UNIVERSITATEA DIN BUCUREȘTI

FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

SPECIALIZAREA INFORMATICĂ

Lucrare de licență

„Adoptă un prieten”

Aplicație Android pentru adopția de animale

Absolvent

Oana-Teodora Dima

Coordonator științific

Lect. Dr. Anca Mădălina Dobrovăț

București, iunie 2022

Rezumat

Aplicația „Adoptă un prieten” urmărește diminuarea numărului de animale de companie abandonate din România, prin facilitarea procesului de adopție al acestora. Utilizatorii pot adăuga anunțuri atunci când vor să trimită un animal spre adopție, și-au pierdut animalul de companie sau au găsit un animal și doresc să îi găsească stăpânul sau o nouă familie adoptivă.

Procesul de adopție este susținut și de către un algoritm care face potriviri între profilul psihologic al utilizatorului și animalele înregistrate în baza de date. Profilul fiecărui utilizator este conturat în urma răspunsurilor acestuia la un test scurt de personalitate. În plus, vine și în ajutorul celor care au deja un animal de companie, prin centralizarea unei liste largi și diversificate de servicii dedicate și localizate cum ar fi: dog walking, bonă la domiciliu, farmacii și cabinete veterinare, adăposturi și pensiuni sau servicii de îngrijire cosmetică pentru animale.

Un alt obiectiv pe care „Adoptă un prieten” și-l propune, este de a dezvolta o comunitate online cât mai implicată în protecția și îngrijirea animalelor. Astfel, o aplicație mobile este modul perfect prin intermediul căreia acest obiectiv poate fi atins. Fiind concepută pentru Android, crește nivelul de implicare și loialitate al utilizatorilor de smartphone, creând o punte între mediul digital și cel fizic și interconectând iubitorii de animale din întreaga țară.

Abstract

The “Adopt a friend” application aims to reduce the number of abandoned pets in Romania by facilitating their adoption process. Users can post ads when they want to send a pet for adoption, if they have lost theirs or if they have found one and want to find its owner or a new foster family.

The adoption process is supported by an algorithm that matches the user’s psychological profile with the animals available in the database. The profile of each user is outlined following their responses to a short personality test. Furthermore, the app helps those who already have a pet by centralizing a wide and diverse list of dedicated and localized services such as dog walking, pet sitting, pharmacies and vet clinics, shelters and hostels or cosmetic care services for pets.

Another goal that “Adopt a friend” has is to develop an online community involved in the protection and taking care of animals. Being a mobile application is the perfect way to achieve this target. Designed for Android, it increases the engagement and loyalty of smartphone users, building a bridge between the digital and physical environment and connecting animal lovers across the country.

Cuprins

Capitolul 1 - INTRODUCERE 8
1.1. Motivație 8
1.2. Preliminarii 8
1.3. Nevoi pe care aplicația le satisface 10
1.4. Aplicații Android similare 11
1.5. Ce aduce nou? 13
1.6. Structura lucrării 14
Capitolul 2 - UTILIZAREA TEHNOLOGIILOR ALESE 17
2.1. Flutter Framework 17
2.1.1. Caracteristici definitorii 17
2.1.2. Plugins 19
2.2. Spring Boot 20
2.2.1. Apache Maven - configurări 21
2.2.2. JSON Web Token 21
2.2.3. Trimitere email 23
2.3. Android 24
2.4. Python - Algoritm de matching 25
2.4.1. Baza științifică 26
2.4.2. Colectarea și prelucrarea datelor de antrenare 27
2.4.3. Implementare model 29
Capitolul 3 - ARHITECTURA SISTEMULUI 31
3.1. Backend 31
3.1.1. Rest API 31

3.1.2. Baza de date 33
3.2. Frontend 34
3.2.1. Deep links 35
3.2.2. Splash Screen 36
Capitolul 4 - PREZENTAREA APLICAȚIEI 37
4.1. UI și UX 37
4.2. Sign in și sign up 38
4.3. Schimbarea parolei 40
4.4. Pagina principală 41
4.5. Publicare anunțuri 42
4.5.1. Publicare anunțuri despre animale 42
4.5.2. Publicare anunțuri despre servicii 43
4.6. Panou de control administrator 44
4.7. Serviciul de suport 46
4.8. Sistemul de salvare în favorite 47
4.9. Matching utilizator - animal 48
4.10. Mesaje informative: ecrane speciale, validări și pop-ups 50
Capitolul 5 - CONCLUZII ȘI PERPESCTIVE 52
5.1. Concluzii 52
5.2. Perspective 53
BIBLIOGRAFIE 54

Listă de figuri

Figura 1.1. - Pagina principală „Adoptă un prieten” 15
Figura 2.1. - Conținut Template Screen 18
Figura 2.2. - Layout mesaj de validare 19
Figura 2.3. - O parte din conținutul fișierului .env 19
Figura 2.4. - Cod pentru redimensionarea automată a componentelor de UI 20
Figura 2.5. - Funcționarea JWT în aplicație 22
Figura 2.6. - Trimiterea unui email de înștiințare utilizatorului care a fost blocat 23
Figura 2.7. - Intent filters pentru configurare deep links 24
Figura 2.8. - Generare iconiță de lansare folosind Image Asset Studio 25
Figura 2.9. - Prelucrarea răspunsurilor formularului în date de input și output 28
Figura 2.10. - Clasificatorul utilizat în algoritmul de matching 29
Figura 2.11. - Matrici de confuzie: LinearSVC vs SVC(kernel ='linear') 30
Figura 3.1. - Proces de funcționare server 31
Figura 3.2. - Serviciul pentru crearea unui animal 32
Figura 3.3. - Diagrama conceptuală a bazei de date 33
Figura 3.4. - Acordarea accesului doar administratorilor 34
Figura 3.5. - Flux de funcționare client 34
Figura 3.6. - Rutare deep links 35
Figura 3.7. - Splash Screen aplicație 36
Figura 4.1. - Onboarding Screens 37
Figura 4.2. - Paleta de culori utilizată 38
Figura 4.3. - Mesaje de informare pentru confirmarea adresei de email 39
Figura 4.4. - Prezentarea ecranelor pentru autentificare 40
Figura 4.5. - Flow pentru schimbarea parolei 41
Figura 4.6. - Pagina principală 41
Figura 4.7. - Creare profil animal 43
Figura 4.8. - Creare profil serviciu 44
Figura 4.9. - Panou de control administrator 45
Figura 4.10. - Acțiuni administrator 45

Figura 4.11. - Formular de contact pentru serviciul de suport 46
Figura 4.12. - Conținutul email-ului primit pentru serviciul de suport 47
Figura 4.13. - Ecran cu favorite 48
Figura 4.14. - Proces de matching utilizator - animal 49
Figura 4.15. - Ecrane speciale de informare 50
Figura 4.16. - Validări formulare 51
Figura 4.17. - Pop-up-uri de informare 51

Capitolul 1 - INTRODUCERE

1.1. Motivație

Încă din copilărie am fost înconjurată de animale, crescând atât în mediul urban, cât și în cel rural. Câinii, pisicile, găinile, porumbeii, iepurii și veveriţele mi-au fost în mare parte prietenii de joacă din timpul vacanţelor de vară. Ulterior, mi s-a imprimat o pasiune pentru ocrotirea și îngrijirea necuvântătoarelor. Astfel, toți câinii și toate pisicile a căror stăpâni au fost membrii familiei mele au fost adoptați, de pe stradă sau de la vecini, neavând un centru de adopție în apropierea locuinței.

Când am pornit în căutarea unei teme pentru această lucrare, mi-am rezervat o perioadă de brainstorming de 3 săptămâni, în care am căutat inspirație în orice sursă de informație din jurul meu. Intenționam să fac o aplicație cu un impact semnificativ asupra comunității și cu perspectivă privind dezvoltarea ulterioară și publicarea acesteia. Până într-o zi, când pe Facebook mi-a apărut o campanie a unei asociații de adopție din orașul meu natal, asociație înființată destul de recent. Atunci mi-am dat seama că pot să mă îndrept către un subiect care este tratat cu ignoranță în țara noastră și anume, salvarea animalelor de companie prin adopție. În plus, pe parcursul cercetării mele, am descoperit și faptul că, în România nu există o aplicație mobile dedicată, cele deja existente fiind construite pentru zone din afara granițelor noastre.

1.2. Preliminarii

În cele ce urmează vor fi ilustrate cele 3 componente principale pe care aplicația le folosește în îndeplinirea obiectivelor propuse:

- Publicarea uni animal
- Publicarea unui serviciu
- Matching utilizator - animal

Publicarea unui animal

Procesul de adopție are la bază publicarea unui anunț care conține date despre un animal sau mai multe de același fel (cazul puilor nou născuți), împreună cu o listă de detalii care să ajute utilizatorul să își contureze o mai bună imagine despre acesta. Formularul pe care un user îl completează pentru publicarea unui astfel de anunț are majoritatea câmpurilor obligatorii, vârsta fiind singura opțională, deoarece în cazul animalelor găsite pe stradă nu se poate specifica cu exactitate. Profilul unui animal cuprinde următoarele informații: nume, categorie (câini, pisici, păsări, hamsteri și altele), rasă (poate fi specificată și ca metis/neidentificat), culoare, sex, descriere, status (spre adopție, găsit sau pierdut), adresa (localitate, județ/sector și țară), talia (mică, medie sau mare, depinzând de categoria selectată), data ultimei modificări (setată automat), vaccinat (da, nu sau nu știu), vârsta și un număr nelimitat de poze care vor fi trunchiate sub forma unui pătrat la afișare. Datele de contact (nume, email, telefon și opțional, site-ul web) ale persoanei sau organizației proprietară vor fi preluate automat și afișate la apăsarea butonului cu simbolul unui telefon din profilul animalului.

Publicarea unui serviciu

„Adoptă un prieten” își propune să ajute toți stăpânii de animale, fiind dedicată și utilizatorilor care nu intenționează să adopte un animal nou, deoarece dețin deja unul. În România, servicii precum cele de dog walking, îngrijire la domiciliu sau pensiuni și hoteluri pentru animale nu sunt foarte populare. În consecință, aplicația reprezintă un loc ideal de popularizare a acestora. Asemănător procesului de publicare al unui anunț despre un animal, se poate crea și un profil cu informațiile legate despre un anumit serviciu. Singurul câmp cu caracter opțional este cel în care se poate specifica site-ul web, restul având caracter obligatoriu: tip, nume, telefon, descriere, data ultimei modificări (setată automat), categoria de animale cărora li se adresează și adresa (stradă, număr, localitate, județ/sector, țară), însoțită și de ilustrarea acesteia pe hartă.

Matching utilizator - animal

O soluție dedicată facilitării procesului de adopție și implicării utilizatorilor în comunitatea pe care aplicația își dorește să o dezvolte, o reprezintă algoritmul de matching. Fiind un algoritm de machine learning, prezice categoria din care ar trebui să facă parte

animalul ideal pentru personalitatea utilizatorului. Datele de antrenare și testare au fost colectate în urma celor 391 de răspunsuri la un formular anonim. În lipsa unui eșantion mai mare de date, acuratețea acestuia pe datele de test este de 54,43%. User-ul completează un test scurt de personalitate și în urma răspunsurilor colectate, algoritmul prezice categoria animalului potrivit pentru acesta. În funcție de conținutul bazei de date de la momentul respectiv, pe server se alege în mod aleatoriu un animal din categoria respectivă, iar răspunsul este trimis către client.

1.3. Nevoi pe care aplicația le satisface

Spre deosebire de alte țări europene, în România nu este constituit un cadru legal aspru privind abandonul sau actele de violență asupra animalelor. Conform legii 205/2004[1], articolul 26, punctul b), abandonul animalelor este considerat act de contravenție și este sancționat cu amendă cuprinsă între 1.000 și 3.000 lei. În plus, conform legii 203/2018[2], articolul 20, punctul 10, persoana în cauză poate achita doar jumătate din minimul amenzii în termen de cel mult 15 zile de la înștiințarea acestuia. În consecință, prevederile actuale constituie niște măsuri mult prea blânde pentru suferința pe care lipsirea unui animal de hrană, adăpost, îngrijire și asistență medicală o poate aduce.

Scopul aplicației „Adoptă un prieten” este de a veni în ajutorul animalelor mai puțin norocoase, ale căror stăpâni nu mai pot să aibă grijă de ele (programul de lucru intens, situația financiară, starea de sănătate precară sau chiar decesul stăpânului), s-au născut într-o familie mult prea numeroasă sau pe stradă, fiind ale nimănui. Practic, contribuie la diminuarea numărului câinilor vagabonzi, al pisicilor care caută hrană printre gunoaie, al păsărilor crescute în casă și apoi lăsate să se descurce singure, a tuturor animalelor moarte de foame sau de frig. Orice iubitor de animale se poate înregistra și poate contribui la salvarea sufletelor necuvântătoare.

Fiind concepută sub forma unei aplicații mobile, utilizatorii au posibilitatea să își aleagă un animal de companie din confortul propriei case. Alegerile nu se limitează doar la animalele

[1] Lege privind protecția animalelor: 205 din 26.04.2004. În Monitorul Oficial al României, 2014. Disponibil: https://legislatie.just.ro/Public/DetaliiDocument/52646. Accesat 20 mai 2022.
[2] Lege privind măsurile de eficientizare a achitării amenzilor contravenționale: 203 din 20.07.2018. În Monitorul Oficial al României, 2018. Disponibil: https://legislatie.just.ro/Public/DetaliiDocument/203115. Accesat 20 mai 2022.

disponibile în centrele specializate, ci la absolut toate cele aflate la nevoie, localizate în proximitatea locuinței sau aflate în limita posibilităților de adopție. În plus, există și o secțiune dedicată de tipul *lost & found*, unde cei care și-au pierdut animalul de companie sau au găsit un animal care pare a fi pierdut pot să posteze un anunț.

În ceea ce privește stăpânii care nu doresc să își mărească familia, „Adoptă un prieten" reprezintă o sursă diversificată de servicii dedicate îngrijirii animalelor de companie. Aceștia pot găsi și servicii noi, ideale nevoilor lor, mai puțin populare în România. De asemenea, sistemele de raportare, suport și control ale administratorului oferă utilizatorului o experiență lipsită de neplăceri, dezvoltând și încrederea și deschiderea acestora în utilizarea aplicației.

În concluzie, aplicația îndeplinește nevoi de siguranță, sprijin, comunitate și îngrijire, atât a celor necuvântătoare, cât și a utilizatorilor săi.

1.4. Aplicații Android similare

- Appets

  „Appets" este o aplicație pentru Android, destinată adopției de animale din diferite zone ale lumii, predominând: Mexic, Peru, Brazilia, Argentina și Spania. Ca simplu utilizator nu ești obligat să îți creezi un cont, decât dacă dorești să publici un anunț. Însă, în lipsa unui cont, nu poți vizualiza datele de contact ale proprietarului unui animal pe care dorești să-l adopți, decât dacă vizionezi o reclamă. În schimb, detaliile serviciilor listate sunt mereu accesibile, fiind independente de un cont de utilizator.

  Câinii, pisicile, păsările și iepurii sunt principalele categorii care sunt evidențiate. În ceea ce privește prezentarea fiecărui animal, sunt afișate date despre: rasă, sex, vârstă, locație, o scurtă descriere și o limită maximă de 5 poze atașate. Anunțurile sunt împărțite în două mari categorii: animale date spre adopție și animale pierdute. Pentru o mai bună navigare, secțiunile dedicate anunțurilor cu animale au și câteva filtre care pot fi aplicate: după oraș, țară, categorie, vârstă și sex. Tipurile de servicii pe care le expune sunt doar cele de dog walking și cabinete veterinare.

❖ Adopt me

„Adopt me” este o aplicație Android dedicată adopției de animale doar din zonele: United States, Canada și Mexic. Un pop-up de informare privind locația de activitate apare la deschiderea acesteia, însă aplicația este accesibilă indiferent de locația curentă a utilizatorului. De asemenea, nu funcționează pe baza unui cont de utilizator, deoarece toate anunțurile provin de la adăposturile de animale din zonele listate anterior, fiind și singurele surse de adopție. Practic, dezvoltatorii aplicației colaborează cu diferite adăposturi pentru a-i ajuta să găsească familii adoptive pentru animalele pe care le îngrijesc.

Principalele categorii menționate sunt: câinii, pisicile, păsările, iepurii și *small and furry*. Pagina principală are un sistem de căutare după rasă și o secțiune de filtrare și sortare după caracteristicile principale enumerate la fiecare anunț, respectiv după data publicării și după distanță. Detaliile oferite privind fiecare animal sunt numeroase: nume animal, sex, data publicării, rasă, vârstă, dimensiune, culoare, status, scurtă descriere și câteva cuvinte cheie. De asemenea, fiecare profil are o secțiune dedicată informațiilor privind adăpostul din care face parte, incluzând chiar și indicații de deplasare cu Google Maps. În plus, în meniul lateral, se poate schimba paleta de culori într-una întunecată și se poate contacta dezvoltatorul prin trimiterea unui email.

❖ Pets Adoption: Adopt pet or post for adoption

„Pets Adoption” este o aplicație Android, care reunește anunțuri despre animalele trimise spre adopție din toate colțurile lumii. Conținutul poate fi vizualizat și fără cont de utilizator, însă pentru adăugarea unui anunț va cere crearea unui cont pe baza numărului de telefon și verificarea acestuia printr-un cod, trimis prin SMS. Aplicația are foarte multe reclame incorporate, navigarea fiind îngreunată de acestea. Însă, există și o versiune fără reclame, dar care nu este gratuită.

Principalele categorii de animale abordate sunt: câini, pisici, păsări și pești. Profilul fiecărui animal are un număr maxim de 3 poze și oferă informații precum: numele animalului, data publicării, rasă, descriere, categorie, sex, dacă a fost sterilizat, strada, localitatea și țara unde este localizat, cât și numele și numărul de telefon al proprietarului. În plus, calculează automat pentru fiecare anunț, distanța dintre locația curentă și locația fiecărui animal afișat, cât și numărul de accesări și de share-uri de pe rețelele de socializare.

Oferă și posibilitatea promovării unui anunț contra cost, prin creșterea nivelului de expunere timp de 7 zile în secțiunea dedicată din pagina principală. În plus, fiecare postare poate fi raportată ca fiind reclamă sau spam. Din meniu se pot dezactiva notificările, se pot trimite email-uri de feedback dezvoltatorului și se poate accesa secțiunea cu animalele care și-au găsit deja o casă.

1.5. Ce aduce nou?

Algoritmul de AI pentru potrivirea personalității utilizatorilor cu unul dintre animalele din baza de date reprezintă un feature nou, nemaiîntâlnit în nicio aplicație de acest gen. Acesta contribuie la implicarea utilizatorilor prin completarea testului scurt de personalitate care este necesar pentru predicție. În plus, un vizitator al aplicației nu poate avea acces la acest algoritm, decât în urma înregistrării unui cont. Astfel, contribuie și la creșterea comunității implicate.

Un alt plus este serviciul de suport prin email accesibil direct din aplicație prin completarea unui formular cu două câmpuri: subiectul și descrierea sesizării. Administratorul aplicației primește automat, pe email, informațiile adiționale alături de email-ul utilizatorului care a sesizat problema. La trimiterea acestuia, user-ul este informat printr-un pop-up că problema lui a fost expediată cu succes și că va fi contactat în cel mai scurt timp pentru remediere.

Posibilitatea încadrării statusului unui animal ca *găsit* este un detaliu care nu se află abordat în nicio aplicație de acest fel. Prin aplicarea filtrelor corespunzătoare, cei care și-au pierdut animalul de companie au mai multe șanse să-l găsească în această secțiune.

Modul explicit de vizitator încurajează procesul de adopție din partea utilizatorilor reticenți atunci când vine vorba de distribuirea datelor personale. Vizitatorii au acces la toate datele proprietarilor de animale, la toate serviciile publicate, la serviciul de suport și la toate modalitățile de căutare și filtrare. Algoritmul de matching, adăugarea unui anunț nou, lista de favorite și raportarea unei postări sunt opțiunile la care fără un cont conectat, nu vor avea acces. Tot din aceleași motive, la crearea unui cont nou în aplicație, nu vor fi cerute datele afișate ca proprietar al unui animal. Numele utilizatorului/ONG-ului/asociației, numărul de telefon și site-ul web vor fi cerute numai în momentul în care un utilizator, cu un cont deja conectat, dorește să publice primul său anunț.

Aplicația urmărește să popularizeze și noi servicii dedicate animalelor din România, abordând numeroase categorii: dog walking, bonă la domiciliu, farmacii și cabinete veterinare, adăposturi, pensiuni/hoteluri, servicii de îngrijire cosmetică pentru animale și altele. Acestea sunt însoțite de nume, scurtă descriere, adresă (stradă, număr, localitate, județ/sector), site web, categoriile de animale cărora li se adresează și telefonul. La fel ca anunțurile animalelor, și serviciile postate au posibilitatea de a fi raportate de utilizatori, având la dispoziție o listă bogată de motive: spam, activități sexuale, limbaj neadecvat, violență, informații false și altele. Administratorul primește notificare împreună cu motivul și utilizatorul care a raportat și este nevoit să decidă dacă va șterge anunțul. În situații excepționale, administratorul are posibilitatea de a bloca sau șterge un alt cont, utilizatorul în cauză fiind notificat prin email.

Sistemul de filtrare atât pentru animale, cât și pentru servicii, conține toate filtrele relevante pentru o navigare ușoară. Filtrele pentru locație, culoare și rasă sunt preluate de la toate anunțurile existente în baza de date, nederutând utilizatorul cu opțiuni în plus care nu au niciun rezultat. De asemenea, bara de căutare a animalelor permite căutarea după orice caracteristică sau cuvânt cheie aflat în profilul acestora.

1.6. Structura lucrării

Prin prezenta lucrare sunt analizate utilitatea, implementarea și funcționalitatea aplicației „Adoptă un prieten”. În capitolele și subcapitolele ce urmează vor fi atinse punctele cheie pentru conturarea unei imagini complete asupra produsului finit și a scopului acestuia.

Modul de utilizare al tehnologiilor alese este prezentat în **capitolul 2**. Structurat pe subcapitole dedicate fiecărei tehnologii, acest capitol descrie caracteristicile care sunt implementate folosind particularitățile limbajelor respective, astfel motivând alegerea făcută în privința acestora.

Partea de client a fost implementată folosind framework-ul Flutter, deoarece oferă dezvoltatorilor *entry-level* o calitate crescută privind UI și UX, prin comparație cu alte opțiuni dedicate. În plus, structura bazată doar pe crearea obiectelor reutilizabile facilitează întregul proces de mentenanță. Astfel, necesitatea constantă a aplicațiilor Android, de a primi actualizări, nu mai reprezintă o problemă.

Serverul a fost construit folosind tehnologia Spring Boot, un instrument care aduce un plus major aplicației prin maleabilitate, management simplist și cod autogenerat și autoconfigurat. Ca

mediu de dezvoltare este ideal pentru backend-ul unei aplicații mobile, deoarece are o arhitectură foarte ușor de întreținut și nu necesită servicii externe adiționale.

Tehnologiile suplimentare celor două anterior menționate sunt folosite pentru întregirea funcționalității și aducerea unui plus de complexitate aplicației. Limbajul Python a fost utilizat în elaborarea algoritmul de matching, acesta fiind implementat folosind Support Vector Machine, deoarece oferă posibilitatea creării eficiente a unui model de machine learning care poate fi salvat și utilizat ulterior pentru predicție. De asemenea, configurațiile necesare bunei funcționări a aplicației pe dispozitivele Android a fost realizată folosind structuri specifice dezvoltării unui proiect pentru acest sistem de operare: proprietăți de compatibilitate, permisiuni de acces, configurații de start și filtre.

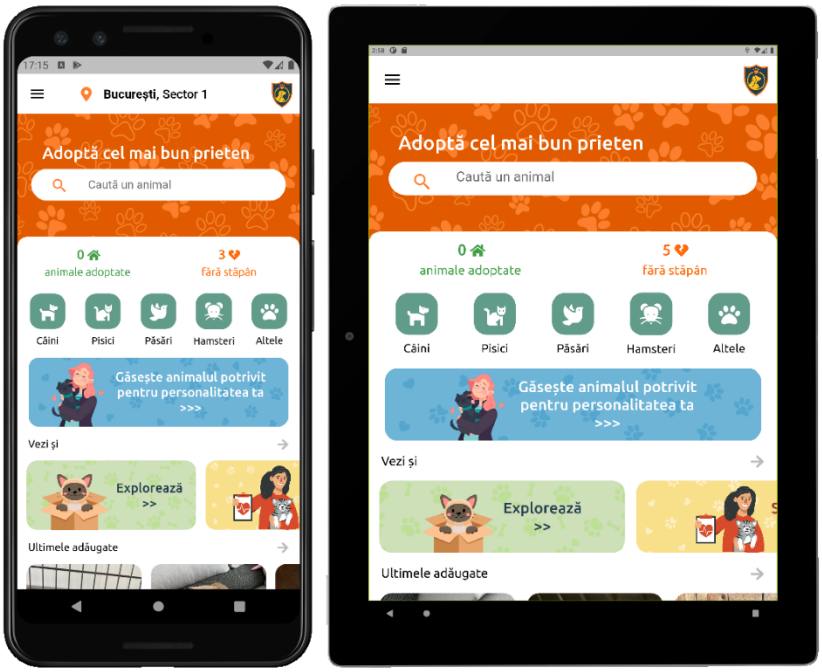

Figura 1.1. - Pagina principală „Adoptă un prieten”

Capitolul 3 descrie detaliat arhitectura aplicației, fiecare subcapitol fiind însoțit și de scheme sugestive de funcționare sau implementare. Întreaga funcționalitate a clientului se învârte în jurul Splash Screen-ului, acesta având un rol dublu: este o interfață prietenoasă pentru utilizatori și un background de actualizare a datelor prin call-uri către server sau inițializări de variabile. De asemenea, în sistemul de sign-in și sign-up sunt utilizate deep link-urile, o altă parte importantă în desfășurarea flow-ului de funcționare a acestuia. În ceea ce privește backend-ul, acesta este realizat

sub forma unui REST API, cu operații specifice CRUD. În plus, baza de date este important de menționat și prezentat, fiind ușor de integrat și gestionat datorită tehnologiei Spring Boot.

În **capitolul 4** este prezentată întreaga funcționalitate a aplicației, din perspectiva unui utilizator obișnuit. Fiecărui subcapitol în parte îi este asociat un feature principal al aplicației, cele secundare fiind menționate și prezentate pe parcursul întregii lucrări. De asemenea, acestui capitol îi sunt atașate imagini cu secvențe din aplicația propriu-zisă, pentru o ilustrare și o analiză mai relevantă a produsului finit.

Ultimul capitol este dedicat concluziilor și perspectivelor de dezvoltare ale aplicației. Este prezentată experiența de dezvoltare, impactul avut asupra utilizatorilor și scopul final al lucrării. De asemenea, un număr semnificativ de perspective de dezvoltare este menționat, atât pe plan intern, cât și pe plan extern.

Capitolul 2 - UTILIZAREA TEHNOLOGIILOR ALESE

2.1. Flutter Framework

Flutter este un framework open-source, cross-platform, creat de Google și publicat în anul 2016. Fiind un software development kit (SDK) a fost construit pentru a aduce un sistem nou, diferit de celelalte deja existente dedicate dezvoltării UI pentru mobile. Astfel, își propune să ofere o funcționalitate separată față de web view-urile clasice sau de obiectele OEM pe care alte limbaje le utilizează.

Dezvoltat tot de echipa celor de la Google și publicat în anul 2011, limbajul Dart se află la baza dezvoltării Flutter. Acesta folosește sistemul de compilare *ahead of time,* dar poate fi compilat și cu un compilator de tip *just in time*. La origini, acesta a fost conceput să devină un viitor înlocuitor al limbajului JavaScript. Spre deosebire de acesta, Dart aduce îmbunătățiri semnificative pe partea de optimizare, permițând astfel cicluri de dezvoltare mult mai rapide.

2.1.1. Caracteristici definitorii

Flutter are 3 mari caracteristici definitorii: *hot reload,* widgets și layout.

Procesul de *hot reload* presupune modificări aduse codului și actualizări ale acestora în timp real, pe dispozitivul de lucru. Acesta reprezintă un mare plus, mai ales în dezvoltarea aplicațiilor mobile, unde o reîmprospătare a modificărilor făcute în timp real nu este atât de accesibilă precum în dezvoltarea aplicațiilor web obișnuite.

Widget-urile reprezintă componenta prin care aplicația interacționează direct cu utilizatorul. Acestea nu au doar rol estetic, ci răspund și la acțiunile user-ului. Spre deosebire de alte limbaje, Flutter se bazează pe crearea și personalizarea propriilor widget-uri, având control deplin asupra comportamentului acestora. Există două tipuri majore din care obiectele create ulterior moștenesc anumite caracteristici: stateless și stateful widgets. Diferența majoră dintre aceasta o reprezintă flexibilitatea privind schimbarea stării și a însușirilor obiectului creat. Stateless widgets sunt obiecte de sine stătătoare, care nu pot fi schimbate pe parcursul interacțiunii utilizatorului cu aplicația o dată ce au fost adăugate în *widget tree,* în timp ce stateful widgets au un comportament contrar, având atașată o componentă specifică de stare. Practic, împărțirea pe

obiecte a framework-ului permite reutilizarea și personalizarea în funcție de context. În figura 2.1. este ilustrat codul unui stateful widget, creat sub forma unui șablon care va fi folosit pentru generarea mai multor tipuri de ecrane asemănătoare ca implementare, dar cu comportament diferit, în funcție de nevoi și de context. Acesta este folosit în crearea ecranelor care listează animalele pentru fiecare categorie în parte: câini, pisici, păsări, hamsteri, altele, găsit, pierdut sau toate, datele fiind colectate în funcție de valoarea string-ului *category*. Starea acestuia poate fi schimbată prin aplicarea filtrelor din pop-ul de filtrate atașat.

```
class TemplatePetsScreen extends StatefulWidget
{
  ...
  @override
  TemplatePetsScreenState createState() =>
      TemplatePetsScreenState(this.category);
}

class TemplatePetsScreenState extends
State<TemplatePetsScreen> {
  ...
  TemplatePetsScreenState(this.category);

@override
  initState() {...}
  Future<void> callAPIs() async {...}
  Future<void> getLocations() async {...}
  Future<void> getColors() async {...}
  Future<void> getBreeds() async {...}
  void getPets() {...}
  List<PetInfo> filteringAndSorting() {...}
  Future<void> refreshAfterPopup(BuildContext
context) async {..}
```

```
Future<void> refreshAfterPopup(BuildContext
context) async {...}

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      key: _scaffoldKey,
      backgroundColor: this.backgroudnColor,
      extendBodyBehindAppBar: true,
      drawer: MenuDrawer(),
      body: Stack(
        children: [
          CoverImage(...),
          SingleChildScrollView(
            scrollDirection: Axis.vertical,
            child: Column(
              children: [
                CustomAppBar(...),
                CoverTitle(...),
                BackgroundContent(...)
              ],
     ....  }

}
```

Figura 2.1. - Conținut Template Screen

Layout-ul reprezintă aranjarea widget-urilor pe ecran. Poziționarea și dimensionarea acestora este făcută prin intermediul altor obiecte. Practic, acest fapt susține la rândul său revalorificarea codului fără duplicarea lui. În figura 2.2. este ilustrată folosirea *Center* pentru centrarea widget-ului *Text*, care afișează un mesaj de eroare la validarea testului de personalitate. De asemenea, organizarea obiectelor se face pe baza combinației dintre rânduri, coloane și stive. Deși aceste 3 componente sunt predominante, Flutter oferă și anumite widget-uri cu proprietăți și funcționalități speciale precum: *Form, AppBar, Scaffold, Container* sau *SingleChildScrollView*. Un

exemplu care evidențiază foarte bine lucrul cu linii și coloane este proiectarea paginii de principale (figura 4.6.).

```
this.validationMessage == true
              ? Center(
                  child: Text(
                    "Toate câmpurile marcate cu * sunt obligatorii!",
                    style: TextStyle(
                      fontFamily: fontRegular,
                      color: redLogo,
                      fontSize: 13.0,
                    ),
                  ),
                )
              : Container(),
```

Figura 2.2. - Layout mesaj de validare

2.1.2. Plugins

Pe lângă partea de UI, în aplicațiile mobile este foarte important și aspectul privind stocarea locală, deoarece utilizatorii își doresc mereu o aplicație care le menține datele în siguranță. Flutter oferă posibilitatea stocării locale pe device, folosind criptare de tip AES cu ajutorul plugin-ul *flutter_secure_storage*. Acesta poate salva doar tipuri de date primitive (string, int sau double) sub forma unui dicționar cheie - valoare. Aplicația folosește stocarea dispozitivului pentru salvarea informațiilor despre utilizatorul curent în procesele de onboarding, autentificare și deep linking.

```
serverURL = http://192.168.0.187:8080/mobile-adoption-app

// ----------- AUTH --------------
signUpAPI = $serverURL/auth/register
loginAPI = $serverURL/auth/login
requestForgotPasswordAPI = $serverURL/auth/forgot-password/
//{email}
updatePasswordAPI = $serverURL/auth/update/
//{password}/{email}
resendEmailAPI = $serverURL/auth/resend/
//{email}

// ----------- USER --------------
testUserAPI = $serverURL/users/valid/
// {userId} (ADMIN/USER)
getUserDetailsAPI = $serverURL/users/details/
//{userId}
```

Figura 2.3. - O parte din conținutul fișierului .env

Plugin-ul *flutter_dotenv* oferă posibilitatea configurării la runtime a aplicației. În „Adoptă un prieten”, fișierul *.env* stochează toate variabilele de sistem care conțin rutele de apel către backend. De asemenea, o astfel de configurație poate aduce un plus securității în cazul în care aplicația devine un proiect open-source, putând fi exclus din repository-ul public.

Un alt plugin foarte important prin care Flutter facilitează proiectarea aplicațiilor pentru toate tipurile de ecrane este *responsive_framework*. Acesta transformă un design conceput pentru dimensiunile unui singur device, într-unul responsive în funcție de dimensiunile telefonului/tabletei pe care este instalată aplicația. Dimensiunile sunt detectate automat astfel că, prin doar câteva linii de cod (figura 2.4.), acesta rezolvă clasica problemă a reutilizării parametrilor comuni și a codului pentru toate formatele de ecrane existente.

```
@override
  Widget build(BuildContext context) {
    return GetMaterialApp(
      builder: (context, child) => ResponsiveWrapper.builder(
        child,
        maxWidth: 1200,
        minWidth: 480,
        debugLog: true,
        defaultScale: true,
        breakpoints: [
          ResponsiveBreakpoint.autoScale(380,
              name: MOBILE), // small value => large objects
          ResponsiveBreakpoint.resize(240, name: TABLET),
        ],
      ),
      debugShowCheckedModeBanner: false,
      title: 'Adoptă un prieten',
      home: SplashScreen(),
    );
  }
```

Figura 2.4. - Cod pentru redimensionarea automată a componentelor de UI

2.2. Spring Boot

Spring Boot este un tool open-source, care optimizează proiectarea serviciilor bazate pe framework-ul Spring. Prima versiune a fost publicată în aprilie 2014, iar în prezent este menținut de compania Pivotal. Este cunoscut pentru timpul scurt de dezvoltare, autoconfigurarea componentelor, server-ele incorporate (Tomcat fiind utilizat de „Adoptă un prieten”) și plugin-uri

diverse pentru interacțiunea cu baza de date. Un alt lucru important de menționat este faptul că, aduce multe dependințe care pot fi interconectate la o aplicație Spring. Dintre acestea, *Spring Security* și *Spring Web Services* sunt utilizate în aplicație.

Spring este unul dintre cele mai utilizate framework-uri Java EE pentru construirea de aplicații, fiind publicat în anul 2003. A fost creat cu scopul de a aduce îmbunătățiri aplicațiilor Java, prin crearea unor modele mai complexe de programare bazate pe *dependency injection*. Astfel, două diferențe notabile între Spring Boot și Spring framework sunt dimensiunea și modularitatea codului.

Spring Boot este un instrument ideal pentru dezvoltarea unui backend web, indiferent de tipul clientului sau al platformei. Prin codul compact, autoconfigurare, plugin-uri diverse, comunicare facilă cu baza de date, flexibilitate, dar și prin integrarea cu Spring framework demonstrează faptul că, acesta reprezintă o alegere foarte bună pentru dezvoltarea de tip *entry-level* cu potențial de ridicare la următorul nivel de development.

2.2.1. Apache Maven - configurări

Apache Maven este un instrument de software management a cărui comportament este bazat pe *project object model* (POM), fiind folosit pentru construirea și gestionarea proiectelor Java. Prin intermediul lui se pot adăuga mult mai ușor dependințe noi și se pot gestiona fișierele de tip *.jar*. Astfel, integrarea lui în backend-ul „Adoptă un prieten” oferă posibilitatea configurării ușoare a dependințelor adăugate pe parcursul dezvoltării proiectului în fișierul *pom.xml*. Acesta conține informații despre pachete și dependințe pe care Maven le gestionează ulterior.

Fișierul *application.properties*, are la rândul său un rol important în configurarea proiectelor Spring Boot. Acesta este folosit pentru scrierea unor anumite proprietăți ale aplicației cum ar fi: credențialele pentru conexiunea la baza de date, la serviciul de email și valori necesare în configurarea JWT.

2.2.2. JSON Web Token

JSON Web Token (JWT) este o modalitate de schimb de informație între client și server care asigură securitatea, autentificarea și autorizarea între părțile implicate. JWT este codificat în base64 (UTF-8), având 3 componente care îl formează, separate prin punct și codificate independent: antet, *payload* și semnătură. În antet se află tipul algoritmului utilizat pentru criptare,

în timp ce *payload-ul* reține date despre entitatea la care se face referință, iar scopul semnăturii este acela de a valida token-ul. În ceea ce privește implementarea și utilitatea acestuia în „Adoptă un prieten”, JWT este folosit în procesul de autentificare al utilizatorului, incorporând informații precum email-ul și data de expirare a token-ului. Spring Boot facilitează generarea, validarea și folosirea JWT prin framework-ul *Spring Security*. Acesta oferă interfețe predefinite, care pot fi implementate și personalizate conform nevoilor proiectului.

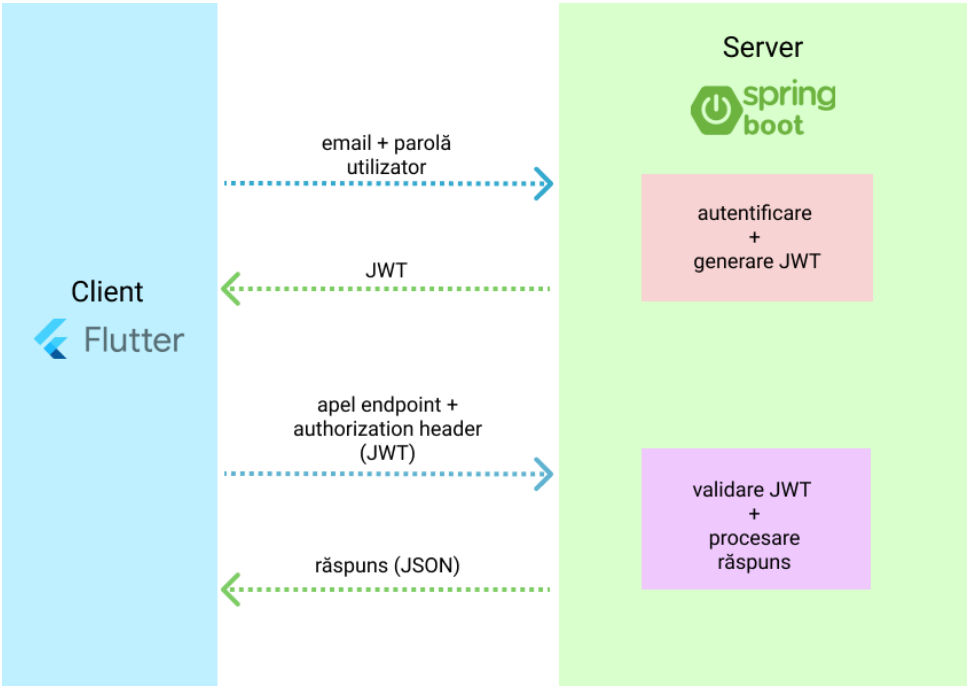

Figura 2.5. - Funcționarea JWT în aplicație

În figura 2.5. este evidențiat drumul pe care JWT îl parcurge în comunicarea dintre client și server. După ce un cont nou de user a fost creat folosind email-ul și parola, iar email-ul a fost confirmat cu succes, utilizatorul se poate conecta în aplicație. Credențialele de conectare sunt preluate printr-un JSON și trimise către backend, unde sunt preluate de o metodă de tip POST, dintr-un controller de autentificare. În această metodă are loc o verificare în prealabil a datelor și a statusului utilizatorului: să nu fie inexistent, blocat sau să nu aibă email-ul neconfirmat. Dacă totul este în regulă, încearcă autentificarea propriu-zisă, iar dacă credențialele sunt corecte, generează JWT-ul și-l atașează într-un JSON, trimis apoi clientului ca răspuns. Mai departe, aplicația adaugă JWT-ul la fiecare apel către rutele server-ului. În acest caz informațiile deținute de JWT sunt verificate, iar în cazul unei validări cererea este procesată, urmând returnarea unui răspuns sub

forma de JSON care va fi prelucrat ulterior, de către client. Astfel, un atacator care ghicește formatul unui endpoint nu poate avea acces la datele aplicației fără antetul de autorizare.

Practic, JWT reprezintă o cheie de deblocare a acțiunilor pe care un utilizator al aplicației le poate face în aceasta. Cum o aplicație mobile necesită un backend ușor de accesat, dar sigur, JWT reprezintă un pas necesar către proiectarea acestuia.

2.2.3. Trimitere email

„Adoptă un prieten” folosește trimiterea email-urilor în următoarele componente: confirmarea email-ului la crearea unui cont nou și a solicitării de schimbare a parolei, serviciul de suport, înștiințarea utilizatorului în cauză la blocarea și deblocarea contului. Procesul de trimitere este general valabil și presupune completarea unui fișier HTML predefinit cu toate informațiile necesare și trimiterea acestuia prin intermediul SMTP. În figura 2.6. este ilustrată trimiterea unui email către un utilizator care a fost blocat de către administratorul aplicației, folosind *JavaMailSender* și *MimeMessageHelper*. Ambele fac parte din API-ul *JavaMail* din Spring framework.

```
@PreAuthorize("hasRole('ADMIN')")
@PutMapping(path = "/block/{userId}", produces = { MediaType.APPLICATION_JSON_VALUE })
public OperationStatusModel blockUser(@PathVariable String userId) {
        DebugLog.saveInfo("Block a user: /users/block/{userId}", this.getClass().getSimpleName());
        ....
        try {
                MimeMessageHelper helper = new MimeMessageHelper(mail, true, "UTF-8");
                helper.setFrom("pet.adoption.app.2022@gmail.com", null);
                helper.setTo(userEntity.getEmail());
                helper.setSubject("Contul tău a fost blocat");
                helper.setText(body, true);

                File file = ResourceUtils.getFile("classpath:warning.png");
                ...
                helper.addInline("warning.png", imageSource, "image/png");

        } catch (Exception e) {
                DebugLog.saveInfo("Error from helper settings", this.getClass().getSimpleName());
                returnValue.setOperationResult(RequestOperationStatus.ERROR.name());
                return returnValue;
        }

        javaMailSender.send(mail);
        ....
}
```

Figura 2.6. - Trimiterea unui email de înștiințare utilizatorului care a fost blocat (preluat din [35])

Simple Mail Transfer Protocol (SMTP) este un protocol utilizat pentru trimiterea email-urilor de la un cont la altul. Protocoalele de email reprezintă seturi de reguli care permit diferitelor

conturi să facă schimb de informații cu ușurință, SMTP fiind și singurul protocol dedicat. Spring Boot oferă *spring-boot-starter-mail dependency* pentru implementarea gratuită, în proiect, a acestuia. În fișierul *application.properties* sunt definite proprietățile necesare configurării printre care: host-ul (aici, *smtp.gmail.com*), port-ul și credențialele de conectare.

2.3. Android

Odată cu dezvoltarea internetului și a tehnologiei mobile, a crescut exponențial numărul utilizatorilor de smartphone, acest fapt ducând la dezvoltarea accelerată a aplicațiilor cu specific. Principalul avantaj pe care acestea le oferă este capacitatea unică de a accesa un număr foarte mare de potențiali utilizatori, datorită versatilității pe care smartphone-urile le aduc în comparație cu laptop-urile sau desktop-urile. Deși este adesea considerat un *tip* de smartphone sau tabletă, Android este un sistem de operare, open-source, deținut și dezvoltat de Google. Astăzi, peste 2,5 miliarde de oameni din întreaga lume au un dispozitiv Android, pe care îl folosesc într-o varietate de moduri în viața de zi cu zi. Din dorința de a fi ușor accesibilă, de a ajunge la cât mai mulți utilizatori și de a crea o comunitate dedicată salvării animalelor de companie, aplicația „Adoptă un prieten” a fost proiectată pentru utilizarea pe dispozitivele Android.

```
<intent-filter android:autoVerify="true">
    <action android:name="android.intent.action.VIEW" />
    <category android:name="android.intent.category.DEFAULT" />
    <category android:name="android.intent.category.BROWSABLE" />
    <data
        android:scheme="https"
        android:host="mobile.adoption.app" />
</intent-filter>
```

Figura 2.7. - Intent filters pentru configurare deep links (preluat din [29])

Fișierul *AndroidManifest.xml* reprezintă baza de dezvoltare a oricărei aplicații Android, deoarece conferă informațiile esențiale pentru compilare, sistemul de operare și aplicațiile terțe. Acesta este localizat în proiectul clientului, într-un director cu același nume, special destinat tuturor configurărilor acestuia alături de alte fișiere importante cum ar fi: *build.gradle*, iconițele de lansare și *MainActivity.kt*. În fișierul manifest sunt declarate proprietăți privind activitățile, furnizorii de conținut, configurațiile dispozitivului pe care le poate gestiona, intent filters, cât și permisiunile de care are nevoie aplicația pentru a accesa părțile protejate ale sistemului. Astfel, sunt specificate

următoarele configurații: numele și versiunea aplicației, orientarea posibilă doar de tip portret, nivelul minim de versiune de Android (KitKat) pe care poate fi instalată, permisiuni privind locația, intent filters pentru deep linking și tipurile de ecrane suportate („Adoptă un prieten” suportă toate tipurile de ecrane ale dispozitivelor Android - figura 1.1.).

Prima interacțiune pe care un utilizator o are cu o aplicație mobile este pictograma de start (de obicei logo-ul), afișat pe pagina de descărcare a aplicației din Google Play. Image Asset Studio este un instrument oferit de Android Studio, prin care dezvoltatorii își pot crea pictograme personalizate pentru aplicațiile mobile. Pornind de la o singură imagine, acesta generează automat un set de pictograme pentru fiecare rezoluție posibilă în parte, pe care apoi le va aranja în foldere specifice. La runtime, se va alege iconița potrivită în funcție de caracteristicile dispozitivului pe care aplicația respectivă a fot instalată. În „Adoptă un prieten”, acest tool a fost folosit la generarea iconițelor de lansare, procesul de configurare al acestora fiind ilustrat în figura 2.8.

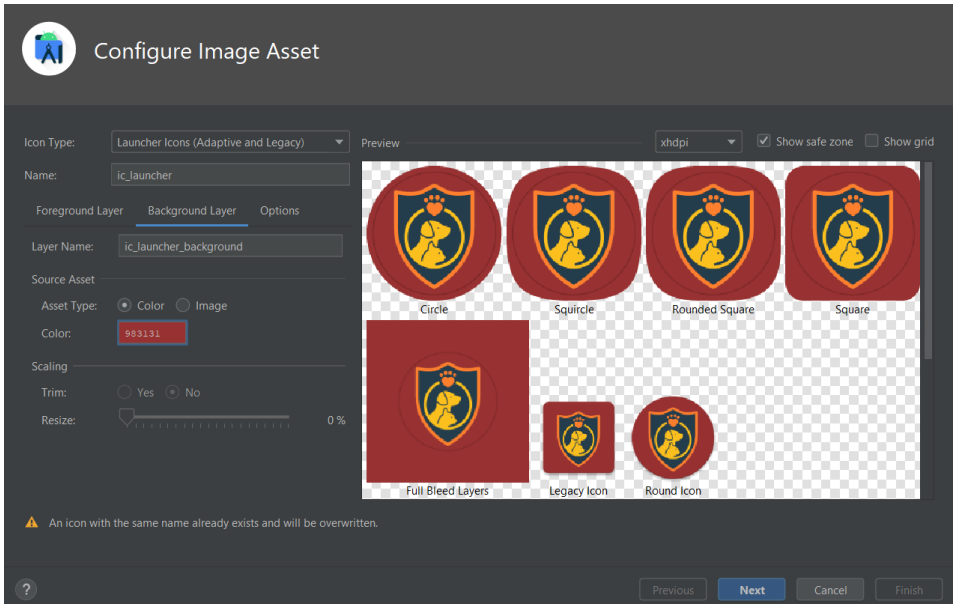

Figura 2.8. - Generare iconiță de lansare folosind Image Asset Studio

2.4. Python - Algoritm de matching

Python este un limbaj de programare interpretat, open-source și platform independent, dezvoltat în anul 1991 de Guido van Rossum. În ultimii ani a devenit popular mai ales în data

science și machine learning, acesta făcându-se remarcat în rândul dezvoltatorilor de software prin setul mare de biblioteci standard pe care le oferă.

Învățarea automată poate fi considerată un subset al inteligenței artificiale. În ultimele două decenii, aceasta a devenit un instrument comun în aproape orice sarcină care necesită extragerea și prelucrarea informațiilor din seturi de date. Algoritmul de matching din „Adoptă un prieten” care potrivește personalitatea unui utilizator cu o categorie de animale, presupune extragerea informațiilor dintr-un set mic de date. În consecință, proiectarea acestuia a fost făcută folosind limbajul Python, prin aplicarea unor concepte specifice de machine learning. Baza științifică, procesul de colectare al datelor și implementarea propriu-zisă, vor fi detaliate în subcapitole care urmează.

2.4.1. Baza științifică

„American Psychological Association” definește personalitatea ca fiind totalitatea diferențelor individuale în modelele caracteristice de gândire, simțire și comportament. Mai multe studii realizate de-a lungul timpului au arătat că anumite diferențe de personalitate sunt corelate cu iubirea față de o anumită categorie de animale. În lucrarea „The Correlation between Personality and Relationships with Pets” de Maia Paluska, autoarea analizează diferențele psihologice dintre iubitorii de câini și pisici, rezultatele arătând faptul că persoanele mai fericite și extrovertite preferă câinii, în timp ce introvertiții preferă pisicile. Aceeași ipoteză este susținută și în articolul „What does your pet say about your personality?” publicat de Lily Yuan pe blog-ul „Personality Psychology”. Acest articol studiază și corelația dintre personalitatea unui individ și alte categorii de animale: păsări, iepuri, broaște țestoase, șerpi, pești, hamsteri și cai.

Fiind concepută pentru utilizarea în România, aplicația „Adoptă un prieten” prezintă ca principale categorii de animale cele care se întâlnesc cel mai des în rândul stăpânilor din țară: câini, pisici, păsări și hamsteri. Categoria intitulată *altele* este menită sa cuprindă orice alte tipuri de animale de la pești și iepuri, până la reptile de orice fel. Așadar, în cele ce urmează, vor fi analizate toate categoriile de interes pentru populația țintă. În articolele anterior menționate sunt atribuite următoarele trăsături de personalitate în funcție de tipul categoriei în care se încadrează animalul ideal:

- iubitorul de câini: loial, extrovertit, energic, amabil, teamworker
- iubitorul de pisici: anxios, sensibil, open-minded, grijuliu

- iubitorul de păsări: extrovertit, optimist, independent
- iubitorul de hamsteri: *night owl*, inteligent, curios, conștiincios
- iubitorul de alte animale: conservativ, calm, stabil, creativ

În concluzie, anumite însușiri psihologice pot prezice, în marea majoritate a cazurilor, categoria de animale de companie potrivită pentru o anumita persoană. Bineînțeles, rezultatele pot să difere de la caz la caz datorită complexității minții umane, concluzia fiind raportată la analiza majorității.

2.4.2. Colectarea și prelucrarea datelor de antrenare

Datele de antrenare și testare au fost colectate folosind platforma Google Forms. Formularul a fost adresat tuturor categoriilor de vârstă, toate întrebările fiind cu răspuns scurt, deoarece ținta sa era un număr cât mai mare de respondenți. Acesta a fost deschis în perioada martie-mai 2022, fiind distribuit pe grupurile de studenți, elevilor de liceu, rudelor, prietenilor și părinților acestora. Marea majoritate a respondenților (79%) au sub 25 de ani, dar au fost înregistrate răspunsuri de la toate categoriile de vârstă. La închiderea acestuia a fost generat un fișier Excel, cu un total de 391 răspunsuri, un număr destul de mic pentru obținerea unei acurateți ridicate a modelului. De asemenea, la o primă analiză a răspunsurilor se remarcă și existența unui dezechilibru a datelor, care va afecta ulterior modelul creat pentru predicție. Acestea sunt împărțite după cum urmează:

- 225 (57,5%) respondenți preferă câinii
- 131 (33,5%) respondenți preferă pisicile
- 16 (4,1%) respondenți preferă păsările
- 9 (2,3%) respondenți preferă hamsterii/porcii de Guineea
- 10 (2,6%) respondenți preferă alte tipuri de animale

Inițial, formularul a fost conceput cu întrebări mai detaliate despre dimensiunea, culoarea și rasa animalului ideal pentru fiecare respondent. Însă, în România, animalele de rasă pură ajung să fie date spre adopție foarte rar, de aceea nu ar fi fost fezabilă predicția unor anumite caracteristici. Astfel, algoritmul a fost redus la predicția tipului de animal. Întrebările dedicate conturării profilului psihologic sunt concepute sub formă de răspunsuri pe o scară cu valori de la 1 la 5. Acestea măsoară nivelul de responsabilitate, empatie, sociabilitate, curiozitate, creativitate și energie, dar și frecvența episoadelor de stres, anxietate, schimbarea cercului de prieteni, cât și tipul de persoană care se consideră (lup singuratic sau team player). Toate aceste întrebări au fost

concepute pe baza caracteristicilor enumerate la subcapitolul anterior, sub forma unor modalități de diferențiere pentru conturarea personalităților de către algoritmul de AI. De asemenea, există și două întrebări cu scopul unei mai bune diferențieri și pe partea de nevoi a individului: modul de desfășurare al activităților zilnice (de acasă, fizic sau hibrid) și existența unei forme de venit cu care s-ar putea întreține animalul adoptat. În „Adoptă un prieten”, utilizatorul trebuie să răspundă la exact aceleași întrebări, o singură dată, algoritmul primind ca input răspunsurile și trimițând ca output, pe server, categoria potrivită.

```
def process_raw_data():
    x, y = [], []
    sheet_obj = xl.load_workbook("raw_data.xlsx").active

    # x (input data) : columns from 3 to 15
    for r in range(2, sheet_obj.max_row + 1):
        array = []
        for c in range(3, 16): # 6 - 14 are floats
                cell_obj = sheet_obj.cell(row = r, column = c)

                if c == 3:
                    array.append(sex[cell_obj.value])
                elif c == 4:
                    array.append(activity[cell_obj.value])
                elif c == 5:
                    ....

        x.append(tuple(array))

    # y (output data) : column no 16
    for r in range(2, sheet_obj.max_row + 1):
        cell_obj = sheet_obj.cell(row = r, column = 16)
        y.append(cell_obj.value)

    return x, y
```

Figura 2.9. - Prelucrarea răspunsurilor formularului în date de input și output

În figura 2.9. este ilustrată funcția *process_raw_data()* care conține procesul de prelucrare al datelor strict necesare antrenării și testării modelului pentru predicția unei categorii de animale. Cum un model de tip SVM are nevoie de niște date de input scalabile (int sau float), datele sub formă de string-uri, care sunt extrase la prelucrarea fișierului Excel, sunt transformate în anumite valori de tip întreg extrase din 4 dicționare. Din totalul obținut, acestea sunt împărțite după cum urmează: 80% date de antrenare și 20% date de test.

2.4.3. Implementare model

Potrivirea dintre personalitatea unui utilizator și un animal are loc între cele 5 categorii disponibile în aplicație: câini, pisici, păsări, hamsteri și altele. Aceasta este realizată de către un model de machine learning, care face preziceri pe baza trăsăturilor unui user. Fiind un algoritm de tip *proof of concept*, cu potențial de dezvoltare pe un set mare de date, a fost implementat folosind Linear Support Vector Classification (LinearSVM) din Scikit-learn.

Scikit-learn este o bibliotecă standard de Python, care expune o mare varietate de algoritmi de învățare automată. Aceasta folosește o interfață orientată spre sarcini, permițând astfel compararea ușoară a metodelor pentru o aplicație dată. Deoarece se bazează pe ecosistemul științific Python, poate fi ușor de integrat în orice tip de aplicații de analiză a datelor. În ceea ce privește structura, are un API simplu și intuitiv cu o interfață familiară, ușor de personalizat, fiind folosit și ca sursă de inspirație pentru multe alte biblioteci. Acesta include procesarea datelor, inginerie de caracteristici, estimatori de modelare, cât și un API pentru evaluarea modelelor antrenate folosind tehnici comune precum validarea încrucișată. În „Adoptă un prieten”, Scikit-learn este utilizată în *feature engineering* și pentru modelarea unui set mic de date.

```
def classifier(x_train, y_train, x_test, y_test):

    classifier = LinearSVC(C=0.01, dual=False)
    classifier.fit(x_train, y_train)
    print('model LinearSVC accuracy is:', classifier.score(x_test, y_test) * 100, '%')

    # confusion matrix
    y_pred = classifier.predict(x_test)
    ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
    plt.show()

    # save the classifier
    filename = 'model_1.sav'
    pickle.dump(classifier, open(filename, 'wb'))
```

Figura 2.10. - Clasificatorul utilizat în algoritmul de matching

Support Vector Machine (SVM) este un instrument de predicție pentru clasificare și regresie, care utilizează tehnici de învățare automată pentru a maximiza precizia unei predicții, evitând în același timp *data overfitting-ul*. De-a lungul timpului s-a constatat că, utilizarea SVM este una dintre cele mai potrivite alegeri în ceea ce privește rezolvarea problemelor de clasificare. În plus, unul dintre marile avantaje ale utilizării SVM este modul de lucru simplist, prin comparație

cu alte modele de clasificare cum sunt rețelele neuronale. Astfel, implementarea unui SVM a fost opțiunea ideală pentru proiectarea algoritmului de matching al „Adoptă un prieten”.

În figura 2.10. este ilustrată implementarea procesului de antrenare și testare al clasificatorului aplicației. LinearSVC utilizează *loss function* din biblioteca LIBLINEAR, care este diferită de forma sa implementată în mod obișnuit într-un *SVC(kernel = 'linear')* din biblioteca LIBSVM. Pentru seturi de date de dimensiuni mici, cum este și cazul de față, regularizarea termenului de interceptare poate duce la rezultate nepotrivite, astfel că LinearSVC este o alegere mai potrivită. Prin intermediul matricelor de confuzie, în figura 2.11. este ilustrată diferența pe datele de test privind tipul predicțiilor efectuate pentru fiecare SVM în parte. Deși diferența privind valoarea acurateții este foarte mică (54,43% - LinearSVC și 53,16% SVC) se poate observa faptul că, LinearSVC prezice corect și câteva animale din categoria pisici, în timp ce SVC prezice doar câini. Bineînțeles, acest fapt este datorat și dezechilibrului datelor de antrenare menționat la subcapitolul anterior, unde 57,5% dintre respondenți preferă câinii, iar restul procentelor sunt împărțite celorlalte categorii (33,5% pisici, 4,1% păsări, 2,3% hamsteri și 2,6% alte tipuri).

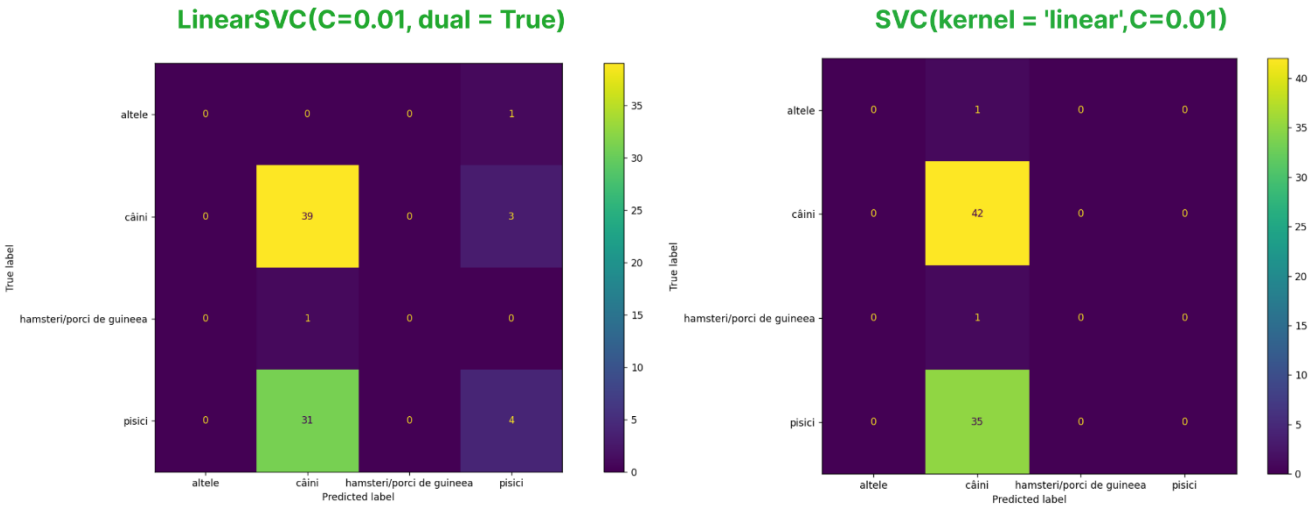

Figura 2.11. - Matrici de confuzie: LinearSVC vs SVC(kernel ='linear')

Capitolul 3 - ARHITECTURA SISTEMULUI

3.1. Backend

3.1.1. Rest API

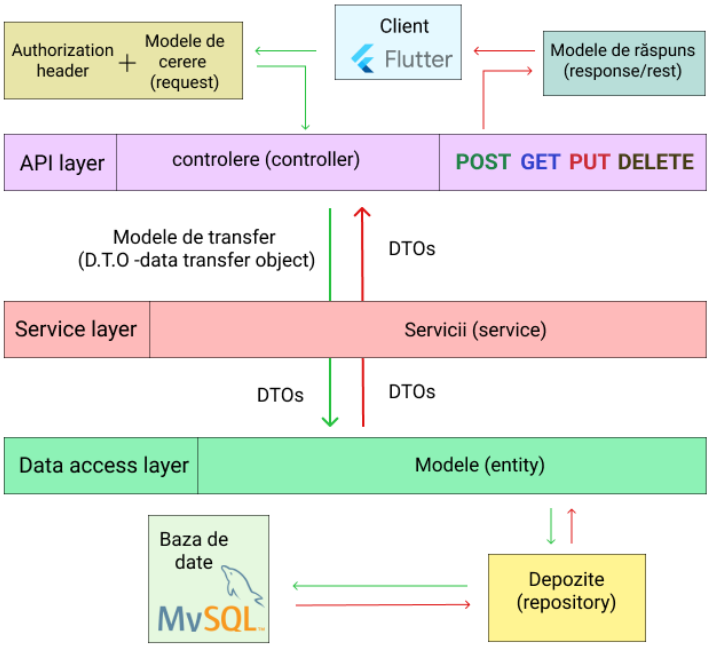

Figura 3.1. - Proces de funcționare server

REST este un stil arhitectural pentru proiectarea serviciilor web care poate comunica cu clienți diferiți prin comenzi simple HTTP. În prezent, această abordare este adoptată de marea majoritate a dezvoltatorilor de backend în aplicațiile web robuste și sigure, deoarece vine cu următoarele avantaje: procese specifice CRUD, *caching*, capacități de redirecționare și securitate ridicată. Acesta optimizează dezvoltarea unui REST API prin interconectarea mai multor componente: Java Persistence API, controlere și depozite.

Dezvoltarea unei aplicații mobile reprezintă un prim pas către dezvoltarea unei aplicații cross-platform, bazată pe o arhitectură cu caracter universal, ușor de dezvoltat și de întreținut. Astfel, Spring Boot este o alegere ideală pentru proiectarea backend-ului aplicației „Adoptă un prieten”. În figura 3.1. este ilustrată structura acestuia și modul în care componentele sale sunt

interconectate. Pentru o modularitate crescută a codului, depanare și dezvoltare ușoară, acesta este împărțit pe 3 nivele: gestionare de date, servicii și API-uri.

Cel mai îndepărtat este nivelul dedicat gestionării datelor. Utilizarea Spring Boot facilitează comunicarea cu baza de date prin intermediul interfețelor de tip *repository* care moștenesc CrudRepository, o interfață generică predefinită. Aceasta are anumite metode deja implementate cum ar fi: *findAll, delete, count* și *findById,* dar pot fi definite și unele noi, respectând tiparul de formare al numelui, fără a fi nevoie de implementare. De asemenea, fiecare tabel din baza de date are o clasă asociată *entity* care înglobează informațiile corespunzătoare fiecărui rând (entitate), fiind folosite de către metodele din repository-uri.

```
@Override
    public PetDto createPet(PetDto pet, String userId) {
        DebugLog.saveInfo("Create pet", this.getClass().getSimpleName());

        PetEntity petEntity = new PetEntity();
        ModelMapper modelMapper = new ModelMapper();
        petEntity = modelMapper.map(pet, PetEntity.class);

        if (userRepository.findByUserId(userId) == null) {
            throw new RuntimeException("User doesn't exist.");
        } else {
            petEntity.setOwnerDetails(userRepository.findByUserId(userId));
        }
        petEntity.setLastModified(new Date());

        PetEntity storedPetDetails = petRepository.save(petEntity);
        DebugLog.saveInfo("Pet was saved in the db succsefully", this.getClass().getSimpleName());

        PetDto returnValue = new PetDto();
        returnValue = modelMapper.map(storedPetDetails, PetDto.class);

        return returnValue;
    }
```

Figura 3.2. - Serviciul pentru crearea unui animal

Nivelul din mijloc conține servicii sub formă de clase și interfețe care lucrează cu fiecare clasă de tip *data transfer object* (DTO). Prin intermediului procedeelor de mapare are loc transformarea acestora în entități sau răspunsuri, în funcție de nivelul unde se va face trecerea. În figura 3.2. este ilustrată implementarea metodei *createPet* din clasa *PetServiceImpl* care primește un DTO cu informațiile unui nou animal, îl salvează în baza de date și întoarce informațiile complete (inclusiv id-ul din baza de date) către *controller*.

Controller-ele se ocupă de formarea rutelor și de filtrarea permisiunilor de acces. Practic, formează *endpoints* care vor fi apelate ulterior, din client. Comunicarea cu acesta este intermediată

de *Authorization header* și fișierele de tip JSON. Clientul trimite împreună cu JWT, un JSON care va fi prelucrat sub forma unor clase de *request*. Însă, procesul poate funcționa și în sens invers, clientul primind drept răspuns fișiere de tip JSON, prelucrate sub forma unor clase de *response*.

3.1.2. Baza de date

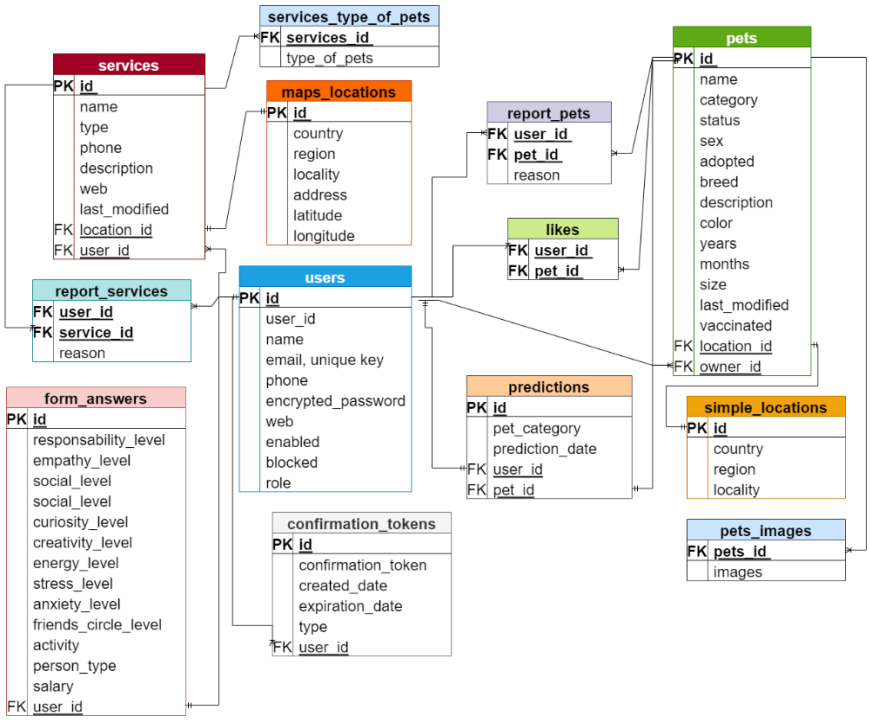

Figura 3.3. - Diagrama conceptuală a bazei de date

Utilizarea Spring Boot oferă și posibilitatea creării, în mod automat, a tabelelor rezultate din colecții. Un astfel de exemplu sunt tabelele *services_type_of_pets* și *pets_images*. Acestea stochează tipurile de animale asociate unui serviciu, respectiv link-urile de descărcare ale imaginilor salvate în cloud-ul Firebase. La declararea structurii entităților asociate tabelelor *services* și *pets*, aceste valori au fost reținute sub forma unor *liste* de string-uri. Spring Boot facilitează procesul și transformă automat listele în tabele asociate celor de pornire.

Aplicația „Adoptă un prieten” are 3 tipuri de utilizatori: vizitator, user și administrator. Pe server sunt implementate doar două dintre acestea: user-ul și administratorul, deoarece vizitatorul poate să acceseze informațiile, să folosească serviciul de suport, dar nu poate face nicio modificare care ar necesita interacțiunea cu baza de date. Fiecare metodă care formează o rută pentru aplicație poate avea ca filtru atașat și verificarea rolului pe care utilizatorul care face request-ul îl are. În figura 3.4. se află codul care formează endpoint-ul *http://localhost:8080/mobile-adoption-*

app/pets/reported/all, prin care doar admin-ul aplicației poate să vadă anunțurile cu animale care au fost raportate ca fiind nepotrivite. Rolul fiecărui utilizator este stocat în baza de date, în tabelul *users*, având ca valori posibile string-urile *ROLE_USER* sau *ROLE_ADMIN*. De asemenea, un plus de securitate îl aduce și stocarea criptată a parolelor utilizatorilor în baza de date. Implementarea acestor două feature-uri de securitate este facilitată de utilizarea framework-ului *Spring Security*.

```
    @PreAuthorize("hasRole('ADMIN')")
    @GetMapping(path = "/reported/all", produces = { MediaType.APPLICATION_JSON_VALUE })
    public ListRest<ReportRest> getAllReported() {
        DebugLog.saveInfo("Get all ids of reported pets: /pets/reported/all",
this.getClass().getSimpleName());
        ListRest<ReportRest> returnValue = new ListRest<ReportRest>();

        returnValue.setResponse(petService.getAllReportedPets());
        return returnValue;
    }
```

Figura 3.4. - Acordarea accesului doar administratorilor

3.2. Frontend

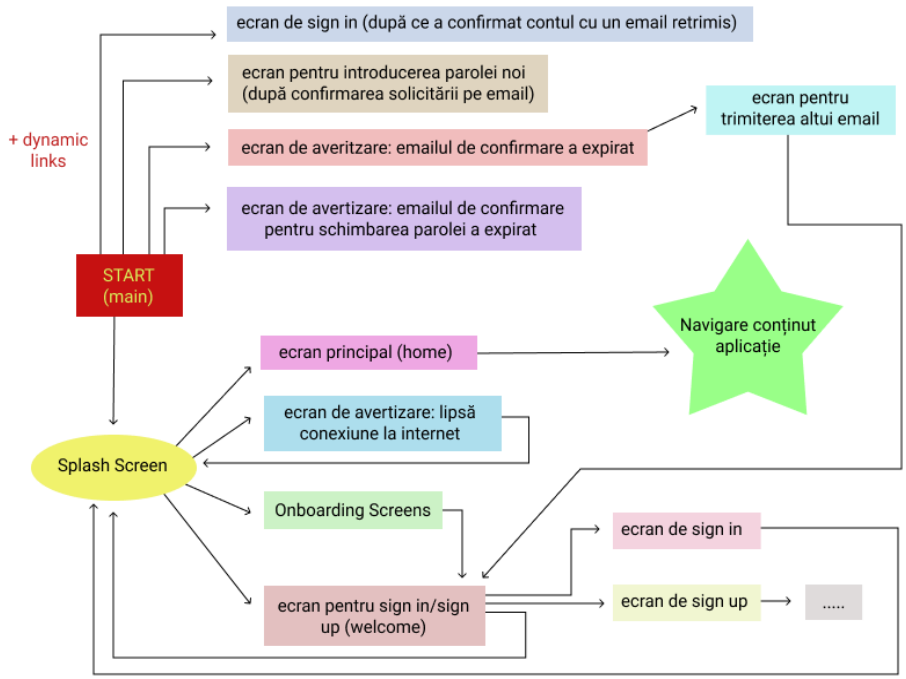

Figura 3.5. - Flux de funcționare client

3.2.1. Deep links

Firebase este o platformă pentru aplicațiile mobile și web. Printre numeroasele funcționalități gratuite pe care le oferă dezvoltatorilor, se numără și *Firebase Dynamic Links*. Acesta este un feature care permite gestionarea deep link-urilor și se ocupă de navigarea din mediul extern în orice parte a aplicației. În „Adoptă un prieten”, acestea sunt utilizate în sistemul de autentificare. Atunci când un cont este creat sau este solicitată schimbarea parolei, aplicația generează un link cu un token unic, care va fi trimis prin email pentru confirmarea cererii. La accesarea lui, utilizatorul este redirecționat, printr-un deep link configurat în Firebase Console, către o anumită parte din aplicație, indiferent dacă aceasta este deschisă sau nu în background. În figura 3.6. sunt descrise toate căile de acces prin intermediul deep link-urilor. În ceea ce privește implementarea, Flutter oferă un plugin dedicat denumit *firebase_dynamic_links*, care se ocupă de gestionarea acestora în aplicație. Implementarea este făcută în fișierul de start (*main.dart*) configurațiile necesare fiind realizate la fiecare pornire a aplicației.

```
Future<Widget> decideScreen(Uri deepLink) async {
  //Debug
  log(deepLink.toString());

  if (deepLink.toString() == dotenv.env['emailConfirmed']) {
    //Login is completed
    final storage = new FlutterSecureStorage();
    await storage.write(key: 'operationResult', value: "SUCCESS");
  } else if (deepLink.toString() == dotenv.env['resentEmailConfirmed']) {
    return LoginScreen();
  } else if (deepLink.toString() == dotenv.env['requestPassword']) {
    return UpdateForgotPasswordScreen();
  } else if (deepLink.toString() == dotenv.env['emailTokenExpired']) {
    return EmailTokenExpiredScreen();
  } else if (deepLink.toString() == dotenv.env['passwordTokenExpired']) {
    return PasswordTokenExpiredScreen();
  }

  return SplashScreen();
}
```

Figura 3.6. - Rutare deep links

3.2.2. Splash Screen

Splash Screen este primul ecran pe care utilizatorul îl vede atunci când deschide o aplicație mobile, fiind destinat încărcării datelor și configurațiilor necesare. Deși în marea majoritate a cazurilor este utilizat doar la deschiderea aplicației, în „Adoptă un prieten” este folosit și sub forma unui ecran de tranziție la înregistrare, conectare și deconectare, pentru încărcarea și prelucrarea datelor. În figura 3.7. este ilustrat Splash Screen-ul aplicației, acesta fiind atât un punct de start, cât și un punct intermediar de funcționare. Conținutul este simplu, având doar logo-ul atașat și un text scurt. Astfel, oferă o tranziție elegantă utilizatorilor, cât timp informațiile necesare sunt prelucrate în background-ul clientului. Flutter oferă *plugin-ul splash_screen_view* pentru implementarea unei astfel de componente.

Figura 3.7. - Splash Screen aplicație

Capitolul 4 - PREZENTAREA APLICAȚIEI

4.1. UI și UX

„Adoptă un prieten” își propune să fie un mediu cât mai prietenos pentru utilizatorii săi, de aceea interfața reprezintă cea mai importantă componentă în atingerea acestui obiectiv. Caracteristicile definitorii UI sunt: consistența, atractivitatea, claritatea și eficiența, în timp ce utilitatea, credibilitatea și accesibilitatea sunt trăsături definitorii pentru UX. Acestea sunt dobândite printr-o schemă de culori complementare, pop-up-uri și mesaje informative, *Onboarding Screens*, tranziții, iconițe și grafică sugestivă.

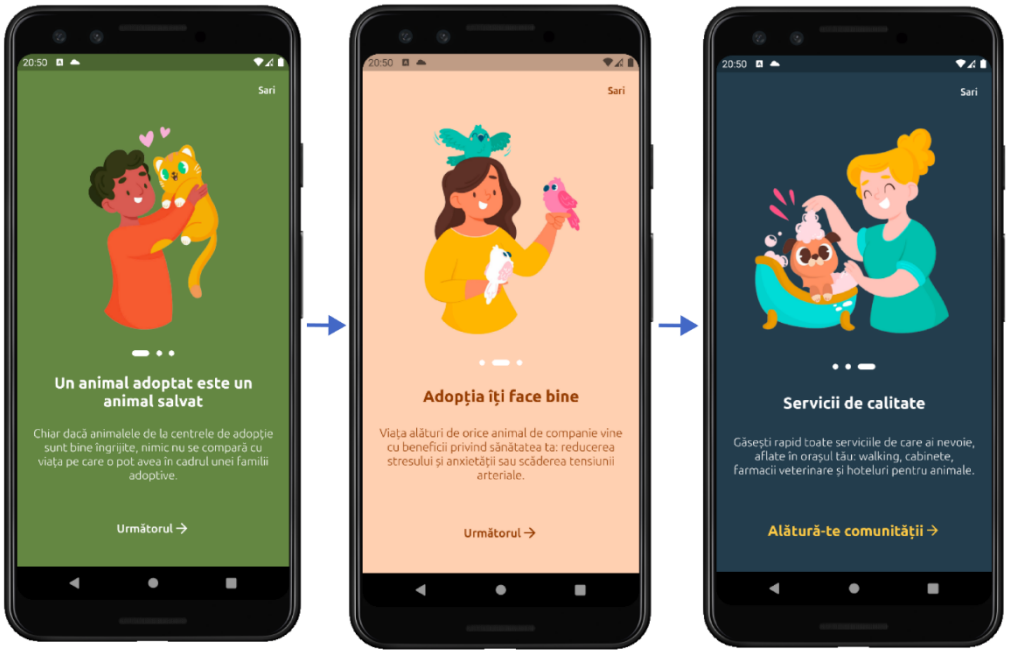

Figura 4.1. - Onboarding Screens

Design-ul aplicației se bazează pe o schemă formată din 5 culori: portocaliu, galben, 2 nuanțe de verde și albastru, la care se adaugă 7 variațiuni ale acestora, albul și negrul, utilizate în anumite puncte ale componentelor de frontend. Alegerea paletei nu este întâmplătoare, aceasta simbolizând culori regăsite predominant în natură, făcând trimitere la viața animalelor.

Figura 4.2. - Paleta de culori utilizată

Grafica a fost aleasă în ton cu tematica aplicației, imaginile fiind preluate de pe *Freepik.com* (https://www.freepik.com/), o platformă de resurse grafice, sub formă de vectori, gratuite sau licențiate contra cost. Vectorii au fost prelucrați folosind software-urile de editare foto Adobe Illustrator și Adobe Photoshop, extrăgându-se doar anumite componente cu background transparent în format *.png*. Logo-ul aplicației a fost realizat prin îmbinarea mai multor imagini de tip SVG, păstrând inclusiv schema de culori utilizată în aplicație și simbolizând ideea de protecție a animalelor.

Așadar, printr-un design intuitiv, viu colorat și grafică sugestivă, „Adoptă un prieten" creează o interfață primitoare, veselă și călduroasă pentru utilizatorii săi.

4.2. Sign in și sign up

„Adoptă un prieten" oferă utilizatorii nou-veniți posibilitatea navigării prin aplicație fără un cont înregistrat, ca vizitatori. Vizatorii pot doar să vizualizeze conținutul aplicației și să folosească serviciul de suport, dar nu vor putea publica anunțuri sau primi o predicție de la algoritmul de matching. În consecință, vor fi nevoiți să își creeze un cont pe baza email-ului și a unei parole dacă doresc să folosească aceste opțiuni. Fiecare cont are nevoie de confirmarea adresei de email, printr-un link unic, valabil doar 15 minute de la trimitere. Dacă link-ul nu a expirat, utilizatorul va fi

redirecționat prin intermediul unui deep link, din email direct către Splash Screen-ul aplicației pentru încărcarea datelor de autentificare și apoi, către pagina principală. În caz contrar, va fi redirecționat către un ecran de avertizare pentru expirarea link-ului de confirmare cu posibilitatea de retrimitere a unuia nou.

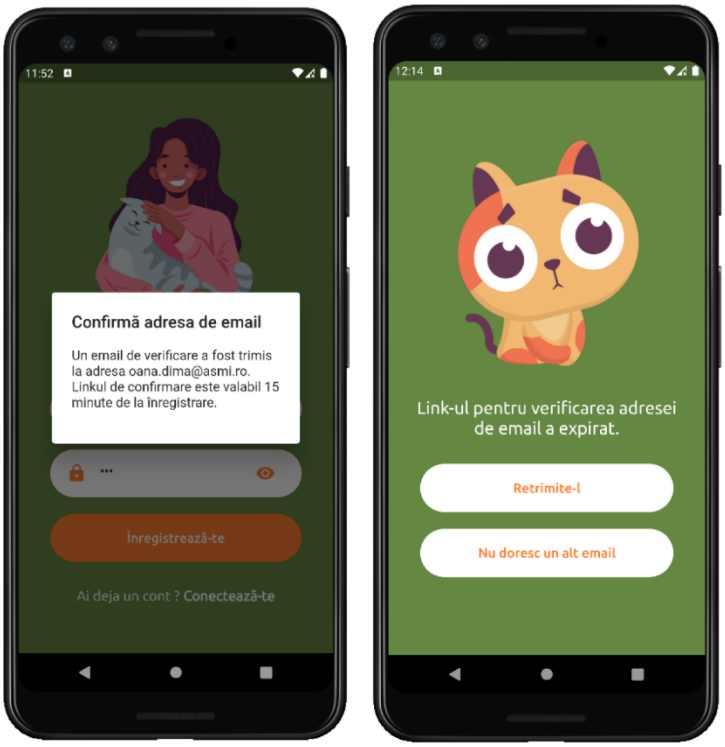

Figura 4.3. - Mesaje de informare pentru confirmarea adresei de email

Login-ul se face pe baza credențialelor folosite la crearea contului. Toate ecranele folosite în implementarea sistemului de sign-in și sign-up au atașate validări necesare câmpurilor de input, însoțite de mesaje specifice: formatul email-ului incorect, parola incorectă, email neconfirmat, user inexistent sau user blocat. Aplicația oferă și posibilitatea deconectării sau ștergerii definitive a contului, cea din urmă având loc cu afișarea unui mesaj de avertizare pentru ștergerea tuturor datelor din baza de date asociate contului.

Figura 4.4. - Prezentarea ecranelor pentru autentificare

4.3. Schimbarea parolei

Un sistem de conectare nu este complet fără posibilitatea schimbării parolei. Utilizatorul poate avea nevoie de această opțiune din motive diverse, cel mai des întâlnit fiind uitarea acesteia. Acesta a fost implementat printr-un flow separat de ecrane, dar cu o funcționalitate relativ asemănătoare cu cea a creării unui cont nou. După alegerea opțiunii, un ecran nou se va deschide unde utilizatorul va fi nevoit să introducă email-ul asociat contului respectiv, urmând sa îi fie trimis un link separat pentru confirmarea acestuia, oferind astfel și un plus de securitate contului. Prin intermediul deep link-urilor, utilizatorul va fi redirecționat din email direct în aplicație într-un ecran separat, unde va putea să introducă noua parolă. Ulterior va avea loc în background procesul de login, pentru generarea unui *Authorization header*, apoi o tranziție către pagina principală prin intermediul Splash Screen-ului.

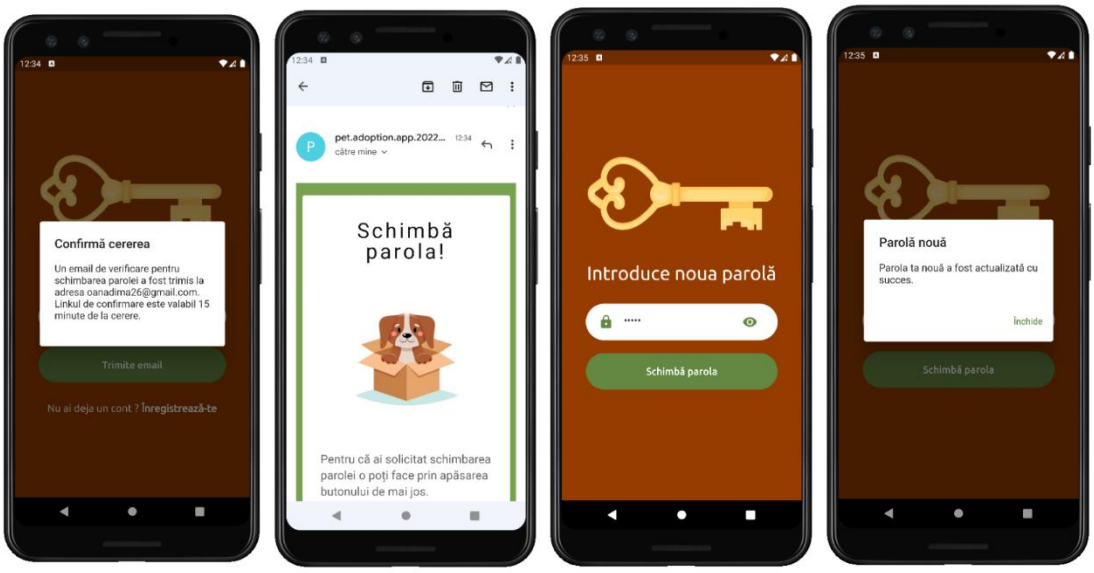

Figura 4.5. - Flow pentru schimbarea parolei

4.4. Pagina principală

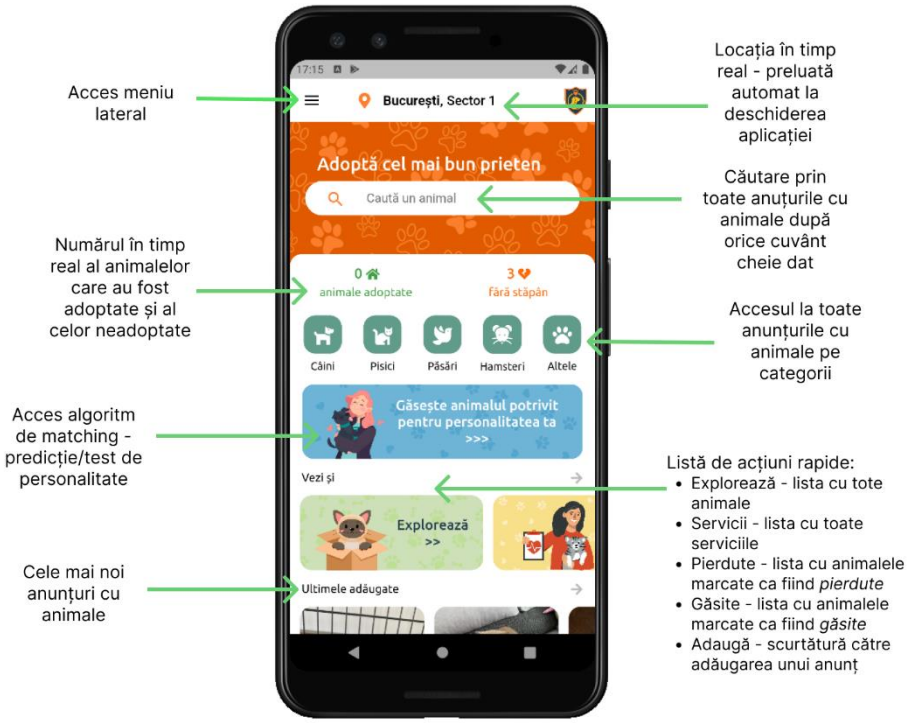

Figura 4.6. - Pagina principală

Pagina principală (home) are ca scop prezentarea opțiunilor principale pe care aplicația le oferă, fiind și un punct de start pentru navigarea prin conținutul acesteia. În figura 4.6. se poate observa cum pagina principală oferă scurtături către toate subiectele de interes: lista de servicii, algoritmul de matching, fiecare categorie de animale în parte și cele mai noi anunțuri publicate. Ecranul poate fi împărțit în 3 componente principale: bara aplicației, secțiunea de căutare și conținutul propriu-zis. Meniul de tip *hamburger* poate fi accesat din bara aplicației. Folosind bara de căutare poate fi găsit orice animal înregistrat pe baza oricărei caracteristici din profilul acestuia. În cazul în care căutarea nu are rezultat, va afișa automat conținutul obișnuit al paginii.

4.5. Publicare anunțuri

Un utilizator poate publica un anunț doar dacă datele sale de contact sunt complete: nume persoană/organizație/asociație, telefon și opțional site-ul web. În cazul lipsei acestora, atunci când dorește să urce anunțul în aplicație, va fi informat prin intermediul unui pop-up că este necesară completarea datelor în prealabil publicării acestuia. După publicare, proprietarul acestuia îi poate edita ulterior toate datele sau îl poate șterge definitiv din baza de date. Un utilizator înregistrat, care nu este proprietarul postării respective, poate să o raporteze pentru diverse motive prezentate anterior în lucrare, urmând ca administratorul să decidă ulterior dacă șterge anunțul.

4.5.1. Publicare anunțuri despre animale

Profilul unui animal este conceput să cuprindă toate detaliile necesare conturării unui anunț complet. Exceptând câmpurile pentru introducerea vârstei, toate au caracter obligatoriu, validarea formularului făcându-se la apăsarea butonului de trimitere și, dacă este cazul, cu afișarea mesajelor corespunzătoare. Nu există un număr limită de poze pe care un utilizator le poate încărca pentru un singur anunț, stocarea acestora făcându-se în Firebase Storage, fiecare utilizator având un director asociat. Datorită faptului că, aplicația este dedicată utilizării mobile, pozele necesită un aspect de 1:1, de forma unui pătrat. Un mesaj de avertizare în privința formatului este afișat în secțiunea dedicată încărcării pozelor. De asemenea, dacă un animal a fost adoptat, autorul anunțului poate alege din meniul de control al postării, dacă dorește să marcheze animalul ca fiind adoptat. Practic, acesta nu va mai apărea în aplicație, dar va rămâne înregistrat în baza de date și adăugat la numărătoarea animalelor care au fost adoptate din pagina de principală.

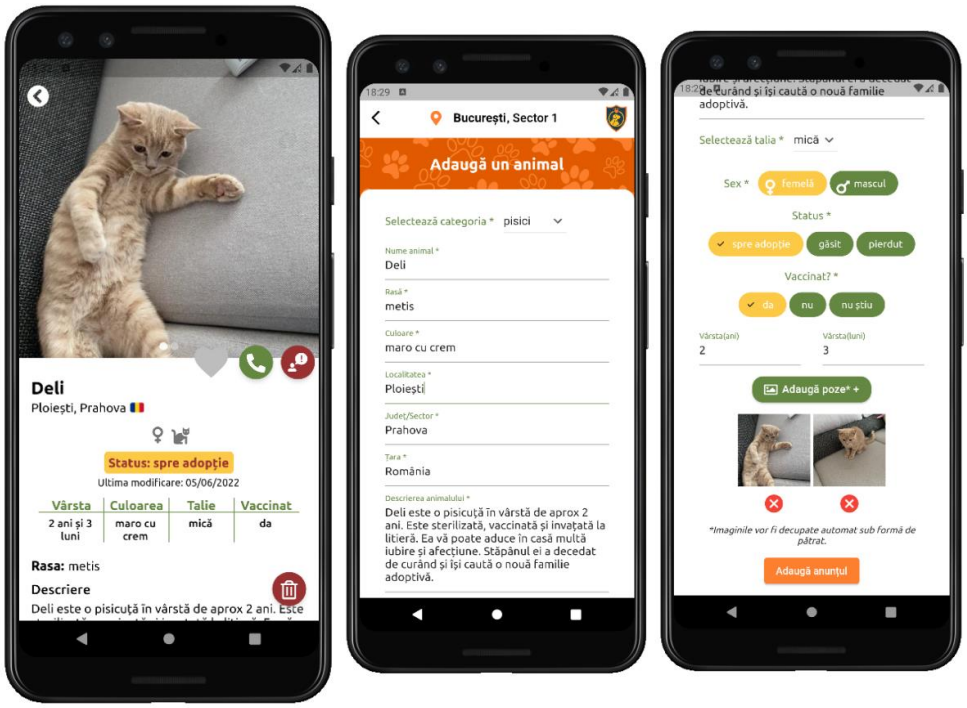

Figura 4.7. - Creare profil animal

4.5.2. Publicare anunțuri despre servicii

Sistemul de colectare a serviciilor este dedicat stăpânilor care dețin deja cel puțin un animal de companie și își doresc să găsească rapid și ușor, serviciile de care au nevoie pentru îngrijirea acestora. Lista de categorii dedicate atinge toate punctele esențiale pentru îngrijirea unui animal de companie, printre acestea numărându-se: veterinari, farmacii, saloane de înfrumusețare, pensiuni și servicii hoteliere.

Modalitatea de adăugare a unui anunț este asemănătoare cu cea a creării profilului unui animal însă, în locul pozelor, este introdusă alegerea adresei cu localizare pe hartă. Hărțile Mapbox au fost alese pentru integrarea acestui feature, deoarece oferă 100 000 de request-uri lunare, gratuite, ale API-ului de geocoding, spre deosebire de alte servicii populare de pe piață, care au devenit foarte costisitoare chiar și pentru development-ul *entry-level*. În client, implementarea hărților Mapbox se face prin intermediul unui plugin de Flutter numit *nominatim_location_picker*. Fiind open source, acesta a necesitat câteva modificări în codul pachetului pentru funcționarea optimă în cadrul „Adoptă un prieten”: afișarea detaliată a adresei, schimburi de stări, mesaje

informative și *widget mounting*.

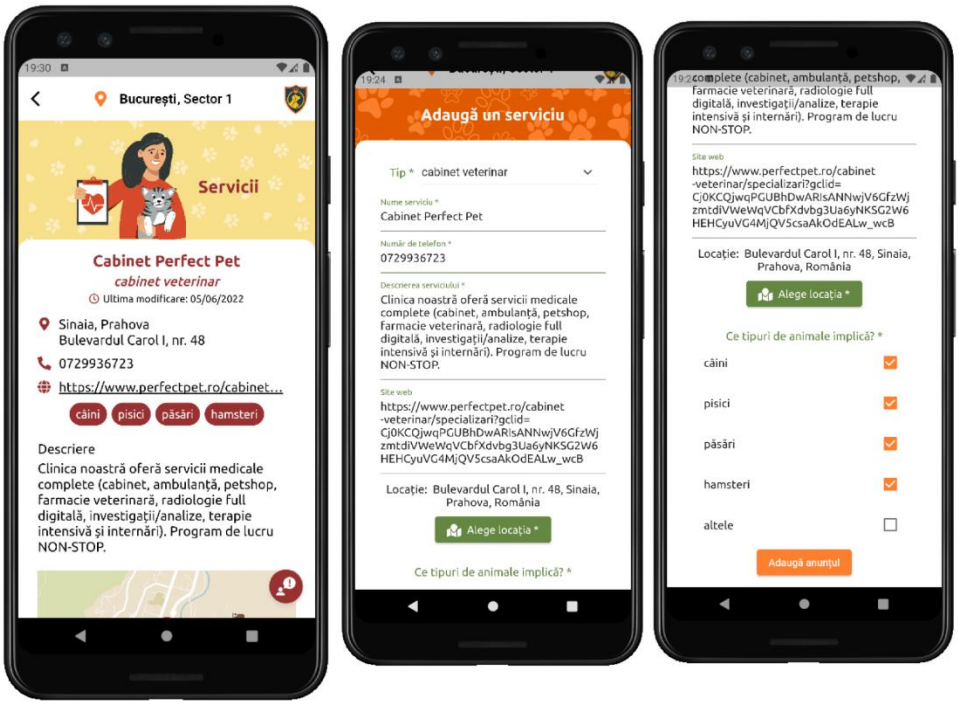

Figura 4.8. - Creare profil serviciu

4.6. Panou de control administrator

Administratorul își poate îndeplini toate atribuțiile din interfața aplicației, mai exact din panoul de control dedicat. Bineînțeles, acesta este accesibil din meniu doar utilizatorilor identificați cu rolul de admin. Structura panoului este împărțită în 4 secțiuni interschimbabile, după cum urmează:

- Secțiunea animalelor raportate
- Secțiunea serviciilor raportate
- Secțiunea utilizatorilor blocați
- Secțiunea tuturor utilizatorilor

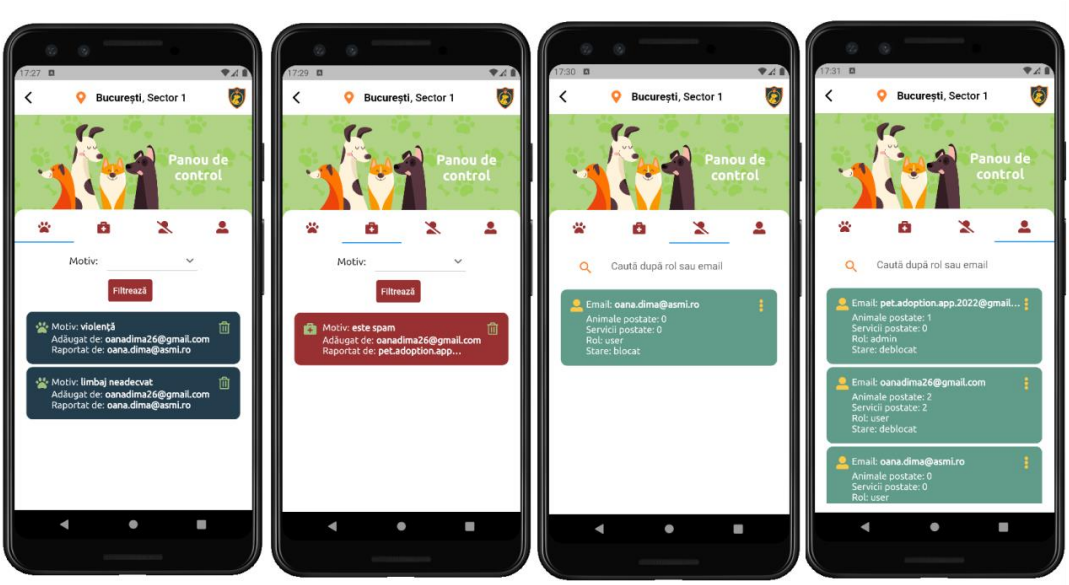

Figura 4.9. - Panou de control administrator

În ceea ce privește acțiunile pe care administratorul le poate face, acestea se împart în 2 mari categorii: acțiuni asupra anunțurilor și acțiuni asupra utilizatorilor. Admin-ul poate să filtreze anunțurile care au fost raportate după motivul raportului, să vizualizeze anunțul și să aleagă să șteargă definitiv postarea respectivă din baza de date. Ca acțiuni pe care le poate face asupra utilizatorilor, acesta îi poate căuta după rolul pe care îl au (user sau admin) sau după email, îi poate bloca sau debloca (un email de informare le va fi trimis automat în ambele cazuri), îi poate șterge, le poate vizualiza lista postărilor făcute în aplicație și îi poate numi administratori ai acesteia.

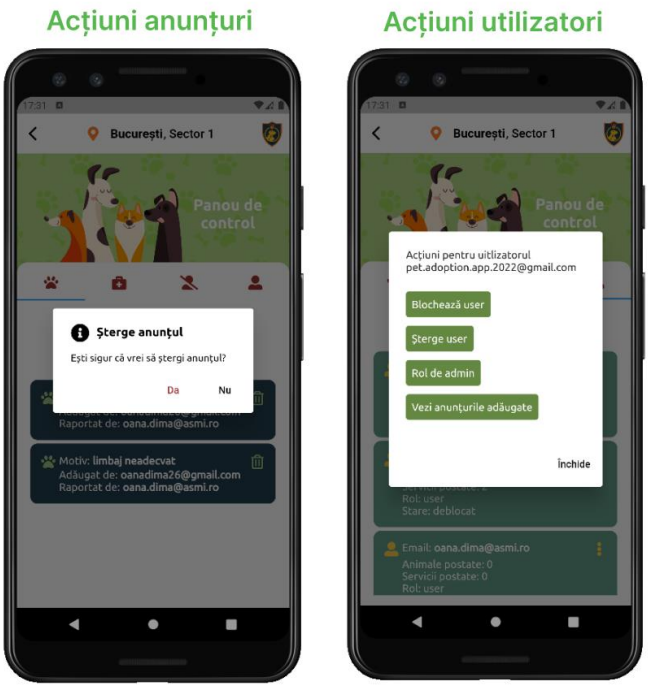

Figura 4.10. - Acțiuni administrator

În consecință, sistemul de raportare a postărilor, cât și controlul administratorului asupra întregii aplicații conferă utilizatorilor un mediu sigur, cald și prietenos. Astfel, aceștia au un motiv în plus să se alăture comunității create și să folosească serviciile oferite de „Adoptă un prieten”.

4.7. Serviciul de suport

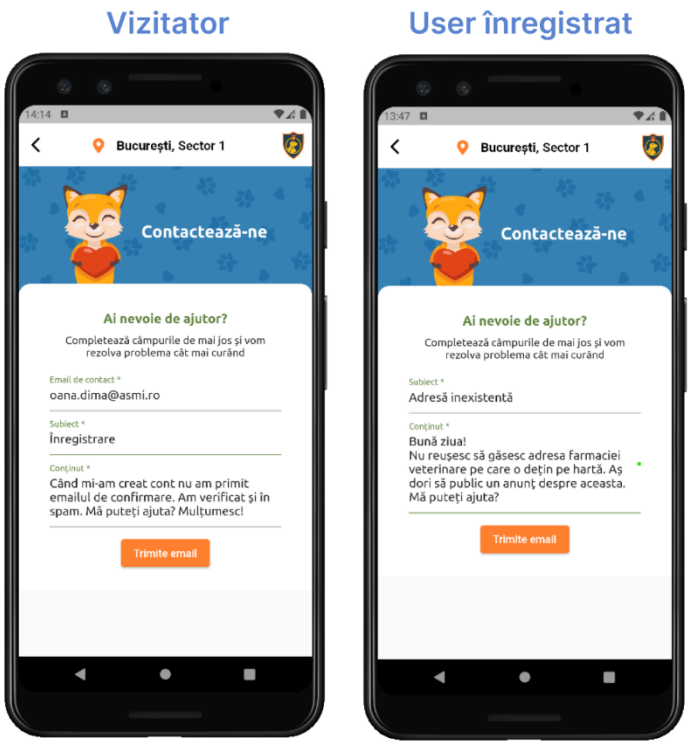

Figura 4.11. - Formular de contact pentru serviciul de suport

Serviciul de suport are un rol foarte important în dezvoltarea comunității de utilizatori ai unei aplicații. Producătorii trebuie să manifeste grijă, înțelegere și sprijin față de fiecare utilizator în parte pentru creșterea nivelului de satisfacție pe care aceștia îl manifestă privind produsul cu care interacționează. În plus, acesta reprezintă un canal deschis către o dezvoltare accelerată, prin feedback direct. Pentru o aplicație aflată la început de drum, serviciul de suport este necesar și reprezintă un plus adus încrederii comunității sale.

În „Adoptă un prieten”, serviciul de suport a fost proiectat să fie disponibil atât pentru utilizatorii înregistrați, cât și pentru vizitatori cu scopul de a le putea oferi ajutorul în cazul unor probleme la conectare sau înregistrare. Acesta funcționează pe baza serviciului de trimitere al email-urilor prezentat în capitolul 2. Fiind construit sub forma unei soluții complet gratuite, toate

informațiile aferente formularului sunt trimise de pe adresa administratorului către el însuși. În figura 4.12. este ilustrat conținutul unui astfel de email: adresa de email de contact, subiectul și descrierea problemei. Email-ul user-ului în cauză este preluat automat dacă acesta este conectat în aplicație sau este cerut explicit în formular (figura 4.11.), dacă are rolul *guest* asignat în aplicație. Pentru ambele tipuri, formularul conține alte două câmpuri obligatorii, subiectul și descrierea problemei. Utilizatorii „Adoptă un prieten” se pot folosi de acest feature mai ales în raportarea bug-urilor, a problemelor de conectare, a funcționalității deficitare și a anunțurilor deranjante. Astfel, aplicația oferă utilizatorilor ei susținere și protecție, prin asistența oferită de către serviciul de suport.

Figura 4.12. - Conținutul email-ului primit pentru serviciul de suport

4.8. Sistemul de salvare în favorite

Odată cu dezvoltarea aplicației și a numărului de utilizatori, găsirea animalului potrivit va fi îngreunată semnificativ. Creșterea numărului de anunțuri postate va face decizia unei persoane foarte dificilă atunci când dorește să adopte un animal, datorită multitudinii de opțiuni pe care le va avea la dispoziție.

În scopul eficientizării procesului de adopție, „Adoptă un prieten” oferă și posibilitatea salvării unui număr nelimitat de anunțuri cu animale din orice categorie. Asemănător *like-ului* de pe pagina de Facebook sau a inimioarei de pe Instagram, fiecare profil de animal are un buton sub forma unei inimi aflat la baza galeriei cu poze (figura 4.7.). Prin intermediul acestuia se poate face salvarea în lista de favorite. De asemenea, își schimbă culoarea în funcție de includerea în listă: gri dacă animalul nu este marcat ca favorit al utilizatorului curent și roșu, în caz contrar. Lista cu toate

favoritele poate fi accesată din meniu și este disponibilă doar utilizatorilor înregistrați, vizitatorilor fiindu-le cerută conectarea în aplicație. Anunțurile salvate sunt ordonate automat după data la care au fost publicate/modificate, cele mai recente fiind primele afișate. Tot de aici, se pot elimina din secțiune prin simpla apăsare a inimioarei roșii din chenarul pentru previzualizare, anunțurile dispărând imediat de pe ecran. În plus, navigarea ușoară printre acestea poate fi făcută prin intermediul sistemului de căutare, după orice cuvânt cheie sau informație conținută în profilul animalelor, fiind același cu cel de pe pagina principală.

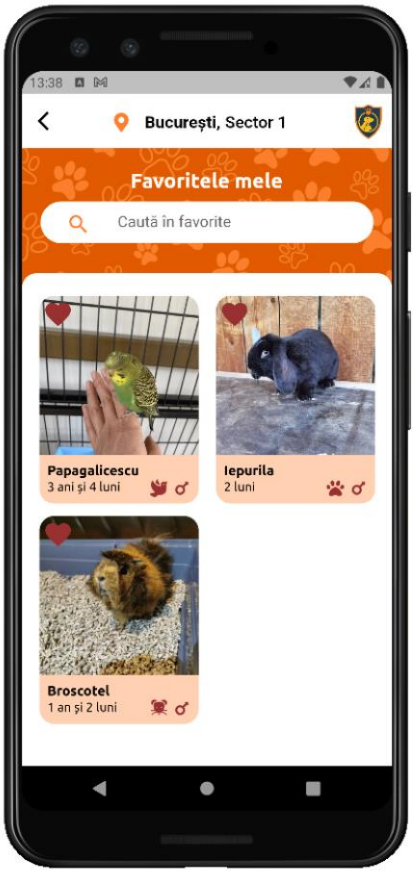

Figura 4.13. - Ecran cu favorite

4.9. Matching utilizator - animal

O opțiune complet nouă pe care „Adoptă un prieten” o aduce în plus față de celelalte aplicații de acest gen, este potrivirea dintre personalitatea unui utilizator și un animal dat spre adopție, ales din lista de anunțuri disponibile la acel moment în aplicație. Practic, utilizatorii

înregistrați au posibilitatea să răspundă la un test cu câteva întrebări menite să construiască un profil psihologic, pe baza căruia vor primi o sugestie. Aceasta este realizată de către un model de machine learning prezentat la capitolul 2, care prezice o categorie de animale potrivită psihologiei utilizatorului în cauză, în funcție de care se va alege un animal potrivit, fără stăpân. Predicția are loc prin alegerea aleatorie a unui animal cu statusul setat *spre adopție*, din categoria prezisă de algoritm. În situația excepțională, când în baza de date nu se află nicio entitate potrivită categoriei, se va alege aleatoriu un animal fără stăpân. În figura 4.14. este ilustrat procesul pe care user-ul îl parcurge pentru obținerea unei predicții: ecranul de informare în privința testului care urmează, testul propriu-zis și ecranul cu predicția, unde butonul *Adoptă acum* redirecționează utilizatorul în profilul animalului respectiv pentru vizualizarea detaliilor și a datele de contact ale proprietarului. Testul va fi completat o singură dată, datele rezultate fiind reținute în baza de date și asociate contului respectiv. La fiecare accesare a acestuia după completare, pagina de informare nu va mai fi afișată, ci va fi înlocuită cu predicția propriu-zisă.

În consecință, această opțiune contribuie semnificativ la creșterea numărul utilizatorilor implicați în utilizarea aplicației, a comunității formate pentru salvarea animalelor, cât și a numărului de animale adoptate prin intermediul acesteia.

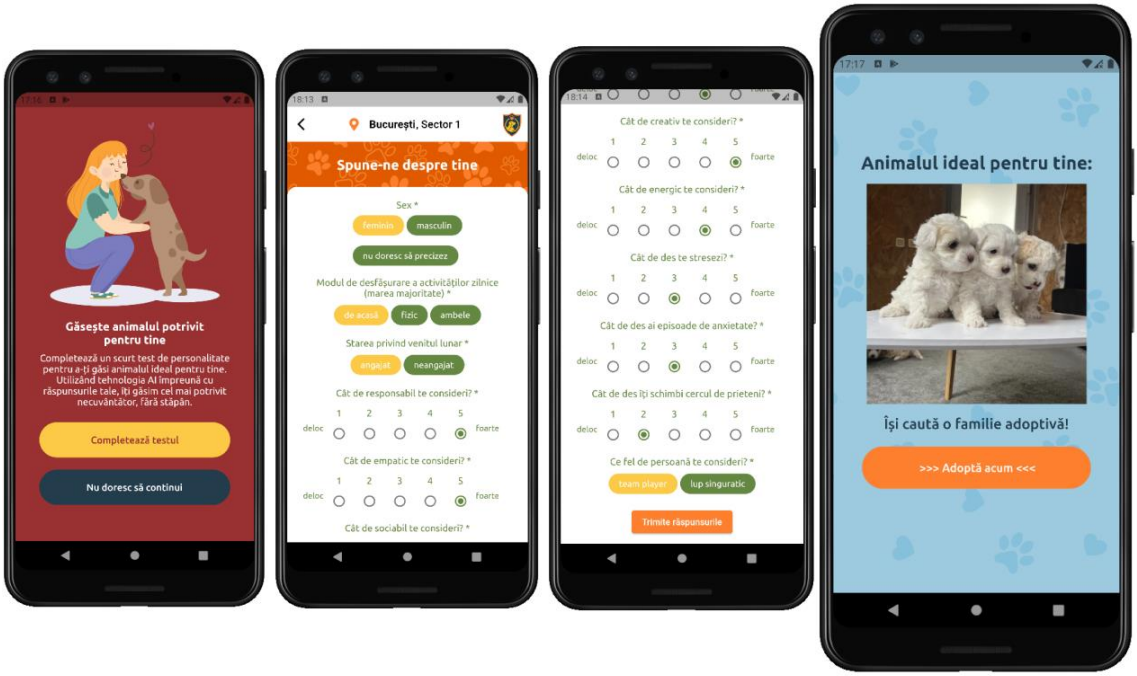

Figura 4.14. - Proces de matching utilizator - animal

4.10. Mesaje informative: ecrane speciale, validări și pop-ups

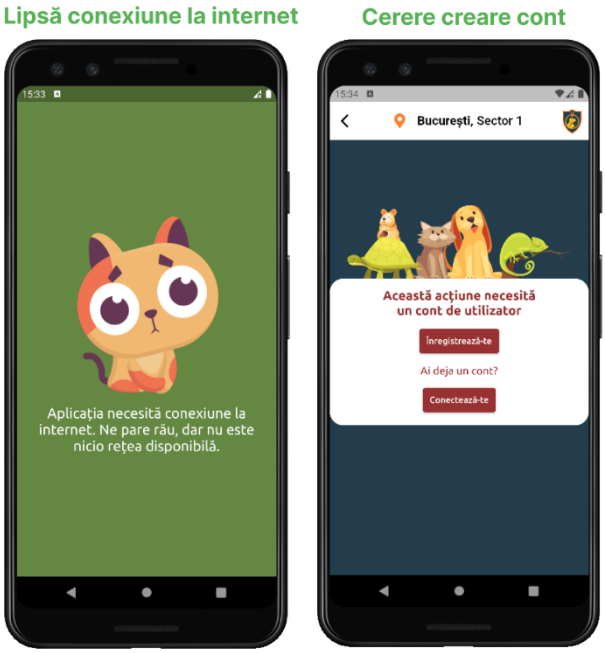

Figura 4.15. - Ecrane speciale de informare

Experiența utilizării aplicației nu poate fi completă fără mesajele informative pe care utilizatorul le primește la începerea/finalizarea anumitor acțiuni. Deși câteva astfel de mesaje au fost prezentate pe parcursul lucrării, este important de menționat ca acestea se află aproape în fiecare feature al acesteia. De asemenea, se remarcă 3 mari categorii în funcție de scopul și locul unde sunt folosite: ecrane speciale, validări și pop-ups.

Marea majoritate a ecranelor speciale sunt folosite în deschiderea aplicației, pentru avertizarea lipsei de conexiune la internet sau în procesul de confirmare a email-ului la crearea unui cont sau schimbarea parolei. În figura 4.15. se află ecranul afișat la deschiderea aplicației în lipsa conexiunii la rețea, acesta rămânând afișat într-un *loop infinit* care se va opri în momentul în care se restabilește o conexiune. În aceeași figură se află ilustrat un alt ecran special, întâlnit de utilizatorii cu rolul de vizitatori care vor să acceseze anumite acțiuni ale aplicației care necesită conectarea sau înregistrarea. De asemenea, menționat în subcapitolele anterioare, se află și ecranul de informare înainte de completarea testului de personalitate (figura 4.14.).

Validările sunt întâlnite în orice feature care presupune completarea unor câmpuri de formular și trimiterea informațiilor rezultate pentru a fi prelucrate sau salvate în baza de date. Acestea sunt folosite în următoarele componente: sign-in și sign-up, crearea unui anunț nou despre

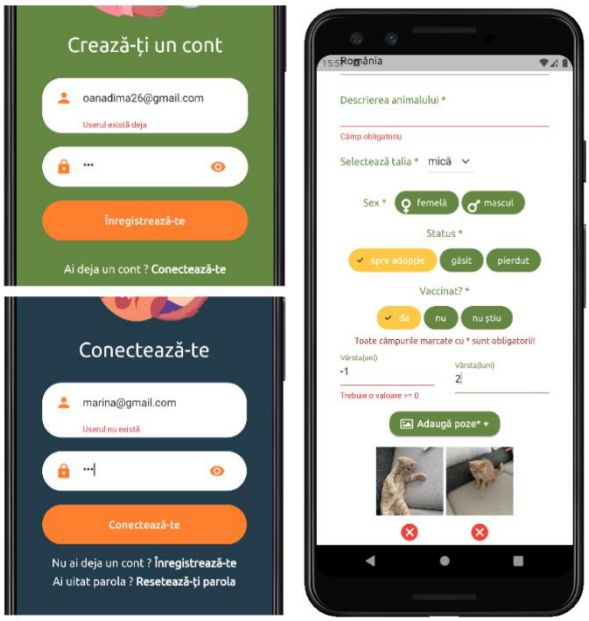

Figura 4.16. - Validări formulare

un animal sau un serviciu, formularul de suport și testul de personalitate. Mesajul nelipsit este cel care avertizează lipsa completării unui câmp cu caracter obligatoriu, dar mai pot fi întâlnite și mesaje precum: vârsta trebuie să fie mai mare sau egală cu 0, user-ul este blocat, email-ul sau parola sunt greșite, utilizatorul nu există.

Pop-up-urile sunt folosite atât ca mesaje de informare, cât și ca mesaje de avertizare. Sunt cel mai des întâlnite în acțiunile care presupun metode de tip POST, PUT, DELETE, care fac modificări asupra bazei de date sau în trimiterea email-urilor. În figura 4.17. se poate observa diversitatea acestora în funcție de situațiile în care sunt folosite, afișând mesaje de succes, eroare sau de informare.

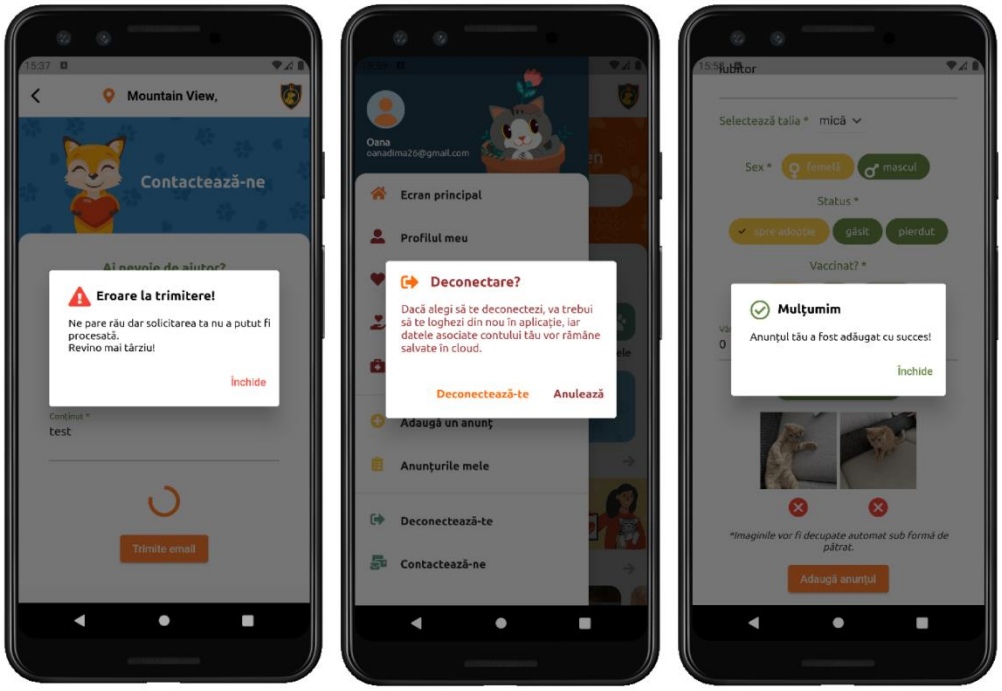

Figura 4.17. - Pop-up-uri de informare

Capitolul 5 - CONCLUZII ȘI PERPESCTIVE

5.1. Concluzii

Prin prezenta lucrare am expus conceptul, rolul cât și procesul de realizare al aplicației „Adoptă un prieten”. Destinată utilizării pe dispozitivele Android, aceasta își propune facilitarea procesului de adopție al animalelor de companie din România. Așa cum a fost detaliat anterior, aplicația oferă utilizatorilor posibilitatea postării și vizualizării anunțurilor despre animale în următoarele situații: trimiterea spre adopție, pierderea sau găsirea pe stradă. De asemenea, prezența algoritmului de matching utilizator - animal contribuie la încurajarea întregului proces de adopție. Un alt aspect care a fost larg dezbătut, a fost oferirea anunțurilor despre servicii dedicate celor care dețin un necuvântător, care pot fi centralizate și filtrate în funcție de anumite criterii.

Pe termen lung posibilitățile sunt multiple, principalul obiectiv al „Adoptă un prieten” fiind dezvoltarea unei comunități online cât mai implicată în îngrijirea și protejarea animalelor de companie. În lipsa unui cadru legal aspru, care să combată violența asupra animalelor si abandonul acestora, soluțiile software sunt un prim pas către combaterea unor astfel de acțiuni imorale.

Fiind prima aplicație căreia îi proiectez și implementez atât backend-ul, cât și frontend-ul de la zero, a reprezentat cea mai mare provocare de care am avut parte în ultimii 3 ani de studiu. Deși eram familiară cu limbajul Java, nu mai folosisem niciodată Spring Boot, iar de Flutter nici măcar nu auzisem până să încep sa fac cercetarea pentru lucrare. Astfel, în ultimele 8 luni, am învățat 2 tehnologii complet noi, foarte utilizate și cerute în industria IT din prezent. Provocările de care am avut parte pe parcursul drumului meu către finalizarea lucrării au fost numeroase, însă cu multă răbdare, perseverență și prin pasiunea pe care o am pentru tema aleasă, am reușit să depășesc orice impediment care mi-a ieșit în cale. Într-un viitor foarte apropiat, îmi doresc publicarea acesteia, deoarece este un domeniu către care majoritatea dezvoltatorilor de software din țară nu și-au îndreptat semnificativ atenția.

Așadar, aplicația „Adoptă un prieten” reunește cele 2 mari obiective pe care mi le-am propus încă de la prima linie de cod scrisă: să realizez un produs finit, care să aducă un mare plus lumii care ne înconjoară.

5.2. Perspective

În ceea ce privește perspectivele de dezvoltare, aplicația are numeroase posibile direcții de extindere:

- Baza de cercetare a algoritmului de matching poate fi extinsă spre colectarea mai multor răspunsuri și antrenarea pe baza acestora a unui model mult mai precis. În plus, sistemul de funcționare bazat pe acesta poate fi extins la noi campanii cum ar fi „Match of the day”, în condițiile creșterii semnificative a numărului de animale publicate în aplicație.
- Întreaga funcționalitate a aplicației poate fi adaptată utilizării la nivel internațional, prin introducerea traducerii în mai multe limbi, cu posibilitatea alegerii acestora la prima pornire după instalare.
- Poate fi implementat un sistem de chat, prin care cei care doresc să afle mai multe informații despre un animal sau despre un serviciu să aibă posibilitatea de a contacta proprietarul anunțului direct din aplicație.
- Sistemul de login poate oferi posibilitatea înregistrării unui cont pe baza conexiunii automate securizate cu o altă rețea de socializare (Gmail, Facebook și altele).
- Hărțile pot să integreze și sisteme de navigație pentru obținerea indicațiilor de orientare direct din aplicație, către servicii cum ar fi: cabinete și farmacii veterinare, pensiuni sau hoteluri pentru animale.
- Fiecare anunț poate avea integrat un sistem de calculare automată a distanței aproximative dintre poziția curentă a utilizatorului și locația specificată în anunț.
- Utilizarea framework-ului Flutter pentru implementarea clientului facilitează semnificativ trecerea de la o aplicație destinată doar dispozitivelor Android la o aplicație de tip cross-platform. Implementarea configurațiilor necesare utilizării pentru dispozitivele iOS reprezintă primul pas în această direcție.

Prin lista mai sus menționată am demonstrat faptul că, „Adoptă un prieten” are un potențial ridicat de dezvoltare atât în România, cât și în afara granițelor țării. Numeroase feature-ruri pot fi integrate ușor în aceasta, starea actuală de funcționare reprezentând doar baza de pornire pentru ceea ce ar putea însemna cu adevărat o astfel de aplicație dedicată salvării și îngrijirii animalelor de companie prin adopție.

BIBLIOGRAFIE

[1] Amiya Ranjan Rout, *Spring Boot – Application Properties*, December 2021. Available: https://www.geeksforgeeks.org/spring-boot-application-properties/. Last accesed 2 June 2022.

[2] Angela Shi, *Why LinearSVC and SVC with a linear kernel are not the same functions?*, July 2021. Available: https://towardsdatascience.com/svm-with-scikit-learn-what-you-should-know-780f1bc99e4a. Last accesed 4 June 2022.

[3] Akshit J. Dhruv, Reema Patel and Nishant Doshi, *Python: The Most Advanced Programming Language for Computer Science Application*s, Computer Science and Engineering, Pandit Deendayal Petroleum University, Gandhinagar, Gujarat. Available: https://www.scitepress.org/Papers/2020/103079/103079.pdf. Last accesed 4 June 2022.

[4] Aman Singh, Akanksha Rai, Nidhi Biet, *Introduction to Apache Maven | A build automation tool for Java projects*, December 2019. Available: https://www.geeksforgeeks.org/introduction-apache-maven-build-automation-tool-java-projects/. Last accesed 8 June 2022.

[5] American Psychological Association, Personality. Available: https://www.apa.org/topics/personality. Last accesed 8 June 2022.

[6] Andy Hilliard, *Why Mobile Applications Are Important; Especially Its Development*, December 2014. Available: https://www.accelerance.com/blog/why-is-mobile-app-development-so-important-today. Last accesed 4 June 2022.

[7] Arvind Singh, *What is a splash screen?*, January 2021, Available: https://bootcamp.uxdesign.cc/what-is-a-splash-screen-2980602f83f8. Last accesed 4 June 2022.

[8] Bettina Specht, Everything you need to know about SMTP, March 2022. Available: https://postmarkapp.com/guides/everything-you-need-to-know-about-smtp#what-is-smtp. Last accesed 6 June 2022.

[9] David Curry, *Android Statistics (2022)*, May 2022. Available: https://www.businessofapps.com/data/android-statistics/. Last accesed 9 June 2022.

[10] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg,

Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot and Edouard Duchesnay, Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research 12 (2011) 2825-2830. Available: https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf. Last accesed 6 June 2022.

[11] Hussein Terek, *Introducing Spring Boot*, March 2018. Available: https://dzone.com/articles/introducing-spring-boot?utm_source=dzone&utm_medium=article&utm_campaign=spring-boot-content-cluster. Last accesed 29 May 2022.

[12] John K. Waters, *Spring Boot 1.0 Launches*, April 2014. Available: https://adtmag.com/articles/2014/04/09/spring-boot-launch.aspx. Last accesed 6 June 2022.

[13] Kavya Guntupally, Ranjeet Devarakonda and Kenneth Kehoe, *Spring Boot based REST API to Improve Data Quality Report Generation for Big Scientific Data: ARM Data Center Example*, 2018 IEEE International Conference on Big Data. Available: https://www.osti.gov/servlets/purl/1542197. Last accesed 6 June 2022.

[14] Lily Yuan, *What does your pet say about your personality?*, May 2021, Available: https://personality-psychology.com/what-does-your-pet-say-about-your-personality/. Last accesed 4 June 2022.

[15] Maia Paluska, *The Correlation between Personality and Relationships with Pets*. Available: https://www.mckendree.edu/academics/scholars/issue18/paluska.htm . Last accesed 2 June 2022.

[16] Metropolia University of Applied Sciences, March 2018. Available: https://www.theseus.fi/bitstream/handle/10024/146232/thesis.pdf?sequence=1. Last accesed 3 June 2022.

[17] Michiel Mulders, *What Is Spring Boot?*, September 2019. Available: https://stackify.com/what-is-spring-boot/. Last accesed 29 May 2022.

[18] Mohit Joshi, *Firebase Dynamic Links In Flutter*, July 2020, Available: https://medium.flutterdevs.com/firebase-dynamic-links-in-flutter-ab49377dfe40. Last accesed 2 June 2022.

[19] Ola Dahl, *Exploring End User's Perception of Flutter Mobile Apps*, July 2019. Available: https://www.diva-portal.org/smash/get/diva2:1480395/FULLTEXT01.pdf . Last accesed 2 June 2022.

[20] Oxera Consulting LLP, *Android in Europe*, October 2018. Available: https://www.oxera.com/wp-content/uploads/2018/10/Android-in-Europe-1.pdf. Last accesed 6 June 2022.

[21] P.N.Siva jyothi si Rohita Yamaganti, *A Review on Python for Data Science, Machine Learning and IOT*, International Journal of Scientific & Engineering Research, Volume 10 Issue 12, December 2019 852 ISSN 2229-5518. Available: https://www.ijser.org/researchpaper/A-Review-on-Python-for-Data-Science-Machine-Learning-and-IOT.pdf . Last accesed 6 June 2022.

[22] Pooja Mahindrakar and Uma Pujeri, *Insights of JSON Web Token*, International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878 (Online), Volume 8 Issue-6, March 2020. Available: https://www.ijrte.org/wp-content/uploads/papers/v8i6/F7689038620.pdf . Last accesed 5 June 2022.

[23] Pramod Kumar Srivastava, *Understanding the Basics of Spring vs. Spring Boot*, August 2018, Available: https://dzone.com/articles/understanding-the-basics-of-spring-vs-spring-boot?utm_source=dzone&utm_medium=article&utm_campaign=spring-boot-content-cluster. Last accesed 3 June 2022.

[24] Sebastian Raschka, Joshua Patterson and Corey Nolet, *Machine Learning in Python: Main developments and technology trends in data science, machine learning, and artificial intelligence*, March 2020. Available: https://arxiv.org/pdf/2002.04803.pdf. Last accesed 3 June 2022.

[25] Vikramaditya Jakkula, *Tutorial on Support Vector Machine (SVM)*, School of EECS, Washington State University, Pullman 99164. Available: https://course.ccs.neu.edu/cs5100f11/resources/jakkula.pdf. Last accesed 8 June 2022.

[26] Wenhao Wu, *React Native vs Flutter, cross-platform mobile application frameworks*, Metropolia University of Applied Sciences, March 2018. Available: https://www.theseus.fi/bitstream/handle/10024/146232/thesis.pdf?sequence=1. Last accesed 21 May 2022.

[27] Android Developers official documentation: https://developer.android.com/guide (including: https://developer.android.com/guide/components/intents-filters). Last accesed 21 May 2022.

[28] Dart official documentation: https://dart.dev/guides (including: https://pub.dev/documentation/get_navigation/latest/get_navigation/GetMaterialApp/GetMaterialApp.html, https://api.flutter.dev/flutter/dart-developer/dart-developer-library.html, https://dart.dev/codelabs/async-await). Last accesed 4 June 2022.

[29] Firebase official documentation: https://firebase.google.com/docs/engage (including: https://firebase.google.com/docs/dynamic-links/android/receive). Last accesed 6 June 2022.

[30] Flutter official documentation: https://docs.flutter.dev/ (including: https://docs.flutter.dev/development/ui/interactive, https://api.flutter.dev/flutter/widgets/SingleChildScrollView-class.html, https://docs.flutter.dev/development/ui/layout, https://docs.flutter.dev/cookbook/design/fonts, https://pub.dev/packages/responsive_framework, https://docs.flutter.dev/release/breaking-changes/layout-builder-optimization, https://pub.dev/packages/flutter_secure_storage). Last accesed 21 May 2022.

[31] ModelMapper official documentation: http://modelmapper.org/getting-started/ (including: http://modelmapper.org/user-manual/spring-integration/). Last accesed 6 June 2022.

[32] OpenPyXL official documentation: https://openpyxl.readthedocs.io/en/stable/. Last accesed 6 June 2022.

[33] Python 3.8 official documentation: https://docs.python.org/3.8/ (including: https://docs.python.org/3/library/stdtypes.html?highlight=tuple#tuple, https://docs.python.org/3/library/pickle.html). Last accesed 4 June 2022.

[34] Scikit-learn official documentation: https://scikit-learn.org/stable/getting_started.html (including: https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html?highlight=linearsvc#sklearn.svm.LinearSVC, https://scikit-learn.org/stable/modules/generated/sklearn.metrics.ConfusionMatrixDisplay.html?highlig

ht=confusionmatrixdisplay#sklearn.metrics.ConfusionMatrixDisplay). Last accesed 4 June 2022.

[35] Spring official documentation: https://docs.spring.io/spring-framework/docs/current/reference/html/ (including: https://docs.spring.io/spring-framework/docs/3.0.x/spring-framework-reference/html/mail.html, https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/resources.html, https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/core/io/InputStreamResource.html, https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/web/bind/annotation/GetMapping.html, https://docs.spring.io/spring-security/site/docs/3.0.x/reference/el-access.html). Last accesed 6 June 2022.

[36] Spring-Boot official documentation: https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/. Last accesed 6 June 2022.