UNIVERSITATEA ALEXANDRU IOAN CUZA IAŞI

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENŢĂ

Aplicație Android pentru gestionarea rețetelor culinare Robo Bucatarul

propusă de

Samuel-Emanuel Bocicu

Sesiunea: *februarie, 2020*

Coordonator ştiinţific

Lect. Dr. Cristian Frăsinaru

UNIVERSITATEA ALEXANDRU IOAN CUZA IAŞI

FACULTATEA DE INFORMATICĂ

Aplicație Android pentru gestionarea rețetelor culinare Robo Bucatarul

Samuel-Emanuel Bocicu

Sesiunea: *februarie, 2020*

Coordonator ştiinţific

Lect. Dr. Cristian Frăsinaru

DECLARAŢIE PRIVIND ORIGINALITATE ŞI RESPECTAREA DREPTURILOR DE AUTOR

Prin prezenta declar că Lucrarea de licenţă cu titlul "*Aplicatie Android pentru gestionarea retetelor culinare Robo Bucataru*" este scrisă de mine şi nu a mai fost prezentată niciodată la o altă facultate sau instituţie de învăţământ superior din ţară sau străinătate. De asemenea, declar că toate sursele utilizate, inclusiv cele preluate de pe Internet, sunt indicate în lucrare, cu respectarea regulilor de evitare a plagiatului:

- toate fragmentele de text reproduse exact, chiar şi în traducere proprie din altă limbă, sunt scrise între ghilimele şi deţin referinţa precisă a sursei;
- reformularea în cuvinte proprii a textelor scrise de către alţi autori deţine referinţa precisă;
- codul sursă, imagini etc. preluate din proiecte *open-source* sau alte surse sunt utilizate cu respectarea drepturilor de autor şi deţin referinţe precise;
- rezumarea ideilor altor autori precizează referinţa precisă la textul original.

Iaşi, *data*

Absolvent: *Samuel-Emanuel Bocicu*

_____

(semnătura în original)

Cuprins

Introducere....6
Motivație....6
Obiectiv....7
Specificațiile funcționale....8
Tehnologii utilizate....9
Android....9
Prezentare Generală a sistemului de operare. 9
Arhitectura sistemului de operare Android....12
Kernel-ul Linux....13
Layer-ul de abstractizare hardware (HAL)....13
Android Runtime....13
Native C/C++ Libraries....14
Framework-ul Java API....14
Android Jetpack....10
Android 10....11
Sistemul de aplicatii....13
Google Firebase....13
Firebase authentication....19
Firebase real-time data base....19
Google analytics....19
Retrofit....20
Picasso....20
PocketSphinx....20
Arhitectura Aplicatiei....21
Autentificare....23
Monitorizare aplicatie....24
Stocarea datelor utilizatorilor....20
Probleme întâmpinate pe parcursul dezvoltării aplicației....28

Eliminarea sunetelor de notificare a recunoașterii vocale 28
Implementarea unui sistem de recomandare în lista de cumpărături 28
Numarul limtat de stocare a caracterelor in QR-coduri. 30
Detaliile aplicației 31
Interfața grafică 31
Numele și logo-ul aplicației 31
Design-ul meniului de navigare 32
Opțiunile disponibile de navigare 33
Detaliile de implementare 38
Speech Recognizer 38
Voice Commander 39
Procesul de cautare produselor din lista de cumparaturi 41
Concluzii 42
Bibliografie 43

Introducere

Simplificarea diferitelor activități din viața de zi cu zi a fost și va fi mereu o problemă pentru societatea noastră. Însă prin rezolvarea acestor probleme se va contribui la evoluția mediului în care trăim.

Înțelegând acest lucru am ales să dezvolt o aplicație mobile ce propune soluții pentru diferite problemele întâlnite în activitatea culinară. Din problemele existente în acest domeniu am ales să studiez și să construiesc o soluție pentru procesul de învățare a rețetelor, procesul de verificare a unei liste de cumpărături și procesul de expunere publică a rețetelor

Motivație

Mereu am fost o persoană fascinată în a învăța lucruri noi, mai ales acele lucruri practice care având răbdarea necesară, efectuând pas cu pas instrucțiunile date, să ajungi într-un final să te bucuri de ceea ce ai creat sau ai reușit să rezolvi.

Partea cea mai importantă în activitățile de genul aceste este chiar efectuarea cu strictețe a pașilor care îți sunt sugerați pentru a finaliza cu succes acea procedură. Personal îmi place să petrec timp în bucătărie, unde datorită capacităților mele de începător în acest domeniu, timpul pe care îl petrec aici e pentru a învăța rețete noi, și în general acestea sunt structurate pe pași.

În cele mai multe cazuri am fost pus în situația în care să am ambele mâini ocupate dar trebuia cumva să îmi controlez telefonul pentru află următorul pas.

O altă problemă ce derivă din contextul prezentată mai sus este aceea a cumpărăturilor produselor pentru gătit. Nu de puține ori mi s-a întâmplat să merge pe la 2 sau chiar 3 magazine până să găsesc toate produsele necesare pentru prepararea unei rețete. E adevărat supermarketurile încearcă să îți ofere o gamă cât mai variată de produse astfel încât să nu existe problema aceasta. Chiar și în condițiile acestea mereu este bine să ai și alte variante în caz că produsul nu mai este pe stoc într-un anumit magazin.

Pasiunea pentru gastronomie a început după o lună de zile petrecută într-o bucătărie a unui restaurant în care am avut de învățat mai multe rețete. Fiind neexperimentat mereu trebuia să merg la frigider unde erau lipiți pașii pe care îi aveam de urmat în vederea preparării diferitelor mâncăruri. Când ești angajat într-o bucătărie nu îți permiți să mai pierzi așa mult timp în efectuarea rețetelor, timpul fiind prețios într-un astfel de context.

Motivat fiind de aceste experiențe am căutat să creez un cadru mai benefic persoanelor neexperimentate în acest domeniu, ajutându-i să parcurgă diferite rețete într-un mod eficient și plăcut. Astfel proiecția acestei aplicații are la bază probleme cotidiene și este direcționată în a aduce un plus în această zonă a bucătăriei.

Obiectiv

Obiectivul principal este de a realiza o aplicație pe platforma de dezvoltare Android ce va ușura procesul de învățare în domeniul gastronomic. Aceasta va asista utilizatorul pe toată durata procesului de efectuare a unei rețete punând la dispoziție conținutul fiecărui pas atât printr-o sursă vizuală cât si una auditivă.

Prin această aplicație utilizatorul va putea de asemenea să își adauge o listă de cumpărături pentru fiecare rețetă creată. La introducerea produselor în lista de cumpărături, utilizatorului i se va recomanda produse, similare cu cel introdus, ce pot fi găsite la magazinele din apropiere. Mai mult de atât, după introducerea produselor in lista de cumpărături, va exista si un proces de verificare a disponibilității.

Un alt obiectiv, atins doar partea utilizatorului de aplicație mobile, este de a adăuga rețete într-un mod mult mai rapid decât o introducere manuală a fiecărui pas. Un cititor de QR-uri capabil să recunoască qr-codurile create specific pentru stocarea acestor rețete. Partea ce lipsește la această soluție este aceea de a crea o aplicație web ce permite transformarea unei rețete în qr-code si de a avea un management al acestora.

Specificațiile funcționale

Pentru a putea folosi această aplicație, utilizator după ce a descărcat-o va trebui să își creeze un cont în aplicație cu o adresă de email, acest pas fiind indispensabil în folosirea acesteia.

După ce se va autentifica cu acest cont, aplicația îi va permite să își creeze noi rețete care vor fi salvate într-o bază de date din cloud, hostată de Firebase. Chiar dacă utilizatorul își va dezinstala aplicația, întreaga lui biblioteca va rămâne salvată.

Prin platforma Firebase asigurăm și cazul în care utilizatorul va uita vreodată parola contului creat, prin funcționalitatea de recuperare a parolei, prin trimiterea unui email cu resetarea acesteia.

Aplicația va oferi utilizatorului funcționalitatea de a crearea noi rețete sau de a le șterge. Prin folosirea funcționalității de scanare QR se va putea adăuga adăugă noi rețete doar scanând un QR code.

Fiecărei rețetă va avea posibilitatea de a i se adaugă și o listă de cumpărături, lista putând fi trimisă și printr-un sms către oricare dintre contactele de pe acel telefon. Mai mult de atât pe o listă de cumpărături asociata unei rețete are posibilitatea de a verifica ce magazin din apropiere au acele produse din lista.

Din structura aplicației nu lipește un meniu accesibil din fiecare componentă a acesteia, oferind astfel utilizatorului acces rapid către orice funcționalitate.

Tehnologii utilizate

În proiectul pe care l-am dezvoltat pe platforma Android am folosit mai multe tehnologii, funcționalități și servicii ce au facilitate evoluția aplicației, aducând un plus de eficiență și claritate în ceea ce privește codul sursă. Pentru început voi prezenta sistemul de operare Android, detaliind arhitectura acestuia și modul cum a evoluat pe piață.

Android

Prezentare Generala a sistemului de operare.

Sistemul de operare Android este o versiune a sistemului Linux adaptat dispozitivelor mobile, inițiată de compania start-up Android Inc. care mai apoi a fost cumpărat de cei de la Google, ca în anul 2007 să fondeze Open Handset Alliance.

Această alianță grupează mai multe companii ce vor contribuii la obținerea a noi produse pe piață ce vor avea la bază acest sistem de operare. Scoaterea la vânzare a primul dispozitiv ce folosea Android să realizat în anul 2008 iar în din perioada 2011-2013 ajunge în top-ul celor mai vândute sisteme de operare pentru dispozitivele mobile din lume.

Aplicațiile suportate de pe astfel de dispozitive sunt scrise în limbaje precum Java sau Kotlin, acestea fiind cele la care se oferă suport în mod oficial. Aplicațiile mai pot fi scrise și în alte limbaje de programare ce vor fi compilate cu ajutorul mașinii ARM. Ajungând să fie un software open source, Android deține o pagină web cu acces gratuit special dedicată documentației în vederea dezvoltării aplicațiilor mobile.

Platforma oferă posibilitatea adaptării la configurații mai mari, librării grafice 2D, VGA librari grafice 3D având la bază specificația OpenGL ES și setări tradiționale smartphone. Stocare datelor se realizează cu ajutorul librăriei SQLite, care un software de baze de date. Suportul pentru conectivitate în Android include compatibilitate cu GSM/EDGE, CDMA, EV-DO, UMTS, Bluetooth și Wi-Fi. Tipurile de mesagerie instantă suportată de acest sistem de operare sunt SMS

și MMS. Navigatorul web pus la dispoziție de Android se bazează pe WebKit, o platformă de aplicații open source.

Dispozitivele mobile cu astfel de sisteme de operare au implementate mașini virtuale Dalvik ce pot executa software compilat în cod mașina Dalvik. Formatele media acceptate de Android pentru audio, video, și imagine sunt: MPEG-4, MP3, H.264, OGG, AAC, AMR, PNG, JPEG, GIF.

Android pune la dispoziție un kit de dezvoltare software asigurând în totalitate acest proces. Kit-ul conține un program de depanare, un emulator de dispozitiv, biblioteci, documentație și exemple de cod. Mediul de lucru de oficial pentru dezvoltarea de aplicații Android este Android Studio. Acesta are suport de construcție bazat pe Gradle având mai multe caracteristic, cum ar fi editorul de layout vizual, analizor APK, sistem de construcție flexibilă, editor de cod inteligent, profiluri în timp real. Android Studio acceptă SDK și NDK pentru dezvoltarea aplicațiilor Native. Acest IDE suportă limbile Java, C ++ și Kotlin. În acest IDE oferă statistici în timp real pentru memorie., activitatea rețelei.

Lista versiunilor de operare Android începe odată cu lansarea în mod oficială a Androidului în data de 5 noiembrie 2007, iar pe piață apare Android 1.0 în septembrie 2008. Numai versiunile 1.0 și 1.1 nu i s-au asignat nume specifice, celelalte având nume de produse din cofetărie în ordine alfabetică. Ultima versiune de Android apărută la ora actuală este 9.0 lansată pe 9 august 2018 sub numele de cod Pie ce permite sistemului să se adapteze în funcție de fiecare utilizator în parte, folosind inteligența artificială reușește să anticipeze modul de lucru, putând astfel să aducă îmbunătățiri la consumul de baterie și o experiență mult mai plăcută pe acest sistem de operare.

În figura 1. este prezentată evoluția sistemului de operare Android modul în care s-a trecut de la o versiune la alta și cum a decurs această trecere la nivel global. Putem observa faptul că din anul 2010 Androidul începe să construiască versiuni stabile, reușind să își construiască o traiectorie clară în ceea ce privește versiunile sistemului.

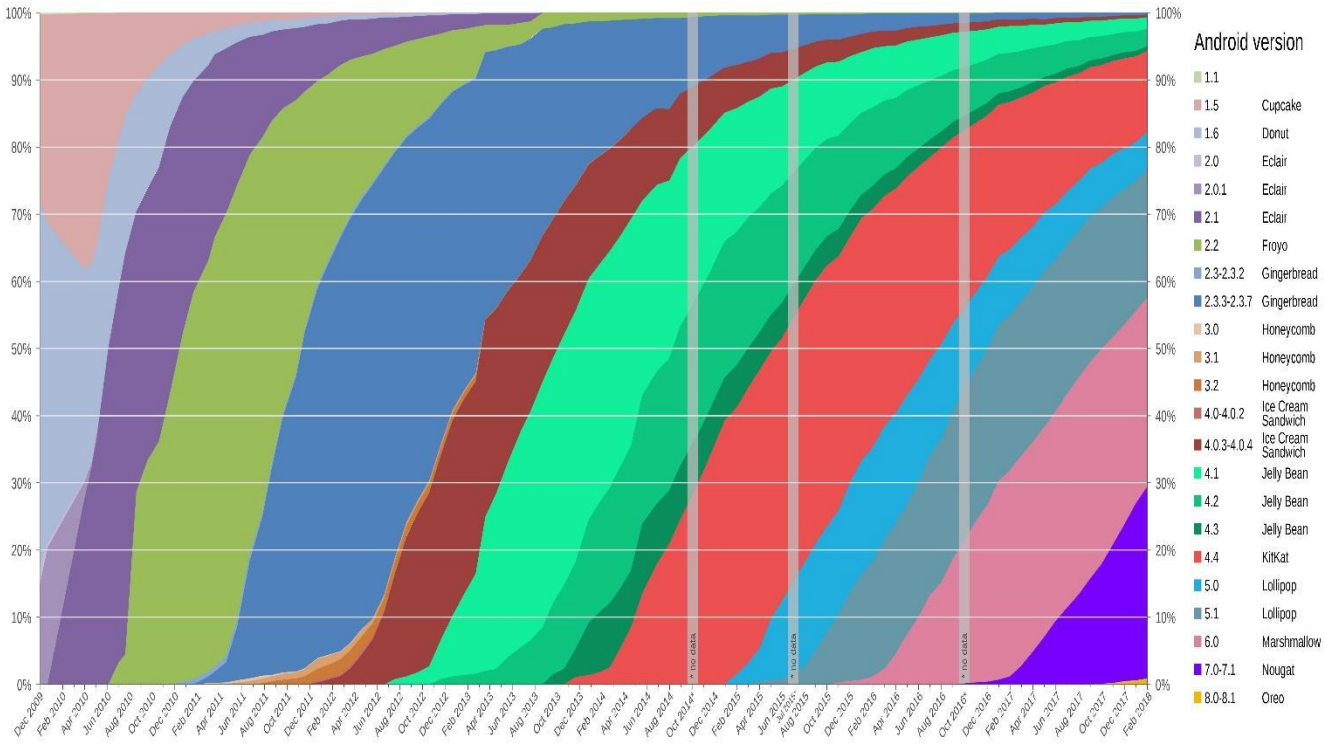

Figura 1. Distribuția globală a versiunii Android din decembrie 2009 pana in februarie 2018

Se poate observa că versiunea 4.4 KitKat a fost cea mai lungă durată în vârful versiunilor Android iar în februarie 2018 având un procentaj de peste 25%. De asemenea versiune Gingerbread at avut un impact major în evoluția sistemul de operare Android, observându-se o ușoară înclinare a diagramei după această versiune.

Sistemul de operare Android a avut și versiuni ce putem spune că au „eșuat” pe piață în ideea în care acestea au trebuie trecute la o altă versiune la o distanță relativ mică de timp de la data lansării nereușind să fie asimilată de către utilizatori. Honeycomb este una dintre ele care după un succes cu Gingerbread, aceasta nu reușește să intre pe piață așa cum au făcut celelalte versiuni, reușind să ajungă doar la 50% dintre utilizatori pentru o perioadă foarte scurtă de timp fiind influențată și de faptul că această versiune a fost creată cu precădere pentru dispozitivele android cu ecranul mai mare decât cel normal, în special pentru tablete.

Arhitectura sistemului de operare Android

Android fiind un sistem open source, software de tip stivă, bazată pe Linux, creată pentru o gamă largă de dispozitive și factori de formă. Următoarea diagramă prezintă principalele componente ale platformei Android.

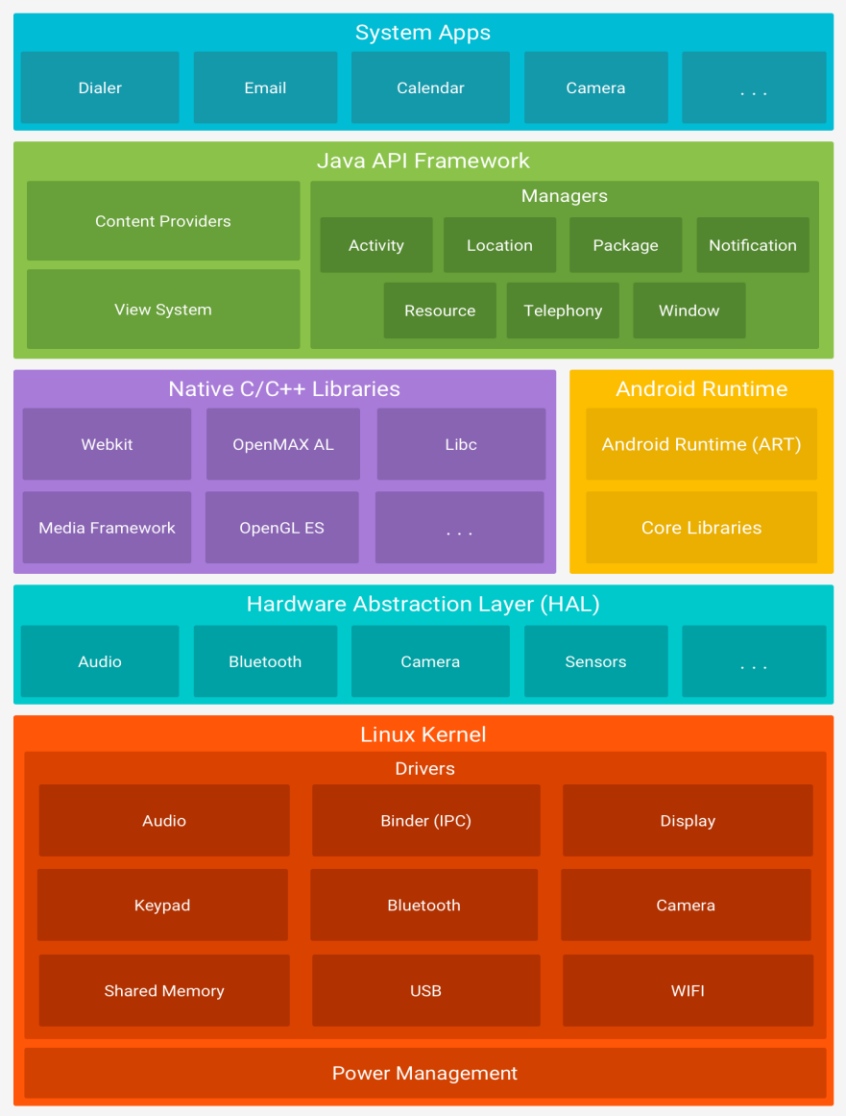

figura 2. Stivă de software Android

Kernel-ul Linux

Platforma android se bazează pe kernel-ul Linux, de exemplu Runtime-ul Android se bazează pe kernel-ul Linux pentru funcționalitățile de bază cum ar fi managementul memoriei de nivel scăzut și executarea firelor de execuție. Folosirea acestui kernel, permite Android-ului să se folosească de caracteristicile cheie de securitate, oferind posibilitatea producătorilor de dispozitive să dezvolte componente hardware pentru un kernel foarte cunoscut.

Layer-ul de abstractizare hardware (HAL)

Layer-ul de abstractizare hardware oferă interfețe standardizate putând astfel să expună funcționalități hardware pentru nivelul superior Java API framework. HAL este alcătuit din mai multe librari de tip modul, fiecare în parte expunând o interfață pentru un anumit tip de componenta hardware, cum ar fi gps-ul sau modulul wi-fi. În momentul în care framework-ul API dorește să acceseze o anumită componentă hardware, sistemul Android încarcă librăriile de tip modul pentru acea componentă hardware.

Android Runtime

Începând cu dispozitivele ce rulează versiunea Android 5.0 (API 21) sau mai mare, având propria instanță de Runtime Android (ART), rulând în propiul proces. Scopul ART-uli este pentru a oferi dispozitivelor cu memorie redusă de a rula mai multe mașini virtuale prin executarea de fișierelor DEX (executabile în format Davlik), un format bytecode special conceput pentru Android.

Conceptele pe care se bazează ART sunt următoarele: Ahead-of-time (AOT) și just-in-time (JIT) îmbunătățește performanța unei aplicații prin managementul aplicațiilor compilate, garbage collection optimizat aduce beneficii prin eliberarea de memorie ce nu mai este folosită, începând

cu versiunea de Android 9 permite conversia pachetelor unei aplicații cu fișiere DEX în cod mașina mult mai compact.

Native C/C++ Libraries

Deoarece majoritatea componentelor de bază ale Sistemului Android, cum ar fi ART și HAL au la bază în construcția lor librării native scrise în C și C++ platforma de dezvoltare Android pune la dispoziție API-uri Java pentru expun și funcționalitatea anumitor biblioteci native ale aplicației. OpenGL Java este un framework prin care putem accesa OpenGL ES pentru a putea avea suport în desenarea și manipularea graficelor 2D și 3D în dezvoltarea unei aplicații. Pentru dezvoltarea unei aplicații care necesită cod în C sau C++, este pus la dispoziție NDK-ul Android ce permite accesul la unele librării native direct din codul nativ.

Framework-ul Java API

Funcționalitatea sistemului de operare Android este disponibil prin API-uri scrise în limbajul Java, construind moduluri care vor fi folosite la dezvoltarea aplicațiilor Android asigurând simplitatea reutilizării componentelor și serviciilor de bază.

Aceste sunt structurate în 5 mari categorii, prima fiind View System, care aduce funcționalități în parte de interfață ajutând dezvoltatorul să își organizeze partea grafică a aplicație. A doua grupă este Mangerul de resurse ce permite manipularea resurselor ce nu sunt sub formă de linii de cod beneficiind de diferite modificări in funcție de contextul in care se afla acesta. Cea de-a treia este Managerul de notificări ce permite un control personalizat a alertelor permițând ca fiecare aplicație să folosească acest conținut. A patra categorie este Managerul de activități punând la dispoziție un management complex al activităților unei aplicații, oferind și o stivă de activități. Ultima parte este legată de furnizorii de conținut, această parte oferă posibilitatea de a accesa date ale altor aplicații sau să expună spre exterior datele printr-un anumit contract pe care trebuie împlinit.

Android Jetpack

Android Jetpack este un set de librării ce sunt recomandate de către cei de la Google pentru a construi aplicații cu un nivel ridicat al performanței. Prin aceste recomandări se dorește eliminarea unei arhitecturi complicate, greu de întreținut și greu de folosit astfel încât aplicațiile dezvoltate respectând acest standard să fie de înaltă calitate și robuste.

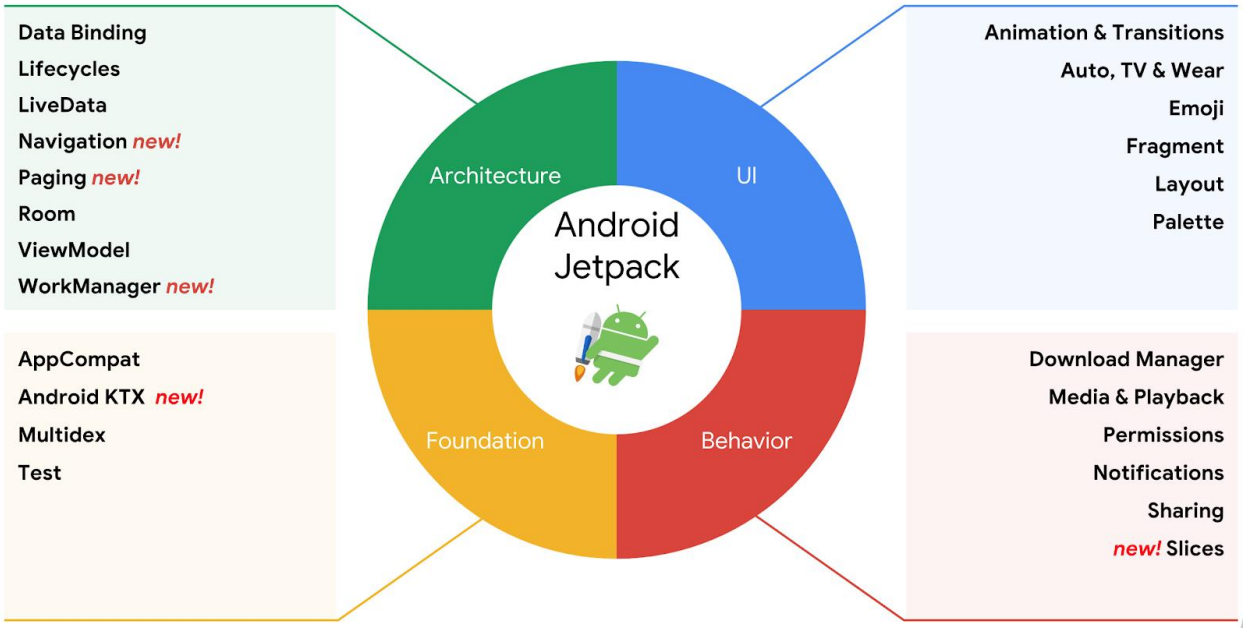

figura 3. arhitectura librăriei jetpack

Cele mai noi îmbunătățiri aduse odată cu ultima versiune lansată a acestei librari sunt Android KTX și Slices. Primul fiind un tool de accelerare a dezvoltării aplicațiilor venind cu o serie de implementări standard pentru diferite api-uri ale sistemului de operare. Iar Slices constituie un șablon de interfață a utilizatorului care poate afișa conținut într-un mod complex, interactiv și dinamic.

Android 10

Este ultima versiune de sistem de operare lansată de către platforma de dezvoltare Android pe data de 3 septembrie 2019 fiind a 17 versiune de Android. Această versiune vine cu o multitudine de noutăți și îmbunătățiri atât în ce privește interfața utilizatorului, performanță cât și un upgrade în ceea ce privește suportarea 5G-ului.

În zona de interfață grafică sistemul de operare android 10 vine cu o nouă temă grafică și anume Dark theme, aceasta economisind bateria telefonului. Tot la această categorie noutățile apar și în zona gesturilor, permițând navigare înainte și înapoi prin glisare la dreapta respectiv stânga și prin glisarea în sus pornind de la baza telefonului pentru a deschide Google Assistant.

figura 4. tema Dark in versiunea Android 10

În ceea ce privește funcționalități noi, această versiune de sistem de operare vine cu diferite concepte ale inteligenței artificiale cum ar fi Smart Replay care are capacitatea de a sugera utilizatorului atât un răspuns la un mesaj sau email cât și acțiuni pe acel dispozitiv deduse din conținutul primit. În același sens de idei o noutate o reprezintă și subtitrarea directă a video-urilor fără a fi nevoie nici măcar de internet. Aceasta este numită Live Caption.

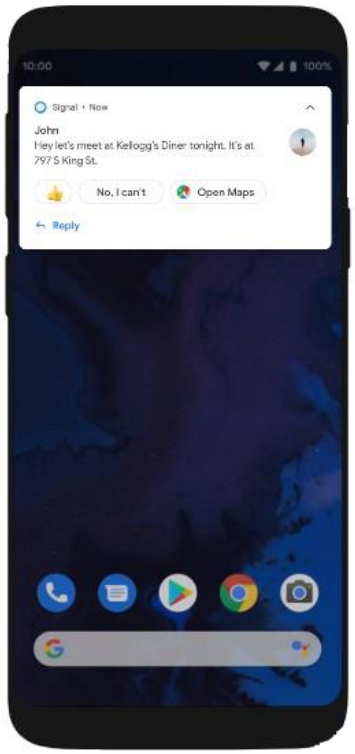

figura 5. smart replay

figura 6. live caption

Pentru performanță, securitate și intimitate, nouă versiune vine cu îmbunătățiri, extinderi și noi trenduri încercând nu doar să țină pasul cu evoluția sistemelor mobile, chiar mai mult de atât, aducând și concepte noi.

Sistemul de aplicații

În ceea ce privește aplicațiile de bază Android asigură e-mail, mesaje SMS, calendare, navigare pe Internet, contacte etc. Aplicațiile ce vine odată cu sistemul de operare nu au prioritate față de aplicațiile pe care utilizatorul alege să le instaleze. Astfel orice aplicație ce nu vine odată cu sistemul de operare poate deveni aplicație implicită pentru diferite acțiuni în sistem.

Google Firebase

Platforma Firebase este un BaaS (Backend-as-a-Service) care a început ca un start-up în 2011, apoi fiind preluată de cei de la Google în 2014 ajung să ofere servicii cloud. Această oferă funcții de analiza, baze de date, mesagerie și rapoarte de erori construind astfel un management prietenos și ușor de utilizat, abstractizând componentele ce îngreunează dezvoltarea unei aplicații.

O ușoară comunicare între serviciile oferite de Firebase face ca aceasta mediu de lucru să fie utilizat de tot mai mulți utilizatori. Aceasta oferind suport pentru aplicațiile de pe platformele de dezvoltare Android, iOS, web, C++ și Unity.

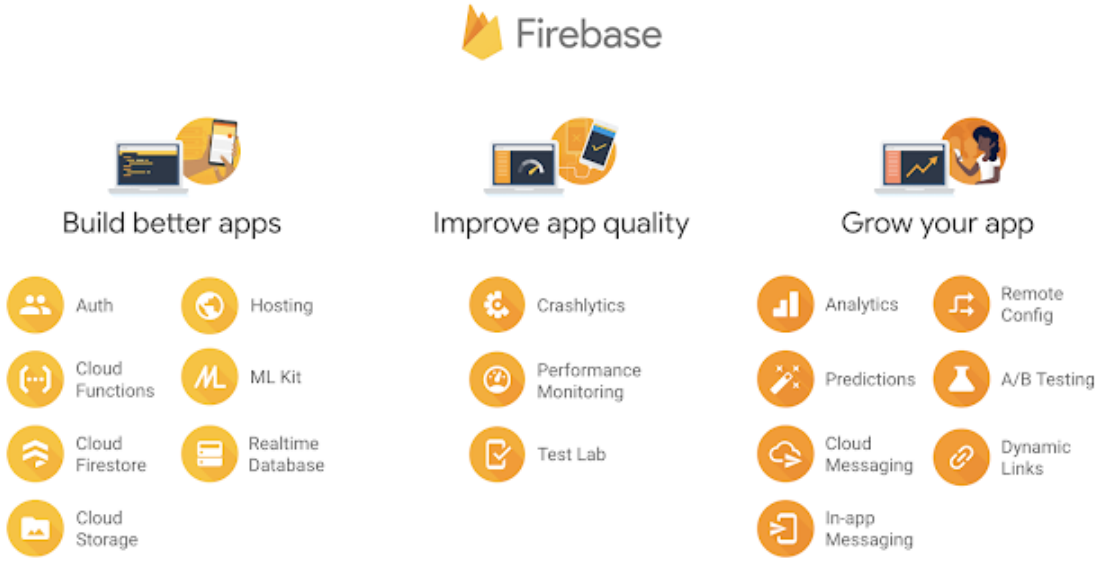

figura 7. serviciile oferite de platforma Firebase

Firebase authentication

Firebase Authentification oferă servicii backend, SDK-uri ușor de utilizat și librării predefinite UI pentru a le utiliza în autentificarea utilizatorului în aplicație. Acesta oferă suport pentru autentificare folosind email, număr de telefon sau diferiți furnizori de identitate federalizați, cum ar fi Google, Facebook, Twitter, Linkedin etc.

Având suport și pentru platforma de dezvoltare android, tot mecanismul de management al utilizatorilor este realizat prin callback-uri ce sunt implementate în interiorul aplicației. Configurările în ce privește complexitatea parolei se realizează din platforma celor de la Firebase.

Pe toată durata unei sesiuni a aplicației, instanța folosită la o autentificare cu succes va putea fi folosit oriunde după aceea în aplicație, astfel orice acțiune în interiorul aplicației este condiționată de această autentificare.

Firebase real-time data base

Bază de date Firebase Realtime fiind găzduită în cloud, este o bază de date ce stochează datele ca JSON și sincronizându-le în timp real cu fiecare client conectat. Pune la dispoziție suport pentru aplicațiile cross-platform cu propriile kituri SDK, astfel utilizatorii vor partaja o singruă instanță a bazei de date, actualizându-se automat, asigurând consistența datelor.

Google analytics

Serviciul Google analytics este o soluție gratuită ce permite o monitorizare aplicației astfel încât să ofere capacitatea de a analiza cum reacționează produsul pe diferite dispozitive sau de a evalua evoluția unei versiuni. De asemenea oferă și detalierea comportamentului utilizatorilor în aplicație construind un sistem de feedback în funcție de evenimentele generate în timpul utilizării acesteia.

Retrofit

Librăria Retrofit ușurează procesul de crearea a call-urilor HTTP prin capacitatea de a crea clienți REST abstractizând marea parte din procesul de retelistică. Această librărie suportă o multitudine de structuri de dată pentru transferul acestora asigurând așadar posibilitatea o manipularea ușoară a datelor consumate sau oferite de diferitele call-uri. Această librărie fiind una open-source..

Picasso

Picasso este o librărie pentru aplicațiile android vine cu un mecanism foarte util în descărcarea și cache-uirea imaginilor. Posibilitatea de configurare a acestui mecanism este de asemenea un punct forțe pe lângă design-ul conceput astfel încât să se integreze într-un mod compact cu platforma de dezvoltare android. Aceasta fiind de asemenea o librărie open-source.

PocketSphinx

Această librărie a fost construită în urma unor cercetări de la universitatea privată Carneige Mellon din Pittsburgh. Ca funcționalitate principală este aceea de recunoaștere continuă a vorbirii independent de difuzor.

Mecanismul de recunoaștere este pornit printr-o frază cheie, configurabilă ca mai apoi acesta să asculte date ce vor fi filtrate peste un set de date stocat într-un dicționar. Are capabilitatea de a pune la dispoziție atât cuvintele ce le-a recunoscut dintr-un dicționar dat cât și un scor asignat răspunsului pentru a putea aplica un filtru rezultatelor crescând semnificativ performanța acesteia.

Arhitectura Aplicatiei

În acest capitol voi prezenta modul în care a fost structurată aplicația *Robo Bucatarul*, motivând alegerile făcute în procesul de modelare a acestui proiect. Atunci când voi expune anumite componente din întreaga structură, voi abstractiza funcționalitățiile ce nu țin neapărat de acea parte pentru a avea o mai mare acuratețe în descrierea acesteia.

În figura 8. este prezentată schema generală a aplicației, unde sunt expuse doar componentele principale și conexiunile între ele. Având doar clasele esențiale proiectului, această figură prezintă relațiile și dependențele la nivel superior.

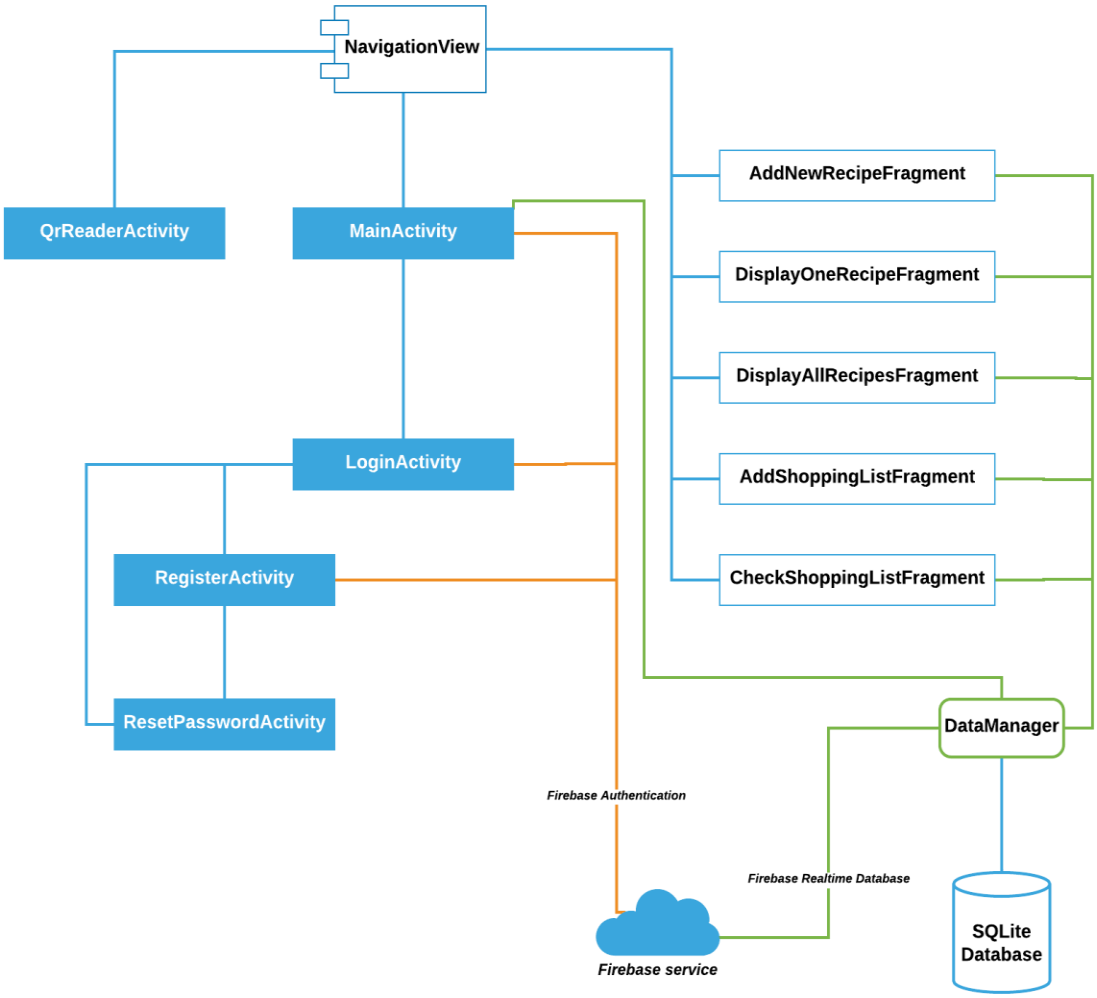

figura 8. arhitectura generala a aplicatiei

Pentru a avea un control cât mai stabil și eficient am ales să folosesc un NavigationView atașat activității principale (MainActivity). Acest view este unul dintre standardele de navigare folosite în dezvoltarea aplicațiilor oferind un design plăcut și un plus considerabil aplicației în ce privește accesibilitatea.

Alături de această componentă, pentru navigate, am mai folosit un manager de fragmente (FragmentManager), prin care fragmentele folosite în acest proiect au fost manipulate cu ajutorul acestuia. Motivul pentru care am folosit acest manager de fragmente este pentru că am dorit ca arhitectura aplicației să fie construită pe fragmente. Acestea reprezentând o soluție optimă atunci când partea de interfață a utilizatorului este des modificată.

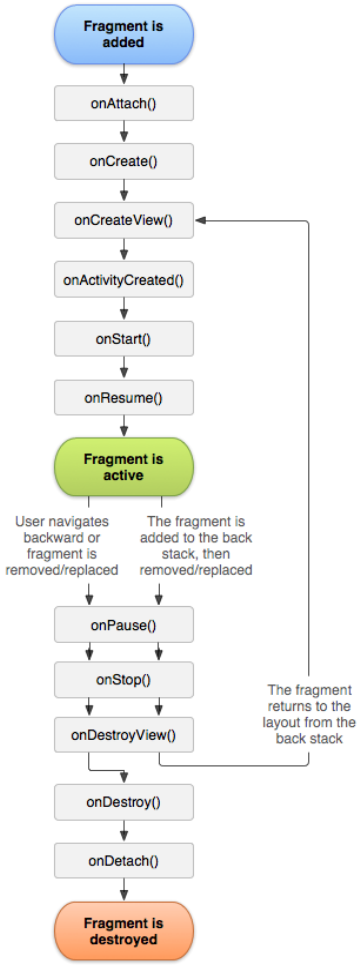

figura 9. ciclul de viata al unui fragment

Un fragment este o componentă de interfață, care atașată unei activități are propiul ciclu de viață dar va fi influențat și odată cu starea activității din care face parte. Poate fi privit ca un sub-modul al unei activități. A fost creat pentru a aduce mai multă dinamică și flexibilitate în ceea ce privește designul grafic.

Autentificare

Pentru stocarea datelor de autentificare a utilizatorilor am folosit serviciul *Firebase Authentication* ce asigura securitatea, durabilitatea datelor si un management foarte bine pus la punct pe partea de stocare a utilizatorilor si monitorizare a acestora.

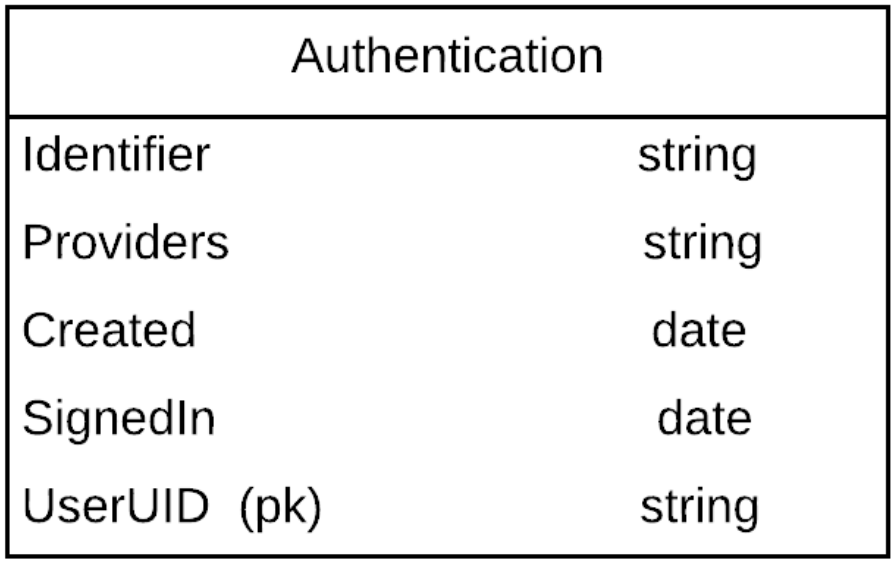

figura 10. structura autentificarii

În figura 10. este expus schema de stocare a informațiilor de autentificare a utilizatorilor. Deoarece detaliile despre autentificare fiind fixe sau cel puțin statice aici s-a folosit o bază de date SQL.

Având construit un kit de dezvoltare și pentru android, acesta a adus simplitate și stabilitate atât în ce privește autentificarea utilizatorului cât și în stocarea altor date ale utilizatorilor. Unicitatea, resetarea parolei și constrângeri legate de complexitatea parolei sunt obiective acoperite de acest serviciu. După înregistrarea unui utilizator, acestuia îi este asignat un identificator unic prin care acesta va putea realiza operațiuni de creare, citire, modificare și ștergere atât timp câte este logat.

Controlul și monitorizarea se poate face chiar din consola platformei, prima fiind găsită la ramura Users iar monitorizarea autentificării se găsește la ramura Usage. Mai mult de atât, configurările pentru email-ul de resetare a parolei, schimbare de email sau mesajul de verificare sunt disponibile pe această platformă în ramura Templates.

Monitorizare aplicatie

Firebase vine cu o multitudine de filtre ce pot fi aplicate peste datele colectate de acest serviciu. La secțiunea Dashboard din meniul principal al aplicației web regăsim aceste filtre unde putem afla date despre originea utilizatorilor, timpul de utilizare a diferitelor layout-uri, procentajul versiunile aplicației folosite de către utilizatori, dispozitivele pe care rulează aplicația, versiunea de android folosite pe aceste dispozitive și altele.

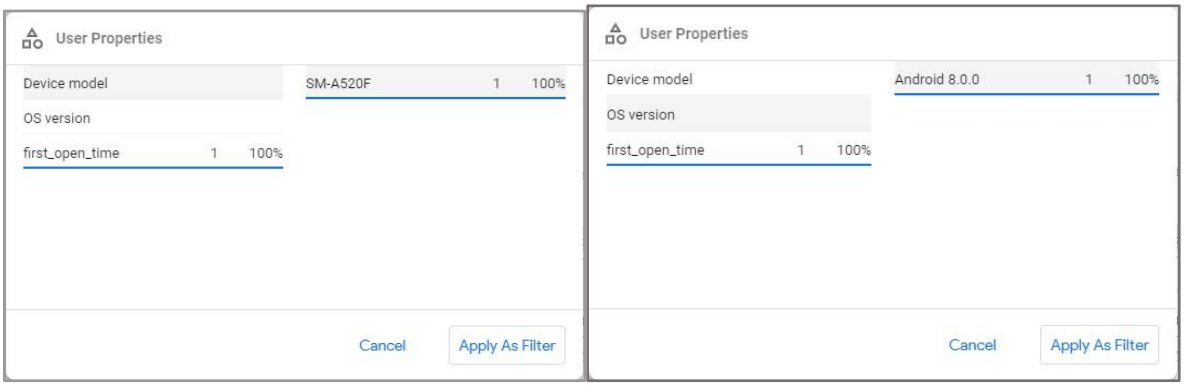

figura 11. monitorizarea dispozitivelor utilizate si a versiunii de android folosite

Un mod interesant de a vizualiza datele despre utilizatori în timp real pus la dispoziție de către *Firebase* este acela de *StreamView*. În cadrul acestui tip de vizualizare, cu un istoric de 30 de minute datele sunt colectate și expuse prin diferite filtre.

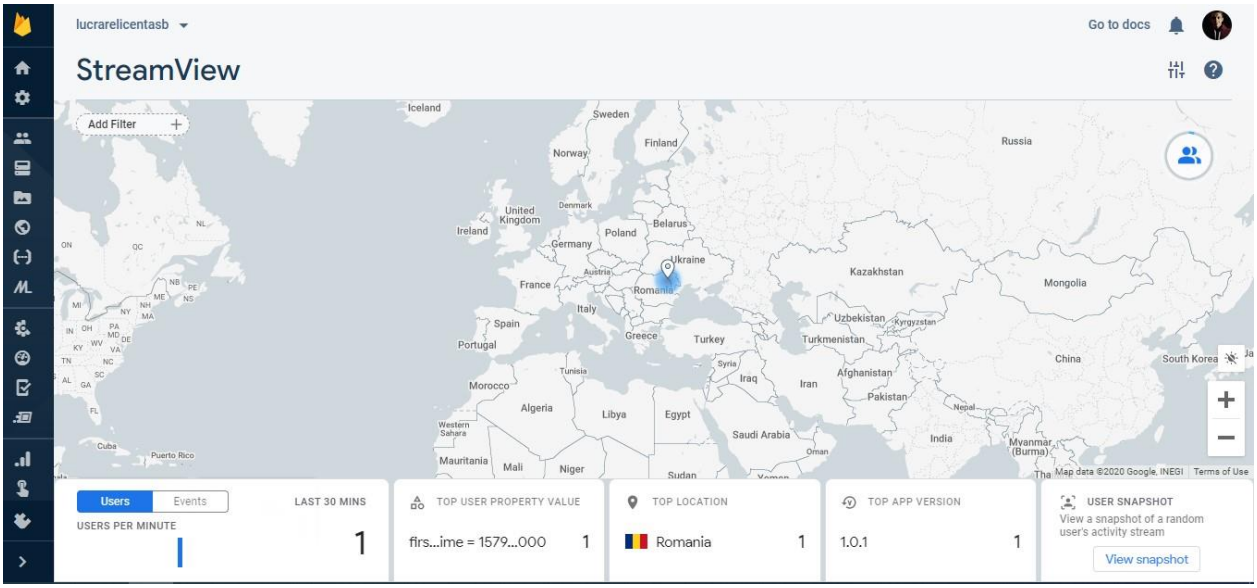

figura 12. modul de vizualizare StreamView pentru monitorizarea aplicatiei

Stocarea datelor utilizatorilor

Pentru stocarea altor date decât cele de autentificare am folosit modelul NoSQL pus la dispoziție de cei de la Firebase prin serviciul Realtime Database, Motivul principal pentru care am ales o stocare a datelor în cloud este faptul că îmi este asigurată durabilitatea datelor, indiferent de starea aplicației.

O caracteristică principală este consistenta datelor asigurată de modul în care este structurată această bază de date, fiind una realtime, punând la dispoziție o librărie ce permite dezvoltatorului să controleze evenimentele ce apar în această bază de date.

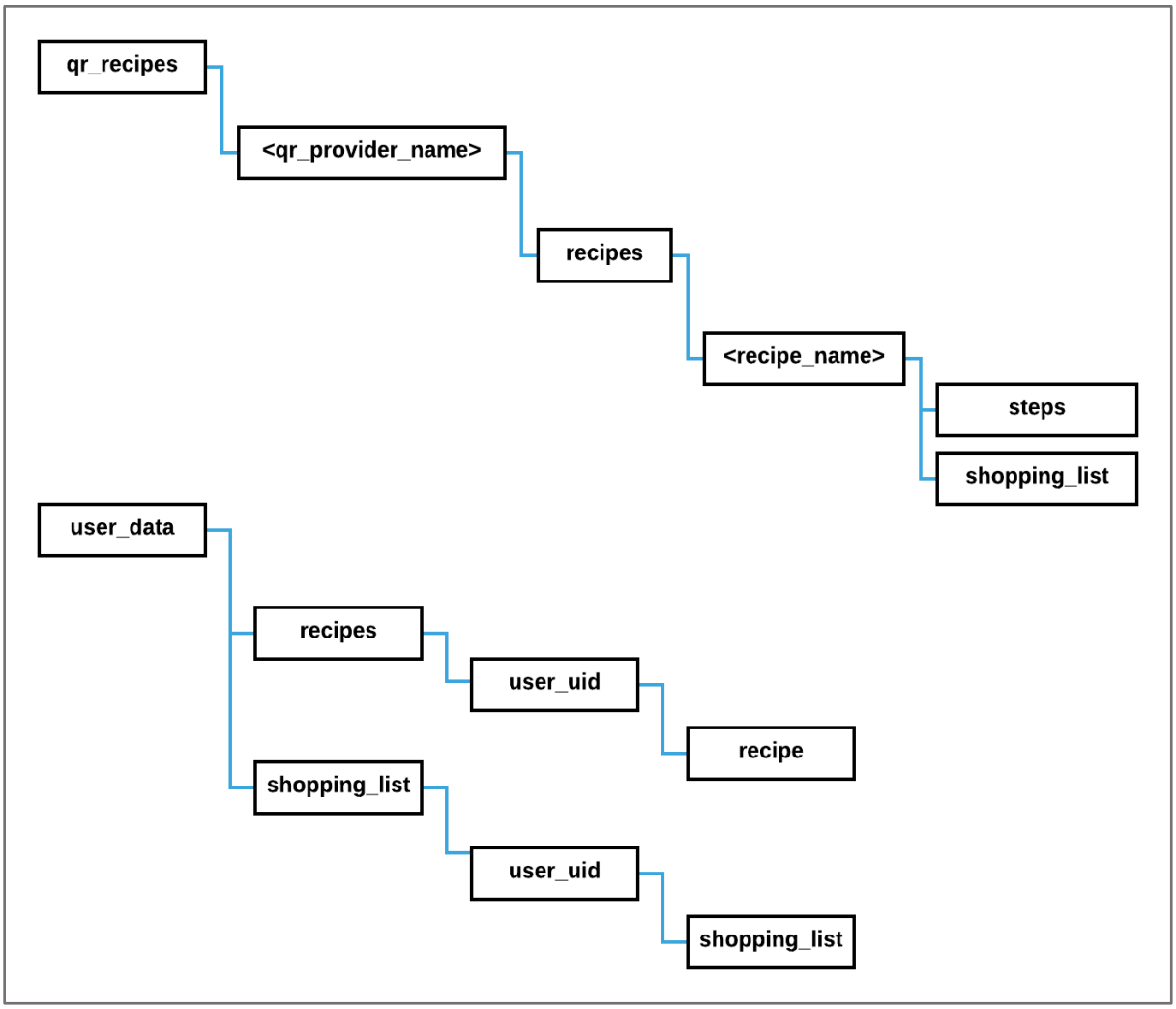

figura 13. stocare datelor folosite de utilizator

În figura 13. este prezentată structura datelor utilizate în aplicație de către utilizatori, toate aceste date fiind salvate în cloud asigură de asemenea securitatea datelor. Datele sunt încărcate dinamic în managerul de date care la rândul lui va expune datele în funcție de contextul în care se află aplicația. O altă responsabilitate pe care o are managerul de date este de a actualiza fiecare modificare realizată de către utilizator. Această modalitate de a stoca datele va necesita o conexiune la internet într-un mod continu.

Alături de datele utilizatorului managerul de date va declanșa atât actualizarea datelor ce vor fi utilizate ca recomandare în lista de cumpărături cât și expunerea lor atunci când aplicația va avea nevoie de ele Aceste date fiind stocate local într-o bază SQLite.

Modul în care au fost grupate clasele și legăturile dintre acestea la nivel de pachet sunt prezentate în figura 14. Pentru a simplifica această expunere am omis pachetele predefinite ale aplicațiilor în Android (drawable, layout, res etc.).

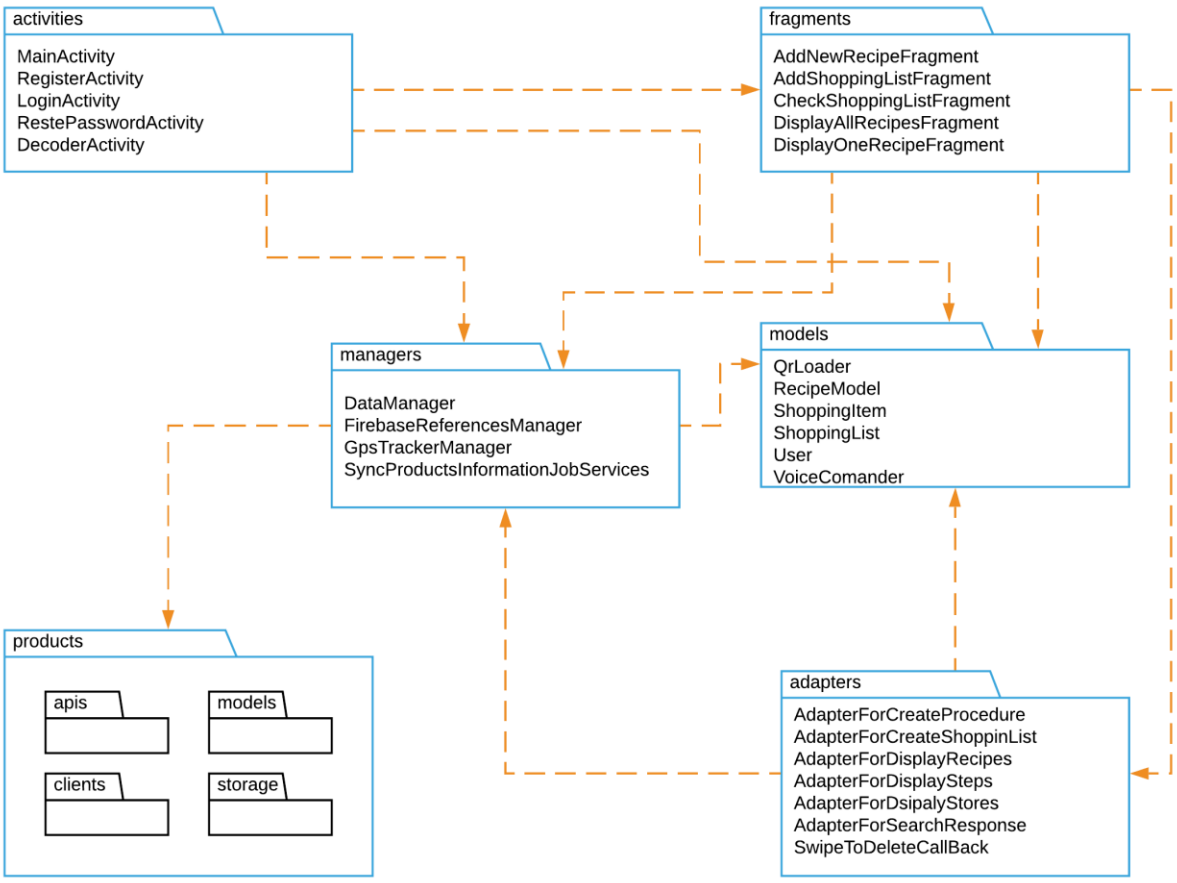

figura 14. structurarea pe pachete

Probleme intampinate pe parcursul dezvoltarii aplicatiei

Următoarele subiecte vor fi despre probleme și soluțiile găsite pe durata dezvoltării a întregului proiect. Voi aborda fiecare dintre aceste arătând modul de abordare în găsirea unei rezolvări optime ale acestora. Unele dintre ele au necesitat mai multe zile de căutări, studii deoarece o parte din ele sunt cauzate de faptul că android nu a creat suport în direcția în care merge această aplicație.

Eliminarea sunetelor de notificare a recunoasterii vocale

O primă problemă pe care am avut-o a fost faptul că android nu expune nici un api pentru recunoașterea vocală fără a emite sunetele de început și de final a acestui proces. Deși există aplicații în sistemul de operare android ce au asemenea funcționalități cum ar fi asistentul Google, sistemul de librării android nu oferă asemenea funcționalitate și nici nu permite configurarea recunoașterii vocale în mod silențios.

De aceea eforturile depuse au fost în direcția unei librării open-source, ușor de configurat și capabilă să răspundă la cerințele aplicației mele. După mai multe librării încercate, PocketSphinx are capabilitatea de a recunoaște nu doar fără sunetele de notificare ci și setul de date peste care se va efectua filtrarea cuvintelor.

Implementarea unui sistem de recomandare in lista de cumparaturi

Pentru a putea aduce puțină noutate în funcționalitatea adăugării listei de cumpărături unei rețete am dorit să construiesc un sistem de recomandare pentru produsele ce puteau fi adăugate. Descoperirea unui site ce oferă informații despre prețurilor produselor alimentare în funcție de locație a fost soluția acestei probleme. Acest site se numește monitorulpreturilor.info și este o inițiativă a Consiliului Concurenței din România alături de Autoritatea Națională pentru Protecția

Consumatorului din România, având ca scop informarea consumatorilor asupra prețurilor produselor vândute de rețelele comerciale participante la acest proiect.

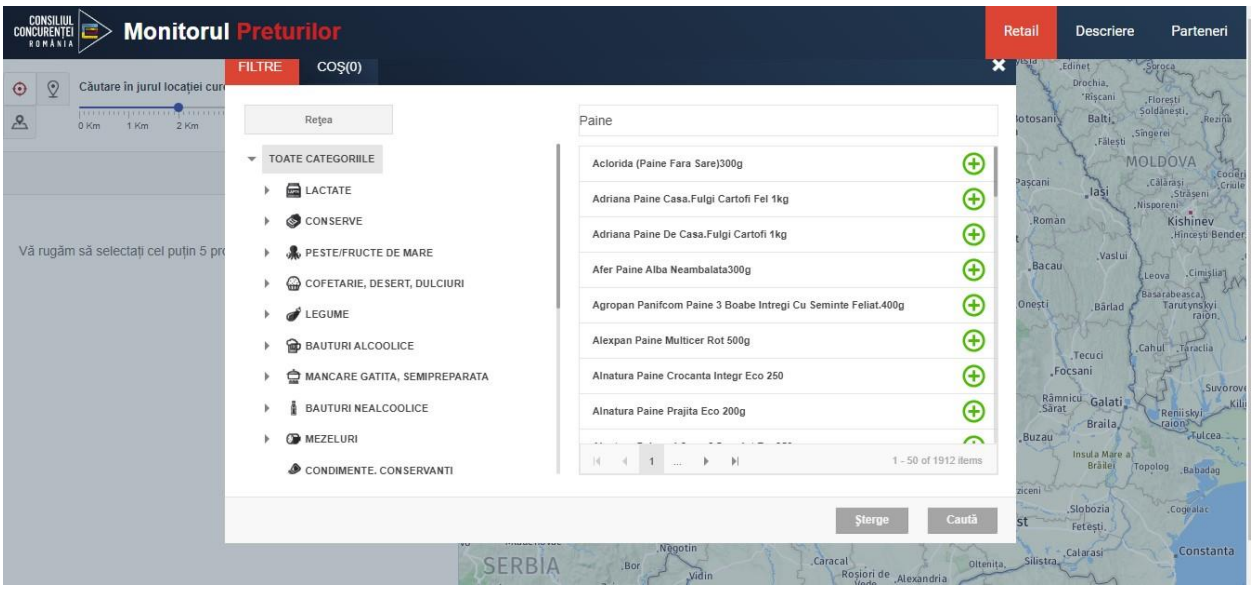

figura 15. Site-ul https://www.monitorulpreturilor.info

După descoperirea acestui site o altă problemă a fost faptul că nu expunea explicit informațiile în exteriorul aplicației web. Așadar neavând api-uri expuse de către sote am ales să exploatez prin consola browser-ului request-urile efectuate în momentul utilizării aplicației web aflând către ce url-uri se fac request-urile.

Următorul pas a fost să folosesc o aplicație ce îmi permite efectuarea request-urilor într-un mod mai organizat pentru a testa url-urile descoperite. Pentru acest procedeu am folosit *Postman Canary*, care m-a ajutat să înțeleg cum trebuie să folosesc request-urile în aplicația mea.

Lucrurile nu s-au oprit aici deoarece acste request-uri returnau datele în format xml, acest lucru crescând semnificativ complexitatea de creare a șabloanelor de răspuns. Pentru a evita acest lucru am generat cu ajutorul unui convertor toate clasele necesare. În final după ce datele au ajuns în aplicația android am salvat doar datele necesare într-o bază SQLite.

Numarul limtat de stocare a caracterelor in QR-coduri.

În procesul de dezvoltare a funcționalității de citire a unei rețete dintr-un qr code a apărut limitarea provenită din numărul maxim de caractere ce pot fi stocate într-un qr code, acesta fiind de 4.296 de caractere. Deoarece acest lucru limitează capcaitatea aplicației de a transfera rețete de orice dimensiune am încercat să rezolv această problemă.

Modul de abordare este unul simplu și în același timp sigur din punct de vedere al expunerii informațiilor. Am ales să stochez datele în baza de date din cloud, aducând cu aceasta și posibilitatea de a capacita acest proiect și în direcția magazinelor ce doresc să ofere rețete pentru aplicația *Robo Bucătarul* astfel încât să fie creată o platformă benefică atât pentru consumatori cât și pentru producători.

După această modificare în codul QR am stocat doar calea în baza mea de date NOSQL, iar printr-un cod generat unic din 36 de caractere pentru fiecare rețetă. Această abordare permite și modificarea dinamică a datelor dintr-un qr, deoarece în acest cod am stocat o referință și nu datele în sine.

figura 16. stocarea rețetei sub formă de QR-code

Detaliile aplicatiei

În următoarele subcapitole voi vorbi despre detaliile aplicației ce a o fost dezvoltată pe platforma de dezvoltare android. Accentul îl voi pune pe modul de în care am ales să implementez diferite funcționalități argumentând alegerea făcută acolo unde este cazul.

Interfata grafica

Pentru interfața grafică a aplicației am ales un număr cât mai mic de culori pentru a aduce un plus în zona de simplitate, reușind astfel să păstrez un oarecare confort vizual la utilizarea acestei aplicații. Culorile au fost alese după mai multe teste asupra combinațiilor dintre ele, căutând să nu impactez negativ partea de interfață.

Numele si logo-ul aplicatiei

Numele aplicației a fost modificat de mai multe ori încercând să transmit totalitatea funcționalităților ce le va oferi aceasta. Deși se numește *Robo Bucătarul* am ajuns la concluzia că folosirea întregului nume în logo nu este o soluție astfel folosind doar primul parte a numelui am păstrat simplitatea și în logo-ul aplicației, fiind chiar ușor de memorat.

figura 17. logo-ul aplicatiei

Design-ul meniului de navigare

După crearea logo-ului am încercat să aplic aceeași tehnică vizuală pe care o folosesc marile companii în aplicațiile lor și anume de a construi interfața aplicației bazată pe culorile predominante din logo.

Astfel meniul principal este construit pe baza culorile ce se regăsesc în logo, arhitectura acestuia fiind una verticală. Elementele din acest meniu sunt selectabile iar background-ul unui element îți specifică în ce context se află utilizatorul. În funcție de contextul în care se afla utilizatorul, opțiunile aflate în meniu se vor modifica. Meniul v-a fi vizibil prin apăsarea unui buton poziționat în parte din stânga sus, în bara de instrumente a aplicației.

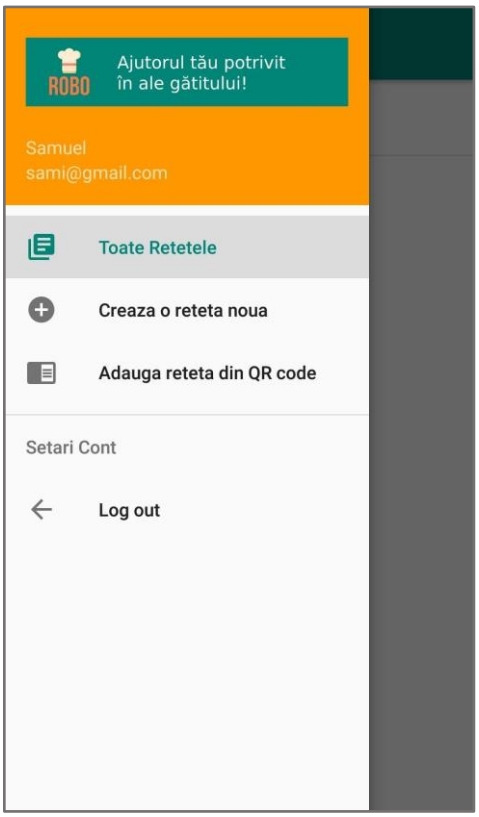

figura 18. meniul aplicației în faza de început

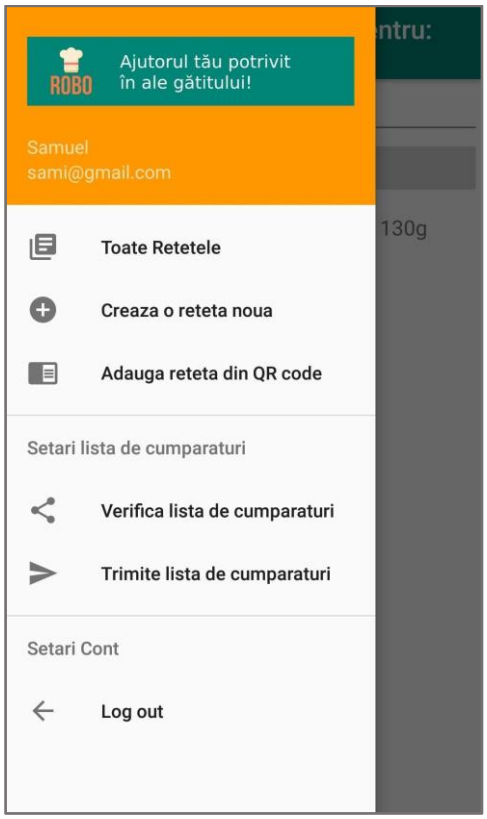

figura 19. meniul aflat in lista de cumpărături

Opțiunile disponibile de navigare

Voi adăuga diferite ecran-uri din aplicație și voi menționa ce reprezintă fiecare dintre aceste analizând fiecare element grafic ce aparține acelui screen. Se va menționa și scopul fiecăruia în ansamblul aplicației mobile.

1. **Ecranul "Toate retetele"**

Acest ecran este și ecranul principal al aplicației, astfel după logare utilizatorul va ajunge în acest ecran. Aici se regăsesc toate rețele utilizatorului ordonate alfabetic pentru a fi ușor de căutat pintre acestea. Fiecare rețetă va putea fi deschisă apăsând pe aceasta. Pe lângă titlul rețetei aceasta mai conține și numărul de pași pe care îl are fiecare rețetă în parte.

figura 20. Ecranul "Toate rețele"

2. **Ecranul “Pasii retetei”**

În acest ecran se poate ajunge doar prin apăsarea pe una dintre rețetele afișate în ecranul prezentat anterior. Aici sunt afișați pașii rețetei selectate, fiecare dintre aceștia fiind numerotați având posibilitatea de a depăna printre aceștia.

În bara de sus regăsim butonul de deschidere a meniului de navigare în partea stângă, titlul ecranului iar că subtitlu este numele rețetei selectate. În partea dreaptă apare un buton nou ce va fi vizibil doar în acest ecran și anume butonul de navigare către lista de cumpărături a rețetei. Motivul pentru care am ales o dimensiune mai mare a dimensiunii textului este pentru a ajuta și vizual utilizatorul dacă acesta va dori vreodată să folosească aplicația fără comenzile vocale.

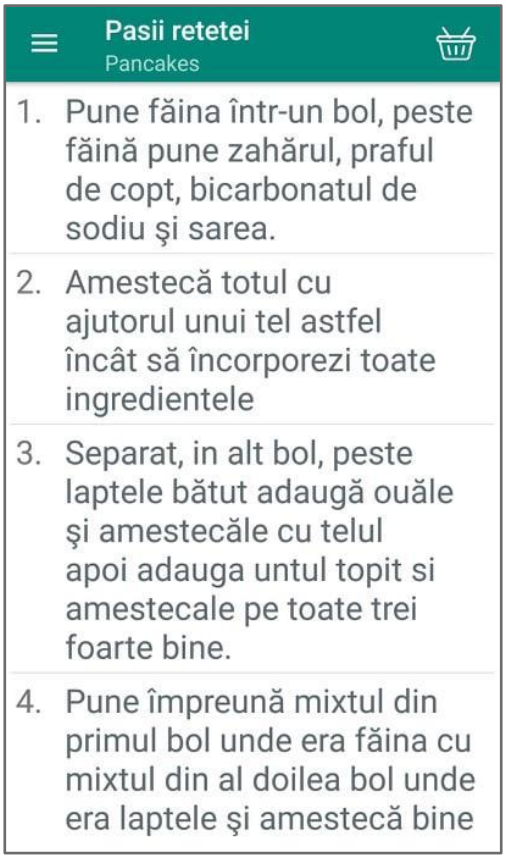

figura 21. ecranul “Pasii retetei”

3. Ecranul “Lista de cumparaturi”

Atașat unei rețete, listă de cumpărături vine cu un ecran relativ simplu ce permite adăugarea diferitelor produse în aceasta. Accesul către lista de cumpărături se realizează din bara de sus, a fiecărei rețete în parte, prin butonul poziționat în colțul din dreapta, reprezentat printr-un coș de cumpărături. Imediat după bara de sus este poziționat un text editabil unde utilizatorul va putea introduce numele produsului pe care îl dorește să îl adauge în listă.

Pe parcurs ce utilizatorul va introduce numele produsului se va deschide în jos ca și recomandare o listă cu toate produsele alimentare a celor mai cunoscute supermarketuri ce s-ar potrivi ca și nume cu produsul introdus. După selectarea produsului dorit utilizatorul v-a apăsa butonul de adăugare a produsului. Fiecare produs adăugat are în partea din stânga un check-box ce poate fi bifat sau nu. În funcție de acesta produsele sunt filtrate atunci când utilizatorul dorește să le trimită prin mesaj.

figura 22. lista de cumpărături

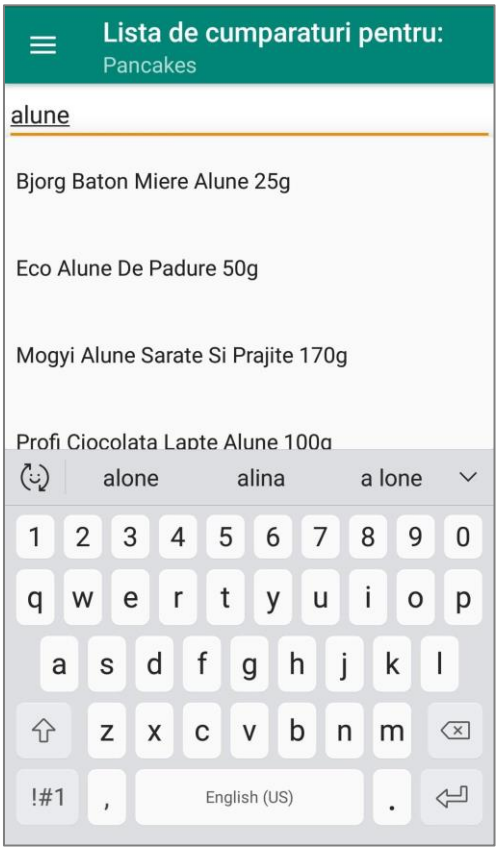

figura 23. lista produselor recomandate

4. Ecranul “Verficare a listei de cumparaturi”

Aici ecranul este compus din același bară pe care o regăsim în fiecare ecran, imediat sub ea se află butonul pe care utilizatorul îl va apăsa atunci când dorește să pornească o căutare pentru produsele din listă. În partea de jos a ecranului se afla un selector orizontal pentru distanța maximă folosită pentru căutarea magazinelor ce conțin produsele.

După apăsarea butonului răspunsul este afișat fiind ordonat după mai multe criterii, primul fiind acela ca numărul produselor găsite să fie cât mai mare, al doilea fiind prețul pentru produsele găsite să fie cât mai mic iar al treilea fiind distanța magazinelor să fie cât mai mic, ordonarea fiind făcută astfel încât să ofere utilizatorului cea mai bună variantă. Pentru o detaliere a fiecărui rezultat în parte utilizatorul va trebui să apese pe butonul poziționat în partea din dreapta a fiecărui răspuns din listă. Aici va avea un răspuns complet despre produsele care se găsesc la acel magazine.

fig.24. răspunsul pentru lista de cumpărături

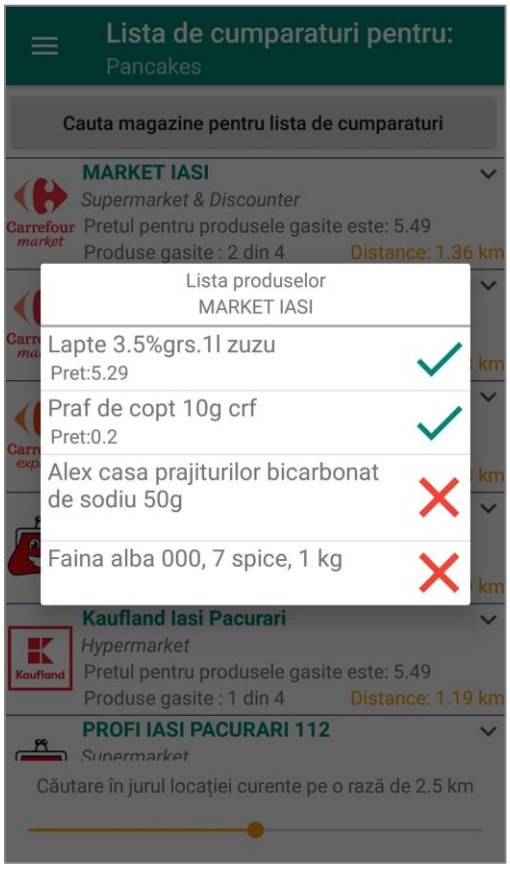

fig. 25. răspunsul detaliat pentru un magazin

5. **Ecranul “Adauga o noua retea”**

Structura acestui ecran va cuprinde toate componentele necesare definiri unei noi rețete indiferent de lungimea pașilor. Astfel mai jos de bara de sus se află textul editabil pentru titlu ce are un design puțin diferit de restul. Pentru a avea o vizibilitate mai bună asupra titlului am ales ca pe acesta să îl poziționez separat de pașii rețetei rămânând fix pe ecran.

În continuare este poziționată listă de pași care implicit vine cu 3 pași iar prin butonul poziționat în stânga jos se pot adăuga noi pași la rețetă. În partea din dreapta a ecranului se află butonul prin care utilizatorul va putea să își salveze noua rețete, odată ce a apăsat pe butonul de salvare, utilizatorul va fi redirecționat către ecranul principal unde va apărea și noua rețetă.

figura 26. ecranul de adaugare a unei noi retete

Detaliile de implementare

În acest capitol voi selecta anumite componente din aplicației și voi prezenta atât modul de implementarea a acestora cât și utilitatea lor în proiect. Mă voi axa pe explicarea în detaliu a modului în care au fost scrise aceste componente.

Speech Recognizer

Componență speech recognizer oferă funcționalitatea de recunoaștere vocală în interiorul aplicației. Prin acest serviciu utilizatorul va transmite aplicației comenzi vocale putând astfel controla flow-ul unei rețete. Acest serviciu configurabil va fi activat prin folosirea unei fraze cheie, ca mai apoi să aștepte ca utilizatorul să spună o comandă vocală, fiind configurată asemănător cu modul în care Google Assistant funcționează.

```
private void setupRecognizer(File assetsDir) throws IOException {
    recognizer = SpeechRecognizerSetup.defaultSetup()
            .setAcousticModel(new File(assetsDir, "en-us-ptm"))
            .setDictionary(new File(assetsDir, "cmudict-en-us.dict"))
            .getRecognizer();
    recognizer.addListener(this);
    recognizer.addKeyphraseSearch(WAITING_FOR_WAKEUP, KEYPHRASE);
    File menuGrammar = new File(assetsDir, "menu.gram");
    recognizer.addGrammarSearch(WAITING_FOR_COMMANDS, menuGrammar);
}

private void switchSearch(String searchName) {
    recognizer.stop();
    if (searchName.equals(WAITING_FOR_WAKEUP))
        recognizer.startListening(searchName);
    else
        recognizer.startListening(searchName, 3000);
}
```

figura. 27. setări aplicate serviciului de recunoaștere vocală

În prima funcție este configurat acest serviciu care va primi cuvântul cheie, componenta listei de cuvinte cu care se va face filtrarea fulx-ului de intrare, acustică și dicționarul la care se va raporta pe parcursul recunoașterii vocale. Înregistrarea pentru a asculta evenimentele expuse în exterior de către acest serviciu să efectuat tot aici. În a doua funcție se realizează interschimbarea modului de ascultare, de la fraza cheie la lista de cuvinte predefinite.

Voice Commander

Alături de serviciul de recunoaștere vocală am ales să folosesc un filtru al comenzilor pe care le va recunoaște acesta. În această clasă sunt stocate atât tipurile de acțiuni recunoscute cât și comenzile asignate fiecăruia dintre ele. Am ales doar o arie mică în care utilizatorul să folosească comenzile vocale. Mai exact doar atunci când dorește să parcurgă pașii unei rețete, în oricare altă stare ar fi aplicația aceasta nu va avea capacitatea de a asculta utilizatorul.

```
public class VoiceCommander {
    private final HashMap<String, String> allKnownCommands = new HashMap<>();
    public final String startRecipeCommand = "START_RECIPE";
    public final String nextStepCommand = "NEXT_STEP";
    public final String previousStepCommand = "PREVIOUS_STEP";
    public final String repeatStepCommand = "REPEAT_STEP";

    public VoiceCommander() {
        init();
    }

    void init() {
        allKnownCommands.put("let's start cooking", startRecipeCommand);
        allKnownCommands.put("let's start", startRecipeCommand);

        allKnownCommands.put("next step", nextStepCommand);
        allKnownCommands.put("next please", nextStepCommand);
        allKnownCommands.put("next step please", nextStepCommand);

        allKnownCommands.put("previous step", previousStepCommand);
        allKnownCommands.put("previous please", previousStepCommand);
        allKnownCommands.put("previous step please", previousStepCommand);

        allKnownCommands.put("repeat please", repeatStepCommand);
        allKnownCommands.put("repeat the last step", repeatStepCommand);
    }

    public String getActionForVoiceCommand(String voiceCommand) {
        return allKnownCommands.get(voiceCommand);
    }
}
```

figura 28. mecanismul de parsare a comenzilor vocale recunoscute de aplicație

Acțiunile pe care aplicația le poate face prin comenzi vocale sunt de a începe o rețetă, de a expune următorul pas, de a repeta pasul curent sau de a se întoarce la pasul precedent. Fiecare dintre acestea având 2 sau 3 moduri de a fi declanșate.

În porțiunea de cod de mai jos este realizată gestiunea exacta a comenzilor vocale ce vor folosi un serviciul de text-to-speech pus la dispoziție de sistemul de operare. Acesta va prelua conținutul unui pas din rețeta afișată și o va expune în format vocal.

Pentru ca o comandă să ajungă în această funcție ea trebuie ca mai întâi să fie recunoscută de serviciul de recunoaștere vocală, iar mai apoi prin VoiceCommander se realizează parsarea comenzii pentru a se ajunge la acțiunea de bază. După ce acțiunea de bază a fost găsită această funcție v-a fi apelată pentru finalizarea acestui proces de control vocal. Pentru a nu permite utilizatorului să crească poziția la care v-a putea crește s-au scădea am limitat aria de lucru a acestuia în funcție de poziția 0 și lungimea fiecărei rețete împarte.

```
private void executeVoiceCommand(String voiceAction) {
    try {
        switch (voiceAction) {
            case VoiceCommander.START_RECIPE:
                Log.i(TAG, "startProcedure: " + mRecipeSteps.get(mCurrentStepPosition));
                mCurrentStepPosition = 0;
                speak(mRecipeSteps.get(mCurrentStepPosition));
                break;
            case VoiceCommander.NEXT_STEP:
                mCurrentStepPosition++;
                if (mCurrentStepPosition < mRecipeSteps.size() && mCurrentStepPosition >= 0) {
                    Log.i(TAG, "nextStep: " + mRecipeSteps.get(mCurrentStepPosition));
                    speak(mRecipeSteps.get(mCurrentStepPosition));
                } else {
                    mCurrentStepPosition--;
                    Log.e(TAG, "parseVoiceCommand: next step didn't exist ");
                }
                break;
            case VoiceCommander.PREVIOUS_STEP:
                mCurrentStepPosition--;
                if (mCurrentStepPosition < mRecipeSteps.size() && mCurrentStepPosition >= 0) {
                    speak(mRecipeSteps.get(mCurrentStepPosition));
                } else {
                    mCurrentStepPosition++;
                    Log.e(TAG, "parseVoiceCommand: back step didn't exist ");
                }
                break;
            case VoiceCommander.REPEAT_STEP:
                if (mCurrentStepPosition >= 0 && mCurrentStepPosition < mRecipeSteps.size()) {
                    speak(mRecipeSteps.get(mCurrentStepPosition));
                } else {
                    Log.e(TAG, "parseVoiceCommand: repeat step didn't exist ");
                }
                break;
            default:
                break;
        }
    } catch (Exception ignored) {}
}
```

figura 29. procesul de executare a comenzilor vocale

Procesul de cautare produselor din lista de cumparaturi

În procesul de căutare a produselor din lista de cumpărături am folosit call-uri către api-urile interne ale site-ului www.monitorulpreturilor.info. Pentru a putea comunica cu site-ul acesta a fost nevoie mai întâi de a crea o bază de date a aplicației ce va conține fiecare nume al produsului alimentar și codul unic de identificare a acestuia. Astfel am asigurat că aplicația va comunica cu api-urile interne pe baza acelor coduri.

Pentru a reuși să ofer utilizatorului date pe baza locației a fost nevoie să folosesc api-urile puse la dispoziție de sistemul de operare pentru a afla locația acestuia. După ce am asigurat toate aceste date nu a mai rămas decât realizarea call-ului către api-ul intern ce va întoarce un răspuns în format XML, structura acestuia fiind modelată pentru fiecare tag în parte.

```
private void getStoresForShoppingListByLocation(String latitude, String longitude,
    String distanceInMeters, String productsDividedByComma, String orderBy) {
    new Thread(new Runnable() {
        @Override
        public void run() {
            Retrofit retrofit = new Retrofit.Builder()
            .baseUrl("https://www.monitorulpreturilor.info/")
            .addConverterFactory(SimpleXmlConverterFactory
            .create())
            .build();

            ProductsAPIs api = retrofit.create(ProductsAPIs.class);

            Call<RetailStores> call = api.getStoresForProductsByLatLon(latitude, longitude,
            distanceInMeters, productsDividedByComma, orderBy);
            call.enqueue(CheckShoppingListFragment.this);
        }
    }).start();
}
```

figura 30. efectuarea call-ului de verificare a produselor in magazinele apropiate

Concluzii

Concluzând totalitatea lucrurilor argumentate mai sus se poate spune că aplicația “Robo Bucătarul" dezvoltată pe platforma de Android acoperă în totalitate obiectivele propuse oferind de asemenea utilizatorului o experiență plăcută în utilizarea acesteia. De asemenea aceasta are ca grup țintă persoanele ce vor să folosească aplicația aceasta în uz personal.

Pot afirma faptul că această aplicație se va plia foarte bine în contextului persoanelor ce doresc să înceapă să exploreze într-un mod mai serios zona gastronomică. Comparând cu alte aplicații, diferența majoră este modul de abordare a probleemelor și focusul mult mai ridicat pe învățare a rețetelor dar totuși păstrând o simplitate în modul de construcție a acesteia.

Un pilon al planului de extindere a acestui proiect este de a realiza o aplicația web care va oferi întreprinderilor ce comercializează produse alimentare oportunitatea de a expune diferite rețete sub formă de QR-code, având control independent la managementul acestora. Scopul acestei extinderi este pentru a crea o platformă comună atât pentru comparator cât și pentru vânzător.

O zonă ce merită exploatată mai mult este aceea a modului în care este recomandat un produs utilizatorului. Aici soluțiile sunt multiple dar aș vedea util atât un istoric al produselor deja prezente în alte liste de cumpărături cât și o aproximare eficientă a produselor din lista de cumpărături ce nu sunt în recomandări.

Cred că o completare la soluția dată de acest proiect este de a comunica cu un device special conceput pentru gastronomie ce va veni cu noi capabilități în acest domeniu. Astfel prin acesta aplicația de pe telefon va putea noi funcționalități utilizatorilor, acesta fiind direct influențate de starea prezentă a evenimentelor din bucătăriile acestora. În alte ordine de idei acest proiect ar putea avea și direcțiile unui start-up în această zonă.

Bibliografie

https://developer.android.com/reference/android/support/v7/widget/RecyclerView

https://www.android.com/android-10/

https://en.wikipedia.org/wiki/Android_(operating_system)

https://github.com/cmusphinx/pocketsphinx-android

https://firebase.google.com/docs

https://square.github.io/retrofit/

https://square.github.io/picasso/

https://developer.android.com/guide/components/fragments

https://developer.android.com/guide/components/activities/activity-lifecycle

https://developer.android.com/guide/topics/manifest/manifest-intro

https://www.monitorulpreturilor.info/