia cât mai ușoară, ar fi ca, la început, prima dată când intră în aplicație, să fi învățați cum să o utilizeze cu ajutorul unui scurt tutorial alcătuit din săgeți, indicații și demonstrații scrute acolo unde este cazul.

Pentru siguranța șoferilor, ar fi un plus ca aplicația să aibă integrat un bot vocal, pentru a nu distrage atenția șoferului de la drum pentru a citi diverse informații. Funcționalitatea de vocale a devenit un “must-have” pentru aplicațiile care vizează domeniul de navigabilitate, transport, în care atenția distributivă nu este o opțiune. Prin adăugarea de comenzi vocale la diverse aplicații utilizate în trafic se va reduce riscul de accidente, iar interacțiunea dintre utilizator și platformă va fi mult mai la îndemâna. [1] De asemenea, tot încadrat la acest capitol de navigare și comenzi vocale, consider că ar fi utilă și adăugarea de mementouri pentru o mai bună organizare a utilizatorului. Aceasta își poate nota diverse obiective pe care le are de înăpărat în funcție de o anumită zonă și când ajunge în zona respectivă, platforma să îi amintească ce obiectiv are de realizat.

Un mare beneficiu ar putea fi adus comercianților prin generarea de rapoarte, care să îi ajute să își alcătuiască strategii de comercializare, atât din activitatea lor în cadrul aplicației (spre exemplu: în ce perioadă s-a comercializat cel mai mult un produs), cât și

din activitatea comercianților din zonă (spre exemplu: ce alimente se vând cel mai bine în zonă, dacă cererea este mai mare decât oferta etc.).

În ce privește dezvoltarea pe plan internațional a aplicației mobiel din Google Play "Made in Ro", aceasta nu poate fi posibilă decât prin implementarea unei opțiuni de selecție a limbilor de circulație internațională precum: limba engleză, maghiară, spaniolă, germană, rusă, arabă, chineză, japoneză etc., în care să se poate interacționa cu platforma.

Pentru acest subiect, dezvoltarea în afara țării, dar și pentru plata online în țară, trebuie realizată o documentare mai temeinică în ce privește legislația și taxele percepute pentru astfel de tranzacții.

Momentan, platforma a fost construită astfel încât utilizatorii să identifice cu ușurință producătorii și să se poată deplasa până la locația acestora pentru a cumpăra produsele necesare, însă acest lucru poate fi dezvoltat. Față de alte sisteme de comerț online, platformei implementate îi lipsește o modalitate de livrare acasă. Pentru ca aplicația să fie sustenabilă, trebuie implementat un astfel de sistem. Acest lucru se poate realiza în mai multe moduri: printr-o colaborare cu o firmă de curierat, lăsând la alegerea comerciantului dacă poate garanta un astfel de serviciu etc.

O ultimă dezvoltare, poate chiar cea mai importantă, ar fi dezvoltarea unei componente care să integreze societățile comerciale și să faciliteze comunicarea dintre acestea și producătorii locali, furnizând diferite funcționalități pentru disponibilitatea produselor, aprovizionare, raport cerere-ofertă și alte funcționalități de interes pentru ambele părți care comercializează produsele, dar și pentru clienții care beneficiază de acestea.

## Bibliografie

- [1] A. AKyle Mew, “Android Design Patterns and Best Practice”, Packt Publishing Ltd, Decembrie 2016.
- [2] D. Chaffey, “E-Business and E-Commerce Management. Strategy, Implementation and Practice”, Prentice Hall Financial Times, Fourth edition, 2009.
- [3] Nilanjan Chatterjee1, Souvik Chakraborty, Aakash Decosta, Dr. Asoke Nath, “Real-time Communication Application Based on Android Using Google Firebase”, Volume 6, Issue 4, April 2018 International Journal of Advance Research in Computer Science and Management Studies.
- [4] JJ Geewax ,“Google Cloud Platform in Action”, Manning Publications.
- [5] Adam Gerber, Clifton Craig, “Learn Android Studio. Build Apps Quickly and Effectively”, Apress, 2014.
- [6] Ted Hagos, Android Studio IDE Quick Reference: A Pocket Guide to Android Studio Development, 2019.
- [7] Chunnu Khawas, Pritam Shah, “Application of Firebase in Android App Development-A Study”, International Journal of Computer Applications (0975-8887),Volume 179, No.56, June 2018.
- [8] Ken Kousen, Gradle Recipes for Android: Master the New Build System for Android, O'Reilly Media, June 2016
- [9] S. P. T. KrishnanJose L. Ugia Gonzalez, Building Your Next Big Thing with Google Cloud Platform.A Guide for Developers and Enterprise Architects, Apress 2015.
- [10] Marcin Moskala, Igor Wojda, „Android Development with Kotlin, Pakt Publishing August 2017, Birmingham.
- [11] V. Oliveira, L. Teixeira and F. Ebert, "On the Adoption of Kotlin on Android Development: A Triangulation Study," *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2020, pp. 206-216, doi: 10.1109/SANER48275.2020.9054859
- [12] Kevin Pelgrims, Gradle for Android,Packt Publishing, July 2015
- [13] Gabriel Svennerberg , “Beginning Google Maps API 3”, Apress, 2010.

## Bibliografie

- [14] Lucian Tanasa, Ioan Brumă, „Promovarea produselor agroalimentare tradiționale românești în regiunea de dezvoltare nord-est. Studii și cercetări de economie rurală” X. 258-265, 2011.
- [15] Mike Wolfson, Android Developer Tools Essentials, O'Reilly, 2013.
- [16] Agrihub, <https://agrihub.ro>.
- [17] BigCommerce, Impactul e-commerce, <https://www.bigcommerce.com/articles/ecommerce/#the-impact-of-ecommerce>
- [18] Bune practici FAB Material.io, <https://material.io/components/buttons-floating-action-button>
- [19] Bursa de legume, <https://www.bursadelegume.ro>
- [20] Dezvoltatori Google, <https://developers.google.com>
- [21] Made in Ro, <https://play.google.com/store/apps/details?id=com.rona.madeinro>
- [22] Microsoft, <https://www.microsoft.com/ro-ro/microsoft-365/business-insights-ideas/resources/guide-to-uml-diagramming-and-database-modeling>
- [23] Niveluri Zoom Google Maps, [https://wiki.openstreetmap.org/wiki/Zoom\\_levels](https://wiki.openstreetmap.org/wiki/Zoom_levels)
- [24] Oferte sector legumicol, <https://www.rndr.ro/legume/>
- [25] Paleta de culori Material.io, <https://material.io/resources/color/#!/?view.left=0&view.right=0&primary.color=4CAF50>
- [26] RubyGarage, Mobile app vs. Movile Website, <https://rubygarage.org/blog/mobile-app-vs-mobile-website>
- [27] Statista, Creșterea comerțului electronic mobil, <https://www.statista.com/chart/13139/estimated-worldwide-mobile-e-commerce-sales/>
- [28] UML Diagrams, Diagrame de comunicare, <https://www.uml-diagrams.org/communication-diagrams.html>
- [29] Visual Paradigm Templates, <https://online.visual-paradigm.com/diagrams/templates/>

## Anexa 1. Lista Figurilor

Figura 1.1 Creșterea vânzărilor în mediul online între 1996-2014 în SUA [17] ....	1
Figura 1.2 Creșterea comerțului electronic pe mobil între 2016-2021 [27] .....	2
Figura 1.3 Harta produselor tradiționale din România la nivelul anului 2008 [4]..	3
Figura 3.1 Timpul petrecut pe aplicațiile mobile vs. platformele web [26] .....	10
Figura 3.2 Capturi de ecran cu principalele funcționalități din aplicația Waze....	12
Figura 3.3 Platforma „Bursa de legume” .....	13
Figura 3.4 Platforma „Oferte Sector Legumicul” .....	14
Figura 4.1 Diagrama Use Case .....	17
Figura 5.1 Diagrama generală arhitecturală a sistemului.....	23
Figura 5.2 Diagrama de desfășurarea a aplicației .....	24
Figura 5.3 Diagrama de pachete .....	25
Figura 5.4 Diagrama de Secvențe (cazul general utilizator - client).....	26
Figura 5.5 Diagrama de comunicare .....	27
Figura 5.6 Digrama de tranziție stări (taskuri) [29] .....	28
Figura 5.7 Diagrama fluxului de activități .....	29
Figura 5.8 Diagrama relațională a bazei de date .....	31
Figura 5.9 Diagrama de fragmente și conexiunea între acestea.....	34
Figura 5.10 Paleta de culori utilizată în aplicația mobilă.....	35
Figura 5.11 Fișierul de lucru din Adobe Illustrator .....	35
Figura 5.12 Captură de ecran cu tipul de acces solicitat utilizatorilor .....	36
Figura 5.13 Capturi de ecran - Opțiuni de navigabilitate.....	36
Figura 5.14 Funcția pentru trasarea cercului.....	38
Figura 5.15 Capturi de ecran pentru înscriere și autentificare .....	38
Figura 5.16 Tip de funcție de căutare în cadrul listelor .....	39
Figura 5.17 Capturi de ecran – proces vânzare-cumpărare.....	39
Figura 5.18 Firebase Assistant – Conexiunea în Cloud .....	40
Figura 5.19 Captură de ecran - conexiune baza de date Firebase .....	41
Figura 6.1 Captură de ecran pentru pornirea în execuție a Record Espresso Test	44
Figura 6.2 Captură de ecran pentru înregistrarea pașilor de execuție .....	45
Figura 6.3 Captură de ecran pentru adăugarea aserțiunilor .....	45
Figura 6.4 Captură de ecran pentru salvarea înregistrării ca test .....	46
Figura 7.1 Instalarea aplicației din Google Play .....	47
Figura 7.2 Prima pagină din aplicație – Solicitarea accesului la locație.....	48
Figura 7.3 Capturi de ecran cu acțiunile din pagina principală a Oaspetelui .....	49
Figura 7.4 Capturi de ecran pentru autentificarea și înregistrarea în cont .....	50
Figura 7.5 Capturi de ecran - Gestiunea utilizatorilor de către administrator .....	51
Figura 7.6 Capturi de ecran - Gestiunea produselor de către administrator .....	51
Figura 7.7 Capturi de ecran corespunzătoare actorului Client.....	52
Figura 7.8 Capturi de ecran corespunzătoarea actorului Comerciant .....	53

## Anexa 2. Lista Tabelelor

Tabel 2.1 Tabel cu cerințele funcționale ale platformei de implementat.....	6
Tabel 3.1 Avantajele și dezavantajele aplicațiilor mobile vs. web .....	9
Tabel 3.2 Tabel cu avantajele și dezavantajele platformelor similare .....	15
Tabel 4.1 Tabel cu actorii și responsabilitățile lor.....	16
Tabel 4.2 Diferențele dintre limbajele de programare Java și Kotlin .....	20
Tabel 5.1 Nivelurile de Zoom Google Maps [23].....	37

### Anexa 3. Glosar de termeni

Abrevieri	Termeni	Definiții
API	Application Programming Interface	Este un set de definiții de sub-programe, protocoale și unele pentru programarea de aplicații software.
E-commerce	Electronic-Commerce	Vânzarea și cumpărarea de bunuri în mediul online.
CRUD	Create Read Update Delete	Sunt cele patru operații de baza ale stocării permanente a datelor
UI	User Interface	Este spațiul în care omul interacționează cu platforma/aplicația.
IDE	Integrated Development Environment	Este un mediu de dezvoltare compus dintr-un set de programe menite să ajute programatorul în scrierea programelor.
UML	Unified Modeling Language	Limbajul de modelare unificat este un limbaj de modelare de uz general, de dezvoltare, în domeniul ingineriei software care este destinat să ofere un mod standard de vizualizare a proiectării unui sistem.
SDK	Software Development Kit	Este un set de instrumente furnizat pentru a ajuta dezvoltatorii să dezvolte aplicații pe o anumită platformă.
APK	Android Package	Este un pachet utilizat de sisteme de operare bazate pe Android, pentru distribuirea și instalarea diverselor aplicații (jocuri, mobile etc.)
ADT	Android Development Tools	ADT este un plugin pentru Eclipse care oferă o suita de instrumente care sunt integrate cu Eclipse IDE. ADT oferă acces GUI la multe dintre instrumentele SDK din linia de comandă, precum și un instrument de proiectare UI pentru prototipare rapidă, proiectare și construirea interfeței utilizator a aplicației.
AS	Android Studio	Android Studio este mediul oficial de dezvoltare integrată pentru sistemul de

### Anexa 3

		operare Android Google, construit pe software-ul IntelliJ IDEA al JetBrains și conceput special pentru dezvoltarea Android.
XML	Extensible Markup Language	XML este un limbaj de marcare care definește un set de reguli pentru codificarea documentelor într-un format care poate fi citit de om și citit de mașină.
SAMF	Single Activity Multiple Fragments	SAMF este arhitectura care are o singură activitate sau un număr relativ mic de activități. În loc să avem o activitate care să reprezinte un ecran, vedem o activitate ca un container mare cu fragmentele din interiorul activității care reprezintă ecranul.
DSL	Domain-Specific Language	Un limbaj specific domeniului este un limbaj de computer specializat pentru un anumit domeniu de aplicație. Acest lucru este în contrast cu un limbaj de uz general, care se aplică în general pe domenii.
JVM	Java Virtual Machine	O mașină virtuală Java este o mașină virtuală care permite unui computer să ruleze programe Java, precum și programe scrise în alte limbi care sunt, de asemenea, compilate în cod bytecode Java.
DAG	Directed Acyclic Graph	În matematică, în special teoria graficelor și informatică, un grafic aciclic direcționat este un grafic direcționat fără cicluri direcționate. Adică este alcătuit din vârfuri și muchii, cu fiecare margine direcționată de la un vârf la altul, astfel încât urmarea acestor direcții nu va forma niciodată o buclă închisă.
GCP	Google Cloud Platform	Google Cloud Platform, oferit de Google, este o suită de servicii de cloud computing care rulează pe aceeași infrastructură pe care Google o folosește intern pentru produsele sale pentru utilizatorii finali, cum ar fi

### Anexa 3

		Căutarea Google, Gmail, stocarea fișierelor și YouTube.
SHA-1	Secure Hash Algorithm 1	Este o funcție de hash din criptografie, care produce un “rezumat” al mesajului de criptat, reprezentat prin 40 de cifre în hexazecimal.

#### **Anexa 4. Secvențe de cod sursă relevant**

## item\_list\_admin\_products.xml:

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <layout xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:tools="http://schemas.android.com/tools"
4     xmlns:app="http://schemas.android.com/apk/res-auto">
5
6     <data>
7         <variable
8             name="product"
9             type="com.android.madeinro.entities.Product" />
10    </data>
11
12    <androidx.cardview.widget.CardView
13        android:layout_width="match_parent"
14        android:layout_height="wrap_content"
15        app:cardCornerRadius="8dp"
16        app:cardBackgroundColor="@android:color/white"
17        android:layout_margin="10dp"
18        app:cardElevation="8dp">
19
20        <LinearLayout
21            android:layout_width="match_parent"
22            android:layout_height="match_parent"
23            android:padding="5dp"
24            android:gravity="center_vertical"
25            android:orientation="horizontal">
26
27            <ImageView
28                android:id="@+id/productAdminImage"
29                android:layout_width="50dp"
30                android:layout_height="50dp"
31                android:scaleType="fitXY"
32                android:contentDescription="@{product.name}" />
33
34
35
36
37
38
39
40
41
42
43
44
45
```

## Anexa 4

```
32     class ProductsFragment : Fragment(), View.OnClickListener {
33
34         private var _productsBinding: FragmentProductsBinding? = null
35         private val productsBinding get() = _productsBinding!!
36         private val productsAdapter by lazy { ProductsAdminAdapter(DataExample.getProductsList()) }
37
38         private fun setupRecyclerView(){
39             productsBinding.productsAdminRecyclerView.adapter = productsAdapter
40             productsBinding.productsAdminRecyclerView.layoutManager = LinearLayoutManager(requireContext())
41         }
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
```

```
21     class PermissionsFragment : Fragment(), EasyPermissions.PermissionCallbacks {
22
23         private var _fragmentPermissionsBinding: FragmentPermissionsBinding? = null
24         private val fragmentPermissionsBinding get() = _fragmentPermissionsBinding!!
25
26         override fun onCreateView(
27             inflater: LayoutInflater, container: ViewGroup?,
28             savedInstanceState: Bundle?
29         ): View? {...}
30
31
32         override fun onRequestPermissionsResult(
33             requestCode: Int,
34             permissions: Array<out String>,
35             grantResults: IntArray
36         ) {
37             EasyPermissions.onRequestPermissionsResult(requestCode, permissions, grantResults, ...receivers: this)
38         }
39
40
41         override fun onPermissionsDenied(requestCode: Int, perms: List<String>) {
42             if(EasyPermissions.somePermissionPermanentlyDenied( host: this, perms)) {
43                 SettingsDialog.Builder(requireActivity()).build().show()
44             } else {
45                 requestLocationPermission( fragment: this)
46             }
47         }
48
49
50
51
52
53
54
55         override fun onPermissionsGranted(requestCode: Int, perms: List<String>) {
56             findNavController().navigate(R.id.action_permissionsFragment_to_mapsFragment)
57         }
58
59
60
61
62
63
64
65
66
67
68
69
70 }
```

## Anexa 4

### MapsFragment.kt:

```
private fun getAutocompleteResults(){

    val autocompleteUserFragment = childFragmentManager.findFragmentById(R.id.autocompleteUserFragment)
        as AutocompleteSupportFragment

    autocompleteUserFragment.setPlaceFields(listOf(
        Place.Field.ID, Place.Field.NAME,
        Place.Field.LAT_LNG, Place.Field.ADDRESS, Place.Field.ADDRESS_COMPONENTS, Place.Field.PHONE_NUMBER))

    autocompleteUserFragment.setOnPlaceSelectedListener(object : PlaceSelectionListener {
        override fun onPlaceSelected(place: Place) {
            Log.i(ContentValues.TAG, msg: "Place: ${place.name}, ${place.id}")
            map.clear()
            map.addMarker(MarkerOptions().position(place.latLng!!).title(place.name).draggable( draggable: false))
            map.animateCamera(CameraUpdateFactory.newLatLngZoom(place.latLng!!, zoom: 14f))
        }
        override fun onError(status: Status) {
            Log.i(ContentValues.TAG, msg: "An error occurred: $status")
        }
    })
}
```