

UNIVERSITATEA "ALEXANDRU-IOAN CUZA" DIN IASI

FACULTATEA DE INFORMATICA



LUCRARE DE LICENTA

Aplicatia Forking Food

propusă de

Adrian-Petru Bleoju

Sesiunea: iulie, 2025

Coordonator științific

Conf. Dr. Gavriluț Dragos

UNIVERSITATEA "ALEXANDRU-IOAN CUZA" DIN IASI

FACULTATEA DE INFORMATICA

Aplicația Forking Food

Adrian-Petru Bleoju

Sesiunea: iulie, 2025

Coordonator științific

Conf. Dr. Gavriluț Dragos

Avizat,

Îndrumător lucrare de licență,

Conf. Dr. Gavriluț Dragos.

Data:

Semnătura:

Declaratie privind originalitatea conținutului lucrării de licență

Subsemnatul **Bleoju Adrian-Petru** domiciliat în **România, jud. Harghita, mun. Miercurea Ciuc, strada Câmpul Mic, nr. 14**, născut la data de **21 iunie 1998**, identificat prin CNP **1980621194032**, absolvent al Facultății de informatică, **Facultatea de informatică** specializarea **informatică**, promoția 2025, declar pe propria răspundere cunoscând consecințele falsului în declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr. 1/2011 art. 143 al. 4 și 5 referitoare la plagiul, că lucrarea de licență cu titlul **Aplicația Forking Food** elaborată sub îndrumarea domnului **Conf. Dr. Gavriluț Dragos**, pe care urmează să o susțin în fața comisiei este originală, îmi aparține și îmi asum conținutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licență să fie verificată prin orice modalitate legală pentru confirmarea originalității, consimțind inclusiv la introducerea conținutului ei într-o bază de date în acest scop.

Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări științifice în vederea facilitării falsificării de către cumpărător a calității de autor al unei lucrări de licență, de diplomă sau de disertație și în acest sens, declar pe proprie răspundere că lucrarea de față nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Data:

Semnătura:

Declarație de consimțământ

Prin prezenta declar că sunt de acord ca lucrarea de licență cu titlul **Aplicația Forking Food**, codul sursă al programelor și celealte conținuturi (grafice, multimedia, date de test, etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de informatică.

De asemenea, sunt de acord ca Facultatea de informatică de la Universitatea "Alexandru-Ioan Cuza" din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Absolvent **Adrian-Petru Bleoju**

Data:

Semnătura:

Cuprins

Introducere	2
1 Descrierea problemei	9
2 Abordări anterioare	10
3 Arhitectura aplicației	12
3.1 Structura proiectului Flutter	13
3.2 Funcționalități avansate și integrarea cu Firebase	15
3.2.1 Autentificare și gestionarea utilizatorilor	15
3.2.2 Stocarea și distribuirea rețetelor	16
3.2.3 Recomandări personalizate pe baza swipe-urilor	17
3.2.4 Păstrarea istoricului și feedback pentru algoritm	18
3.3 Modelul de date și structura colecțiilor Firebase	18
4 Implementarea aplicației	20
4.1 Organizarea codului și bune practici	20
4.2 Fluxul de autentificare	21
4.2.1 Ecrane și navigare	21
4.2.2 Integrarea cu Firebase Authentication	22
4.3 Structura principală de navigare	24
4.3.1 TabBar și BottomNavigationBar	25
4.3.2 Gestionarea stării aplicației	26
4.4 Mecanismul de swipe și recomandare	27
4.4.1 Logica de gestionare a swipe-urilor	27
4.4.2 Întreținerea pool-ului de rețete	28
4.4.3 Recomandări personalizate	29
4.5 Adăugarea unei rețete	30

4.5.1	Formularul complet: structură și validare	31
4.5.2	Salvarea datelor în Firestore și Storage	33
4.6	Alte funcționalități implementate	34
4.6.1	Profilul utilizatorului	35
4.6.2	Secțiunea Today's Top	35
4.6.3	Gestionarea rețetelor proprii și salvate	36
4.7	Capturi de ecran și interfața aplicatiei	36
5	Concluzii și direcții viitoare	38
5.1	Reflectie personală	38
5.2	Direcții de dezvoltare viitoare	38
5.3	Concluzie generală	39
Bibliografie		40

Introducere

Motivația alegerii temei

Ideea dezvoltării aplicației *Forking Food* s-a conturat într-o perioadă în care exploram posibilitatea realizării unei aplicații de tip *Progressive Web App* (PWA), folosind Next.js. Eram într-un proces activ de căutare a unei idei potrivite pentru un proiect personal, cu intenția de a experimenta o abordare practică și relevantă pentru platformele web moderne. Nu aveam încă o direcție fixă, dar încercam să găsesc un concept care să îmbine interacțiunea și un conținut atractiv.

Într-un astfel de moment de explorare, în timp ce găteam prima mea porție de paste carbonara, am urmat o rețetă sugerată de un model de limbaj natural. Savurând rezultatul și căutând, în paralel, variante ale profesioniștilor pe YouTube, mi-a venit ideea unei aplicații care să faciliteze descoperirea rețetelor culinare într-un mod inedit, prin interacțiuni de tip *swipe*, asemenea famoaselor aplicații de dating.

Conceptul initial a purtat numele de *MashOrTrash* și presupunea alegerea între rețete pe baza unei acțiuni binare: da sau nu. Ulterior, ideea a evoluat într-un format mai clar, căpătând identitatea actuală *Forking Food* — o aplicație mobilă dedicată descoperirii, împărtășirii și salvării de rețete culinare.

Am renunțat la ideea de PWA în favoarea unei aplicații mobile native, deoarece conceptul acesta se preta mai bine unei experiențe native, pe ecran complet, cu gesturi și feedback vizual. Dintre opțiunile disponibile pentru dezvoltarea mobilă, am ales *Dart - Flutter*, limbaj și framework deja explorat în cadrul activității de laborator. Această alegere mi-a permis o implementare rapidă, cu suport cross-platform și cu un control mai precis asupra interfeței.

Gradul de noutate al temei

Aplicația *Forking Food* propune o abordare inedită în domeniul aplicațiilor culinare prin introducerea unui model de interacțiune inspirat din platformele de tip *dating*. În locul clasicelor liste sau grile de rețete, utilizatorii interacționează cu conținutul prin gesturi de tip *swipe*, salvând sau trecând peste rețete într-un mod intuitiv, simplu și plăcut. Această metodă transformă procesul de explorare într-o experiență specială, care se diferențiază categoric de restul aplicațiilor dedicate explorării de rețete.

Din observațiile personale și din cercetările efectuate, am constatat că această abordare nu este întâlnită frecvent în aplicațiile din domeniul culinar. Deși mecanismul de *swipe* este extrem de popular în aplicațiile de *dating*, adaptarea lui într-un context culinar este originală și poate avea un impact pozitiv asupra modului în care utilizatorii o utilizează.

Pe lângă valoarea funcțională, aplicația are la bază și o asociere simbolică interesantă: dacă în platformele de *dating* utilizatorii caută potențiale relații, în *Forking Food* aceștia aleg rețete care le stârnesc interesul sau pofta. Într-un mod metaoric, se mai spune că „dragostea trece prin stomac” — iar eu cred că este adevărat.

Obiectivele lucrării

Principalul obiectiv al lucrării de față a fost dezvoltarea unei aplicații mobile funcționale, care să poată constitui o bază solidă pentru un proiect de licență aplicat. Alegerea temei a fost ghidată în primul rând de dorința de a crea ceva original, care să aducă o abordare nouă în rândul aplicațiilor culinare. Astfel, componenta de inovație conceptuală a fost un criteriu de bază în conturarea direcției lucrării.

Dincolo de atingerea unui obiectiv concret, lucrarea a avut și o dimensiune educațională: punerea în practică a tehnologiilor studiate în cadrul facultății, în special utilizarea framework-ului Flutter și integrarea acestuia cu servicii oferite de Firebase (precum autentificare, stocare în cloud și baze de date).

Un obiectiv secundar, dar cu valoare reală, este posibilitatea ca aplicația să fie dezvoltată în continuare și publicată, la un moment dat, devenind un produs accesibil publicului larg.

Publicul țintă vizat de aplicație este larg: oricine deține un smartphone și dorește să exploreze rețete culinare într-un mod rapid și intuitiv se încadrează. Aplicația poate

fi utilizată atât ocazional, pentru explorare ocazională, cât și frecvent, ca o formă de explorare zilnică prin câteva interacțiuni rapide. În mod special în momentele în care utilizatorii nu știu ce să gătească, *Forking Food* poate deveni o sursă utilă de inspirație.

Metodologia utilizată

Realizarea aplicației *Forking Food* a urmat un proces ghidat de obiectivul de a obține un produs funcțional și complex, dar și de nevoia personală de învățare și consolidare a cunoștințelor acumulate. Proiectul a început cu dezvoltarea interfeței pentru metodele de autentificare (ecranele welcome, login, register și forgot password), urmând apoi integrarea serviciilor de autentificare oferite de Firebase.

Ulterior, a fost creat ecranul principal — *Home Screen* — care a evoluat într-o structură mai complexă, organizată pe trei tab-uri: Home, Profile și Add Recipe. Interfața aplicației a fost gândită inițial astfel încât să respecte structura datelor așteptate din Firebase, permitând o integrare ulterioară simplă la partea de backend. După finalizarea prototipului funcțional (MVP), atenția s-a mutat mai mult pe optimizarea manipulării datelor și pe îmbunătățirea interfeței utilizatorului.

Un pas important în evoluția complexității aplicației a fost adăugarea unui nou ecran — *Discover* — devenit ulterior tab în cadrul ecranului principal. Acesta conține două secțiuni: *Today's Top* (top 3 cele mai apreciate rețete din ziua curentă) și *Fork You* (o secțiune de rețete recomandate alese utilizând un sistem de scor personalizat al rețetelor, calculat pe baza preferințelor exprimate de-a lungul utilizării aplicației de utilizatori).

Butonul pentru adăugarea rețetelor a fost repozitionat strategic pentru a fi accesibil din toate locurile, sub forma unui buton clasic de tip „plus” în AppBar-ul aplicației dar și a unui card dedicat în secțiunea de profil, la începutul grilei cu rețete proprii *My Recipes*.

Pe partea de autentificare, au fost implementate metodele de tip OAuth de la Google și Facebook. Datorită unor incompatibilități recente în SDK-ul Facebook cu Firebase, această din urmă funcționalitate a fost oprită temporar.

Proiectul a fost gestionat cu ajutorul unui repository Git activ încă de la început și cu o aplicație de tip note, folosită într-un mod asemănător platformelor de planificare precum Trello. De asemenea, pe tot parcursul dezvoltării, m-am sprijinit intensiv de instrumente AI (modele de limbaj natural), care m-au asistat în structurarea codului,

organizarea și implementarea funcționalităților, rafinarea logicii de business și generarea de date de test. Consider că menționarea acestui aspect este relevantă, întrucât reflectă o practică tot mai prezentă în fluxurile moderne de dezvoltare software.

Tehnologiile alese — Flutter și Firebase — au fost selectate în primul rând pentru că au fost studiate în cadrul facultății, în special la materia TPPM, dar și pentru că sunt potrivite extrem de bine cu cerințele aplicației. Flutter a oferit avantajul dezvoltării cross-platform, iar Firebase a permis o integrare rapidă și robustă cu serviciile de autentificare, stocare și baze de date NoSQL.

Aplicația a fost testată local, pe propriul dispozitiv, însă nu a fost distribuită mai departe prin canale oficiale precum TestFlight. A existat, totuși, o formă minimală de testare: o serie de demonstrații informale către o persoană apropiată, care a rezultat în obținerea de feedback util legat de claritatea interfeței și accesibilitatea și predictibilitatea acesteia.

În ceea ce privește design-ul, am urmărit o estetică minimalistă, orientată spre utilitate. Am încercat să reduc complexitatea vizuală și am ținut să pun accentul pe afișarea corectă a conținutului și ușurința în utilizare, evitând elementele care ar putea distrage atenția de la scopul principal al aplicației.

Descrierea generală a aplicației Forking Food

Forking Food este o aplicație mobilă destinată utilizatorilor pasionați de gătit, oferind un mod modern, interactiv și intuitiv de a descoperi, salva și distribui rețete culinare la nivel global. Aplicația funcționează ca un spațiu personal sau chiar și unul de explorare, în care utilizatorii se pot întâlni cu rețete din diverse categorii, își pot posta propriile idei culinare și vor păstra un istoric al preferințelor lor gastronomice.

Structura aplicației este gândită în jurul mai multor ecrane funcționale:

- **Splash Screen** – ecran inițial de încărcare cu efect vizual placut.
- **Welcome Screen** – oferă acces către metodele de autentificare.
- **Login/Register/Reset Password Screens** – fluxul standard pentru autentificări și recuperare de cont.
- **Main Screen**, împărțit în trei tab-uri principale:

- *Discover* – inclusând două secțiuni: *Today's Top* (topul zilei) și *ForkYou* (recomandările personalizate).
- *Home* – ecranul principal unde utilizatorii pot salva (*Fork-In*) sau sări peste (*Fork-Out*) rețete, într-o interfață cu mod de funcționare tip *card-swiper*. Include și o filtrare pentru rețete după durata medie de preparare și/sau criterii dietetice.
- *Profile* – ecranul cu informațiile personale ale utilizatorului (poza, nume, statistici), alături de două secțiuni: *My Recipes* (rețetele proprii) și respectiv *Saved* (rețete salvate prin acțiunea de *fork-in*).
- **Add Recipe Screen** – formular complet pentru adăugarea unei rețete noi, cu suport pentru imagine, ingrediente, instrucțiuni pas cu pas, tag-uri și criterii dietetice.

Funcționalitățile cheie includ: autentificare prin email și OAuth (Google), *swiping* pentru interacțiunea cu rețetele, vizualizarea detaliată a rețetelor, adăugarea și stergerea rețetelor proprii, editarea profilului și explorarea topurilor și a recomandărilor zilnice.

Aplicația folosește un model de date definit în codul sursă și sincronizat cu structura bazei de date din Firestore. Principalele entități sunt:

- **UserData** – include câmpuri precum `id`, `displayName`, `email`, `photoURL`, `createdAt`, `lastUpdated`.
- **Recipe** – include titlu, descriere, imagine, ingrediente, instrucțiuni, tag-uri, timp estimat, informații despre creator, scoruri și criterii dietetice.
- **InstructionStep** – fiecare pas din rețetă are descriere și o imagine (optională).

Datele sunt gestionate prin Firestore Database (pentru text și metadate) și Firebase Storage (pentru imagini). Utilizatorii interacționează cu aplicația folosind gesturi tactile de tip swipe, tap și navigare prin tab-uri, într-un mod natural și intuitiv. Fiecare rețetă poate fi vizualizată în mod full screen, și toate ecranele permit refresh manual prin swipe down.

Formularele de adăugare a rețetelor oferă o funcționalitate complexă: adăugarea și stergerea dinamică a pașilor, căutare și selectare de tag-uri, propunerea și adăugarea

de noi tag-uri și selecție multiplă și pentru criteriile dietetice și o validare completă a rețetei la publicare.

Forking Food este deci o aplicație completă, care îmbină funcționalitatea și simplitatea cu interactivitatea într-un format modern și prietenos.

Structura lucrării

Lucrarea de licență este structurată în șase capitole, fiecare având rolul de a construi treptat contextul, fundamentele teoretice, soluția propusă și concluziile rezultate din dezvoltarea aplicației *Forking Food*.

- **Capitolul 1 — Descrierea problemei** expune contextul actual al aplicațiilor culinare și identifică o nișă relativ neacoperită: lipsa unei experiențe tip *swiping*, care să transforme descoperirea de rețete într-o activitate pe cât de interactivă pe atât de utilă.
- **Capitolul 2 — Abordări anterioare** analizează metodele și platformele actuale prin care utilizatorii accesează rețete culinare — de la motoare de căutare la platforme de streaming video sau retele sociale — și aduce la lumină lipsa unui model de interacțiune bazat pe gesturi tactile și preferințe exprimate în mod direct.
- **Capitolul 3 — Descrierea soluției propuse** oferă o privire detaliată asupra modului în care aplicația Forking Food ajunge să răspundă acestei nevoi. Sunt prezentate arhitectura aplicației, sistemul de autentificare, modelul de date, mecanismul de swipe implementat cu pachetul `flutter_card_swiper`, algoritmul de determinare a scorurilor rețetelor, precum și sistemul de personalizare a preferințelor și a restricțiilor culinare. Acest sistem care calculează și actualizează scorurile pe baza comportamentului utilizatorilor, reprezintă una din cele mai relevante componente ale aplicației.
- **Capitolul 4 — Implementare** detaliază pașii concreți ai dezvoltării, descrie modulele aplicației, principalele clase, integrarea Firebase și mecanismele de manipulare a datelor.
- **Capitolul 5 — Testare și rezultate** prezintă modul în care aplicația a fost validată funcțional, precum și unele dificultăți întâmpinate (cum ar fi gestionarea

instantelor de widget reutilizate în pachetul swiper, optimizarea performanței pentru interogările de date ale userilor și deciziile arhitecturale pentru păstrarea datelor dinamice).

- **Capitolul 6 — Concluziile și direcțiile viitoare** sintetizează lucrurile învățate, cu accent pe importanța feedback-ului vizual, a clarității interacțiunilor, dar și a unui *UX* coherent. Sunt propuse îmbunătățiri precum introducerea unui scurt ghid introductiv vizual, consolidarea accesibilității și rafinarea utilizării scorurilor de recomandare.

Capitolul 1

Descrierea problemei

În prezent, majoritatea utilizatorilor interacționează cu rețetele culinare prin căutări pe internet, prin platforme video sau rețelele sociale. Aceste metode presupun o căutare activă și conștientă a unui anumit tip de rețetă, ceea ce limitează caracterul surprinzător al experienței. În plus, lipsa unei interfețe gamificate face ca procesul să fie mai degrabă serios decât plăcut.

Nu există suficiente aplicații care să ofere un mod rapid, intuitiv și plăcut de a descoperi rețete noi, fără o intenție din partea utilizatorului de căutare predefinită. Așa apare oportunitatea de a crea o aplicație care să transforme tot acest proces într-o experiență interactivă, bazată pe gesturi simple și feedback vizual captivant.

Forking Food răspunde acestei nevoi printr-un model de interacțiune familiar tip *swipe*, adaptat unui domeniu în care acesta este inedit. În acest fel, aplicația facilitează descoperirea spontană și ușoară a conținutului culinar, oferind în același timp posibilitatea de personalizare pe baza unor criterii proprii.

Capitolul 2

Abordări anterioare

Interacțiunea cu rețetele culinare în mediul digital se face, în general, prin două direcții majore: fie prin căutare activă, fie prin descoperire pasivă. Căutarea activă are loc pe motoare de căutare sau pe platforme precum YouTube, unde utilizatorul pornește de la o intenție clară (exemplul propriu: „paste carbonnara”) și se așteaptă să găsească un rezultat specific.

Pe de altă parte, rețelele sociale precum Instagram și TikTok chiar oferă o experiență mai spontană, în care utilizatorii pot descoperi rețete fără să le caute în mod intentionat. Algoritmii acestor platforme pot recomanda conținut culinar divers, de la preparate simple la creații ale unor *chefs* renumiți. Cu toate astea, aceste rețele nu sunt dedicate exclusiv rețetelor culinare și deci conținutul lor este amestecat cu alte forme media, ceea ce limitează consecvența și utilitatea explorării în scopuri culinare.

În ceea ce privește aplicațiile specializate pe rețete, există destule soluții consacrata precum:

- **Tasty** — oferă video-uri scurte, listă de ingrediente, mod de preparare pas cu pas și sugestii personalizate. Este orientată mai degrabă spre ideea de rețete „virale”, cu accent pe prezentare vizuală și ușurință de preparare.
- **Yummly** — folosește filtre avansate pentru a oferi sugestii personalizate (timp, ingrediente, preferințe dietetice), cu un sistem de „like” și salvare a rețetelor.
- **Paprika** — axată pe organizarea personală a rețetelor: permite importul din alte surse, salvare *offline*, planificare de mese și listă de cumpărături.
- **Kitchen Stories** — se remarcă printr-o interfață elegantă și prezentarea de articole editoriale pe lângă rețete. Oferă video-uri, ghiduri și o experiență orientată

spre educație culinară.

Toate aceste aplicații oferă funcționalități utile, în timp ce adoptă un model standard de interacțiune: navigare prin liste, grile sau fluxuri, căutare după cuvinte cheie sau filtrare tradițională. Deși eficiente, aceste modele se bazează pe un comportament obișnuit, ghidat de o nevoie concretă.

Forking Food introduce în joc o diferență esențială: explorarea rețetelor într-un mod jucăuș, bazat pe gesturi rapide de tip *swipe*. Interacțiunea nu este condiționată de o intenție clară de căutare — utilizatorul nu trebuie să știe ce vrea să gătească sau să găsească. Astfel, experiența în aplicație devine una de inspirație, nu doar de eficiență.

Din acest motiv, *Forking Food* nu își propune să concureze direct cu aplicațiile existente, ci dorește să ofere o alternativă diferită, cu o abordare axată pe descoperire pasivă și gamificată. Proiectul nu a fost conceput ca răspuns la o lipsă identificată, ci ca o explorare a unui concept original, născut dintr-o idee personală — aceea de a aduce swipe-ul într-un spațiu nou.

Capitolul 3

Arhitectura aplicației

Aplicația *Forking Food* este structurată modular, făcându-i-se o separare clară între interfața utilizatorului, logica aplicației și serviciile externe (Firebase). Această separare facilitează întreținerea și extinderea aplicației pe termen lung.

Navigarea în aplicație urmează un flux intuitiv, pornind de la ecranul introductiv cu rol estetic de tip *Splash*, urmat de autentificare (acolo unde este necesară), iar pe urmă, ajungând în interfața principală, organizată pe tab-uri. Diagrama urmatoare ilustrează acest flux clasic de navigare între ecranele și tab-urile aplicatiei:

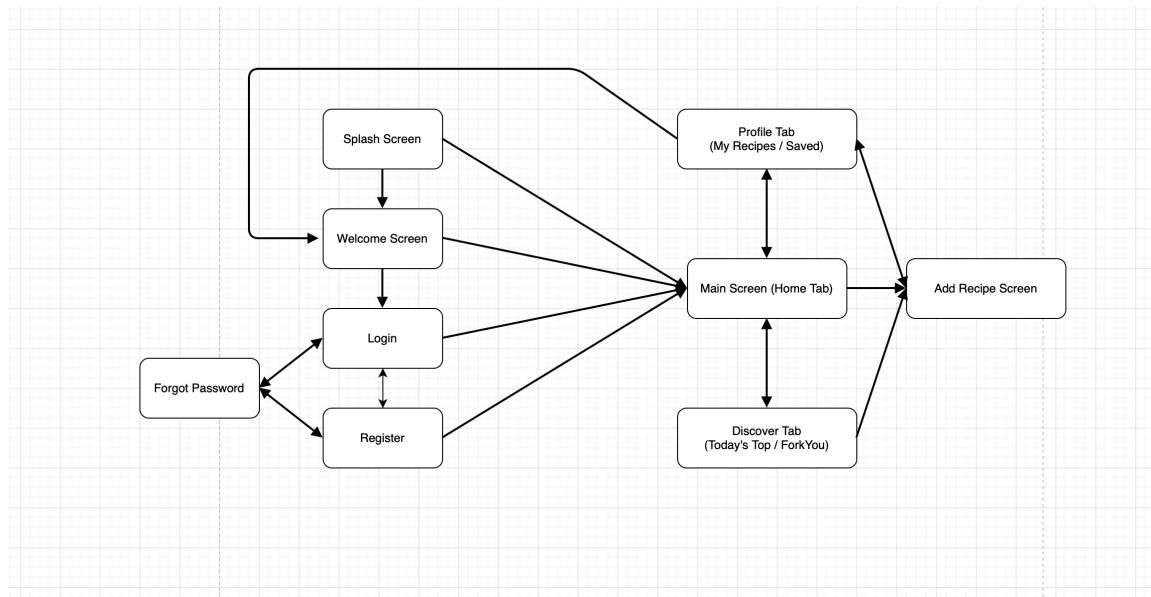


Figura 3.1: Fluxul de navigare principal în aplicația Forking Food

Fluxul de navigare include și cazuri speciale precum autentificarea prin Google OAuth, accesarea directă a ecranului principal în cazul utilizatorilor autențificați, dar și utilizarea ecranului „Forgot Password”.

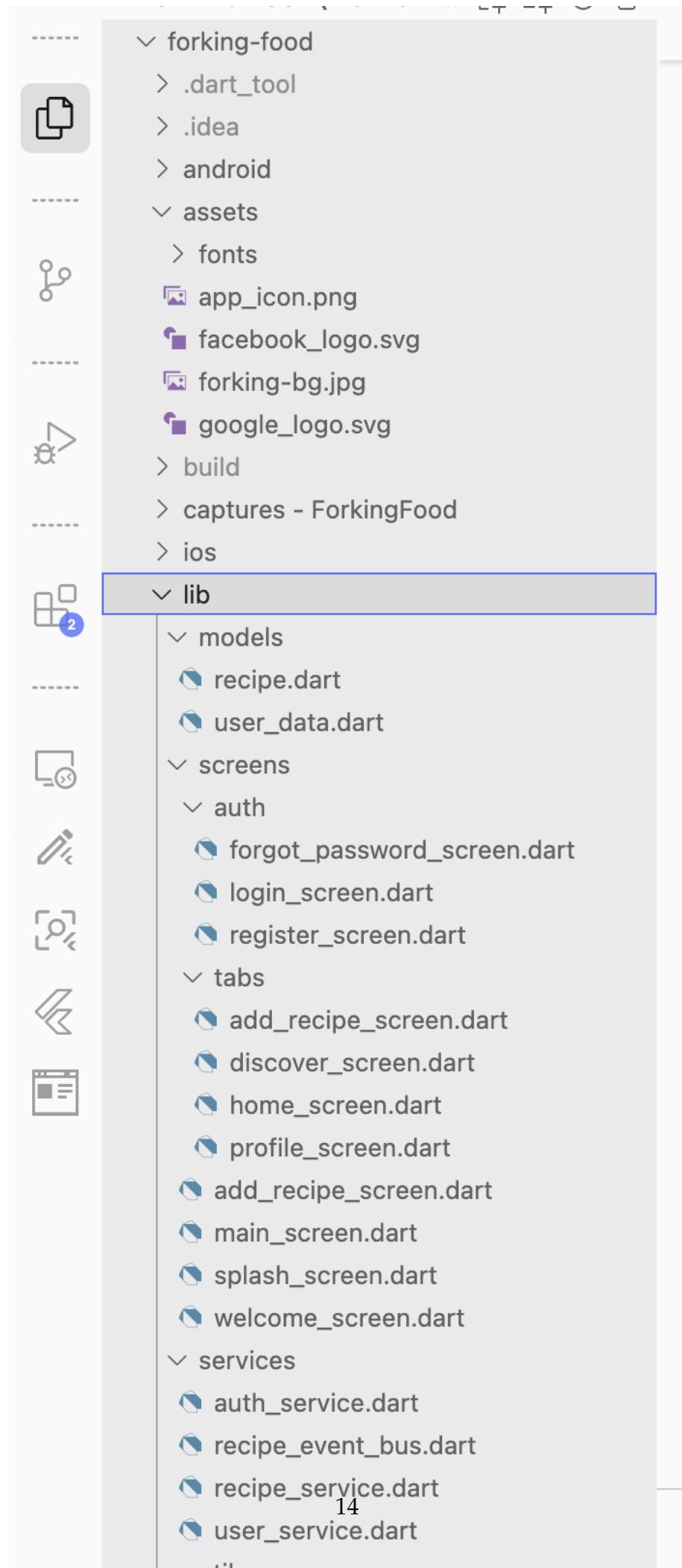
Tab-ul principal, *Home*, este primul ecran afișat după autentificare. De aici utilizatorul poate naviga către secțiunea *Discover* (cu cele două subsecțiuni: reprezentate de tab-urile „Today's Top” și „ForkYou”), către ecranul *Profile* sau către ecranul de adăugare de rețete.

Navigarea se face în general prin intermediul unei bare de navigare cu tab-uri persistente și a unor butoane interne, iar acțiunile de tip *back* sau *pop* sunt controlate pentru a preveni pierderea de date (ex: confirmarea la ieșirea din meniul de creare a unei rețete).

3.1 Structura proiectului Flutter

Proiectul este organizat într-o structură modulară, specifică aplicațiilor Flutter de dimensiuni medii, și cu accent pe separarea responsabilităților. Mai jos sunt prezentate directoarele principale:

- **lib/screens/** – conține toate ecranele aplicației, fiecare în fișierul dedicat. Exemplu: `home_screen.dart`, `discover_screen.dart`, `add_recipe_screen.dart`.
- **lib/widgets/** – componente reutilizabile din interfața grafică, precum cardul pentru rețete `recipe_card.dart`.
- **lib/services/** – gestionează comunicarea cu Firebase și alte servicii externe. Aici se regăsesc `auth_service.dart`, `recipe_service.dart`, etc.
- **lib/models/** – definește clasele model pentru datele din aplicație: `Recipe`, `UserData`, `InstructionStep`, etc.
- **lib/utils/** – culori, dimensiuni, fonturi și alte constante partajate, sau utilitare incluse în fișierul `constants.dart`.
- **lib/main.dart** – punctul de intrare în aplicație, locul unde se configurează Firebase, tema aplicației și navigarea de bază.



Această structurare urmărește principii de separare a responsabilităților și scalabilitate. De exemplu, logica de autentificare este izolată în `auth_service.dart`, iar afișarea listelor de rețete este tratată în `discover_screen.dart`, folosind widget-uri dedicate.

3.2 Funcționalități avansate și integrarea cu Firebase

Aplicația *Forking Food* se bazează pe ecosistemul Firebase pentru a gestiona autentificarea utilizatorilor, stocarea datelor și livrarea conținutului multimedia. Arhitectura este gândită astfel încât responsabilitățile să fie bine separate, iar interacțiunea cu serviciile externe să fie cât mai transparentă definită.

3.2.1 Autentificare și gestionarea utilizatorilor

Sistemul de autentificare este construit pe baza Firebase Authentication și permite conectarea prin email/parolă, dar și autentificare OAuth cu Google. Utilizatorii logați au acces la interfața principală a aplicației și pot efectua acțiunile de adăugare sau salvare de rețete.

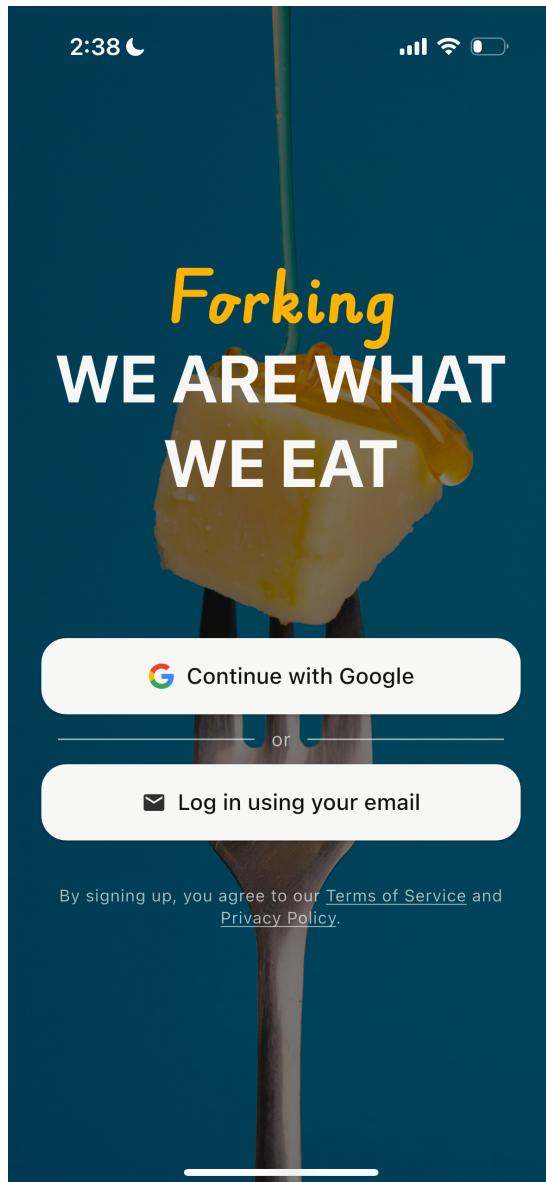


Figura 3.3: Ecranul de autentificare cu opțiuni Google și email/parolă

3.2.2 Stocarea și distribuirea rețetelor

Rețetele sunt stocate în Firestore Database sub forma unor documente ce conțin detalii ca titlul, descriere, ingrediente, mod de preparare, criterii dietetice și identificatorul utilizatorului care a creat rețeta. Imaginele sunt încărcate în Firebase Storage, iar în baza de date este salvat doar URL-ul asociat acestora.

Fiecare pas din modul de preparare pentru o rețetă este reprezentat printr-o instanță de `InstructionStep`, inclusă în lista de pași din obiectul `Recipe`. Pașii pot avea imagini atașate, dar obligatorie este doar prezența unei descrieri tip text.

Figura 3.4: Document din colecția `recipes` cu detalii despre rețetă

3.2.3 Recomandări personalizate pe baza swipe-urilor

Una dintre funcționalitățile cele mai importante ale aplicației este sistemul de recomandare a rețetelor pe baza preferințelor formate în urma acțiunilor de tip swipe efectuate mai multor rețete de către utilizator. La fiecare interacțiune de tip *Fork-In* (like) sau *Fork-Out* (skip), aplicația actualizează date asociate preferințelor utilizatorului — pentru ingrediente, taguri, criterii dietetice și altele.

Aceste scoruri sunt stocate în colecția `user_preferences` și sunt utilizate pentru ordonarea rețetelor în secțiunea *ForkYou*, dar și în alte componente ale aplicației. Algoritmul de scor permite ajustarea sugestiilor în timp real, în funcție de acțiunile recente ale utilizatorului.

Figura 3.5: Document din colecția `user_preferences` cu scoruri generate pe urma swipe-urilor

3.2.4 Păstrarea istoricului și feedback pentru algoritm

Pentru fiecare utilizator se păstrează un istoric minim al rețetelor cu care a interacționat (colecția `swipes`). Acest istoric este utilizat pentru a exclude rețetele deja vizualizate și a evita afișarea conținutului duplicat.

De asemenea, în colecția `daily_likes` este menținut un contor zilnic pentru cele mai populare rețete astăzi, folosit pentru a construi secțiunea *Today's Top*. Această abordare oferă atât o dimensiune personalizată, cât și una socială în modul de afișare a conținutului.

3.3 Modelul de date și structura colecțiilor Firebase

Modelul de date este centrat în jurul a două entități esențiale: `UserData` și `Recipe`. Acestea sunt salvate în colecțiile principale din Firestore și organizate după următoarea structură:

Colecția recipes

Conține toate rețetele publicate în aplicație. Fiecare document are câmpuri precum:

- `id` – ID unic al rețetei
- `title, description` – informații de bază
- `imageUrl` – link către imaginea principală din Firebase Storage
- `ingredients` – listă de siruri de caractere
- `instructions` – listă de pași cu descriere și imagine (optională)
- `tags, dietaryCriteria` – etichete și criterii de natură dietetică
- `creatorId` – ID-ul utilizatorului care a creat rețeta
- `forkInCount, forkOutCount` – contoare pentru interacțiuni
- `totalEstimatedTime` – timp de preparare estimat în secunde

Colecția user_preferences

Stochează date despre preferințelor fiecărui utilizator, recalculate la fiecare swipe:

- dietaryScores, ingredientScores, tagScores – obiecte cu chei dinamice și valori numerice
- userId – ID-ul utilizatorului
- lastUpdated – data ultimei actualizări

Colecția swipes

Reține istoricul interacțiunilor individuale pentru fiecare utilizator:

- userId, recipeId și type – „forkIn” sau „forkOut”
- timestamp – data la care a fost efectuat swipe-ul

Colecția daily_likes

Folosită pentru monitorizarea evenimentelor pe zile. Necesare construirii secțiunii *Today's Top*. Contine ID-ul rețetei și un contor actualizat zilnic.

Capitolul 4

Implementarea aplicației

Capitolul anterior a prezentat arhitectura aplicației *Forking Food*, structura proiectului și interacțiunea cu serviciile Firebase. În continuare, vom trata detaliile implementării concrete a funcțiilor principale: autentificarea utilizatorilor, sistemul de navigare, mecanismul tip swipe și recomandările, gestionarea rețetelor și interfața vizuală.

Pentru fiecare dintre aceste componente, vor fi explicate atât fluxurile funcționale, cât și modul în care au fost structurate în codul sursă. De asemenea, vom evidenția și bunele practici aplicate pentru a menține aplicația robustă, clară și scalabilă.

4.1 Organizarea codului și bune practici

Pentru a menține aplicația *Forking Food* clară, extensibilă și ușor de întreținut, codul a fost organizat într-o structură modulară, împărțită pe funcționalități și responsabilități. Această abordare respectă principii similare modelului MVC (Model – View – Controller).

La nivelul codului, s-a urmărit:

- separarea logicii (în `services/`) de interfață (`widgets/`, `screens/`);
- folosirea componentelor reutilizabile pentru a evita duplicarea codului (`RecipeCard`);
- izolarea accesului la Firebase în clase specialize (ex: `RecipeService`, `AuthService`);
- aplicarea principiului „single responsibility” – fiecare clasă sau fișier având un scop clar și unic;

- gestionarea stării local (prin `setState`) sau asincron (prin `StreamBuilder` și `FutureBuilder`) acolo unde a fost necesar.

De exemplu, un widget responsabil cu afișarea unei rețete nu conține logică de autentificare sau interogări de bază de date, ci doar preia datele și le afișează într-o manieră clară și adaptabilă.

Această organizare ajută la testare, reutilizare și extindere ulterioară, dar și la înțelegerea mai ușoară a desfășurării codului de către alți dezvoltatori.

4.2 Fluxul de autentificare

4.2.1 Ecrane și navigare

Fluxul de autentificare din aplicația *Forking Food* este unul simplu și bine definit, cuprinzând următoarele ecrane principale:

- `welcome_screen.dart` – ecranul principal de întâmpinare cu opțiunea de autentificare Google și optarea pentru varianta email/parolă;
- `login_screen.dart` – formular pentru autentificare prin email;
- `register_screen.dart` – formular de creare cont pe bază de email;
- `forgot_password_screen.dart` – formular pentru resetarea de parolă prin email.

Navigarea între aceste ecrane se face folosind metoda `Navigator.push()` sau prin `Navigator.pushReplacement()`, în funcție de context, pentru a păstra un flux coerent și a preveni întoarcerile nedorite la ecrane anterioare după etapa de autentificare.

Navigarea principală este orchestrată prin clasa `AppEntryPoint`, care decide, la pornirea aplicației, dacă utilizatorul va fi direcționat către `WelcomeScreen` sau direct către `MainScreen`, în funcție de starea autentificării.

Listing 4.1: Clasa `AppEntryPoint` responsabilă de inițializare și redirecționare

```
class AppEntryPoint extends StatelessWidget {
  ...
}
```

```

86 <-- class AppEntryPoint extends StatelessWidget {
87   const AppEntryPoint({super.key});
88
89   Future<void> _initializeApp() async {
90     // Initialize Firebase
91     await Firebase.initializeApp();
92
93   // await Future.delayed(const Duration(milliseconds: 2000));
94 }
95
96   @override
97   Widget build(BuildContext context) {
98     return SplashScreen(
99       initialization: _initializeApp,
100    child: FutureBuilder(
101      future: Future.value(), // Firebase already initialized
102      builder: (context, snapshot) {
103        // Check auth state
104        final authService = AuthService();
105        if (authService.isSignedIn) {
106          return const MainScreen();
107        } else {
108          return const WelcomeScreen();
109        }
110      },
111    ), // FutureBuilder
112  ); // SplashScreen
113 }
114 
```

Figura 4.1: Clasa AppEntryPoint în fișierul main.dart – captură din editor

4.2.2 Integrarea cu Firebase Authentication

Pentru autentificare, aplicația *Forking Food* utilizează serviciul **Firebase Authentication**, facilitând conectarea utilizatorilor prin email/parolă, Google sau Facebook. Toate interacțiunile cu Firebase sunt gestionate prin clasa AuthService, care abstracționează logica de conectare, înregistrare dar și actualizările de profil ale utilizatorilor.

Aplicația pornește cu inițializarea Firebase prin metoda `Firebase.initializeApp()` în cadrul fișierului `main.dart`, urmând verificarea utilizatorului pentru a vedea dacă este sau nu logat. Această verificare se face prin getterul `isSignedIn`, care returnează `true` dacă există un utilizator autentificat.

Clasa AuthService oferă metode precum:

- `signIn(email, password)` – autentificare cu email și parolă;
- `register(email, password, displayName)` – creare cont nou;
- `signInWithGoogle()` – autentificare OAuth cu Google;
- `signInWithFacebook()` – autentificare cu Facebook;
- `signOut()` – deconectare din toate sesiunile.

Pentru utilizatorii autentificați prin Google sau Facebook, aplicația extrage numele și poza de profil, care sunt apoi prelucrate (redimensionate și comprimate) pentru a fi salvate în Firebase Storage, dacă este cazul. Informațiile utilizatorului sunt sincronizate cu o colecție proprie în Firestore, prin intermediul clasei `UserService`.

```

53  ...// Google-Sign-In
54  Future<UserCredential?> signInWithGoogle() async {
55  ...
56  ...
57  ...
58  ...
59  ...
60  ...
61  ...
62  ...
63  ...
64  ...
65  ...
66  ...
67  ...
68  ...
69  ...
70  ...
71  ...
72  ...
73  ...
74  ...
75  ...
76  ...
77  ...
78  ...
79  ...
80  ...
81  ...
82  ...
83  ...
84  ...
85  ...
86  ...
87  ...
88  ...
89  ...

```

Figura 4.2: Metoda `signInWithGoogle()` – captură din `auth_service.dart`

Configurarea metodelor de autentificare a fost realizată din consola Firebase, la secțiunea *Authentication*, tab-ul *Sign-in method*, unde se observă că sunt activate opțiunile Google, Facebook și Email/Password.

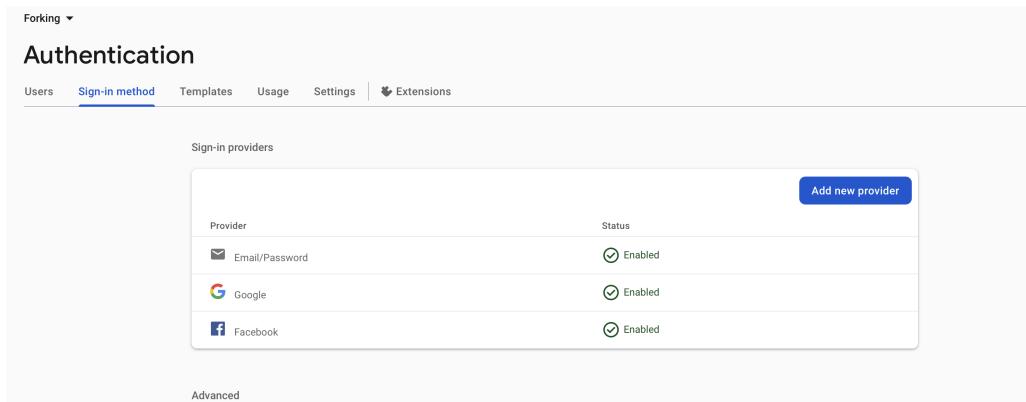


Figura 4.3: Metodele de autentificare activate în consola Firebase

4.3 Structura principală de navigare

După autentificare, utilizatorul este direcționat către interfața principală a aplicației, definită de clasa `MainScreen`. Aceasta este organizată în trei tab-uri accesibile printr-un `BottomNavigationBar` persistent:

- **Discover** – permite explorarea rețetelor disponibile în cele două moduri: *Today's Top* și *ForkYou*;
- **Home** – spațiul dedicat activității specifice tip swipe a aplicației;
- **Profile** – oferă acces și permite editarea datelor de profil, rețetelor proprii și a celor salvate.

Navigarea între tab-uri este gestionată printr-o stare internă (`currentIndex`) în `MainScreen`, care controlează ce ecran este afișat. Acest comportament este implementat cu ajutorul unui `BottomNavigationBar`, cu itemi personalizați (pictograme și etichete) pentru fiecare secțiune.

```
14 class _MainScreenState extends State<MainScreen> {
15   static const _storageKey = 'main_screen_selected_index';
16   int _selectedIndex = 1;
17
18   final List<Widget> _screens = const [
19     DiscoverScreen(),
20     HomeScreen(),
21     ProfileScreen(),
22   ];
23
24   @override
25   void initState() {
26     super.initState();
27     // restore saved index
28     WidgetsBinding.instance.addPostFrameCallback((_) {
29       final storedIndex = PageStorage.of(context)
30         .readState(context, identifier: _storageKey) as int?;
31       if (storedIndex != null) {
32         setState(() => _selectedIndex = storedIndex);
33       }
34     });
35   }
36
37   void _onItemTapped(int index) {
38     FocusScope.of(context).unfocus();
39     PageStorage.of(context).writeState(context, index, identifier: _storageKey);
40     setState(() => _selectedIndex = index);
41   }
42
43   @override
44   Widget build(BuildContext context) {
45     SystemChrome.setSystemUIOverlayStyle(SystemUiOverlayStyle.dark);
46     return Scaffold(
47       body: IndexedStack(
48         index: _selectedIndex,
49         children: _screens,
50       ), // IndexedStack
51       bottomNavigationBar: NavigationBar(
52         selectedIndex: _selectedIndex,
53         onDestinationSelected: _onItemTapped,
54         labelBehavior: NavigationDestinationLabelBehavior.alwaysHide,
55         indicatorColor: Theme.of(context).colorScheme.primary,
56         backgroundColor: Colors.transparent,
57         elevation: 0,
58         destinations: const [
59           NavigationDestination(
60             icon: Icon(Icons.explore_outlined),
61             selectedIcon: Icon(Icons.explore),
62             label: 'Discover',
63           ), // NavigationDestination
64           NavigationDestination(
65             icon: Icon(Icons.home_outlined),
66             selectedIcon: Icon(Icons.home),
67             label: 'Home',
68           ), // NavigationDestination
69           NavigationDestination(
70             icon: Icon(Icons.person_outline),
71             selectedIcon: Icon(Icons.person),
72             label: 'Profile',
73           ), // NavigationDestination
74         ],
75       ), // NavigationBar
76     ); // Scaffold
77   }
78 }
```

Figura 4.4: Structura principală de tab-uri în ecranul `MainScreen`

4.3.1 TabBar și BottomNavigationBar

Aplicația *Forking Food* utilizează o bară de navigare persistentă (BottomNavigationBar) pentru a facilita accesul rapid la cele trei secțiuni principale: **Home**, **Discover** și **Profile**. Această structură contribuie la o experiență de utilizare comodă, intuitivă și familiară, similară altor aplicații moderne de conținut.

Navigarea este implementată printr-un widget central, `MainScreen`, care conține o componentă tip `BottomNavigationBar` și o listă de ecrane asociate fiecărui tab. La schimbarea indexului selectat, se actualizează starea și se afișează ecranul corespunzător.

Fiecare dintre cele trei tab-uri are propriile componente:

- **Home** – afișează fluxul principal cu rețete, filtrarea acestora și butonul de creare rețetă;
- **Discover** – permite explorarea rețetelor în cadrul celor două taburi interne („Today's Top” și „ForkYou”);
- **Profile** – afișează datele utilizatorului și grilele interne cu rețetele create și cele salvate.

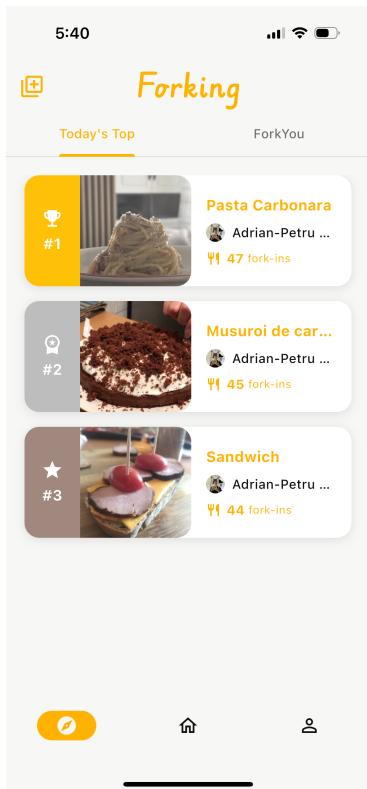


Figura 4.5: *
Discover



Figura 4.6: *
Home

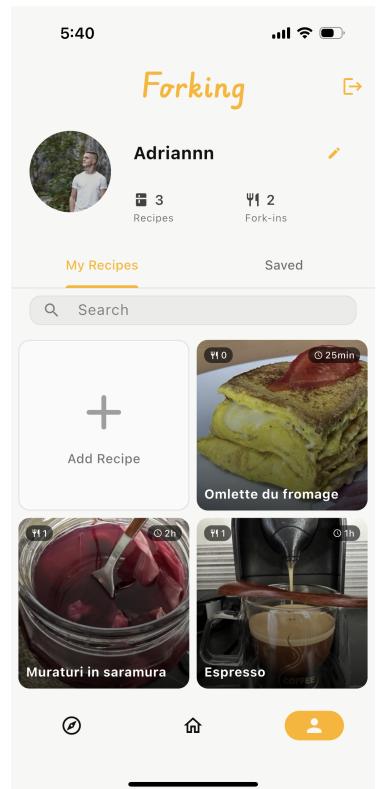


Figura 4.7: *
Profile

Figura 4.8: Navigarea principală în aplicație prin cele trei tab-uri: Home, Discover și Profile

4.3.2 Gestionarea stării aplicației

Starea activă a tab-urilor este gestionată local printr-un `setState()` în `MainScreen`, iar pentru secțiunile care depind de date externe (ex: Discover), se folosesc `FutureBuilder` și logica asincronă implementată în metode dedicate.

În fișierul `discover_screen.dart`, datele sunt încărcate asincron din Firebase folosind metode ca `_loadRecommendations()` și `_loadTodayFavorites()`, care actualizează starea ecranului după primirea rezultatelor. Pe durata încărcării sunt afișate elemente tip `CircularProgressIndicator`, iar mai apoi sunt afișate datele relevante în tab-urile Today's Top și ForkYou.

Fluxul este inițiat în `initState()`, unde sunt inițializate controlerele și pornite ascultările relevante asupra evenimentelor de swipe (din alt ecran), menținând sincronizarea între secțiuni.

Listing 4.2: Inițializarea controlerelor și încărcarea asincronă a datelor în Discover

```
@override
void initState() {
    super.initState();
    _tabController = TabController(length: 2, vsync: this);
    _loadTodayFavorites();
    _loadRecommendations();

    ...
}

33 ...@override
34 ...void initState(){[
35 ...|super.initState();
36 ...|_tabController = TabController(length: 2, vsync: this);
37 ...|_loadTodayFavorites();
38 ...|_loadRecommendations();
39 ...
40 ...// Listen to swipe events from home screen
41 ..._homeSwipeSubscription = RecipeEventBus.homeSwipeStream.listen((recipeId) {
42 ...|_removeRecipeFromRecommendations(recipeId);
43 ...});
44 ...]
```

Figura 4.9: Inițializarea logicii de stare în discover_screen.dart

4.4 Mecanismul de swipe și recomandare

Unul dintre elementele centrale ale aplicației *Forking Food* este interacțiunea de tip *swipe*, inspirată din aplicațiile tip dating. Utilizatorii pot glisa spre dreapta pentru a salva o rețetă (*Fork-in*) sau spre stânga pentru a o ignora (*Fork-out*) și a trece la următoarea. Această funcționalitate este implementată cu ajutorul pachetului `flutter_card_swipe` care oferă suport pentru animații fluide și control asupra comportamentului cardurilor.

4.4.1 Logica de gestionare a swipe-urilor

Funcția centrală care gestionează swipe-ul utilizatorului este `_onSwipe`, definită în fișierul `home_screen.dart`. Aceasta detectează direcția swipe-ului, apelează metoda `saveSwipe()` din clasa `RecipeService` pentru a salva acțiunea în Firebase, și actualizează pool-ul local de rețete.

În plus, `_onSwipe` emite un eveniment prin `RecipeEventBus`, care permite sincronizarea ecranului Discover (tabul ForkYou) cu Home, eliminând cardurile care au fost deja swipe-uite din alte secțiuni ale aplicației, menținând consistența la nivelul datelor.

Listing 4.3: Gestionează swipe-ul utilizatorului în Home

```
void _onSwipe(int index, CardSwiperDirection direction) async {
    if (index >= recipes.length) return;

    final recipe = recipes[index];
    final userId = authService.userId;

    if (userId != null) {
        await recipeService.saveSwipe(
            userId: userId,
            recipeId: recipe.id,
            direction: direction == CardSwiperDirection.right
                ? SwipeDirection.right
                : SwipeDirection.left,
        );
    }

    RecipeEventBus.emitHomeSwipe(recipe.id);

    maintainPoolSize();
}
```

4.4.2 Întreținerea pool-ului de rețete

Pentru a asigura un flux continuu de carduri cu rețete, aplicația folosește o structură locală numită `availableRecipes`, un „pool” de rețete obținute din Firebase. Metoda `maintainPoolSize()` verifică periodic dimensiunea pool-ului și apelează `loadMoreRecipes()` la nevoie.

Aceasta din urmă obține rețete prin intermediul `getRecipesForFeed()` din `RecipeService`, care interoghează rețetele aleator din Firestore, fără vreun algoritm

de personalizare în acest caz.

Listing 4.4: Întreținerea pool-ului local de rețete

```
void maintainPoolSize() {
    const int desiredPoolSize = 10;
    if (availableRecipes.length < desiredPoolSize) {
        loadMoreRecipes();
    }
}

Future<void> loadMoreRecipes() async {
    final userId = authService.userId;
    if (userId == null) return;

    final newRecipes = await recipeService.getRecipesForFeed(
        userId: userId,
        limit: 10,
    );

    setState(() {
        availableRecipes.addAll(newRecipes);
    });
}
```

După fiecare swipe, o rețetă nouă este promovată din pool către lista principală prin `promoteRecipeFromPool()`, menținând interfața fluidă și constant populată.

4.4.3 Recomandări personalizate

Algoritmul de recomandare este aplicat exclusiv în secțiunea `Fork You` din tab-ul `Discover`, precum și în funcția de căutare din ecranul `Profile`. Aceste recomandări se bazează pe date colectate din swipe-urile precedente, scoruri de preferință și alte metadate salvate în Firebase.

Pentru secțiunea `Home`, fluxul de rețete este general și aleator, fără personalizare, dar logica de interacțiune și întreținere este similară, cu diferența majoră că `Fork You` afișează doar rețete relevante pentru utilizator.



Figura 4.10: *

Fork-Out

Figura 4.12: Interfață de swipe din Home cu carduri dinamice



Figura 4.11: *

Fork-In

4.5 Adăugarea unei rețete

Una dintre funcționalitățile esențiale ale aplicației *Forking Food* este capacitatea utilizatorilor de a-și publica propriile rețete culinare. Această funcție este implementată printr-un formular complex, ce acoperă toate tipurile de detalii relevante unei rețete: imagine, titlu, descriere, ingrediente, instrucțiuni pas-cu-pas, etichete și criterii dietetice.

Procesul de creare este asistat de o interfață intuitivă și de validări progresive,

iar datele sunt ulterior salvate în Firestore Database, împreună cu imaginile aferente stocate în Firebase Storage. În continuare, vom detalia structura acestui formular și modul în care datele completate sunt procesate și expediate în sistem.

4.5.1 Formularul complet: structură și validare

Pentru a adăuga o rețetă nouă, aplicația *Forking Food* oferă un formular detaliat implementat în fișierul `add_recipe_screen.dart`. Acesta include toate câmpurile necesare pentru completarea unei rețete valide, precum și mecanisme de validare și feedback vizual.

Formularul este împărțit în mai multe secțiuni:

- **Imagine principală** – utilizatorul trebuie să selecteze o imagine reprezentativă. Imaginea este comprimată, redimensionată și decupată într-un raport de aspect predefinit;
- **Titlu și descriere** – ambele câmpuri sunt obligatorii și sunt validate local prin starea internă a formularului;
- **Timp estimat de preparare** – ales cu ajutorul unui CupertinoPicker, validat pentru a nu permite valoarea zero;
- **Ingrediente** – adăugate individual și afișate într-o listă interactivă, fiecare cu opțiune de ștergere;
- **Instrucțiuni pas-cu-pas** – fiecare pas include un câmp de text și o imagine optională; se validează existența a cel puțin unui pas descriptiv;
- **Etichete (tags)** – sistem de selecție/autocomplete conectat la baza de date, cu posibilitatea de adăugare de etichete noi;
- **Criterii dietetice** – asemănător cu etichetele, utilizatorul poate selecta sau introduce noi criterii (ex: Vegan, Fără gluten).

Validarea este realizată printr-o combinație de:

- `FormState.validate()` – pentru câmpurile de text;
- stări interne (`_showTitleError`, `_showTimeError`, etc.) – pentru afișarea mesajelor de eroare vizuală;

- funcții dedicate (`_scrollToInvalidField()`, `_showConfirmDialog()`) – pentru ghidarea utilizatorului în completarea corectă a formularului.

După completarea cu succes, toate datele sunt transformate într-un obiect `Recipe` și transmise către serviciul `RecipeService` pentru a fi salvate în Firebase Firestore, iar imaginile sunt încărcate în Firebase Storage.

The screenshot shows the 'Forking' app interface for creating a new recipe. At the top, it says 'Forking'. Below that is a placeholder for a photo with a camera icon and the text 'Tap to add a photo'. There are two input fields: 'Recipe Title' and 'Description'. Under 'Description', there is an 'Estimated time:' field containing '0h 0min'. Below these is a 'Tags' section with an 'Add or search tag' input field. At the bottom is a large yellow 'Publish Recipe' button.

Figura 4.13: *

Formular gol

Figura 4.15: Formularul de adăugare rețetă în Forking Food

The screenshot shows the 'Forking' app interface for creating a new recipe at 2:51. It includes all the fields from the previous screenshot, but with validation feedback. The 'Description' field now contains 'Omleta cu branza paine prajita si rosii'. The 'Estimated time:' field is highlighted with a yellow oval containing '0h 25min'. The 'Tags' section has a note '# Add or search tag'. The 'Ingredients' section has a red border and the message 'No ingredients added yet.' and 'Please add at least one ingredient.' Below it is an 'Ingredient' input field with a plus sign icon. At the bottom is a large yellow 'Publish Recipe' button.

Figura 4.14: *

Exemplu de validare

4.5.2 Salvarea datelor în Firestore și Storage

După completarea formularului, rețeta este validată local și salvată în Firebase. Procesul începe în metoda `_saveRecipe()`, care asigură integritatea datelor printr-o serie de verificări: titlu și descriere complete, imagine principală atașată, cel puțin un ingredient și un pas valid de instrucțiuni.

În cazul în care unele criterii nu sunt îndeplinite, sunt afișate mesaje de eroare corespunzătoare, iar utilizatorul este ghidat vizual către secțiunea problematică. După validare, rețeta este transformată într-un obiect `Recipe`, ce conține toate câmpurile relevante: autorul, imaginea, lista de ingrediente, instrucțiunile, timp aproximativ, tag-uri și criterii dietetice.

Imaginiile (atât cea principală, cât și cele atașate pasilor) sunt încărcate în **Firebase Storage**, iar URL-urile publice rezultate sunt incluse în obiectul rețetei. În paralel, dacă utilizatorul a introdus tag-uri sau criterii dietetice noi, acestea sunt adăugate în colecțiile corespunzătoare din Firestore.

În final, rețeta este salvată în colecția `recipes` din Firestore prin metoda `saveRecipe()` din `RecipeService`. După salvare, utilizatorul este notificat cu un mesaj de confirmare, formularul este resetat pentru introducerea unei noi rețete iar utilizatorul redirectionat la ecranul din care a accesat meniul de adăugare de rețete.

```

451     - if (_userId == null || displayName == null) {
452         throw Exception('User not authenticated');
453     }
454
455     // Upload new tags to database first
456     final existingTags = await _recipeService.getAllTags();
457     for (String tag in _selectedTags) {
458         if (!existingTags.contains(tag)) {
459             try {
460                 await _recipeService.addTag(tag);
461             } catch (e) {
462                 // Continue
463             }
464         }
465     }
466     final existingCriteria = await _recipeService.getAllDietaryCriteria();
467     for (String criteria in _selectedDietary) {
468         if (!existingCriteria.contains(criteria)) {
469             try {
470                 await _recipeService.addDietaryCriteria(criteria);
471             } catch (e) {
472                 // Continue
473             }
474         }
475     }
476     final recipe = Recipe(
477         id: DateTime.now().millisecondsSinceEpoch.toString(),
478         title: _titleController.text.trim(),
479         imageUri: _pickedImage.path,
480         description: _descriptionController.text.trim(),
481         ingredients: List.from(_ingredients),
482         instructions: _instructions.where((i) => i.description.trim().isNotEmpty).toList(),
483         totalEstimatedTime: Duration(hours: _totalHours, minutes: _totalMinutes),
484         tags: List.from(_selectedTags),
485         creatorName: displayName,
486         creatorPhotoURL: photoURL,
487         createdAt: DateTime.now(),
488         dietaryCriteria: _selectedDietary.toList(),
489     ); // Recipe
490
491     await _recipeService.saveRecipe(recipe);
492
493     if (mounted) {
494         Navigator.of(context).pop();
495     }
496
497     if (mounted) {
498         ScaffoldMessenger.of(context).showSnackBar(
499             const SnackBar(content: Text('Recipe saved successfully!')),
500         );
501
502         resetForm();
503         if (widget.onRecipeAdded != null) {
504             widget.onRecipeAdded?.call();
505         } else {
506             Navigator.pop(context);
507         }
508     }
509 }
510 } catch (e) {
511     if (mounted) {
512         Navigator.of(context).pop();
513     }
514
515     if (mounted) {
516         ScaffoldMessenger.of(context).showSnackBar(
517             SnackBar(
518                 content: Text('Failed to save recipe: ${e.toString()}'),
519                 backgroundColor: Theme.of(context).colorScheme.error,
520             ), // SnackBar
521         );
522     }
523 }
524 finally {
525     if (mounted) {
526         setState(() {
527             _isSaving = false;
528         });
529     }
530 }

```

Figura 4.16: *

cod 1

Figura 4.18: Funcția de salvare a unei rețete în add_recipe_screen.dart

```

forking-food > lib > screens > add_recipe_screen.dart > _AddRecipeScreenState > _saveRecipe
29 class _AddRecipeScreenState extends State<AddRecipeScreen> {
30
31     Future _saveRecipe() async {
32         if (_formKey.currentState!.validate()) {
33             _scrollToInvalidField();
34             return;
35         }
36
37         if (_pickedImage == null) {
38             setState(() => _showImageError = true);
39             _scrollToInvalidField();
40             return;
41         }
42
43         if (_titleController.text.trim().isEmpty) {
44             setState(() => _showTitleError = true);
45             _scrollToInvalidField();
46             return;
47         }
48
49         if (_descriptionController.text.trim().isEmpty) {
50             setState(() => _showDescriptionError = true);
51             _scrollToInvalidField();
52             return;
53         }
54
55         if (_totalHours == 0 && _totalMinutes == 0) {
56             setState(() => _showTimeError = true);
57             _scrollToInvalidField();
58             return;
59         }
60
61         if (_ingredients.isEmpty) {
62             setState(() => _showIngredientsError = true);
63             _scrollToInvalidField();
64             return;
65         }
66
67         // Validate instructions - at least one step with description
68         final validInstructions = _instructions.where((i) => i.description.trim().isNotEmpty).toList();
69         if (validInstructions.isEmpty) {
70             setState(() {
71                 _showInstructionErrors.fillRange(0, _showInstructionErrors.length, true);
72             });
73             _scrollToInvalidField();
74             return;
75         }
76
77         // Check for steps with images but no description
78         for (int i = 0; i < _instructions.length; i++) {
79             final step = _instructions[i];
80             if ((step.mediaUrl != null && step.description.trim().isEmpty) ||
81                 (step.localMediaFile != null && step.description.trim().isEmpty)) {
82                 setState(() {
83                     _showInstructionErrors[i] = true;
84                 });
85                 _scrollToInvalidField();
86             }
87         }
88
89         setState(() {
90             _isSaving = true;
91         });
92
93         // Show loading indicator
94         showDialog(
95             context: context,
96             barrierDismissible: false,
97             builder: (context) => const Center(
98                 child: CircularProgressIndicator(),
99             ), // Center
100         );
101
102         try {
103             final String? userId = authService.userId;
104             final String? displayName = authService.userName;
105             final String? photoURL = authService.userPhotoURL;
106
107             if (userId == null || displayName == null) {
108                 throw Exception('User not authenticated');
109             }
110
111             // Upload new tags to database first
112             final existingTags = await _recipeService.getAllTags();
113             for (String tag in _selectedTags) {
114                 if (!existingTags.contains(tag)) {
115                     try {
116                         await _recipeService.addTag(tag);
117                     } catch (e) {
118                         // Continue
119                     }
120                 }
121             }
122
123             final existingCriteria = await _recipeService.getAllDietaryCriteria();
124             for (String criteria in _selectedDietary) {
125                 if (!existingCriteria.contains(criteria)) {
126                     try {
127                         await _recipeService.addDietaryCriteria(criteria);
128                     } catch (e) {
129                         // Continue
130                     }
131                 }
132             }
133
134             final recipe = Recipe(
135                 id: DateTime.now().millisecondsSinceEpoch.toString(),
136                 title: _titleController.text.trim(),
137                 imageUri: _pickedImage.path,
138                 description: _descriptionController.text.trim(),
139                 ingredients: List.from(_ingredients),
140                 instructions: _instructions,
141                 totalEstimatedTime: Duration(hours: _totalHours, minutes: _totalMinutes),
142                 tags: List.from(_selectedTags),
143                 creatorName: displayName,
144                 creatorPhotoURL: photoURL,
145                 createdAt: DateTime.now(),
146                 dietaryCriteria: _selectedDietary,
147             ); // Recipe
148
149             await _recipeService.saveRecipe(recipe);
150
151             if (mounted) {
152                 Navigator.of(context).pop();
153             }
154
155             if (mounted) {
156                 ScaffoldMessenger.of(context).showSnackBar(
157                     const SnackBar(content: Text('Recipe saved successfully!')),
158                 );
159             }
160
161             if (mounted) {
162                 Navigator.pop(context);
163             }
164
165             if (mounted) {
166                 ScaffoldMessenger.of(context).showSnackBar(
167                     SnackBar(
168                         content: Text('Failed to save recipe: ${e.toString()}'),
169                         backgroundColor: Theme.of(context).colorScheme.error,
170                     ), // SnackBar
171                 );
172             }
173
174         } finally {
175             if (mounted) {
176                 setState(() {
177                     _isSaving = false;
178                 });
179             }
180         }
181     }
182 }

```

Figura 4.17: *

cod 2

4.6 Alte funcționalități implementate

Pe lângă funcționalitățile principale descrise anterior, aplicația *Forking Food* integrează și alte componente utile pentru experiența completă a utilizatorului. Acestea includ gestionarea profilului personal, secțiuni speciale de explorare precum *Today's Top*, mecanisme de căutare avansată, un sistem de feedback tactil cât și administrarea rețetelor proprii și a celor salvate.

În această secțiune sunt prezentate aceste elemente secundare, dar importante, care contribuie la utilitatea și ergonomia aplicației.

4.6.1 Profilul utilizatorului

Ecranul de profil oferă utilizatorului acces la propriile date, statistici și rețetele cu care a interacționat. În partea superioară, sunt afișate numele și avatarul, ambele fiind editabile printr-un simplu tap. Sub acestea sunt afișate două seturi de statistici: numărul total de rețete publicate și totalul de *Fork-In*-uri primite de acestea (adică numărul de salvări a rețetelor acestora făcute de alți utilizatori).

Navigarea în profil se face printr-un TabBar cu două secțiuni: **My Recipes** (rețetele publicate de utilizator) și **Saved** (rețetele salvate prin swipe tip *Fork-In*). Utilizatorul poate comuta între cele două fie prin apasare pe tab-uri, fie prin swipe orizontal.

Ambele secțiuni prezintă rețetele într-un layout de tip grilă, fiecare rețetă fiind afișată sub formă de card. La apăsarea unui card, se deschide un overlay fullscreen de tip bottom sheet, care oferă o vizualizare detaliată a rețetei (similară cu cea din cardurile din ecranul de swipe) și care poate fi închis printr-un simplu gest de glisare în jos.

În partea de sus a ecranului se regăsește și un buton de logout, care permite deconectarea din aplicație și revenirea la ecranul inițial de întâmpinare.

Un element avansat al ecranului de profil este funcționalitatea de **căutare inteligență**, care permite filtrarea rețetelor prin introducerea unor termeni relevanți. Căutarea se face pe baza titlului, descrierii, a ingredientelor și a etichetelor fiecărei rețete, iar rezultatele sunt sortate intelligent în funcție de scorul de compatibilitate calculat pe baza preferințelor salvate ale utilizatorului (rezultate din swipe-urile anterioare).

4.6.2 Secțiunea Today's Top

Secțiunea Today's Top, disponibilă în tab-ul Discover, oferă un podium de 3 rețete, sortate după numărul de *Fork-In*-uri primite în ziua curentă. Aceasta este comun pentru toți utilizatorii și nu este personalizat.

Scopul acestei secțiuni este de a oferi inspirație rapidă și acces la cele mai apreciate rețete ale zilei. Rețetele sunt încărcate asincron din Firebase, iar afișarea este organizată sub formă de listă scrollabilă vertical, cu carduri de rețete în format compact, care pot fi deschise în format *fullscreen*.

Această abordare permite utilizatorilor noi să descopere rețete relevante și bine cotate, chiar înainte de a-și construi un profil de preferințe prin swipe-uri sau căutări.

4.6.3 Gestionarea rețetelor proprii și salvate

Secțiunea Profile oferă utilizatorului acces la rețetele publicate de el, precum și la cele pe care le-a salvat anterior prin swipe dreapta (*Fork-In*). Acestea sunt organizate în două tab-uri: My Recipes și Saved, între care se poate naviga fie prin *swipe* orizontal, fie selectând tab-ul dorit.

Fiecare rețetă este afișată sub formă de card într-un grid responsive. La apăsarea pe un card, se deschide un overlay pe toată înălțimea ecranului, cu detaliile complete despre rețeta respectivă, într-un format similar cu cel folosit în swiper-ul din ecranul Home.

În tabul My Recipes, utilizatorul are opțiunea de a șterge rețetele publicate, apăsând butonul dedicat din colțul cardului. O fereastră de confirmare va fi afișată înainte de ștergerea permanentă din baza de date.

Căutarea este implementată direct în această secțiune și permite filtrarea rețetelor pe bază de text. Sunt luate în considerare câmpurile titlu, descriere, ingrediente și etichete. Pentru rezultatele din tabul Saved, se aplică o sortare intelligentă care ține cont de scorul de compatibilitate cu preferințele utilizatorului.

De asemenea, profilul afișează două statistici relevante:

- **Număr total de rețete poste;**
- **Număr total de Fork-In-uri primite** pe rețetele proprii.

În colțul din dreapta sus se găsește butonul de Logout, care permite utilizatorului să se deconecteze din aplicație în orice moment.

4.7 Capturi de ecran și interfața aplicației

Pentru a ilustra modul de funcționare al aplicației *Forking Food*, în această secțiune sunt prezentate o serie de capturi de ecran reprezentative. Acestea evidențiază interfața vizuală, fluxurile principale de utilizare dar și alte elementele cheie care contribuie la o experiență intuitivă și modernă.

Aplicația a fost dezvoltată cu un design coherent, folosind elemente vizuale clare și un layout adaptiv, optimizat pentru dispozitive mobile.

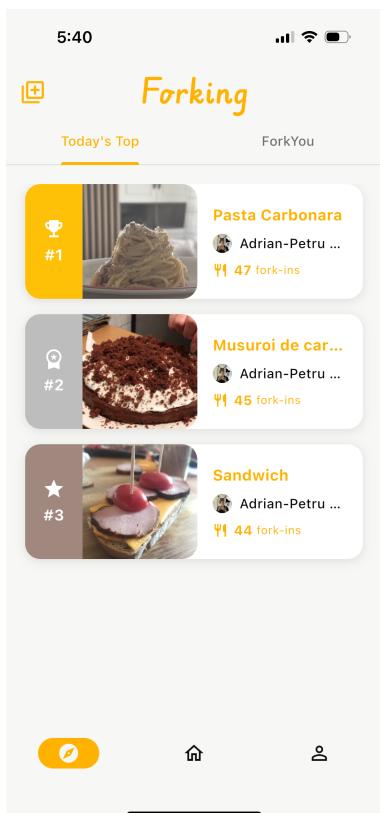


Figura 4.19: *
Sectiunea Today's Top



Figura 4.20: *
Splash Screen

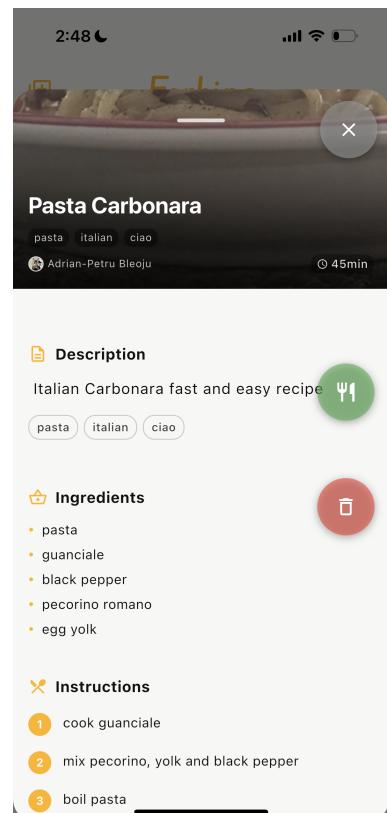


Figura 4.21: *
Vizualizare reteata
fullscreen

Figura 4.22: Elemente suplimentare din interfață

Capitolul 5

Concluzii și direcții viitoare

5.1 Reflecție personală

Dezvoltarea aplicației *Forking Food* a fost o provocare asumată și un exercițiu de învățare aplicată. Am trecut prin momente în care nu știam exact care e pasul următor, dar faptul că am reușit să duc acest proiect până la capăt mi-a consolidat încrederea în capacitatea mea de a construi aplicații funcționale, reale și coerente. Am învățat importanța planificării, a modularității în cod, dar mai ales a răbdării și perseverenței într-un proces de dezvoltare continuu.

5.2 Direcții de dezvoltare viitoare

Cu toate că aplicația este complet funcțională și în stadiul actual, există mai multe direcții prin care ar putea fi extinsă:

- **Notificări push** – pentru a anunța utilizatorii despre rețete noi, like-uri primite sau activitate relevantă;
- **Funcționalități sociale** – posibilitatea de a urmări alți utilizatori, a comenta rețete sau a crea colecții personale;
- **Mod offline / caching intelligent** – pentru a permite accesul la rețetele deja salvate fără o conexiune la internet;
- **Gamificare** – insigne, badge-uri, progres de tip „chef level” pentru a crește motivația utilizatorului. Limitarea numarului de *Fork-In*-uri zilnici și recompensarea cu un număr de *Fork-In*-uri extra pentru adăugarea de rețete proprii;

Aceste îmbunătățiri ar contribui la fidelizarea utilizatorilor și ar duce aplicația mai aproape de un produs pregătit pentru lansare publică.

5.3 Concluzie generală

Forking Food pornește de la o idee simplă – explorarea rețetelor prin swipe – și o transformă într-o aplicație coerentă, scalabilă și pregătită pentru utilizatori reali. Implementarea sa m-a ajutat să integrez și să aplic concepte esențiale din dezvoltarea mobilă modernă, iar rezultatul final reflectă atât munca depusă, cât și progresul personal din timpul acestui proiect.

Bibliografie

- flutter_card_swiper, *Swipe-based card stack*, https://pub.dev/packages/flutter_card_swiper, accesat în iunie 2025
- firebase_core, *Firebase Core plugin*, https://pub.dev/packages/firebase_core, accesat în iunie 2025
- firebase_auth, *Firebase Authentication plugin*, https://pub.dev/packages/firebase_auth, accesat în iunie 2025
- cloud_firestore, *Cloud Firestore plugin*, https://pub.dev/packages/cloud_firestore, accesat în iunie 2025
- firebase_storage, *Firebase Storage plugin*, https://pub.dev/packages/firebase_storage, accesat în iunie 2025
- google_sign_in, *Google Sign-In plugin*, https://pub.dev/packages/google_sign_in, accesat în iunie 2025
- flutter_facebook_auth, *Facebook Auth plugin*, https://pub.dev/packages/flutter_facebook_auth, accesat în iunie 2025
- google_fonts, *Use Google Fonts in Flutter*, https://pub.dev/packages/google_fonts, accesat în iunie 2025
- vibration, *Trigger device vibration on feedback*, <https://pub.dev/packages/vibration>, accesat în iulie 2025
- *Materiale oficiale Flutter*, <https://flutter.dev>, accesat în iulie 2025
- *TPPM – Tehnici de Programare pe Platforme Mobile*, https://gdt050579.github.io/dart_course_fii/administrative.html, accesat în iunie 2025