DogHealt: Connexió pel cuidatge dels gossos

Bonaventura Munill Díaz

26 de juny de 2025

Resum– L'enginyeria del Software és una professió dedicada a oferir solucions a diversos problemes de la vida real mitjançant la creació de programari especialitzat. En aquest projecte, es pretén donar resposta a la necessitat que tenen molts propietaris de gossos de trobar, de manera senzilla i eficient, cuidadors per a les seves mascotes. En aquest sentit, l'objectiu principal del projecte és analitzar i comprendre el disseny, l'evolució i l'execució d'un procés de creació d'un programa informàtic que doni resposta a aquesta problemàtica. Al llarg del informe s'explicarà com s'ha dissenyat el sistema, els materials i eines utilitzats, el desenvolupament progressiu del projecte i, finalment, com s'ha assolit el resultat final. Addicionalment, es farà una explicació dels problemes trobats durant l'evolució del projecte i de com es van solucionar, a més, de coses que s'haurien d'implementar al futur per millorar les funcionalitats del programari.

Paraules clau– Cuidatge de gossos, Cuidador, Propietari, Java, Api, Android Studio, Visual studio, Disseny de software, Mongodb.

Abstract– Software engineering is a profession dedicated to providing solutions to various real-life problems through the creation of specialized software. In this project, the aim is to respond to the need that many dog owners have to find, in a simple and efficient way, caregivers for their pets. In this sense, the main objective of the project is to analyze and understand the design, evolution and execution of a process of creating a computer program that responds to this problem. Throughout the report, it will be explained how the system was designed, the materials and tools used, the progressive development of the project and, finally, how the final result was achieved. Additionally, an explanation will be made of the problems encountered during the evolution of the project and how they were solved, as well as things that should be implemented in the future to improve the functionalities of the software.

Keywords– Dog care, Caretaker, Owner, Java, Api, Android Studio, Visual studio, Software design, Mongodb.

1 INTRODUCCIÓ

DogHealth és una aplicació mòbil dissenyada per facilitar la connexió entre propietaris de gossos i cuidadors, oferint una solució pràctica per a la recerca de professionals en la cura d'animals de companyia. Aquest projecte es desenvolupa com una aplicació real dels coneixements adquirits al llarg del procés de formació i implementació de programari, amb l'objectiu de posar en pràctica les competències tècniques i metodològiques necessàries en aquest àmbit.

La motivació principal per a la creació d'aquest projecte sorgeix d'una experiència personal. En el passat, vaig tenir una gossa anomenada Lluna i, en diverses ocasions, quan viatjàvem, no la podíem portar amb nosaltres. Això ens obligava a buscar cuidadors de gossos, una tasca que sovint resultava complicada i laboriosa. La manca d'una eina centralitzada que facilités aquesta cerca va evidenciar la necessitat d'una aplicació com DogHealth, que podria simplificar aquest procés i proporcionar una solució més eficient per als propietaris d'animals de companyia.

2 METODOLOGIA

Per al desenvolupament d'aquest projecte, es va utilitzar una metodologia específica adaptada a les necessitats

• E-mail de contacte: itri2805@gmail.com
• Menció realitzada: Enginyeria del Software
• Treball tutoritzat per: nom i cognoms del tutor (departament)
• Curs 2024/25

del treball. Concretament, el projecte es va dur a terme mitjançant una estructura basada en sprints. Un **sprint** corresponia a una setmana de treball en què es va desenvolupar una part concreta del projecte. A més, cada sprint es va dividir en diverses fases, cadascuna d'elles agrupava un conjunt de dies amb tasques diferenciades.

Les fases del sprint van ser les següents:

- **Anàlisi situacional (Dilluns):** A l'inici de cada sprint es va dur a terme una anàlisi situacional per determinar l'estat actual del projecte. Primer, es va revisar la planificació per identificar les tasques que calia desenvolupar en el sprint corresponent. A continuació, es van analitzar les funcionalitats que no s'havien pogut implementar en sprints anteriors i, finalment, es va realitzar una planificació detallada per definir el full de ruta del sprint i les funcionalitats que s'havien d'implementar.
- **Implementació (Dimarts a dissabte):** Durant aquesta fase, es va procedir a l'execució de les tasques planificades en la fase anterior. Això va incloure el desenvolupament i la programació de les funcionalitats assignades al sprint, seguint les directrius establertes en la planificació.
- **Avaluació final (Diumenge):** En aquesta fase es va revisar el treball realitzat durant el sprint i es va valorar si s'havien complert els objectius planificats. En cas que es detectessin desviacions o problemes, es van analitzar les causes de les incidències i es van dissenyar mesures correctores per evitar la repetició d'errors en futurs sprints.

Aquesta va ser la metodologia que es va seguir per tenir un compliment eficient en el desenvolupament del treball de final de grau.

3 METARIALS

Per dur a terme aquest projecte, va ser necessari l'ús d'un conjunt d'eines i recursos tecnològics fonamentals per al desenvolupament de l'aplicació. A continuació, es presenta una descripció detallada dels materials seleccionats per poder garantir un desenvolupament eficient i estructurat del projecte:

- **Visual Studio Code:** Va ser un entorn de desenvolupament integrat (IDE) àmpliament utilitzat per a la programació de diferents llenguatges. En aquest projecte, Visual Studio Code [1] va servir com a eina principal per a la redacció del codi font, permetent implementar les funcionalitats necessàries per a l'execució de l'aplicació. El llenguatge de programació emprat per al desenvolupament del codi va ser JavaScript, tant per al backend com per a la interacció amb altres tecnologies.
- **Node.js:** Per gestionar la comunicació entre el backend i el frontend de l'aplicació, es va utilitzar Node.js [2], una llibreria basada en JavaScript que va permetre la creació d'una arquitectura API escalable i eficient. Node.js va ser una eina clau per a la gestió de sol·licituds i respostes entre el client i el servidor, assegurant una transmissió fluida de dades.
- **Flutter:** Per al desenvolupament de la interfície d'usuari i la implementació de les funcionalitats del frontend, es va optar per Flutter [3], un framework de codi obert desenvolupat per Google. Aquesta eina va permetre la creació d'interfícies intuïtives i dinàmiques mitjançant Dart, oferint una experiència d'usuari fluida i atractiva. A més, Flutter va facilitar el desenvolupament d'aplicacions multiplataforma, optimitzant els temps i costos de desenvolupament.
- **MongoDB:** Per a l'emmagatzematge i la gestió de dades de l'aplicació, es va utilitzar MongoDB, una base de dades NoSQL que va permetre una estructura flexible i escalable per a l'emmagatzematge d'informació. MongoDB[4] es va basar en documents en format JSON, cosa que va facilitar la integració amb Node.js i va permetre una gestió eficient de les dades sense necessitat d'una estructura de taules rígida, com en les bases de dades SQL tradicionals.

La combinació d'aquestes eines i tecnologies es va considerar adequada per garantir un desenvolupament sòlid i funcional de l'aplicació. S'esperava que aquests recursos fossin suficients per cobrir totes les necessitats del projecte i assegurar-ne el correcte funcionament.

4 ANÀLISIS

En aquest apartat s'exposarà l'anàlisi que es va realitzar per poder definir la planificació del projecte, així com els diferents components i interaccions que configuren l'aplicació. Aquest estudi va permetre establir les bases per a un desenvolupament estructurat i eficient. A més, de poder de crear uns fonaments pel disseny de la planificació.

4.1 Casos d'ús

Per tal d'identificar les interaccions principals de l'aplicació, es van determinar dos tipus de clients que faran ús del sistema:

- **Usuaris (propietaris de gossos):** Són aquelles persones que utilitzaran l'aplicació per contactar amb cuidadors de gossos i gestionar el servei de cura de les seves mascotes.
- **Cuidadors de gossos:** Són professionals o particulars que ofereixen els seus serveis a través de l'aplicació amb l'objectiu d'augmentar la seva quota de mercat i ampliar la seva xarxa de clients.

Un cop definits els perfils d'usuari, es detallen els principals casos d'ús de l'aplicació:

- **Gestió del perfil d'usuari:** Tant els cuidadors com els propietaris hauran de crear i configurar un perfil d'usuari per poder utilitzar l'aplicació i personalitzar-la segons les seves necessitats.
- **Gestió de perfils de gossos:** Els propietaris hauran d'introduir informació sobre els seus gossos perquè els cuidadors puguin conèixer les seves necessitats específiques. Aquesta funcionalitat permetrà als cuidadors accedir a detalls rellevants abans d'acceptar una sol·licitud.

- **Consulta d'ofertes:** Tant els cuidadors com els propietaris podran accedir a un llistat de cuidadors disponibles. Els usuaris podran visualitzar els cuidadors propers, comparar tarifes i comprovar la seva disponibilitat.
- **Enviament d'ofertes:** Els propietaris, un cop hagin trobat un cuidador que s'adapti a les seves necessitats, podran enviar-li una sol·licitud amb la informació del gos que volen que sigui atès.
- **Acceptació o rebuig d'ofertes:** Els cuidadors rebran notificacions quan un propietari sol·liciti els seus serveis. A partir d'aquí, podran revisar els perfils dels gossos i, segons la seva disponibilitat i criteris, acceptar o rebutjar l'oferta.

Fig. 1: Diagrama de casos d'ús

Aquests casos d'ús van ser bastant útils per poder entendre bé els requisits del nostre sistema i el seu funcionament. Permetem d'aquesta forma un bon disseny de l'aplicació.

4.2 Disseny del Front-end

El front-end és la part de l'aplicació amb la qual interactuaran els usuaris. Per a una millor organització, es divideix en diferents pantalles:

- **Pantalla d'inici:** En aquesta pantalla, l'usuari accedirà a l'aplicació. Si no disposa d'un compte, se li demanarà que es registri.
- **Pantalla d'ofertes:** Mitjançant un mapa interactiu, es mostraran els cuidadors disponibles en una determinada àrea, així com la seva informació i tarifes.
- **Pantalla de perfil d'usuari:** Espai destinat a la gestió del perfil de cada usuari, permetent la modificació de dades personals i configuracions específiques.
- **Pantalla de notificacions:** Els usuaris rebran i gestionaran notificacions relacionades amb sol·licituds d'ofertes, confirmacions i altres comunicacions del sistema.
- **Pantalla de perfil de gossos:** Secció en què els propietaris podran afegir, modificar o eliminar els perfils dels seus gossos per facilitar la gestió dels serveis de cura.

4.3 Disseny del Back-end

El back-end és l'encarregat de gestionar totes les operacions del sistema, encara que no serà perceptible directament pels usuaris. Per tal d'estructurar el codi de manera clara i modular, aquest es divideix en diferents classes.

Ara mostrarem com va acabar el disseny del Back-end:

- **Classe Propietari:** Classe que representa el propietari i les accions que pot dur a terme.
- **Classe Cuidador:** Responsable de gestionar l'enviament de sol·licituds al cuidador i de mostrar la informació relacionada.
- **Classe Notificació:** Gestiona tota la informació i les funcionalitats relacionades amb les notificacions.
- **Classe Gos:** Classe encarregada de gestionar tota la informació relacionada amb els gossos.

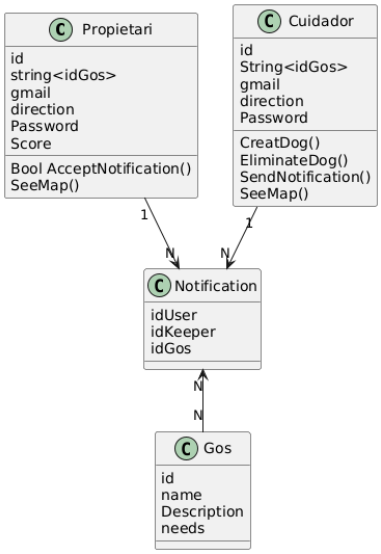

Fig. 2: Diagrama de classes

4.4 Base de dades

El sistema té una base de dades que l'utilitza per emmagatzemar la informació que maneja l'aplicació. Aquesta base de dades es compon de dues taules: La taula Usuaris i la taula Gossos.

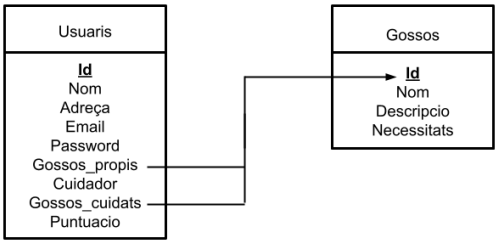

Fig. 3: Representacio visual de la base de dades

Taula Usuari, aquesta taula estarà composta per les següents variables:

- **id:** Número identificatiu únic assignat a cada usuari dins del sistema. Aquest identificador és irrepetible i permet diferenciar cada usuari de manera unívoca.
- **nom:** Nom associat al perfil de l'usuari, que serà el que aquest faci servir dins de l'aplicació.
- **adreça:** Lloc de residència de l'usuari. Aquesta informació és essencial per facilitar la detecció de cuidadors propers.

- **email:** Correu electrònic de l'usuari, que servirà com a mitjà de contacte entre propietaris i cuidadors.
- **password:** Aquesta variable emmagatzema les contrasenyes que hauran d'utilitzar els usuaris per identificar-se en l'aplicació.
- **gossos propis:** Llista d'identificadors numèrics que permet associar cada propietari amb els seus gossos registrats.
- **cuidador:** Variable booleana que indica si l'usuari està registrat com a cuidador.
- **gossos cuidats:** Llista d'identificadors numèrics que associen un cuidador amb els gossos que ha tingut a càrrec.
- **puntuació:** Valor numèric que representa l'avaluació del cuidador per part d'altres usuaris en funció de la qualitat del servei ofert.

Taula Gossos. Aquesta taula emmagatzema la informació específica de cada gos registrat en l'aplicació i conté les següents variables:

- **id:** Número identificatiu únic assignat a cada gos dins del sistema. Aquest identificador és irrepetible i permet associar-lo correctament amb el seu propietari.
- **nom:** Nom que el propietari ha assignat al seu gos.
- **descripció:** Text que proporciona una descripció general del gos, incloent-hi aspectes com el seu caràcter, mida o particularitats específiques.
- **necessitats:** Informació detallada sobre les cures específiques que requereix el gos per garantir el seu benestar, com alimentació, medicació o preferències d'activitat.

Aquest disseny de la base de dades assegura una gestió òptima de la informació dels usuaris i dels seus gossos, facilitant així la interacció entre propietaris i cuidadors dins de l'aplicació.

5 Evolució del projecte

En aquesta secció es donarà una explicació de com va ser l'evolució del projecte i quin estat es va finalitzar.

5.1 Primera fase: Configuració de l'entorn de programació

Aquesta primera fase se'n va preparar l'entorn de l'espai de programació. Primera es van crear els directoris:

- **Project:** Directori on es guardava tots els arxius necessaris per al funcionament de l'aplicació.
- **Project/doghealt:** En aquest directori es desava tota el programari relacionat amb les pantalles de l'aplicació.
- **Project/Doghealt-Backend:** En aquest directori era on desàvem tot el codi que gestionava la connexió entre el front-end amb la base de dades.

Posteriorment, es van descarregar i instal·lar els següents programaris: Android studio, flutter, visual studio, node.js i MongodDB compass.

Amb Android studio i flutter, conjuntament amb el directori Project/doghealt era on es programava les pantalles de l'aplicació utilitzant el llenguatge Dart.

En el directori Project/Doghealt-Backend amb el node.js era on programem totes les funcions que s'havien d'executar en el backend i les crides a les bases de dades.

5.2 Segona fase: Primer contacte

La segona fase es va consistir en pogramar un splascreen, una pantalla de iniciar usari y una de registre pero poder entendre els conceptes basics del llenguatge dart, les crides de node.js i el funcionament de les bases de dades. Es va crear l'arxiu service.js per tenir una compilació de totes les crides a les bases de dades. En la base de dades es va crear la col·lecció User (on guardaríem els usuaris creats i els de prova) i la col·lecció Dogs (on guardaríem les dades relacionades amb gossos.)

Posteriorment, vam implementar la pantalla splascreen que mostra el logo provisional de l'aplicació. A continuació es van fer la pantalla iniciar usuari i registres. Les versions d'aquestes eren senzilles per poder agilitzar els processos i implementar les restes de funcions. Es plantejava que en el futur aplicar millores com l'encriptació de la contrasenya i ajustar el registre segons els canvis fets en la base de dades.

Fig. 4: Visualització de com era la pantalla Splashscreen

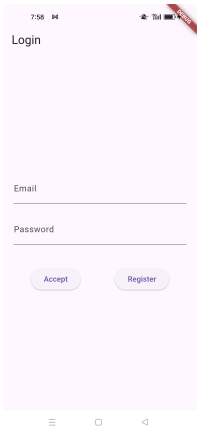

Fig. 5: Visualització de com era la pantalla login

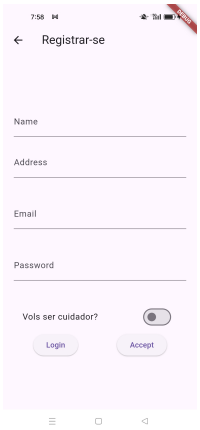

Fig. 6: Visualització de com era la pantalla Registre. Cal remarcar que l'opció de si era cuidador o no era important el funcionament de l'app.

5.3 Tarcera fase : implementacio del mapa y entorn de la aplicacio

En aquesta fase ens vam concentrar en la implementació d'un mapa que permetés la visualització dels cuidadors en la zona de l'usuari. Per aconseguir l'efecte es va decidir optar utilitzar el servei de google de mapes. En aquest sentit, es va poder implementar el que ens interessava.

Fig. 7: Visualització de com era la pantalla del mapa. El marcador blau marca la posició de l'usuari i el marcador vermell indica la posició d'un cuidador de gossos.

De forma addicional es van afegir tres pantalles (gossos, notificacions i usuaris) per permetre'ns poder programar un bottom navigation. En futures fases s'aniria implementen funcions en aquestes pantalles.

5.4 Quarta fase: implementacio del enviament de notificacions

En aquesta fase es van implementar diferents funcionalitats del sistema. Primer es va implantar en la pantalla gossos. Això consistia que en la pantalla visualitzes els gossos que tenies a propietat.

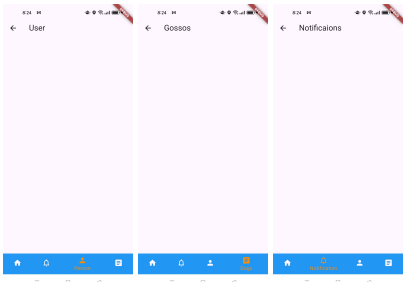

Fig. 8: Visualització del funcionament del bottom navigation.

Fig. 9: Visualització de com va quedar la pantalla de gossos.

A continuació, es va implementar una nova funcionalitat dins del sistema de mapes de l'aplicació. Aquesta funcionalitat permetia que, en pressionar un marcador ubicat al mapa —el qual representa un cuidador de gossos disponible—, es desplegués una opció que oferís a l'usuari la possibilitat d'enviar una petició de cuidatge al cuidador seleccionat.

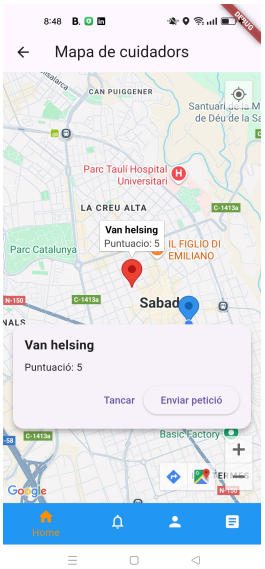

Fig. 10: Visualització de enviar peticions.

5.5 Cinquena fase: Arreglar el codi

En aquesta fase, es va optar per reorganitzar i optimitzar el codi del backend per tal de facilitar-ne la creació i el manteniment. Inicialment, tot el codi es trobava concentrat en un únic arxiu anomenat service.js. Això provocava una acumulació de codi que feia difícil gestionar i escalar el programari del backend.

Per aconseguir aquesta optimització, es va decidir modularitzar el procés, dividint el codi en diferents arxius segons la seva funcionalitat. El resultat final de l'estructura del codi va ser el següent:

- **service.js:** Arxiu principal que actua com a punt de connexió amb la resta de mòduls.
- **config/:** Directori que conté l'arxiu db.js, l'encarregat de gestionar la connexió amb el servidor de base de dades.
- **controllers/:** Directori amb els arxius que defineixen el comportament de les crides a la base de dades.
- **models/:** Directori que conté les definicions de les estructures de dades de la base de dades.
- **routes/:** Directori que unifica models i controladors, definint les rutes que s'utilitzen a les crides.

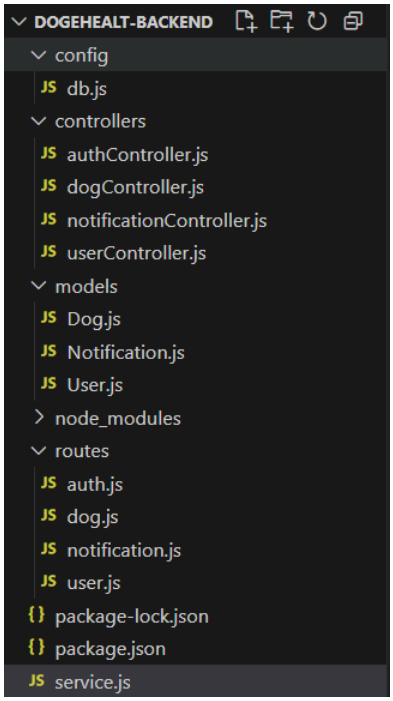

Fig. 11: Estructura de com va acabar al backend.

A més d'aquesta reorganització, es va crear una variable global (baseUrl) amb la IP del servidor per simplificar-ne la gestió. D'aquesta manera, en cas que calgués modificar la IP, només seria necessari fer-ho en un sol lloc, sense haver d'editar diversos arxius manualment. Fent que les crides quedin de la següent forma: Uri.parse('baseUrl/api/ControllerType/Callname');

5.6 Sisena fase: implementació del sistema de gestió de notificacions

En aquesta fase, es va implementar tota la gestió de les notificacions. Quan un cuidador rebia una notificació, podia escollir entre dues opcions: acceptar o denegar la proposta.

Si l'acceptava, tant al cuidador com al propietari els apareixia al sistema de notificacions que la proposta havia estat acceptada. En canvi, si la proposta era denegada, la notificació desapareixia del sistema del cuidador i, al propietari, li apareixia un missatge indicant que la proposta havia estat rebutjada. A partir d'aquest moment, l'usuari podia decidir si volia eliminar la notificació o mantenir-la.

Per aconseguir aquest funcionament, es van crear els següents arxius al backend:

- **notificationController.js:** Arxiu que conté la lògica de les crides relacionades amb les notificacions.
- **Notification.js:** Arxiu que defineix l'estructura de la taula Notification a la base de dades.
- **notification.js:** Arxiu que unifica el model i el controlador, i que, a més, ofereix les rutes corresponents per a les crides relacionades amb notificacions.

Pel que fa al frontend, es va utilitzar l'arxiu Notification.dart, que ja s'havia creat durant la tercera fase del projecte.

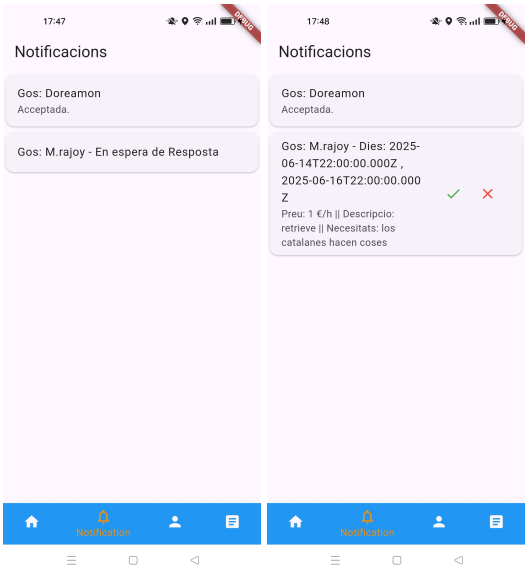

Fig. 12: Visualització de les notificacions del propietari (dreta) i del cuidador (esquerra).

A més, es va crear una nova taula a la base de dades per gestionar les notificacions. Aquesta taula contenia les següents variables:

- **pricePerHour:** El preu per hora que el cuidador cobrava pel servei.
- **startDay:** Dia d'inici del període de cuidatge.
- **endDays:** Dia de finalització del període de cuidatge.
- **accepted:** Variable booleana que indica si l'oferta havia estat acceptada o no.

- **dogId:** Identificador del gos que s'havia de cuidar.
- **keeperId:** Identificador del cuidador que rebia l'oferta.
- **userId:** Identificador del propietari que enviava l'oferta.

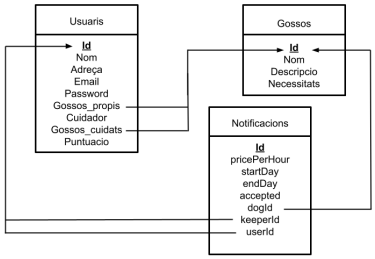

Fig. 13: La nova estructura de las bases de dades.

Finalment, com a part d'aquesta fase, es va afegir un botó de tancament de sessió a la pantalla de l'usuari. Perquè d'aquesta forma els usuaris podran tancar sessió quan vulguin.

5.7 Sèptima fase: Últims detalls

En la part final del Treball de Final de Grau es van completar aquelles funcionalitats que encara estaven pendents de desenvolupament.

En primer lloc, es va implementar la possibilitat de registrar-se, desenvolupant la part corresponent del backend que fins aleshores no havia estat implementada a la pantalla de registre.

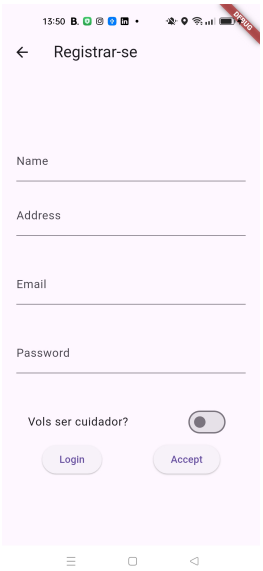

Fig. 14: Pantalla final de registre.

Posteriorment, es va implementar la funcionalitat que permet als usuaris amb rol de propietari crear perfils per als seus gossos dins de l'aplicació. Aquesta funcionalitat

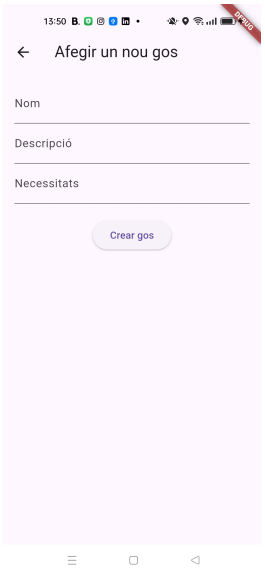

Fig. 15: Pantalla creació de gossos.

facilita que cada propietari pugui introduir les dades corresponents als seus animals, com ara el nom, la descripció i les necessitats específiques de cada gos.

Amb la implementació d'aquestes dues funcionalitats es va donar per finalitzat el període de desenvolupament corresponent al Treball Final de Grau.

6 Problemas

En aquesta secció descrivírem diferents problemes que es van trobar al llarg del desenvolupament.

6.1 Simulacions i connexions.

Inicialment, es feia ús del simulador integrat a Android Studio per provar el funcionament de les pantalles de l'aplicació i avaluar-ne el rendiment. No obstant això, amb el pas del temps es va constatar que determinades funcionalitats, com ara la visualització del mapa, es carregaven amb molta lentitud dins del simulador. Davant d'aquesta limitació, es va optar per executar i comprovar el funcionament de l'aplicació en un dispositiu mòbil real.

Aquesta decisió, però, va posar de manifest un nou repte tècnic: la connexió entre el dispositiu i el servidor. Perquè l'aplicació funcionés correctament, era imprescindible que tant el dispositiu mòbil com el servidor estiguessin connectats a la mateixa xarxa Wi-Fi. A més, calia modificar manualment la IP utilitzada per realitzar les crides al servidor, ja que, a causa del funcionament dinàmic de les xarxes, aquesta IP podia canviar periòdicament.

Per obtenir la nova adreça IP del servidor, es feia ús de la comanda ipconfig. Gràcies a aquest procediment, va ser possible executar i testejar correctament l'aplicació des del dispositiu mòbil.

6.2 Api de google maps.

A l'inici de la tercera fase, es va detectar la necessitat d'integrar el servei de Google Maps per tal de poder implementar

la funcionalitat que permet als usuaris visualitzar els cuidadors propers a la seva ubicació. Afortunadament, era possible obtenir un compte d'API de Google de manera gratuïta, la qual cosa va facilitar l'accés al servei sense cap cost inicial. Aquesta disponibilitat ens va permetre, durant un període de temps, dur a terme proves i implementar les funcionalitats específiques que requeríem.

6.3 Errors de configuracio.

A les primeres etapes del desenvolupament del projecte, es van crear múltiples versions del mateix amb noms molt similars entre si. Aquest procés de duplicació i eliminació successiva de projectes va generar una sèrie de conflictes interns, especialment relacionats amb la crida de determinades funcions i la correcta referència als fitxers dins de l'estructura del programari.

Aquests problemes es van manifestar en forma d'errors de compilació o mal funcionament de certes funcionalitats, causats per la confusió en els enllaços interns entre fitxers i la manca d'un sistema unificat per a les importacions. Per tal de resoldre aquesta situació, es va decidir reanomenar de manera definitiva el projecte, establint una nomenclatura clara i consistent.

A més, es va procedir a unificar la forma d'importar els mòduls i components interns, assegurant que tots els fitxers utilitzessin el mateix format d'importació. Un exemple concret d'aquesta correcció és la importació del fitxer argumentswidget.dart, que es va establir de forma estàndard mitjançant la sintaxi següent: ïmport 'package:dogehealt/screens/argumentswidget.dart';". Amb aquestes mesures, es va aconseguir estabilitzar l'entorn de desenvolupament, evitar confusions futures i garantir un funcionament més robust i coherent del programari.

7 Funcionalitats a implementar al futur

En aquesta secció tractarem d'explicar de funcionalitats que no es van poder implementar per temes de temps i que s'haurien d'afegir en futures revisions del projecte o del codi.

7.1 Mecanismes de seguretat

Actualment, l'estat de la nostra aplicació no disposa de cap mesura de seguretat implementada. Per tal de millorar la protecció de les dades i la seguretat general del sistema, caldria aplicar diversos mecanismes.

En primer lloc, és fonamental protegir les contrasenyes mitjançant tècniques de hashing. En l'estat actual, les contrasenyes s'envien a través de les crides sense cap tipus d'encriptació, fet que les exposa a possibles atacs del tipus man-in-the-middle. Per evitar-ho, s'hauria d'aplicar una funció de hash abans de transmetre-les, de manera que la informació sensible no sigui accessible per tercers.

En segon lloc, seria recomanable implementar un sistema de tokens d'autenticació. Quan un usuari inicia sessió, es podria generar un token que s'adjunti a cada crida futura, permetent identificar l'usuari de forma segura i garantint que només les accions autoritzades siguin permeses.

Finalment, caldria restringir l'accés a les crides del servidor. Actualment, qualsevol usuari pot executar qualsevol crida, sense cap control d'autenticació o autorització. Per solucionar-ho, el servidor hauria de validar que les crides provenen d'usuaris autenticats, ja sigui mitjançant els tokens esmentats anteriorment o mitjançant alguna variable o mecanisme intern de validació de l'aplicació.

7.2 Editar gossos

L'aplicació actual permet la creació i eliminació dels perfils de gossos, però no contempla la possibilitat de modificarlos. Per tal de millorar l'experiència d'ús i oferir una gestió més completa, seria necessari implementar una funcionalitat que permeti als propietaris editar la informació dels perfils dels seus gossos.

7.3 Perfil usuari

L'actual pantalla de gestió d'usuaris només permet que l'usuari pugui tancar sessió. Caldria implementar un conjunt de funcionalitats que milloressin la qualitat i usabilitat de l'aplicació des del punt de vista de l'usuari. Entre aquestes, seria recomanable incorporar la possibilitat que l'usuari pugui modificar el seu perfil i/o actualitzar la seva contrasenya. A més, també seria convenient afegir opcions relacionades amb la configuració de l'aplicació i la gestió dels permisos associats.

7.4 Peticions amb múltiples gossos

Actualment, quan el propietari fa una petició de cuidatge, només pot enviar una sol·licitud per a un únic gos cada vegada. Per tal de millorar l'experiència d'ús de l'aplicació, seria convenient implementar la funcionalitat que permeti als propietaris incloure múltiples animals en una mateixa petició. Aquesta millora optimitzaria el procés i faria l'aplicació més eficient i pràctica per als usuaris amb més d'un gos.

7.5 Procés de testing

Per concloure, cal fer esment al procés de proves (testing). A causa de l'abast i la durada del projecte, no s'ha dut a terme un procés de testing estructurat i sistemàtic. En aquest sentit, seria necessari realitzar un conjunt de proves per verificar la integritat i el correcte funcionament de l'aplicació. En primer lloc, caldria fer proves de flux per identificar possibles fragments de codi no executats. Posteriorment, es recomana fer proves d'usuari en entorns reals, proporcionant l'aplicació als usuaris perquè la provin i ofereixin el seu feedback. Finalment, s'haurien d'efectuar proves de validació per assegurar que l'enviament de dades no generi errors ni al servidor ni a l'aplicació mateixa.

8 Conclusions

Arribem a la conclusió que desenvolupar i portar a terme un projecte d'aplicació no és una tasca senzilla. Requereix concentració, planificació i una bona gestió del temps per assolir els objectius marcats. Es tracta d'un procés complex, especialment quan és dut a terme per una sola persona. El que més valoro d'aquest projecte són els coneixements que

he pogut consolidar al llarg del desenvolupament del Treball de Final de Grau.

AGRAÏMENTS

Vull expressar el meu agraïment, en primer lloc, a Emanuel Adolfo, per haver estat el tutor que m'ha acompanyat i guiat al llarg del desenvolupament d'aquest projecte. La seva orientació ha estat clau durant tot el procés. En segon lloc, vull agrair a la meva família el suport incondicional que m'ha ofert al llarg de tota la carrera, especialment en els moments més exigents. Finalment, també vull donar les gràcies al lector per dedicar part del seu temps a llegir aquesta memòria.

REFERÈNCIES

[1] https://code.visualstudio.com/

[2] https://nodejs.org/es

[3] https://docs.flutter.dev/

[4] https://www.mongodb.com/