UNIVERSITATEA ”ALEXANDRU-IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

Dogo - Pets: Walk & Care

propusă de

Leonard Rumeghea

Sesiunea: iunie, 2023

Coordonator științific

Lect. Dr. Frasinaru Cristian

UNIVERSITATEA ”ALEXANDRU-IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

Dogo - Pets: Walk & Care

Leonard Rumeghea

Sesiunea: iunie, 2023

Coordonator științific

Lect. Dr. Frasinaru Cristian

Avizat,

Îndrumător lucrare de licență,

Lect. Dr. Frasinaru Cristian.

Data: ..... Semnătura: .....

Declarație privind originalitatea conținutului lucrării de licență

Subsemnatul **Rumeghea Leonard** domiciliat în **România, jud. Botoșani, mun. Darabani, Str. Văii, nr. 51**, născut la data de **20 februarie 2001**, identificat prin CNP **5010220071367**, absolvent al Facultății de informatică, **Facultatea de informatică** specializarea **informatică**, promoția 2023, declar pe propria răspundere cunoscând consecințele falsului în declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr. 1/2011 art. 143 al. 4 și 5 referitoare la plagiat, că lucrarea de licență cu titlul **Dogo - Pets: Walk & Care** elaborată sub îndrumarea domnului **Lect. Dr. Frasinaru Cristian**, pe care urmează să o susțin în fața comisiei este originală, îmi aparține și îmi asum conținutul său în întregime.

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

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

Data: ..... Semnătura: .....

Declarație de consimțământ

Prin prezenta declar că sunt de acord ca lucrarea de licență cu titlul **Dogo - Pets: Walk & Care**, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test, etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de informatică.

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

Absolvent **Leonard Rumeghea**

Data: ..... Semnătura: .....

Cuprins

Motivație **2**

Introducere **3**

1 Specificații Funcționale **5**

1.1 Descrierea problemei 5

1.2 Cerințe funcționale 6

1.2.1 Cerințe funcționale pentru „Owners” 6

1.2.2 Cerințe funcționale pentru „Walkers” 7

1.3 Cerințe non-funcționale 8

1.3.1 Performanță 9

1.3.2 Scalabilitate 9

2 Arhitectura aplicației **10**

2.1 Arhitectura serverului 10

2.1.1 Stratul de bază 11

2.1.2 Stratul de aplicație 12

2.1.3 Stratul de infrastructură 13

2.1.4 Stratul de interfață 13

2.2 Arhitectura aplicației mobile 14

2.3 Arhitectura bazei de date 14

2.3.1 Avantajele aduse de Sql Server 14

2.3.2 Diagrama bazei de date 15

3 Implementare **16**

3.1 Sericii externe 16

3.1.1 Google Maps Platform 17

3.1.2 Geolocator 18

3.2 Implementarea serverului 18

Concluzii **19**

Bibliografie **20**

Motivație

În societate modernă, animalele de companie joacă un rol din ce în ce mai important în viața oamenilor de zi cu zi. Cercetările recente evidențiază multiplele beneficii aduse de acestea omului, atât de natură fizică cât și psihică. Printre cele fizice se enumeră: creșterea imunității, reducerea alergiilor, îmbunătățirea sănătății cardiovasculare și activitatea fizică zilnică. Cele mai importante beneficii sunt însă de natură psihiologica, iar animalele de companie sunt o modalitate excelentă de a reduce stresul cât și anxietatea provocate de rutină omului modern cât și pentru îmbunătățirea stării de spirit și creșterea a sentimentului general de bunăstare. Prin mângâierea unui câine sau a unei pisici, în organism se produc o serie de modificări fizice, prin creșterea nivelului de serotonină, numită și „hormonul fericirii", dar și prin scăderea nivelului de cortizol, numit și „hormonul stresului".

Cu toate acestea mulți oameni consideră că nu au sau nu pot avea destul timp la dispoziție pentru a avea grijă de un animal de companie și de nevoile acestora, cum ar fi plimbările zilnice sau vizitele periodice la salon sau veterinar. Pe lângă acestea mai apare și o problema în momentul în care persoane dorește să fie plecată mai multe zilele consecutive de acasă. Aceste probleme ar putea fi rezolvate totuși folosind o aplicație specializată pe nevoile „programate" ale animalului. Această aplicație este un intermediar dintre cele 2 persoane: una fiind deținătorul animalului de companie care nu are timpul necesar pentru a se ocupă de o activitate, sau are nevoie de ajutor, iar cealaltă este reprezentată persoană, iubitoare de animale, care dorește să se îl ajute atât de deținător cât și pe animal.

Introducere

Animalele de companie joacă un rol important în viața oamenilor de zi cu zi și oferă multe beneficii mentale, fizice și sociale. Acești însoțitori devin adesea parte din familie, oferind dragoste necondiționată, companie și sprijin moral. Interacțiunea cu animalele de companie reduce stresul și anxietatea, îmbunătățește starea de spirit și contribuie la bunăstarea generală.

Activitățile legate de îngrijirea animalelor de companie, cum ar fi plimbările, vizitele la salon și veterinar, sunt esențiale pentru menținerea sănătății și bunăstării animalului dvs. de companie. De exemplu, timpul în care vă plimbați câinele depinde de mărimea și nivelul de energie, dar în general sunt recomandate între 30 și 60 de minute de plimbare în fiecare zi. Vizitele la salon pot varia în funcție de nevoile fiecarui animal de companie si complexitatea serviciu, cum ar fi tunsul, spălatul sau îngrijirea blănii. Vizitele la veterinar sunt recomandate cel puțin o dată pe an pentru a verifica starea de sănătate a animalului de companie și pentru a preveni eventualele boli.

În aceste situații a avea propria aplicație de îngrijire a animalelor de companie are multe beneficii. Poate oferi programe pentru plimbări, saloane, vizite la veterinar și multe altele, ajutându-i pe proprietari să-și gestioneze mai bine rutină zilnică și să se asigure că nevoile animalelor de companie sunt satisfăcute în mod adecvat și în atunci când apar situații neprevăzute.

Este important să rețineți că aplicația nu înlocuiește atenția sau interacțiunea directă cu animalul dvs. de companie, dar poate fi un instrument util pentru a asigura îngrijirea adecvată și pentru a vă menține animalul de companie sănătos și fericit.

Există o varietate de aplicații de îngrijire a animalelor de companie astăzi pe piață, care acoperă diferite aspecte, cum ar fi plimbări, programarea unei vizite la salon sau la veterinar și servicii de „sitting" a animalelor de companie. Cu toate acestea, majoritatea aplicațiilor se concentrează pe un singur aspect sau specie de animal de companie, iar utilizatorii trebuie să folosească mai multe aplicații pentru a-și gestiona sarcinile

zilnice. În plus, majoritatea aplicațiilor nu oferă proprietarilor o modalitate de a găsi îngrijitori de animale de companie dacă nu au timp să aibă grijă de animalele lor de companie.

Această lucrare de licență prezintă o aplicație mobilă care oferă o soluție completă pentru îngrijirea animalelor de companie. Aplicația permite utilizatorilor să-și gestioneze rutina zilnică, să programeze plimbări, vizite la salon și la veterinar, să găsească îngrijitori de animale de companie și să comunice cu aceștia.

Aplicația prezentată este dezvoltată folosind tehnologiile Flutter[1] și ASP.NET[2] și oferă multe beneficii și caracteristici utile pentru utilizatori. Flutter este un framework dezvoltat de Google care vă permite să dezvoltați aplicații mobile native pentru platformele Android și iOS cu un singur cod sursă. ASP.NET este un alt framework dezvoltat de Microsoft care oferă un mediu robust pentru construirea de aplicații web scalabile[3], adaptabile și sigure.

[1] Alessandro Biessek, Flutter for Beginners, 2019

[2] Andrew Troelsen, Philip Japikse, Introducing ASP.NET MVC, 2017

[3] Sebastian Faust, Using Google's Flutter Framework for the Development of a Large-Scale Reference Application, 2020

Capitolul 1

Specificații Funcționale

Funcția principală a acestei aplicații este de a oferi utilizatorilor o soluție completă și ușoară pentru a găsi pe cineva care să se ocupe de nevoile potențiale ale animalului lor de companie. Utilizatorii aplicației se încadrează în două categorii mari: „Owners" și „Walkers". Un utilizator care are un animal de companie și are nevoie de ajutor este un „Owner", iar un utilizator care răspunde acestei nevoi este un „Walker".

1.1 Descrierea problemei

În prezent, există o varietate de aplicații de îngrijire a animalelor de companie pe piață, care acoperă diferite aspecte, cum ar fi plimbări, programarea unei vizite la salon sau la veterinar și servicii de „sitting" a animalelor de companie. Cu toate acestea, majoritatea aplicațiilor se concentrează pe un singur aspect sau specie de animal de companie, iar utilizatorii trebuie să folosească mai multe aplicații pentru a-și gestiona sarcinile zilnice. În plus, majoritatea aplicațiilor nu oferă proprietarilor o modalitate de a găsi îngrijitori de animale de companie dacă nu au timp să aibă grijă de animalele lor de companie.

Aplicația prezentată în această lucrare de licență nu își propune să integreze toate aceste aplicații într-una singură, ci să ofere o soluție comodă în completarea aplicațiilor dezvoltate de diverse companii de îngrijire a animalelor care necesită deplasarea lor la un sediu fizic. În același timp, aplicația acoperă o gamă mai largă de specii de animale de companie în comparație cu alte aplicații disponibile, incluzând nu numai animale „clasice" precum câini și pisici, ci și rozătoare, păsări, reptile, pești etc. Fiecare persoană este liberă să își adauge propriul animal de companie în aplicație, indiferent de tipul

de animal, iar Walker-ul este liber să aleagă tipul de animal de care vrea să aibă grijă, astfel încât niciun animal să nu fie lăsat în urmă.

Întrucât există o gama largă de specii și servicii disponibile în aplicație, Walker-ul poate deveni dezorientat atunci când navighează și selectează serviciile. În același timp, unii oameni au fobii diferite, cum ar fi cinofobia[1] sau fobia de reptile. Acești oameni nu ar dori că în aplicația lor să apară servicii care nu sunt relevante pentru ei. Din acest motiv, am decis să introducem un sistem de preferințe pentru fiecare Walker care ne permite să oferim o anumită prioritate pe specie de animal și serviciu. Prin urmare, plimbătorii pot alege să ofere doar plimbări pentru câini sau pisici și nu servicii veterinare pentru rozătoare. De asemenea, plimbătorii pot alege să ofere servicii de plimbare pentru câini, pisici și rozătoare, dar nu pot oferi servicii veterinare pentru nicio specie. În acest fel, aplicația devine mai ușor de utilizat și mai intuitivă pentru toți utilizatorii.

1.2 Cerințe funcționale

Principala cerință a acestei aplicații este de a oferi utilizatorilor o interfață intuitivă și ușor de utilizat, care să medieze comunicarea între „Owner" și „Walker". În același timp, atât „Owner-ul", cât și „Walker-ul" ar trebui să folosească aceeași aplicație, dar această ar trebui să ofere capacități diferite specifice fiecăruia. În cele ce urmează, vom prezenta cerințele funcționale pentru fiecare tip de utilizator.

1.2.1 Cerințe funcționale pentru „Owners"

După conectare, prima cerință și sarcină a utilizatorului este de a adauga unul sau mai multe animale de companie in aplicație. Pentru acestea există un meniu dedicat de adăugare și modificare unde puteți alege numele, specia, rasa, data nașterii, sexul și vom putea atașa o scurtă descriere dedicata Walker-ului. După adăugarea unui animal de companie, acesta va apărea în lista de animale de companie a utilizatorului. Aceste informații vor fi folosite de către „Walker" pentru a decide dacă poate să se ocupe de animalul respectiv.

După adăugarea unui animal de companie, utilizatorii pot căuta un „Walker"

[1]Cinofobia (cuvânt care provine din alăturarea a două cuvinte: latinescul canis sau grecescul kýon „câine" și fobie) este o fobie (teamă patologică) de câini.

care să aibă grijă de animalul de companie. Pentru a face acest lucru, utilizatorul trebuie să plaseze anunțuri în aplicație. Anunțul trebui să conțină cele mai importante informații de care are nevoie un „Walker". Acestea includ profilul animalului, dată și ora la care persoană are nevoie de asistență, tipul serviciului solicitat, durata, locația la care trebuie să ajungă animalul de companie dacă este cazul și un scurt mesaj către "Walker". Odată ce anunțul este publicat, acesta va apărea în lista de anunțuri a utilizatorului. În acest moment, anunțul dvs. va fi vizibil pentru toți „Walker-ii". După ce unul din acestea a acceptat anunțul, "Owner-ul" poate vedea asta în aplicație în dreptul rezervării.

Pentru a crește acuratețea și a evita confuzia, locațiile clinicilor veterinare sau a saloanelor pentru animele sunt selectate folosind o hartă. În acest fel, „Walker-ul" știe exact unde să meargă pentru a ajunge la locația dorită. Același lucru este valabil și pentru adresa utilizatorului. Acesta trebuie să selecteze pe hartă o locație de unde „Walker-ul" va ridica animalul de companie.

"Owner-ul" va avea la dispoziție și un meniu în care poate vizualiza istoricul anunțurilor publicate de acesta și satisfăcute de către "Walker-i".

1.2.2 Cerințe funcționale pentru „Walkers"

În aplicație un „Walker" are două îndatoriri principale: să caute noi anunțuri care i se potrivesc și să satisfacă anunțurile pe care le-a acceptat deja.

Pentru a căuta anunțuri noi, „Walker-ul" are o listă cu toate anunțurile postate de proprietarii, care sunt relevante pentru acel „Walker". Pentru a face lista mai ușor de navigat, această poate fi filtrată după specie sau serviciu. Pentru a modifică preferințele sale legate de animale sau servicii, „Walker-ul" are la dispoziție un meniu dedicat. Acesta poate alege pentru fiecare specie de animal de companie sau serviciu un nivel de prioritate. Aceste preferințe vor fi folosite de către aplicație pentru a afișa anunțuri relevante pentru „Walker".

După ce a găsit un anunț care i se potrivește, „Walker-ul" poate vedea detaliile anunțului pentru a-și da seama dacă dorește să îl accepte. Pe lângă detaliile introduse de utilizat, și enumerate mai sus, "Walker-ul" poate vizualiza și o harta care conține drumul de la locația sa curentă până la locația „Owner-ului" și după, dacă este cazul, până la locația unde trebuie să îl ducă pe animalul de companie. Pe lângă traseu "Walker-ul" poate vizualiza distanțele și timpul estimat pentru a ajunge la locația

”Owner-ului” de la locația sa și de la locația ”Owner-ului” la locația unde trebuie să îl ducă, dacă este cazul. Aceste informații sunt oferite de Google Maps API[2] și Distance Matrix API[3]. Acestea sunt calculate pentru mersul pe jos în momentul în care ”Walker-ul” vizualizează anunțul. Timpii necesari pot varia în funcție de traficul din zona la momentul respectiv.

După ce anunțul este acceptat, „Walker-ul” poate vedea asta în aplicație în meniul special pentru acestea. În momentul în care „Walker-ul” dorește să înceapă serviciul îi va fi afișată o harta care conține locațiile la care acesta trebuie să ajungă pentru a prelua animalul de companie și locația unde trebuie să îl ducă, dacă este cazul. În același timp, el poate vedea distanțele și timpul estimat pentru a ajunge la locațiile respective. Totodată pe harta va apărea și locația acestuia în timp real. Aceste informații vor fi disponibile și pentru ”Owner” dacă acesta deschide anunțul în momentul în care ”Walker-ul” a început activitatea. În acest fel, ”Owner-ul” poate vedea când ”Walker-ul” este pe drum spre el și poate să îl aștepte în locația stabilită. După preluarea animalului de companie ”Owner-ul” poate vizualiza în continuare locația ”Walker-ului” pentru a se asigura că totul decurge bine.

1.3 Cerințe non-funcționale

Pe lângă cerințele funcționale, aplicațiile trebuie să îndeplinească și multe cerințe nefuncționale. Acestea nu au nicio legătură cu funcționalitatea aplicației, dar sunt importante pentru a asigura o experiență plăcută atât utilizatorilor, cât și programatorilor care vor lucra la această aplicație în viitor. Aceste cerințe includ două lucruri cheie: performanță și scalabilitate.

[2]https://developers.google.com/maps/documentation

[3]https://developers.google.com/maps/documentation/distance-matrix/distance-matrix

1.3.1 Performanță

Una dintre caracteristicile notabile ale framework-ul Flutter este reprezentată de performanța crescută a acestuia, atât pe platforma Android cât și pe iOS. Datorită arhitecturii native Flutter și compilării AOT (Ahead-of-Time)[4], aplicațiile create cu aceste framework rulează eficient și fără probleme. Ca limbaj de programare Flutter folosește Dart care compilează cod nativ pe mai multe platforme, inclusiv pe cele mobile profitând de avantajele native ale dispozitivelor pe care rulează. Pentru a obține performanțe maxime, Flutter folosește un motor de randare numit Skia[5] care este folosit și de Google Chrome. Acesta este un motor de randare 2D care oferă o performanță excelentă și o experiență de utilizare fluidă.

1.3.2 Scalabilitate

Flutter este cunoscut pentru abordarea reactivă și arhitectura bazată pe widget care facilitează dezvoltarea de aplicații scalabile. Structura widgetului permite o separare clară a responsabilităților și o reutilizare sporită a codului. Aceasta înseamnă că dezvoltatorii pot crea și gestiona cu ușurință interfețe complexe și dinamice, indiferent de dimensiune sau complexitate.

.NET, pe de altă parte, are suport nativ pentru scalabilitate și este folosit pentru a dezvolta aplicații de la desktop și web până la servicii cloud și aplicații pentru întreprinderi. Platforma .NET oferă capabilități puternice de gestionare a memoriei, concurență și distribuție care permit dezvoltatorilor să construiască și să ruleze aplicații scalabile, de performanță înaltă. În plus, framework-urile și serviciile suplimentare precum ASP.NET și Azure oferă instrumente avansate pentru scalabilitatea aplicațiilor web și cloud.

Atât Flutter, cât și .NET beneficiază de comunități active de dezvoltatori și de sprijin din partea unor jucători mari precum Google în cazul Flutter și Microsoft pentru .NET. Deci dezvoltatorii au acces la resurse, documentație și actualizări continue pentru a-i ajuta să depășească provocările de scalabilitate și să implementeze soluții eficiente.

[4]În programare, compilarea Ahead-of-Time (AOT) reprezintă compilarea unui limbaj de programare de nivel înalt într-un limbaj de nivel inferior înainte de execuția programului, de obicei la momentul compilării, pentru a reduce cantitatea de lucru necesară la momentul execuției.

[5]https://skia.org/docs/

Capitolul 2

Arhitectura aplicației

Conform cerințelor non-funcționale, aplicația constă din două componente principale: server și aplicație mobilă. Pentru server, am ales .NET care oferă o bază solidă pentru gestionarea datelor și logicii aplicației. .NET este un framework scalabil și robust, oferind suport pentru baze de date și servicii web, asigurând totodată securitate și performanță. Aplicația mobilă este dezvoltată în Flutter, un framework cross-platform, care permite dezvoltarea aplicațiilor pentru Android și iOS native folosind un limbaj de programare comun, Dart.

2.1 Arhitectura serverului

Pentru realizarea acestui server a folosind o arhitectură "Clean Architecture", fiind un sistem software modular, bine structurat, care promovează separarea clară a responsabilităților și încurajează dezvoltarea de cod ușor de întreținut și testat. O arhitectură „arhitectură curată" se bazează pe principiul inversării dependenței (Dependency Inversion Principle), împărțind o aplicație în niveluri concentrice, fiecare cu un rol bine definit. Straturi principale ale arhitecturii sunt: Stratul de bază (Core Layer), Stratul de aplicație (Application Layer) și Stratul de infrastructură (Infrastructure Layer) alături de Stratul de interfață (Interface Layer).

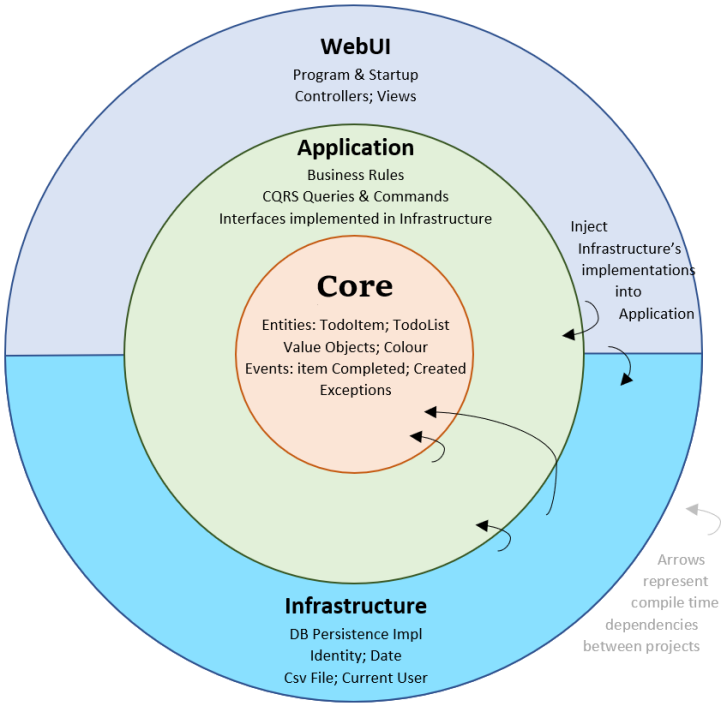

Figura 2.1: Clean Architecture

2.1.1 Stratul de bază

Stratul de bază, sau central, reprezintă nucleul aplicației și conține regulile de business și logică aplicației. Acesta este independent față de restul straturilor, neavând dependințe către niciun dintre ele. Acestă independența permite testarea și refacerea acestuia independent de restul aplicației. Acesta este împărțit în trei părți importante: Entities, Use Cases și Interfaces.

- **Entities** - reprezintă obiectele de bază ale aplicației, care conțin regulile de business și logică aplicației. În acest caz, printre entități se numără: User, Pet, Appointment, etc. și conțin doar datele și metodele necesare pentru a le manipula, fără a avea niciun fel de dependință către alte componente ale aplicației.

- **Use Cases** - reprezintă regulile de business ale aplicației, de exemplu: adăugarea unui nou utilizator, inregistrarea unui animal de companie, realizarea unei programări, etc. Acestea sunt independente de orice altă parte a aplicației, ceea ce înseamnă că pot fi testate fără a fi nevoie de alte componente.

- **Interfaces** - reprezintă interfețele de comunicare cu celelalte straturi ale aplicației. Aici se regăsesc interfețele pentru baza de date, serviciile web, etc.

2.1.2 Stratul de aplicație

Stratul de aplicație în arhitectura "Clean Architecture" reprezintă puntea între interfața utilizatorului și logica de afaceri din Stratul de paza. Gestionează fluxului de date și de interacțiunea între componentele UI (User Interface) și componentele din Core.

Nivelul de aplicație conține următoarele componente:

- **Commands** - Comenzile sunrt reprezentate de cereri sau acțiuni specifice pe care un utilizator le poate efectua in intermediul aplicației. Aceastea includ acțiuni precum crearea obiectelor, actualizarea datelor sau ștergerea acestora. Comenzile sunt de obicei reprezentate de structuri de date care conțin informațiile necesare pentru a efectua acțiunea dorită.

- **Handlers** Handler-ii, sau gestionatorii, sunt componente responsabile pentru acceptarea și gestionarea comenzilor primite, ocupându-se de execuția logicii asociate cu acea comandă. Handler-ul primește comanda, extrage datele necesare și apelează funcțiile corespunzătoare din stratul de bază pentru a efectua operația necesară. Ele pot interacționa cu alte servicii din stratul curent, cum ar fi mappers sau responses, pentru a efectua acțiunile dorite și a returna rezultatele corespunzătoare.

- **Queries** - interogările reprezintă in general cereri de informații din partea utilizatorului. Acestea implica de obicei extragerea datelor din baza de date, prelucrarea lor intr-un anumit mod și returnarea acestora către utilizator. Spre deosebire de comenzi care modifică starea sistemului, interogările au doar rolul de a returna informații într-o formă cerută.

- **Mappers** - mapperele sunt componente care se ocupă de transformarea datelor in formatul necesar. Acestea sunt folosite pentru a transforma obiectele primite de la utilizator in obiecte din stratul de bază, sau pentru a transforma obiectele din stratul de bază în obiecte care pot fi trimise către utilizator, în funcție de caz.

- **Responses** - răspunsurile sunt entități asemănătoare celor din stratul de bază, fiind formate însă doar variabile. Rolul acestora este de a returna datele cerute de utilizator, după ce au fost prelucrate de către handler, dar într-un format sigur. Returnarea unui obiect din stratul de bază ar putea expune către exterior date sensibile, cum ar fi parolele, id-uri, sau ar putea expune detalii despre implementarea internă a aplicației.

2.1.3 Stratul de infrastructură

Stratul de infrastructură oferă implementarea concretă a interfețelor definite ulterior în stratul de bază. Aici sunt incluse componentele care îndeplinesc accesul la baza de date, serviciile de comunicare externă, sistemul de fișiere și alte resurse externe. Stratul conține următoarele componente:

- **Repositories** - reprezintă implementarea detaliată a interfețelor din stratul de bază și sunt responsabile pentru accesul la baza de date. Ele sunt folosite de handleri pentru a introduce, extrage, modifica sau șterge informațiile din baza de date.

- **Services** - serviciile sunt componente care asigură comunicarea cu alte servicii externe, precum servicii web sau servicii de stocare în cloud. Acestea sunt folosite de handleri pentru a efectua operațiunile necesare.

2.1.4 Stratul de interfață

Stratul de interfață reprezintă este responsabil pentru interacțiunea cu utilizatorul. Aici se gasesc componentele pentru interfața cu utilizatorul (interfață web, API, etc.). Acest strat are rolul de a prelua datele de la utilizator, de a le trimite mai departe către handleri si de a returna datele cerute. Principala componentă a acestui strat este API-ul, care este folosit de aplicația mobilă pentru a comunica cu serverul.

- **Controllers** - Controller-ii din API-uri acționează ca intermediari între partea de prezentare, interfața utilizatorului, și restul aplicației. Aceștea primesc cereri HTTP de la clienți pe care le trimit către handlerii din stratul de applicatie. Aici cererea este îndeplinită și ulterior handlerii returnează datele către controller care, la rândul sau, le trimite mai departe către client.

2.2 Arhitectura aplicației mobile

2.3 Arhitectura bazei de date

Pentru baza de date, am ales să folosesc SQL Server, un sistem de management al bazelor de date relaționale. Dezvoltat de Microsoft, SQL Server oferă o bază solidă pentru stocarea datelor într-un mod structurat și eficient. Fiind un sistem relațional, acesta permite definirea și gestionarea tabelelor și relațiilor dintre ele, facilitând organizarea datelor într-un format coerent.

SQL Server acceptă o varietate de tipuri de date, cum ar fi șiruri, numere, date și ore care vă permit să gestionați și manipulați aceste informații folosind funcții și operațiuni specifice. De asemenea, oferă suport pentru funcții și proceduri stocate, permițându-vă să creați o logică personalizată reutilizabilă în baza de date.

Un aspect cheie al SQL Server este capacitatea sa de a lucra cu date spațiale. Aceasta înseamnă că puteți stoca și manipula informații geospațiale, cum ar fi coordonatele geografice și poligoane. Această caracteristică este utilă în aplicații precum cea prezentată care necesită manipularea datelor spațiale pentru a oferi funcționalități precum navigarea sau afișarea unei locații pe hartă.

2.3.1 Avantajele aduse de Sql Server

- **Performanța**[1] - SQL Server este recunoscut pentru capacitatea sa de gestiona cantități mari de date si performanța sa ridicată. Oferă suport pentru indexarea datelor, care permite acces rapidă la date, și optimizarea interogărilor. De asemenea, oferă suport pentru funcții și proceduri stocate, care permit crearea de funcționalități complexe care pot fi apoi reutilizate în cadrul aplicației.

- **Securitatea** - acesta pune la dispozitie o mulțime de funcționalități de securitate robuste pentru protejarea datele stocate în baza de date. Aici sunt incluse autorizarea, criptarea și auditarea, care asigură integritatea și confidențialitate informaților.

[1] Itzik Ben-Gan, Adam Machanic, Dejan Sarka, Kevin Farlee, Performance Tuning and Optimization in SQL Server, Microsoft Press

- **Integrare cu mediul .NET** - ambele produse fiind dezoltate de Microsoft, SQL Server si .NET sunt compatibile și se integrează perfect. Acest lucru facilitează interacțiunea dintre serverul ASP.NET și baza de date, permițându-ne să accesăm și să manipulăm date într-un mod simplu și eficient.

- **Măsuri de siguranță** - SQL Server oferă o serie de mecanisme utile pentru a asigura integritatea datelor în cazul unor evenimente neprevăzute. Funcții precum tranzacții, jurnale de tranzacții și backup-uri periodice ajută la protejarea datelor și la recuperarea lor în cazul unor erori de sistem sau erori umane.

2.3.2 Diagrama bazei de date

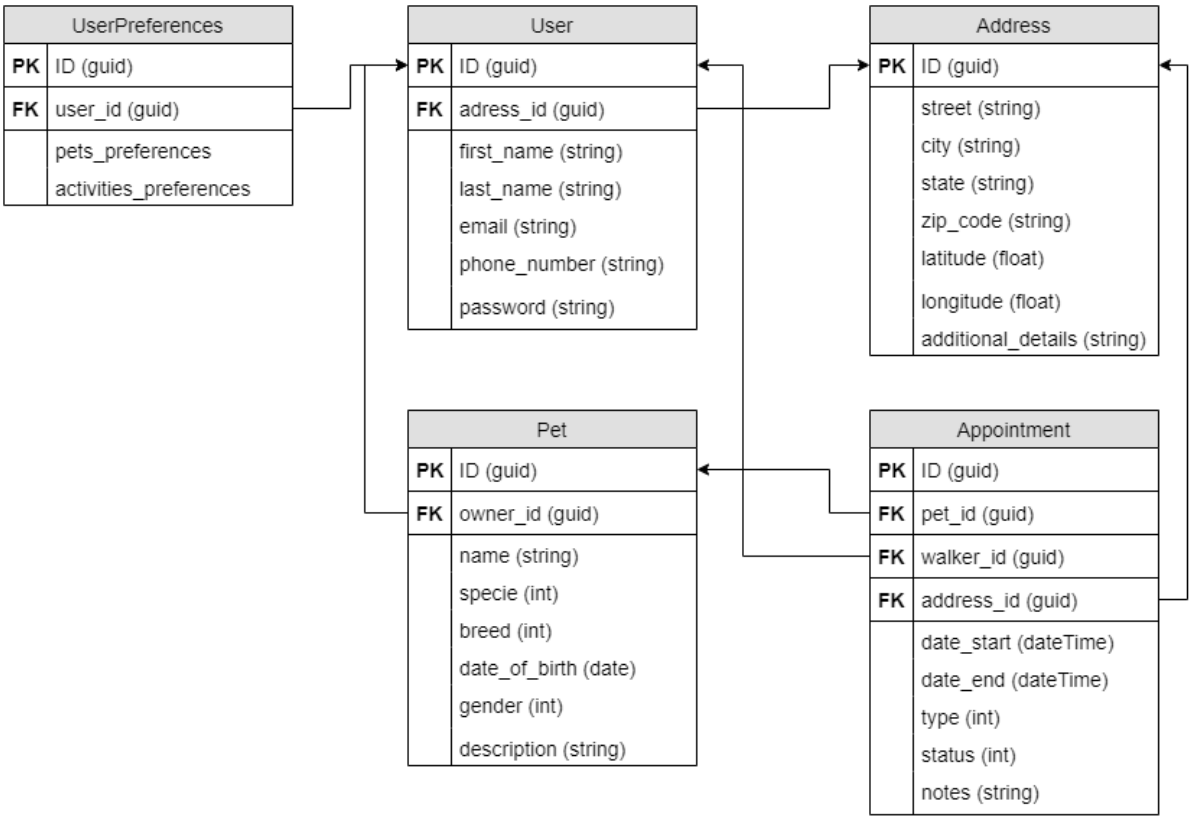

Figura 2.2: Diagrama bazei de date

Capitolul 3

Implementare

Implementarea aplicației a avut loc în două etape. În prima etapă a fost dezvoltat serverul, care oferea serviciile necesare pentru funcționarea aplicației. În a două etapă a fost dezvoltată aplicația mobilă, care folosește serviciile oferite de server pentru a oferi funcționalitatea necesară utilizatorului final. O dată cu dezvoltarea aplicației, au fost modificate și adăugate noi funcționalități serverului, pentru a oferi noi funcționalități care nu se aflau inițial în cerințele proiectului. Pe parcursul implementării, au fost folosite diferite tehnologii și servicii externe, actuale și moderne, care au permis dezvoltarea unei aplicații robuste și scalabile. În continuare, sunt prezentate tehnologiile folosite, precum și modul în care au fost folosite.

3.1 Sericii externe

Pentru dezvoltarea aplicației, au fost folosite diferite servicii externe, in special cele oferite de Google pe partea de navigare. Am preferat folosirea unor servicii externe, deoarece acestea oferă funcționalități complexe, care ar fi necesitată o perioadă mai lungă de timp pentru a fi implementate si nu a fost scopul proiectului de a dezvolta aceste funcționalități. De asemenea, aceste servicii sunt oferite de companii mari, care oferă suport și actualizări constante, ceea ce asigură o aplicație robustă și sigură.

3.1.1 Google Maps Platform

Pentru a putea oferi funcționalitatea de navigare, am folosit serviciul Google Maps. Acesta oferă funcționalitatea de a afișa harta bine cunoscută din cadrul aplicație „Google Maps” și de a calcula și afișa rute între două puncte de pe harta. Pentru a putea folosi serviciul, a fost necesar crearea unui cont de dezvoltator și generarea unei chei de autentificare. Această cheie este folosită de aplicație pentru a se autentifica în cadrul serviciul Google Maps și pentru a putea folosi serviciile oferite de acesta. Din cadrul serviciului Google Maps, au fost folosite următoarele servicii:

Figura 3.1: Vizualizarea hărții în cadrul aplicației

- **Maps SDK** - oferă funcționalitatea de a afișa harta pe dispozitivul mobil. Acest serviciu este folosit pentru a afișa harta în cadrul aplicației. Harta conține pe lângă străzi și clădiri, și punctele de interes, precum parcuri, restaurante, etc.

- **Directions API** - oferă funcționalitatea de a calcula rute între două puncte. Acest serviciu este folosit pentru a calcula rutele dintre locația curentă a utilizatorului și locația unde se află animalul de companie, sau pentru a calcula rutele dintre locația animalului de companie și locația unde se află salonul sau cabinetul veterinar.

- **Distance Matrix API** - oferă funcționalitatea de a determina distanțele și timpii de parcurgere a drumului dintre două puncte. Acest serviciu este folosit pentru a ajută utilizatorul să estimeze timpul necesar pentru a ajunge la locația unde se află animalul de companie, sau pentru a ajunge la locația unde se află salonul sau cabinetul veterinar pentru o mai bună planificare a timpului.

3.1.2 Geolocator

Am utilizat acest servici pentru a determina locația curentă a utilizatorului și ne permite să urmărim locația acestuia pe hartă în momentul deplasării.

Geolocator oferă posibilitatea de a determina locația curentă a utilizatorului folosindu-se de GPS, rețele mobile și conexiuni WiFi. Acesta poate urmării locația utilizatorului în timp real, oferind informații despre locația curentă, viteza de deplasare, altitudine, etc. Acest serviciu este util atât „Walker" pentru vizualizarea în timp real a drumului pe care îl mai are de parcurs până la destinația curentă cât și „Owner" pentru a spori gradul de siguranță al animalului de companie.

3.2 Implementarea serverului

Concluzii

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Nunc mattis enim ut tellus elementum sagittis vitae et. Placerat in egestas erat imperdiet sed euismod. Urna id volutpat lacus laoreet non curabitur gravida. Blandit turpis cursus in hac habitasse platea. Eget nunc lobortis mattis aliquam faucibus. Est pellentesque elit ullamcorper dignissim cras tincidunt lobortis feugiat. Viverra maecenas accumsan lacus vel facilisis volutpat est. Non odio euismod lacinia at quis risus sed vulputate odio. Consequat ac felis donec et odio pellentesque diam volutpat commodo. Etiam sit amet nisl purus in. Tortor condimentum lacinia quis vel eros donec. Phasellus egestas tellus rutrum tellus pellentesque eu tincidunt. Aliquam id diam maecenas ultricies mi eget mauris pharetra. Enim eu turpis egestas pretium.

Bibliografie

- Alessandro Biessek, *Flutter for Beginners*, 2019
- Andrew Troelsen, Philip Japikse, *Introducing ASP.NET MVC*, 2017
- Sebastian Faust, *Using Google's Flutter Framework for the Development of a Large-Scale Reference Application*, 2020
- Google Maps Platform Documentation, *https://developers.google.com/maps/documentation*
- Distance Matrix API Documentation, *https://developers.google.com/maps/documentation/distance-matrix/distance-matrix*
- ASP.NET documentation, *https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-7.0*
- Itzik Ben-Gan, Adam Machanic, Dejan Sarka, Kevin Farlee, *Performance Tuning and Optimization in SQL Server*, Microsoft Press