

**UNIVERSITATEA BABEȘ-BOLYAI CLUJ-NAPOCA**  
**FACULTATEA DE MATEMATICĂ ȘI**  
**INFORMATICĂ**  
**SPECIALIZAREA INFORMATICA ROMÂNĂ**

**LUCRARE DE LICENȚĂ**

**Realizarea unui frigider inteligent și  
ecologic combinând tehnici de IOT,  
logică fuzzy și inteligență artificială**

**Coordonator științific**  
Lect. Dr. MIRCEA Ioan-Gabriel

**Absolvent**  
Petrus Ștefania Rubinia

**2020**

## **Abstract**

The modern digital age has seen tools being developed to improve the quality of life. For centuries before the medieval period, and for centuries afterward, human beings in all parts of the world used a variety of methods to preserve foods for later consumption. Drying was used to preserve all sorts of foods, in our days for preserving the food we used refrigerators.

The motivation of this work is to further develop the concept, and to create an modern refrigerator which is capable to adjust the optimum temperature for our comestible products. For our application we used artificial intelligence in order to detect fruits and vegetables and for package products we scan the barcode in order recognise them.

The optimal temperature is calculated using fuzzy logic. Fuzzy logic is an approach to computing based on "degrees of truth" rather than the usual "true or false" boolean logic on which the modern computer is based.

In order to accomplish this, we used a mobile application, studying the available technologies and resources, and choosing the best possible solutions. For back-end we choose Java language and for front-end we use Kotlin, for an attractive look for the user.

This work is the result of my own activity. I have neither nor received unauthorized assistance on this work.

# Contents

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Introducere</b>   | <b>4</b>  |
| <b>2</b> | <b>Stabilirea clară a problemei și soluțiile existente</b>   | <b>5</b>  |
| 2.1      | Consumul energiei electrice în locuințe . . . . .  | 5         |
| 2.2      | Cauzele consumului ridicat de energie electrică în cazul frigiderelor . . . . .                        | 6         |
| 2.2.1    | Consumul influențat de temperatura camerei . . . . .   | 6         |
| 2.2.2    | Consumul influențat de setarea termostatului . . . . .   | 7         |
| 2.2.3    | Consumul influențat de deschiderea ușilor . . . . .  | 8         |
| 2.2.4    | Consumul influențat de depozitarea alimentelor calde . . . . .   | 9         |
| 2.3      | Soluții existente în ceea ce privește consumul ridicat de energie în cazul electrocasnicelor . . . . . | 10        |
| 2.3.1    | Etichetarea claselor de energie . . . . .  | 10        |
| 2.3.2    | Frigiderele inteligente . . . . .  | 11        |
| 2.4      | Analiza soluțiilor existente în ceea ce privește frigiderele inteligente . . . . .                     | 11        |
| 2.4.1    | Identificarea produselor stocate în frigider . . . . .   | 13        |
| 2.5      | Depistarea problemelor întâlnite în ceea ce privește deținerea unui frigider inteligent . . . . .      | 15        |
| <b>3</b> | <b>Propunerea unei soluții</b>   | <b>17</b> |
| 3.1      | Ideea soluției propuse . . . . .   | 17        |
| 3.2      | FlowChart-ul aplicației . . . . .  | 18        |
| 3.3      | Implementarea ideii într-un API . . . . .  | 21        |
| 3.3.1    | Adăugarea și ștergerea fructelor și a legumelor . . . . .  | 21        |
| 3.3.2    | Adăugarea și ștergerea alimentelor ambalate și a celor preparate . . . . .                             | 23        |
| 3.3.3    | Determinarea temperaturii optime . . . . .   | 23        |
| 3.4      | Specificații clare și testare endpointuri API . . . . .  | 24        |
| 3.5      | Detalii de implementare API . . . . .  | 26        |
| 3.5.1    | Identificarea fructelor și a legumelor . . . . .   | 27        |
| 3.5.2    | Determinarea temperaturii optime folosind logica fuzzy . . . . .                                       | 30        |
| <b>4</b> | <b>Dezvoltarea aplicației mobile și arhitectura</b>  | <b>32</b> |
| 4.1      | Motivație . . . . .  | 32        |
| 4.2      | Arhitectura aplicației . . . . .   | 33        |
| 4.2.1    | Comportament . . . . .   | 33        |
| 4.2.2    | Structura . . . . .  | 34        |
| 4.2.3    | Arhitectura MVVM . . . . .   | 35        |
| 4.2.4    | Modul în care se leagă structura și comportamentul . . . . .   | 37        |

|          |  |           |
|----------|--|-----------|
| 4.2.5    | Ștergere produs . . . . .                                    | 37        |
| 4.2.6    | Adăugarea manuală a produsului . . . . .                     | 38        |
| 4.2.7    | Editare produs . . . . .                                     | 39        |
| 4.2.8    | Vizualizarea datelor de valabilitate . . . . .               | 40        |
| 4.2.9    | Vizualizarea temperaturii actuale și a celei optime . . . .  | 41        |
| 4.2.10   | Scanarea unui produs care deține cod de bare . . . . .       | 42        |
| 4.2.11   | Detectarea produselor din categoriile "Fructe" si "Legume" . | 43        |
| <b>5</b> | <b>Modalitatea de implementare</b>                           | <b>44</b> |
| 5.1      | Caracterul abstract al arhitecturii . . . . .                | 44        |
| 5.2      | Implementarea arhitecturii abstracte . . . . .               | 45        |
| 5.3      | Justificarea limbajului și a tehnologiei folosite . . . . .  | 48        |
| 5.3.1    | Front-end . . . . .  | 48        |
| 5.3.2    | Back-end . . . . .   | 49        |
| 5.3.3    | Baza de date . . . . .                                       | 50        |
| 5.4      | Specificarea si testarea aplicatiei . . . . .                | 51        |
| 5.5      | Descrierea formei de persistenta . . . . .                   | 51        |
| 5.6      | UX si manual de utilizare al aplicatiei . . . . .            | 52        |
| <b>6</b> | <b>Concluzii si dezvoltări ulterioare</b>                    | <b>55</b> |
| <b>7</b> | <b>Bibliografie</b>  | <b>56</b> |

# 1 Introducere

Numărul frigiderelor prezente în locuințe este în continuă creștere, reprezentând o necesitate, însă acestea cresc vizibil consumul de energie electrică, astfel ajungându-se ca în ultimii ani acest domeniu să fie din ce în ce mai cercetat. Pornindu-se de la electrocasnice cu un consum net superior față de cele actuale, consumul s-a îmbunătățit datorită multiplelor eforturi manifestate prin studii făcute asupra acestui aspect, respectiv aducerea unor soluții funcționale. Luând în vizor această temă, în capitolul 2 a acestei lucrări se aduc la cunoștință diferite modalități prin care alți cercetători au reușit să relateze pericolul consumului ridicat de energie și rata de creștere a acestuia din pricina emancipării populației. Secțiunile 2.1 și 2.2 scot în evidență și detaliază motivația abordării acestei teme. În secțiunea 2.3 se prezintă soluțiile aduse din partea cercetătorilor în privința problematicii consumului ridicat, una dintre soluții fiind crearea și folosirea frigiderelor inteligente. Mergând pe această idee, în secțiunea 2.4 se aprofundează această soluție, ilustrând diferite modalități prin care oamenii de știință au reușit să creeze un astfel de dispozitiv. În secțiunea 2.5 se aduc în prim-plan problemele des întâlnite care împiedică crearea unui electrocasnic inteligent precum costurile ridicate și accesibilitatea la un astfel de dispozitiv electric, urmând ca în capitolul 3 să se propună o soluție practică ca rezolvare la problemele cel mai des întâlnite, soluția constând în realizarea unei aplicații mobile capabile să înregistreze prin diferite metode produsele ce urmează să fie stocate în frigider, iar pe baza acestora să se determine o temperatură optimă de care este nevoie pentru ca produsele să se pastreze comestibile. În indeplinirea acestei soluții apar ca elemente de noutate o modalitate proprie de detectare a fructelor/legumelor folosind **inteligenta artificială**, o modalitate de abordare nouă a **logicii fuzzy** care să contribuie la reducerea consumului de energie și un **Raspberry pi** care să gestioneze temperatura frigiderului.

Începând cu capitolul 4 se motivează necesitatea creării unei aplicații mobile care să contribuie eficient la scopul acestei teme. Arhitectura aplicației este ilustrată prin diagrama cazurilor de utilizare, prin diagrama de componente și prin cea de secvență. Capitolul 5 descrie caracterul abstract al arhitecturii și implementarea acestuia, detaliind clasele și legăturile dintre acestea prin diagramele de clase. În ceea ce privește limbajul și tehnologiile folosite, sunt descrise în cadrul acestui capitol la secțiunea 5.3. Se realizează și o testare a aplicației propuse, iar metoda este descrisă în secțiunea 5.4. Diagrama bazei de date se află în secțiunea 5.5. Manualul de utilizare al aplicației mobile este pus în valoare în cadrul ultimei secțiuni a capitolului 5 urmând ca apoi să se pună accent asupra performanței API-ului propus. În final se vor prezenta modalitățile de dezvoltare a aplicației propuse și a acestei abordări.

## 2 Stabilirea clară a problemei și soluțiile existente

### 2.1 Consumul energiei electrice în locuințe

În ultimele două decenii economia s-a dezvoltat la nivel global. În China, creșterea accelerată a economiei a ajutat la instalarea mai rapidă a urbanizării, acest factor introducând de altfel și consumul mai ridicat al produselor care oferă un trai mai bun în viața de zi cu zi. Creșterea achiziționării de electrocasnice s-a făcut marcată, astfel că în zona urbană, în anul 2008 s-a estimat că aproximativ 93 de familii din 100 dețineau frigidere. [16] Un an mai târziu, în 2009, s-a realizat un alt studiu, vizând de această dată Malaysia. În acest caz se estimează că 26% din consumul destinat locuințelor, este consumat doar pentru electrocasnice, în această categorie încadrându-se frigiderele, congelatoarele, mașinile de spălat vase, cuptoarele electrice și mașinile de spălat rufe.[3] Potrivit unui studiu care se bazează pe tendințele de piață ale Suediei, se așteaptă ca din anul 2015 până în 2035, consumul energiei electrice folosit în punerea în funcțiune a electrocasnicelor să scadă cu un procentaj de 8%, acest procentaj bazându-se pe rata de înlocuire a electrocasnicelor vechi cu cele noi, care au un consum mai redus de energie.[18] Deocamdată, comparativ cu alte țări din Europa de Est, România se numără printre țările cu cea mai scăzută rată de înlocuire a produselor electrocasnice.

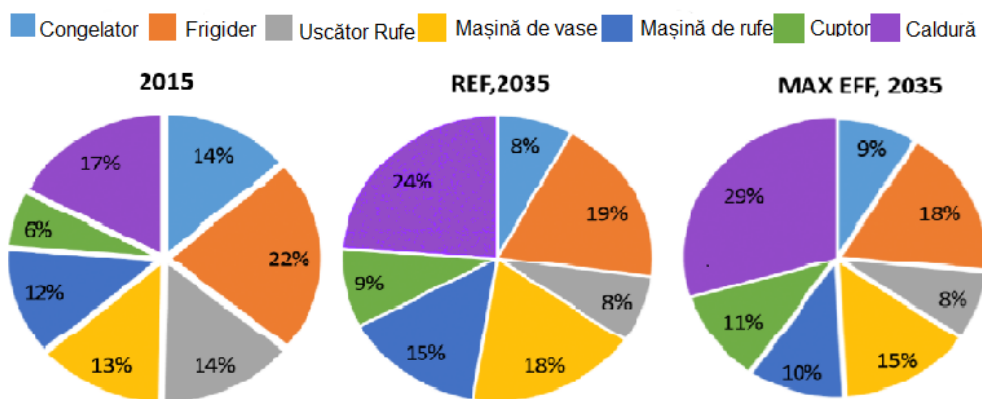


Figure 1: Estimarea consumului de energie 2015-2023 [15]

În anul 2013, studiul [2], publicat în Germania, afirmă faptul că 13.4% din electricitatea atribuită sectorului rezidențial este consumată de către aparatele frigorifice.

## **2.2 Cauzele consumului ridicat de energie electrică în cazul frigiderelor**

Petru a putea controla într-un mod mai eficient consumul exagerat de energie electrică, trebuie mai întâi să înțelegem cauzele principale care produc acest consum nedorit. Deși apariția claselor energetice au estompat semnificativ consumul, mai există alți factori care au rămas neidentificați pentru majoritatea consumatorilor. În [15] se pune accentul pe acești factori și se disting astfel 3 mari cauze, iar în [2] se adaugă un al 4-lea factor și anume influența pe care o are depozitarea alimentelor calde. Tot în cadrul acestui articol se subliniază consumul care variază de la câțiva wati-ora la 200 wati-ora(wh), acest consum distorsionat fiind influențat de modul de utilizare al frigorificelor și de atenția utilizatorilor asupra următorilor factori.

### **2.2.1 Consumul influențat de temperatura camerei**

În [15] se relatează faptul că cea mai mare parte a puterii termice produsă de frigider este condusă prin pereții acestuia, adică aproximativ 53 Wh pentru fiecare grad în plus din mediul exterior. Această putere termică produsă este proporțională cu diferența dintre temperatura ambiantă și temperatura din interior. Cu cât este mai mare diferența aceasta, cu atât este mai mare sarcina impusă frigiderului. Din acest motiv, temperatura aerului din jurul unui frigider este un factor semnificativ al consumului de energie. Putem confirma faptul că pe măsură ce temperatura încăperei crește, crește de asemenea și consumul de electricitate.

În studiul [2], realizat în 2013, se afirmă faptul că căldura ambiantă este cel mai mare factor care influențează consumul de energie. Realizându-se un experiment în acest sens, s-a descoperit o creștere a consumului cu 6-7% pentru fiecare grad în plus în mediul ambiant, iar când temperatura atinge 25 grade Celsius, procentajul consumului alternează între 4% și un maxim de 12%.

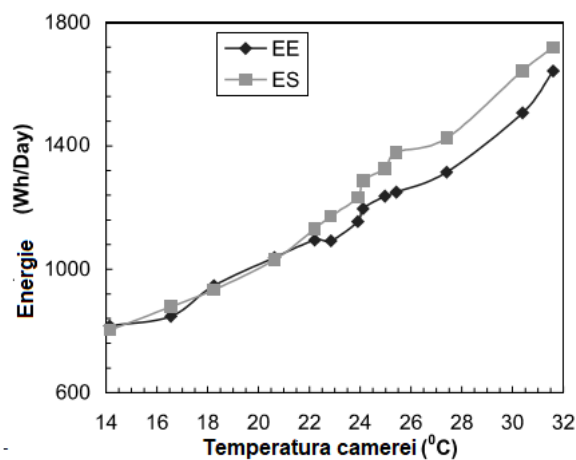


Figure 2: Consumul a două aparate frigorifice din punct de vedere al temperaturii camerei [15]

### 2.2.2 Consumul influențat de setarea termostatului

În ipostaza în care termostatul este setat pe o valoare mai mare, se poate confirma faptul că motorul frigiderului va trebui să facă un efort mai redus pentru ca temperatura din interior să fie cea dorită. Privind această imagine din perspectiva opusă, când termostatul este setat pe o valoare mică, datele analizate în studiul [15] arată o creștere de consum de aproximativ 7,8% pentru fiecare grad.

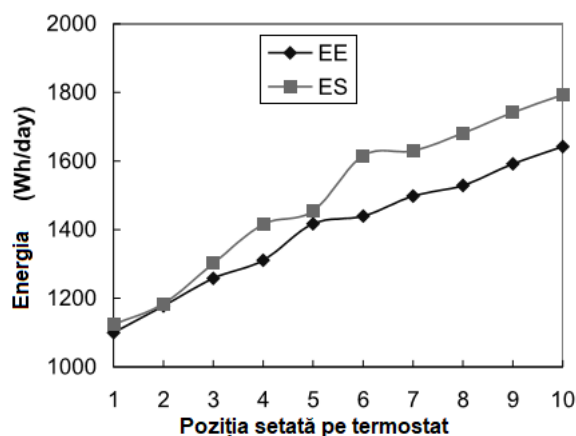


Figure 3: Consumul a doua aparate frigorifice din punct de vedere al temperaturii camerei [15]

### 2.2.3 Consumul influențat de deschiderea ușilor

În cazul deschiderii ușilor, o mare parte din căldura și umiditatea din exterior pătrunde în interiorul frigierului, creându-se un schimb de gaze care are ca efect distorsionarea temperaturii setate de către utilizator.

În experimentul făcut pe un frigider de 300L, în articolul [15], se observă o creștere de consum de 12,4 Wh pentru fiecare ușă deschisă timp de 12 secunde. Deschiderea ușilor are ca efect negativ și deteriorarea alimentelor, grăbind procesul de alterare.

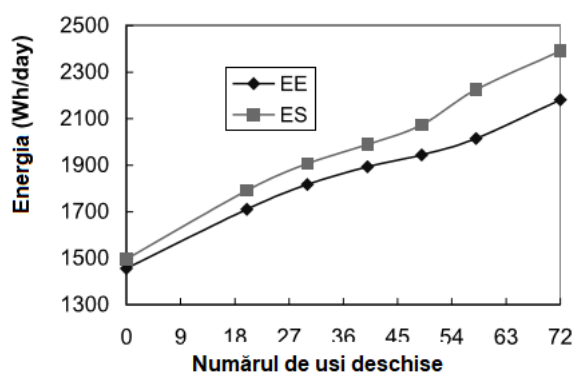


Figure 4: Consumul a două aparate frigorifice din punct de vedere al numărului de deschideri al ușilor [15]

Conform studiului [1] realizat în 1987, se disting 4 tipuri de transferuri ce se realizează în timpul deschiderii ușilor și care :

1. Transferul de căldură convectivă care constă în trecerea aerului ambiant pe suprafețele din interiorul frigiderului
2. Transferul de căldură latentă ce constă în condensarea vaporilor de apă din aerul umed, urmând ca acești vapori să se prelingă pe suprafețele frigorifice
3. Transferul de căldură radiativă
4. Transferul de căldură sensibilă din masa de aer cald în interiorul frigorificelor după închiderea ușilor.

Pentru a evidenția faptul că această problemă nu s-a rezolvat pe parcursul anilor deși s-au creat modele noi, studiul [7], apărut în anul 2013, testează de asemenea efectul pe care îl are deschiderea ușii în contextul în care temperatura și umiditatea mediului exterior rămâne constant, iar temperatura

din interior se pastrează în intervalul 3-5 grade Celsius. Testarea se face pe o durată de 6 ore, iar numărul ușilor alternează între 12 și 48. În acest context, cantitatea de energie electrică are o creștere de 6.67% până la 30%.

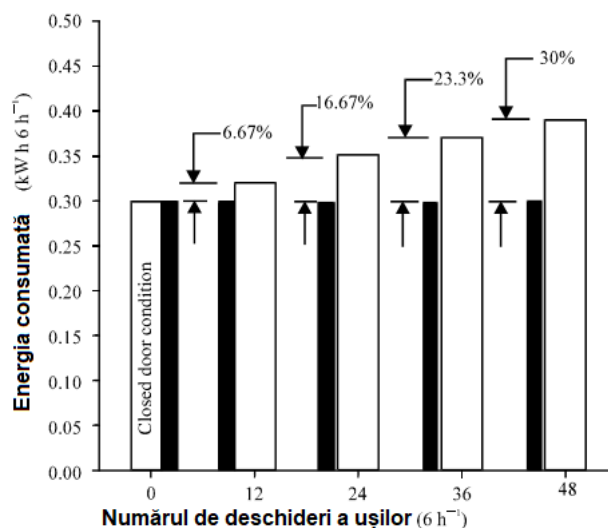


Figure 5: Creșterea procentajului [7]

#### 2.2.4 Consumul influențat de depozitarea alimentelor calde

Introducerea unui aliment cu o temperatură relativ mare față de cea din interiorul frigiderului, are ca efect, cuplarea termostatului astfel încât frigiderul va fi adus într-o stare de funcțiune până în momentul în care mâncarea va atinge temperatura cerută de termostat. Conform studiului [2] acest proces de restabilire va ridica consumul de curent electric și poate afecta starea de funcționare al electrocasnicului dacă alimentele sunt în cantități mari, iar deseori această cauză duce la defectarea frigiderului. Această problemă este mai des întâlnită în cazul congelatoarelor deoarece deseori se întâmplă să se introducă în rafturile acestuia cantități mari de carne care are o temperatură mai ridicată, iar ca urmare a acestei acțiuni, congelatorul va funcționa fără întrerupere pe o perioadă de timp mai lungă pentru a se asigura că produsul din interior are temperatura dorită. Dacă după un timp, nu se înregistrează vreo întrerupere a funcționării congelatorului, acesta emite un semnal de alarmă prin intermediul unui bec de culoare roșie.

## 2.3 Soluții existente în ceea ce privește consumul ridicat de energie în cazul electrocasnicelor

În ultimii 30 de ani, Comisia Europeană a emis o serie de amendamente în domeniul energiei, ca reacție la factori din ce în ce mai îngrijorători, cauzăți de impactul emisiilor de gaze cu efect de seră și a consumului major de energie electrică. Soluțiile care au avut un impact pozitiv și care s-au stabilit pentru a reduce masiv consumul energiei și implicit a emisiilor de carbon, constau în următorii factori.

### 2.3.1 Etichetarea claselor de energie

Acest mijloc de economisire s-a propus pentru ca consumatorii să poată opta pentru un consum după propriile prejudecăți. Eticheta energetică, ce conține diferite litere, culori și săgeți descriptiv, indică eficiența energetică pe care o are un electrocasnic. În Europa, apariția etichetării claselor energetice superioare A și B, de la începutul anilor 90, a avut ca efect reducerea energiei electrice, atribuite locuințelor, cu aproximativ 10%. În 1999 acest procentaj a crescut considerabil și s-a ajuns până la 57%. Conform [4], clasele existente până în anul 2003 au fost etichetate folosind alfabetul englez: A, B, C, D, E, F, G, iar apoi au apărut și clasele A+, A++, respectiv A+++ care au fost superioare față de celelalte. Un frigider aflat în clasa A+ consumând cu 80% mai puțină energie decât un frigider mai vechi, unul din clasa A++ cu 25% mai puțin decât un frigider din clasa A, iar unul din clasa A+++ cu 60% mai puțin decât cel din clasa A. Modalitatea de etichetare este bazată pe indicii de eficiență energetică, care este o valoare constantă, iar clasa din care face parte un anumit electrocasnic este determinată de valoarea acestui indice. O mai bună înțelegere în ceea ce privește acest aspect se poate realiza vizualizând următorul tabel.

| Indice de eficiență energetică: "I" | Clasa de eficiență energetică |
|-------------------------------------|-------------------------------|
| $I < 30$                            | A++                           |
| $30 < I < 42$                       | A+                            |
| $42 < I < 55$                       | A                             |
| $55 < I < 75$                       | B                             |
| $75 < I < 90$                       | C                             |
| $90 < I < 100$                      | D                             |
| $100 < I < 110$                     | E                             |
| $110 < I < 125$                     | F                             |
| $125 < I$                           | G                             |

Figure 6: Creșterea procentajului [7]

Achiziționarea unui electrocasnic nou este un factor cheie în protejarea mediului înconjurător pe lângă economiile pe termen lung pe care le face un consumator. Amprenta de carbon este un subiect discutat amănunțit pe parcursul ultimilor ani deoarece impactul stocurilor de carbon asupra mediului înconjurător are ca repercursiune încălzirea globală, ecosistemul neputând să asigure un echilibru. Deși plantele folosesc dioxidul de carbon în realizarea fotosintezei, amprenta de carbon a fiecărei persoane este mult prea mare comparativ cu cantitatea folosită de vegetația existentă. Fluctuațiile ce au loc în cadrul stocurilor de carbon înmagazinate în atmosferă sunt îngrijorătoare și se află într-o creștere exponențială. Un alt factor care încurajează achiziționarea unui electrocasnic nou, este adus de prezența CFC-urilor (cloroflorocarburilor) în aparatele vechi, adică a freonului. Acest gaz fiind o descoperire inofensivă pentru oameni, dar dăunătoare pentru stratul de ozon deoarece moleculele de freon ajunse în stratosferă sunt descompuse de razele UV astfel eliberând clor, fluor și brom care descompun și distrug moleculele de ozon. Deși freonul folosit în instalațiile aparatelor frigorifice nu se epuizează, pot exista totuși fisuri în instalații, care provoacă eliberarea acestui gaz.

### **2.3.2 Frigiderele inteligente**

Frigiderele inteligente au fost introduse pe piață datorită extinderii ariei tehnologice cu scopul de a aduce noi beneficii consumatorilor. Începând cu anii 2000, ideea de a conecta aparatele electrice la internet a devenit din ce în ce mai populară, astfel încât marile firme producătoare de electrocasnice precum Siemens, Samsung etc. au inclus pe piață diferite modele. Inteligența acestor aparate constând în puterea de comunicare cu utilizatorii prin intermediul unor aplicații posibil de accesat atât de pe telefon, cât și de pe o anumită tabletă incorporată pe frigider. Unul dintre cele mai importante beneficii aduse odată cu apariția acestor aparate constă în estomparea consumului de electricitate provocat atât de deschiderea repetată a ușilor, cât și de setarea termostatului. Un alt avantaj marcat de acestea este monitorizarea constantă a produselor aflate în interior astfel încât consumatorul să aibă obținerea de a urmări prin diverse metode alimentele respective.

## **2.4 Analiza soluțiilor existente în ceea ce privește frigiderele inteligente**

Modalitățile de proiectare a electrocasnicelor de acest tip sunt multiple. De-a lungul anilor s-au făcut mari progrese în acest domeniu. În articolul [9], realizat în anul 2002, se prezintă invenția unui aparat care are capacitatea

de a stoca pe un server alimentele aflate în interior prin intermediul unor mesaje vocale, prin introducerea manuală a detaliilor legate de aliment sau prin identificarea codurilor de tip IC. O altă abilitate prezentă este dată de posibilitatea de a vedea imagini din interior prin intermediul unei camere amplasate în interior. În articolul [13], realizat în anul 2011, se prezintă un frigider inteligent care conține un sistem de aranjare a produselor și cel puțin un container care permite stocarea și monitorizarea alimentelor. Invenția urmărește ca utilizatorul să fie cât mai bine informat în ceea ce privește timpul de valabilitate a produselor stocate în containere, acest lucru fiind posibil prin intermediul etichetelor electrice amplasate pe fiecare container. Durata de viață putând fi setată atât manual, dar și prin intermediul unui scanner sau a unui detector RFID (radio frequency identification). Detectorul putând obține date consistente ca de exemplu: tipul de aliment, data de expirare sau temperatura la care să se păstreze produsul. Pentru a fi posibilă notificarea consumatorilor în cazul în care vreun aliment se apropie de data expirării, în exteriorul electrocasnicului este amplasată o unitate de control conectată la internet, capabilă de efectuarea acestui task. O abordare asemănătoare este sesizată în articolul [11]. Punctele comune sunt date de modalitatea de depozitare a produselor, fiecare gamă de produse având un anumit compartiment, și de modul de detectare a acestora, însă detaliile legate de produsele stocate sunt afișate în exterior prin intermediul unui monitor și nu pe etichetele electronice din interior. O astfel de abordare este benefică deoarece se păstrează produsele comestibile într-un mod extrem de organizat, însă spațiul ocupat de către acest model este unul relativ mare față de ceea ce se găsește deja în gospodărie, de altfel: costurile sunt mari, iar aplicarea metodei este costisitoare. În articolul [8] apare o nouă modalitate de a identifica produsele achiziționate și anume prin intermediul unei chitanțe electronice emise de către vânzător printr-un mesaj sau printr-o aplicație dedicată strict chitanțelor de acest tip. Pe lângă afișarea produselor de către un device atașat de electrocasnic, apare și modalitatea de a afișa pe telefon lista alimentelor cu detalii mai ample. Unitatea de control folosită are capacitatea de a filtra datele de pe chitanța electronică și de a căuta în mediul online detalii precum categoria produsului, camera în care trebuie depozitat produsul și perioada maximă de stocare. Identificarea produselor cu ajutorul unei camere montate în interior este posibilă de realizat mulțumită inteligenței artificiale. Folosirea acestei ramuri a informaticii face ca detectarea alimentelor să devină o arie vastă de investigat. Astfel că apar posibilități precum: capturarea unor imagini la un anumit interval de timp [5] sau realizarea unui video la fiecare deschidere a frigiderului, iar apoi efectuarea unor capturi de ecran pe baza videoului [10]. După aplicarea pasului anterior, se urmăresc o serie de pași în realizarea detectării

precum: segmentarea pe contururi, eliminarea părților blurate, segmentarea pe obiecte, identificare tip aliment și analiza pe baza culorilor pentru a se afla calitatea [5]. Tehnologia de recunoaștere și clasificare a vegetalelor se bazează pe forma, culoarea, mărimea și textura acestora. Iar pentru a nu pune în calcul efectele date de nivelul de luminare a imaginilor, imaginile sunt convertite în imagini de tipul HSV în majoritatea studiilor efectuate. De asemenea procesul de analizare a imaginilor se face folosind Matlab [5], sau biblioteci din python în majoritatea cazurilor.

#### **2.4.1 Identificarea produselor stocate în frigider**

Un factor cheie în ceea ce privește realizarea unui frigider inteligent este dat de capacitatea de identificare a produselor stocate și mai cu seamă a produselor care nu dețin un cod de bare, astfel că acest subcapitol relatează modul în care alți cercetători detectează fructele și legumele în diferite domenii folosind inteligența artificială. Am ales să examinez cu atenție o arie mai vastă a acestei probleme deoarece identificarea fructelor este un domeniu des cercetat în domeniul recoltei.

Inteligența artificială a făcut mari progrese pe parcursul anilor. Pornind de la aplicarea IA ului în simple jocuri de șah sau de damă, iar mai apoi asupra programelor capabile de înțelegere a limbajului natural( ex ELIZA) urmând ca mai apoi această domeniu să fie aplicat în realizarea sistemelor expert evolute. Marele avantaj expus de IA este dat de posibilitatea de îmbunătățire a sistemului care efectuează o anumită activitate folosind această tehnică, prin colectarea mai multor informații noi. Inteligența artificială este folosită în diferite domenii precum medicina, marketing, sport, shopping virtual( oferirea recomandărilor personalizate) sau agricultura. În cadrul domeniului dat de agricultură, folosirea inteligenței artificiale ajută la detectarea sau numărarea fructelor din livezi. Deoarece ne interesează în mod special modul de detectare a fructelor și a legumelor stocate în frigider, în următoarele articole sunt prezentate diferite modalități de implementare a unei rețele neuronale capabile să îndeplinească această atribuție. Articolele capturează de altfel și modul în care este creat setul de date sau tipul de arhitectură folosită. În articolul [12] o rețea neuronală convoluțională (CNN) este realizată pentru a număra fructele dintr-o recoltă, cu scopul de a ajuta crescătorii în luarea unor decizii mai bune în ceea ce privește cultivarea și îngrijirea fructelor cât și pentru a estima calitatea recoltei din producția respectivă. Punctul forte care aduce o importanță majoră acestei lucrări este dat de modalitatea de antrenare al rețelei. Setul de date conceput pentru acest antrenament folosește imagini sintetice, iar testarea rețelei folosește date reale. Acest aspect face ca antrenarea algoritmului să fie unul mai puțin costisitor în ceea

ce privește colectarea datelor.

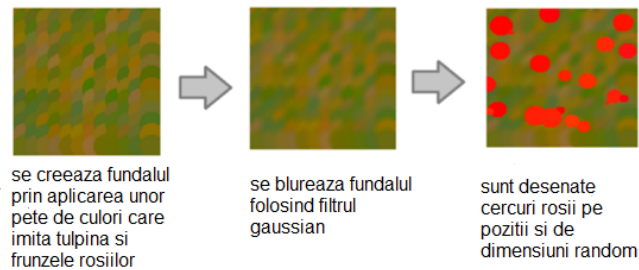


Figure 7: Crearea imaginilor sintetice [12]

Avantajul acestei metode este dat de viteza de numerotare a obiectelor și de ușurința de confecționare a imaginilor sintetice. Deși antrenamentul folosește date sintetice, acuratețea este de 91% asupra datelor reale.

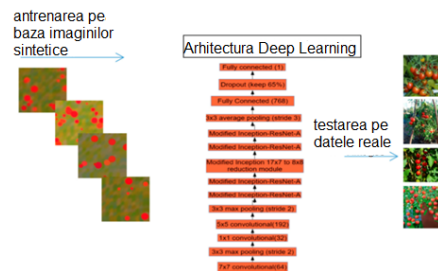


Figure 8: Modalitatea de antrenare. [12]

În articolul [14] se prezintă mai multe modalități de construire a unei rețele neuronare capabile de a detecta fructe din imagini. Modalitățile prezentate se deosebesc prin tipul de imagini folosite( imagini RGB, imagini cu infraroșu) și prin aplicarea tipurilor de fuziune(early fusion”, late fusion”) , iar partea comună este dată de folosirea rețelei de tip Faster- RCNN. Folosind aceste tipuri de abordări asupra detectării fructelor, se evidențiază îmbunătățirea adusă de către imaginile de tip NIR(cu infraroșu), detectarea obiectelor asupra imaginilor întunecate fiind mult mai ușor de realizat decât atunci când se folosesc doar imaginile de tip RGB.

Abordarea care implică "early fusion" concatenează 4 canale și anume 1 pentru imaginile NIR și 3 pentru imaginile RGB, în timp ce abordarea late fusion implică folosirea a două rețele de tip RCNN, o rețea folosește imagini de tip RGB, iar cealaltă rețea folosește imagini de tip NIR.

| RGB   | NIR   | Early Fusion | Late Fusion |
|-------|-------|--------------|-------------|
| 0.816 | 0.797 | 0.799        | 0.838       |

Figure 9: Diferențele de scor. [14]

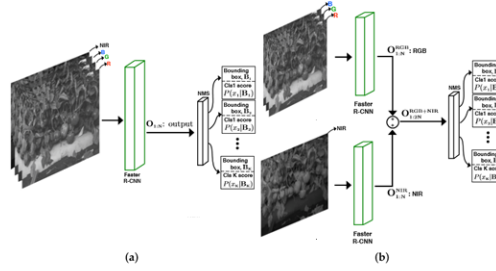


Figure 10: Diferențele de scor pentru. [14]

Cele 4 modalități prezentate folosesc o rețea de tip Faster RCNN preantrenată pe un set mare de date ImageNet”. Folosirea unei rețele de acest tip face posibil ca setul de date folosit pentru antrenare sa fie unul mai mic. În ceea ce privește recunoașterea alimentelor preparate în bucătărie, există o abordare asemănătoare ca cea folosită în detectarea fructelor, descrisă în articolul X și care folosește o rețea neuronală convoluțională(CNN) pentru a identifica diferite tipuri de mâncare, însă întocmirea unui set de date pentru această sarcină necesită eforturi foarte mari, de aceea intervenția detectării mâncării ce constă într-o clasificare binară care decide dacă este sau nu este mâncare este o opțiune mult mai eficientă.

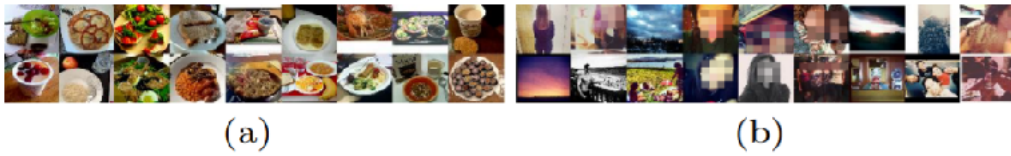


Figure 11: Exemple de imagini. (a)cu mâncare (b) fara mâncare. [6]

## 2.5 Depistarea problemelor întâlnite în ceea ce privește deținerea unui frigider inteligent

Există deja multe patente de frigidere care țin cont de ce alimente sunt stocate în ele în timp real însă achiziționarea unui frigider inteligent sau chiar

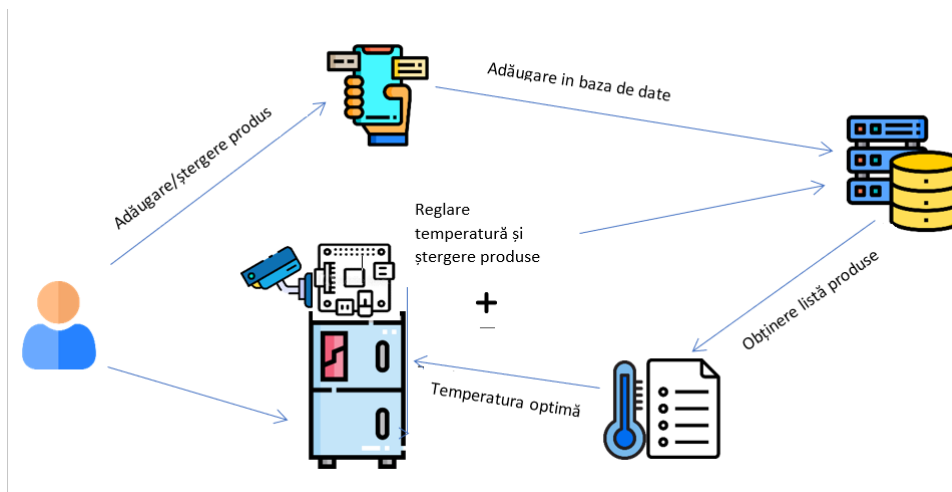
transformarea unui frigider obișnuit în unul inteligent implică costuri foarte mari, iar în ceea ce privește detectarea alimentelor sau a caracteristicilor acestora, apar anumite impedimente.

Pentru detectarea produselor folosind camera din interior, există riscul ca anumite produse să nu se poată identifica din cauza faptului că sunt acoperite de altele. Această problema apare și în cazul în care se dorește identificarea datei de valabilitate necesară în cazul în care se dorește atenționarea utilizatorului dacă un anumit produs se apropie de expirare. Mai există și problema detectării obiectelor când ușile sunt închise, moment în care iluminarea frigiderului este scoasă din funcțiune iar cea mai simplă opțiune ar fi detecția prin intermediul unei camere cu infraroșu. O altă alternativă în acest caz fiind identificarea unei modalități de controlare a iluminării frigiderului. Pe lângă problematica detectării produselor există de asemenea și diferite monitoare atașate pe frigider care să transmită informații precum data de expirare a unui produs pe telefoanele mobile, însă de această dată apare ca problemă faptul că monitorul atașat pe ușa frigiderului este ușor de accesat de membrii familiei și inclusiv de către copii.

Deoarece până în momentul actual s-a pus baza pe modalitatea de organizare a produselor depozitate în frigider, o nouă abordare ar fi crearea unui frigider inteligent axat în principal pe ecologie și de aceea până în momentul actual s-a pus accentul în mod deosebit pe modalitățile prin care un frigider consumă în exces energie electrică prin diferite modalități și mai ales prin deschiderea ușii[15]. Factorii prezentați au avut o însemnătate aparte din cauza faptului că motorul este obligat să funcționeze pentru ca temperatura din interior să se păstreze constantă într-un anumit interval. După studiul amănunțit al bibliografiei din acest domeniu, nu am descoperit soluții existente care să urmărească în mod deosebit această cauză pe care noi o considerăm absolut necesară în contextul încălzirii globale accentuate în ultimul secol. De aceea considerăm oportună dezvoltarea unui frigider inteligent care să urmărească pe cât posibil, în mod automatizat, situația produselor existente la un moment dat în frigider (modelând realist modelarea și scoaterea unui produs din frigider) pentru a putea folosi lista de produse în determinarea temperaturii optime necesare păstrării corespunzătoare a tuturor produselor din frigider.

## 3 Propunerea unei soluții

### 3.1 Ideea soluției propuse



Scopul nostru este de a crea un frigider inteligent și ecologic capabil să calculeze temperatura optimă pe care o necesită produsele stocate astfel încât acestea să rămână într-o stare comestibilă pe o durată de timp cât mai lungă. După obținerea temperaturii optime, o următoare sarcină ar fi modificarea automată a temperaturii din interiorul frigiderului. Datele, pe baza cărora se execută acțiunile precedente sunt stocate prin intermediul unei aplicații mobile care identifică unele produse prin detecție pe baza imaginilor, iar altele prin introducere manuală. Înregistrarea tuturor alimentelor are scopul și de a reda utilizatorului lista de alimente cât și alte informații necesare despre acestea.

În ceea ce privește modalitatea de stocare a alimentelor, am ales intervenția telefonului mobil deoarece acesta aduce beneficii precum accesibilitatea ridicată în ceea ce privește posesia unui astfel de dispozitiv cu acces la internet, limitarea costurilor care fac posibilă transformarea unui frigider obișnuit în unul inteligent, ușurința de a transforma orice frigider obișnuit în unul inteligent și independența față de modul de stocare al alimentelor. Detectarea alimentelor și informarea utilizatorului în ceea ce privește conținutul frigiderului are scopul de a micșora energia consumată ce este cauzată de către deschiderea repetată a ușilor, acest factor fiind demonstrat în secțiunile precedente. Un alt scop important este de a face utilizatorul conștient de durata de viață a alimentelor dar și de tipurile de alimente în cazul în care acesta dorește să prepare un anume fel de mâncare și nu se află acasă sau nu știe ce îi lipsește pentru întocmirea rețetei. În momentul eliminării un produs

din frigider, Raspberry pi-ul atașat de frigider are sarcina de a înregistra această acțiune cu ajutorul unei camere și de a elimina automat din baza de date o anumita categorie de produse. Scopul final este dat de posibilitatea ca frigiderul să-si regleze singur temperatura pe baza alimentelor pe care le detine prin intermediul Raspberry PI-ului atasat care are rolul si de a obține temperatura actuală din interiorul frigiderului cu ajutorul unor senzori speciali. Implementarea acestei aplicații necesită identificarea tuturor produselor și determinarea temperaturii finale pe care acestea o necesită. Inteligența frigiderului constă în cuplarea la internet și totodată la un sistem de gestiune a datelor.

### 3.2 FlowChart-ul aplicației

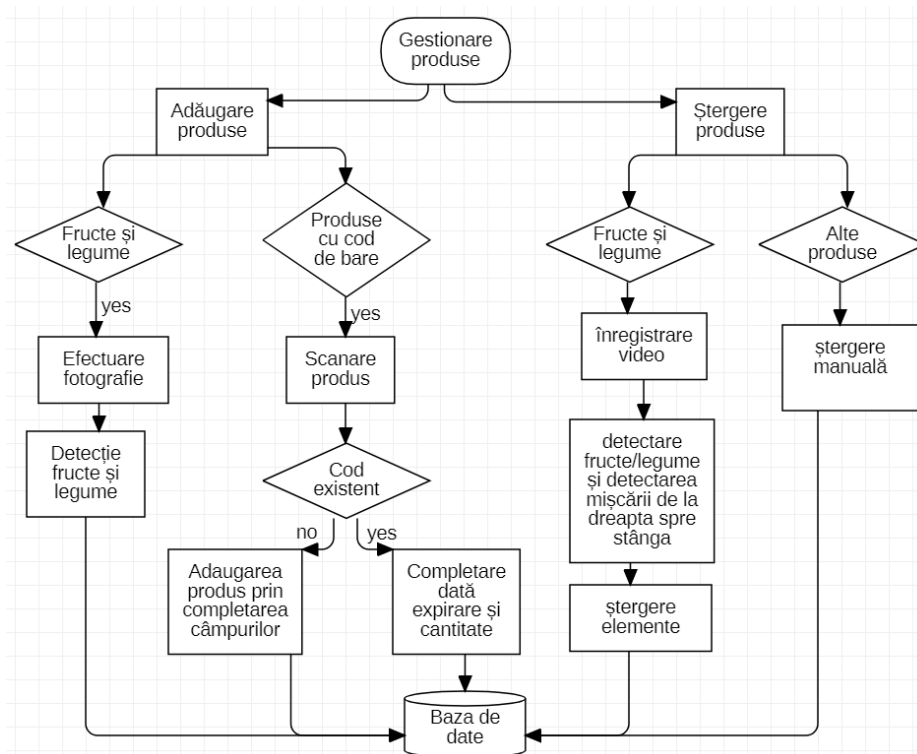


Figure 12: FlowChart-ul care ilustrează factorii esențiali ai aplicației

În procesul de determinare a temperaturii optime este nevoie de reținerea tuturor produselor puse în frigider precum și a unor informații de bază care să permită obținerea temperaturii dorite. Focusandu-ne pe această sarcină, flowchart-ul prezentat în figura alăturată ilustrează modalitate de gestionare a

produselor. În categoria gestionarii, se încadrează modalitățile de adăugare a unui aliment precum și modalitățile de eliminare din mediul de stocare virtuală. Pentru ca un produs să se adauge în baza de date, unde va fi stocat, există 3 posibilități. O variantă este adăugarea prin completarea anumitor câmpuri care cer informații exacte despre produs, a 2-a variantă este dată de posibilitatea de adăugare prin intermediul codului de bare. După deschiderea cititorului de bare care efectuează identificarea codului există două posibilități. O posibilitate este ca codul citit să nu fie înregistrat, prin urmare se necesită completarea a mai multor câmpuri pentru înregistrarea codului nou, o altă posibilă variantă este dată de adăugarea unui produs care este înregistrat, caz în care este necesară doar completarea datei de expirare și a cantității introduse. O ultimă variantă de identificare a produselor se realizează prin detecție automată folosind inteligența artificială. Din cauza faptului ca forma produselor alimentare se deosebește destul de mult de la o firmă la alta și condițiile de păstrare a produselor sunt diferite, aria de identificare a produselor folosind inteligența artificială s-a restrâns considerabil, rămânând doar produsele care își păstrează forma și nu se deosebesc din punct de vedere a temperaturii pe care o necesită precum: fructele și legumele. Modalitatea de detectare a acestei categorii este una mai amplă și necesită antrenarea unei rețele neuronale care în final să fie capabilă să identifice tipul de fruct sau legumă dintr-o imagine. Acest proces este unul mult mai benefic pentru utilizator din cauza faptului că economisește considerabil timpul de stocare virtuală a produselor deoarece procesul constă în efectuarea unei fotografii care să cuprindă fructele și legumele achiziționate. În timp ce produsele care conțin codul de bare necesită înregistrarea acestora pe rând și înainte ca ele să fie introduse în frigider, avantajul înregistrării fructelor și a legumelor este că fotografia poate să fie efectuată oricând, chiar și în momentul în care ele sunt depozitate în rafturile destinate lor, iar apariția mai multor tipuri de fructe sau legume în cadru, nu este considerat un impediment. Deși detectarea are scopul de a recunoaște cât mai exact produsele, este esențial să se ia în considerare o marjă de eroare și de aceea, în urma detecției, este transmisă spre utilizator o listă care să conțină rezultatele detecției, iar utilizatorul va fi responsabil de confirmarea sau eliminarea acestora.

Procesul de eliminare a produselor din baza de date se împarte și de această dată pe categorii, o posibilă modalitate de eliminare se face prin intermediul aplicației mobile accesând anumite butoane sugestive, iar cealaltă categorie printr-un Raspberry Pi atașat pe frigider care este capabil să detecteze mișcarea de eliminare a produselor din interiorul electrocasnicului și tipul de aliment. Tot astfel cum în cazul adăugării produselor, eliminarea automată se face folosind inteligența artificială ce vizează și în acest caz doar

categoria de produse fructe/legume. Deoarece ștergerea unui element prin intermediul aplicației necesită mai întâi căutarea acestuia, se pune la dispoziție o împărțire a alimentelor pe categorii în funcție de tipul de aliment, astfel că există tipurile : lactate, fastfood, fructe, legume, dulciuri, preparate, lichide, iar pentru produsele care nu își găsesc încadrarea în aceste tipuri, există categoria intitulată ”altele”. Încadrarea unui produs în aceste categorii se face de către utilizatorul aplicației, însă dacă nu se dorește o astfel de abordare, produsele vor fi adăugate automat în cadrul categoriei ”altele”.

Aplicația oferă utilizatorului libertatea de a alege scopul utilizării acesteia deoarece, în cazul în care este interesat să determine temperatura optimă pentru a economisi curent electric, este nevoit să completeze temperatura pe care fiecare element adăugat îl necesită, iar întrucât temperatura necesară nu este ușor de aflat de pe ambalajul unui produs, este pus la dispoziție un tabel care să ajute utilizatorul în îndeplinirea acestei sarcini. Pe de altă parte, dacă utilizatorul este interesat să vizualizeze strict produsele stocate în frigider, cât și data lor de expirare, acesta nu este nevoit să completeze temperatura ci poate omite acest câmp.

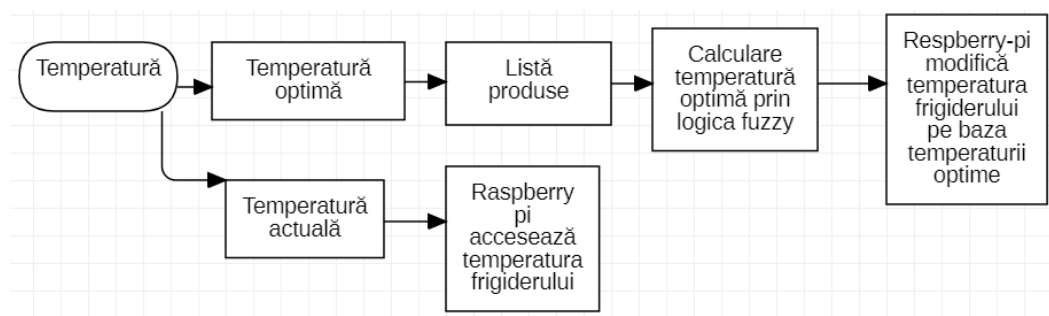


Figure 13: FlowChart-ul care ilustrează modalitatea de identificare a temperaturilor

Scopul principal al aplicației este de a deduce temperatura optimă de care produsele stocate în frigider au nevoie pentru ca starea lor să rămână una stabilă pe o durată de timp cât mai mare. Acest scop are ca repercursiune economisirea curentului electric consumat. Deseori în cazul frigiderelor, termostatul care este responsabil de păstrarea unui anumit interval de temperatura pentru păstrarea produselor, este setat pe o valoare mult mai mare decât este defapt nevoia, iar din aceasta cauză apare consumul exagerat de curent electric care are ca și consecințe costuri mari și chiar afectarea stratului de ozon.

Determinarea temperaturii optime se face pe baza produselor stocate, iar pentru ca această sarcină să fie îndeplinită, este nevoie ca pentru fiecare

produs adăugat să se cunoască temperatura optimă de refrigerare. În cazul produselor ambalate, fiecare producător are obligația de a adăuga această informație pe produs doar că în cele mai multe cazuri informația respectivă este greu de găsit și de aceea adăugarea unui produs folosind codul de bare este esențială, deoarece prin citirea codului se identifică și temperatura, astfel ușurând mult procesul de stocare.

Bazându-ne pe ideea că fiecare aliment își cunoaște temperatura optimă, mai este necesară determinarea cantității pe care o au produsele. Acest factor este necesar fiindcă prezența unei cantități mai mari de mâncare necesită o putere mai mare de refrigerare pentru ca produsele să rămână comestibile. Având lista produselor care conțin cei doi factori principali, se poate determina temperatura optimă aplicând o logică fuzzy. Acest tip de logică aparține domeniului inteligenței artificiale și este folosit deseori în procesul de funcționare a electrocasnicelor și mai ales în modalitatea de funcționare a mașinilor de spălat haine. După aflarea temperaturii optime se ia în calcul temperatura actuală care este determinată cu ajutorul unui senzor conectat la un Raspberry Pi atașat pe frigier. În cazul în care temperatura optimă nu este identică cu cea curentă, se modifică temperatura frigiderului prin intervenția Raspberry pi-ului care transmite electrocasnicului o anumită comandă.

### **3.3 Implementarea ideii într-un API**

Pentru ca ideea stabilită anterior să își mărească valoarea și să fie folosită în cât mai multe cazuri, propunem o implementare a unui API capabil de a surprinde principalele atribuții pe care aplicația le poartă. Subcapitolele prezentate în următoarele secțiuni descriu mai amănunțit marile categorii care fac specială ideea principală.

#### **3.3.1 Adăugarea și ștergerea fructelor și a legumelor**

În procesul de ștergere și de adăugare a fructelor și a legumelor, se creează o rețea neuronală cu ajutorul căreia se vor putea detecta pe baza imaginilor sau a unui video, tipurile și numărul de fructe sau legume ce se regăsesc în imagini. Acest pas este des abordat și are o importanță majoră în îndeplinirea scopului propus, de aceea în capitolul "Identificarea produselor stocate în frigider" se pune un accent deosebit pe modalitatea de abordare ce este găsită în cadrul articolelor din bibliografie.

Urmărind modalitățile prin care cercetătorii au decis să detecteze fructele și legumele în diferite ipostaze cu ajutorul inteligenței artificiale, se evidențiază

2 etape importante în îndeplinirea acestei sarcini. O primă etapă ține de alegerea unei arhitecturi potrivite de rețea neuronală, iar cea de-a doua etapă constă în alegerea setului de date necesar, aceasta fiind în strânsă legătură cu prima deoarece alegând o rețea preantrenată există beneficiul ca setul de date să fie unul mult mai mic decât în condițiile în care se alege o rețea care nu a mai fost antrenată. Ținând cont de aceste etape și din dorința de a aduce în prim plan o soluție cât mai eficientă, am ales ca abordare utilizarea unei rețele neuronale preantrenate pe setul de date COCO care cuprinde imagini cu 80 de tipuri de obiecte. Deși cele mai multe studii bazate pe acest subiect se folosesc de rețele de tip RCNN sau Fast-RCNN, în dezvoltarea acestui tip de API am folosit, în final, o rețea bazată pe segmentare.

Într-o primă încercare a implementării rețelei am folosit RCNN împreună cu framework-ul Tensorflow destinat detectării obiectelor care se găsește sub denumirea de "Tensorflow Object Detection API". Setul de date a fost creat folosind instrumentul grafic "LabelImg" cu ajutorul căruia s-au făcut adnotările necesare. Etichetările fructelor și a legumelor din imagine s-au făcut folosind dreptunghiuri a căror coordonate au fost salvate în fișiere .xml. Deoarece acest framework folosește un set de rețele preantrenate, nu a fost necesară colectarea unui set de date uriaș. Problemele care au apărut în realizarea acestui pas au fost de natură tehnică și au avut legătură cu incompatibilitatea dintre versiunea de CUDA și antrenarea propriu-zisă deoarece după un anumit număr de iterații efectuate, antrenamentul eșua. În încercarea de a găsi o soluție care să rezolve această problemă, am descoperit platforma Colab oferită de Google care pune la dispoziție libertatea de a executa cod Python și în special algoritmi din domeniul inteligenței artificiale împreună cu accesul gratuit la GPU. Această platformă oferă accesul gratuit la resursele oferite de Google timp de 12h, timp în care se poate antrena o rețea neuronală cu un număr acceptabil de iterații.

Pe parcursul duratei de aprofundare a informațiilor despre Colab, am ales o abordare diferită a modalității de antrenare al rețelei și anume folosind platforma Detectron2 [17] oferită de FAIR(Facebook AI Research). Această platformă a adus un beneficiu deoarece deține multiple exemple de implementare folosind Colab și se folosește în principiu de acesta pentru antrenare. Detectron2 include implementări pentru următorii algoritmi de detectare al obiectelor: Mask R-CNN, RetinaNet, Faster R-CNN, RPN, Fast R-CNN, TensorMask etc.

Bazându-ne pe faptul că adăugarea unui produs necesită detectarea obiectelor pe baza unei imagini, în timp ce ștergerea unui produs necesită detectarea obiectelor pe baza unui video, am ales în cele din urmă ca modalitate de a antrena rețeaua, algoritmul Mask R-CNN, acesta fiind o extensie a algoritmului Fast R-CNN care are avantaje din punct de vedere al calității în

momentul detecției pe baza videoclipului. În ceea ce privește modalitatea de etichetare a obiectelor, am ales pentru această implementare "LabelMe" care este un instrument grafic de adnotare a imaginilor. Privilegiul deosebit a acestui instrument este posibilitatea de a adnota imaginile folosind diferite forme geometrice plane pentru o mai bună identificare a obiectului care permite chiar și crearea unui contur ce imită forma reală a oricâtui obiect.

În ceea ce privește aria aceasta a produselor, am decis că pentru început să antrenăm un algoritm în urma căruia să se poată detecta următoarele alimente: mere, pere, roșii, banane, portocale și castraveti. Am ales aceste fructe și legume deoarece sunt unele dintre cele mai des întâlnite alimente ce sunt depozitate în frigider.

### **3.3.2 Adăugarea și ștergerea alimentelor ambalate și a celor preparate**

Adăugarea unui produs ambalat care conține un cod de bare și adăugarea unui produs preparat în bucătărie se face pe baza informațiilor de bază despre alimentul respectiv. Aceste informații țin cont de denumirea produsului, de temperatura pe care produsul îl necesită, de data de expirare și de achiziție a produsului adăugat și de cantitatea respectiv numărul de produse adăugate de același tip. Ștergerea unui produs ambalat sau a unui produs preparat în bucătărie se face pe baza unui id unic pe care fiecare produs stocat în baza de date îl primește automat.

### **3.3.3 Determinarea temperaturii optime**

În lupta împotriva consumului ridicat de electricitate în domeniul gospodăriilor, este necesară aducerea în prim-plan a modalității de combatere a acestei probleme prin calcularea temperaturii optime ce este necesară în frigider pentru ca produsele să nu se deterioreze din cauza temperaturii ridicate, iar consumul să nu fie unul în exces.

Determinarea temperaturii optime necesită ca în momentul în care produsele sunt adăugate în baza de date, ele să conțină temperatura necesară, cantitatea și numărul de produse de același fel, iar pe baza acestor produse se aplică logica fuzzy în urma căreia se află temperatura cea mai potrivită ce este necesară în mediul de stocare.

Logica Fuzzy este o paradigmă bazată pe modalitatea de gândire a oamenilor astfel că după cum un om acționează în funcție de anumite informații pe care le primește la nivelul creierului, tot așa și logica Fuzzy ia decizii în funcție de anumite date. Diferența este că un om, după ce primește anumite informații, ia o decizie în funcție de incertitudinile, impreciziile și judecata

sa, în timp ce un calculator poate să ia decizii doar în funcție de valorile exacte pe care le primește. Logica fuzzy este o metoda exactă de rezolvare a unor probleme și se diferențiază de logica clasică printr-o extindere care cuprinde valori ale adevărului parțial, adică o posibilă valoare între "complet adevărat" (1) și "complet fals" (0). În timp ce logica clasică poate să ia valori doar de 0 sau 1, unde 0 reprezintă falsitatea completă iar 1 adevărul complet, logica fuzzy poate să ia valori care să indice gradul de apartenență la una dintre cele doua clase. Această logică este aplicată în domeniul vânzărilor și din ce în ce mai mult în domeniul electrocasnicelor precum mașinile de spălat rufe, cuptoarele cu microunde, aparatele de aer condiționat etc. În îndeplinirea cauzei propuse, am ales ca modalitate de rezolvare interferența Mamdani și Sugeno luând în calcul două reguli. O regulă este dată de cantitatea tuturor produselor stocate în frigider, iar cealaltă regulă vizează gradul mediu și ponderat de refrigerare a produselor. Combinând cele două reguli se poate îndeplini scopul nostru de a determina temperatura optimă deoarece este necesară emanarea unei temperaturi mai scăzute în cazul în care în frigider sunt depozitate un număr mai mare de produse. Se poate zice în acest caz că temperatura depinde de cantitatea depozitată și de temperatura pe care un produs o necesită.

### 3.4 Specificații clare și testare endpointuri API

| Request    |  | Adăugare manuală produs  |
|------------|--|--|
| Metoda     |  | POST   |
| URL        |  | /api/aliment/add   |
| Parametrii |  | idUser: int (identificatorul unic al utilizatorului)   |
|            |  | cod_de_bare: String( codul de bare al produsului sau string null daca nu există)                         |
|            |  | nume:String (denumirea produsului)   |
|            |  | categorie: String (Încadrarea produsul într-o anumită categorie)   |
|            |  | cantitate: String (cantitatea produsul în grame)   |
|            |  | bucati: int (numărul de dubluri ale unui anumit produs)  |
|            |  | temperatura: int( temperatura necesară păstrării produsului)   |
|            |  | data_expirare: String( în formatul dd-MM-yyyy)   |
|            |  | Data_achizitionare: String( în formatul dd-MM-yyyy)  |
| Response   |  | Confirmarea prin format JSON care conține câmpurile din secțiunea Request plus codul unic al produsului. |

În ceea ce privește adăugarea manuală a unui produs, este necesară crearea unei cereri de tip POST catre API care să conțină parametrii specificați în tabelul anterior. Calea de acces către această funcționalitate este formată din cuvântul "api" urmat de cuvântul "aliment" care sugerează asupra căror tip

de date se efectuează adăugarea, iar în cele din urmă apare cuvântul "add" care sugerează tipul acțiunii. Ca răspuns a acestei cereri, se primesc înapoi datele trimise și în plus un identificator unic a produsului adăugat.

| Request  | Adăugare automată produs prin detecție  |
|----------|---|
| Metoda   | POST  |
| URL      | /api/detectare/add  |
| Paramtru | file: MultipartFile ( contine o imagine)  |
| Headers  | Content-Type = multipart/form-data  |
| Response | Listă in format JSON cu câmpurile: id, idUser, cod_de_bare, nume, categorie, cantitate, bucati, temperatura, data_expirare, data_achizitionare. |

Adăugarea automată necesită o cerere de tip POST care conține un parametru "MultipartFile" ce stochează o imagine. După ce această cerere a fost efectuată, se începe crearea setului de date care va trebui trimis ca răspuns. Setul de date conține produsele ce se vor detecta pe baza imaginii primite aplicând un algoritm capabil să recunoască cât mai multe dintre obiectele respective. Odată ce detecția a luat sfârșit, se trimite spre client setul de date pentru ca clientul să afirme sau să șteargă posibilele detectări efectuate greșit.

| Request   | Ștergere manuală produs                      |
|-----------|--|
| Metoda    | DELETE                                       |
| URL       | /api/aliment/remove/{id}                     |
| Parametru | id: int (identificatorul unic al produsului) |

Ștergerea manuală a produsului se face prin intermediul identificatorului unic atribuit în mod automat fiecărui aliment în momentul în care acesta este adăugat în baza de date. Se procedează în acest fel din cauza faptului ca ștergerea după identificatorul unic este cea mai sigură cale pentru ca datele să nu se piardă. Din cauza acestei abordări, în momentul adăugării unui produs, se trimit ca răspuns informațiile plus identificatorul folosit în cadrul acestei metode de tip DELETE.

| Request  | Ștergere automată produs prin detecție  |
|----------|---|
| Metoda   | POST                                    |
| URL      | /api /detectare/remove                  |
| Paramtru | file: MultipartFile ( contine un video) |
| Headers  | Content-Type = multipart/form-data      |

Ștergerea automată se efectuează prin intermediul unui videoclip pe baza căruia se face detecția mișcării de la dreapta la stânga a produsului și detecția asupra tipului de aliment. În această categorie vor fi detectate doar fructele și legumele. Pentru îndeplinirea acestui pas se face o cerere de tip POST folosind url-ul `"/api/detectare/remove"` cu un parametru ce încorporează videoclipul respectiv.

| Request  |  | Obținere temperatură optimă |
|----------|--|-----------------------------|
| Metoda   | GET                                      |                             |
| URL      | <code>/api/aliment/optimTemp</code>      |                             |
| Response | Temp: int (valoarea temperaturii optime) |                             |

Temperatura optimă se obține prin solicitarea unei cereri de tip GET în urma căreia datele stocate virtual în baza de date, vor fi luate, iar prin aplicarea unui algoritm care încorporează logica fuzzy se va determina temperatura optimă ce este necesară produselor stocate.

În cazul eșuării unei solicitări spre API se iau măsuri astfel încât utilizatorul să fie informat cu privire la cauza care a contribuit eșecului. Grație faptului că API-ul prezentat folosește protocolul HTTP(Hypertext Trasfer Protocol) pentru a comunica cu posibillii clienți, se folosesc codurile oferite de acest protocol pentru a informa utilizatorii în privința posibilelor eșecuri sau chiar și în cazul reușitelor. Codurile utilizate de HTTP se numesc coduri de stare și sunt clasificate în 5 clase. Forma acestora este de tipul: 1xx, 2xx, 3xx, 4xx si 5xx. În timpul folosirii API-ului propus, cel mai des se vor întâlni raspunsuri de tipul 2xx, iar în cazul apariției codului 200, cererea efectuată a fost realizată cu succes. Un alt tip de cod care este des întâlnit aparține clasei 4xx și indică prezența diferitelor eșecuri din partea utilizatorului. Această clasă este folosită în cazul în care utilizatorul ar putea greși formularea cererii sau în cazul în care nu este autentificat. Posibilele defecțiuni sau imposibilitatea realizării unei sarcini ce țin de partea de server vor fi semnalizate prin codurile din categoria 5xx.

### 3.5 Detalii de implementare API

Implementarea acestui API necesită o descriere mai amănunțită a modului de funcționare și în special în a modalității de detectare a produselor în momentul adăugării și ștergerii, dar și a modului de determinare a temperaturii optime. Această aprofundare este necesară din cauza faptului că

aceste abordări sunt mult mai amble, iar explicarea lor s-a făcut până în acest moment într-o modalitate mai abstractă.

### 3.5.1 Identificarea fructelor și a legumelor

După cum s-a specificat și în secțiunile anterioare în care s-a abordat acest subiect, identificarea automată a produselor este valabilă doar în cazul fructelor și a legumelor deoarece forma acestora este destul de asemănătoare, în timp ce produsele ambalate au diferite forme și necesită diferite temperaturi de păstrare. În categoria fructelor ce se detectează intră bananele, merele, portocalele, perele și roșiile, iar în ceea ce privește categoria legumelor, am ales o singură clasă și anume castraveții.

Detecția obiectelor menționate anterior a necesitat parcurgerea și realizarea setului de date pe baza carora s-a aplicat un algoritm și mai exact s-a antrenat o rețea neuronală capabilă să efectueze acest pas. În ceea ce privește setul de date, s-au colectat date în mai multe moduri și în principiu descarcând imagini de pe internet prin intermediul extensiei "Download Images" care se găsește în magazinul web Chrome. După o observare mai detaliată a imaginilor descărcate, am căutat imagini care să aibă un fundal cât mai apropiat de realitate sau am realizat unele care să cuprindă cât mai multe fructe în cadru. Imaginile colectate au fost împărțite în două categorii, în setul de date necesar pentru antrenament, respectiv în setul de date necesar pentru testare. Din aproximativ 120 de imagini destinate fiecărui tip de aliment, aproximativ 85% a fost folosit pentru antrenament, iar restul de 15% a fost folosit pentru procesul de testare. În figura alăturată se evidențiază numărul de adnotări făcute pe imagini. Se poate observa faptul că pentru un anumit produs, apar un număr mult mai mare de instanțe deoarece într-o poză se pot afla 1 produs sau chiar mai multe.

| Categorii | Nr. instanțe | Categorii | Nr. instanțe | Categorii  | Nr. instanțe |
|-----------|--------------|-----------|--------------|------------|--------------|
| mere      | 227          | banane    | 220          | castraveți | 279          |
| portocale | 254          | pere      | 381          | roșii      | 320          |
| total     | 1681         |           |              |            |              |

Figure 14: Numarul de adnotari pentru fiecare categorie

Adnotarea setului de date se face folosind instrumentul grafic LabelMe cu ajutorul căreia se poate realiza conturul obiectului dorit. Conturul se face prin unirea unor drepte sub formă de poligon cu foarte multe laturi. Cu ajutorul acestui instrument se pot face și adnotări folosind cercuri sau forme

dreptunghiulare, însă în cazul de față, folosirea poligonului a reprezentat cea mai bună opțiune deoarece forma obiectelor merita pusă în evidență și de asemenea se dorește eliminarea fundalului în cea mai mare măsură astfel încât distincția dintre produsul adnotat și obiectele din jur să fie ușor de înfăptuit.

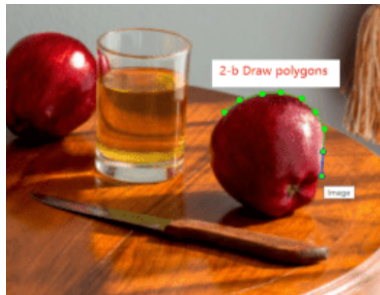


Figure 15: Exemple de adnotare

Detectarea obiectelor este o sarcină provocatoare deoarece presupune localizarea unui sau a mai multor obiecte și clasificarea fiecărui obiect localizat. În ceea ce privește modul de antrenare a rețelei, am ales ca modalitate de abordare modelul Mask R-CNN din dorința ca pe baza unei imagini sau a unui videoclip să se poată efectua cât mai multe operații pentru ca extinderea folosirii acestui API să fie un pas ușor de realizat.

Modelul arhitectural Mask R-CNN a fost introdus în anul 2017 prin intermediul lucrării "Mask R-CNN" [4] și este cel mai nou model folosit în acest domeniu, reprezentând o extindere a detectării obișnuite deoarece implică marcarea pixelilor specifici din imagine ce aparțin obiectului identificat, în timp ce în detectarea obișnuită se scot în evidență obiectele detectate folosind forme dreptunghiulare plane. Extinderea care are loc în cadrul acestui model al rețelei neuronale, se face pe baza arhitecturii Faster R-CNN adăugând o ramură pentru a prezice masca obiectului în paralel cu ramura existentă pentru crearea casetei de delimitare obișnuite.

Sistemul software Detectron2[17] oferă din gama arhitecturilor Faster/Mask R-CNN 3 tipuri de abordări. Abordarea folosită în implementarea acestui API este folosind rețeaua vertebrală FPN care are cea mai mare viteză și acuratețe. În ducerea la îndeplinire a scopului se diferențiază existența a două etape principale în aplicarea Mask R-CNN-ului. În primul rând, se generează propuneri despre regiunile în care ar putea exista un obiect bazat pe imaginea de intrare, în al doilea rând, se prezice clasa obiectului, se rafinează caseta de delimitare și se generează o mască la nivelul de pixeli al obiectului pe baza propunerii primei etape. Aceste etape sunt conectate la rețeaua principală în stil FPN (Feature Pyramid Networks) care are forma piramidală și conține

două trasee și anume: un traseu care are rolul de a extrage caracteristici din imagini folosind ResNet(Residual neural network) și un traseu care generează o hartă piramidală caracteristică, ca dimensiune similară cu calea precedentă.

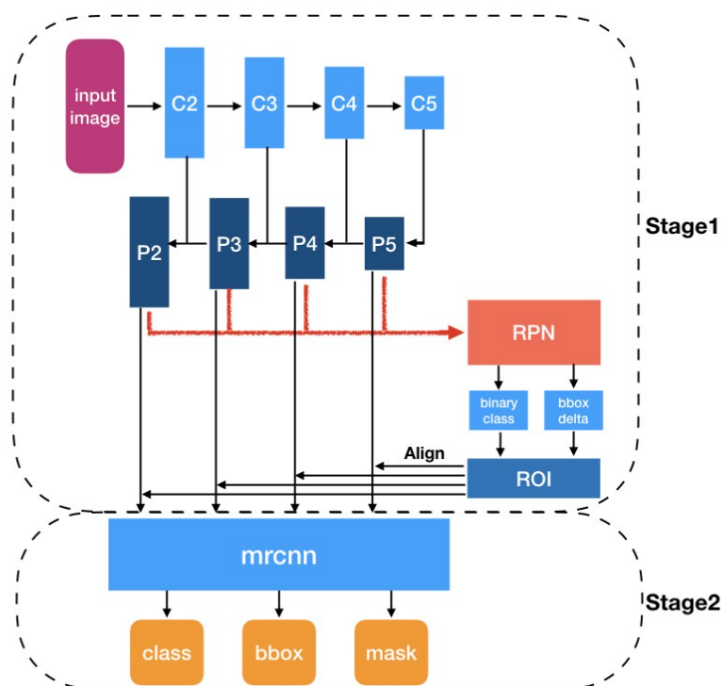


Figure 16: Arhitectura rețelei Mask R-CNN[19]

După obținerea caracteristicilor se aplică o rețea de propuneri a regiunilor (RPN) care prezice practic dacă un obiect este prezent în acea regiune sau nu. În acest pas, obținem acele regiuni sau caracteristici în care s-ar afla obiectul dorit. Un ultim pas din aceasta etapă este dat de determinarea regiunilor de interes(ROI). Regiunile obținute din RPN pot avea forme diferite și prin urmare, se aplică un strat de colectare transformând toate regiunile în aceeași formă. În continuare, regiuni care prezintă interes sunt trecute printr-o rețea complet conectată pentru a se eticheta clasele și pentru a se determina casetele delimitatoare.

Până în acest moment, etapele sunt aproape similare cu modul în care funcționează Faster R-CNN, urmând ca în următoarea etapă să se genereze masca de segmentare. Pentru aceasta, se calculează mai întâi regiunea de interes, astfel încât timpul de calcul să poată fi redus iar apoi se calculează coeficientul IoU (Intersection over Union) cu ajutorul căruia se decide dacă

intr-adevăr regiunea selectată este de o importanță mare. După obținerea ROI-urile bazate pe valorile IoU, putem adăuga o ramură "Mask" arhitecturii existente, aceasta returnând masca de segmentare pentru fiecare regiune care conține un obiect.

Atât arhitectura folosită cât și valorile care ies în urma aplicării acestui model, sunt prezentate în figura 16 pentru a avea un reper mai exact a modului de funcționare.

Ponderile obținute în urma antrenamentului sunt salvate într-un fișier de tip ".pth", iar pentru ca acesta să fie mai ușor de utilizat, fișierul obținut se convertește într-un fișier .pb cu ajutorul careia se poate salva un graf al ponderilor. Deoarece implementarea API-ului se face în limbajul Java, în timp ce antrenamentul este realizat în limbajul Python, detecția se face apelând un script de tip .py din interiorul limbajului Java.

### 3.5.2 Determinarea temperaturii optime folosind logica fuzzy

Obținerea temperaturii optime se face aplicând logica fuzzy, interferența Mamdani și Sugeno. Această abordare combină două reguli stabilite pe baza produselor stocate în baza de date. După cum s-a precizat și în secțiunea de implementare a ideii, cele două reguli iau în calcul cantitatea introdusă în frigider și respectiv, temperatura de care fiecare produs are nevoie.

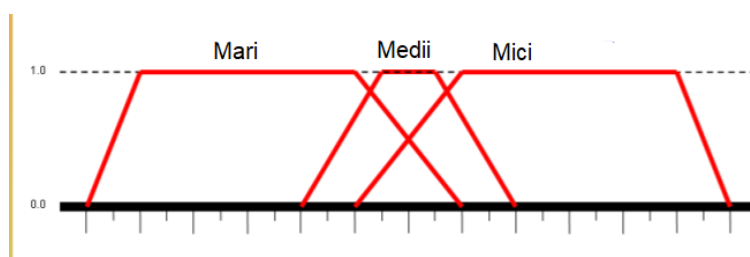


Figure 17: Forma trapezoidală

Regula ce calculează temperatura medie ponderată a fiecărui produs, folosește forme trapezoidale pentru a distinge 3 clase ce se întrepătrund și anume: clasa în care media calculată are valori mari( peste 4 grade Celsius), clasa în care media ponderată are valori medii(între 4,5 și 1,5 grade) și clasa în care produsele necesită valori foarte scăzute(de la 2 până la 0 grade).

În ceea ce privește compoziția celei de-a doua regulă, se deosebesc 5 funcții pentru care se vor calcula gradul de apartenență, iar aceste funcții delimitează următoarele 5 clase care răspund la întrebarea "Care este cantitatea tuturor produselor? : "foarte mică", "mică", "medie", "mare" și "foarte

mare”. Formele ilustrate în figura alăturată combină funcții sigmoidale cu clopotul lui Gauss.



Figure 18:

Ca rezultat a suprapunerii acestor funcții și a aplicării regulilor, se efectueaza defazificarea cu ajutorul careia se obține rezultatul dorit și anume temperatura optimă.

Deși scopul inițial a fost ca în urma determinării temperaturii optime, prin intermediul raspberry pi-ului atașat, să se modifice termostatul frigiderului astfel încât temperatura din interior să fie cea obținută în urma testării fuzzy. Realizarea acestui pas s-a dovedit a fi imposibil de realizat în momentul actual din cauza faptului că fiecare frigider are un sistem diferit de funcționare, iar modificarea dorită necesită cunoașterea unui driver special creat pentru acel sistem. Din această cauză am ales ca temperatura calculată să fie returnată utilizatorului pentru ca el să acționeze în aceasta privinta prin modificarea temperaturii în mod manual.

## 4 Dezvoltarea aplicației mobile și arhitectura

### 4.1 Motivație

Dezvoltarea acestei aplicații a fost motivată în primul rând de către domeniul de activitate al părinților mei, aceștia deținând un atelier de reparații electrotcasnice. Informația de bază care m-a făcut să mă gândesc la realizarea acestui tip de aplicație am obținut-o chiar de la tatăl meu care îmi relatea problemele frecvente pe care le întâlnește și anume consumul exagerat de curent electric pe parcursul reparării unui frigider.

Un alt motiv pentru care am ales realizarea acestei aplicații este dat de problematica încălzirii globale care a ajuns fundamentală și care este relatată din ce în ce mai des prin intermediul documentarelor și a interesului dat de cât mai mulți oameni în diferite domenii.

Un al treilea motiv este dat de către dorința de a aprofunda anumite domenii care mi-au trezit un interes mai aparte pe parcursul facultatii și anume inteligența artificială și modul de interacțiune al omului cu tehnologia. Domeniul inteligenței artificiale apare în aplicația propusă în momentul identificării fructelor și legumelor cât și a modului de identificare a temperaturii optime, iar modul de interacțiune a omului cu tehnologia constă în realizarea mai multor tematici de culori care oferă o ambianță cât mai plăcută, dar și în modalitatea ușoară de accesare a taskurilor de o importanță mare.

## 4.2 Arhitectura aplicatiei

### 4.2.1 Comportament

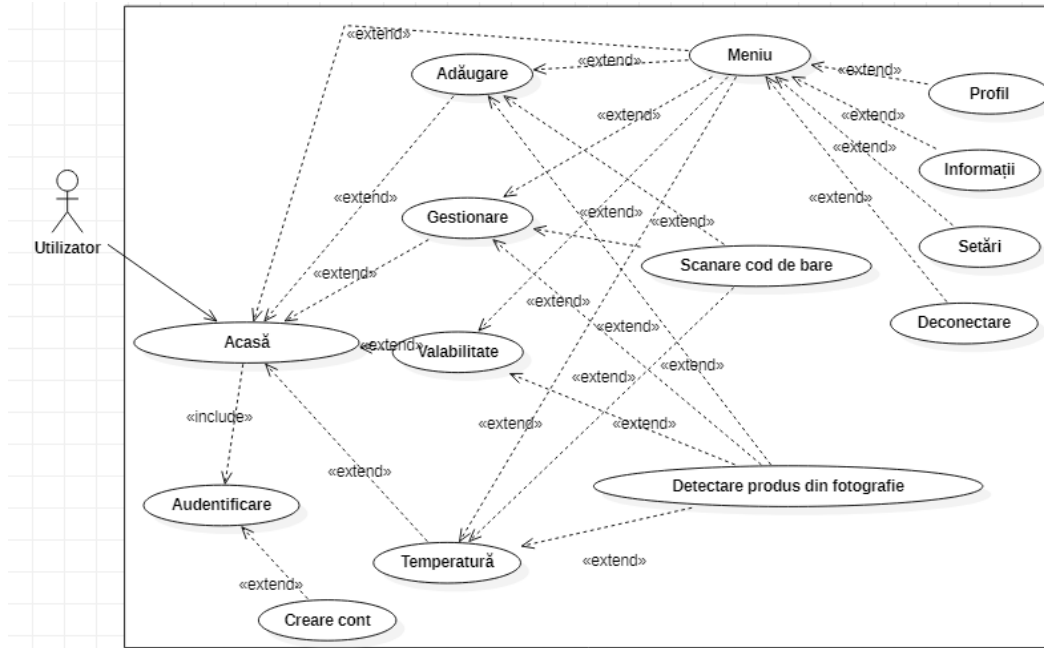


Figure 19: Diagrama cazurilor de utilizare

Comportamentul sistemului este ilustrat prin intermediul diagramei cazurilor de utilizare care conține un actor reprezentat prin utilizatorul aplicației de mobile. Conține de asemenea cazurile de utilizare și legăturile dintre acestea care indică posibilitatea de accesare a utilizatorului. La pornirea aplicației, pentru ca utilizatorul să vizualizeze fereastra principală, acesta este obligat să se autentifice sau să își creeze un cont. Din fereastra principală intitulată "Acasă", utilizatorului îi sunt afișate diferite posibilități de accesare precum : adăugarea unui produs care nu folosește cod de bare ("Adăugare"), vizualizarea și gestionarea produselor stocate ("Gestionare"), afișarea produselor care expiră într-o anumită lună selectată ("Valabilitate"), determinarea temperaturii optime și a temperaturii actuale ("Temperatură"). O importanță deosebită este dată posibilităților: de scanare a produselor ("Scanare cod de bare"), de detectare a produselor prin efectuarea unei fotografii ("Detectare produs din fotografie") și de accesare a unei meniuri care face posibilă deconectarea, modificarea unor setări, accesarea informațiilor și vizualizarea datelor persoanele. Posibilitățile menționate mai sus, care au o importanță mai mare, sunt accesibile din orice fereastră din aplicație.

## 4.2.2 Structura

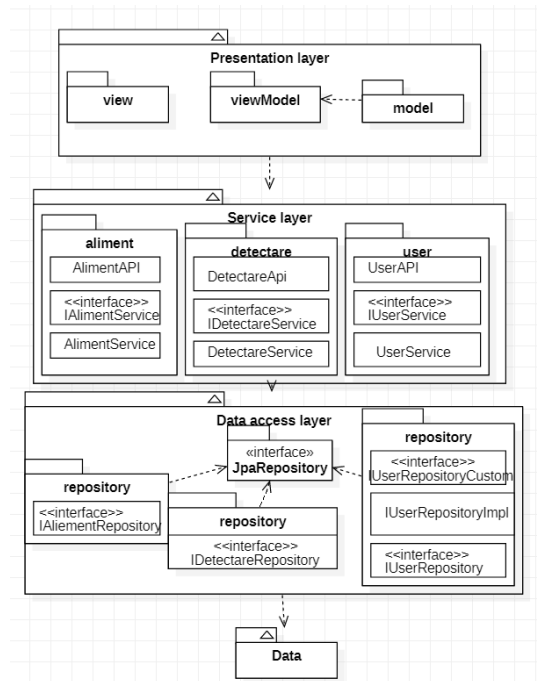


Figure 20: Diagrama de componente

Diagrama de componente prezentată mai sus are scopul de a modela aspectul fizic dat de sistem. În cazul unei aplicații mai ample, modalitatea de aranjare a fișierelor, pachetelor, bibliotecilor etc. este esențială și de aceea folosirea unei arhitecturi aduce îmbunătățiri remarcabile precum: o întreținere mai ușoară a aplicației și o înțelegere mai rapidă și mai bună a codului sursă.

Aplicația creată este repartizată în 4 părți. O primă parte intitulată "Presentation Layer" conține pattern-ul arhitectural MVVM(Model-View-ViewModel) care are scopul de a afișa și de a prelucra datele primite de la server. Modalitatea de funcționare a acestei arhitecturi aplicate este descrisă detaliat în secțiunea următoare. Al doilea strat intitulat "Service layer" se ocupă de partea mai amblă de gestiune a datelor. Comunicarea dintre cele doua pachete se face prin intermediul unor mesaje care se transmit în format Json. Modalitatea de comunicare a pachetului "Presentation" se realizează prin Retrofit, care este un pachet REST proiectat pentru aplicațiile mobile. De asemenea partea de Service trimite și de asemenea primește mesaje prin intermediul Framework-ului Spring. Cea de-a 3-a parte a acestei diagrame este ilustrată prin intermediul stratului "Data access" care descrie modalitatea de

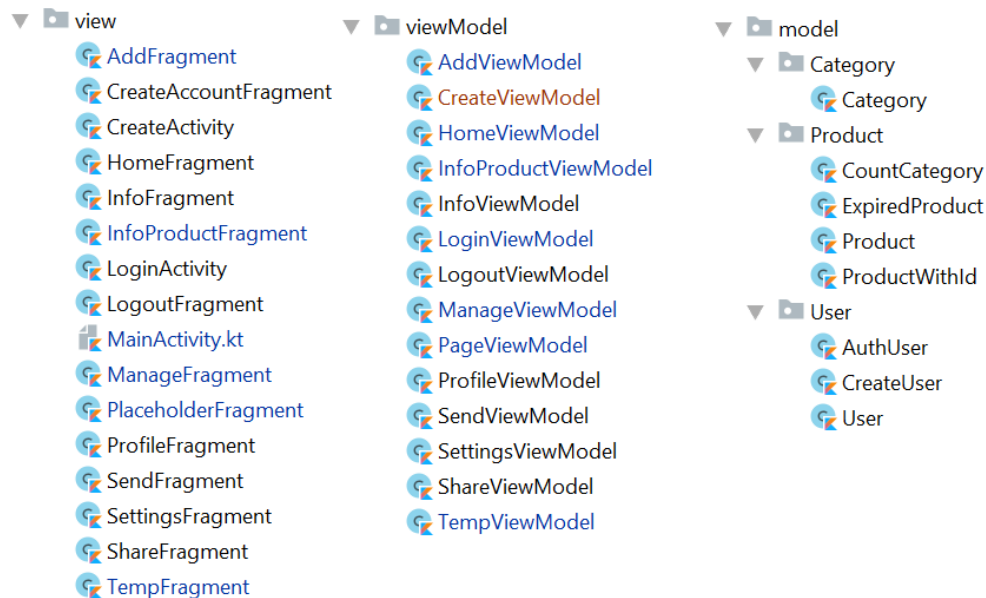
interacțiune a server-ului cu partea în care se stochează datele. Pentru ca această comunicare să fie posibilă se folosește interfața JpaRepository. Ultimul strat "Data" conține baza de date implementată în MySQL.

### 4.2.3 Arhitectura MVVM

Arhitectura unei aplicații se definește ca fiind modul în care este organizat și gestionat codul sursă astfel încât acesta să fie unul funcțional. În momentul proiectării unei aplicații se decide tipul de arhitectură folosită, menită să structureze aplicația într-un mod cât mai avantajos pentru cerințele avute. Alegerea unei arhitecturi este un pas esențial deoarece modificarea structurii în urma implementării este foarte costisitoare și de aceea se recomandă ca această decizie să fie luată înainte de începerea scrierii codului sursă.



MVVM (Model- View - ViewModel) este o arhitectură software care îmbunătățește modul de aranjare al codului și de asemenea facilitează modalitatea de separare a interfețelor grafice față de partea de logică și prelucrare a datelor. Se poate zice că acest model reprezintă o evoluție al modelului MVC ( Model - View - Controller), doar că în locul Controller-ului apare partea de ViewModel. Exceptând înlocuirea etichetelor, există și diferențe mai considerabile, la nivelul codului, care constau în: incapacitatea View-ului de a comunica cu parte de Model și modalitatea de comunicare între View și ViewModel. Pachetul intitulat "Model" conține definițiile obiectelor. Această parte comunică doar cu ViewModel-ul. View-ul este un pachet ce conține partea vizuală, adică interfețele grafice (butoanele, graficele, controalele etc.). De asemenea, este responsabil de conectarea elementelor prezente în interfața cu partea de cod, însă deciziile logice asupra datelor nu își au locul în acest pachet. Comunicarea care are loc este tot cu ViewModel-ul, ca și în cazul pachetului Model. Legătura dintre Model și View se face prin intermediul pachetului ViewModel. Această parte a arhitecturii este cea mai stufoasă deoarece aici se efectuează operațiile logice asupra datelor, cât și validarea acestora. Modalitatea de comunicare a ViewModel-ului cu View-ul se face



prin mijlocul pattern-ului Observable, astfel că în momentul unor schimbări făcute la nivelul View-ului, ViewModel-ul este responsabil să reacționeze. În imaginea de mai sus apare modalitatea de structurare și denumire a fișierelor Kotlin. Pachetele sunt denumite conform arhitecturii MVVM, iar fiecare fișier poartă numele ferestrei de care este responsabil. La pornirea aplicației apare fereastra de autentificare, iar fișierele responsabile de îndeplinirea acestei sarcini sunt următoarele: "LoginActivity" din pachetul view, "LoginViewModel" din pachetul viewModel și "AuthUser" din pachetul model. Pentru crearea unui cont nou, fișierele responsabile sunt: "CreateActivity" din view, "CreateViewModel" din ViewModel și obiectul "CreateUser" din pachetul model. În urma autentificării, se afișează mai multe opțiuni precum adăugarea manuală a unui anumit produs, posibilitatea verificării temperaturii actuale și a temperaturii optime, afișarea produselor care expiră într-o anumită lună, afișarea cât și gestionarea produselor, citirea unui cod de bare și efectuarea unei fotografii pe baza căreia se efectuează detecția. Fiecărei ferestre îi sunt atribuite câte un fișier din pachetul view, respectiv un fișier din pachetul ViewModel. Aceste două fișiere comunică între ele prin Observer-ul LiveData care este conștient de activitatea sau inactivitatea utilizatorului. Comunicarea se realizează între fișierele: "HomeFragment" și "HomeViewModel", "AddFragment" și "AddViewModel", "InfoProductFragment" și "InfoProductViewModel", "ManagerFragment" și "ManagerViewModel", "TempFragment" și "TempViewModel" etc.

#### 4.2.4 Modul în care se leagă structura și comportamentul

#### 4.2.5 Ștergere produs

Participanți: Utilizator

Flux de evenimente:

1. Utilizatorul deschide aplicația și se autentifică.
2. Accesează fereastra de gestionare produse.
3. Alege produsul pe care dorește să îl șteargă.
4. Apasă butonul care ilustrează ștergerea produsului.
5. Confirmă ștergerea

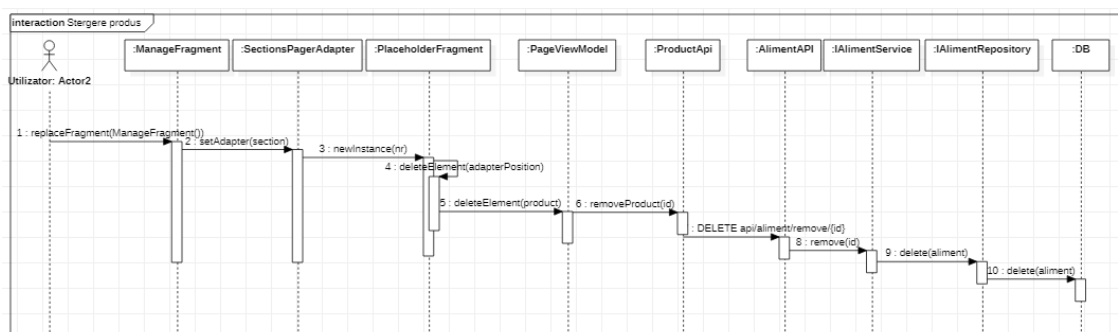


Figure 21: Diagrama de secvență pentru ștergere produs

În situația în care posesorul aplicației este conștient că un anumit produs stocat în frigider nu s-a șters automat din aplicație, poate elimina manual produsul respectiv. În acest caz, utilizatorul are obligația de a se autentifica și de a fi conectat la internet în timpul operațiunii de ștergere a produsului, iar după realizarea acestor condiții de intrare, utilizatorul accesează butonul care ilustrează posibilitatea de gestionare a produselor stocate. Luând în considerare împărțirea alimentelor pe categorii, alimentele sunt afișate în 5 ferestre diferite pentru ca gestionarea lor să fie mai ușoară. După alegerea produsului și apăsarea iconiței de ștergere, utilizatorul trebuie să confirme acțiunea pentru ca acesta să nu comită vreo greșală, iar în cazul în care consideră că ștergerea nu trebuie să fie realizată, are opțiunea de anulare a comenzii. În urma efectuării unei ștergeri, utilizatorul este informat în privința realizării cu succes sau nu, a comenzii date.

#### 4.2.6 Adăugarea manuală a produsului

Participanți: Utilizator

Flux de evenimente:

1. Utilizatorul deschide aplicația și se autentifică.
2. Accesează fereastra de adăugare produs.
3. Introduce datele necesare.
4. Apasă butonul de salvare a datelor.

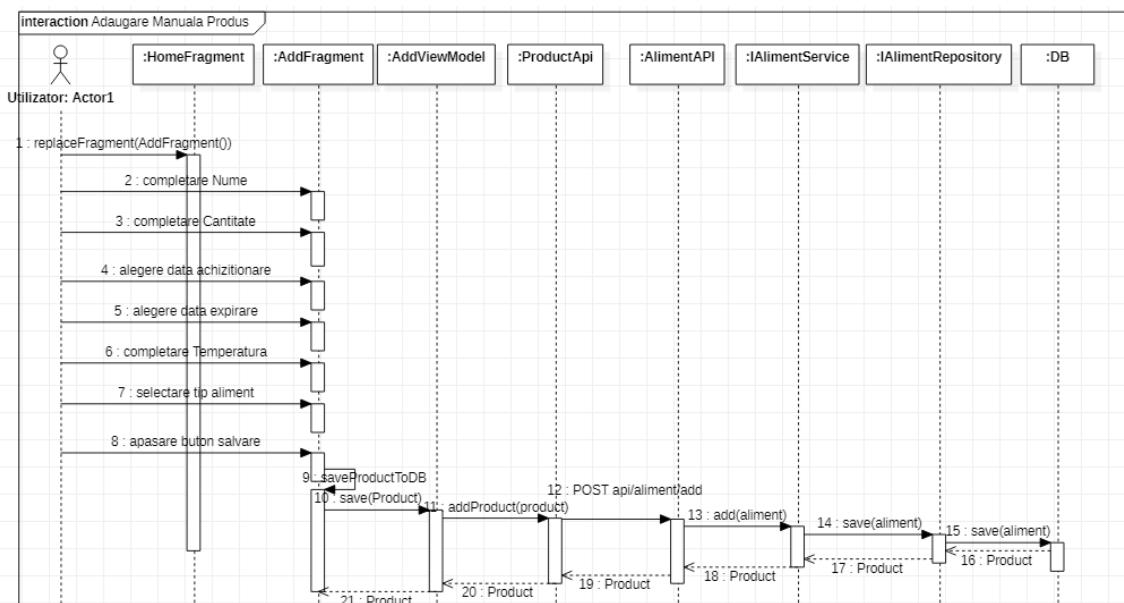


Figure 22: Diagrama de secvență - Adăugare produs

Adăugarea manuală a unui produs este necesară în cazul depozitării alimentelor preparate în bucătărie sau a alimentelor care nu sunt identificate folosind detecția produselor din imagine sau codul de bare, iar pentru aceasta atribuire, utilizatorul este obligat să completeze cateva campuri informative. Alegerea numelui produsului, alegerea datei de expirare, selectarea tipului de aliment și completarea temperaturii sunt campuri obligatorii care trebuie completate, iar cantitatea fiind un camp opțional. În ceea ce privește alegerea datei de achiziționare, se consideră că această dată este identică cu data actuală, în caz contrar, ea poate să fie modificată.

Precondițiile care trebuie să fie îndeplinite constau în autentificarea și în accesul la internet, iar în urma realizării acțiunilor dorite, utilizatorul este informat în legătură cu succesul sau insuccesul acestora.

#### 4.2.7 Editare produs

Participanți: Utilizator

Flux de evenimente:

1. Utilizatorul deschide aplicația și se autentifică.
2. Accesează fereastra de gestionare produse.
3. Alege produsul pe care dorește să îl editeze.
4. Apasă butonul care ilustrează editarea produsului și confirmă editarea.
5. Modifică campurile dorite.
6. Salvează modificările.

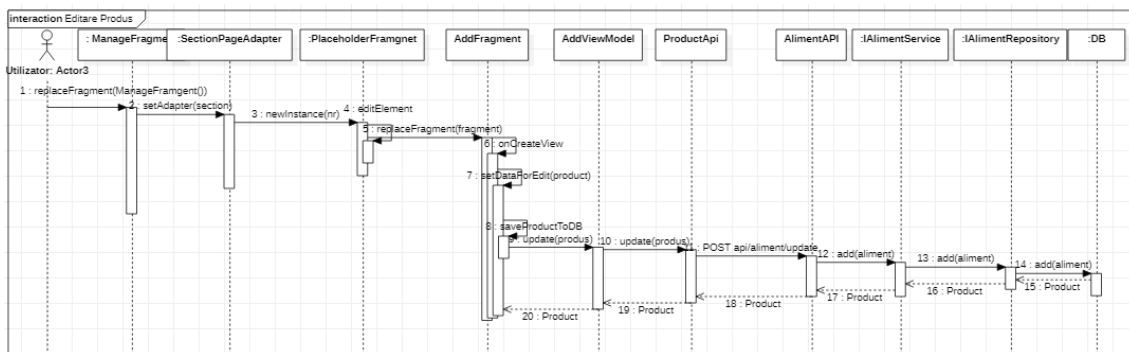


Figure 23: Diagrama de secvență - Modificare produs

Editarea unui produs este necesară în cazul în care utilizatorul introduce date greșite, în cazul în care detecția se efectuează greșit sau în scenariul în care informațiile despre un anumit produs ce conține cod de bare nu sunt suficiente sau nu convin utilizatorului. Pentru a efectua editarea, utilizatorul trebuie să fie autentificat și să aibă conexiune la internet. În urma îndeplinirii acestor condiții, acesta trebuie să acceseze butonul "Gestionare" pentru ca aplicația să îi afișeze toate alimentele pe care le are stocate. Afișarea produselor se face prin intermediul a 5 pagini care impart produsele după tipul lor, de aceea paginile sunt intitulate "Fructe Legume", "Lactate", "Fastfood", "Dulciuri Băuturi" și "Preparate". În urma navigării la tipul de produs dorit, se apasă butonul care indică editarea, se confirmă, iar utilizatorul este transferat la fereastra în care acesta poate să modifice fiecare camp completat cu datele actuale despre produs. După salvarea modificărilor, aplicația înștiințează utilizatorul în privința succesului sau a insuccesului.

#### 4.2.8 Vizualizarea datelor de valabilitate

Participanți: Utilizator

Flux de evenimente:

1. Utilizatorul deschide aplicația și se autentifică.
2. Accesează fereastra numită "Valabilitate" care afișează un chart ce reprezintă numărul de produse care expiră/ luna.
3. Se selectează luna dorită.
4. Se afișează lista cu produsele ce urmează să expire în luna selectată.

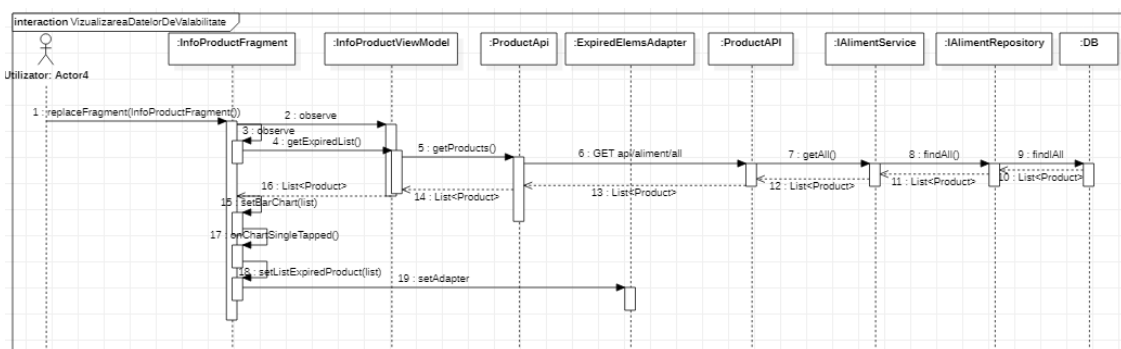


Figure 24: Diagrama de secvență - Vizualizare date de expirare

Deoarece produsele precum fructele, legumele, alimentele preparate, lactatele, dar și alte produse sunt valabile o scurtă perioadă de timp, este necesar ca utilizatorul să poată vizualiza aceste produse pentru a fi conștient de starea lor curentă. Pentru ca această opțiune să poată fi accesată, utilizatorul trebuie să fie autentificat și conectat la internet. Vizualizarea produselor și data lor de valabilitate se listează în fereastra "Valabilitate", iar accesând această opțiune, se poate alege luna actuală sau orice lună dorită, în urma căreia se pot vedea alimentele care expiră în luna respectivă și data în care acestea vor expira. În momentul în care un produs expiră, dar nu este consumat, utilizatorul este informat în privința acestui aspect printr-un mesaj.

#### 4.2.9 Vizualizarea temperaturii actuale și a celei optime

Participanți: Utilizator

Flux de evenimente:

1. Utilizatorul deschide aplicația și se autentifică.
2. Accesează fereastra numită "Temperatură".
3. Se apasă una dintre etichetele afișate

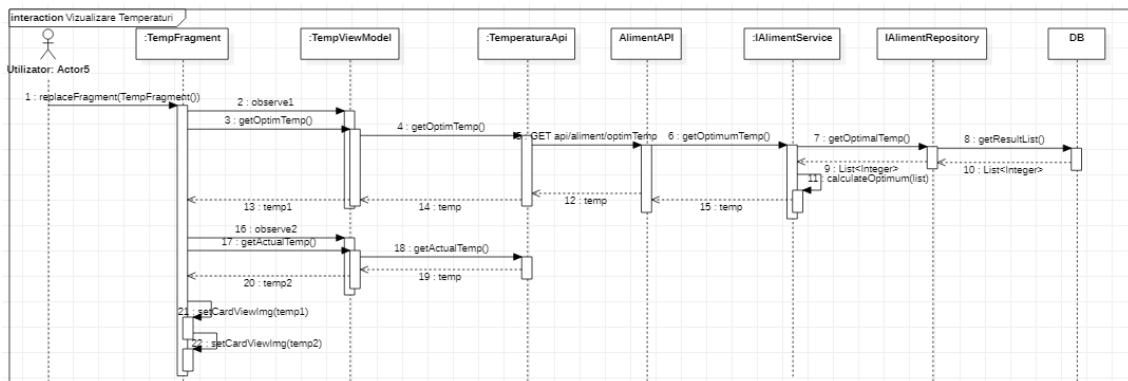


Figure 25: Modul de afișare a temperaturii actuale și a celei optime

Cu scopul reducerii consumului electric, este esențial ca utilizatorul aplicației să fie conștient de temperatura pe care o are frigiderul, dar și de temperatura optimă pe care acesta ar trebui să o aibă. Temperatura optimă se obține pe baza alimentelor stocate în frigider. În această situație se consideră că toate elementele stocate sunt înregistrate în aplicație. Pentru aflarea celor două temperaturi, utilizatorul trebuie să fie autentificat, să aibă acces la internet, dar și să dețină un raspberry și capabil de a obține temperatura actuală a electrocasnicului.

Utilizatorul poate să vizualizeze cele două temperaturi prin accesarea butonului "Temperaturi", urmată de apăsarea pentru actualizare a celor două imagini care conțin temperaturile vechi. În cazul în care obținerea temperaturii actuale sau obținerea temperaturii optime nu este realizată, aplicația afișează un mesaj corespunzător.

#### 4.2.10 Scanarea unui produs care detine cod de bare

Participanti: Utilizator

Flux de evenimente:

1. Utilizatorul deschide aplicația și se autentifică.
2. Utilizatorul accesează iconița care ilustrează scanarea unui produs.
3. Acesta scanează produsul.
4. Completează câmpul "cantitate", selectează tipul de aliment și selectează data de expirare.
5. Apasă butonul "salvare".

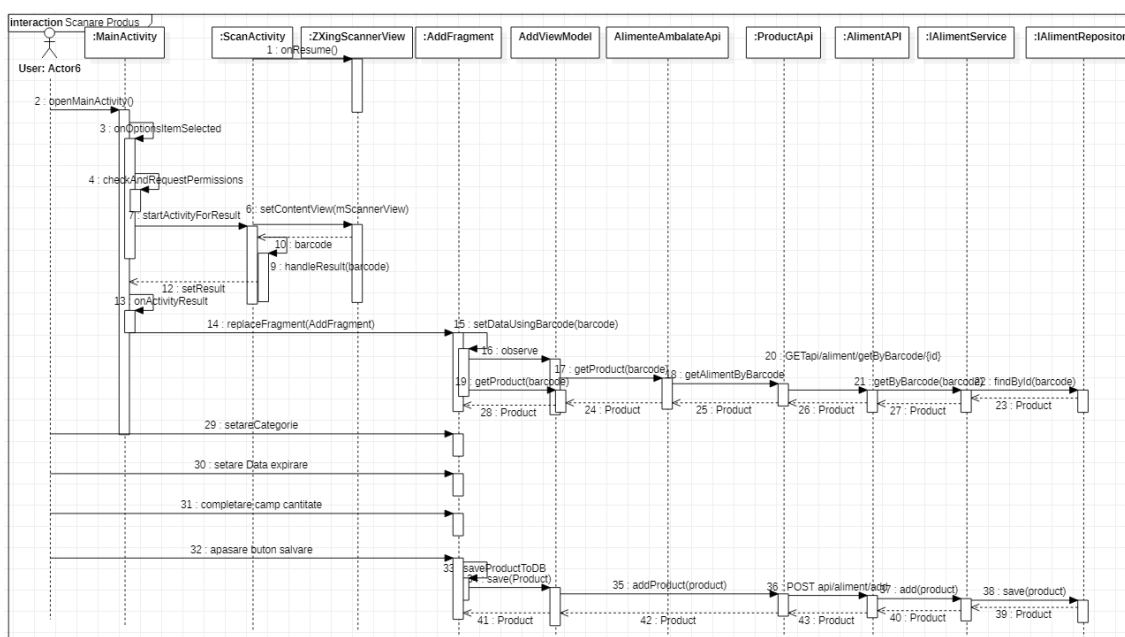


Figure 26: Diagrama de secvență - Scanarea unui produs

Utilizatorul este obligat să se autentifice și să aibă conexiune la internet. Această opțiune este utilizabilă pentru produsele care dețin un cod de bare și anume pentru orice produs achiziționat din magazin. În urma identificării codului de bare prin intermediul unui cititor de bare, aplicația conduce utilizatorul la fereastra de adăugare unde acesta mai trebuie să selecteze data de expirare, cantitatea și categoria în care se încadrează produsul dacă el există deja adăugat în baza de date destinată acestui domeniu. După salvarea datelor, utilizatorul este informat dacă datele s-au salvat cu succes sau a apărut vreo eroare în timpul transmiterii datelor.



## 5 Modalitatea de implementare

### 5.1 Caracterul abstract al arhitecturii

În proiectarea unei arhitecturi, partiționarea sistemului software în pachete și deciderea modalităților de comunicare între acestea este o sarcină importantă care necesită o perioadă substanțială de timp, iar explicațiile pe care trebuie să le facă proiectanții pentru programatori sau clienți pentru ca arhitectura să fie înțeleasă, necesită un anumit nivel de abstractizare. Problemele care nu au o importanță aparte în aplicație trebuie îndepărtate și astfel se aduc la cunoștință doar problemele substanțiale care țin de arhitectură. Există mai multe tipuri de a analiza o arhitectură, însă în aplicația prezentată se pune accent pe arhitectura dintr-o perspectivă logică care are rolul de a descrie elementele semnificative ale unei arhitecturi și relațiile dintre ele.

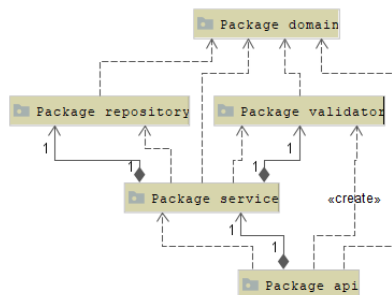


Figure 28: Diagrama de clase

Pachetele, din cadrul aplicației, care se află în același spațiu de memorie vor comunica prin apelarea unor metode. În partea de back-end al aplicației este folosită arhitectura stratificată pentru pachetul "aliment" care se ocupă de toate funcționalitățile care conțin alimente, cu excepția detecției fructelor și a legumelor. Un alt pachet, denumit "user", se ocupă de gestionarea informațiilor personale despre utilizatori, iar ultimul pachet, "detectare", se ocupă strict de detecția fructelor și legumelor din imagini. Aceste pachete conțin la rândul lor mai multe pachete ce sunt denumite astfel: "api", "domain", "service", "validator" și "repository" și care surprind folosirea arhitecturii stratificate. Pachetul "api" conține o clasă care este responsabilă de efectuarea cu succes a comunicării dintre cerințelor aduse de utilizatorul prin intermediul aplicației mobile și realizarea acestor cerințe. Pachetul "domain" conține entitățile din program și ia naștere din momentul în care datele se primesc de la aplicația mobilă până la momentul comunicării cu baza de date, astfel că fiecare strat prezent are o relație de dependență cu "domain". Pachetul "service" se ocupă de partea logică a programului și conține interfața

și implementarea acestora. Comunicarea dintre pachetele menționate anterior este realizabilă prin conectarea acestora printr-o relație de compoziție, adică clasa din pachetul "api" conține un obiect de tipul interfeței aflate în pachetul "service" și astfel se pot apela metode din cadrul interfeței. În cadrul acestui pachet se validează datele prin apelarea unor metode aflate în validator. Apelarea se poate realiza datorită relației de compoziție dintre clasa aflată în "service" și validatorul din pachetul "validator". În cadrul următoarei secțiuni se prezintă diagrama de clase și relațiile dintre acestea.

## 5.2 Implementarea arhitecturii abstracte

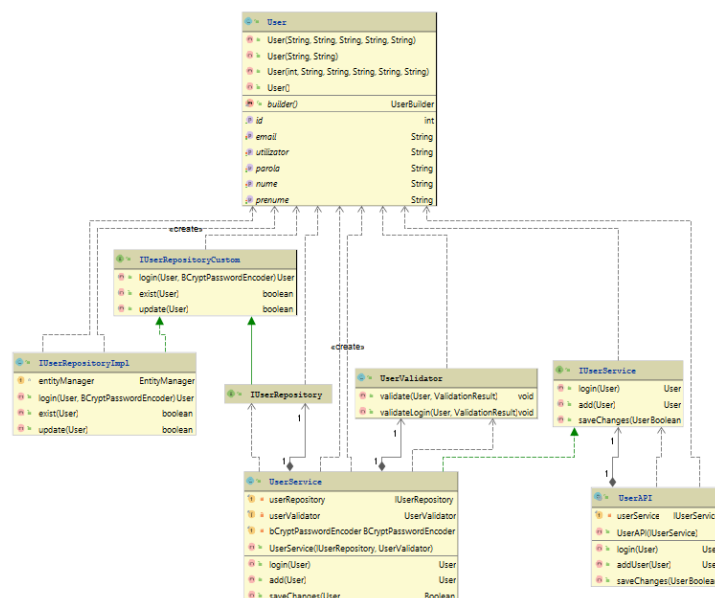


Figure 29: Diagrama de clase pentru gestionarea utilizatorilor

Partea de gestionare a utilizatorilor este repartizată în mai multe pachete conform arhitecturii stratificate, însă în imaginea alăturată în care se prezintă diagrama de clase, sunt excluse pachetele respective și se ilustrează doar conținutul acestora, respectiv legăturile. Clasa din partea de sus a imaginii aparține pachetului "domain" și are ca și câmpuri id-ul unic al utilizatorului, numele, prenumele, email-ul, numele de utilizator și parola cu care acesta se autentifică. Aceste date sunt salvate pentru fiecare dintre utilizatorii care folosesc aplicația. Constructorii mapează diferite ipostaze ale folosirii acestui obiect. Primul constructor este utilizat în momentul în care se

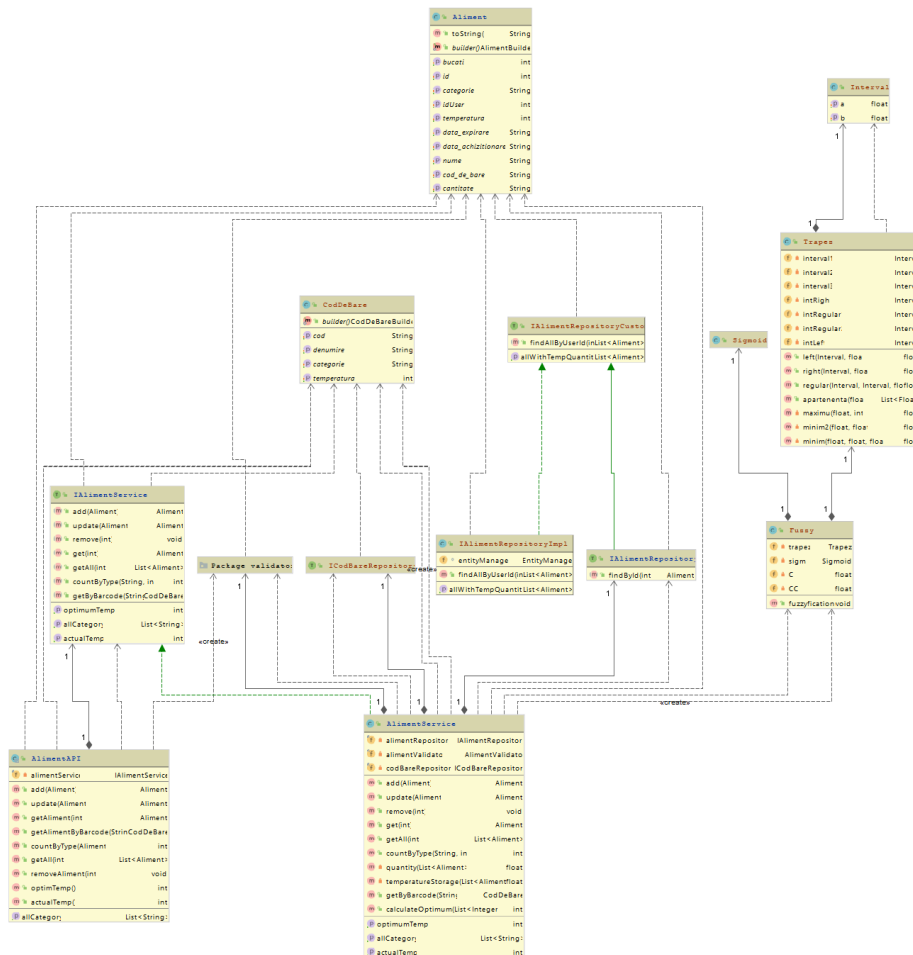


Figure 30: Diagrama de clase pentru gestionarea alimentelor

Diagrama de mai sus reproduce modalitatea prin care clasele și interfețele care gestionează alimentele sunt conectate. Clasa din partea de sus a imaginii aparține pachetului "domain" și reprezintă entitatea de bază a aplicației, iar câmpurile asociate acestei clase ilustrează caracteristicile necesare unui produs stocat în frigider. Acest obiect, în arhitectura stratificată, este folosit începând cu stratul care face conexiunea dintre aplicația de mobile și server până la stratul ce face conexiunea cu baza de date. Primul strat este reprezentat de clasa "AlimentAPI" împreună cu clasa "AlimentAPIException" care se află în pachetul validator și care prinde posibilele excepții. Următorul strat conține interfața "IAlimentService" și implementarea acestei interfețe "AlimentService". În cadrul acestui strat se validează și datele ce urmează să fie introduse în baza de date, iar efectuarea acestei etape se face folosind clasa "AlimentValidator". Un pas important care aparține acestui strat, este determinarea temperaturii optime care se face aplicând logica fuzzy. Operațiile efectuate se execută cu ajutorul clasei "Fuzzy", "Trapez", "Sigmoid" și "Interval", acestea purtând nume sugestive în funcție de scopul și de operațiile pe care le efectuează. Ultimul strat se ocupă de gestiunea datelor pe care le extrage din baza de date și conține interfața "IAlimentRepository" care extinde interfața "JPARepository" și interfața "IAlimentRepositoryCustom". După cum îi zice și numele, ultima interfața precizată, implementează operații mai aparte care nu se găsesc în repository-ul JPA.

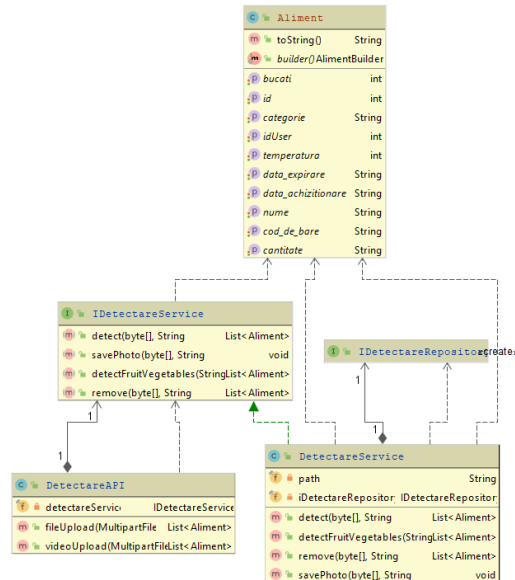


Figure 31: Diagrama de clase pentru gestionarea detectiilor

Am ales ca pentru partea de detectare a fructelor si a legumelor sa cream un pachet care respectă și în acest caz o structură stratificată a arhitecturii, astfel că apare și în această situație stratul prin care clientul comunică cu serverul, "DetectareAPI", stratul în care se execută operațiile logice, adică "IDetectareService" și "DetectareService", iar nu în cele din urmă stratul care cuprinde interfața "IDetectareRepository". Partea care ține de domeniul problemei conține și în acest caz clasa "Aliment".

## 5.3 Justificarea limbajului și a tehnologiei folosite

### 5.3.1 Front-end

În ceea ce privește folosirea limbajelor, am ales ca pentru partea de client să folosesc limbajul Kotlin. Acest limbaj, creat de JetBrains, este relativ nou și a avut o rampă de lansare deosebită în ultimii ani, mai ales din cauza faptului că este promovat de Google pentru crearea aplicațiilor bazate pe Android. Din ce în ce mai mulți programatori au început să folosească acest limbaj datorită popularității pe care o are, dar și din cauza similitudinilor pe care le are cu limbajul Java, care este limbajul de bază pentru programarea în Android. Evoluția a făcut ca în prezent, limbajul de programare Kotlin să poată fi utilizat în orice împrejurări în care este utilizat Java-ul: aplicații Android, desktop, server, consolă etc. dar și în crearea aplicațiilor care folosesc sistemul de operare iOS sau a aplicațiilor Web. Conform site-ului oficial dezvoltat de Kotlin, un limbaj de programare evolutiv constă în menținerea modernității pe care o are limbajul, în comunicarea eficientă cu utilizatorii și în ușurința prin care o nouă versiune se instalează, iar Kotlin-ul având toate aceste 3 elemente, se consideră ca fiind un limbaj modern și de aceea este recomandabilă utilizarea lui. Deși este considerat un limbaj nou, utilizarea acestuia nu este dificilă, iar modalitatea de implementare a codului este accesibilă. Accesibilitatea se datorează în mod deosebit compatibilității pe care o are cu limbajul Java deoarece un cod Java poate să fie convertit în cod Kotlin prin intermediul unor platforme sau chiar automat în Android Studio.

Implementarea codului pentru partea de mobil, am realizat-o în Android Studio care este un mediu de dezvoltare gratuit și accesibil atât pentru sistemul de operare folosit de mine: Windows, cât și pentru Mac respectiv Linux. Android Studio este bazat pe software-ul IntelliJ IDEA creat tot de JetBrains și este recunoscut ca fiind mediul principal de dezvoltare a aplicațiilor de mobil.

### 5.3.2 Back-end

Partea de server al aplicației este implementată folosind cod Java care este un limbaj orientat-obiect dezvoltat de către James Gosling și lansat în 1995. Acest limbaj reprezintă o evoluție a limbajului C/C++ deoarece deși împrumută unele concepte, partea inovatoare este dată de portabilitatea pe care o are, aceasta constând în faptul că un cod Java poate să fie rulat pe orice platformă. Un mare beneficiu dat de programarea în Java apare din cauza diferitelor instrumente utile care fac posibilă dezvoltarea unei aplicații de orice tip. Așadar, pentru comunicarea cu partea de client și pentru implementarea unor funcționalități am folosit Framework-ul Spring care asigură securitate dezvoltată, rapiditate, productivitate și flexibilitate deoarece permite utilizatorilor să-și dezvolte aplicația dorită mult mai ușor în limbajul Java. Spring este recomandat deoarece este cunoscut și folosit în diferite aplicații populare precum Amazon, Google, Microsoft etc., iar deoarece comunitatea de utilizatori este una mare, găsirea și lămurirea unor probleme în timpul dezvoltării unei aplicații se realizează cu atât mai ușor. Folosirea codului Java în combinație cu Spring aduce avantaje majore, iar rapiditatea implementării este unul dintre factorii principali deoarece folosirea adnotărilor puse la dispoziție, în combinație cu repository-ul "JpaRepository" face ca aria de scriere a codului sursă să se restrângă considerabil. În cazul în care repository-ul pus la dispoziție nu are anumite funcționalități pe care dezvoltatorul le dorește, este posibilă și implementarea unor noi funcții.

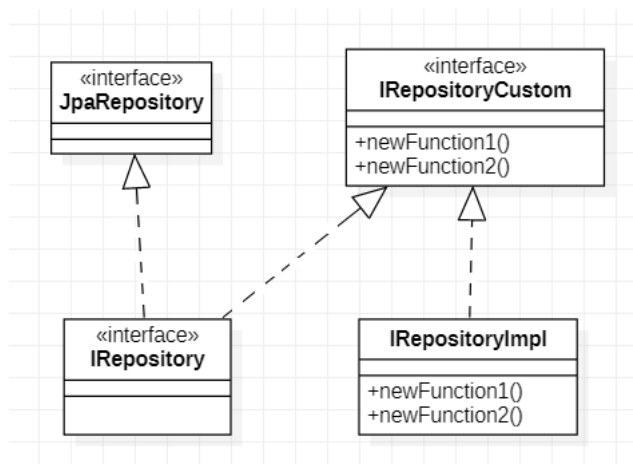


Figure 32: Implementarea unor funcționalități noi

Imaginea de mai sus ilustrează modalitatea de implementare a unor noi metode. "IRepository" este interfață principală și extinde două interfețe, una care este oferită de Spring și care poartă denumirea "JpaRepository",

iar cealaltă interfață conține antetele funcțiilor pe care dezvoltatorul le implementează în clasa "IRepositoryImpl" cu ajutorul clasei "EntityManager" care creează interogări asupra datelor stocate în baza de date. Denumirile claselor sunt importante în acest proces.

Obiectele folosite în program se pot conecta la entitățile din baza de date sau pot fi create noi entități folosind Java Persistence API cu ajutorul căruia, pentru fiecare câmp ce aparține obiectului, i se asociază o coloană din entitate. Asocierea realizată constă în oferirea unui nume și a unui set de caracteristici coloanei respective. Java Persistence API, prescurtat JPA, este o colecție de date care oferă diferite funcționalități pentru a spori interogarea asupra datelor stocate în baza de date. JPA oferă o abstractizare ridicată astfel încât toate informațiile din baza de date sunt stocate la nivel de obiect, programatorul nefiind obligat să cunoască alte detalii despre sistemul de gestiune a bazei de date.

Editorul folosit în implementare este IntelliJ IDEA. Acesta este un editor de calitate deoarece interacționează cu codatorul într-un mod excepțional, atenționându-l când acesta a comis o greșală în timpul scrierii codului și oferindu-i diferite modalități de reparare însoțite de explicații.

### 5.3.3 Baza de date

Pentru stocarea datelor am utilizat MySQL care este un sistem de gestiune al bazelor de date relațional, produs de compania suedeză MySQL AB și distribuit sub Licență Publică Generală GNU. Este cel mai popular SGBD open-source la ora actuală conform Wikipedia. Deși în majoritatea cazurilor MySQL este folosit împreună cu PHP, s-au dezvoltat pe parcursul anilor diferite instrumente care face ca comunicarea dintre limbaj și MySQL să fie ușor de realizat. Conform site-ului principal oferit de MySQL, acest sistem de gestionare a datelor este utilizat de cele mai mari companii precum Facebook, Yahoo, Twitter, YouTube și multe altele. Folosirea în profunzime a acestui sistem, face ca informarea și învățarea utilizării să fie mai ușoară și mai accesibilă. Interfața grafică pusă la dispoziție, care poartă denumirea de "MySQL Workbench", face posibilă crearea și vizualizarea bazelor de date. Pe de altă parte, este posibilă și inserarea și ștergerea manuală a datelor, dar și efectuarea acestor operații prin execuția unor interogări.

Din punct de vedere al limbajelor folosite pentru realizarea aplicației, acestea aduc un element de confort deoarece tehnologiile folosite sunt utilizate des în industrie, iar utilizarea lor aduc siguranța în menținerea sau dezvoltarea pe viitor a aplicației fără costuri exagerate.

## 5.4 Specificarea si testarea aplicatiei

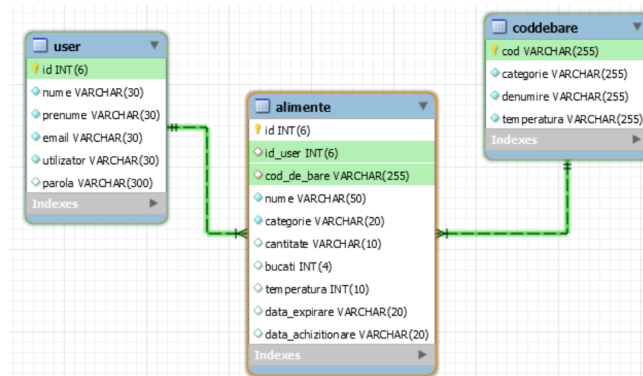
În ceea ce privește testarea aplicației, am folosit ca tehnică testarea bazată pe criteriul black-box. Avantajele testării black-box sunt date în mod aparte de independența implementării testelor față de modalitatea de implementare a aplicației. De asemenea această testare reflectă mai bine punctul de vedere al utilizatorului și surprinde ambiguitățile și inconsistențele din specificații, deoarece implementarea testelor se face astfel încât să surprindă datele clare ce constituie problema, dar și limitele acestor date. În interesul realizării testelor, s-au implementat tehnici de testare automată precum testarea independentă a unei funcționalități unde s-a testat procesul de autentificare, iar o altă tehnică a fost realizarea testării bazate pe scenarii de utilizare, caz în care s-a surprins modul în care un utilizator se autentifică, adaugă un produs completând câmpurile necesare, iar în final șterge produsul adăugat. Acest scenariu este încheiat doar dacă toți pașii au fost îndepliniți cu succes.

Implementarea testelor s-au realizat prin intermediul SDK-ului în Java oferit de platforma TestProject și mai exact cu ajutorul open-sorce-ului Selenium și a framework-ului Appium cu ajutorul căreia s-a rulat testul implementat pe dispozitivul mobil.

## 5.5 Descrierea formei de persistenta

Modalitate de gestionare a datelor stocate în baza de date este o sarcină importantă. Fiecare aliment adăugat conține informații necesare pe baza cărora se efectuează anumite evenimente precizate în secțiunile anterioare. Aceste informații sunt: numele produsului, categoria din care face parte, cantitatea și numărul de bucăți, temperatura necesară, data de achiziționare a produsului și data de expirare. Pentru ca un utilizator să utilizeze aplicația este necesar ca acesta să își creeze un cont de utilizator și să se autentifice. Crearea contului impune completarea anumitor date personale precum : numele, prenumele, emailul, numele de utilizator și o parolă. Deși pentru autentificare sunt necesare doar numele de utilizator și parola , numele și prenumele sunt folosite cu scopul de a personaliza aplicația din dorința de a face utilizatorul mult mai confortabil fiind dovedit științific că auzirea sau citirea propriului nume face utilizatorul mult mai comod și mai fericit. Emailul este folosit pentru momentul în care se dorește trimiterea produselor stocate unui prieten sau unui membru al familiei prin email astfel că odată stocat în momentul creării contului, nu mai este necesar completarea acestuia de fiecare dată când se dorește trimiterea email-ului dorit.

O parte care ajută utilizatorul în înregistrarea cât mai rapidă a produselor este dată de prezența codului de bare. Înregistrarea unui nou cod se poate



face de către orice utilizator în momentul actual deoarece baza de date nu pre-dispune de un număr mare de înregistrări. Am ales această metodă deoarece nu există un API care să predisună de o bază de date suficient de mare care să încorporeze produsele românești ce sunt achiziționate. Relația dintre diagrama care stochează informațiile despre utilizatori și diagrama de alimente și relația dintre diagrama care stochează informații despre produse pe baza codului de bare și diagrama de alimente este de "unu la mai mulți" deoarece un utilizator poate stoca mai multe produse, iar în diagrama de alimente se pot stoca mai multe produse cu același cod de bare.

## 5.6 UX si manual de utilizare al aplicatiei

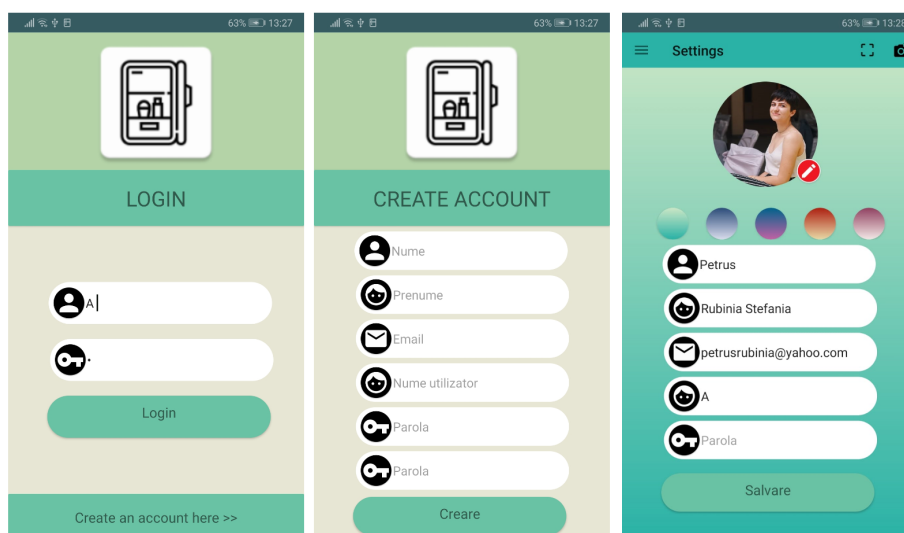


Figure 33: Gestiune cont

Aplicația este dezvoltată astfel încât utilizatorul să aibă o interacțiune cât mai prielnică și mai favorabilă cu aplicația.

Gestionarea contului de utilizator se face utilizând ferestrele prezentate în figura de mai sus. În partea din stânga a imaginii este prezentată fereastra de autentificare urmând ca în partea din mijloc să se prezinte modalitatea de creare a unui cont nou. Fereastra de setări, ilustrată în partea din dreapta, face posibilă personalizarea aplicației prin aplicarea unei palete de culori favorabile utilizatorului, acesta putând alege de la o paletă de culori cu o nuanță mai deschisă, la o paletă de culori în care predomină culorile închise. De asemenea, în cadrul acestei ferestre se poate face și modificarea contului de utilizator.



Figure 34:  
Meniu

În prima fereastră din figura de mai sus apare pagina principală ca urmare a autentificării. Aceasta conține o bară de activități în partea de sus a ecranului cu ajutorul căreia se poate deschide camera(2) pentru a se efectua fotografia dorită, se poate deschide de altfel și cititorul de bare(1) sau se poate accesa un meniu secundar(3) al aplicației cu care utilizatorul poate naviga spre pagina de setări, spre pagina unde se poate trimite prin email un raport a produselor stocate în frigider, spre pagina de informații în care se va găsi un tabel cu o anumită gamă de produse și cu temperaturile pe care le necesită și în ultimul rând, utilizatorul poate accesa din cadrul acestui meniu, butonul de delogare. În partea de jos a acestei ferestre se afla meniul principal pe care utilizatorul îl va accesa cel mai des și conține un buton

prin care se poate accesa fereastra de adăugare prin completarea diferitelor caracteristici a unui produs, un buton care trimite utilizatorul spre partea de gestionare a produselor, un buton cu ajutorul căreia se deschide fereastra unde se pot vedea produsele ce se apropie de expirare și un buton care să navigheze spre fereastra de temperaturi. Ultimele două ferestre se află chiar în această figură.

În fereastra în care se listează produsele cele mai apropiate de expirare, repartiția acestora se face pe luni, iar vizualizarea unei liste de produse se face accesând una dintre barele destinate fiecărei luni. În exemplul afișat produsele expiră în luna a 6-a.

În ultima fereastră se ilustrează temperatura actuală a frigiderului și temperatura optimă calculată pe baza produselor stocate.

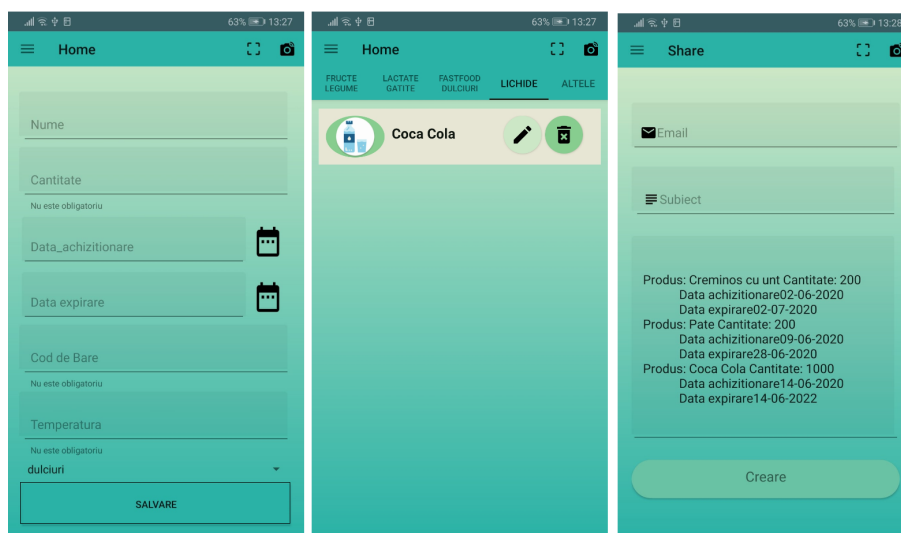


Figure 35: Gestionare produse

Pentru adăugarea produselor preparate în bucătărie și pentru adăugarea produselor folosind codul de bare se vor completa anumite câmpuri din cadrul ferestrei prezentate în figura alăturată (în partea stângă). Posibilitatea de editare și de ștergere, cât și posibilitatea de a vizualiza produsele sectionate pe categorii, se face în fereastra a 2-a din figura de mai sus, iar distribuirea datelor stocate în frigider unui membru a familiei sau unui prieten se face completând câmpurile din ultima fereastră. Trimiterea unui email de această natură poate fi considerată un beneficiu în momentul în care altcineva dorește să facă o listă de cumpărături și vrea să fie informat în privința produselor deja existente în frigider.

## 6 Concluzii si dezvoltări ulterioare

Realizarea acestei aplicații introduce în domeniul frigiderelor inteligente o modalitate mai deosebită de reducere a consumului de energie electrică aplicând logica fuzzy. Pornind de la ideea de a determina temperatura optimă, am introdus în aplicația dezvoltată, algoritmi din domeniul inteligenței artificiale care să detecteze pe baza imaginilor și pe baza videoclipurilor produse ce se depozitează în viața de zi cu zi în interiorul frigiderului. Utilizarea rețelelor neuronale ajută utilizatorii, ușurând timpul depus în procesul de stocare virtuală a produselor. Deși numărul de produse ce pot fi detectate este unul relativ mic față de produsele existente pe piață, se poate mări în timp numărul lor antrenând rețeaua neuronală actuală. În privința produselor ce nu se pot detecta se oferă prezența cititorului de bare cu ajutorul căreia se primesc informațiile necesare.

O posibilă dezvoltare pe viitor constă în extinderea setului de date destinat detecției fructelor și legumelor. O altă posibilitate de dezvoltare are legătură cu o mai bună gestionare a produselor, făcându-se posibilă realizarea unor rețete culinare pe baza produselor din interiorul frigiderului.

O ultimă idee de posibilă dezvoltare are o valoare mai importantă și ține de modalitatea de setare automată a temperaturii frigiderului. Acest pas, deși aduce probleme din cauza faptului că frigiderele au un sistem diferit de funcționare, este posibil de realizat introducerea unui driver care să vizeze cel puțin o anumită marcă de electrocasnice și de ce nu, să se extindă captivând cât mai multe firme producătoare.

## 7 Bibliografie

### References

- [1] MS Alissi. The effect of ambient temperature, ambient humidity and door openings on household refrigerator energy consumption. *MSME thesis, Purdue University, Indiana*, 1987.
- [2] Jasmin Geppert and Rainer Stamminger. Analysis of effecting factors on domestic refrigerators' energy consumption in use. *Energy conversion and management*, 76:794–800, 2013.
- [3] Md Hasanuzzaman, Rahman Saidur, and Haji Hassan Masjuki. Effects of operating variables on heat transfer and energy consumption of a household refrigerator-freezer during closed door operation. *Energy*, 34(2):196–198, 2009.
- [4] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.
- [5] Eman Khalid Justaniah, Ibtehaj Mohammad Audah Al-zhrani, Elaf Jameel Siraj Islam, and Nafisah Matouk Mohammad Khashaifaty. System and process for monitoring the quality of food in a refrigerator, July 14 2016. US Patent App. 14/593,837.
- [6] Hokuto Kagaya, Kiyoharu Aizawa, and Makoto Ogawa. Food detection and recognition using convolutional neural network. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 1085–1088, 2014.
- [7] Md Imran Hossen Khan and HM Afroz. An experimental investigation of door opening effect on household refrigerator; the perspective in bangladesh. *Asian Journal of Applied Sciences*, 7(2):79–87, 2014.
- [8] Se Il Kim, Su Ho Jo, Jae Hoon Cha, Na Jeong Han, and Kwan Joo Myoung. Refrigerator and mobile terminal for food management, October 9 2014. US Patent App. 14/248,621.
- [9] Hisanori Kiyomatsu. Refrigerator with a function to confirm items stored therein, June 6 2002. US Patent App. 09/992,218.

- [10] Jungho Lee, Jeongsuk Yoon, and Seungmin Baek. Method for managing storage product in refrigerator using image recognition, and refrigerator for same, March 24 2016. US Patent App. 14/783,612.
- [11] Alphan Manas, Kerem Guvenc, and Murat Armagan. Refrigerator capable of stock monitoring by providing ideal storage conditions, November 12 2009. US Patent App. 11/721,283.
- [12] Maryam Rahnemoonfar and Clay Sheppard. Deep count: fruit counting based on deep simulated learning. *Sensors*, 17(4):905, 2017.
- [13] Joel Rozendaal and John Francis Broker. Refrigerator including food product management system, January 4 2011. US Patent 7,861,542.
- [14] Inkyu Sa, Zongyuan Ge, Feras Dayoub, Ben Upcroft, Tristan Perez, and Chris McCool. Deepfruits: A fruit detection system using deep neural networks. *Sensors*, 16(8):1222, 2016.
- [15] Rahman Saidur, Haji Hassan Masjuki, and IA Choudhury. Role of ambient temperature, door opening, thermostat setting position and their combined effect on refrigerator-freezer energy consumption. *Energy Conversion and Management*, 43(6):845–854, 2002.
- [16] Jing Tao and Suiran Yu. Implementation of energy efficiency standards of household refrigerator/freezer in china: Potential environmental and economic impacts. *Applied Energy*, 88(5):1890–1905, 2011.
- [17] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. <https://github.com/facebookresearch/detectron2>, 2019.
- [18] Selin Yilmaz, Dasa Majcen, M Heidari, Jasmin Mahmoodi, Tobias Brosch, and Martin Kumar Patel. Analysis of the impact of energy efficiency labelling and potential changes on electricity demand reduction of white goods using a stock model: The case of switzerland. *Applied Energy*, 239:117–132, 2019.
- [19] Xiang Zhang. Simple understanding of mask rcnn.