

APLICAȚIE MOBILĂ PENTRU GESTIONAREA REȚETELOR CULINARE

Candidat: Alexandra-Doriana NOVACESCU
Conducător științific: ȘI. Dr. Ing. Răzvan CIOARGĂ

Sesiunea: Iunie 2025

REZUMAT

Lucrarea de față prezintă proiectarea și implementarea unei aplicații mobile denumită EasyChef, concepută pentru utilizatorii care își doresc să identifice rapid rețete culinare personalizate, compatibile cu ingredientele disponibile în propria cămară și în funcție de preferințele alimentare. Obiectivul principal al aplicației este de a răspunde unei nevoi reale și frecvente, adică optimizarea deciziilor alimentare, printr-un sistem intuitiv, eficient și accesibil de selecție, planificare și preparare a meselor. EasyChef vizează în special economisirea timpului și încurajarea unui stil de viață organizat.

Din punct de vedere tehnic, aplicația utilizează tehnologii moderne, precum framework-ul React Native, împreună cu platforma Expo, pentru dezvoltarea unei interfețe compatibile cu dispozitivele Android și iOS. Infrastructura de backend este susținută de Firebase, care asigură autentificarea utilizatorilor, stocarea datelor în Firestore și sincronizarea în timp real a conținutului. Aplicația oferă funcționalități precum căutarea rețetelor în funcție de ingrediente, adăugarea de rețete proprii, salvarea preferințelor alimentare, planificarea meselor și marcarea rețetelor finalizate. Imaginele asociate rețetelor sunt gestionate cu ajutorul serviciului Cloudinary, care permite încărcarea optimizată a conținutului media.

Procesul de dezvoltare a inclus toate etapele esențiale, de la analiza aplicațiilor existente și identificarea cerințelor, până la proiectarea modelului de date, implementarea funcționalităților și testarea finală a aplicației. Arhitectura adoptată este una modulară, cu separare clară între logica aplicației și interfața vizuală, ceea ce facilitează întreținerea și extinderea ulterioară a proiectului.

Lucrarea evidențiază soluțiile tehnice aplicate pentru a asigura o experiență stabilă și intuitivă, dar și provocările întâmpinate în integrarea serviciilor externe și în filtrarea dinamică a conținutului în funcție de utilizator. În încheiere, sunt propuse direcții viitoare de dezvoltare care includ extinderea funcționalităților multimedia, introducerea unui sistem de analiză nutrițională, integrarea unui scanner de coduri de bare și dezvoltarea unor funcții vocale, în vederea creșterii accesibilității și gradului de personalizare oferit utilizatorilor.

CUPRINS:

1. INTRODUCERE	9
1.1. CONTEXTUL GENERAL.....	9
1.2. MOTIVAȚIA ALEGERII TEMEI	10
1.3. OBIECTIVELE APLICAȚIEI	11
1.4. STRUCTURA LUCRĂRII.....	12
2. ANALIZĂ DE PIAȚĂ.....	15
2.1. APLICAȚII SIMILARE EXISTENTE.....	15
2.2. PUBLICUL ȚINTĂ	16
2.3. FUNCȚIONALITĂȚI COMUNE ȘI DIFERENȚIATORI	17
2.4. CERINȚE FUNCȚIONALE.....	18
2.5. CERINȚE NON-FUNCȚIONALE.....	19
3. PROIECTAREA APLICAȚIEI.....	21
3.1. ARHITECTURA GENERALĂ	21
3.2. DIAGRAAME UML.....	22
3.2.1. DIAGRAAME DE CAZURI DE UTILIZARE	23
3.2.2. DIAGRAAME DE ACTIVITATE	25
3.2.3. DIAGRAAME DE SECVENTĂ.....	29
3.3. ASPECTUL INTERFEȚEI	32
3.3.1. NAVIGARE PRIN FILE.....	32
3.3.2. ELEMENTE VIZUALE.....	32
3.4. MODELUL BAZEI DE DATE	33
4. IMPLEMENTAREA APLICAȚIEI	37
4.1. TEHNOLOGII UTILIZATE.....	37
4.1.1. REACT NATIVE.....	37
4.1.2. FIREBASE.....	38
4.1.3. CLOUDINARY	39
4.2. STRUCTURA PROIECTULUI.....	40
4.2.1. ORGANIZAREA FIŞIERELOR	40

4.2.2. SISTEMUL DE RUTE PRINCIPALE	41
4.2.3. NAVIGAREA CU EXPO-ROUTER	44
4.3. FUNCȚIONALITĂȚI IMPLEMENTATE	45
 4.3.1. PROCESUL DE ÎNREGISTRARE ȘI AUTENTIFICARE	45
 4.3.2. PROFILUL UTILIZATORULUI	46
 4.3.3. PROCESUL DE GESTIONARE A PREFERINȚELOR	46
 4.3.4. PROCESUL DE GESTIONARE A CĂMĂRII	51
 4.3.5. PROCESUL DE GĂSIRE A UNEI REȚETE	53
 4.3.6. PROCESUL DE GĂTIT PROPRIU-ZIS	55
 4.3.7. FINALIZAREA UNEI REȚETE	56
 4.3.8. PROCESUL DE ADĂUGARE A UNEI REȚETE	56
 4.3.9. PROCESUL DE PLANIFICARE A REȚETELOR	58
4.4. INTERACȚIUNEA CU THEMEALDB	60
4.5. INTERACȚIUNEA CU FIREBASE	62
4.6. INTERACȚIUNEA CU CLOUDINARY	64
5. TESTARE ȘI PROBLEME ÎNTÂMPINATE	67
 5.1. PROCESUL DE TESTARE	67
 5.2. PROBLEME ȘI SOLUȚII	68
6. MANUAL DE UTILIZARE	71
 6.1. ÎNCEPEREA UTILIZĂRII	71
 6.2. GHID NAVIGARE ȘI FUNCȚII PRINCIPALE	72
 6.3. FLUX DE UTILIZARE	74
 6.4. SCENARIU COMPLET	75
7. CONCLUZII ȘI DIRECȚII DE DEZVOLTARE VIITOARE	93
 7.1. CONCLUZII	93
 7.2. LIMITĂRI ACTUALE	94
 7.3. FUNCȚIONALITĂȚI POSIBILE ÎN VERSIUNI VIITOARE	96
8. BIBLIOGRAFIE	99

TABEL FIGURI:

Figură 1 - Diagrama de cazuri de utilizare cu modulele principale	23
Figură 2 - Diagrame de cazuri de utilizare pentru modulele Acasă și Rețete	24
Figură 3 - Diagrame de cazuri de utilizare pentru modulele Cămară și Profil.....	24
Figură 4 - Diagrama de activitate a procesului de autentificare.....	26
Figură 5 - Diagrama de activitate a procesului de gătit.....	27
Figură 6 - Diagrama de activitate a procesului de adăugare a unei rețete noi....	28
Figură 7 - Diagrama de secvență a autentificării utilizatorului	29
Figură 8 - Diagrama de secvență a adăugării unei rețete	30
Figură 9 - Diagrama de secvență a generării sugestiilor	31
Figură 10 - Diagrama arhitecturii bazei de date.....	36
Figură 11 – Exemplu structura câmpului bannedIngredients.....	49
Figură 12 - Ecranele de autentificare și înregistrare	76
Figură 13 – Exemple de mesaje de eroare la autentificare	77
Figură 14 - Exemple de mesaje de eroare la înregistrare.....	77
Figură 15 – Exemplu de creare a contului cu succes	78
Figură 16 – Exemplu de personalizare a profilului.....	78
Figură 17 – Exemplu de gestionare a preferințelor alimentare	79
Figură 18 - Exemplu de populare a cămării.....	80
Figură 19 – Exemplu de căutare și adăugare a cantității disponibile în cămară .	81
Figură 20 – Exemplu de actualizare a cantității în cămară	81
Figură 21 – Exemplu de accesare scenariul de folosire a ingredientelor proprii.	82
Figură 22 – Exemplu de pagină cu detalii și sursa externă a unei rețete.....	83
Figură 23 – Exemplu de funcționalități pagina cu detalii a unei rețete	84
Figură 24 – Exemplu de pagină a unei rețete ce poate fi doar planificată	84
Figură 25 – Exemplu de marcarea unei rețete ca fiind finalizată	85
Figură 26 – Exemplu de vizualizare a unei rețete finalizate.....	86
Figură 27 – Exemple de sugestii de rețete culinare.....	86
Figură 28 – Exemplu de răsfoire al listei de rețete, cu filtre	87
Figură 29 – Exemple din secțiunea de rețete salvate	88
Figură 30 – Exemplu de adăugare a unei rețete proprii.....	89
Figură 31 – Exemplu de planificare a unei rețete	89
Figură 32 – Exemplu de calendar și rețete alocate pe parcursul unei zile	90
Figură 33 - Ecranul Cămară și gestionarea listei de cumpărături	91
Figură 34 - Procesul de scanare și identificare a produselor prin cod de bare ...	91
Figură 35 – Ecranele de setări, modificare nume, modificare parolă.....	92

LISTĂ TABELE:

Tabel 1 - Structura principalelor colecții din baza de date Firestore	34
Tabel 2 - Structura principalelor subcolecții din baza de date Firestore	34
Tabel 3 - Structura unui element al listei de ingrediente a unei rețete	35
Tabel 4 - Structura unui element al listei de istoricul al numelui unui utilizator ...	35
Tabel 5 - Structura unui element al listei de ingrediente a unei rețete	35
Tabel 6 - Structura unui element al obiectului de planificare zilnică	36
Tabel 7 - Structura sistemului de rute al directorului principal	42
Tabel 8 - Structura sistemului de rute al directoarelor	43

LISTĂ SECVENȚE DE COD:

Secvență de cod 1 - Inițializarea preferințelor și încărcarea din Firestore	47
Secvență de cod 2 - Salvarea preferințelor în Firestore	48
Secvență de cod 3 - Regulile din consola Firebase.....	48
Secvență de cod 4 – Gestionarea de ingrediente interzise în Firestore	50
Secvență de cod 5 – Detectia prezenței ingredientelor interzise într-o rețetă.....	51
Secvență de cod 6 – Alertă vizuală cu ingredientele interzise detectate	51
Secvență de cod 7 – Actualizarea cantității unui ingredient.....	52
Secvență de cod 8 – Afisarea vizuală a ingredientelor în listă.....	53
Secvență de cod 9 – Afisarea rețetelor din lista de potriviri	54
Secvență de cod 10 – Scalarea cantităților ingredientelor în funcție de porții.....	55
Secvență de cod 11 – Salvarea unei rețete noi în Firestore	57
Secvență de cod 12 – Selectia ingredientelor din listă filtrabilă	58
Secvență de cod 13 – Alocarea unei rețete în planul alimentar.....	59
Secvență de cod 14 – Reguli pentru rețete din Firebase	64

1. INTRODUCERE

1.1. CONTEXTUL GENERAL

În ultimele decenii, digitalizarea a devenit foarte răspândită și a avut un impact semnificativ asupra modului în care oamenii lucrează, se conectează, învață și își trăiesc viața de zi cu zi. O industrie afectată de această schimbare este cea alimentară, care este esențială pentru sănătate și bunăstare. Planificarea meselor și alegera celor mai potrivite rețete pot fi dificile, în special pentru persoanele cu program încărcat sau cu o experiență culinară restrânsă, cu toate că procesul de gătit acasă este de obicei asociat cu o dietă mai sănătoasă și mai echilibrată.

Nevoia de tehnologii digitale care să ofere suport în luarea deciziilor în bucătărie a crescut considerabil ca urmare a acestei realități. Utilizatorii moderni nu mai sunt interesați doar de clasicele rețete, ci caută aplicații mobile interactive, capabile să personalizeze experiența culinară în funcție de nevoile și resursele lor proprii. Potrivit unei analize recente, aplicațiile mobile dedicate acestui context, contribuie la îmbunătățirea stării de sănătate și creșterea calității dietei alimentare prin funcții de monitorizare automată și furnizare de recomandări personalizate [1]. Aceste soluții sunt utile chiar și pentru profesioniști ai domeniului nutriției și sănătății.

Aplicația din acest proiect va avea scopul de a satisface necesitatea persoanelor ce doresc să gătească rapid, ușor și într-o manieră care să funcționeze cu ceea ce este ce au deja acasă. Aplicația se va diferenția de altele similare prin câteva funcționalități esențiale: un modul de gestionare a cămării, un sistem automat de sugestii bazat pe ingredientele disponibile și posibilitatea de a modifica flexibil rețetele, în funcție de preferințele personale sau restricțiile dietei. Acest tip de abordare își propune să vină în sprijinul utilizatorii pentru a menține un program regulat de masă și a accelera procesul decizional.

În plus, din perspectivă psihologică, cercetările arată faptul că, în contexte cu multiple opțiuni, precum este selecția unei rețete culinare, deciziile rapide bazate pe informații contextuale pot fi la fel de eficiente sau chiar superioare celor elaborate prin analiză îndelungată. De exemplu, în studiul [2] s-a prezentat că expunerea utilizatorilor la mai puțină informație, dar mai relevantă, îmbunătățește semnificativ eficiența și consistența alegerilor alimentare în condiții de presiune temporală.

De altfel, odată cu evoluția comportamentului digital, și așteptările legate de alimentație s-au transformat. Dorința de a face alegeri alimentare mai chibzuite este în

creștere, însă constrângerile de timp și abundența de informații de pe internet pot face acest proces epuizant. Oamenii caută soluții mai eficiente, care să le ofere nu doar rețete, ci și ghidaj potrivit stilului lor de viață. Această tendință spre personalizare este deja reflectată în unele aplicații mobile ce testează metode de prezentare a sfaturilor nutriționale într-un mod direct și adaptiv. Un exemplu în acest sens este un studiu recent desfășurat în Australia [3].

1.2. MOTIVАȚIA ALEGERII TEMEI

Alegerea temei aplicației a fost determinată, în principal, de o nevoie personală remarcată în activitatea cotidiană: provocarea de a găsi rețete potrivite în funcție de ingredientele disponibile. De multe ori, procesul de căutare a unei rețete pe internet, pe rețele sociale sau în diverse aplicații culinare se dovedea a fi, pentru mine, consumator de timp și frustrant, mai ales atunci când lipsea un ingredient esențial, chiar dacă rețeta părea promițătoare.

Un alt factor care a influențat alegerea temei a fost lipsa unui mediu digital intuitiv care să asigure sugestii relevante pornind de la ingrediente existente în cămară. Aplicațiile testate anterior erau centrate în principal pe planuri de masă, calorii sau obiective nutriționale, dar nu acopereau eficient nevoia de a transforma ce ai deja în rețete rapide și realizabile.

Deși reducerea semnificativă a risipei alimentare nu a fost obiectivul principal, aplicația din acest proiect oferă indirect acest beneficiu. Aspectul acesta este susținut și de studii recente care arată că aplicațiile mobile orientate spre monitorizarea și gestionarea ingredientelor contribuie la diminuarea risipei în gospodării [4]. Principalul scop rămâne însă economisirea timpului și cauță să ofere o soluție practică pentru cei care vor să ia rapid o decizie în ceea ce privește pregătirea unei rețete culinare.

Tema aleasă reflectă și dorința de a aduce o contribuție utilă în viața de zi cu zi, mai ales într-o societate definită prin ritm alert și lipsă de timp. Funcționalitatea de adăugare a rețetelor de către utilizatori sporește potențialul aplicației și creează premisele unei comunități active și a unei baze de date dinamice, adaptate preferințelor și nevoilor reale ale utilizatorilor.

În concluzie, acest proiect nu va fi doar un exercițiu tehnic, ci o soluție cu aplicabilitate concretă, născută dintr-o experiență directă și adaptată cerințelor actuale ale utilizatorilor moderni.

1.3. OBIECTIVELE APLICAȚIEI

Obiectivul principal al aplicației descrise în această lucrarea este de a oferi utilizatorilor o platformă intelligentă și completă pentru gestionarea întregii experiențe culinare, de la identificarea rețetelor potrivite până la planificarea meselor și monitorizarea progresului în bucătărie. Aplicația este concepută pentru a elimina timpul pierdut în căutarea rețetelor convenabile și pentru a preveni frustrarea generată de lipsa unor ingrediente esențiale, prin implementarea unui sistem automatizat de filtrare și sugestie, bazat pe ingredientele disponibile, preferințele culinare și eventualele restricții dietetice ale utilizatorului. Această abordare va reprezenta o soluție practică pentru gătitul zilnic, mai ales în cadrul unui stil de viață activ, în care deciziile rapide sunt o necesitate.

Personalizarea este unul dintre pilonii centrali ai aplicației. Utilizatorii vor putea defini preferințe alimentare, exclude ingrediente specifice (inclusiv alergeni precum glutenul, lactoza sau nucile) și vor putea opta pentru stiluri de alimentație precum vegetarian sau vegan. Aplicația va detecta automat conflictele alimentare și ajusta rezultatele în consecință, oferind o experiență culinară sigură și relevantă. Această funcționalitate, mai puțin prezentă în aplicațiile existente, adaugă un nivel suplimentar de flexibilitate și control. În plus, va fi încurajată contribuția utilizatorilor, prin adăugarea de rețete proprii. Această componentă sustine colaborarea și are ca scop extinderea constantă a bazei de date și formarea unei platforme culinare active, construită direct de către comunitatea ce o utilizează. Baza de date cu rețete va include atât surse externe (precum TheMealDB), cât și posibilitatea de creare și salvare a rețetelor proprii. Rețete ce vor cuprinde și imagini din galeria personală a utilizatorilor, porționare dinamică, instrucțiuni detaliate și opțiunea de a marca preparatele finalizate, cu scor și notițe personale, asemenea celor din sursele externe.

Aplicația va integra un sistem de planificare a meselor cu calendar interactiv, permitând utilizatorilor să organizeze meniurile pe săptămâni întregi și să monitorizeze progresul în bucătărie în timp real. Interfața intuitivă va asigura o experiență fluidă pe toate dispozitivele, în timp ce sistemul de notificări și urmărire progresului vor motiva utilizatorii să mențină obiceiurile culinare sănătoase. Lista de cumpărături va permite utilizatorilor să mențină o evidență a ingredientelor necesare. Aceasta se va sincroniza în timp real cu stocul din bucătărie, marcând automat ingredientele care sunt deja disponibile, chiar și cele în cantități insuficiente, și evidențierind doar cele care trebuie achiziționate. Funcționalitatea include opțiuni de organizare pe baza rețetei pentru care au fost planificate. Pe lângă aceste funcționalități de bază, aplicația va integra o cămară digitală, în care utilizatorii vor putea adăuga ingrediente. Acest modul va permite corelarea automată între stocul real și rețetele planificate, optimizând astfel gestionarea alimentelor și reducând risipa.

Din punct de vedere tehnic, aplicația va fi dezvoltată cu ajutorul framework-ului React Native, ceea ce îi va permite să funcționeze pe mai multe sisteme de operare mobile, fără eforturi suplimentare semnificative, pentru dezvoltări viitoare. Interfața aplicației este proiectată pentru a fi intuitivă și accesibilă pentru orice tip de utilizator, iar elementele vizuale vor fi tratate cu atenție pentru a susține o experiență de utilizare clară și coerentă. Aplicația este gândită pentru a fi utilizată frecvent, ca un instrument practic în rutina zilnică a utilizatorului, iar toate caracteristicile propuse sunt orientate spre simplitate, utilitate și adaptabilitate. Pentru funcționalitatea de backend, va fi utilizată platforma Firebase, care va gestiona autentificarea, stocarea datelor în Firestore, sincronizarea în timp real și livrarea imaginilor prin integrare cu Cloudinary. Sincronizarea în timp real va contribui la reactivitatea interfeței, iar arhitectura modulară va permite întreținerea și extinderea facilă a aplicației.

Pe termen lung, aplicația va putea fi extinsă prin integrarea de funcționalități sociale (precum partajarea planurilor de masă), notificări inteligente sau sugestii predictibile, bazate pe istoricul utilizatorului. Aceste direcții vor transforma aplicația într-un ecosistem complet de suport pentru decizii alimentare rapide și bine informate, adaptate unui stil de viață modern.

1.4. STRUCTURA LUCRĂRII

Lucrarea este structurată astfel încât să reflecte etapele principale ale dezvoltării aplicației propuse, de la analiza de piață și fundamentarea teoretică, până la implementare, testare și direcțiile viitoare de dezvoltare. Capitolele acestei lucrări urmează o progresie logică și cuprind pașii principali pentru a crea o aplicație funcțională.

În prima parte, sunt prezentate contextul general al aplicațiilor mobile, justificarea temei și obiectivele urmărite, pentru a oferi o imagine clară asupra scopului proiectului. Urmează o analiză a aplicațiilor existente, identificarea publicului țintă și evidențierea elementelor care diferențiază aplicația prezentată de alte soluții similare. Sunt formulate cerințele funcționale și nefuncționale care au ghidat dezvoltarea aplicației.

Secțiunea de proiectare detaliază arhitectura generală a aplicației, interacțiunile interne prin diagrame UML, structura bazei de date și principiile care au stat la baza interfeței grafice. Ulterior, este descris procesul de implementare, prin prezentarea tehnologiilor folosite, a integrărilor externe (API, Cloudinary, Firebase) și a funcționalităților esențiale.

Lucrarea continuă cu prezentarea etapelor de testare, unde sunt evidențiate problemele întâmpinate, soluțiile aplicate și rezultatele obținute în urma validării aplicației.

Este prezentat și un ghid de utilizare, cu scenarii complete de navigare și capturi de ecran reprezentative, utile pentru înțelegerea interfeței din perspectiva utilizatorului final. Această secțiune contribuie la demonstrarea valorii practice a aplicației și facilitează începerea utilizării de către persoane fără experiență tehnică.

Partea finală oferă concluzii asupra proiectului, extrase din procesul de dezvoltare și testare, și discută limitările curente, cum ar fi lipsa unor funcționalități sociale sau integrarea parțială a unor surse externe de rețete. Sunt propuse direcții de dezvoltare ulterioară, printre care se numără personalizarea inteligentă a sugestiilor, notificări automate, partajarea planurilor alimentare și extinderea suportului pentru diverse diete. În cadrul acestui proiect, au fost utilizate instrumente de tip AI (precum ChatGPT) pentru: structurarea inițială a ideilor și capitolelor, generarea de imagini ilustrative și asistență în identificarea și corectarea unor erori de cod, în situațiile în care soluțiile nu au fost identificate prin surse convenționale precum Stack Overflow, GitHub Discussions, Reddit, documentație oficială sau tutoriale de specialitate.

2. ANALIZĂ DE PIAȚĂ

2.1. APLICAȚII SIMILARE EXISTENTE

Există numeroase aplicații mobile de gătit pe piață în prezent, fiecare cu o abordare diferită a procesului de identificare și organizare a rețetelor. În contextul acestei lucrări, care propune o aplicație simplificată, orientată spre folosirea ingredientelor disponibile, sunt relevante câteva exemple de aplicații existente care oferă funcționalități similare sau complementare.

„Yummly” este o aplicație mobilă și platformă web orientată spre personalizarea experienței culinare, care permite utilizatorilor să configureze profiluri alimentare: preferințe, alergii, tipuri de dietă, nivel de dificultate sau durată de preparare. Rețetele sunt selectate automat pe baza acestor preferințe, folosind un sistem de recomandare intern. Aplicația pune la dispoziție opțiuni pentru planificarea meselor, generarea automată a listei de cumpărături. Deși dezvoltată în cadrul companiei Whirlpool, aceasta oferă integrare directă cu anumite modele de cuplare inteligente, care pot fi controlate din aplicație pentru rețetele compatibile.

„Whisk” este o aplicație multiplatformă a cărei principală funcționalitate este opțiunea de a importa rețete din surse diverse, inclusiv site-uri culinare, bloguri sau aplicații sociale, folosind procesare de limbaj natural. Utilizatorii pot crea propriile colecții de rețete, planuri alimentare sau liste de cumpărături. De asemenea, este orientată spre colaborare, permitând partajarea de rețete și planuri între utilizatori. Nu dispune însă de un sistem de generare de rețete pe baza inventarului alimentar curent, funcționând mai degrabă ca un ajutor în organizarea de conținut culinar.

„SuperCook” este o aplicație specializată în generarea de rețete pornind de la ingredientele disponibile ale utilizatorului. Aceasta poate introduce manual sau vocal alimentele pe care le are la dispoziție, iar aplicația returnează rețete compatibile, filtrate în timp real. Baza de date a aplicației este extinsă și agregată din surse externe și alte platforme culinare. Interfața este ușor de utilizat, deși nu pune accent pe componenta vizuală sau socială. „SuperCook” nu permite adăugarea de rețete proprii și nu include personalizare bazată pe preferințe sau alergii, însă compensează prin simplitate și rapiditate în utilizare.

„Cookpad” este o platformă culinară de colaborare, axată pe conținut generat de utilizatori. Oricine își poate crea un cont și poate publica ulterior rețete proprii, însă cu imagini, descrieri și etape detaliate. Aplicația reprezintă o rețea socială pentru pasionații

de gătit, în cadrul căreia rețetele pot fi comentate, salvate și distribuite. „Cookpad” este activă în peste 70 de țări, disponibilă în mai multe limbi, inclusiv română. Baza de date este alcătuită exclusiv din rețete publicate de utilizatori, fără o structură standardizată sau verificare editorială.

2.2. PUBLICUL ȚINTĂ

Aplicația mobilă se adresează în principal persoanelor care doresc să își organizeze alimentația zilnică într-un mod mai eficient, fără a pierde timp căutând rețete sau planificând mese. Publicul țintă este alcătuit din utilizatori care au frecvent acces la dispozitive mobile și sunt deja obișnuiți să integreze tehnologia în rutina lor cotidiană. Acest lucru este valabil mai ales pentru persoanele ce preferă soluții digitale rapide, intuitive și adaptate proprietelor nevoii alimentare, fără a parcurge liste lungi de ingrediente sau metode complicate de preparare. Utilizatorii vizati nu urmăresc calcule nutriționale sau timpul de pregătire, ci soluții aplicabile imediat, în funcție de ce au disponibil în frigider sau cămară.

O categorie esențială a publicului vizat este formată din tinerii adulți cu vârste cuprinse între 25 și 44 de ani, care reprezintă peste 60% dintre utilizatorii de aplicații mobile dedicate nutriției [5]. Aceștia locuiesc adesea singuri sau în cupluri tinere, lucrează cu normă întreagă sau studiază, și sunt expuși constant provocării de a găti rapid, cu resurse limitate, dar într-un mod care să rămână sănătos și variat. În acest context, aplicații precum cea descrisă în această lucrare pot deveni instrumente importante în rutina lor, oferind sugestii imediate pe baza ingredientelor disponibile sau a preferințelor alimentare declarate de ei.

Pe lângă structura demografică, este important și profilul psihologic și digital al publicului țintă. Aplicația este adresată celor care preferă o interfață elementară, dar prietenoasă, recomandări automatizate și procese decizionale simplificate. Acești utilizatori au un comportament digital orientat spre funcționalitate și viteză: nu doresc să parcurgă liste lungi de opțiuni, ci se bazează pe selecții relevante generate automat, în funcție de nevoile momentului. Această abordare este susținută de date recente care arată că utilizatorii petrec în medie 12 minute pe zi în aplicații de nutriție, iar peste 70% dintre ei declară o îmbunătățire clară a obiceiurilor alimentare după trei luni de utilizare [5].

Un segment important al publicului țintă este reprezentat de persoanele care suferă de oboselă decizională, o stare de extenuare mentală cauzată de procesul continuu de a lua decizii, dintr-o multitudine de opțiuni, inclusiv cele legate de alimentație.

După un număr mare de alegeri pe parcursul zilei, calitatea deciziilor scade considerabil, iar mintea caută variante rapid accesibile, nu neapărat cele mai sănătoase. [6]. Într-un context alimentar, acest fapt înseamnă că utilizatorul, obosit, poate alege soluții mai convenabile și mai puțin sănătoase. Aplicația descrisă intervine eficient în acest punct: filtrează opțiunile, oferă sugestii mai reduse, dar personalizate și reduce povara alegerii, facilitând alegeri alimentare coerente.

De altfel, această aplicație a fost concepută pentru a se adapta atât utilizatorilor ocazionali, care intră sporadic în aplicație pentru inspirație sau rețete, cât și celor consecvenți, care folosesc constant funcțiile de sugestii automate și gestionare a cămarii. Această flexibilitate permite aplicației să fie atractivă pentru o varietate largă de utilizatori, cu comportamente digitale și alimentare diverse.

2.3. FUNCȚIONALITĂȚI COMUNE ȘI DIFERENȚIATORI

Aplicațiile mobile din domeniul culinar integrează, în mod uzual, un set de funcționalități standard care vizează accesul rapid la rețete și asistență în organizarea meselor. Printre acestea se numără afișarea unui catalog de rețete, posibilitatea de a parcurge detaliile aferente fiecărei rețete (titlu, listă de ingrediente, pași de preparare), precum și o interfață organizată logic, care permite utilizatorului să navigheze rapid între secțiunile aplicației.

Structura aplicației este intuitivă, bazată pe filele principale: rețete (unde utilizatorul poate explora lista propusă) și profil (unde utilizatorul poate vizualiza informații personale și interacționa cu anumite setări de bază).

Aplicația prezentată în cadrul lucrării implementează aceste funcționalități de bază, dar propune și o serie de diferențiatori funcționali care o poziționează distinct în raport cu aplicațiile similare existente. Un prim element de diferențiere este modulul de gestionare a proprietăților ingrediente, organizat sub forma unei „cămară digitale”. Utilizatorul introduce manual produsele disponibile, iar aplicația folosește aceste date pentru a genera sugestii de rețete compatibile. Acest lucru evită procesul clasic, în care utilizatorul caută o rețetă dorită și ulterior este obligat să caute sau să achiziționeze ingrediente suplimentare. Modelul de operare sugerat ajută la reducerea risipei alimentare și la valorificarea la maximum a resurselor deja existente.

Un al doilea factor de diferențiere este sistemul de recomandări al aplicației. Aceasta sugerează rețete bazate pe ceea ce se află în prezent în cămară și elimină abundența de opțiuni inutile. Aplicația facilitează selecția rapidă a meselor prin minimizarea spațiului de decizie.

Posibilitatea de a adăuga propriile rețete este o altă caracteristică notabilă care permite utilizatorilor să contribuie activ la creșterea conținutului aplicației. Această capacitate este implementată printr-un formular de completare ce salvează rețeta într-o bază de date partajată accesibilă tuturor utilizatorilor. Utilizatorii pot de asemenea să încarce o imagine proprie pentru fiecare rețetă, aceasta fiind stocată automat prin integrarea cu un serviciu extern, care generează un link unic ce va fi atașat rețetei. În acest fel, aplicația nu se limitează la un set închis de rețete, ci devine o platformă de colaborare, în care conținutul este generat parțial de comunitate.

Funcționalitățile dezvoltate în contextul aplicației reflectă o combinație între cerințele obișnuite ale utilizatorilor aplicațiilor de gătit și o serie de soluții adaptate scenariilor reale de utilizare, bazate pe eficiență, adaptabilitate și ușurință în interacțiune.

2.4. CERINȚE FUNCȚIONALE

Cerințele funcționale vor viza comportamentul direct al aplicației în raport cu utilizatorul final și vor defini modul în care aceasta trebuie să reacționeze la acțiunile întreprinse în interfață. Ele stabilesc funcțiile esențiale pe care aplicația urmează să le implementeze.

În primul rând, aplicația va include un mecanism de autentificare care va permite accesul personalizat al utilizatorilor. Autentificarea se va realiza prin adresă de e-mail și parolă, iar conturile vor fi gestionate cu ajutorul unui serviciu extern pentru a asigura securitatea și manevrarea ulterioară a unui volum mare de utilizatori [7]. După autentificare, utilizatorul va avea acces la un profil individual, prin intermediul căruia va putea vizualiza și gestiona date proprii.

Aplicația va permite accesul la o colecție de rețete culinare, care vor putea fi vizualizate într-o listă generală. Pentru fiecare rețetă, se vor afișa informații detaliate, precum titlul, imagine, zona de origine, categoria, lista de ingrediente, alături de cantitățile corespunzătoare și instrucțiunile de preparare. În plus, utilizatorul va putea adăuga rețete proprii în baza de date, prin intermediul unui formular dedicat. Această funcționalitate presupune completarea unor câmpuri obligatorii cu detaliile rețetei. Utilizatorul va avea și opțiunea de a atașa o imagine reprezentativă a preparatului, ce va putea fi selectată din galeria dispozitivului mobil. Imaginea va fi procesată local și transmisă automat către un serviciu extern de găzduire media, care generează un link unic, apoi stocat în baza de date alături de restul informațiilor rețetei. În cazul în care utilizatorul nu va încărca o imagine personalizată, aplicația va atribui una dintre imaginile implicate disponibile în sistem.

Aplicația va oferi posibilitatea introducerii, modificării și ștergerii produselor din cămara personală, care va fi asociată fiecărui cont de utilizator. În baza acestor date, sistemul va analiza conținutul disponibil și va genera sugestii de rețete care pot fi preparate complet sau parțial, în funcție de potrivirea ingredientelor. Algoritmul de selecție va trata cu prioritate rețetele relevante și va reduce numărul opțiunilor afișate, pentru a facilita luarea unei decizii rapide.

Aplicația va fi structurată pe secțiuni funcționale majore, accesibile printr-un sistem de navigare cu pagini principale și alte opțiuni. Navigarea între acestea va fi permanent vizibilă și ușor accesibilă, astfel încât interacțiunea cu interfața să fie coerentă și intuitivă. Toate datele introduse de utilizator, inclusiv cele legate de autentificare, ingrediente și rețete proprii, vor fi salvate prin intermediul unui sistem de stocare în timp real, cu sincronizare automată.

2.5. CERINȚE NON-FUNCȚIONALE

Cerințele nefuncționale definesc proprietățile generale ale sistemului care nu privesc comportamentul specific în interacțiunea cu utilizatorul, ci caracteristici precum performanța, securitatea, disponibilitatea, portabilitatea și experiența generală de utilizare.

Aplicația descrisă va trebui să asigure un timp de reacție corespunzător așteptărilor normale pentru aplicații mobile. Operațiunile de bază, precum încărcarea rețetelor sau actualizarea listei de ingrediente, vor fi gestionate asincron și vor include mecanisme vizuale de așteptare. Comportamentul de încărcare va fi optimizat pentru ca interacțiunile să nu depășească intervale percepute ca lente de către utilizator. Performanța efectivă va fi influențată de viteza conexiunii și de răspunsul serviciilor de stocare. În special, pentru gestionarea imaginilor asociate rețetelor, aplicația va folosi serviciul Cloudinary, care oferă un mecanism eficient și rapid de încărcare, stocare și livrare a fișierelor media. Interfața va oferi feedback vizual în timpul încărcării imaginilor pentru a menține claritatea interacțiunii și a reduce incertitudinea utilizatorului.

Din punct de vedere al securității, aplicația va utiliza un serviciu central de administrare a conturilor, gestionat în totalitate de Google. Acesta va oferi astfel un mecanism standardizat de verificare a identității, cu protecție la acces neautorizat. Accesul la datele utilizatorului va fi filtrat prin reguli explicite, pentru a preveni expunerea datelor personale. Persistența datelor se va realiza în timp real, prin conectarea aplicației la o bază de date aflată pe internet, adică Firebase Firestore. În lipsa unei conexiuni stabile, aplicația nu va oferi funcționalitate completă offline, însă sincronizarea se va relua

automat la reconectare. În această etapă, nu este prevăzut un mecanism local de cache. Linkurile generate de Cloudinary vor fi gestionate fără a include date personale identificabile și vor fi folosite exclusiv pentru afișarea publică a imaginilor în aplicație.

Portabilitatea este asigurată prin utilizarea platformei React Native împreună cu Expo, ceea ce permite distribuția aplicației atât pe sisteme Android, cât și iOS, fără diferențe semnificative de comportament. Aplicația este concepută să funcționeze uniform pe diverse dimensiuni de ecran și să ofere o aranjare flexibilă, pe baza principiilor actuale de dispunere vizuală adaptivă. Aplicația va fi structurată modular, ceea ce va permite adăugarea ulterioară de funcționalități fără restructurări majore ale codului de bază.

În ansamblu, cerințele nefuncționale urmăresc să susțină stabilitatea aplicației, accesibilitatea multiplatformă și protecția datelor utilizatorilor, fără a compromite simplitatea și viteza de utilizare.

3. PROIECTAREA APlicațIEI

3.1. ARHITECTURA GENERALĂ

Arhitectura pe care este construită această aplicație mobilă este de tip client-cloud, în care logica de interfață și procesare rulează local pe dispozitivul mobil, iar datele persistente și funcționalitățile backend sunt gestionate prin servicii cloud. Această structură permite o sincronizare în timp real între dispozitivul utilizatorului și infrastructura de date și menține în același timp o separare clară între nivelul de prezentare, logica de aplicație și gestionarea resurselor externe.

În cadrul proiectului prezentat, folosesc framework-ul React Native, în combinație cu platforma Expo pentru facilitarea procesului de dezvoltare și testare multiplatformă. Rutarea este implementată prin biblioteca Expo Router, care permite definirea ecranelor și navigarea între acestea printr-o structură de directoare bazată pe convenții, inspirată din modelul utilizat de Next.js. Fiecare ecran corespunde unui fișier .tsx din directorul /app, iar rutele sunt generate automat în funcție de această structură.

Pentru logica de backend, folosesc Firebase, o platformă BaaS (Backend-as-a-Service) care oferă servicii de autentificare, stocare de date și reguli de securitate. Datele sunt gestionate în Firestore, o bază de date NoSQL organizată pe colecții și documente, cu suport pentru sincronizare bidirectională și monitorizări în timp real, prin onSnapshot (o metodă a obiectelor DocumentReference și CollectionReference din SDK-ul Firestore). Autentificarea utilizatorilor este realizată prin Firebase Authentication, iar structura datelor este modulară, fiecare utilizator având propriile subcolecții.

Imaginiile asociate rețelilor se încarcă în Cloudinary, un serviciu extern dedicat gestionării și optimizării conținutului media. Imaginea este selectată local, transformată și transmisă printr-o cerere de tip POST, iar link-ul rezultat este salvat în Firestore. În această manieră, aplicația evită stocarea directă a fișierelor mari în baza de date și permite încărcarea rapidă a conținutului vizual.

Am structurat cod modular, cu separare clară între componente funcționale și cele vizuale. Fiecare ecran este implementat într-un fișier care conține logica interactivă, în timp ce stilurile sunt definite separat, în fișiere dedicate. Interacțiunile sunt implementate în mod asincron (o operațiune nu blochează execuția principală a programului) și reactiv (interfața răspunde imediat la interacțiunile utilizatorului și la schimbările de date, fără să pară blocată). Aceste comportamente sunt asigurate prin

aplicarea bunelor practici de dezvoltare React Native, prin valorificarea API-urilor sale esențiale (hooks), cum ar fi useEffect pentru gestionarea efectelor secundare, useState pentru starea locală, useLocalSearchParams pentru datele rutei, și useContext pentru starea globală, alături de o organizare explicită a stărilor și a actualizărilor interfeței.

Accesul la Firestore este abstractizat în funcții și servicii auxiliare, care se ocupă de operațiile de citire, scriere sau actualizare a datelor. Această abordare permite separarea logicii de rețea de componentele de interfață și facilitează testarea, depanarea și reutilizarea codului. În același timp, am inclus componente vizuale pentru feedback imediat, cum ar fi ActivityIndicator (componentă predefinită din React Native care indică o stare de încărcare sau procesare), mesaje de confirmare de tip pop-up, alerte sau ferestre modale de informare și confirmare de acțiune. Prin integrarea acestor elemente vizuale, am urmărit să ofer utilizatorului o înțelegere clară a stării aplicației și să consolidez încrederea în sistem. Regulile de acces sunt configurate pentru a permite fiecărui utilizator să vizualizeze și să modifice doar propriile date. Accesul se bazează pe identificatorul unic al utilizatorului (UID) autentificat, iar toate scriurile și citirile sunt filtrate în funcție de acesta.

Navigarea este organizată ierarhic, prin gruparea ecranelor în directoare tematicice (precum recipes-folder, meal-planner, settings-folder), iar aspectul vizual comun este gestionat prin fișiere _layout.tsx, care definesc comportamentul fiecărui grup de rute. Prin această arhitectură, proiectul reșește să combine flexibilitatea tehnologiilor frontend moderne cu avantajele unei infrastructuri cloud scalabile, asigurând totodată o experiență fluentă pentru utilizator și un cod clar structurat.

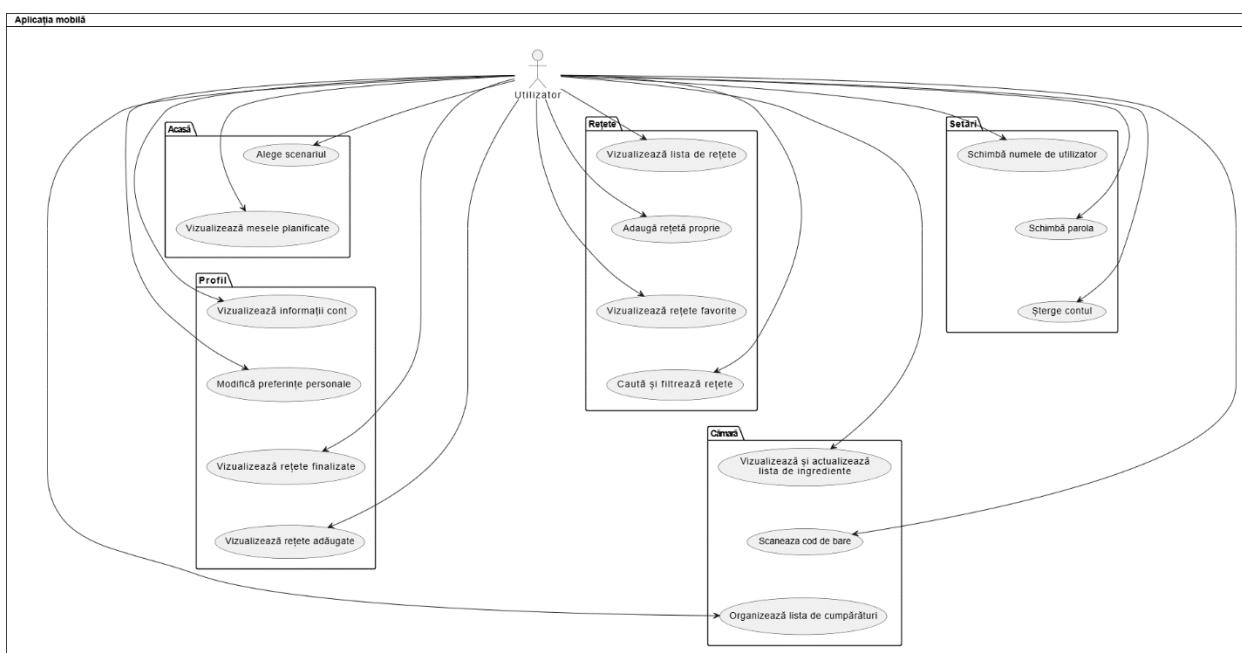
3.2. DIAGRAME UML

Modelarea conceptuală reprezintă o etapă importantă în procesul de proiectare software, întrucât facilitează înțelegerea structurii interne, a comportamentului sistemului și a relațiilor dintre componente, înaintea fazei de implementare. Pentru această etapă, este frecvent utilizat limbajul de modelare UML (Unified Modeling Language), un standard consacrat pentru reprezentarea abstractă și tehnologic-independentă a sistemelor software [8].

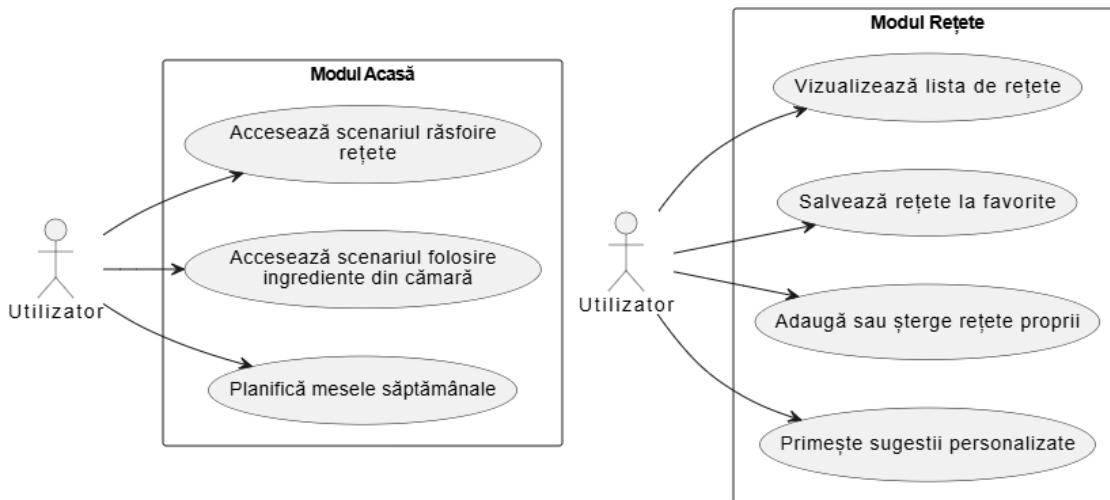
Diagramele UML contribuie la identificarea cerințelor funcționale, la evidențierea interacțiunilor dintre utilizator și sistem, precum și la structurarea clară a claselor și componentelor implicate. Utilizarea acestor reprezentări grafice are scopul de a sprijini validarea cerințelor și de a furniza un cadru extensibil pentru evoluția aplicației [9].

3.2.1. DIAGRAME DE CAZURI DE UTILIZARE

Diagrama de cazuri de utilizare de mai jos redă interacțiunile directe dintre utilizator și modulele principale ale aplicației prezentate. Structura este organizată pe patru componente funcționale: Acasă, Rețete, Cămară și Profil. Fiecare componentă este asociată cu două scenarii de utilizare esențiale, care reflectă funcționalitățile disponibile în interfața aplicației. Reprezentarea respectă standardul UML și oferă o privire de ansamblu asupra comportamentului sistemului din perspectiva utilizatorului.

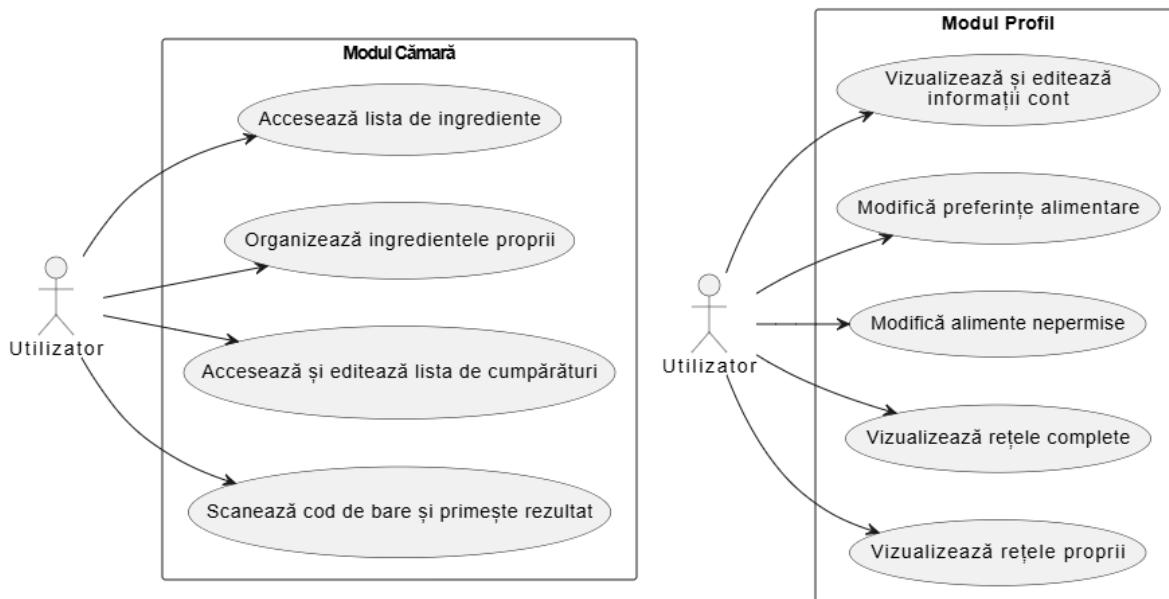


Figură 1 - Diagrama de cazuri de utilizare cu modulele principale



Figură 2 - Diagramme de cazuri de utilizare pentru modulele Acasă și Rețete

Modulul Acasă servește ca punct de pornire pentru utilizator, oferind acces rapid la conținut personalizat și funcționalități esențiale. Modulul Rețete permite consultarea și gestionarea rețetelor, fiind componenta principală a aplicației în ceea ce privește conținutul culinar. Diagramele din figura 2 descriu cazurile de utilizare aferente acestor două secțiuni.



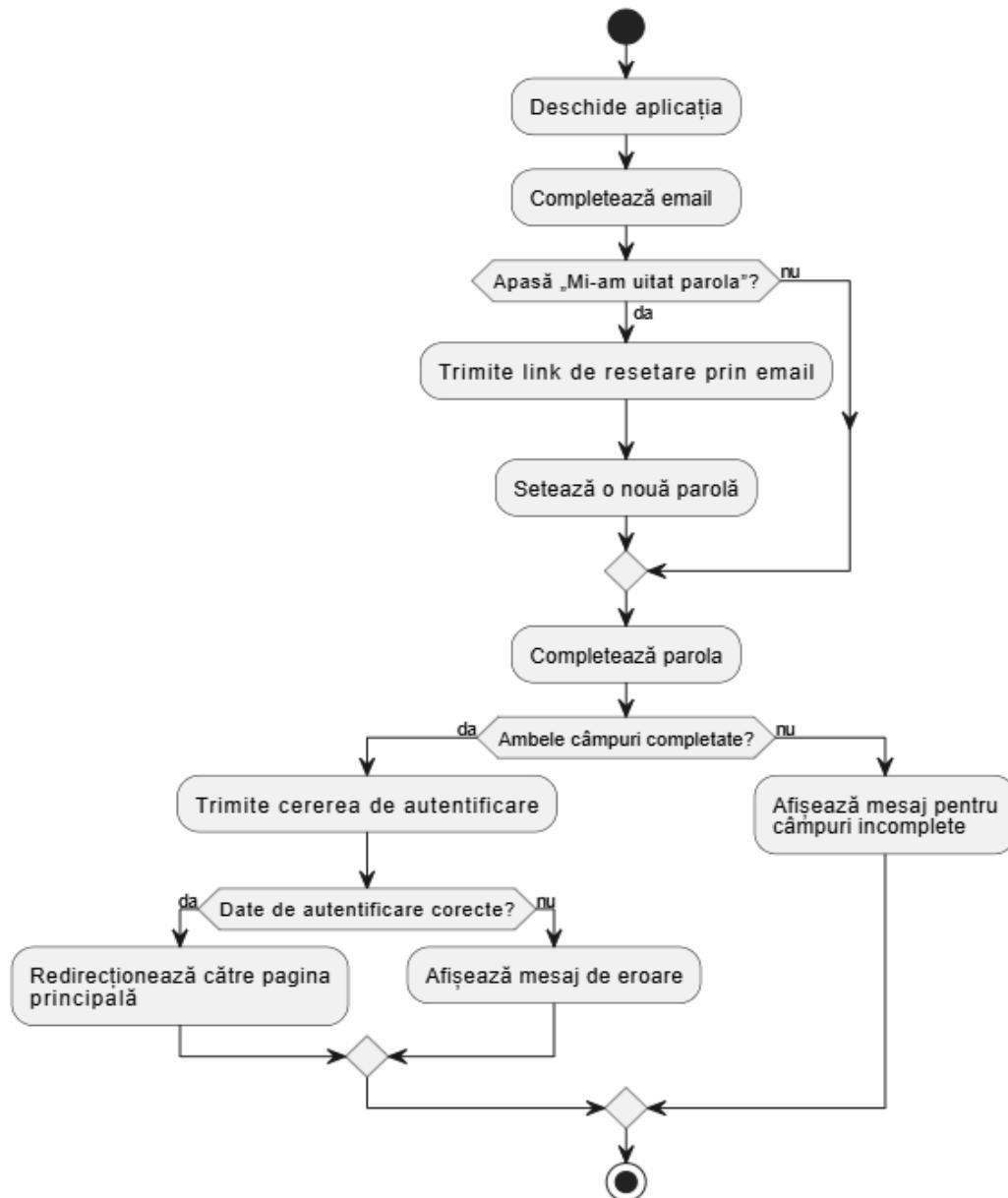
Figură 3 - Diagramme de cazuri de utilizare pentru modulele Cămară și Profil

Modulele Cămară și Profil completează funcționalitatea aplicației prin opțiuni de gestionare a resurselor și personalizare. Modulul Cămară este dedicat introducerii și actualizării ingredientelor disponibile, iar Profilul oferă posibilitatea administrării preferințelor și a informațiilor de cont. Diagramele din figura 3 ilustrează scenariile de utilizare corespunzătoare acestor două componente.

3.2.2. DIAGRAAME DE ACTIVITATE

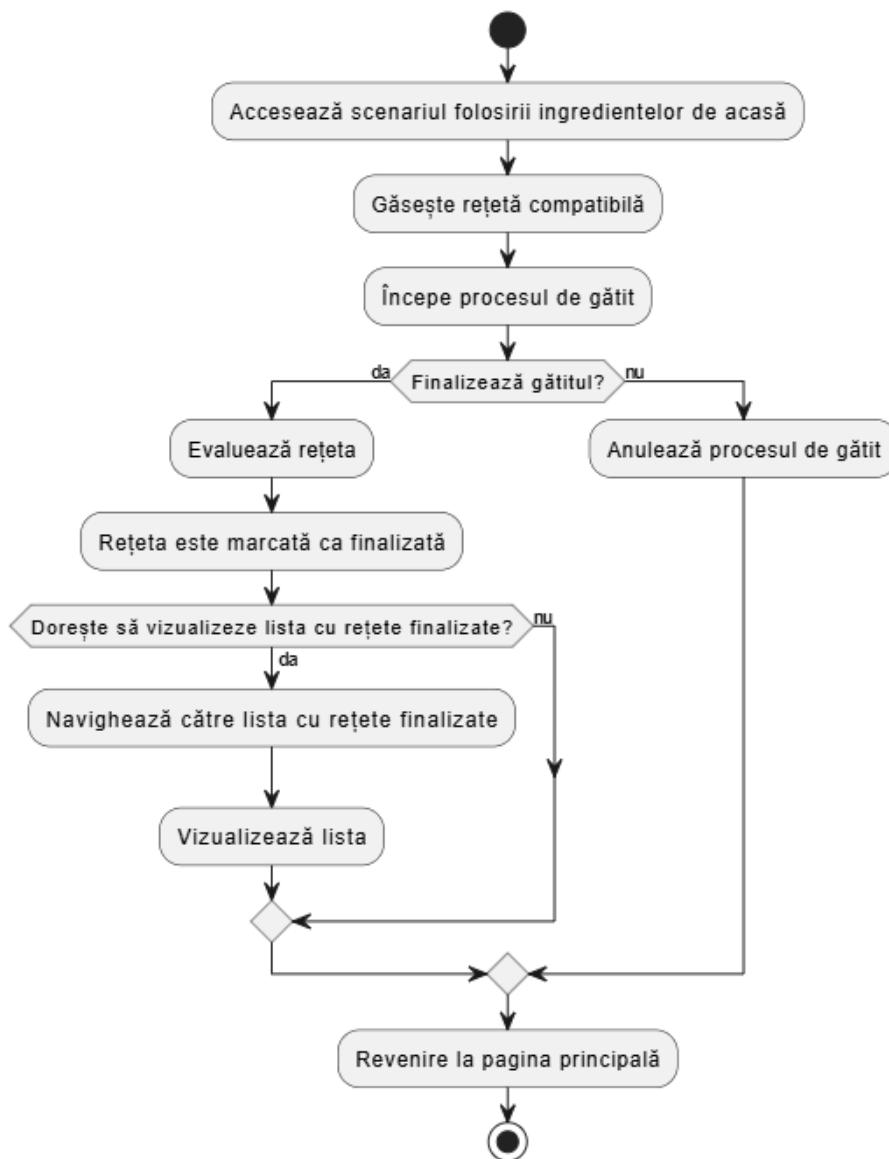
Diagramele de activitate sunt folosite pentru a modela fluxul logic de acțiuni care au loc într-un anumit proces, punând accent pe succesiunea operațiilor, deciziilor și ramificărilor posibile. Acestea sunt utile pentru a evidenția comportamentul dinamic al aplicației în contexte concrete de utilizare, într-o formă vizuală clară și structurată [10].

În cadrul proiectului, am utilizat aceste diagrame pentru a reprezenta câteva scenarii, mai importante, precum autentificarea utilizatorului, adăugarea unei rețete noi și parcurgerea completă a unei rețete recomandate. Fiecare diagramă surprinde acțiunile inițiate de utilizator, verificările condiționale și interacțiunile cu sistemul, pentru a fi un sprijin în înțelegerea detaliată a comportamentului aplicației în raport cu cerințele funcționale.



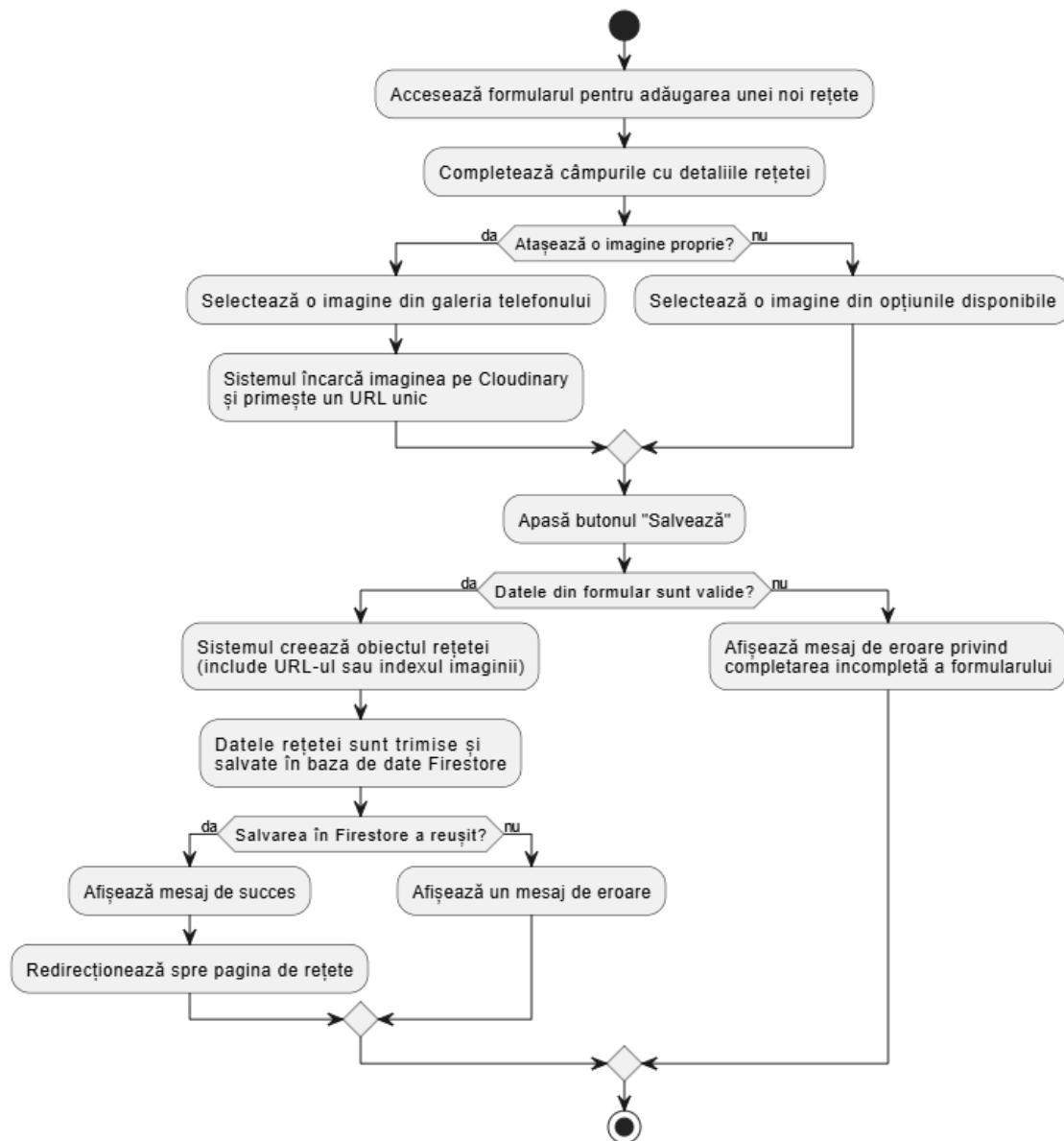
Figură 4 - Diagrama de activitate a procesului de autentificare

Fluxul modelat include validarea locală a câmpurilor, trimitera cererii de autentificare către Firebase, tratarea rezultatului (autentificare reușită sau eșuată), precum și afișarea mesajelor de eroare sau redirecționarea către interfața principală. Diagrama reflectă atât verificările obligatorii cât și mecanismul de feedback vizual implementat.



Figură 5 - Diagrama de activitate a procesului de gătit

Figura 5 modelează parcursul unui utilizator care a selectat o rețetă complet compatibilă cu ingredientele existente în cămară. Procesul include deschiderea paginii de rețetă, parurgerea pașilor de gătit, marcarea ca finalizată, și completarea unui scurt formular de evaluare (notă și mențiuni). Dacă rețeta este completată sistemul înregistrează rețeta în secțiunea dedicată preparatelor finalizeze, altfel procesul este anulat.

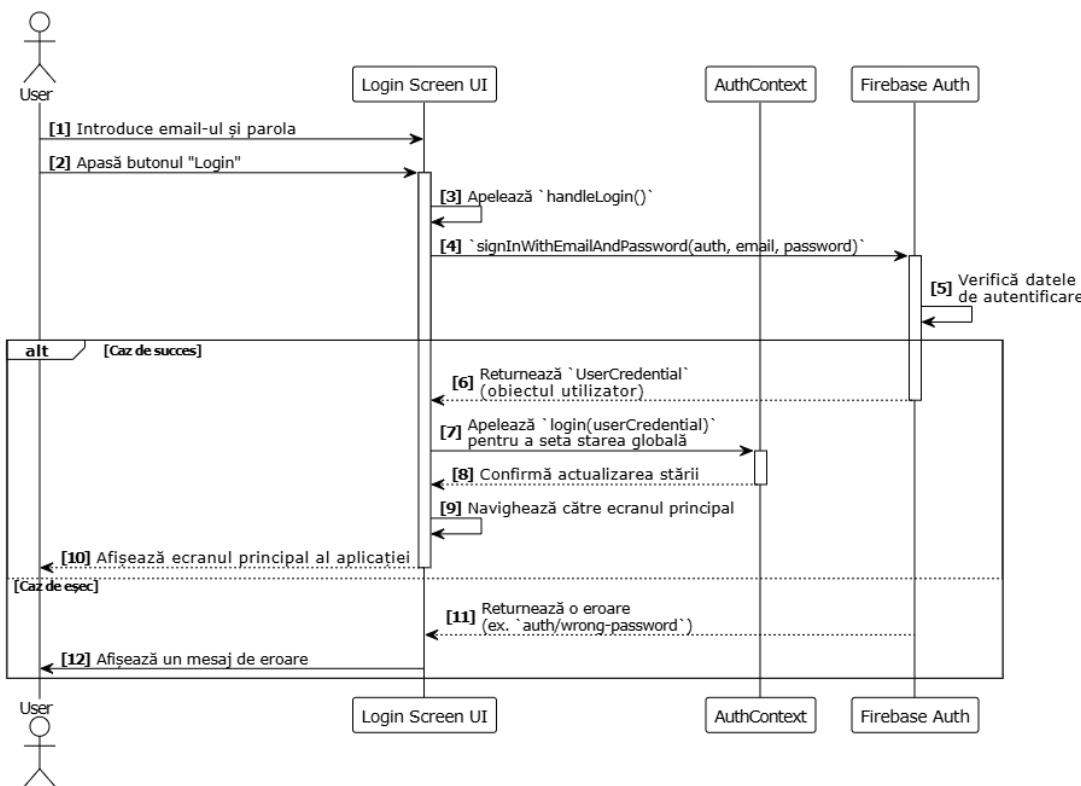


Figură 6 - Diagrama de activitate a procesului de adăugare a unei rețete noi

Fluxul detaliază interacțiunea acestuia cu formularul (completarea câmpurilor, selecția sau încărcarea unei imagini din galerie), transmiterea datelor către Firebase Firestore și a imaginii către Cloudinary, urmată de confirmarea succesului și redirecționarea utilizatorului spre lista de rețere în care va apărea și cea recent adăugată. În caz de eroare sau date incomplete, sistemul semnalează problema și solicită corectarea.

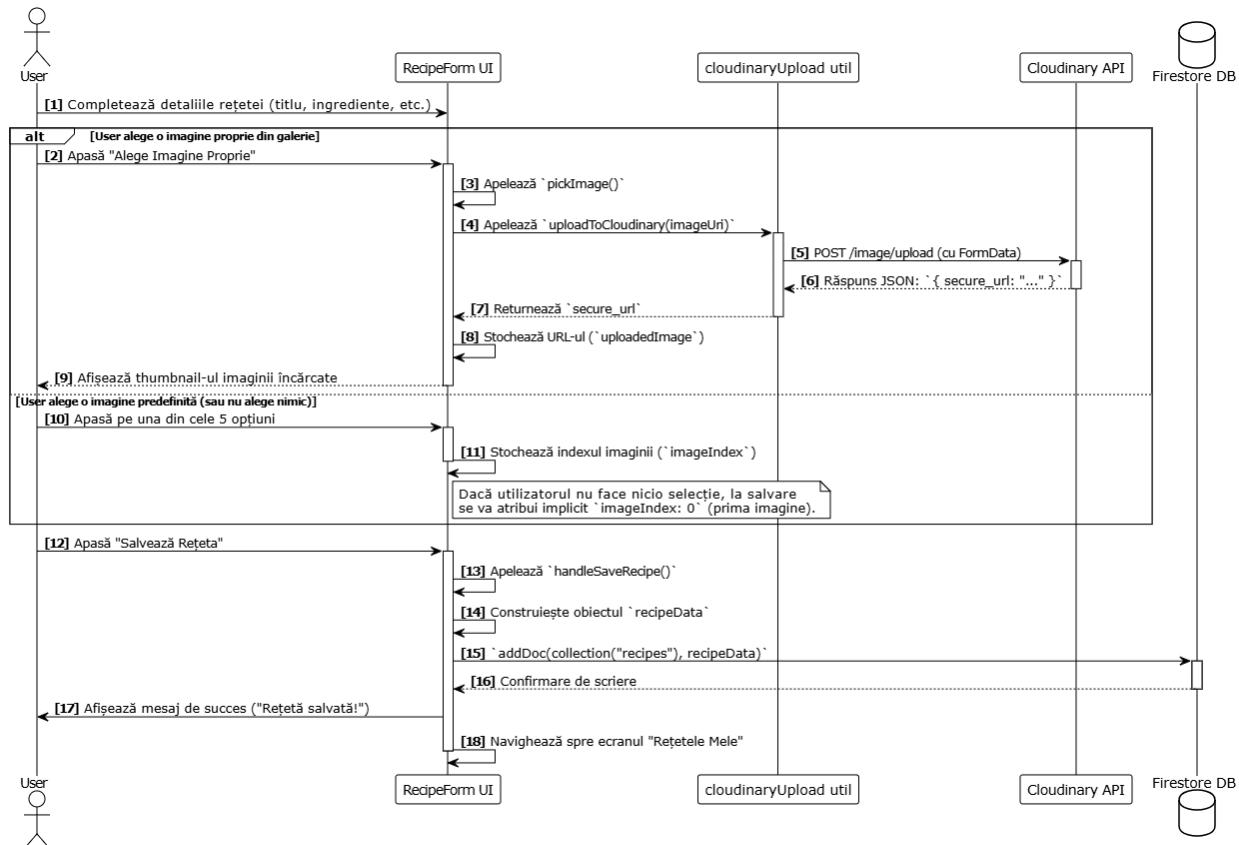
3.2.3. DIAGRAME DE SECVENTĂ

Diagramele de secvență reprezintă un instrument fundamental în proiectarea sistemelor software, cu rolul de a descrie interacțiunile dintre actorii unui sistem și componente sale, în ordinea cronologică în care au loc. Aceste modele evidențiază schimbul de mesaje dintre entități, ciclul de viață al unei operațiuni și dependențele dintre nivelurile aplicației. Utilizarea diagramelor de secvență contribuie la identificarea rapidă a ambiguităților din cerințe, reduce semnificativ riscul unor implementări greșite și oferă o documentație clară, utilă atât dezvoltatorilor, cât și părților interesate [11]. În contextul dezvoltării unor aplicații mobile, ele sprijină alinierea dintre interfață, logica de business și infrastructura backend.



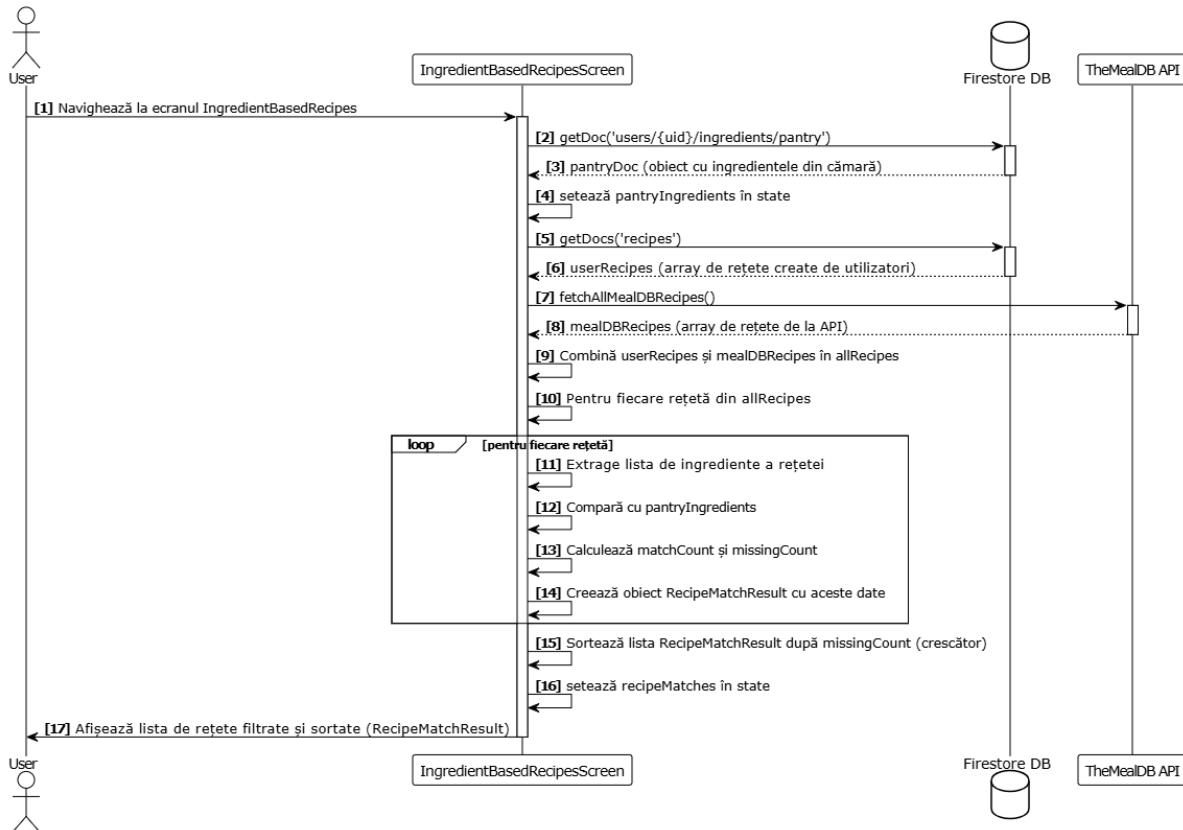
Figură 7 - Diagrama de secvență a autentificării utilizatorului

Diagrama din figura 7 prezintă procesul de autentificare a utilizatorului, inclusiv scenariul în care acesta își uită parola. După introducerea emailului, utilizatorul poate solicita resetarea parolei, primind un link prin email. După introducerea datelor se trimite cererea de autentificare. Sistemul validează datele și utilizatorul este redirecționat către pagina principală, dacă sunt corecte. În caz contrar, aplicația semnalează eroarea și solicită reintroducerea datelor.



Figură 8 - Diagrama de secvență a adăugării unei rețete

În figura 8, este prezentat fluxul de adăugare a unei noi rețete în aplicație. Utilizatorul completează formularul cu detaliile rețetei și are posibilitatea de a selecta o imagine personalizată din galerie sau una prestabiliță. În cazul unei imagini proprii, aceasta este încărcată automat pe platforma Cloudinary, iar sistemul obține un link unic care va fi salvat în baza de date. După apăsarea butonului „Salvează”, datele sunt validate și, dacă formularul este complet, obiectul rețetei este trimis către Firestore. Aplicația afișează apoi un mesaj de confirmare sau eroare, în funcție de rezultatul operației.



Figură 9 - Diagrama de secvență a generării sugestiilor

Diagrama din figura 9 ilustrează procesul de generare a sugestiilor de rețete în funcție de ingredientele din cămară. După accesarea ecranului dedicat, aplicația preia ingredientele utilizatorului din Firestore și rețetele disponibile, atât cele proprii cât și cele din API-ul TheMealDB. Fiecare rețetă este analizată comparativ, calculându-se numărul de ingrediente potrivite și cel lipsă. Rezultatele sunt sortate și afișate utilizatorului sub formă unei liste filtrate, ordonate după relevantă.

3.3. ASPECTUL INTERFEȚEI

3.3.1. NAVIGARE PRIN FILE

Navigarea principală a aplicației se face prin intermediul barei de file din partea de jos a ecranului. Aceasta este împărțită în patru secțiuni: Acasă, Rețete, Cămară și Profil,. Acest model de navigare a fost selectat pentru a oferi utilizatorului o experiență simplă, fără să fie necesare căutări sau pași mulți pentru a accesa funcționalitățile importante.

Tipul acesta de navigare este specific aplicațiilor mobile moderne, în special pentru aplicațiile mobile cu secțiuni principale limitate. Conform studiilor, utilizarea a trei până la cinci file scurtează timpul necesar utilizatorilor pentru a se familiariza cu o aplicație și o face mai ușor de utilizat [12]. Acest lucru este valabil mai ales pentru aplicațiile ce sunt proiectate pentru a fi utilizate frecvent, dar pe perioade scurte de timp.,.

În plus, plasarea acestor butoane de navigare în partea inferioară a ecranului, acolo unde degetul mare le poate accesa cel mai ușor și rapid, promovează o utilizare mai facilă și mai naturală, conform recomandărilor [13]. Consider că această amplasare este optimă, motiv pentru care am ales-o, deoarece permite utilizarea chiar și cu o singură mână.

Un alt aspect important este legat de claritatea vizuală. Fiecare filă este însoțită de o pictogramă și de un text scurt, ceea ce ajută utilizatorul să recunoască imediat funcțiile relevante. Astfel, aplicația nu doar că este ușor de folosit, dar oferă și o experiență predictibilă, ceea ce este un avantaj pentru persoanele grăbite sau care nu sunt foarte familiarizate cu tehnologia. Acest lucru este susținut și de un studiu care arată că utilizatorii preferă aplicațiile cu navigare simplă prin file față de cele cu meniuri complexe sau chiar ascunse [14].

În concluzie, sistemul de navigare este conceput pentru a fi ușor de înțeles, intuitiv și eficient pentru sarcinile zilnice.

3.3.2. ELEMENTE VIZUALE

Interfața aplicației este construită pe baza unor principii moderne de design vizual și funcțional, adaptate mediului mobil și platformei React Native. Designul urmărește claritatea funcțională, lizibilitatea conținutului și o estetică coerentă care să reflecte domeniul culinar într-un mod plăcut și accesibil.

O proporție semnificativă, 62-90%, din percepția inițială asupra unui produs se bazează pe culoare [15]. Aplicația folosește o paletă dominată de nuanțe de verde, asociată cu natura și prospetimea, atribute relevante într-un context culinar. De

asemenea, verdele activează sistemul nervos simpatetic, inducând o stare de calm și încredere. Aceste nuanțe sunt aplicate consecvent în carduri, butoane și fundaluri, pentru a contribui la claritate și coerentă vizuală.

Conform cercetărilor recente privind comportamentul vizual în interfețele mobile, unele tipuri de componente atrag semnificativ mai mult atenția utilizatorilor decât altele. S-a constatat că elementele cel mai frecvent fixate sunt imaginile (22,4%), texte (15,7%), grupurile de text (15,2%), cardurile (13,3%) și pictogramme (12,8%), aceste categorii dominând distribuția atenției în timpul utilizării unei aplicații mobile [16]. În acest proiect, observația este reflectată în mod direct prin folosirea frecventă a cardurilor de rețete, a textelor scurte și clare, precum și a imaginilor reprezentative.

Icoanele din aplicație sunt gestionate prin biblioteca @expo/vector-icons, ce oferă o gamă variată de simboluri standardizate și recunoscute (de exemplu pentru navigare, salvare sau filtrare). Acestea sunt folosite consecvent în bara de file, în anteturi și în meniuri contextuale, contribuind la o navigare predictibilă și la reducerea ambiguității în interacțiune.

În completare, am integrat și imagini proprii pentru a reda specificul culinar, cum sunt imaginile reprezentative pentru butoane, secțiuni, rețete sau mesaje informative. Această contribuție personală vizuală sprijină diferențierea aplicației și creează o identitate vizuală unitară.

3.4. MODELUL BAZEI DE DATE

Pentru o înțelegere mai clară a organizării datelor în Firebase Firestore am prezentat principalele colecții, structura documentelor și câmpurile din baza de date. Logica modulară și felul în care fiecare entitate este gestionată sunt ilustrate prin următoarele tabele.

Colecție (document)	Cheie	Câmp	Tip	Exemplu Valoare	Descriere
recipes	recipId	area	string	“Italian”	Zonă origine rețetă
		authorId	string	“3C4tZL0...”	ID utilizatorului autor
		authorName	string	“User Android”	Nume autorului
		category	string	“Dessert”	Categorie rețetei
		createdAt	string	“2025-04-21 T10:39:37.90...”	Data creare rețetei
		imageIndex	number	0	Index opțiune imagine
		ingredients	array	Tabel 3	Listă ingrediente
		instructions	array	[“Prepare ingredients. , “Mix”]	Instrucții preparare
		title	string	“Tiramisu”	Titlu rețetă
		uploadedImage	string	“https://res.cloudinary.com...”	URL imagine rețetă
users	userId	bannedIngredients	array	[“almond milk, “basil”]	Ingrediente interzise
		nameHistory	array	Tabel 4	Istoric nume
		displayName	string	“User Android”	Nume utilizator actual

Tabel 1 - Structura principalelor colecții din baza de date Firestore

Subcolecție	Cheie	Câmp	Tip	Exemplu Valoare	Descriere
users/{userId}/completedRecipes	recipId	completedDate	timestamp	“2025-04-22”	Dată completare
		id	string	“52892_1750...”	ID rețetă completată (unic în această listă, deși o rețetă poate apărea de mai multe ori)
		image	string	“https://www.themealdb.com...”	URL imagine rețetă
		ingredients	array	Tabel 5	Ingrediente rețetă
		instructions	string	“Mix all ingredients and bake.”	Instrucții rețetă
		isUserRecipe	boolean	false	Rețetă proprie
		mention	string	“Good”	Mențiuni
		rating	number	4	Scor
		recipId	string		ID rețetă
users/{userId}/dateMealPlans	date	scalingFactor	number	0.5	Factor scalare
		title	string	“Honey Teriyaki Salmon”	Titlu rețetă
		date	string	“2025-05-10”	Data planului
users/{userId}/favorites	recipId	meals	map	Tabel 6	
		id	string	“52892”	ID rețetă favorită
		image	string	“https://...”	URL imagine rețetă
		savedAt	string	“2025-05-21 T09:35:23.08...”	Dată salvarei
		title	string	“Banana Pancakes”	Titlu rețetă favorită
users/{userId}/ingredients	ingredientName	measurementType	string	“weight”	Tip măsurare
		quantity	string	“40”	Cantitate ingredient
users/{userId}/preferences	“dietary”	dairyFree	boolean	false	Fără lactate
		glutenFree	boolean	false	Fără gluten
		nutFree	boolean	true	Fără nuci
		vegan	boolean	false	Vegan
		vegetarian	boolean	false	Vegetarian
users/{userId}/shoppingList/items	itemName	“profile”	profileImage	“f-ok.png”	Imagine profil selectată
		category	string	“Spices”	Categorie
		checked	boolean	false	Bifare
		name	string	“almonds”	Nume ingredient
		quantity	string	“85g”	Cantitate și unitate
		recipeName	string	“Honey Yogurt Cheesecake”	Nume rețetă asociată

Tabel 2 - Structura principalelor subcolecții din baza de date Firestore

Pentru anumite câmpuri cu tipul „array” sau „map”, conținutul intern este detaliat separat în tabelele următoare (Tabel 3, Tabel 4, Tabel 5, Tabel 6), unde sunt prezentate structurile. Aceste tabele suplimentare oferă o imagine completă asupra modului în care datele sunt organizate în baza de date și accesate în aplicație.

Câmp	Tip	Exemplu Valoare	Descriere
name	string	“Egg”	Numele ingredientului
quantity	string	“2”	Cantitatea
unit	string	“pcs”	Unitatea de măsură

Tabel 3 - Structura unui element al listei de ingrediente a unei rețete

Câmp	Tip	Exemplu Valoare	Descriere
newName	string	“Alex”	Noul nume
oldName	string	“User Android”	Numele anterior
timestamp	timestamp	“2025-06-16T...”	Data schimbării numelui

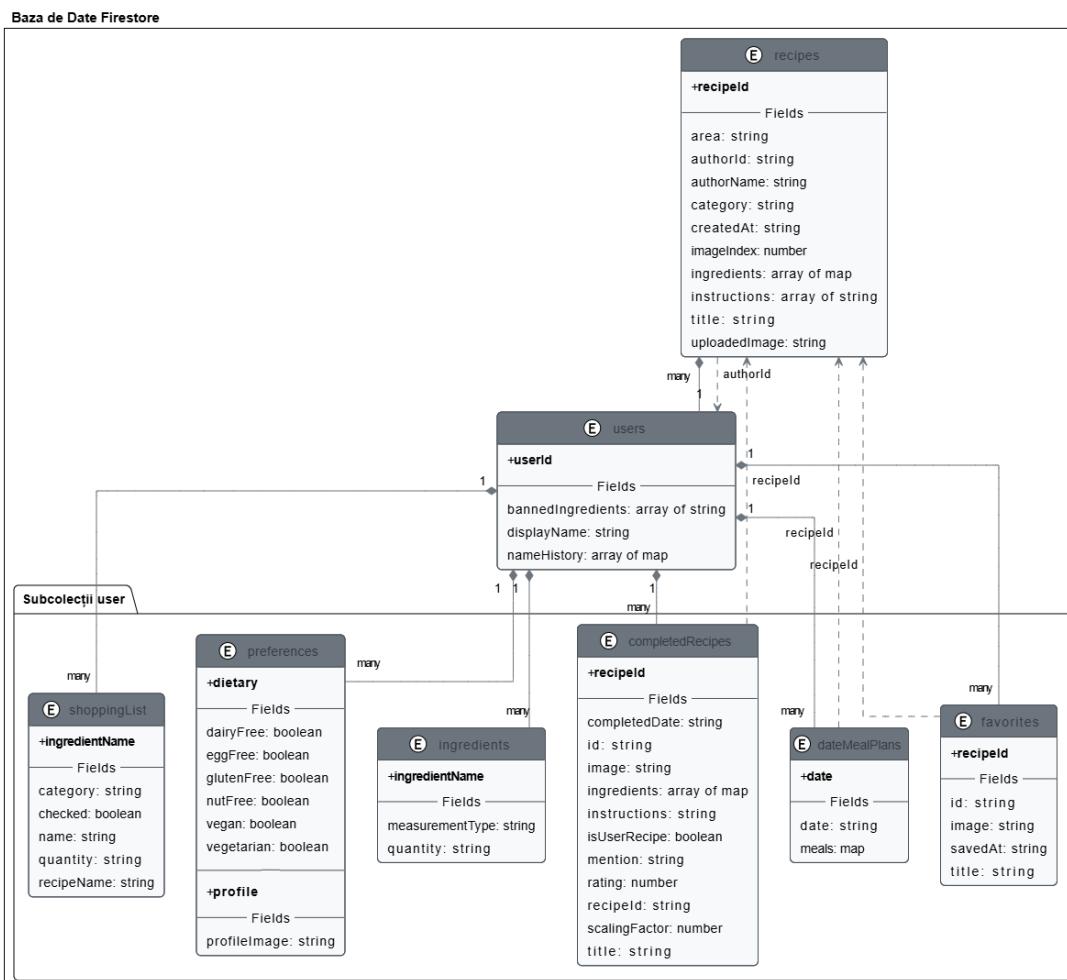
Tabel 4 - Structura unui element al listei de istoricul al numelui unui utilizator

Câmp	Tip	Exemplu Valoare	Descriere
name	string	“Egg”	Numele ingredientului
quantity	string	“2”	Cantitatea folosită
unit	string	“pcs”	Unitatea de măsură
category	string	“Side”	Categoria ingredientului
measurementType	string	“weight”	Tipul de măsurare
recipeName	string	“Omelette”	Numele rețetei asociate

Tabel 5 - Structura unui element al listei de ingrediente a unei rețete

Cheie	Tip cheie	Câmp	Tip câmp	Descriere
breakfast	map	id	string	ID-ul unic al rețetei
		title	string	Titlul rețetei
		image	string	URL imagine rețetă
lunch	map	id	string	ID-ul unic al rețetei
		title	string	Titlul rețetei
		image	string	URL imagine rețetă
dinner	map	id	string	ID-ul unic al rețetei
		title	string	Titlul rețetei
		image	string	URL imagine rețetă

Tabel 6 - Structura unui element al obiectului de planificare zilnică



Figură 10 - Diagrama arhitecturii bazei de date

4. IMPLEMENTAREA APLICAȚIEI

4.1. TEHNOLOGII UTILIZATE

4.1.1. REACT NATIVE

În realizarea aplicației mobile din această lucrare, am optat pentru utilizarea framework-ului React Native, o tehnologie dezvoltată de compania Meta, recunoscută pentru capacitatea sa de a permite dezvoltarea de aplicații mobile compatibile cu sisteme de operare mobile majore, atât cu Android, cât și cu iOS, folosind o singură bază de cod. Această abordare de tip multiplatformă („cross-platform”) permite o experiență de utilizare coerentă pe diferite platforme și reduce considerabil timpul și efortul necesar pentru dezvoltarea, dar mai ales pentru menținerea produselor software.

Prin utilizarea acestei tehnologii, aplicația beneficiază de performanțe apropiate de cele ale aplicațiilor native, adică acele programe software dezvoltate exclusiv pentru un anumit sistem de operare. De exemplu, o aplicație nativă Android ar fi scrisă în Kotlin, iar una nativă iOS - în Swift, fiecare optimizată strict pentru mediul său. De asemenea, acest framework este compatibil cu o gamă largă de biblioteci externe, iar ecosistemul său este bine documentat și activ. În contextul de față, arhitectura este bazată pe componente funcționale și funcții hooks din React, fără utilizarea programării orientate pe obiecte. Structurarea în această manieră a permis definirea clară și izolată a fiecărui ecran.

Logica fiecărei componente este gestionată separat de stilurile vizuale, asigurând modularitate și claritate în cod. Aplicația valorifică avantajele oferite de funcționalități precum vizualizarea imediată a modificărilor din cod („hot reload” și „fast refresh”) și integrarea cu API-urile native ale platformei cu suportul pentru elemente hardware, cum este camera foto. Sunt utilizate componente standard precum FlatList, ScrollView, TextInput, TouchableOpacity, Image, Alert, Modal și ActivityIndicator, toate transpusă în echivalente native la rulare. Utilizarea interfeței este realizată cu ajutorul funcției StyleSheet.create(), folosind valori receptive și o paletă de culori unitară definită în fișiere centralizate.

React Native permite reducerea timpului de dezvoltare cu până la 40% comparativ cu dezvoltarea separată nativă pentru Android și iOS și menține, în același timp, o experiență de utilizare fluidă și o performanță satisfăcătoare [17]. În evaluările

experimentale, acest framework a fost considerat adecvat pentru aplicații comerciale de dimensiune medie, cu interfețe vizuale complexe, dar care nu necesită procesare grafică intensivă. Totodată, este utilizat motorul Hermes, integrat în versiunile recente ale React Native, ce contribuie la optimizarea timpului de pornire și la reducerea consumului de memorie, aspecte importante pentru funcționarea fluidă pe dispozitive mobile variate [18].

TypeScript este o extensie a limbajului JavaScript care introduce tipizare statică și verificare la compilare, fiind dezvoltat de Microsoft. În comparație cu JavaScript, care este un limbaj dinamic, TypeScript permite definirea explicită a tipurilor pentru variabile, funcții, parametri și obiecte complexe, ceea ce ajută la identificarea erorilor înainte de execuție. În proiectul meu, această caracteristică a fost utilă pentru menținerea consistenței în manipularea structurilor de date care au numeroase proprietăți specifice și reguli de validare. Utilizarea fișierelor de tip .tsx și .ts a permis aplicarea strictă a acestor tipuri în toate componente și serviciile aplicației, în special în interacțiunea cu Firebase, unde greșelile de tip ar fi putut produce erori [19].

În concluzie, integrarea React Native în dezvoltarea aplicației prezentate oferă un echilibru solid între rapiditatea implementării, experiența nativă pentru utilizatori și posibilitățile de extindere ulterioară, ceea ce confirmă eficiența acestei tehnologii ca alegere principală pentru aplicații mobile moderne, robuste și scalabile.

4.1.2. FIREBASE

Firebase reprezintă soluția principală de backend utilizată în dezvoltarea aplicației, oferind o suită completă de servicii cloud optimize pentru aplicații mobile. Platforma, dezvoltată de Google, permite integrarea rapidă a funcționalităților critice precum autentificarea, baza de date, stocarea de fișiere și analize de utilizare, reducând semnificativ nevoia de configurare manuală a infrastructurii server-side (operațiunile și procesele au loc pe server, nu pe dispozitivul utilizatorului).

În contextul aplicației descrise, Firebase este utilizat în special pentru gestionarea autentificării utilizatorilor, stocarea datelor în timp real și implementarea regulilor de acces pentru securitatea documentelor. Firestore, sistemul de baze de date NoSQL oferit de Firebase, permite structurarea informațiilor sub formă de colecții și documente, asigurând atât flexibilitate, cât și sincronizare automată între dispozitive. Acest model este ideal pentru aplicații cu un grad ridicat de interactivitate și actualizare, cum este cazul actual, unde utilizatorii își actualizează frecvent datele despre rețete, ingrediente și preferințe.

Autentificarea în aplicație este gestionată prin Firebase Authentication, care oferă metode sigure de autentificare, respectiv prin e-mail și parolă, dar și posibilitatea integrării

cu alte servicii (de exemplu Google, Facebook). Toate datele utilizatorului sunt protejate prin reguli stricte definite în consolă, ce permit accesul exclusiv pe baza UID-ului corespunzător. Conform documentației oficiale, această platformă este proiectată să modifice scalar automat în funcție de numărul de utilizatori activi, fără intervenții suplimentare din partea dezvoltatorului [20]. Astfel, aplicația poate susține un număr semnificativ de utilizatori simultan, fără modificări arhitecturale.

În plus, studii recente din literatura de specialitate confirmă avantajele oferite de acest serviciu BaaS în dezvoltarea rapidă a aplicațiilor mobile. BaaS (Backend as a Service) este un model cloud ce oferă dezvoltatorilor funcționalități de backend pre-construite (de exemplu baze de date și autentificare), eliminând nevoia de a gestiona propria infrastructură de servere. Într-un caz real, folosirea Firestore a redus timpul de citire și timpul de scriere într-un proces de rezervare cu aproximativ 80 % față de o metodă manuală [21].

Un alt aspect important este acela al securității. Această platformă Google permite implementarea de reguli declarative la nivel de document sau colecție, ceea ce oferă un control granular asupra datelor. Regulile sunt evaluate în timp real și sunt complet integrate cu sistemul de autentificare, reducând riscul accesului neautorizat [22].

În concluzie, Firebase oferă o infrastructură care răspunde nevoilor aplicației atât din punct de vedere funcțional, cât și din perspectiva securității și eficienței în dezvoltare. Alegerea acestei tehnologii a asigurat un echilibru între simplitatea integrării și complexitatea cerințelor aplicației.

4.1.3. CLOUDINARY

Pentru stocarea imaginilor asociate rețetelor, am folosit serviciul Cloudinary, o platformă de tip cloud specializată în gestionarea conținutului media. Alegerea acestei soluții s-a bazat pe capacitatea sa de a oferi un flux optimizat de încărcare, stocare și livrare a imaginilor în aplicații mobile, cu o latență redusă și o compatibilitate mare cu infrastructuri frontend moderne. Cloudinary este recunoscută pentru funcționalitățile sale avansate [23], fiind utilizat în mod frecvent în aplicații comerciale datorită API-urilor sale de tip REST, a suportului pentru conversii automate de format și redimensionări dinamice, precum și a integrării facile cu alte servicii backend.

În cadrul aplicației, imaginile pot fi încărcate de utilizator la momentul creării unei rețete. Procesul este gestionat pe dispozitiv cu ajutorul modulului expo-image-picker, care permite selectarea unei imagini din galeria locală sau chiar capturarea uneia noi. După selectare, imaginea este convertită local în format Base64 (o metodă de codificare și decodificare ce reprezintă date binare pe baza a 64 de caractere imprimabile), pentru

a le putea transporta prin sisteme care sunt construite să proceseze doar text. Ulterior, imaginea este trimisă către endpoint-ul Cloudinary printr-o cerere POST, împreună cu un set de reguli predefinite, denumit „preset”, configurat în consola de administrare a serviciului. În cazul actual, acest „preset” utilizat este de tip „unsigned”, adică nu necesită o semnătură criptografică generată pe server pentru a autoriza încărcarea. Acest lucru înseamnă că autentificarea se face prin validarea implicită și securizată realizată de Cloudinary pe baza numelui setului predefinit de reguli și configurații, fără a expune date sensibile în codul clientului.

După procesarea cererii, Cloudinary returnează un obiect JSON (JavaScript Object Notation). JSON reprezintă un format ușor de descifrate, ideal pentru schimbul de date dintre diferitele limbaje de programare și platforme [24]. Acest obiect conține informații despre fișierul încărcat, inclusiv linkul public de acces, ce este apoi salvat în Firestore într-un câmp specific al rețetei. Astfel, la afișarea ulterioară a rețetei, aplicația poate încărca imaginea direct de pe rețea de livrare a conținutului a acestei platforme (CDN - Content Delivery Network), fără a stoca sau transporta conținut media prin Firebase, reducând astfel costurile și timpul de răspuns.

Avantajul principal al utilizării acestei tehnologii constă în faptul că permite o gestionare eficientă a fișierelor media fără a fi necesară dezvoltarea unui sistem propriu de stocare și livrare a imaginilor. Platforma include, de asemenea, funcționalități suplimentare precum optimizarea automată a dimensiunilor imaginilor, conversia în formate moderne, ca WebP (format de compresie eficientă pentru web), generarea de imagini de previzualizare personalizate. Aceste opțiuni pot fi explorate în versiunile viitoare ale aplicației.

În concluzie, integrarea serviciului Cloudinary a oferit o soluție stabilă, sigură și scalabilă pentru gestionarea conținutului media generat de utilizatori. Aceasta a permis separarea logicii de stocare media față de cea a datelor structurale și a redus considerabil complexitatea implementării unei infrastructuri proprii pentru upload și livrare de imagini.

4.2. STRUCTURA PROIECTULUI

4.2.1. ORGANIZAREA FIȘIERELOR

Structura internă a aplicației mobile respectă principiile modularității și separării responsabilităților, fiind construită conform bunelor practici din ecosistemul React Native împreună cu Expo. Arhitectura proiectului permite dezvoltarea incrementală, reutilizarea codului și o mențenanță eficientă.

Directorul rădăcină al proiectului conține următoarele module funcționale:

- /app/ - include toate ecranele aplicației, iar fiecare fișier .tsx corespunde unei rute (sau subrute) definite automat de expo-router, de exemplu app/index.tsx și app/profile.tsx;
- /components/ - cuprinde componentele vizuale reutilizabile (carduri de rețete, butoane, modale, indicatori), fiecare organizată împreună cu propriul fișier de stil, sprijinind reutilizarea fără redundanță și reducând duplicarea codului între module;
- /constants/ - conține fișiere cu valori statice (culori, fonturi, spațieri), utilizate global în aplicație pentru a asigura consistență vizuală;
- /services/ - cuprinde logica de interacțiune cu Firebase, de exemplu recipeService.ts;
- /utils/ - oferă funcții ajutătoare, cum ar fi validările de formular și conversiile de unități, utilizate în mai multe componente;
- /hooks/ - include funcții auxiliare personalizate pentru gestionarea stării și logicii complexe de business, ca useMealPlan sau useUserPreferences.
- /types/ - definește interfețele și tipurile TypeScript partajate în aplicație;
- /assets/ - stochează resurse statice precum imagini, fonturi personalizate și pictograme;

Fișierele de stilizare sunt scrise local, în interiorul fiecărei componente (StyleSheet.create()), sau partajate din constants/colors.ts și constants/fonts.ts, în funcție de caz. Această separare între logică și prezentare a îmbunătățit lizibilitatea codului și a facilitat ajustarea designului în timp. În rădăcina proiectului se regăsesc și fișiere de configurare esențiale: app.json, tsconfig.json, firebase.rules, metro.config.js, babel.config.js, necesare pentru rulare, compilare, transpilare sau integrarea cu Expo și Firebase.

Această organizare clară a sprijinit dezvoltarea modulară, testarea separată a fiecărui ecran, extensibilitatea aplicației și evitarea dependențelor rigide. Structura de directoare a fost aleasă astfel încât să fie scalabilă și să susțină viitoarele funcționalități fără a compromite arhitectura existentă.

4.2.2. SISTEMUL DE RUTE PRINCIPALE

Proiectul este structurat în jurul unui sistem de rute automat generate, bazat pe organizarea fișierelor din directorul principal /app, conform convențiilor impuse de biblioteca Expo Router. Acest sistem permite o asociere directă între fișierele .tsx și căile URL accesibile în aplicație, eliminând necesitatea unei configurații explicite a rutelor. Ele

sunt grupate logic în baza funcționalității pe care o deservesc, după cum sunt organizate în următoarele două tabele.

	Fișierul	Rol	Ruta
app/ (tabs)/	_layout.tsx	Layout general pentru grupul pagini principale	-
	index.tsx	Acces la ecranul principal	/
	pantry.tsx	Acces la secțiunea Cămară	/pantry
	profile.tsx	Acces la Profilul utilizatorului	/profile
	recipes.tsx	Acces la Rețete	/recipes

Tabel 7 - Structura sistemului de rute al directorului principal

Fișierul		Rol	Ruta
app/ (auth)/	_layout	Layout general pentru grupul de autentificare	-
	login.tsx	Pagina de autentificare	/auth/login
	register.tsx	Pagina de înregistrare	/auth/register
app/ meal- planner	_layout.tsx	Layout general pentru secțiunea de planificare a meselor	-
	CalendarModal.tsx	Selectare dată pentru planificare	/meal-planner/CalendarModal
	MealPlanModal.tsx	Modal cu lista de rețete pentru adăugare sau editare masă	/meal-planner/MealPlanModal
	MealPlanner.tsx	Interfață principală de planificare mese	/meal-planner/MealPlanner
app/ pantry- folder	_layout.tsx	Layout general pentru secțiunea Cămară	-
	BarcodeResults.tsx	Rezultate obținute în urma scanării codului de bare	/pantry-folder/BarcodeResults
	BarcodeScanner.tsx	Scanare cod de bare pentru identificarea unui ingrediant	/pantry-folder/BarcodeScanner
	ingredientsList.tsx	Listă completă de ingrediente	/pantry-folder/ingredientsList
	myIngredients.tsx	Lista personală de ingrediente disponibile	/pantry-folder/myIngredients
app/ recipes- folder	_layout.tsx	Layout general pentru secțiunea Retete	-
	completedRecipes.tsx	Retetele finalizate de utilizator	/recipes-folder/completedRecipes
	cookingInProgress.tsx	Mod de gătire pas cu pas pentru o rețetă	/recipes- folder/cookingInProgress
	ingredientBasedRecipes.tsx	Retete filtrate pe baza ingredientelor disponibile	/recipes- folder/ingredientBasedRecipes
	myRecipes.tsx	Retetele proprii ale utilizatorului	/recipes-folder/myRecipes
	RecipeForm.tsx	Formular pentru adăugarea unei rețete	/recipes-folder/RecipeForm
	recipeList.tsx	Listă completă a rețetelor disponibile	/recipes-folder/recipeList
	recipeMatch.tsx	Potrivire a rețetelor cu ingredientele din cămară	/recipes-folder/recipeMatch
	savedRecipes.tsx	Retetele salvate ca favorite	/recipes-folder/savedRecipes
app/ settings- folder	_layout.tsx	Layout general pentru secțiunea Setări	-
	bannedIngredients.tsx	Gestionarea ingredientelor interzise	/settings- folder/bannedIngredients
	changeName.tsx	Schimbarea numelui afișat de utilizator	/settings- folder/changeName
	changePassword.tsx	Modificarea parolei contului	/settings- folder/changePassword
	preferences.tsx	Configurarea preferințelor alimentare	/settings- folder/preferences
	settings.tsx	Pagina principală de setări	/settings- folder/settings

Tabel 8 - Structura sistemului de rute al directoarelor

4.2.3. NAVIGAREA CU EXPO-ROUTER

Pentru implementarea sistemului de navigare între ecrane în cadrul aplicației, am utilizat biblioteca Expo Router, o soluție modernă din ecosistemul React Native, construită pe baza React Navigation. Aceasta implementează un mecanism de rutare declarativ, bazat pe convenții de denumire și pe structura fizică a fișierelor din directorul /app, inspirat direct din arhitectura Next.js.

În acest sistem, fiecare fișier .tsx din /app definește automat o rută accesibilă în aplicație, fără a necesita definirea manuală într-un fișier de configurare centralizat. Spre exemplu, app/recipes.tsx asociază ruta /recipes, iar app/recipes/[id].tsx definește o rută dinamică accesibilă prin identificatorul unic al unei rețete. Aceste fișiere pot primi parametri prin hook-ul useLocalSearchParams(). Termenul „hook” reprezintă o funcție specială care permite accesarea unor caracteristici specifice ale React sau bibliotecilor React (cum sunt parametrii de rută).

Organizarea logică a navigării este îmbunătățită prin utilizarea dosarelor cu denumiri între paranteze rotunde, precum „(tabs)”, care grupează rute fără a complica adresele URL (rutele) finale ale aplicației. Prin crearea acestei grupări am putut plasa toate ecranele care apar într-o bară de navigare și aceasta a permis definirea de comportamente comune pentru mai multe rute. Un element important în structura de navigare este reprezentat de fișierele denumite _layout.tsx, care configurează aspectul vizual și comportamental pentru un întreg grup de rute. Fișierul app/(tabs)/_layout.tsx definește bara de navigare inferioară și organizează cele patru secțiuni principale ale aplicației: Acasă (Home), Cămară (Pantry), Rețete (Recipes) și Profil (Profile). Acest aspect permite reutilizarea structurii comune a paginilor și reduce duplicarea codului de navigare.

Navigarea programatică este implementată cu ajutorul funcției router.push(), iar navigarea declarativă se face prin componenta <Link>. Ambele sunt oferite nativ de expo-router și permit trecerea între ecrane, transmiterea de parametri și controlul fluxului de utilizare (de exemplu deschiderea unei rețete, revenirea la listă, modificarea profilului). În cadrul acestui proiect, am utilizat și navigare ierarhică („nested routing”), obținută prin structurarea subdirectoarelor din /app. De exemplu, fișierul app/pantry-folder/shoppingList.tsx definește ruta /pantry-folder/shoppingList. Abordarea aceasta a permis o corelare clară între structura de business și cea de navigare, cu impact pozitiv asupra mentenanței și extensibilității.

La nivel tehnic, Expo Router se bazează intern pe React Navigation, dar expune un API superior prin automatizarea rutelor, integrare completă cu tipuri TypeScript, animații native, stivă de navigare și suport pentru „deep linking”. Acest termen se referă la permisiunea ca un link extern să deschidă direct aplicația mobilă exact la conținut sau

o secțiune internă specifică, în loc de pagina principală a acestora). Astfel, logica de rutare a rămas predictibilă, scalabilă și ușor de testat.

Prin utilizarea acestui sistem de navigare, am obținut un grad înalt de coerență arhitecturală, o interfață clar structurată și o experiență unitară pentru utilizator, fără complexitatea configurației manuale a rutelor. Alegerea Expo Router s-a dovedit adecvată pentru acest proiect, cu multiple secțiuni interconectate și interacțiuni dinamice.

4.3. FUNCȚIONALITĂȚI IMPLEMENTATE

4.3.1. PROCESUL DE ÎNREGISTRARE ȘI AUTENTIFICARE

Procesul de înregistrare și autentificare este componenta fundamentală ce asigură accesul controlat și personalizat. Prin integrarea serviciului Firebase Authentication, aplicația beneficiază de un sistem sigur și extensibil pentru administrarea conturilor de utilizatori, fără a fi necesară gestionarea manuală a criptării sau stocării parolelor.

Procesul de înregistrare permite crearea unui cont nou pe baza unei adrese de e-mail și a unei parole. Înainte de trimitera datelor către Firebase, aplicația efectuează o validare locală a parolei. Astfel asigură respectarea unor cerințe minime de securitate: lungime minimă de 8 caractere și maximă de 20, dintre care cel puțin o literă mare, cel puțin o literă mică, cel puțin o cifră și să nu conțină spații. Această validare oferă feedback clar și imediat utilizatorului, evitând erorile din partea serverului. Acest proces este însoțit de mesaje vizuale (de tip pop-up) care confirmă succesul operațiunii sau semnalează eșecul și erorile întâlnite, contribuind la o experiență intuitivă pentru utilizator. În cazul în care adresa de e-mail este deja utilizată sau formatul este incorrect, sunt afișate alerte personalizate.

După validarea locală, aplicația trimite cererea de creare a contului prin metoda `createUserWithEmailAndPassword`, specifică Firebase, cu adresa de e-mail și parola. Obiectul `userCredential` returnat conține date despre utilizatorul nou creat. După acest pas, Firebase gestionează intern datele de înregistrare, criptând parola și asociind utilizatorul cu un `UID` (User ID) unic, utilizat ulterior în bazele de date pentru identificare. Această arhitectură elimină necesitatea de a stoca parole pe client sau în `Firestore`. După înregistrare, se realizează și actualizarea automată a profilului prin setarea numele complet introdus la înregistrare.

Autentificarea este disponibilă pentru utilizatorii existenți și presupune introducerea adresei de e-mail și a parolei asociate. Se verifică aceste date folosind metoda `signInWithEmailAndPassword`, iar la autentificare reușită, sunt salvate local,

folosind AsyncStorage, informații relevante despre utilizator: adresa de e-mail, UID-ul unic generat de Firebase și momentul autentificării.

Un aspect important al acestui proces este opțiunea de „keep me logged in”, implementată prin stocarea preferinței în memoria locală a aplicației. Astfel, la redeschiderea aplicației, utilizatorul poate fi autentificat automat, fără a reintroduce datele de acces, ceea ce îmbunătățește semnificativ experiența de utilizare. Pentru salvarea preferinței de a rămâne conectat se folosește AsyncStorage. În lipsa alegerii acestei opțiuni de către utilizator, sesiune se menține activă 24 de ore, după care acesta este deconectat și va fi nevoie să reintroducă datele proprii de autentificare.

În cazul erorilor, precum utilizator inexistent, parolă greșită sau prea multe încercări eşuate, aplicația oferă feedback explicit și direcțional, reducând frustrarea utilizatorului și ajutându-l să corecteze problema.

După autentificare reușită, aplicația folosește router.replace('/(tabs)') pentru a redirecționa utilizatorul către zona principală a aplicației, ecranul central.

4.3.2. PROFILUL UTILIZATORULUI

Fiecare utilizator are un profil asociat imediat după înregistrare, iar acest profil este folosit pentru a gestiona date precum numele afișat, adresa de e-mail, preferințele alimentare, rețetele salvate, istoricul rețetelor finalizate și informațiile despre cămară.

Datele de bază ale profilului (nume, email, UID) sunt gestionate prin Firebase Authentication, iar informațiile suplimentare (cum sunt preferințele, restricțiile, favoritele) sunt salvate în Cloud Firestore, sub forma unui document individual pentru fiecare utilizator. Fiecare document este stocat într-o colecție users, identificat prin UID-ul generat automat la înregistrare.

Aplicația actualizează informațiile în Firestore folosind updateDoc, iar orice modificare este sincronizată în timp real.

4.3.3. PROCESUL DE GESTIONARE A PREFERINȚELOR

Gestionarea preferințelor alimentare în aplicație permite personalizarea conținutului afișat fiecărui utilizator, în special în etapa de filtrare și recomandare a rețetelor. Sistemul permite configurarea unor regimuri alimentare generale (vegetarian, vegan, fără gluten, fără lactate, fără nuci) prin comutatoare disponibile într-o pagină dedicată.

Valorile comutatoarelor sunt stocate local într-un obiect de tip DietaryPreferences, cu câmpuri booleene inițializate implicit cu false. La montarea componentei, aplicația apelează o funcție asincronă initializeData(), care la rândul ei invocă metoda

loadPreferences() pentru a obține datele salvate din Firebase Firestore. Datele sunt preluate din documentul users/{uid}/preferences/dietary, iar sincronizarea în timp real este realizată prin onSnapshot().

Dacă documentul există, conținutul său este preluat și starea este actualizată folosind setPreferences(). În caz de eroare la accesarea documentului sau la stabilirea ascultătorului, sunt afișate mesaje de eroare corespunzătoare în consolă și în interfața aplicației. Funcția returnează un unsubscribe pentru a putea fi apelat la demontarea componentei, asigurând curățarea corespunzătoare a resurselor. AGetDoc() și actualizează starea aplicației prin setPreferences() doar dacă documentul există.

```
const [preferences, setPreferences] = useStateDietaryPreferences<{  
    vegetarian: false,  
    vegan: false,  
    glutenFree: false,  
    dairyFree: false,  
    nutFree: false,  
}>();  
  
useEffect(() => {  
    let unsubscribe: (() => void) | undefined;  
  
    const initializeData = async () => {  
        try {  
            const cleanup = await loadPreferences();  
            unsubscribe = cleanup;  
        } catch (error) {  
            console.error('Error initializing dietary preferences:', error);  
        }  
    };  
  
    initializeData();  
    return () => {  
        if (unsubscribe) {  
            unsubscribe();  
        }  
    };  
, []);  
  
const loadPreferences = async () => {  
    const user = auth.currentUser;  
    if (!user) return;  
  
    try {  
        const userPreferencesRef = doc(db, 'users', user.uid, 'preferences', 'dietary');  
  
        const unsubscribe = onSnapshot(userPreferencesRef, (preferencesDoc) => {  
            try {  
                if (preferencesDoc.exists()) {  
                    setPreferences(preferencesDoc.data() as DietaryPreferences);  
                }  
            } catch (error) {  
                console.error('Error processing dietary preferences:', error);  
                Alert.alert('Error', 'Failed to load preferences. Please try again.');//  
            }  
        }, (error) => {  
            console.error('Error listening to dietary preferences:', error);  
            Alert.alert('Error', 'Failed to load preferences. Please try again.');//  
        });  
  
        return unsubscribe;  
    } catch (error) {  
        console.error('Error setting up dietary preferences listener:', error);  
        Alert.alert('Error', 'Failed to load preferences. Please try again.');//  
    }  
};
```

Secvență de cod 1 - Inițializarea preferințelor și încărcarea din Firestore

Modificările efectuate de utilizator sunt salvate la nivel de cloud prin funcția savePreferences(), prezentată în secvența de cod 2. Această funcție verifică dacă utilizatorul este autentificat, iar în caz afirmativ, actualizează documentul din Firestore cu noile valori ale preferințelor utilizând setDoc().

```
const savePreferences = async (newPreferences: DietaryPreferences) => {
  const user = auth.currentUser;
  if (!user) {
    Alert.alert('Error', 'You must be logged in to save preferences');
    return;
  }

  try {
    const userPreferencesRef = doc(db, 'users', user.uid, 'preferences', 'dietary');
    await setDoc(userPreferencesRef, newPreferences);
  } catch (error) {
    console.error('Error saving preferences:', error);
    Alert.alert('Error', 'Failed to save preferences. Please try again.');
  }
};
```

Secvență de cod 2 - Salvarea preferințelor în Firestore

Pentru asigurarea confidențialității și integrității datelor, accesul la documentele de preferințe este restricționat prin reguli de securitate definite în Firestore. Aceste reguli permit accesul la date doar utilizatorului autentificat care deține documentul, utilizând funcții auxiliare precum isAuthenticated() și isOwner(userId). Această abordare garantează că fiecare utilizator are acces exclusiv la propriile date, oferind protecție împotriva accesului neautorizat. Utilizarea funcțiilor reutilizabile în definiția regulilor contribuie, totodată, la claritatea și întreținerea facilă a politicilor de securitate.

```
// Preferences subcollection
match /preferences/{document=**} {
  allow read, write: if isAuthenticated() && isOwner(userId);
}
```

Secvență de cod 3 - Regulile din consola Firebase

Preferințele alimentare sunt utilizate în mod direct în logica de filtrare și selecție a rețetelor. Aplicația elimină automat din sugestiile culinare, dar nu și din lista de rețete obișnuită. În acest caz, aplicația semnalează explicit conflictele în cazul în care acestea există. Sincronizarea preferințelor este realizată automat la autentificare, asigurând astfel coerenta setărilor pe întreg parcursul utilizării aplicației.

În completarea preferințelor alimentare generale, utilizatorul are posibilitatea să definească o listă personalizată de ingrediente interzise. Această funcționalitate este destinată evitării stricte a unor alimente care pot cauza alergii, intoleranțe sau care pur și simplu nu sunt dorite în rețetele culinare.

Interfața dedicată permite selecția ingredientelor interzise dintr-o listă completă preluată din API-ul extern TheMealDB. Ingredientele sunt sortate alfabetic, iar starea lor, interzis sau permis, este reflectată vizual în timp real. Utilizatorul poate adăuga sau elimina un ingredient din listă printr-un simplu gest de apăsare, caz în care aplicația actualizează atât starea locală, cât și documentul utilizatorului din Firestore. Ingredientele interzise sunt stocate în câmpul `bannedIngredients` din documentul `users/{uid}`, sub forma unui vector de șiruri de caractere scrise cu minuscule (în format lowercase).



Figură 11 – Exemplu structura câmpului `bannedIngredients`

Actualizarea listei se face asincron, utilizând metodele `arrayUnion` și `arrayRemove` furnizate de Firestore. Aceste metode permit adăugarea sau eliminarea unui ingredient fără a suprascrie întregul vector, păstrând astfel integritatea datelor și consistența sincronizării.

```
const toggleBannedIngredient = async (ingredientId: string) => {
  const user = auth.currentUser;
  if (!user) return;

  try {
    const userRef = doc(db, 'users', user.uid);
    const ingredient = ingredients.find(ing => ing.id === ingredientId);
    if (!ingredient) return;

    const isCurrentlyBanned = bannedIngredients.includes(ingredient.name.toLowerCase());

    if (isCurrentlyBanned) {
      await setDoc(userRef, {
        bannedIngredients: arrayRemove(ingredient.name.toLowerCase())
      }, { merge: true });
    } else {
      await setDoc(userRef, {
        bannedIngredients: arrayUnion(ingredient.name.toLowerCase())
      }, { merge: true });
    }
  } catch (error) {
    console.error('Error updating banned ingredients:', error);
  }
};
```

Secvență de cod 4 – Gestionarea de ingrediente interzise în Firestore

Atunci când utilizatorul selectează un ingredient din listă, funcția `toggleBannedIngredient` determină dacă acel ingredient este deja prezent în lista de ingrediente interzise (`bannedIngredients`). Verificarea se face pe baza numelui ingredientului convertit la lowercase, pentru consistență. Dacă ingredientul este deja interzis, funcția îl elimină din documentul utilizatorului din Firestore folosind `arrayRemove`. Dacă nu este prezent, îl adaugă cu `arrayUnion`. În ambele cazuri, modificarea este realizată cu `setDoc`, cu opțiunea `{ merge: true }` pentru a păstra restul structurii documentului intactă.

Funcția verifică mai întâi existența utilizatorului și a ingredientului selectat, iar în cazul apariției unei erori, aceasta este capturată și logată. Modificarea datelor este realizată direct în Firestore, iar actualizarea interfeței se bazează ulterior pe sincronizarea în timp real. La nivel funcțional, lista ingredientelor interzise este utilizată în mod direct în procesul de filtrare a rețetelor. Orice rețetă care conține cel puțin un ingredient aflat pe lista `bannedIngredients` nu este automat exclusă din rezultate, doar din sugestiile culinare. În cazul în care utilizatorul selectează o rețetă ce conține ingrediente interzise, aplicația semnalează această situație printr-un mesaj de alertă și evidențiere vizuală a ingredientelor care intră în conflict cu alegerile acestuia secvența de cod 5.

```
const recipeHasBannedIngredients = (recipe: any) => {
  if (!recipe.ingredients || !Array.isArray(recipe.ingredients)) return false;
  return recipe.ingredients.some((ingredient: any) => {
    const ingredientName = typeof ingredient === 'string'
      ? ingredient
      : ingredient?.name;
    if (!ingredientName) return false;
    return bannedIngredients.includes(ingredientName.toLowerCase());
  });
};
```

Secvență de cod 5 – Deteceția prezenței ingredientelor interzise într-o rețetă

Pentru a notifica utilizatorul într-un mod vizibil, aplicația utilizează și o funcție dedicată de afișare a unui mesaj de alertă, prezentată în secvența de cod 6. Aceasta listează explicit ingredientele interzise găsite în rețetă.

```
const showBannedIngredients = (recipe: any) => {
  if (!recipe.ingredients || !Array.isArray(recipe.ingredients)) return;

  const banned = recipe.ingredients
    .map((ingredient: any) => {
      const ingredientName = typeof ingredient === 'string'
        ? ingredient
        : ingredient?.name;
      if (!ingredientName) return null;
      return bannedIngredients.includes(ingredientName.toLowerCase())
        ? ingredientName
        : null;
    })
    .filter(Boolean);

  if (banned.length > 0) {
    Alert.alert(
      'Banned Ingredients Alert',
      `This recipe contains ingredients you have banned:\n\n${banned.join('\n')}`,
      [{ text: 'OK' }]
    );
  }
};
```

Secvență de cod 6 – Alertă vizuală cu ingredientele interzise detectate

4.3.4. PROCESUL DE GESTIONARE A CĂMĂRII

Funcționalitatea de gestionare a cămării virtuale permite utilizatorilor să mențină o evidență actualizată a ingredientelor disponibile din cămară proprie. Acest modul joacă un rol central în personalizarea experienței, deoarece algoritmul de sugestie culinară

utilizează exclusiv informațiile din cămară pentru a calcula gradul de compatibilitate între rețete și utilizator și a oferi astfel opțiunile.

Datele aferente cămarii utilizatorului sunt stocate în Firestore la calea `users/{uid}/ingredients/pantry`, sub forma unui obiect în care fiecare cheie reprezintă un nume de ingredient (normalizat), iar valoarea asociată conține două câmpuri: `quantity` (cantitate) și `measurementType` (tipul unității de măsură). Structura este menținută și actualizată prin operații de tip `setDoc()` cu opțiunea `{ merge: true }`, astfel încât modificările să fie aplicate punctual, fără a suprascrie întreaga colecție. Funcția `handleEditQuantity` permite modificarea valorii cantitative a unui ingredient existent. La declanșare, aceasta validează dacă există un ingredient selectat și o valoare introdusă. Se determină unitatea de măsură asociată ingredientului curent fie prin extragerea literei din cantitatea deja existentă (ex: „g”, „ml”), fie printr-o funcție auxiliară care returnează unitatea pe baza tipului de măsură. După formarea noii valori (`newQuantity`), documentul Firestore este actualizat prin adăugarea sau suprascrierea cheii corespunzătoare ingredientului. De asemenea, starea locală a aplicației este sincronizată imediat, pentru a reflecta modificarea vizual. În cazul apariției unei erori, utilizatorul este notificat, iar eroarea este menționată în consolă pentru depanare.

```
const handleEditQuantity = async () => {
  if (!selectedIngredient || !quantityInput) return;
  const user = auth.currentUser;
  if (!user) {
    Alert.alert('Error', 'You must be logged in to update ingredients');
    return;
  }
  try {
    const userIngredientsRef = doc(db, 'users', user.uid, 'ingredients', 'pantry');
    const ingredientsDoc = await getDoc(userIngredientsRef);
    const currentData = ingredientsDoc.exists() ? ingredientsDoc.data() : {};
    let unit = '';
    const match = selectedIngredient.quantity.match(/[^a-zA-Z]+$/);
    if (match) {
      unit = match[0];
    } else {
      unit = getUnitDisplay(selectedIngredient.measurementType);
    }
    const newQuantity = quantityInput + unit;
    await setDoc(userIngredientsRef, {
      ...currentData,
      [selectedIngredient.name.trim().toLowerCase()]: {
        quantity: newQuantity,
        measurementType: selectedIngredient.measurementType
      }
    }, { merge: true });
    // se actualizează state loc
    const updatedIngredients = ingredients.map(ing =>
      ing.name === selectedIngredient.name
        ? { ...ing, quantity: newQuantity }
        : ing
    );
    setIngredients(updatedIngredients);
    setQuantityVisible(false);
    setSelectedIngredient(null);
    setQuantityInput('');
    Alert.alert('Success', `Quantity updated for ${selectedIngredient.name}`);
  } catch (error) {
    console.error('Error updating ingredient quantity:', error);
    Alert.alert('Error', 'Failed to update ingredient quantity');
  }
};
```

Secvență de cod 7 – Actualizarea cantității unui ingredient

Pentru fiecare ingredient, aplicația oferă o interfață de editare rapidă a cantității și o opțiune de ștergere. Afisarea cantităților, a unităților de măsură și a butoanelor de acțiune este realizată în cadrul funcției `renderItem`, responsabilă pentru generarea fiecărui rând din listă.

```
const renderItem = ({ item }: { item: Ingredient }) => (
  <View style={styles.ingredientItem}>
    <View style={styles.ingredientInfo}>
      <Text style={styles.ingredientName}>{item.name.toLowerCase()}</Text>
    </View>
    <View style={styles.rightActions}>
      <TouchableOpacity
        style={styles.quantitySquareBadge}
        onPress={() => {
          setSelectedIngredient(item);
          setQuantityInput(item.quantity.match(/\d+(?:\.\d+)?/)?.[0] || '');
          setQuantityModalVisible(true);
        }}
      >
        <Text style={styles.quantitySquareBadgeText}>{getQuantityDisplay(item)}</Text>
      </TouchableOpacity>
      <TouchableOpacity
        style={{ marginLeft: 8 }}
        onPress={() => {
          setSelectedIngredient(item);
          setRemoveModalVisible(true);
        }}
      >
        <Ionicons name="close" size={22} color={Colors.error} />
      </TouchableOpacity>
    </View>
  </View>
);
```

Secvență de cod 8 – Afisarea vizuală a ingredientelor în listă

Aplicația permite de asemenea ștergerea individuală sau colectivă a ingredientelor. Pentru eliminarea completă a inventarului, printr-un buton sugestiv, utilizatorul poate apela funcția handleDeleteAll, care resetează întreg documentul respectiv din Firestore. În cazul ingredientelor speciale, precum cele marcate cu „to taste”, aplicația oferă un flux separat, cu confirmare explicită prin dialog vizual.

Gestionarea cămării este integrată cu motorul de sugestii al aplicației și cu secțiunea de planificare a rețetelor. Ingredientele existente sunt utilizate pentru a evalua în ce măsură o rețetă este potrivită pentru a începe prepararea acesteia, în raport cu resursele disponibile. Aplicația evidențiază sugestiv, prin procentaj și culori specifice, gradul de potrivire dintre rețete și cămara digitală a utilizatorului.

4.3.5. PROCESUL DE GĂSIRE A UNEI REȚETE

Modulul de găsire a rețetelor compatibile utilizează ingredientele din cămara virtuală a utilizatorului pentru a identifica, filtra și ordona rețetele posibile, în funcție de gradul de potrivire. Funcționalitatea este implementată în cadrul componentei IngredientBasedRecipesScreen și combină date externe, obținute de la API-ul TheMealDB, cu rețetele salvate în baza de date Firestore.

La inițializare, aplicația stabilește o conexiune în timp real cu documentul users/{uid}/ingredients/pantry, de unde extrage ingredientele disponibile. Acestea sunt procesate și utilizate ca filtru pentru rețetele descărcate. Pentru fiecare rețetă, se determină ingredientele comune cu cămara, ingredientele lipsă și procentul de compatibilitate. Lista finală este sortată descrescător în funcție de acest procent.

Rețetele afișate sunt rețete externe sau proprii, iar fiecare card include titlul, imaginea, procentul de potrivire și numărul de ingrediente lipsă. Afișarea acestora este realizată cu ajutorul funcției renderRecipeItem, prezentată în secvența de cod 9.

```
const renderRecipeItem = ({ item }: { item: Recipe }) => {
  const isSaved = savedRecipes[item.id];

  // Obține sursa corectă pentru imagine bazată pe tipul rețetei
  const getImageSource = () => {
    if (typeof item.image === 'number') {
      return item.image; // imagine din require()
    }
    return { uri: item.image }; // URL string
  };

  return (
    <TouchableOpacity
      style={styles.recipeCard}
      onPress={() => {
        router.push(item.isUserRecipe
          ? `/recipes-folder/recipeMatch?id=${item.id}`
          : `/recipes-folder/recipeMatch?id=${item.id}`);
      }}
    >
    {/* Badge pentru regiunea rețetei */}
    [...]
    <Text style=[...]></Text>
    </View>
  );
}

/* Imaginea rețetei */
<Image
  source={getImageSource()}
  style={styles.recipeImage}
  defaultSource={require('../../../assets/images/default-recipe.png')}
/>
```

```
  /* Informatiile rețetei */
  <View style={styles.recipeInfo}>
    <Text style={styles.recipeTitle}>{item.title}</Text>

    /* Container pentru procentul de potrivire */
    <View style={styles.matchContainer}>
      <Ionicons
        name="checkmark-circle"
        size={28}
        color={item.matchPercentage >= 80 ? Colors.owned : Colors.medium}
      />
      <Text style={[
        styles.matchPercentage,
        { color: item.matchPercentage >= 80 ? Colors.owned : Colors.medium }
      ]}>
        {item.matchPercentage}% Match ({item.matchingIngredientsCount}/{item.totalIngredientsCount} ingredients)
      </Text>
    </View>

    {item.missingIngredients.length > 0 && (
      <Text style={styles.missingText}>
        Missing {item.missingIngredients.length} ingredients
      </Text>
    )}
  </View>

  <TouchableOpacity
    style={commonStyles.heartButton}
    onPress={() => toggleSaveRecipe(item)}
  >
    <Ionicons
      name={isSaved ? 'heart' : 'heart-outline'}
      size={28}
      color={isSaved ? Colors.error : Colors.border}
    />
  </TouchableOpacity>
</TouchableOpacity>

```

Secvență de cod 9 – Afișarea rețetelor din lista de potriviri

Aplicația permite salvarea sau eliminarea rețetelor favorite, iar lista de sugestii este actualizată în timp real în funcție de modificările din cămară. Funcționalitatea este integrată cu fluxul general al aplicației, permitând trecerea directă către detaliu sau către gătitul propriu-zis.

4.3.6. PROCESUL DE GĂTIT PROPRIU-ZIS

Ecranul dedicat preparării rețetei este conceput pentru a oferi o experiență interactivă și personalizată în timpul procesului efectiv de gătit. Acesta este activat atunci când utilizatorul selectează o rețetă pentru care dispune de toate resursele necesare și o alege pentru a începe procesul de gătit, prin componenta CookingInProgressScreen. Atât timp cât numărul de ingrediente și cantitatea necesară nu sunt corespunzătoare cerințelor din rețetă, utilizatorul nu are posibilitatea de a continua procesul de gătit.

La deschidere, aplicația identifică sursa rețetei, extrage toate datele necesare (inclusiv lista ingredientelor, instrucțiunile și imaginea reprezentativă), apoi normalizează și adaptează valorile cantitative ale ingredientelor în funcție de factorul de scalare selectat. Dacă rețeta provine din API-ul extern TheMealDB, valorile textuale ale măsurătorilor sunt procesate și convertite prin funcția normalizeQuantity.

Pentru rețetele culinare interne, provenite de la alți utilizatori, cantitățile sunt deja structurate pe baza unităților standardizate. Aplicația adaptează aceste valori în timp real în funcție de scalingFactor, asigurând proporționalitatea ingredientelor.

```
const scaleQuantity = (quantity: string, factor: number): string => {
  if (!quantity || quantity === 'to taste' || quantity.toLowerCase().includes('dash')) {
    return quantity;
  }

  const match = quantity.match(/(\d+(?:\.\d+)?)\s*([a-zA-Z]+)?/);
  if (!match) return quantity;

  const numericValue = parseFloat(match[1]);
  const unit = match[2] || '';
  const scaledValue = numericValue * factor;

  const roundedValue = Math.round(scaledValue * 100) / 100;

  return unit ? `${roundedValue}${unit}` : roundedValue.toString();
};
```

Secvență de cod 10 – Scalarea cantităților ingredientelor în funcție de porții

În plus, pentru fiecare ingredient, se verifică existența unor conflicte cu preferințele dietetice configurate de utilizator. Dacă se identifică o incompatibilitate (ex. gluten într-o rețetă pentru un utilizator cu glutenFree: true), aplicația semnalează vizual ingredientul problematic și permite afișarea detaliilor printr-o alertă interactivă.

Interfața principală este structurată în secțiuni relevante ca: lista ingredientelor (cu cantități adaptate și evidențieri), instrucțiunile de preparare și un buton de finalizare. După parcurgerea completă a rețetei, utilizatorul poate marca rețeta ca fiind completă, caz în

care este afișat un formular pentru adăugarea unei note (scor) și a unor mențiuni, ambele opționale.

După salvare, utilizatorul are opțiunea de a naviga către secțiunea rețetelor completate sau de a reveni la ecranul principal. Prin această funcționalitate, aplicația oferă nu doar un suport tehnic pentru prepararea rețetelor, ci și o componentă motivatională și de urmărire a progresului culinar.

4.3.7. FINALIZAREA UNEI REȚETE

După încheierea procesului de gătit, utilizatorul are opțiunea de a marca rețeta ca finalizată. Acest pas nu este doar simbolic, ci activează un sistem de stocare a progresului culinar, permitând astfel urmărirea istoricului rețetelor preparate, evaluarea lor subiectivă și planificarea viitoare.

Interacțiunea este realizată printr-un sección specifică (RecipeCompletionModal), declanșat de funcția handleCompleteRecipe. În momentul salvării, aplicația construiește un nou obiect de tip CompletedRecipe, care include toate informațiile relevante pentru rețeta selectată (titlu, imagine, data completării, numărul de porții, ingrediente cu cantități adaptate numărului de porții care a fost selectat de utilizator înaintea începerii procesului de gătit, scor, mențiune) și îl salvează într-un document dedicat din subcolecția users/{uid}/completedRecipes/recipes. Ulterior, aceste rețete sunt accesibile în secțiunea pentru rețete finalizate, unde sunt afișate într-o listă cronologică, sortabilă în funcție de dată, titlu sau scor. La selectarea unei rețete finalizate, utilizatorul poate vizualiza toate detaliile într-un ecran modal interactiv.

Aplicația oferă și posibilitatea de ștergere individuală sau colectivă a istoricului de rețete finalizate. Butonul pentru ștergerea integrală a listei de rețete finalizate resetează întregul document corespunzător. Această funcționalitate încurajează urmărirea activă a parcursului culinar al utilizatorului, sprijinind dezvoltarea de obiceiuri sănătoase și oferind feedback despre rețetele preferate sau chiar despre cele nereușite.

4.3.8. PROCESUL DE ADĂUGARE A UNEI REȚETE

Aplicația oferă utilizatorilor autenificați posibilitatea de a contribui cu propriile rețete culinare. Procesul este accesibil din interfața principală și este implementat în componența RecipeForm.

Interfața de adăugare include toate câmpurile esențiale pentru definirea unei rețete complete: titlul, zona geografică, categoria, lista ingredientelor, instrucțiunile pas cu pas și o imagine reprezentativă. Pentru comoditate, aplicația pune la dispoziție fie o selecție de imagini prestabile, fie posibilitatea de a încărca o imagine personalizată din galeria locală a utilizatorului. Adăugarea ingredientelor este susținută de un sistem de căutare asincronă, bazat pe ingredientele disponibile în API-ul extern TheMealDB. Utilizatorul poate selecta ingredientele, dar nu îi este permis să introducă ingrediente proprii. Apoi introduce cantitățile și unitățile de măsură aferente: grame, mililitri sau bucăți. Pentru fiecare ingredient, aplicația păstrează formatul: nume, cantitate, unitate.

Instrucțiunile de gătit sunt adăugate secvențial, fiecare într-un câmp individual, pentru a asigura claritatea și ușurința în urmărirea pașilor ulteriori. Toate datele sunt validate local înainte de a fi transmise spre stocare, pentru a preveni salvarea unor rețete incomplete. La trimiterea formularului, aplicația construiește obiectul recipeData, care conține toate câmpurile necesare și îl salvează în colecția recipes din Firebase Firestore.

```
const handleSubmit = async () => {
  if (!title.trim()) {
    Alert.alert('Error', 'Please enter a title');
    return;
  }
  if (!area) {
    Alert.alert('Error', 'Please select an area');
    return;
  }
  if (!category) {
    Alert.alert('Error', 'Please select a category');
    return;
  }
  if (ingredients.length === 0) {
    Alert.alert('Error', 'Please add at least one ingredient');
    return;
  }
  if (ingredients.some(ing => !ing.name || !ing.quantity)) {
    Alert.alert('Error', 'Please fill in all ingredient fields');
    return;
  }
  if (instructions.some(inst => !inst.trim())) {
    Alert.alert('Error', 'Please fill in all instruction fields');
    return;
  }
  setLoading(true);
  try {
    const user = auth.currentUser;
    if (!user) {
      Alert.alert('Error', 'You must be logged in to create a recipe');
      return;
    }
    let authorName = user.displayName;
    if (!authorName) {
      authorName = 'Anonymous';
    }
    const recipeData = {
      title: title.trim(),
      ingredients: ingredients.map(ing => ({
        name: ing.name.trim(),
        quantity: ing.quantity.trim(),
        unit: ing.unit
      })),
      instructions: instructions.map(inst => inst.trim()),
      category,
      area,
      imageIndex: selectedImage,
      ...uploadedImage ? { uploadedImage } : {},
      authorId: user.uid,
      authorName,
      createdAt: new Date().toISOString(),
    };
    console.log('Saving recipe with data:', recipeData);
    await addDoc(collection(db, 'recipes'), recipeData);
    Alert.alert('Success', 'Recipe created successfully!');
    router.back();
  } catch (error) {
    console.error('Error creating recipe:', error);
    Alert.alert('Error', 'Failed to create recipe');
  } finally {
    setLoading(false);
  }
};
```

Secvență de cod 11 – Salvarea unei rețete noi în Firestore

Pentru optimizarea experienței de selecție a ingredientelor, aplicația afișează o fereastră modală cu un FlatList de sugestii filtrate în timp real în funcție de textul introdus. Astfel, erorile de denumire sunt reduse, iar selecția este accelerată.

```
const getFilteredIngredients = () => {
  const search = ingredientSearch.trim().toLowerCase();
  if (!search) {
    return apiIngredients.filter(ing => !ingredients.some(sel => sel.name === ing));
  }
  const startsWith = apiIngredients.filter(
    ing => !ingredients.some(sel => sel.name === ing) && ing.toLowerCase().startsWith(search)
  ).sort((a, b) => a.localeCompare(b));
  const contains = apiIngredients.filter(
    ing => !ingredients.some(sel => sel.name === ing) && !ing.toLowerCase().startsWith(search) && ing.toLowerCase().includes(search)
  ).sort((a, b) => a.localeCompare(b));
  return [...startsWith, ...contains];
};
```

Secvență de cod 12 – Selecția ingredientelor din listă filtrabilă

Rețeta este salvată cu un authorId unic, permitând filtrarea ulterioară după autor sau oferirea opțiunii de ștergere doar autorului. Astfel, funcționalitatea este complet integrată cu mecanismele de securitate și stocare deja definite în Firestore, structura unei rețete fiind exemplificată în figura 12.

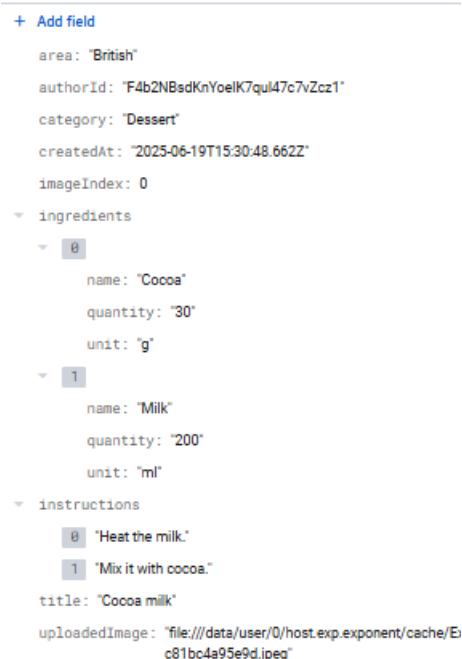


Figura 12 - Exemplu de rețetă adăugată de către un utilizator

4.3.9. PROCESUL DE PLANIFICARE A REȚETELOR

Funcționalitatea de planificare a meselor permite utilizatorului să aloce o rețetă unei zile și unui tip de masă (mic dejun, prânz sau cină), facilitând astfel organizarea în

avans a preparatelor culinare. Această caracteristică răspunde nevoilor utilizatorilor care își doresc o gestionare eficientă a timpului și a ingredientelor disponibile.

La nivel de interfață, accesarea opțiunii de programare se face printr-un buton dedicat situat în pagina unei rețete. Acesta deschide un dialog modal (PlanRecipeModal) în care utilizatorul selectează data și masa corespunzătoare. Odată confirmată alegerea, se actualizează planul alimentar al utilizatorului în colecția mealPlan din Firestore, utilizând funcția updateDateMealSlot.

```
const handlePlanRecipe = async (date: string, mealType: string, replaceExisting?: boolean) => {
    setPlanLoading(true);
    try {
        if (!recipe) {
            Alert.alert('Error', 'No recipe loaded.');
            setPlanLoading(false);
            return;
        }

        const success = await updateDateMealSlot(date, mealType, {
            id: recipe.id,
            title: recipe.title,
            image: recipe.image,
        }, replaceExisting);

        if (success) {
            setPlanModalVisible(false);
            Alert.alert('Planned!', `Recipe planned for ${date} (${mealType})`);

            loadExistingMealPlans();
        }
    } catch (error) {
        Alert.alert('Error', 'Failed to plan recipe.');
    } finally {
        setPlanLoading(false);
    }
};
```

Secvență de cod 13 – Alocarea unei rețete în planul alimentar

Pentru selecția intuitivă a zilei, aplicația utilizează un calendar personalizat, afișat prin componenta CalendarModal. Aceasta se bazează pe biblioteca react-native-calendars și aplică un stil coherent cu restul aplicației. Utilizatorul poate alege o dată din calendar, care va fi marcată vizual, împreună cu săptămâna corespunzătoare și ziua curentă. Marcajele suplimentare pot indica prezența altor rețete deja planificate.

Datele sunt structurate în Firestore după modelul users/{uid}/mealPlan/{date}, fiecare document conținând sloturi pentru mesele principale ale zilei (breakfast, lunch, dinner). În cazul în care un slot este deja ocupat, aplicația semnalează această situație, solicitând confirmarea utilizatorului pentru înlocuirea rețetei existente. Funcționalitatea permite astfel o flexibilitate sporită în configurarea meniului zilnic.

4.4. INTERACȚIUNEA CU THEMEALDB

O componentă esențială în arhitectura aplicației descrise este reprezentată de integrarea cu API-ul public TheMealDB. Acesta joacă un rol important în furnizarea conținutului dinamic al aplicației, permitând accesul la o colecție vastă de rețete culinare, organizate în funcție de ingrediente, categorii, arii geografice și alte criterii. TheMealDB este un API de tip REST care expune date structurate în format JSON, optimizat pentru consum rapid de către aplicații web și mobile. Tipul acesta de interfață, cunoscut sub denumirea completă de Representational State Transfer (REST), reprezintă un model arhitectural definit inițial de Roy Fielding în teza sa de doctorat [25].

Caracteristicile definitorii ale unui API REST includ organizarea logică a resurselor, folosirea metodelor HTTP (GET, POST, PUT, DELETE) pentru operații CRUD, și respectarea principiului de „statelessness”, conform căruia fiecare cerere adresată serverului trebuie să conțină toate informațiile necesare pentru procesare, fără a depinde de contextul anterior al unei sesiuni. Acronimul „CRUD” este pentru cele patru operațiuni de bază pe care le poți efectua asupra resurselor: „Create” (creare - asociată cu POST), „Read” (citire - asociată cu GET), „Update” (actualizare - asociată cu PUT) și „Delete” (stergere - asociată cu DELETE). De asemenea, trebuie să expună o interfață uniformă și să ofere reprezentări standardizate ale resurselor, cel mai frecvent în format JSON, pentru a asigura interoperabilitatea cu diverse tipuri de clienți.

În contextul actual, integrarea cu acest API respectă aceste principii. Apelurile sunt realizate folosind metoda HTTP GET, iar structura răspunsurilor este JSON, ceea ce asigură ușurință în analiza și prelucrarea datelor în cadrul aplicației. Resursele sunt accesate prin endpoint-uri semnificative semantic. Aceste endpoint-uri reprezintă resurse REST distincte, fiecare adresabilă printr-un URI unic, conform arhitecturii REST [26].

Interacțiunea cu acest API este implementată printr-un set de funcții asincrone care utilizează metoda `fetch()` pentru realizarea apelurilor HTTP. Aceste funcții sunt declanșate în momente cheie ale interfeței prin intermediul funcțiilor hook React Native (în special `useEffect` și `useState`), facilitând încărcarea reactivă a datelor. Arhitectura modulară permite astfel separarea clară între logica de business și prezentare, fiecare interacțiune cu API-ul fiind tratată ca o unitate logică reutilizabilă.

Una dintre funcționalitățile principale bazate pe API este căutarea rețetelor în funcție de ingrediente selectate de utilizator. Acest aspect este implementat prin apelarea endpoint-ului `filter.php`, care acceptă drept parametru o listă de ingrediente separate prin virgulă. Spre deosebire de o căutare simplă după un singur ingredient, aplicația este capabilă să trimită o listă compusă de ingrediente către API-ul TheMealDB, sub forma <https://www.themealdb.com/api/json/v1/1/filter.php?i=chicken,garlic,onion>, obținând astfel numai rețetele care conțin toate ingredientele specificate. Această

abordare permite personalizarea rezultatelor în funcție de conținutul real al cămării utilizatorului și evidențiază o adaptare funcțională intelligentă la constrângerile API-ului. La nivel de implementare, ingredientele sunt colectate într-un tablou și concatenate într-un sir de caractere, fiind trimise într-un singur request asincron. Rezultatul este o listă de rețete minimale (ID, titlu și imagine), care este apoi afișată utilizatorului.

Pentru afișarea detaliată a unei rețete, aplicația utilizează endpoint-ul `lookup.php`, care oferă informații extinse despre un preparat selectat. Prin apelul <https://www.themealdb.com/api/json/v1/1/lookup.php?i={id}>, se obțin date precum numele complet al rețetei, lista detaliată a ingredientelor și cantităților, instrucțiuni de preparare, categoria culinară, originea geografică, precum și un link asociat. Aceste informații sunt apoi asociate într-o interfață clar structurată, concepută să ghideze utilizatorul pas cu pas în procesul de gătit. Acest flux de lucru, în care datele sunt obținute secvențial în funcție de acțiunile utilizatorului, demonstrează o arhitectură de tip „client-driven” în care aplicația mobilă controlează în totalitate starea și afișajul. Toate transformările de stare se produc local, pe baza răspunsurilor asincrone primite de la API, ceea ce respectă pe deplin constrângerile REST privind separarea dintre client și server.

Pe lângă aceste două fluxuri principale, sunt integrate și alte endpoint-uri relevante din TheMealDB pentru a extinde funcționalitatea aplicației. Printre acestea se numără:

- `list.php?i=list` - utilizat pentru a obține lista completă de ingrediente disponibile în API. Aceste date sunt folosite pentru validarea inputului, completarea automată și generarea sugestiilor contextuale;
- `list.php?c=list` - pentru obținerea categoriilor culinare, utile în filtrarea rețetelor;
- `list.php?a=list` - pentru identificarea ariilor geografice (cum ar fi „Italian”, „Mexican”), folosite pentru segmentarea tematică a rețetelor;

Această utilizare diversificată a API-ului oferă aplicației un grad de dinamism și adaptabilitate și susține mai multe scenarii de utilizare, de la căutări intenționate până la explorare liberă. De asemenea, interacțiunea cu acest API este completată de tratamente robuste de gestionare a erorilor. Fiecare apel este însoțit de un mecanism try-catch care interceptează eventualele eșecuri de rețea sau de structură a răspunsului. În caz de eroare, utilizatorul este informat printr-un mesaj adecvat. Am aplicat acest tip de tratament al erorilor, deoarece conexiunile instabile au dus frecvent la eșecuri de rețea. Prin gestionarea locală a excepțiilor și informarea utilizatorului, aplicația poate rămâne funcțională și previzibilă, evitând afișarea de date corupte sau lipsă.

Deși API-ul TheMealDB nu impune oficial o limită de acces (rate limiting), am folosit un mecanism de cache local, în vederea posibilei extinderi viitoare a aplicației cu alte surse externe ce pot include asemenea limitări sau chiar posibilitatea de a întâmpina limitări cu sursa actuală. Acest cache permite memorarea temporară a răspunsurilor pentru o durată prestabilă, evitând cereri repetitive inutile către server. Un exemplu

concret al acestor mecanisme poate fi observat în fișierul unde sunt centralizate funcțiile de interacțiune cu API-ul, împreună cu sistemul de cache local. Funcția `fetchWithCache` este responsabilă pentru gestionarea cererilor HTTP, verificând mai întâi dacă răspunsul există deja în memorie și este valabil (timp de o oră), iar dacă nu, efectuează o cerere proaspătă și salvează rezultatul în cache. Ca urmare, utilizatorul beneficiază și de timpi de răspuns semnificativ mai mici, deoarece datele deja obținute sunt disponibile instant din memorie, mai ales în scenarii de navigare sau căutări recente. De exemplu, dacă aceeași cerere este făcută din nou, atunci datele sunt returnate instant din cache, nu se mai face un nou request către API.

Justificarea alegerii TheMealDB ca sursă externă de date s-a bazat pe o serie de criterii clare: disponibilitatea gratuită, lipsa necesității unei autentificări prin token, răspunsuri rapide și consistente în format JSON, o documentație accesibilă și o bază de date suficient de cuprinsătoare pentru prototiparea funcționalităților unei aplicații culinare. Alte astfel de API-uri culinare, precum Spoonacular sau Edamam, impun limitări mai stricte privind rata de acces și necesitatea autentificării prin chei de acces sau abonamente premium, adică plătite. În contrast, TheMealDB oferă un nivel suficient de deschidere și autonomie pentru aplicații educaționale, de tip proiect sau prototipuri, ceea ce a reprezentat, pentru mine, un criteriu decisiv în selecția sa pentru EasyChef. În plus, API-ul ales oferă suport pentru multiple moduri de filtrare și o gamă largă de informații, ce contribuie la o experiență de utilizare interactivă și plăcută. și frecvența relativ redusă a modificărilor structurale în răspunsuri a fost un criteriu considerabil, deoarece contribuie la stabilitatea aplicației în timp.

În concluzie, interacțiunea cu API-ul TheMealDB este fundamentalul logicii dinamice a aplicației, fiind sursa principală a conținutului din interiorul aplicației. Alegerea de integrare a acestuia în proiect reflectă aplicarea riguroasă a principiilor de proiectare software modernă, cu o infrastructură flexibilă, fiabilă și ușor de întreținut, în care sursa externă de date este exploatată eficient pentru a deservi nevoile reale ale utilizatorului final.

4.5. INTERACȚIUNEA CU FIREBASE

În acest proiect, am utilizat Firebase ca infrastructură principală pentru gestionarea autentificării utilizatorilor, stocarea datelor aplicației și salvarea fișierelor media. Această alegere tehnologică a oferit un set complet de servicii backend gestionate, ușor de integrat în aplicații mobile bazate pe React Native și capabile să scaleze odată cu creșterea numărului de utilizatori. Ecosistemul folosit a pus la dispoziție următoarele:

Firebase Authentication - pentru autentificare, Firestore - pentru stocarea datelor, Firebase Storage - pentru gestionarea fișierelor media (imagini rețete), Context și AsyncStorage - pentru persistența locală a sesiunilor.

Pentru autentificare, se folosește Firebase Authentication, care permite înregistrarea și conectarea utilizatorilor pe baza unei adrese de e-mail și a unei parole. Procesul este completat cu validări locale de securitate, iar, după autentificare, datele utilizatorului sunt stocate temporar în memoria locală a dispozitivului prin AsyncStorage. Acest mecanism oferă utilizatorului posibilitatea de a rămâne conectat, chiar și după închiderea aplicației, îmbunătățind astfel experiența de utilizare. Datele returnate de Firebase, cum ar fi UID-ul unic și adresa de e-mail, sunt accesate din contextul aplicației, printr-un AuthContext global, ceea ce permite afișarea personalizată a conținutului și filtrarea datelor specifice fiecărui cont.

Pentru stocarea datelor dinamice, aplicația folosește serviciul Cloud Firestore, o bază de date NoSQL în timp real. În Firestore sunt gestionate documentele corespunzătoare fiecărui utilizator (profil, preferințe, cămară, plan alimentar), precum și rețetele globale sau personale. Accesul la aceste date se face prin operații asincrone de tip getDoc, addDoc, updateDoc sau deleteDoc, iar sincronizarea este imediată. Structura colecțiilor reflectă organizarea logică (de exemplu users, recipes, completedRecipes, favorites), fiecare dintre ele având documente identificate prin UID-ul Firebase sau ID-uri generate automat. Această organizare permite filtrarea usoară a datelor și menținerea consistenței între componente. Pentru conținutul media, cum sunt imaginile asociate rețetelor, am folosit un serviciu extern specializat (Cloudinary). Linkul generat în urma încărcării este stocat ca referință în Firestore, fiind ulterior folosit pentru afișarea imaginilor.

Persistența sesiunii și accesul constant la datele utilizatorului sunt posibile prin salvarea informațiilor de autentificare în AsyncStorage și prin utilizarea contextelor React Native, care oferă acces la UID, email și alte date în toate componentele aplicației. Astfel, chiar și după redeschiderea aplicației, starea utilizatorului este păstrată, iar interacțiunea cu baza de date este imediată și personalizată.

Pentru a asigura protecția datelor, am aplicat reguli stricte de securitate în Firestore. Fiecare utilizator are acces doar la propriile documente, acest lucru fiind controlat la nivelul backend-ului Firebase prin reguli de acces care validează UID-ul autentificat. Drept exemplu, regulile interzic modificarea datelor altui utilizator sau accesul la colecții sensibile fără autentificare. Consider că acest nivel de izolare a datelor este vital într-o aplicație care gestionează informații personale și preferințe individuale. Un exemplu semnificativ este controlul accesului la rețete, unde orice utilizator autentificat poate citi și crea rețete, dar doar autorul rețetei poate modifica sau șterge documentul.

```
match /recipes/{recipeId} {
    allow read: if isAuthenticated();
    allow create: if isAuthenticated();
    allow update, delete: if isAuthenticated() && isOwner(resource.data.authorId);
}
```

Secvență de cod 14 – Reguli pentru rețete din Firebase

În concluzie, Firebase oferă, pentru proiectul de față, o platformă completă care gestionează toate componentele esențiale ale aplicației (de la autentificare până la stocarea conținutului utilizatorilor) într-un mod sigur, eficient și scalabil. Integrarea sa nativă cu React Native și suportul extins pentru operații asincrone fac ca dezvoltarea și extinderea aplicației să fie simplificate considerabil.

4.6. INTERACȚIUNEA CU CLOUDINARY

Imaginiile rețetelor reprezintă un element esențial pentru experiența vizuală și implicarea utilizatorului. Pentru a permite încărcarea imaginilor proprii de către utilizatori și gestionarea acestora într-un mod performant, am integrat serviciul Cloudinary - o platformă cloud specializată în manipularea, stocarea și livrarea de fișiere media. Această integrare oferă avantaje semnificative față de alternativele tradiționale, precum Firebase Storage, datorită vitezei de livrare prin CDN, capacitații de scalare și funcționalităților avansate de transformare a imaginilor.

Procesul de interacțiune cu Cloudinary este declanșat în cadrul componentei RecipeForm.tsx, atunci când utilizatorul accesează, printr-un buton dedicat, ecranul de adăugare a unei rețete și optează pentru atașarea unei imagini din galeria personală a dispozitivului mobil. Interfața utilizează biblioteca expo-image-picker, care permite selectarea unei imagini locale. Ulterior, imaginea este prelucrată într-un obiect de tip FormData, codificată și transmisă printr-o cerere HTTP de tip POST către endpoint-ul public Cloudinary (https://api.cloudinary.com/v1_1/dr25rls8t/image/upload). În funcția uploadToCloudinary, definită separat în fișierul cloudinaryUpload.js, sunt atașate și alte informații necesare, precum upload_preset, care reprezintă cheia publică de acces configurată în contul Cloudinary al dezvoltatorului. Aceasta permite utilizarea unui canal de încărcare nesecurizat (unsigned), ideal pentru aplicații în fază de prototip cum este cea prezentată în acest proiect.

Răspunsul primit de la Cloudinary conține un obiect JSON în care se regăsesc diverse metadate despre fișierul încărcat (cum ar fi dimensiune, format, ID intern), dar, mai important, include un câmp secure_url - un link public, securizat prin HTTPS, care duce direct la imaginea stocată. Acest link este salvat în baza de date Firestore, în câmpul uploadedImage al documentului de tip rețetă din colecția recipes. În acest mod, imaginea nu este stocată direct în baza de date, ci este doar referențiată, reducând consumul de spațiu și costurile asociate cu stocarea directă în Firebase.

În cazurile în care utilizatorul nu încarcă nicio imagine proprie, el poate alege una dintre opțiunile prestabilite, selectate local dintr-un set predefinit. În cazul în care nu alege nimic, aplicația asociază rețetei prima dintre acele imagini. Alegerea este controlată printr-o proprietate imageIndex, care face trimitere la o imagine statică dintr-un tablou de imagini. Acest mecanism oferă un fallback vizual consistent și ușurează procesul de completare a formularului pentru utilizatorii care preferă simplitatea.

Avantajele folosirii Cloudinary includ nu doar livrarea rapidă a fișierelor prin infrastructura proprie de tip CDN, ci și posibilitatea de a aplica transformări în timp real (redimensionare, compresie, conversie de format) direct prin parametrii URL. În versiunea curentă a aplicației, aceste funcționalități avansate nu sunt activate, dar arhitectura aleasă permite integrarea lor facilă în viitor. Astfel, aplicația poate evoluă ușor spre funcționalități precum personalizarea calității imaginilor în funcție de rețea sau scalarea automată a dimensiunii pe baza rezoluției ecranului.

În concluzie, integrarea cu Cloudinary răspunde cerințelor de performanță și flexibilitate ale aplicației. Această alegere reflectă o decizie tehnică informată, cu un echilibru între eficiență, simplitate și extindere ulterioară a funcționalităților.

5. TESTARE ȘI PROBLEME ÎNTÂMPINATE

Procesul de testare reprezintă o etapă relevantă în ciclul de viață al unei aplicații mobile, având rolul de a valida funcționalitățile implementate, de a identifica eventuale erori și de a asigura o experiență coerentă pentru utilizatori. În cazul actual, testarea a fost realizată gradual, în paralel cu dezvoltarea modulelor principale, având ca obiectiv principal verificarea corectitudinii logice, a stabilității interfeței și a integrării eficiente cu serviciile externe.

Testele au vizat atât funcționalitatea aplicației în condiții uzuale, cât și comportamentul acestoria în situații neprevăzute, precum pierderea conexiunii la internet, date introduse greșit sau utilizatori fără drepturi de acces. De asemenea, în timpul dezvoltării, au fost urmărite și aspecte legate de performanță, sincronizare în timp real și consum de resurse. Problemele întâmpinate pe parcurs au fost documentate și soluționate progresiv, astfel încât aplicația să atingă un nivel satisfăcător de stabilitate și confort în utilizare.

5.1. PROCESUL DE TESTARE

Procesul de testare al aplicației descrise a fost realizat în mod manual, în paralel cu dezvoltarea funcționalităților. Testarea nu a urmat o metodologie formală sau automată, ci s-a bazat pe scenarii de utilizare reală, parcuse în mod direct de către mine, ca dezvoltator. Accentul a fost pus pe observarea comportamentului aplicației în condiții obișnuite de utilizare și pe verificarea modului în care componentele principale interacționează între ele.

Aplicația a fost testată pe dispozitive fizice, atât cu sistem de operare Android, cât și iOS, folosind platforma Expo pentru rulare locală. Interacțiunile au fost analizate din perspectiva unui utilizator obișnuit, iar testele au vizat în special fluxurile esențiale precum autentificarea, adăugarea de rețete, configurarea cămării, organizarea listei de cumpărături, planificarea meselor și salvarea preferințelor din profil.

Fiecare funcție nou implementată a fost testată imediat după dezvoltare, dar și ulterior, prin acțiuni directe în interfața aplicației. Am urmărit funcționarea corectă a fiecărui modul și felul în care datele sunt salvate, reactualizate și afișate. De asemenea, am observat comportamentul aplicației în cazul unor erori tipice, cum ar fi date lipsă, conexiune slabă la internet, navigare rapidă între pagini sau acțiuni repetate.

Deși testarea a fost limitată ca amploare și resurse, aceasta a oferit o imagine suficient de clară asupra stabilității aplicației și a zonelor care pot fi îmbunătățite. Cele mai importante observații obținute în urma acestui proces sunt prezentate în subsecțiunea următoare, alături de soluțiile identificate pe parcursul dezvoltării.

5.2. PROBLEME ȘI SOLUȚII

Testarea aplicației în stadiul actual de dezvoltare a evidențiat o funcționare general stabilă, cu reacții adecvate la majoritatea acțiunilor utilizatorului și fără erori critice persistente. În utilizarea obișnuită, interfața a răspuns corespunzător comenzilor, iar integrarea cu baza de date a funcționat corect.

Cu toate acestea, au existat unele comportamente imprevizibile în contexte specifice. De exemplu, în modulul de planificare a meselor, eliminarea sau modificarea rețetelor nu a actualizat întotdeauna imediat procentajul de completare, ceea ce crea confuzie pentru utilizator.

Am observat un alt comportament problematic în momentul navigării rapide între secțiuni sau file. În anumite cazuri, datele afișate nu reflectau modificările recente, de exemplu, o rețetă marcată ca finalizată nu apărea în lista de rețete complete. Pentru a evita aceste inconveniențe, am introdus metoda `useFocusEffect` care asigură reîncărcarea conținutului la revenirea pe o pagină și oferă o experiență mai predictibilă.

Inițial, implementarea funcției de scanare a codurilor de bare a fost concepută pentru formatele mai comune, utilizate frecvent pe produse alimentare standard. Cu toate acestea, în timpul testării, am constatat că aplicația întâmpina dificultăți în recunoașterea codurilor de bare mai scurte, cum sunt cele cu 8 cifre (de tip EAN-8), întâlnite adesea pe ambalaje de dimensiuni reduse. Aceste coduri nu erau detectate corect sau generau erori de validare, din cauză că nu au fost luate în calcul în etapa inițială de dezvoltare. După identificarea acestei limitări, am ajustat logica de scanare pentru a recunoaște și aceste formate alternative pentru a extinde compatibilitatea cu mai multe tipuri de produse.

Setarea corectă a regulilor de acces în Firestore a fost, de asemenea, o provocare relevantă. Regulile inițiale nu acopereau complet cazurile particulare ale aplicației, iar uneori permiteau accesul neautorizat sau blocau acțiuni permise ale utilizatorului. Pentru remediere, am implementat funcții clare de validare a identității și apartenenței utilizatorului la datele accesate, precum `isAuthenticated()` și `isOwner(userId)`, în contextul configurației Firestore.

Integrarea rețetelor personale cu cele preluate din API a ridicat, la rândul ei, dificultăți tehnice. Inițial, rețetele adăugate de utilizatori nu au respectat același format cu

cele externe, ceea ce a dus la afișări incorecte sau la erori în momentul filtrării. Pentru rezolvarea acestor incompatibilități, am implementat o logică de unificare a structurii de date și un mecanism de identificare clară a sursei fiecărei rețete, evitând astfel suprapunerile sau clasificările greșite.

Preferințele alimentare, deși stocate corect în profil, nu influențau imediat sugestiile de rețete, ceea ce crea impresia că setările nu sunt aplicate, deși pagina avea nevoie doar de o reîncărcare. Pentru a rezolva această problemă, am modificat codul astfel încât preferințele alimentare să fie preluate în timp real din Firestore folosind o metodă (onSnapshot). Astfel, funcția de filtrare a rețetelor primește întotdeauna datele actualizate, iar informațiile sunt sincronizate la fiecare accesare a paginii de rețete.

Într-o etapă anterioară a proiectului, am utilizat stocarea locală prin AsyncStorage pentru salvarea rețetelor preferate. Totuși, această abordare a dus la pierderea informațiilor după deconectare și reconectare, deoarece nu erau sincronizate cu contul de utilizator din Firebase. Problema a fost soluționată prin mutarea logicii de salvare în Firestore, pentru a asigura persistența datelor în funcție de utilizatorul conectat.

De asemenea, o problemă majoră a fost legată de standardizarea unităților de măsură pentru ingrediente. Rețetele din sursa externă prezintau cantități exprimate în formate diverse, cum ar fi „dash”, „oz”, „can” sau chiar fără o unitate de măsură, care au generat confuzii și nu au permis manipularea eficientă și constantă a acestora. A fost necesară o normalizare parțială, astfel încât ingredientele să fie afișate într-un mod consecvent, folosind termeni uzuali precum grame (g), mililitri (ml), bucăți (pcs) sau „după gust” (to taste). Deși sistemul actual nu acoperă toate cazurile posibile, el contribuie semnificativ la o experiență mai clară pentru utilizator.

Pe ansamblu, testarea a arătat că aplicația oferă o experiență coerentă și funcțională în condiții normale de utilizare. Componentele sunt bine legate între ele, datele sunt gestionate corect, iar feedback-ul vizual oferit utilizatorului este în mare parte intuitiv. Problemele apărute pot fi adresate ușor în etape ulterioare de rafinare și optimizare.

6. MANUAL DE UTILIZARE

6.1. ÎNCEPEREA UTILIZĂRII

Aplicația mobilă descrisă în această lucrare este accesibilă în stadiul actual de dezvoltare prin intermediul platformei Expo, fără a necesita compilarea sub forma unui pachet instalabil sau publicarea într-un magazin oficial. Rularea se realizează local, de pe calculatorul dezvoltatorului, iar afișarea interfeței pe un dispozitiv mobil compatibil se face prin scanarea unui cod de acces generat automat.

Pentru lansare, utilizatorul are nevoie de aplicația Expo Go instalată pe telefon. Aceasta este disponibilă gratuit și permite conectarea cu mediul de dezvoltare local. După inițializarea proiectului de pe stația de lucru, comanda de rulare generează un cod QR unic. Scanarea acestuia din Expo Go, folosind camera foto a dispozitivului, declanșează deschiderea imediată a aplicației pe dispozitiv, fără a necesita pași suplimentari de instalare. Această metodă a permis testarea completă a funcționalităților într-un mediu real de utilizare. Toate modulele aplicației, printre care se regăsesc autentificarea, salvarea datelor, interacțiunea cu baza de date și filtrarea conținutului, sunt complet operaționale, funcționează în condiții normale. Totodată, modificările efectuate în codul sursă sunt sincronizate automat pe dispozitiv, ceea ce a facilitat depanarea și optimizarea.

Primul ecran afișat la deschiderea aplicației este cel de autentificare. Utilizatorii care detin deja un cont se pot conecta folosind adresa de e-mail și parola. Optional, pot selecta păstrarea sesiunii active pe termen mai lung prin bifarea unei opțiuni dedicate. În lipsa acestei selecții, sesiunea este limitată la 24 de ore, după care utilizatorul își uită parola, este disponibilă o funcție de recuperare. După introducerea adresei de e-mail, aplicația trimite automat un mesaj cu instrucțiuni de resetare. Procesul nu necesită intervenții manuale sau contacte externe. După autentificare, utilizatorul este direcționat către ecranul principal al aplicației, unde sunt disponibile toate funcționalitățile importante: explorarea rețetelor, gestionarea ingredientelor, configurarea preferințelor sau planificarea meselor.

Pentru utilizatorii noi, fără cont, este disponibilă o pagină de înregistrare. Aceasta solicită o adresă de e-mail validă, alegerea unui nume și a unei parole, alături de confirmarea acesteia. Sistemul impune respectarea unor reguli stricte de securitate pentru parolă: minimum opt caractere, cel puțin o literă mare, una mică, o cifră și un caracter special. După verificarea tuturor condițiilor, contul este creat automat, iar utilizatorul este redirecționat către pagina principală a aplicației.

Această abordare de testare, prin rulare locală în Expo Go, a permis verificarea întregului flux de utilizare într-un mediu controlat și a demonstrat stabilitatea funcțională a aplicației din acest proiect, fără a fi necesară generarea unui fișier instalabil sau publicare oficială. Procesul de acces și utilizare este simplu, fără cerințe tehnice avansate și oferă o experiență completă pentru scopurile propuse în cadrul acestei lucrări.

6.2. GHID NAVIGARE ȘI FUNCȚII PRINCIPALE

Aplicația mobilă descrisă este structurată în patru secțiuni principale, organizate într-o bară de navigare fixă, plasată în partea inferioară a ecranului: Acasă, Rețete, Cămară și Profil. Această structură facilitează accesul rapid și intuitiv la toate funcționalitățile esențiale, asigurând o experiență coerentă de utilizare, chiar și pentru persoanele cu o experiență limitată în utilizarea aplicațiilor mobile.

Secțiunea Acasă funcționează ca punct de pornire și concentrează trei scenarii principale de interacțiune: explorarea rețetelor, generarea de sugestii pe baza ingredientelor din cămară și planificarea săptămânală a meselor. Primul scenariu trimită utilizatorul către pagina generală de „Rețete”, unde poate explora întreaga colecție disponibilă. Al doilea scenariu implică utilizarea ingredientelor deja existente în cămară pentru generarea automată de rețete. Aplicația analizează atât lista ingredientelor disponibile, cât și preferințele și restricțiile alimentare ale utilizatorului, oferind sugestii de rețete care pot fi pregătite imediat sau care necesită completări minime. Fiecare rețetă este însotită de un indicator vizual al gradului de compatibilitate cu conținutul din cămară. Al treilea scenariu permite planificarea meselor pentru întreaga săptămână, sau pentru următoarele săptămâni, cu alocarea rețetelor preferate pentru fiecare zi. Interfața de planificare este simplă, ușor de editat, și oferă posibilitatea de a salva sau modifica ulterior planurile create.

Secțiunea Rețete oferă acces complet la conținutul culinar al aplicației. În partea superioară a paginii, sunt afișate câteva sugestii de rețete personalizate, generate în funcție de preferințele alimentare și ingredientele excluse de utilizator. Aceste sugestii contribuie la simplificarea procesului de selecție, aducând în prim-plan rețete relevante pentru profilul individual. Sub zona de sugestii se regăsesc trei opțiuni: Explorare rețete, Rețete salvate și Adăugare rețetă. În zona de explorare, utilizatorul poate consulta toate rețetele disponibile, inclusiv cele publice și cele create de el însuși. Fiecare rețetă este prezentată într-un card vizual ce conține titlul, imaginea, zona de origine a rețetei și autorul, dacă este cazul. În același card vizual, utilizatorul are opțiunea de a salva rețete prin simpla activare a butonului de „favorite”. Lista poate fi filtrată după tipul mesei (mic

dejun, prânz, cină), zona de origine (italiană, asiatică, indiană) sau categorie alimentară (vegetarian, desert, carne de porc, paste). Opțiunea „Rețete salvate” reunește toate rețetele marcate anterior de utilizator pentru referințe ulterioare. Fiecare rețetă din listă conține un rezumat al ingredientelor, evidențiind care dintre ele sunt deja disponibile, care lipsesc și care intră în conflict cu preferințele sau alimentele interzise de utilizator. Această previzualizare facilitează luarea unor decizii rapide [27].

La selectarea unei rețete, utilizatorul poate accesa întreaga fișă: imagine ilustrativă, un link către sursa originală a rețetei (unde este cazul), pașii de preparare, ingredientele și porțiile (care pot fi ajustate între 0.5 și 6, cu recalcularea automată a cantităților ingredientelor). Dacă ingredientele lipsesc, aplicația semnalează imposibilitatea începerii preparării. Dacă ingredientele sunt suficiente, utilizatorul poate începe gătitul printr-un buton dedicat. După finalizare, rețeta poate fi marcată ca fiind completată, iar aplicația solicită o evaluare personală, sub formă de scor și observații. Acestea sunt salvate într-o secțiune privată („Rețete completate”), accesibilă doar din pagina de profil a utilizatorului. Opțiunea de „Adăugare rețetă” deschide un formular cu mai multe câmpuri: titlul rețetei, zona, categoria, lista de ingrediente, pașii de preparare și selecția unei imagini reprezentative. Rețeta adăugată este stocată în profilul utilizatorului, dar este integrată și în colecția generală de rețete, devenind astfel accesibilă pentru toți ceilalți utilizatori, contribuind la îmbogățirea continuă a platformei.

Secțiunea Cămară permite gestionarea completă a ingredientelor. Pagina este organizată în patru zone funcționale: Toate ingrediente, Ingrediente proprii, Scanare cod de bare și Lista de cumpărături. Zona de previzualizare din partea de sus afișează câteva ingrediente necesare, preluate din lista de cumpărături. Selectarea acestei zone deschide pagina completă a listei, unde sunt afișate toate ingredientele de achiziționat, împreună cu cantitățile și sortarea după data adăugării, alfabetic sau în funcție de rețeta pentru care sunt planificate.

Butonul „Toate ingredientele” afișează o listă completă cu ingrediente existente în sistem, extrase din baza externă. Utilizatorul poate căuta, sorta și adăuga ingrediente direct în cămara proprie, indicând cantitățile necesare. Butonul „Ingrediente proprii” reprezintă cămara proprie-zisă a utilizatorului. Aici sunt afișate toate ingredientele disponibile, cu posibilitatea de editare a cantităților, ștergere individuală sau resetare completă a listei. Interfața include un câmp de căutare și opțiuni de sortare. Funcționalitatea de „Scanare cod de bare” permite identificarea produselor prin scanarea ambalajului sau introducerea manuală a codului. Sistemul recunoaște mai multe formate de coduri și oferă informații precum numele ingredientului, brandul (dacă este disponibil) și categoria. În stadiul actual, această funcție are rol informativ, iar utilizatorul este avertizat că integrarea automată în sistemul de filtrare și planificare nu este încă activă, dar este prevăzută ca direcție viitoare de dezvoltare.

Secțiunea Profil conține informațiile personale și preferințele care influențează comportamentul aplicației. Aceasta include o imagine generică aleasă la înregistrare, numele utilizatorului (cu posibilitate de editare și istoric al modificărilor), precum și trei casete informative privind rețetele salvate, ingredientele disponibile și rețetele complete. Sub acestea se află alte trei casete importante, fiecare : „Preferințe alimentare”, „Ingrediente interzise” și „Rețetele mele”. Prima oferă cinci opțiuni activabile independent: vegetarian, vegan, fără gluten, fără lactate și fără nuci. A doua permite introducerea manuală a ingredientelor care nu trebuie să apară în rețete. Ambele opțiuni influențează direct sistemul de recomandare și filtrare. A treia casetă direcționează utilizatorul către o pagină ce afișează întreaga listă a rețetelor culinare adăugate de acesta, cu toate detaliile specifice fiecărei. La finalul paginii se regăsește butonul de deconectare, care închide sesiunea activă și redirecționează utilizatorul către pagina de autentificare.

6.3. FLUX DE UTILIZARE

Fluxul de utilizare în interiorul aplicației se bazează pe un parcurs coerent și progresiv, conceput astfel încât utilizatorul să parcurgă toate funcțiile relevante în mod intuitiv, fără a fi necesare explicații suplimentare sau intervenții externe. Acest flux începe din momentul accesării aplicației și continuă până la utilizarea avansată a funcționalităților, cum ar fi finalizarea unei rețete sau planificarea meselor săptămânale.

Primul contact cu aplicația este reprezentat de interfața de autentificare. Utilizatorul se poate conecta cu un cont existent sau poate crea unul nou, folosind un formular de înregistrare. Odată conectat, este direcționat către pagina principală, secțiunea Acasă, de unde poate accesa oricare dintre cele patru module majore.

Într-un scenariu tipic de folosire utilizatorul parurge următorii pași:

- Configurarea inițială a profilului - utilizatorul are opțiunea să își seteze preferințele alimentare și lista de ingrediente interzise. Aceste informații sunt esențiale pentru personalizarea conținutului și sunt utilizate ulterior în filtrarea rețetelor.
- Popularea cămării - pentru ca sistemul de recomandare să funcționeze eficient, utilizatorul este nevoie să adauge ingrediente disponibile în gospodărie. Acestea pot fi introduse manual din lista completă de ingrediente. Cantitățile pot fi ajustate, iar lista poate fi gestionată în orice moment.
- Explorarea rețetelor - după setarea preferințelor și adăugarea ingredientelor, utilizatorul poate accesa secțiunea de rețete. Aici, fie selectează una dintre

sugestiile compatibile cu ingredientele proprii, fie acceseează întreaga colecție din zona de explorare. După ce identifică o rețetă potrivită, are posibilitatea să o salveze, să îi vizualizeze detaliile sau să o planifice pentru o zi specifică.

- Utilizarea funcției de gătit - în momentul în care ingredientele necesare sunt disponibile integral în cămară, utilizatorul poate activa funcția pentru începerea gătitului. Aceasta deschide un mod de prezentare special, care afișează pașii rețetei într-un format ușor de urmărit și cantitățile adaptate numărului de porții selectat. După parcurgerea întregului proces, rețeta poate fi marcată ca fiind finalizată, iar utilizatorul este invitat să evalueze experiența printr-un formular simplu.
- Planificarea meselor - în paralel cu gătitul sau explorarea rețetelor, utilizatorul poate accesa modulul de planificare săptămânală. Aici poate aloca rețete pentru fiecare zi, în funcție de programul personal sau de ingredientele disponibile.
- Gestionarea listei de cumpărături - pentru rețetele neacoperite integral de cămară, utilizatorul poate adăuga automat ingredientele lipsă în lista de cumpărături. Această listă este ușor de gestionat și permite bifarea produselor achiziționate, împreună cu specificarea cantităților.
- Accesarea și actualizarea profilului - în orice moment, utilizatorul poate reveni în secțiunea „Profil” pentru a vizualiza rețetele proprii, cele salvate, cele completate, ingredientele disponibile sau pentru a modifica preferințele și informațiile personale.

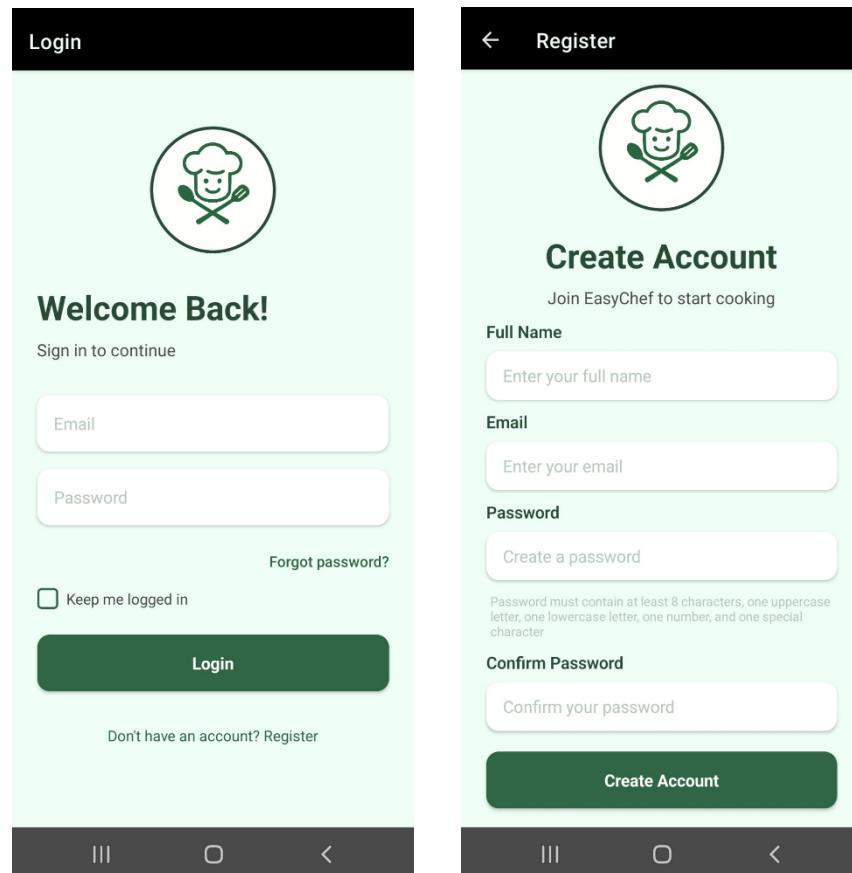
Întregul flux este susținut de sincronizări în timp real între aplicație și baza de date, astfel încât modificările efectuate în orice modul sunt reflectate imediat în celelalte secțiuni. Acest mecanism contribuie la coerenta comportamentului aplicației și la o experiență de utilizare predictibilă și logică.

Prin această structură, aplicația nu impune un traseu fix, ci unul flexibil, în care permite utilizatorului să decidă propriul mod de interacțiune, fie că dorește să exploreze rețete, să își organizeze mesele, să gătească direct sau doar să-își gestioneze stocurile. Totuși, logica de utilizare favorizează o progresie naturală: de la configurare, la explorare, acțiune și evaluare, fiecare pas susținând funcțional următorul.

6.4. SCENARIU COMPLET

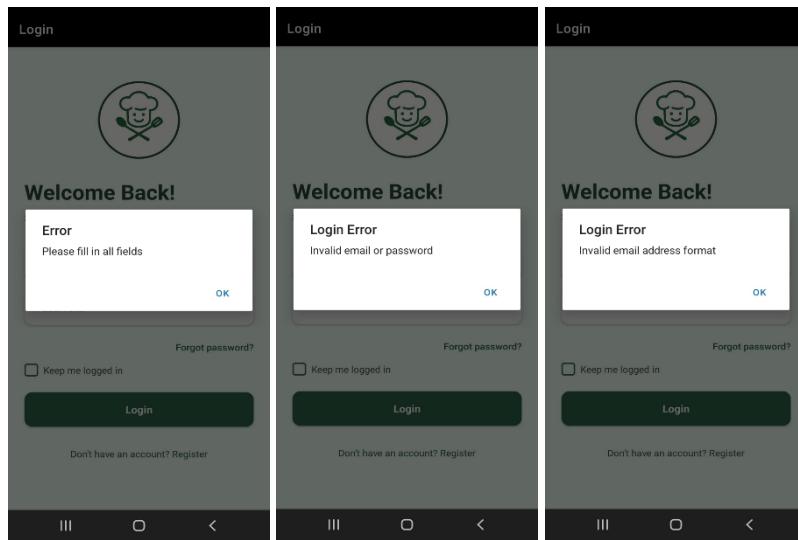
Pentru a evidenția modul concret în care aplicația este utilizată, se prezintă în continuare un scenariu complet, ce urmărește parcursul tipic al unui utilizator.

După ce instalează aplicația Expo Go pe telefonul mobil, utilizatorul scană codul QR generat local și accesează aplicația. Este întâmpinat de ecranul de autentificare, unde are opțiunea de a se conecta sau de a-și crea un cont nou. Alege înregistrarea, completează datele necesare, iar după validare este redirecționat către interfața principală.

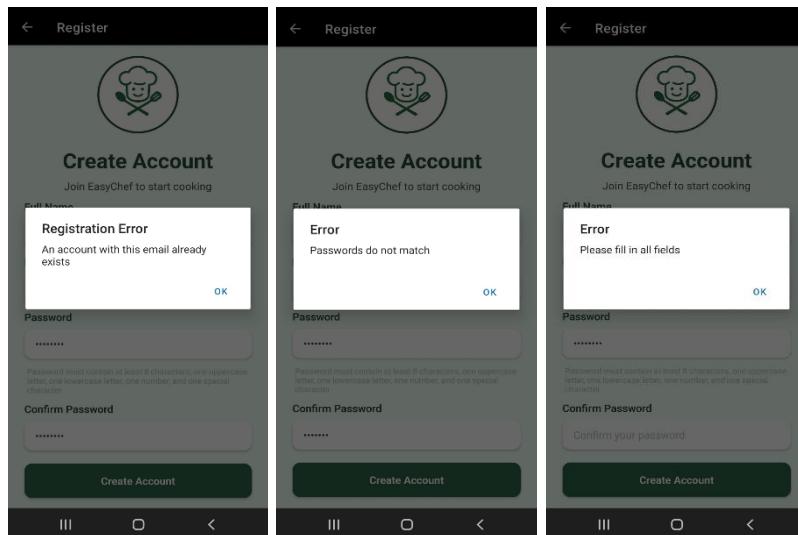


Figură 12 - Ecranele de autentificare și înregistrare

Pentru prevenirea confuziilor, aplicația afișează mesaje clare de eroare în cazul unor situații frecvente. Printre acestea se numără și dacă utilizatorul încearcă să se înregistreze cu o adresă de e-mail deja utilizată, dacă parola introdusă este incorectă la autentificare sau dacă unul dintre câmpuri este lăsat necompletat, aplicația semnalează prompt problema, permitând corectarea ei fără a întrerupe fluxul de utilizare. Acest mecanism de atenționare este prezent atât în pagina de autentificare, cât și în cea de înregistrare.

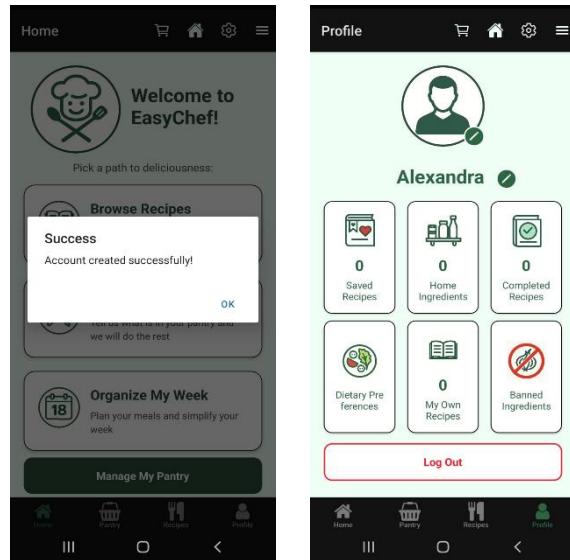


Figură 13 – Exemple de mesaje de eroare la autentificare

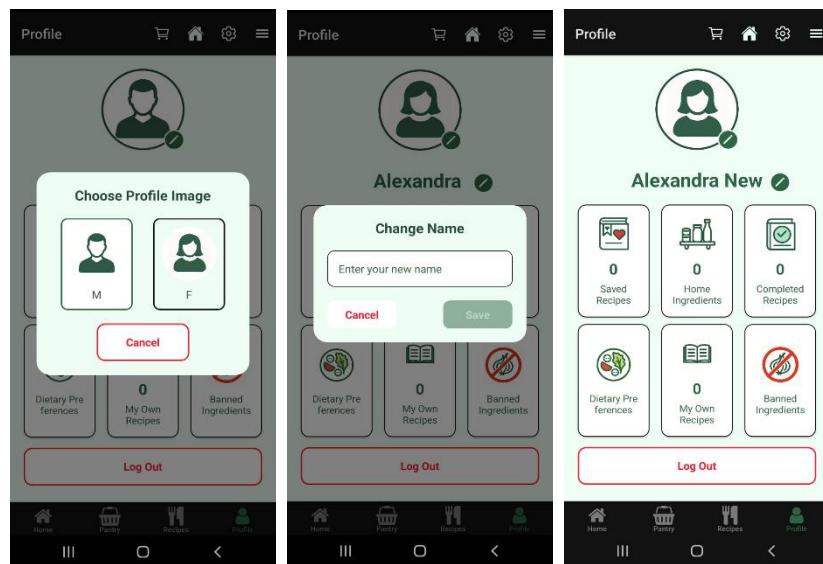


Figură 14 - Exemple de mesaje de eroare la înregistrare

După finalizarea cu succes a procesului de înregistrare, utilizatorului i se afișează un mesaj ce confirmă crearea contului. Ulterior, acesta este direcționat automat către ecranul de profil. În această secțiune, poate să își personalizeze imaginea de profil (alegând între cele două opțiuni implicite) și să modifice numele afișat. Deoarece este un cont nou, celelalte secțiuni ale aplicației sunt inițial goale și urmează să fie completate pe parcurs.



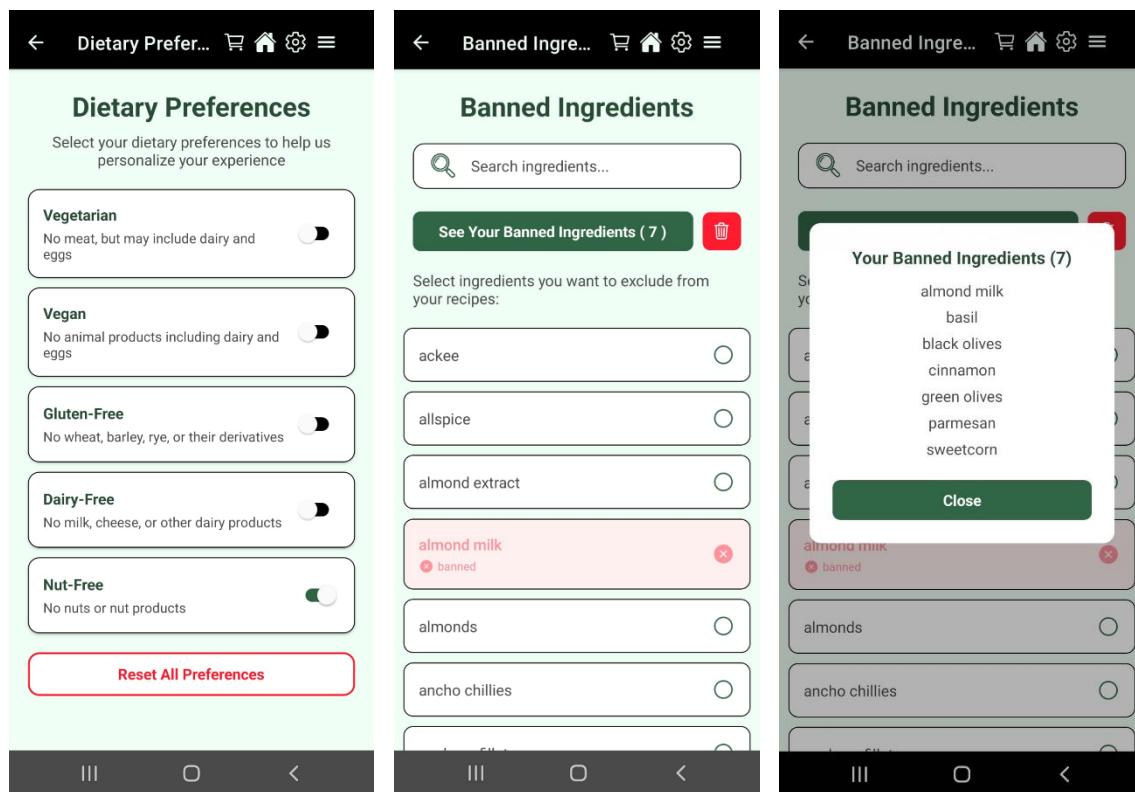
Figură 15 – Exemplu de creare a contului cu succes



Figură 16 – Exemplu de personalizare a profilului

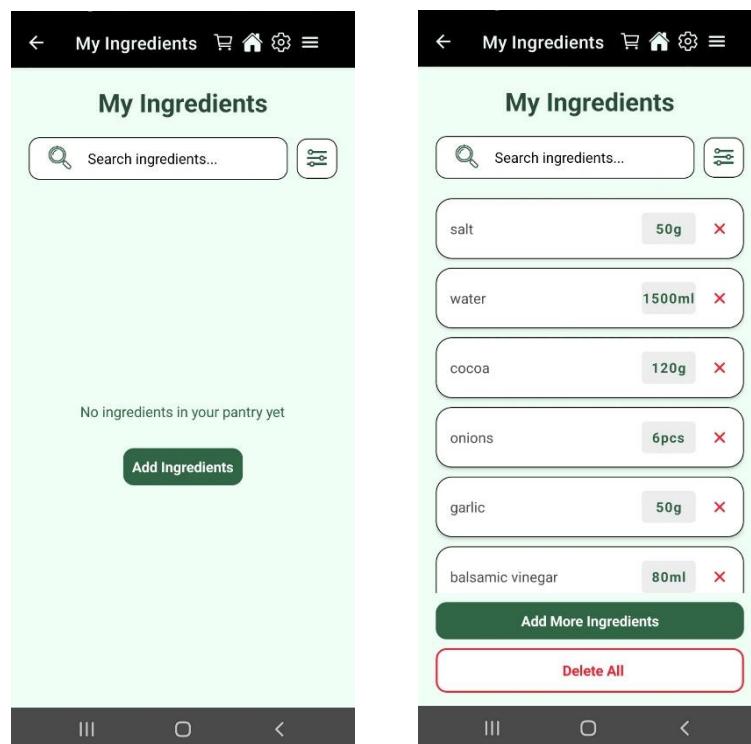
Imediat după crearea contului, utilizatorul să își poată configura opțiunile, pentru ca sugestiile de rețete oferite de aplicație să fie adaptate nevoilor și restricțiilor sale. În secțiunea de preferințe alimentare, acesta are la dispoziție cinci opțiuni de tip (vegetarian, vegan, fără gluten, fără lactate, fără nuci), care pot fi activate printr-un simplu comutator. Activarea uneia sau mai multora dintre aceste opțiuni va influența modul în care sunt afișate și filtrate rețetele în aplicație.

În completare, secțiunea de ingrediente interzise oferă posibilitatea de a marca individual ingredientele pe care utilizatorul dorește să le evite complet. Prin deschiderea listei complete de ingrediente, sortate alfabetic, utilizatorul poate naviga sau căuta rapid elementele dorite folosind bara de căutare integrată. Ingredientele selectate sunt marcate vizual, iar un buton dedicat permite afișarea rapidă a listei cu cele deja excluse, pentru a oferi transparentă și posibilitatea de revizuire.



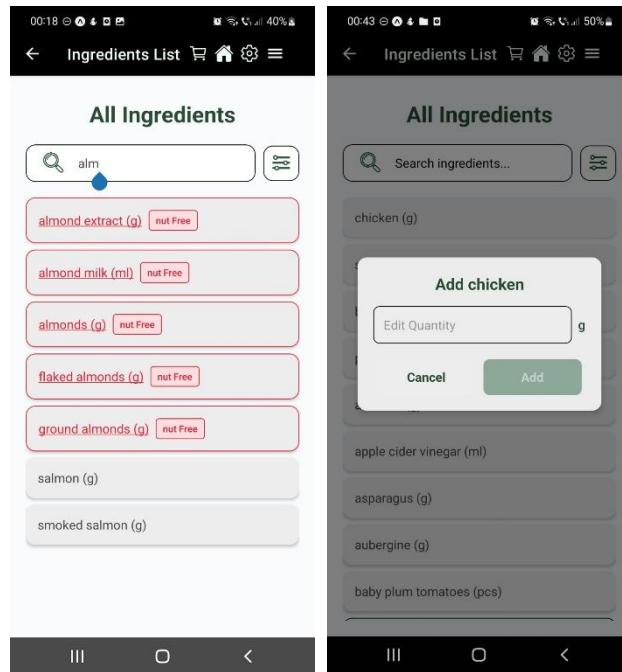
Figură 17 – Exemplu de gestionare a preferințelor alimentare

Următorul pas este completarea cămării digitale, accesibilă prin secțiunea despre ingrediente de acasă. Aici, utilizatorul poate începe să marcheze ingredientele pe care le are deja acasă. Ingredientele pot fi selectate manual din listă, care include un camp de căutare pentru identificare rapidă și posibilitatea de sortare după relevantă sau alfabetic.

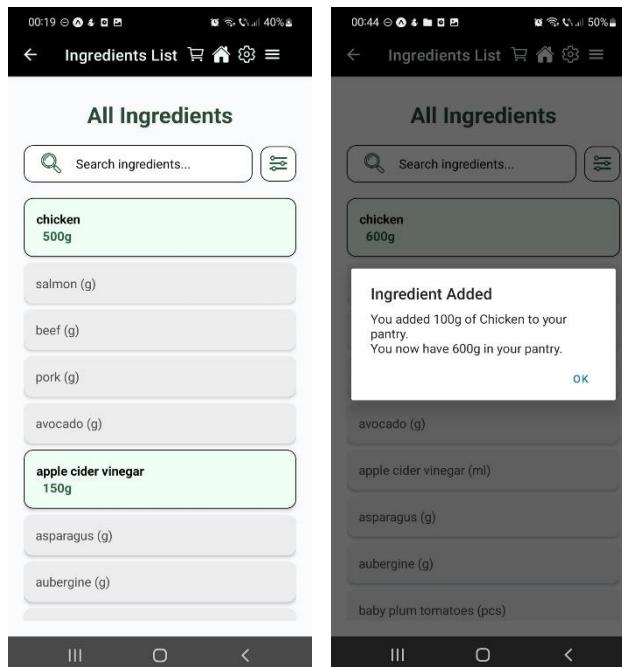


Figură 18 - Exemplu de populare a cămării

Pentru fiecare ingredient adăugat sau actualizat, utilizatorul este nevoit să specifică cantitatea, astfel încât aplicația să poată evalua mai precis compatibilitatea cu rețetele disponibile. Ingredientele care au fost marcate ca interzise sau fac parte din categorii de preferințe ale dietei alimentare sunt marcate vizual. Informațiile introduse sunt salvate în contul utilizatorului și sincronizate automat, urmând să fie utilizate în toate fluxurile aplicației care implică sugestii, planificare sau potrivire de rețete. La finalizarea fiecărei acțiuni de adăugare sau modificare, aplicația afișează un mesaj informativ, prin care utilizatorul este notificat cu privire la cantitatea actualizată a ingredientului respectiv din cămară.

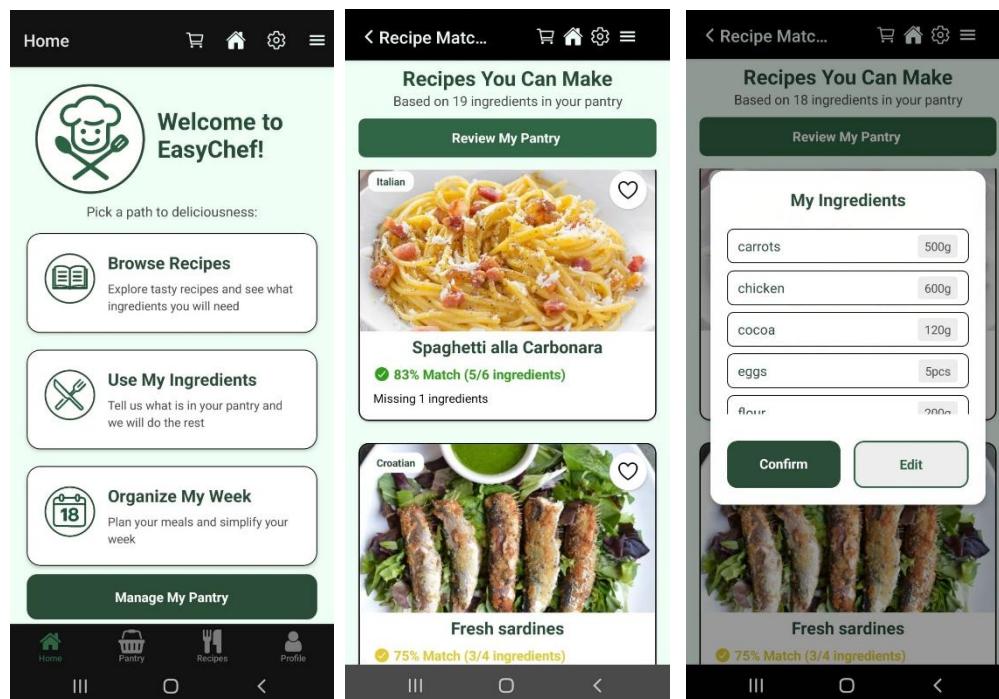


Figură 19 – Exemplu de căutare și adăugare a cantității disponibile în cămară



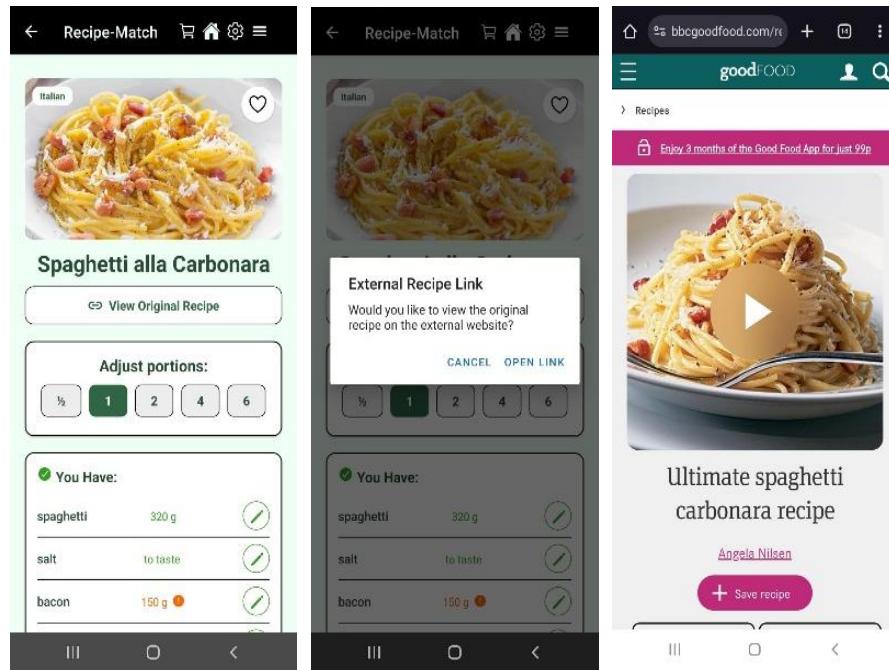
Figură 20 – Exemplu de actualizare a cantității în cămară

Apoi, utilizatorul poate naviga prin paginile principale, accesibile direct din bara de navigare inferioară: Acasă, Cămară, Rețete. Totuși, ca pas următor al manualului de utilizare, am ales să prezint funcționalitatea centrală a aplicației, sugestiile de rețete generate pe baza ingredientelor din cămară. Pe ecranul Acasă, sunt afișate 3 scenarii principale, printre care și cel de a găti, folosind ingredientele proprii. În pagina dedicată a acestui scenariu, se generează lista personalizată de rețete compatibile, filtrate automat atât după conținutul real al cămării, cât și după preferințele alimentare configurate anterior. Pe această pagină, există și butonul revizuire al cămării, care deschide o listă a ingredientelor deja marcate ca fiind în secțiunea de cămară, alături de cantitatea fiecărui. Am introdus această opțiune pentru a oferi o privire de ansamblu rapidă asupra stocului curent, permitându-i utilizatorului să verifice ingredientele, fără a părăsi fluxul principal de utilizare.



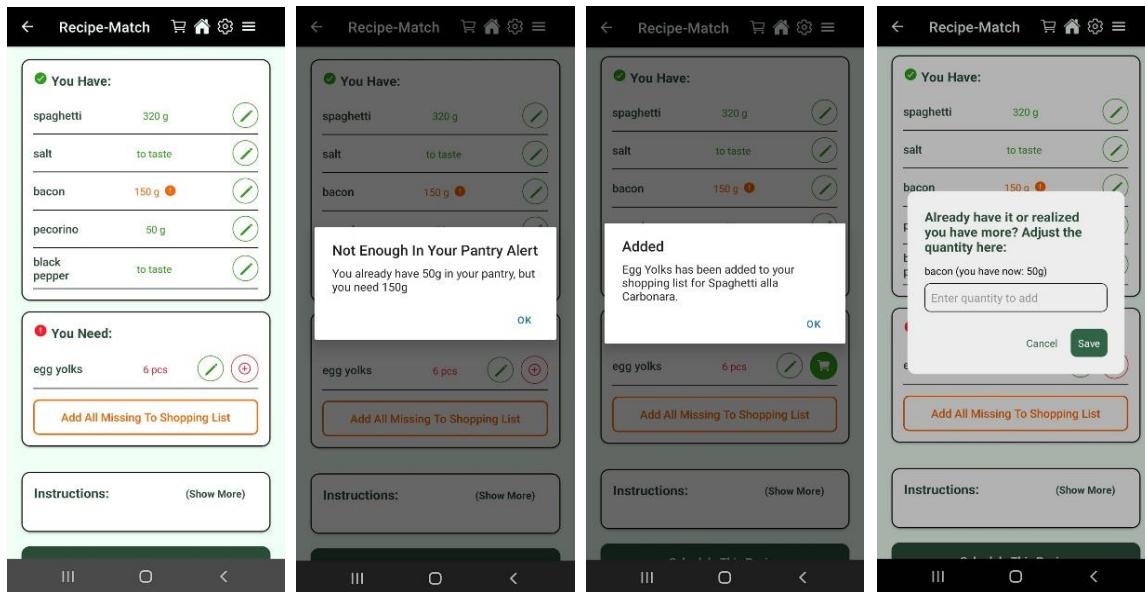
Figură 21 – Exemplu de accesare scenariul de folosire a ingredientelor proprii

Atunci când utilizatorul selectează o rețetă din lista de sugestii, se deschide o pagină dedicată detaliilor complete ale preparatului ales. În partea superioară a ecranului este afișată imaginea ilustrativă a rețetei, urmată de un link către sursa originală, dacă aceasta există. Sub aceste elemente, există posibilitatea de a ajusta numărul de porții, ceea ce determină recalcularea automată a cantităților pentru fiecare ingredient din listă.



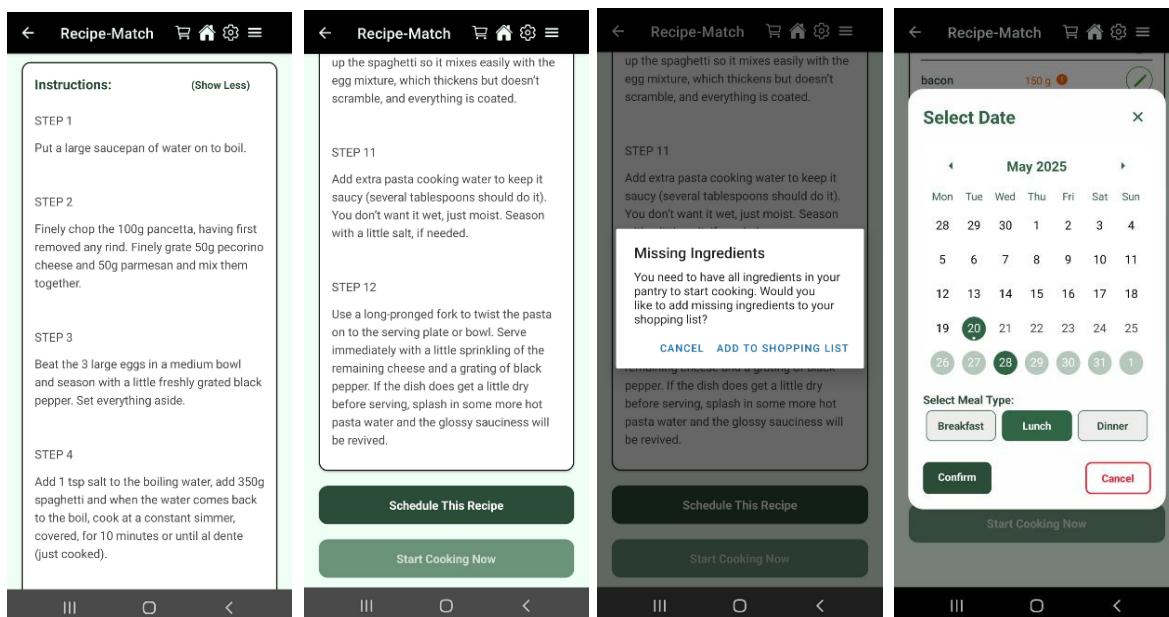
Figură 22 – Exemplu de pagină cu detalii și sursa externă a unei rețete

Lista de ingrediente este structurată vizual pentru a evidenția compatibilitatea cu conținutul actual al cămarii. Ingredientele ce se regăsesc în cămară în cantitatea suficientă sunt marcate cu verde. Cele existente, dar în cantitate insuficientă, sunt evidențiate cu portocaliu și dispun de un icoană de tip informațional. Prin apăsarea acesteia, se deschide un mesaj care informează utilizatorul cu exactitate ce cantitate are nevoie și ce cantitate are disponibilă. În plus, aceste ingrediente pot fi ajustate rapid printr-un buton de editare, reprezentat printr-un simbol cu creion. Ingredientele care lipsesc complet din cămară sunt marcate cu roșu și includ un buton ce permite adăugarea rapidă a acestora în lista de cumpărături.



Figură 23 – Exemplu de funcționalități pagina cu detalii a unei rețete

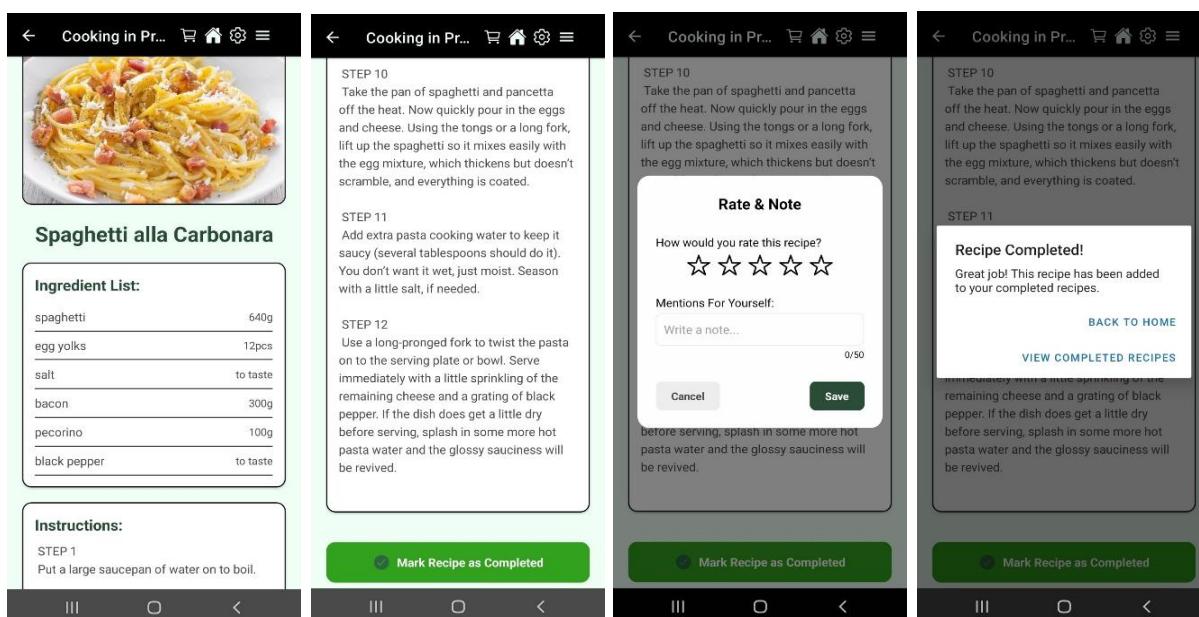
Sub secțiunea ingredientelor se află zona instrucțiunilor de preparare. Aceasta este restrânsă implicit, dar poate fi extinsă la nevoie. La finalul paginii, se regăsesc două acțiuni importante: „Planifică rețeta” și „Gătește acum”. Butonul de gătit este activ doar în momentul în care toate ingredientele sunt disponibile în cantitatea necesară. În caz contrar, utilizatorul primește un mesaj care semnalează lipsurile din inventar, încurajând completarea acestora înainte de începerea procesului de gătit.



Figură 24 – Exemplu de pagină a unei rețete ce poate fi doar planificată

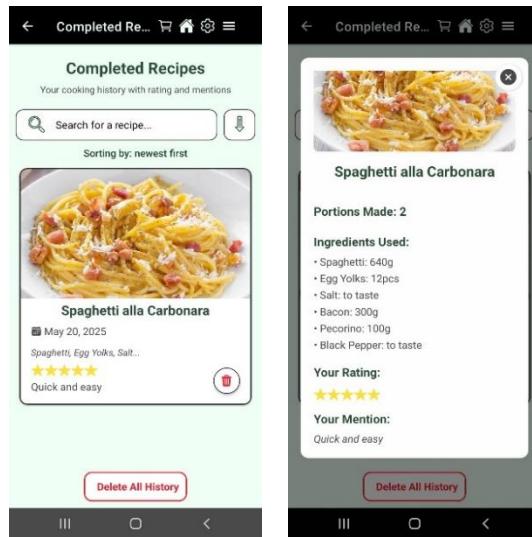
Butonul de planificare a rețetei deschide un calendar interactiv în care ziua curentă este evidențiată, iar utilizatorul poate selecta orice altă zi din viitor pentru a programa prepararea rețetei. Selectarea unei date din trecut nu este permisă. După alegerea zilei, urmează specificarea momentului zilei, mic dejun, prânz sau cină, pentru care utilizatorul dorește să planifice respectiva rețetă.

În momentul în care toate ingredientele necesare sunt disponibile în cantitățile corespunzătoare, butonul „Gătește acum” devine activ. La apăsarea acestuia, utilizatorul este redirecționat către un ecran dedicat, pentru procesul de gătit în desfășurare, unde îi sunt afișate ingredientele adaptate în funcție de numărul de porții selectat, precum și instrucțiunile de preparare. După ce consideră că a finalizat rețeta, utilizatorul o poate marca drept finalizată. Acțiunea declanșează o fereastră unde poate acorda un scor și eventuale mențiuni. Aceste câmpuri sunt optionale, iar dacă sunt lăsate necompletate, scorul va fi înregistrat cu valoarea implicită de 0 din 5. La final, utilizatorul poate alege să se întoarcă la ecranul principal sau să acceseze pagina dedicată rețetelor finalizate.



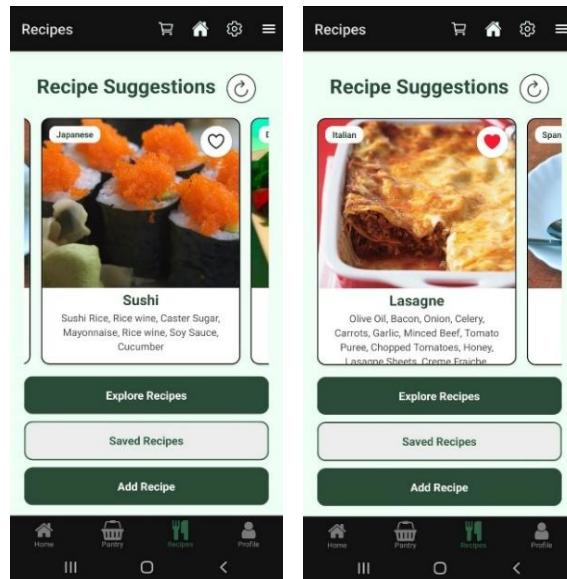
Figură 25 – Exemplu de marcarea a unei rețete ca fiind finalizată

În secțiunea rețetelor finalize, utilizatorul are posibilitatea de a sorta lista în funcție de scorul acordat, în ordine alfabetică sau cronologică, în funcție de data preparării. De asemenea, este disponibilă o bară de căutare, posibilitatea de ștergere a rețetelor din această secțiune.



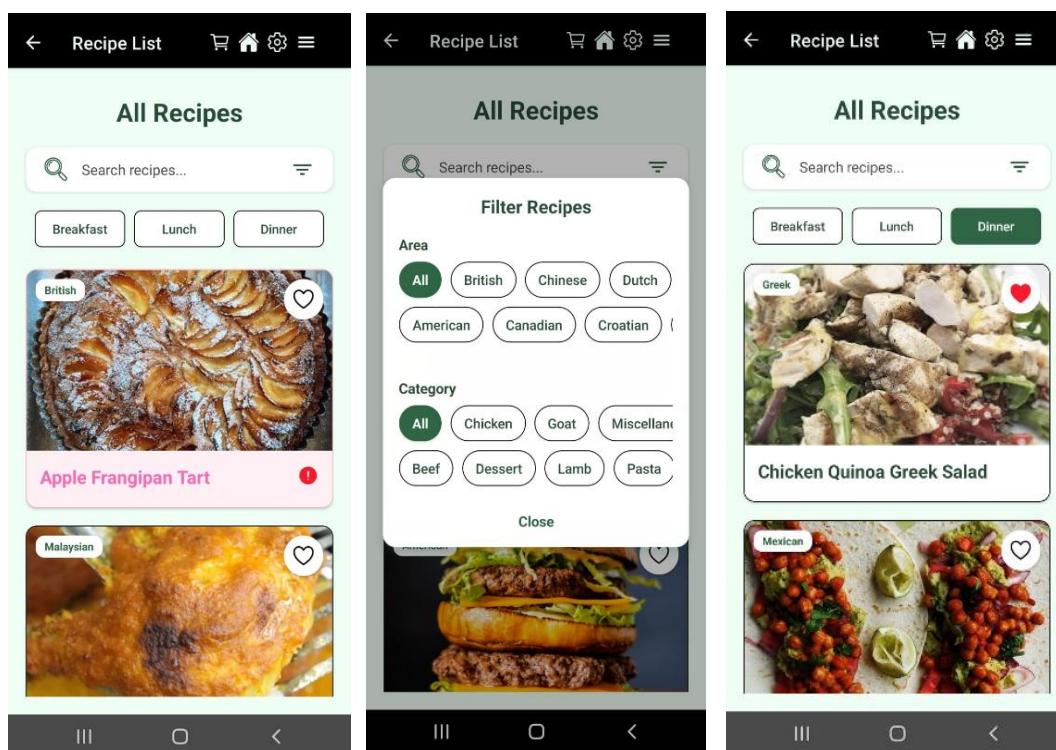
Figură 26 – Exemplu de vizualizare a unei rețete finalize

Înapoi pe pagina de acasă, un alt scenariu este cel pentru răsfoire a rețetelor, care direcționează utilizatorul pe fila principală de rețete, ce poate fi accesată și din bara de navigare inferioară. Aici, interfața afișează inițial câteva sugestii culinare. Utilizatorul are posibilitatea de a reîmprospăta selecția printr-un buton, disponibil permanent în partea superioară a secțiunii. Sub aceste sugestii, sunt plasate trei acțiuni principale: explorarea completă a tuturor rețetelor disponibile în catalog, accesarea rețetelor favorite și adăugarea unei rețete noi.



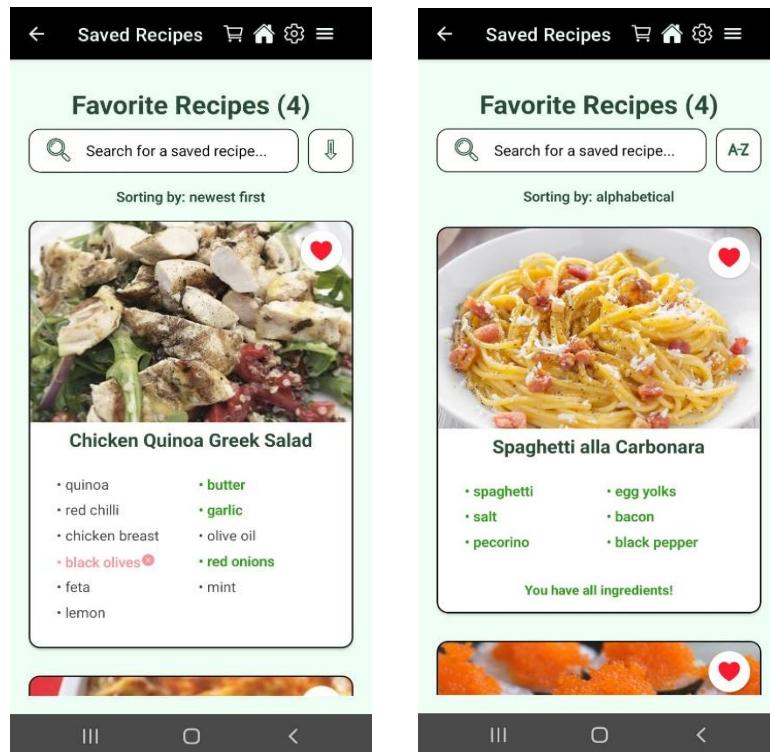
Figură 27 – Exemple de sugestii de rețete culinare

Prin accesarea opțiunii de explorare a rețetelor, utilizatorul ajunge în catalogul complet disponibil în aplicație. Această pagină conține o bară de căutare pentru identificarea rapidă a rețetelor după titlu, precum și o serie de filtre utile, printre care: momentul zilei (mic dejun, prânz, cină), zona de origine a preparatului și categoria culinară (precum deserturi, carne, vegetarian). Fiecare rețetă din catalog poate fi adăugată la lista de favorite prin apăsarea butonului în formă de inimă. Totodată, rețetele care conțin ingrediente interzise sau nu respectă preferințele alimentare ale utilizatorului sunt marcate vizual, pentru a permite o decizie informată înainte de accesare.



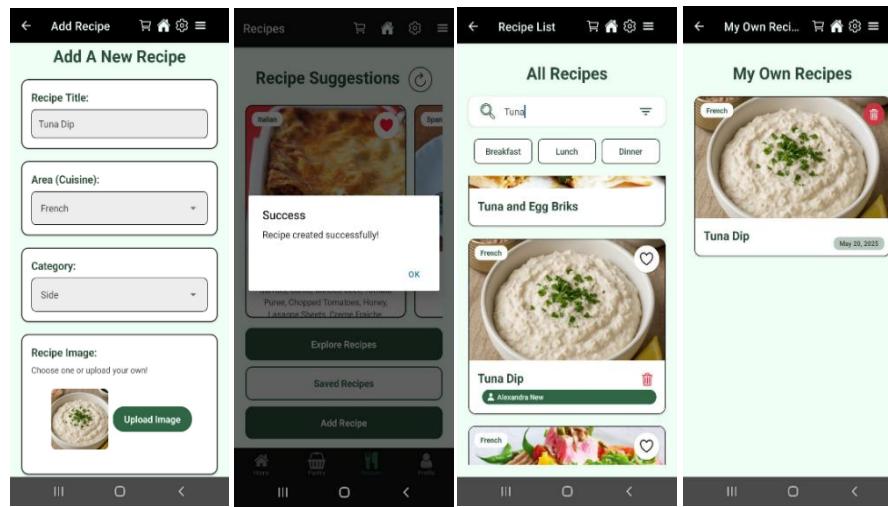
Figură 28 – Exemplu de răsfoire al listei de rețete, cu filtre

Pagina dedicată rețetelor favorite permite vizualizarea rețetelor salvate anterior. Ca altele, pagina dispune de bară de căutare după titlu și opțiuni de sortare după data adăugării, sau în ordine alfabetică. Fiecare card afișează o listă de ingrediente necesare, fără cantități: cele deja existente în cămară sunt scrise cu verde, cele lipsă - cu negru, iar cele care intră în conflict cu preferințele sau restricțiile alimentare ale utilizatorului - în nuanțe de roz sau roșu.



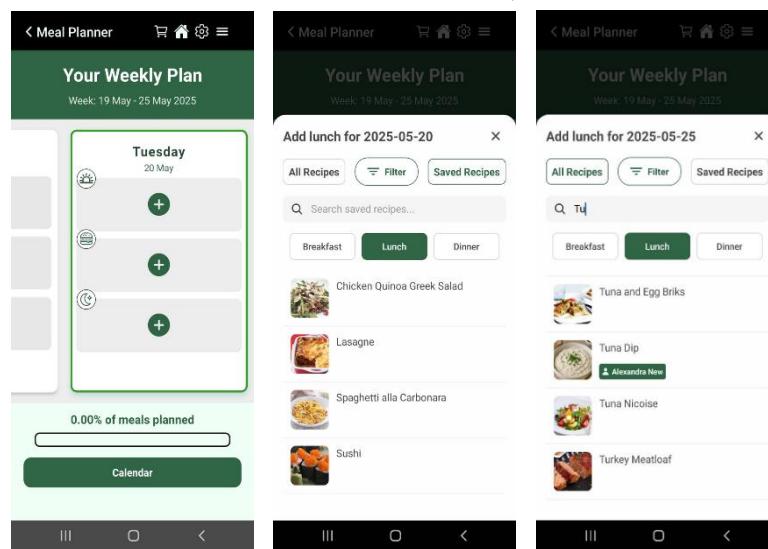
Figură 29 – Exemple din secțiunea de rețete salvate

Adăugarea unei rețete noi presupune completarea unui formular. Utilizatorul trebuie să introducă titlul rețetei, să selecteze zona de origine și categoria culinară din opțiunile disponibile, să aleagă o imagine reprezentativă fie din galeria proprie, fie dintre cele disponibile, apoi să adauge lista de ingrediente împreună cu cantitățile aferente și să completeze instrucțiunile de preparare. După ce toate câmpurile obligatorii sunt completate, aplicația afișează un mesaj de confirmare a succesului, iar rețeta devine imediat disponibilă în catalogul general de rețete, fiind vizibilă și celorlalți utilizatori și în pagina dedicată rețetelor proprii ale utilizatorului autor. Aceasta are și dreptul de a șterge rețetele proprii. În întreaga interfață, rețetele adăugate de utilizatori au autorul menționat, acest aspect făcând diferența între cele provenite din API-ul extern și cele din comunitatea de utilizatori.



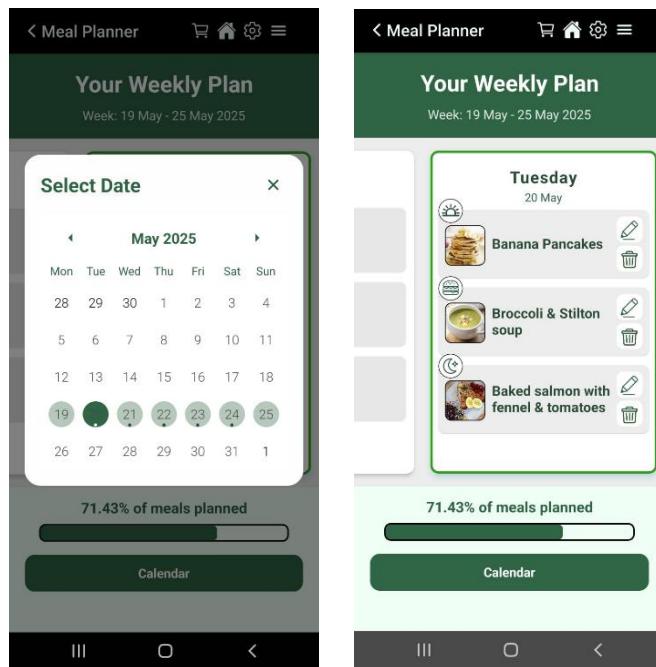
Figură 30 – Exemplu de adăugare a unei rețete proprii

Scenariul de organizare a săptămânii, accesibil tot din pagina de Acasă, permite planificarea meselor pe zile și momente ale zilei într-un mod structurat. Această pagină afișează zilele săptămânii în curs, cu focalizarea automată pe ziua curentă, initial, sau selectată din calendar, ulterior. Aceasta este evidențiată distinct față de celelalte și printr-un contur vizual. Pentru orice masă din prezent sau din viitor, utilizatorul poate apăsa butonul de adăugare corespunzător, care deschide o fereastră de selecție a rețetei, fără a părăsi pagina de planificare. Fereastra permite căutarea rețetelor din catalog, cu posibilitatea de a filtra în funcție de momentul zilei, zona de origine sau categoria rețetei. De asemenea, utilizatorul poate alege din lista de rețete favorite deja salvate.



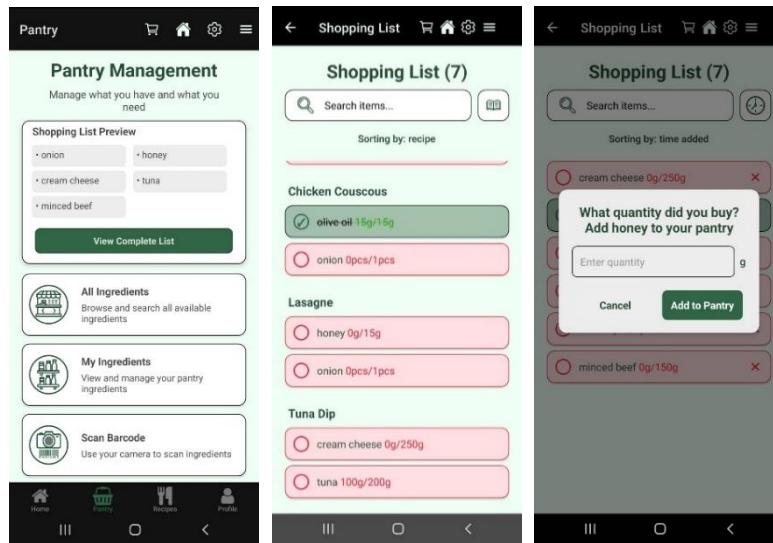
Figură 31 – Exemplu de planificare a unei rețete

Pagina de planificare include o bară de progres care evidențiază procentul de completare al meniului săptămânal. Pentru fiecare masă planificată, există opțiunea de editare sau ștergere, în cazul în care utilizatorul dorește să modifice alegerile făcute. În plus, un buton de tip calendar oferă acces fie la planificarea meselor din alte săptămâni viitoare, fie la vizualizarea retrospectivă a meselor planificate anterior, fără posibilitatea de a selecta zile din trecut pentru planificări sau modificări. Este evidențiată automat ziua curentă, precum și ziua selectată de utilizator, alături de întreaga săptămână din care face parte. Pentru fiecare zi în care este deja planificată cel puțin o masă, calendarul afișează un marcator vizual specific, sub forma unui punct, astfel încât utilizatorul poate observa rapid ce zile au mese planificate.



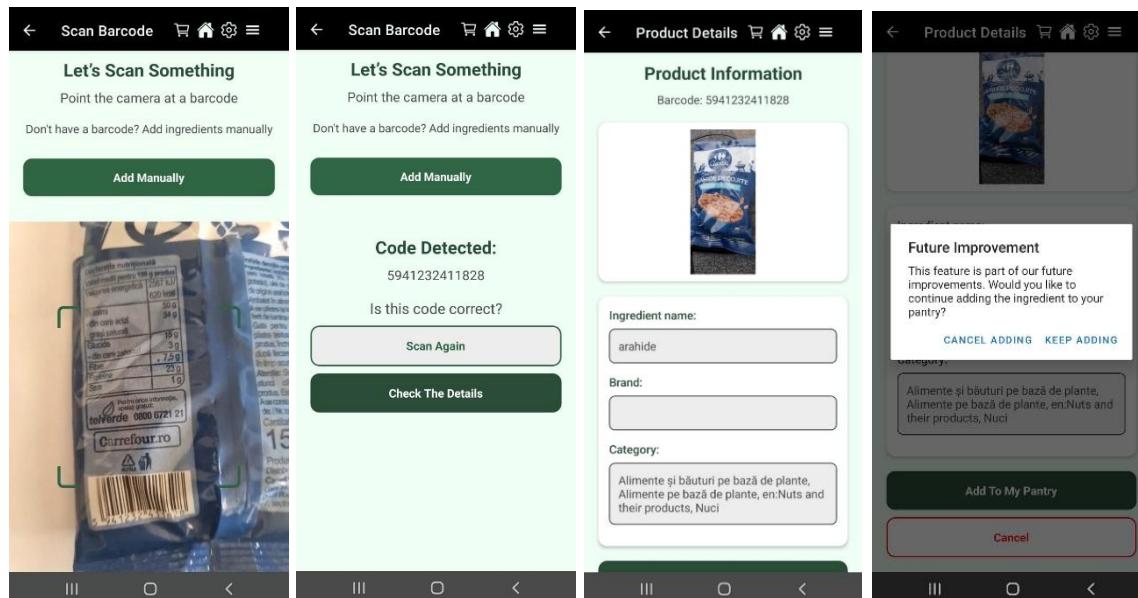
Figură 32 – Exemplu de calendar și rețete alocate pe parcursul unei zile

Pagina cămării oferă o previzualizare a listei de cumpărături și acces rapid către catalogul complet de ingrediente, precum și lista personală a utilizatorului. De aici se poate activa și funcția de scanare a codurilor de bare ale produselor. Lista de cumpărături are și un ecran propriu, accesibil prin butonul conceput pentru vizualizarea completă, unde ingredientele sunt afișate cu detalii precum cantitate și rețeta asociată, la sotarea aferentă. Utilizatorul poate marca produsele achiziționate și cantitatea acestora, printr-o fereastră, fiind un suport în gestionarea cumpărăturilor în timp real. Ingredientele care se marchează ca fiind deja achiziționate cu întreaga cantitate necesară sunt automat bifate, iar la următoarea interacțiune cu acestea ele se șterg din lista de cumpărături.



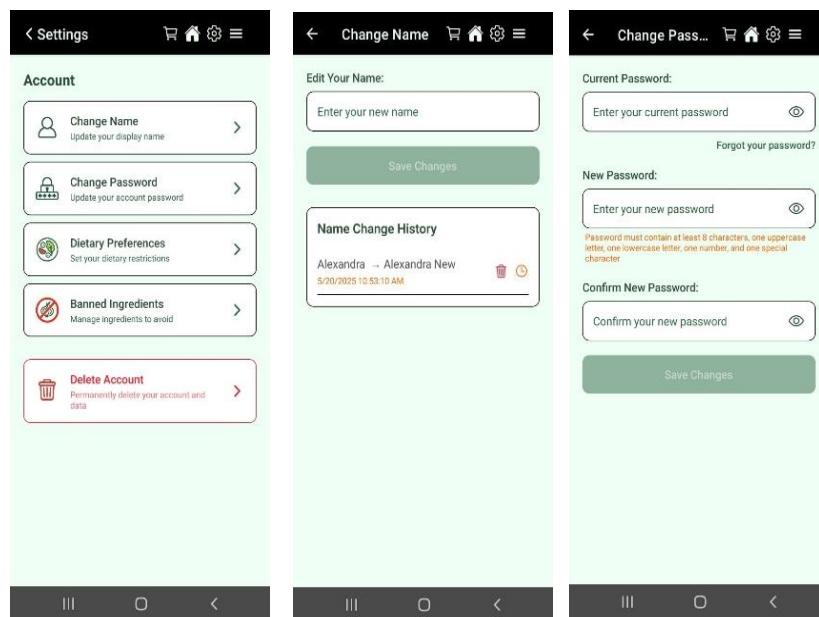
Figură 33 - Ecranul Cămară și gestionarea listei de cumpărături

Opțiunea de scanare a codurilor de bare utilizează camera telefonului pentru a identifica automat produsele alimentare. După scanare, aplicația interoghează API-ul Open Food Facts și returnează informații precum numele produsului, brandul (dacă este disponibil) și categoriile alimentare din care face parte. Rolul acestei opțiuni de scanare este de a oferi baza unei direcții viitoare de dezvoltare. La apăsarea butonului pentru adăugare a produsului în lista de ingrediente personală, utilizatorul primește un mesaj despre acest aspect.



Figură 34 - Procesul de scanare și identificare a produselor prin cod de bare

Pagina de setări este accesibilă din bara superioară, alături de meniul rapid, lista de cumpărături și ecranul principal. Această secțiune centralizează opțiunile de configurare ale contului, inclusiv ștergerea completă a contului de utilizator, schimbarea numelui și a parolei, precum și gestionarea preferințelor alimentare. Funcționalitatea de schimbare a numelui oferă un câmp de introducere cu actualizare în timp real și afișează un istoric al denumirilor anterioare, indicând versiunea veche și data modificării. Pentru modificarea parolei, aplicația solicită atât parola curentă, cât și noua parolă, împreună cu validarea regulilor de securitate.



Figură 35 – Ecranele de setări, modificare nume, modificare parolă

7. CONCLUZII ȘI DIRECȚII DE DEZVOLTARE VIITOARE

7.1. CONCLUZII

Lucrarea de față a avut ca obiectiv dezvoltarea și documentarea aplicației mobile, o soluție modernă destinată optimizării procesului decizional în gătitul de zi cu zi și în planificarea meselor. Pe parcursul etapelor de analiză, proiectare, implementare și testare, s-a demonstrat că integrarea tehnologiilor actuale, precum React Native și Firebase, poate conduce la obținerea unei aplicații funcționale, scalabile și adaptate cerințelor utilizatorilor contemporani.

Aplicația EasyChef reușește să ofere o experiență culinară asistată digital, în care utilizatorul beneficiază de recomandări personalizate pe baza ingredientelor disponibile în cămară și a preferințelor alimentare proprii. Prin funcționalități precum gestionarea inventarului de ingrediente, planificarea meselor pe zile, generarea automată de sugestii de rețete, marcarea rețetelor complete și salvarea celor favorite, aplicația contribuie la reducerea timpului alocat luării deciziilor alimentare și susține un comportament alimentar mai organizat și mai sănătos.

Din punct de vedere tehnic, dezvoltarea aplicației a pus accent pe o arhitectură modulară, cu separarea clară a logicii de business de cea de interfață, ceea ce facilitează întreținerea, testarea și extinderea ulterioară a sistemului. Utilizarea React Native a permis dezvoltarea multiplatformă (Android și iOS) cu un singur cod sursă, reducând semnificativ efortul de implementare și mențenanță. Firebase a fost ales ca backend datorită integrării rapide a serviciilor de autentificare, stocare a datelor (Firestore), gestionare a imaginilor (alături de Cloudinary) și sincronizare în timp real, asigurând astfel o experiență fluidă și coerentă pentru utilizator. Pe lângă avantajele funcționale, Firebase asigură o scalabilitate dinamică a infrastructurii, permitând gestionarea unui număr ridicat de utilizatori simultan fără necesitatea administrării manuale a serverelor. Mecanismul de sincronizare în timp real prin onSnapshot contribuie la reactivitatea interfeței, iar arhitectura cloud reduce latența în actualizarea datelor distribuite.

Un aspect important al proiectului a fost proiectarea modelului de date, astfel încât să permită atât flexibilitate, cât și performanță. Am optat pentru o structură bazată pe subcolecții Firestore (precum dateMealPlans, completedRecipes, favorites), care a permis acces rapid la datele relevante pentru fiecare utilizator și a facilitat extinderea cu noi funcționalități.

De asemenea, am implementat mecanisme de filtrare și sortare a rețetelor direct pe baza ingredientelor și preferințelor, folosind algoritmi eficienți pentru potrivirea rețetelor și generarea sugestiilor.

Pe parcursul dezvoltării, am identificat și rezolvat diverse provocări, precum gestionarea corectă a imaginilor, sincronizarea preferințelor alimentare în timp real, eliminarea redundanțelor din modelul de date și optimizarea fluxului de utilizare pentru a răspunde atât nevoilor utilizatorilor avansați, cât și celor cu experiență tehnologică limitată. Pentru gestionarea imaginilor, am optat pentru integrarea cu serviciul Cloudinary, ce permite compresie automată, scalare dinamică și livrare rapidă prin CDN. Link-urile generate sunt salvate în Firestore, iar imaginile sunt încărcate asincron în interfață. Testele efectuate au demonstrat stabilitatea și fiabilitatea aplicației în condiții diverse de utilizare.

În concluzie, aplicația răspunde unei nevoi reale din viața cotidiană – aceea de a lua decizii alimentare rapide, informate și adaptate contextului personal. Aplicația propusă poate reprezenta un punct de plecare solid pentru dezvoltări ulterioare, fiind construită pe baze tehnologice moderne și pe principii de design orientate către utilizator. Printre direcțiile posibile de extindere se numără integrarea cu surse externe de rețete, adăugarea de funcționalități sociale (partajarea planurilor de masă sau a rețetelor), implementarea de notificări inteligente sau personalizarea avansată a sugestiilor pe baza istoricului de utilizare. Astfel, EasyChef demonstrează potențialul tehnologiei de a transforma și simplifica activitățile cotidiene, oferind un instrument util, intuitiv și adaptabil pentru gestionarea alimentației moderne.

7.2. LIMITĂRI ACTUALE

Deși aplicația prezentată reușește să ofere o experiență coerentă și funcțională în forma sa actuală, există o serie de limitări care influențează atât performanța tehnică, cât și gradul de satisfacție al utilizatorului. Identificarea acestor aspecte reprezintă o etapă importantă în planificarea unor versiuni viitoare, mai robuste și mai adaptate cerințelor reale.

Una dintre limitările semnificative ale versiunii actuale este dependența de o conexiune activă la internet. Aplicația rulează, în prezent, prin platforma Expo Go, ceea ce presupune că întregul cod al aplicației este încărcat din rețea, iar fără acces la internet aplicația nu poate fi lansată deloc. Pe lângă această dependență structurală impusă de mediul de rulare, funcționalitățile principale ale aplicației (autentificarea, sincronizarea datelor prin Firebase Firestore, încărcarea imaginilor din Cloudinary sau generarea

sugestiilor culinare prin API-ul TheMealDB) necesită conexiune permanentă. În absența acesteia, utilizatorii nu pot accesa rețetele, nu pot salva modificări sau planifica mese, ceea ce reduce considerabil utilitatea aplicației. Deși anumite date sunt stocate local temporar prin mecanisme precum AsyncStorage, aplicația nu oferă în prezent un mod offline complet funcțional, nici o versiune compilată care să permită rularea fără rețea.

O altă limitare derivă din modul de funcționare al algoritmului de sugestii de rețete, care se bazează exclusiv pe potrivirea ingredientelor din cămară cu cele necesare rețetelor. Deși este eficient din punct de vedere tehnic, acest sistem nu ia în considerare factori complementari precum timpul de preparare, dificultatea rețetei, preferințele sezoniere, bugetul sau istoricul utilizatorului. Astfel, există riscul ca sugestiile să nu fie întotdeauna relevante pentru contextul imediat al utilizatorului, chiar dacă sunt corecte din punct de vedere al ingredientelor disponibile.

Din perspectiva surselor externe, o limitare suplimentară este legată de calitatea datelor oferite de TheMealDB. În anumite cazuri, rețetele furnizate de acest API conțin câmpuri incomplete sau inconsistente, în special la nivelul cantităților sau al unității de măsură. Aceste inconveniențe afectează negativ experiența de utilizare și logica de potrivire implementată. Sunt necesare tratamente de corecție sau validare, însă nici acestea nu acoperă toate cazurile, în forma actuală a proiectului. În plus, unele rețete includ linkuri externe către platforma sursă, însă câteva dintre acestea sunt inactive sau nu mai disponibile, ceea ce reduce utilitatea acestor redirecționări.

Aplicația permite setarea unor preferințe alimentare și excluderea unor ingrediente, însă, în implementarea din prezent, acestea nu sunt aplicate cu rigurozitate în toate fluxurile. Rețetele care nu respectă preferințele sunt doar marcate vizual, nu eliminate complet din interfață. Această alegere a fost justificată de dimensiunea redusă a catalogului intern de rețete, deoarece o filtrare agresivă ar fi condus la golirea totală a listei în cazul utilizatorilor cu multe restricții. Totuși, această abordare poate genera confuzie, întrucât utilizatorii se așteaptă ca setările să fie aplicate automat, iar rezultatele ce nu corespund, să fie eliminate complet. Odată cu extinderea bazei de date, se poate adopta un mecanism de excludere automată a rețetelor incompatibile.

Pe partea de interfață multimedia, fiecare rețetă include în prezent doar o imagine, fără suport video sau pași ilustrativi. Deși această alegere contribuie la menținerea dimensiunii reduse a aplicației și la performanțe bune, ea limitează componenta vizuală, care este esențială pentru utilizatorii care caută acest tip de ghidare pas cu pas. Linkurile externe incluse în rețetele API-ului trimit uneori către materiale video, dar acestea nu sunt integrate în mod nativ în aplicație.

Absența notificărilor push este o altă limitare notabilă. Utilizatorul nu primește memento-uri automate despre mesele planificate sau actualizările din lista de cumpărături. Astfel, este necesară o verificare manuală constantă pentru a nu rata acțiuni

importante. Această funcționalitate ar putea crește nivelul de automatizare și gradul de implicare a utilizatorului. De asemenea, interfața actuală oferă posibilități limitate de personalizare. Nu există opțiuni pentru schimbarea temei, rearanjarea secțiunilor principale sau adaptarea design-ului la preferințele individuale. În plus, experiența utilizatorului este în prezent disponibilă doar în limba engleză, ceea ce poate constitui o barieră pentru vorbitorii de alte limbi.

În ceea ce privește interacțiunea cu rețetele proprii, imaginile adăugate sunt vizibile doar în aplicație, fără opțiuni de partajare externă sau export. Această limitare poate restrânge potențialul aplicației ca instrument de partajare sau colaborare în grupuri (familie, prieteni).

În final, posibilitățile de căutare sunt limitate. Sistemul actual nu permite filtrări avansate, precum după timp de gătire, dificultate sau popularitate. O structură de căutare mai detaliată ar îmbunătăți semnificativ eficiența navigării în catalogul de rețete și gradul de satisfacție al utilizatorilor.

7.3. FUNCȚIONALITĂȚI POSIBILE ÎN VERSIUNI VIITOARE

Versiunea actuală oferă o bază solidă de funcționalități utile pentru gestionarea ingredientelor, planificarea meselor și generarea de sugestii culinare personalizate. Cu toate acestea, în contextul unei posibile extinderi a aplicației și al diversificării nevoilor utilizatorilor, se conturează o serie de direcții strategice care pot contribui semnificativ la îmbunătățirea funcțională, tehnică și experiențială a produsului.

Una dintre prioritățile identificate este integrarea unui modul de analiză nutrițională, care să permită afișarea valorilor calorice și a distribuției macronutrienților (proteine, carbohidrați, grăsimi) pentru fiecare rețetă. Această funcționalitate ar spori utilitatea aplicației în rândul utilizatorilor preocupați de dietă sau de obiective alimentare specifice și ar transforma acest proiect într-un instrument valoros pentru monitorizarea echilibrului nutrițional. Acest modul ar putea fi extins cu grafice de evoluție, praguri personalizabile și sugestii alimentare în funcție de obiectivele declarate.

Un alt element propus este implementarea unui scanner complex pentru coduri de bare, destinat completării automate a cămării. Utilizatorul ar putea adăuga produse direct prin scanarea etichetei, preluând automat denumirea, cantitatea, unitatea de măsură și chiar data de expirare. Pentru aceasta, aplicația s-ar putea conecta la baze de date alimentare externe, precum OpenFoodFacts, care oferă acces gratuit la informații detaliate despre produse. Această integrare ar eficientiza fluxul de actualizare a inventarului și ar elimina introducerea manuală a multor produse standardizate.

Pe lângă scanare, o extensie logică a modului de gestionare a cămării ar fi introducerea unei opțiuni prin care utilizatorii să poată marca ingrediente ca fiind „permanent disponibile”. Produsele de bază, precum sare, piper, ulei sau apă, nu ar mai trebui reintroduse constant în sistem, ci ar fi considerate implicit în calculul sugestiilor de rețete. În aceeași linie de accesibilitate, adăugarea unui sistem de comenzi vocale ar permite gestionarea camerei prin voce, inclusiv adăugarea, modificarea sau ștergerea ingredientelor. Acest lucru ar fi deosebit de util în timpul gătitului sau pentru utilizatori care doresc o interacțiune hands-free.

Pentru o experiență de utilizare mai coerentă și un angajament mai mare din partea utilizatorului, se poate pune accent pe implementarea notificărilor push. Acestea ar reaminti utilizatorului despre mesele planificate, expirarea anumitor ingrediente sau elemente încă nebifate din lista de cumpărături. Sistemul ar putea include și notificări personalizate pentru rețete noi, sugestii zilnice sau mesaje motivaționale legate de alimentație.

Un alt vector important de dezvoltare este creșterea gradului de personalizare a aplicației. Utilizatorii ar putea beneficia de posibilitatea de a modifica tema, fonturile, culorile sau structura meniurilor principale, în funcție de preferințele individuale. De asemenea, filtrarea rețetelor ar putea fi îmbunătățită prin adăugarea unor opțiuni precum bugetul, timpul de preparare, nivelul de dificultate sau preferințele sezoniere. Extinderea funcționalităților poate include și dezvoltarea unor componente sociale, precum posibilitatea de partajare a rețetelor proprii sau a planurilor de mese cu prietenii și familia. Aceasta ar transforma EasyChef într-un spațiu colaborativ în care utilizatorii își pot împărtăși preferințele culinare, pot comenta sau evalua rețete și pot contribui activ la comunitatea culinară digitală.

În plus, integrarea cu alte API-uri culinare sau platforme de rețete ar fi relevantă pentru a diversifica baza de date și a crește relevanța sugestiilor. Acest pas ar permite extinderea conținutului oferit și adaptarea aplicației la cerințele unei game mai largi de utilizatori. Pentru o mai bună accesibilitate, aplicația ar putea fi tradusă în mai multe limbi și adaptată pentru alte platforme, precum web sau smartwatch, oferind astfel acces rapid la rețete și liste de cumpărături din orice context. În plus, un modul de căutare avansată, cu filtre multiple (ingredient, categorie, timp, dificultate) ar îmbunătăți semnificativ capacitatea utilizatorilor de a găsi rețete potrivite în funcție de nevoi precise.

Toate aceste direcții de dezvoltare reflectă dorința de a transforma acest proiect dintr-o aplicație de gestionare a meselor într-o platformă completă de asistență culinară personalizată, interactivă și inteligentă.

8. BIBLIOGRAFIE

- [1] A. R. G. B. C. S. M. D. S. S. C. R. E. G. M. Alessio Abeltino, „Digital applications for diet monitoring, planning, and precision nutrition for citizens and professionals: a state of the art,” *Nutrition Reviews*, vol. 83, nr. 2, pp. e574-e601, 2025.
- [2] M. A. P. Samir Huseynov, „Food decision-making under time pressure,” *Food Quality and Preference*, vol. 88, p. 104072, 2021.
- [3] J. B. S. P. G. L. N. J. I. M. L. S. A. G. F. J. A. P.-C. R. Katherine Marie Appleton și H. H. Manfred Ronge, „A mobile phone app for the provision of personalized food-based information in an eating-out situation: development and initial evaluation,” *JMIR formative research*, vol. 3, nr. 4, p. e12966, 2019.
- [4] G. Mastorakis, I. Kopanakis, J. Makridis, C. Chroni, K. Synani, K. Lasaridi și T. Manios, „Managing household food waste with the FoodSaveShare mobile application,” 27 Martie 2024. [Interactiv]. Available: <https://www.mdpi.com/2071-1050/16/7/2800>. [Accesat 2025].
- [5] T. Deb, „Diet and Nutrition Apps Statistics 2025 By Tracking, Health and Wellness,” 13 Ianuarie 2025. [Interactiv]. Available: <https://media.market.us/diet-and-nutrition-apps-statistics/>. [Accesat 2025].
- [6] L. Braithwaite, „Decision Fatigue,” 2021. [Interactiv]. Available: <https://thedecisionlab.com/biases/decision-fatigue>. [Accesat 2025].
- [7] M. Ž. N. B. Katarina Milojković, „Agile Multi-user Android Application Development With Firebase: Authentication, Authorization, and Profile Management,” în *Sinteza 2024 - International Scientific Conference on Information Technology and Data Related Research*, Belgrad, Serbia, 2024.
- [8] L. A. S. K. R. H. H. Kevin Lano, „Synthesis of mobile applications using AgileUML,” 26 Aprilie 2021. [Interactiv]. Available: <https://dl.acm.org/doi/abs/10.1145/3452383.3452409>.
- [9] A. M. E. B. a. S. P. Hatice Koç, „UML diagrams in software engineering research: A systematic literature review,” *Synthesis of mobile applications using AgileUML*, vol. 180, p. 111021, 2021.

- [10] V. S. P. S. Chanda Chouhan, „Test case generation based on activity diagram for mobile application,” *International Journal of Computer Applications*, vol. 57, nr. 23, pp. 1-6, Noiembrie 2012.
- [11] X. Zhang, „The significance of sequence diagrams in system design: a comprehensive overview,” 7 Martie 2024. [Interactiv]. Available: <https://zenuml.com/blog/2024/03/07/2024/sequence-diagrams-importance-system-design/>. [Accesat 2025].
- [12] C. N. F. T. M. D. F.-L. M. G.-R. J. D. A. Muhammad Moiz Riaz, „Navigation patterns and design strategies to minimize mobile usability issues,” 9 Octombrie 2023. [Interactiv]. Available: <https://www.researchsquare.com/article/rs-3399474/v1>. [Accesat 2025].
- [13] UX World, „Bottom tab bar navigation design best practices,” 17 Ianuarie 2025. [Interactiv]. Available: <https://uxdworld.com/bottom-tab-bar-navigation-design-best-practices/>. [Accesat 2025].
- [14] M. L. Emilia Henriksson, „Navigation systems' impact on usability in mobile applicatons: a study on mobile newspaper applications,” 12 Iunie 2021. [Interactiv]. Available: <https://www.diva-portal.org/smash/record.jsf?dswid=7264&pid=diva2%3A1569908>. [Accesat 2025].
- [15] S. Singh, „Impact of color on marketing,” *Management decision*, vol. 44, nr. 6, pp. 783-789, 2006.
- [16] Y. X. A. B. H. R. T. T. K. J. D. N. R. D. A. O. Luis A. Leiva, „Understanding Visual Saliency in Mobile User Interfaces,” 5 Octombrie 2020. [Interactiv]. Available: <https://dl.acm.org/doi/abs/10.1145/3379503.3403557>. [Accesat 2025].
- [17] N. C. Ramachandrapappa, „A comparative analysis of native vs react native mobile app development,” *International Journal of Computer Trends and Technology*, vol. 72, pp. 57-62, 16 Septembrie 2024.
- [18] J. Bidny, „React Native Pros and Cons,” 8 Ianuarie 2025. [Interactiv]. Available: <https://www.netguru.com/blog/react-native-pros-and-cons>. [Accesat 2025].
- [19] D. Stevenson, „Why you should use TypeScript for writing Cloud Functions,” 18 Ianuarie 2018. [Interactiv]. Available: <https://firebase.blog/posts/2018/01/why-you-should-use-typescript-for/>. [Accesat 2025].
- [20] Firebase, „Firebase developer documentation,” 2025. [Interactiv]. Available: <https://firebase.google.com/docs>. [Accesat 2025].

- [21] A. B. P. T. L. I. S. Audy Fitri Arian, „Implementation Of Firebase In The Development Of Android-Based Queue Reservation And,” *JSiL / Jurnal Sistem Informasi*, vol. 12, nr. 1, pp. 63-71, 2025.
- [22] Firebase, „Firebase Security Rules,” 2025. [Interactiv]. Available: <https://firebase.google.com/docs/rules>. [Accesat 2025].
- [23] Cloudinary, „Cloudinary Image & Video API Platform - Docs Home Page | Documentation,” 20 Mai 2025. [Interactiv]. Available: <https://cloudinary.com/documentation>. [Accesat 2025].
- [24] J. Erickson, „What Is JSON?,” 4 Aprilie 2024. [Interactiv]. Available: <https://www.oracle.com/ro/database/what-is-json/>. [Accesat 2025].
- [25] R. T. Fielding, „Architectural styles and the design of network-based software architectures,” University of California,, Irvine, 2000.
- [26] GeeksforGeeks, „REST API Introduction,” Iunie 2025. [Interactiv]. Available: <https://www.geeksforgeeks.org/node-js/rest-api-introduction/>. [Accesat 2025].
- [27] Interaction Design Foundation, „What is Visual Design?,” 2025. [Interactiv]. Available: <https://www.interaction-design.org/literature/topics/visual-design>.
- [28] H. T. P. C. F. & T. S. R. Ting, „Consumer behaviour and disposition decisions: The why and how of smartphone disposition,” *Journal of Retailing and Consumer Services*, vol. 51, pp. 212-220, 25 Iunie 2019.
- [29] T. D. B. D. J. N. Bardia Doosti, „A computational method for evaluating UI patterns,” 11 Iulie 2018. [Interactiv]. Available: <https://arxiv.org/abs/1807.04191>. [Accesat 2025].
- [30] S. G. A. J. S. P. R. S. Prakhar Gupta, „Saliency prediction for mobile user interfaces,” 7 Mai 2018. [Interactiv]. Available: <https://ieeexplore.ieee.org/abstract/document/8354275>. [Accesat 2025].
- [31] J. D. H. A. M. B. K. Jesenka Pibernik, „The effects of the floating action button on quality of experience,” 6 Iulie 2019. [Interactiv]. Available: <https://www.mdpi.com/1999-5903/11/7/148>. [Accesat 2025].
- [32] S. M. J. S. J. W. S. G. K. P. A. D. P. Prachi R. Saraf, „A review on Firebase (backend as a service) for mobile application development,” *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, vol. 10, p. 1, Ianuarie 2022.

**DECLARAȚIE DE AUTENTICITATE A
LUCRĂRII DE FINALIZARE A STUDIILOR**

Subsemnatul NOVACESCU ALEXANDRA-DORIANA, legitimat cu CI, seria TZ, nr. 925408, CNP 6021004350034, autorul lucrării APLICAȚIE MOBILĂ PENTRU GESTIONAREA REȚETELOR CULINARE, elaborată în vederea susținerii examenului de finalizare a studiilor de Licență, organizat de către Facultatea Automatică și Calculatoare, din cadrul Universității Politehnica Timișoara, sesiunea Iunie a anului universitar 2024-2025, coordonator ş.i. dr. ing. Răzvan-Dorel Cioargă, luând în considerare art. 34 din Regulamentul privind organizarea și desfășurarea examenelor de licență/diplomă și disertație, aprobat prin HS nr. 109/14.05.2020 și cunoșcând faptul că în cazul constatării ulterioare a unor declarații false, voi suporta sancțiunea administrativă prevăzută de art. 146 din Legea nr. 1/2011 – legea educației naționale și anume anularea diplomei de studii, declar pe proprie răspundere, că:

- această lucrare este rezultatul propriei activități intelectuale;
- lucrarea nu conține texte, date sau elemente de grafică din alte lucrări sau din alte surse fără ca acestea să nu fie citate, inclusiv situația în care sursa o reprezintă o altă lucrare/alte lucrări ale subsemnatului;
- sursele bibliografice au fost folosite cu respectarea legislației române și a convențiilor internaționale privind drepturile de autor;
- această lucrare nu a mai fost prezentată în fața unei alte comisii de examen/prezentată public/publicată de licență/diplomă/disertație;
- În elaborarea lucrării am utilizat instrumente specifice inteligenței artificiale (IA) și anume ChatGPT, menționat în cadrul lucrării.

Declar că sunt de acord ca lucrarea să fie verificată prin orice modalitate legală pentru confirmarea originalității, consimțind inclusiv la introducerea conținutului său într-o bază de date în acest scop.

Timișoara,

Data

27.06.2025

Semnătura

