UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Ingegneria dell'Informazione ed Elettrica e Matematica applicata

Corso di Laurea in Ingegneria Informatica

"DoggoFinder": progetto e sviluppo di un'applicazione nativa iOS per fornire una piattaforma di supporto per smarrimento cani. Implementazione e gestione della persistenza dei dati tramite l'utilizzo di Firebase

Anno Accademico 2023/2024

Sommario

Introduzione

Il progetto oggetto di discussione si colloca in un contesto tecnico contemporaneo in cui le sempre più avanzate tecnologie disponibili sul mercato offrono la possibilità di risolvere problematiche sociali in maniera sempre più efficiente. È stato sostenuto durante l'attività di tirocinio "Swift App Development Bootcamp" [1], un programma formativo per sviluppatori "Full Stack" di applicazioni mobile native iOS/iPadOS «dall'idea al prodotto» in linguaggio Swift [2]. Esso verte sul supporto al ritrovamento di cani smarriti. Nella società odierna le metodologie utilizzate per la risoluzione di questo problema risultano essere poco efficienti e caratterizzate da un tasso di successo relativamente basso. Questa inefficienza è principalmente dovuta alla difficoltà di reperimento e scambio di informazioni, aggravata dal basso impiego di tecnologie moderne. Il progetto ha avuto come obiettivo lo sviluppo di un'applicazione nativa iOS che mira a risolvere queste criticità offrendo una piattaforma centralizzata che facilita lo scambio di informazioni grazie all'uso di tecnologie avanzate basate sull'intelligenza artificiale e sulla geolocalizzazione, incrementando le possibilità di successo nel ritrovamento dei cani.

Parte fondamentale del mio contributo personale al progetto è stata l'implementazione e la gestione della persistenza dei dati in tempo reale dell'applicazione, tramite l'utilizzo di un cloud database. Sono state sviluppate funzionalità sia per la gestione dei dati inerenti alle informazioni condivise tra i vari utenti, sia per l'autenticazione di questi ultimi e la cura dei loro dati sensibili. Inoltre, è stata utilizzata anche un'interfaccia per il salvataggio e la gestione in locale di specifiche informazioni. L'uso di questi strumenti ha permesso di realizzare un sistema robusto ed efficiente per la conservazione sicura e l'accesso agevole ai dati dell'applicazione.

In sintesi, essa rappresenta un avanzamento significativo rispetto alle soluzioni esistenti, integrando tecnologie moderne per aumentare il tasso di successo del ritrovamento di cani smarriti e fornendo un sistema centralizzato e facilmente accessibile per la gestione delle informazioni.

Proseguendo, verrà presentato nel dettaglio l'intero progetto oggetto di discussione. Nel capitolo 1, verrà fornita una panoramica di tutte le tecnologie utilizzate, sia hardware che software, le quali hanno permesso lo sviluppo dell'applicazione finale. Successivamente, nel capitolo 2, verrà discussa l'intera progettazione dell'applicazione, partendo dai requisiti che essa deve rispettare, illustrando i vari casi d'uso possibili e mostrando i mockup utilizzati come base per la realizzazione del suo design. In seguito, il capitolo 3 tratterà la fase implementativa, descrivendo nei minimi dettagli tutte le varie funzionalità dell'applicazione tramite l'utilizzo di specifiche tecnologie. In conclusione, nel capitolo 4 verrà presentata una sintesi del progetto portato a termine, fornendo spunti per futuri miglioramenti o sviluppi di ulteriori funzionalità sulla base degli aspetti trattati.

Capitolo 1 – Panoramica delle tecnologie utilizzate

In questo capitolo verranno analizzate le varie tecnologie adoperate durante il progetto. Sono state utilizzate sia tecnologie hardware specifiche, sia tecnologie software innovative per lo sviluppo e il testing dell'applicazione prodotta.

1.1. Tecnologie hardware

L'hardware scelto per lo svolgimento del progetto, visto l'obiettivo di sviluppo di un'applicazione con sistema operativo iOS, appartiene nella sua totalità all'ecosistema Apple.

Sono stati utilizzati:

- un MacBook Air [3] da 13,6 pollici con chip Apple M2, memoria unificata da 8 GB e memoria SSD da 256 GB, per l'intero sviluppo dell'applicazione e per parte del testing;
- un iPad Air [4] Wi-fi di quinta generazione con memoria interna da 64 GB dedicato al testing dell'applicazione.

Inoltre, è stato possibile usufruire anche di iPhone personali per il testing dell'applicazione. Il portatile MacBook è stato essenziale per il progetto poiché ha a disposizione un ambiente di sviluppo esclusivo per iOS, chiamato Xcode. Per quanto riguarda l'iPad, dotato di un sistema operativo iPadOS compatibile con il sistema operativo iOS di cui sono dotati gli iPhone, è stato cruciale per il testing dell'applicazione in assenza di un iPhone personale da poter utilizzare.

1.2. Tecnologie software

Il software scelto per lo sviluppo dell'applicazione può essere suddiviso in tre categorie:

- strumenti per lo sviluppo
- linguaggi di programmazione
- framework

1.2.1. Strumenti per lo sviluppo

Gli strumenti utilizzati per lo sviluppo dell'applicazione durante il corso del progetto sono principalmente tre: Xcode, Figma e GitHub. Essi hanno svolto un ruolo cruciale nel semplificare i compiti riguardanti la progettazione e l'implementazione delle varie funzionalità dell'app e la coordinazione tra i vari membri del gruppo partecipanti.

Xcode [5] è l'ambiente di sviluppo integrato (IDE) ufficiale di Apple, indispensabile per la creazione di applicazioni native iOS, iPadOS, macOS, watchOS e tvOS usufruibili su iPhone, iPad, Mac, Apple TV, Apple Watch e Apple Vision Pro. Nel caso discusso in questa tesi l'uso di Xcode è volto allo sviluppo di un'applicazione nativa iOS/iPadOS usufruibile su iPhone/iPad. Questo IDE supporta come linguaggio di programmazione il linguaggio Swift, oggetto di studio nella prima fase del progetto, essenziale per l'impiego del framework integrato SwiftUI, utilizzato per lo sviluppo di interfacce utente. Xcode integra in maniera completa tutto ciò riguardante l'ecosistema Apple e offre vari metodi per il testing del codice prodotto. Infatti, mette a disposizione del programmatore sia una "Preview", per dare un primo feedback allo sviluppatore, sia un simulatore integrato, che permette il testing su ogni dispositivo Apple virtuale in assenza del rispettivo dispositivo fisico. Inoltre, è dotato anche di una documentazione integrata ricca e facilmente accessibile agli sviluppatori.

Figma [6] è uno strumento di design, basato sul cloud, che offre numerosi vantaggi per quanto riguarda la creazione di mockup e user interface. Permette una collaborazione in tempo reale tra i vari membri del team. Questo particolare è uno dei suoi punti di forza, poiché consente in tempo reale sia di visionare modifiche effettuate da terzi, commentare e contribuire al design, rendendo la collaborazione nel gruppo fluida ed efficiente, cosa che non viene garantita in altri tool dediti allo sviluppo del design. Inoltre, mette a disposizione una vasta gamma di prototipi da cui prendere spunto per lo sviluppo di mockup o interfacce grafiche. In aggiunta, offre la possibilità di creare prototipi interattivi direttamente all'interno della piattaforma per facilitare il testing delle interazioni dell'utente. Oltretutto, mette a disposizione anche uno storico, per tenere traccia di tutte le versioni del design e, nel caso ci fosse bisogno, ripristinare versioni precedenti. In relazione al progetto, Figma ha permesso di creare vari mockup e prototipi dell'applicazione, permettendo un'ampia scelta sul design finale da utilizzare.

GitHub [7] è una piattaforma che offre numerosi vantaggi per gli sviluppatori nella gestione dei progetti software a cui stanno lavorando. Nel progetto trattato, questo strumento è stato utilizzato per coordinare i vari membri del gruppo e tenere traccia dei vari sviluppi. GitHub utilizza Git, un sistema di controllo delle versioni distribuito che consente di tracciare ogni modifica al codice sorgente, facilitando il rollback a versioni precedenti e il confronto tra diverse versioni del codice. Mette a disposizione una piattaforma centralizzata dove salvare i progetti, rendendoli accessibili e modificabili dai vari membri del team autorizzati. Inoltre, su GitHub, grazie alla possibilità di creare più "branch" secondari, oltre a quello principale chiamato "main", ogni membro del gruppo di sviluppo ha avuto la possibilità di lavorare indipendentemente dagli altri sul proprio branch, per poi unire sul main i risultati raggiunti. Nel caso sorgessero conflitti nella sincronizzazione delle modifiche tra diversi sviluppatori, GitHub mette a disposizione anche una funzionalità per la risoluzione di questi conflitti, permettendo di scegliere la versione che si considera più adatta.

1.2.2. Linguaggi di programmazione

L'unico linguaggio di programmazione utilizzato per lo sviluppo dell'applicazione è il linguaggio Swift, oggetto di studio durante la prima fase del progetto. È stato possibile apprendere in maniera semplice ed efficiente questo linguaggio grazie ai "Playgrounds", una funzionalità inclusa in Xcode che permette di scrivere codice Swift e vedere i risultati in tempo reale senza dover creare progetti, consentendo un rapido apprendimento. Inoltre, in aggiunta ai Playgrounds, è stato impiegato anche il libro "Develop in Swift Fundamentals" [8], messo a disposizione da Apple Education, per fornire le conoscenze di base del linguaggio. Esso è open source, supportato da una comunità di sviluppatori attiva e in crescita che permette una continua evoluzione e un continuo miglioramento del linguaggio. In aggiunta, Swift è un linguaggio compilato, orientato agli oggetti, il quale supporta anche la programmazione funzionale. Include inoltre una gestione automatica della memoria tramite ARC (Automatic Reference Counting), il quale previene perdite di memoria e semplifica la gestione delle risorse. Essendo il linguaggio principale per lo sviluppo di software in ecosistema Apple, Swift ha a disposizione una documentazione ufficiale ricca e facilmente accessibile, oltre a svariate fonti non ufficiali che aiutano nella risoluzione di problemi durante la scrittura del codice.

1.2.3. Framework

I framework utilizzati durante il progetto sono stati vari. Essi sono stati essenziali per l'implementazione di tutte le funzionalità dell'applicazioni, in particolare per lo sviluppo dell'app in generale, per l'uso del database, per l'utilizzo dell'intelligenza artificiale, per la gestione della connessione alla rete internet e per l'implementazione di una mappa, usufruendo della geolocalizzazione del dispositivo. Di seguito saranno elencati e descritti i vari framework.

SwiftUI [9] è un framework sviluppato da Apple che consente la creazione di interfacce utente per applicazioni appartenenti all'ecosistema Apple. Introdotto per la prima volta alla Worldwide Developers Conference (WWDC) del 2019, SwiftUI si affianca ai framework UIKit [10] e AppKit [11] , distinguendosi per il suo approccio dichiarativo rispetto a quello imperativo dei suoi predecessori. Ciò fa intendere che gli sviluppatori che utilizzano SwiftUI devono concentrarsi maggiormente sulla descrizione della soluzione che vogliono implementare rispetto al come questa poi viene effettivamente implementata. Questo framework è strettamente integrato con l'IDE Xcode, ambiente di sviluppo Apple. Esso fornisce viste, controlli e strutture layout per dichiarare l'interfaccia utente desiderata. La struttura dell'applicazione e delle sue interfacce, definita con questo framework, sono conformi all'"App protocol" e al "View protocol", due protocolli per lo sviluppo di applicazioni native iOS. SwiftUI fornisce anche varie funzionalità per la gestione di eventi derivanti da tocchi, gesti o altre tipologie di input, strumenti per la gestione del flusso di dati dai vari modelli della propria applicazione alle rispettive viste disponibili agli utenti e strumenti per la gestione dello stato delle varie viste, permettendo un aggiornamento in tempo reale dei vari cambiamenti. Un fattore importante di questo framework è la possibilità di usufruire di numerose viste già implementate per la composizione di viste più complesse, riducendo il carico di lavoro dello sviluppatore.

SwiftData [12] è un framework sviluppato da Apple per la gestione semplificata dei dati persistenti nelle applicazioni. Esso combina la tecnologia già affermata di Core Data [13] con le più moderne features di Swift, rendendo possibile l'implementazione della persistenza dei dati nella propria applicazione in maniera rapida, con una minima parte di codice e nessuna dipendenza esterna. Con questo framework i dati vengono modellati come oggetti puri di Swift, facilitando il lavoro degli sviluppatori. Inoltre, con SwiftData è stata introdotta anche una sintassi fluida e intuitiva per quanto riguarda l'esecuzione di query sui dati persistenti. Per l'uso di questi modelli dei dati è necessario, all'interno del proprio codice, stabilire una variabile d'ambiente chiamata contesto del modello, utilizzata per l'accesso ad esso. Questo framework, inoltre, garantisce una sincronizzazione automatica dei dati che ha modellato. Per lo sviluppo dell'applicazione sotto esame, SwiftData è stato utilizzato in larga parte in parallelo con MapKit e CoreLocation per la modellazione dei dati riguardanti una mappa.

Create ML [14] è un framework sviluppato da Apple che consente la costruzione e l'addestramento di modelli di machine learning tramite l'utilizzo di un'interfaccia semplice e intuitiva. Introdotto nel WWDC del 2018, è progettato per essere integrato all'interno delle applicazioni Apple tramite l'utilizzo del framework Core ML. Oltre alla sua semplicità, Create ML sfrutta al pieno GPU e CPU disponibili sul Mac, permettendo un training distribuito del modello, accelerandone il processo. Inoltre, durante la fase di addestramento, fornisce feedback immediati sulle prestazioni del modello tramite dati rilevanti e grafici. Dopo aver concluso l'addestramento, il framework mette a disposizione tutte le statistiche riguardanti il modello ottenuto e ne permette il testing immediato tramite la propria interfaccia. Se da un lato Create ML risulta essere estremamente semplice, anche per coloro che non hanno dimestichezza in ambito di Machine Learning, dall'altro lato non permette l'addestramento di modelli la cui complessità risulta essere elevata o il cui scopo è estremamente specifico. I modelli addestrabili tramite l'utilizzo di questo framework sono di vario tipo:

- modelli per la classificazione di immagini;
- modelli per la classificazione di video e azioni registrate;
- modelli per la classificazione di attività rilevate dai sensori del dispositivo;
- modelli per la classificazione di suoni;
- modelli per la classificazione di testo;
- modelli per classificare dati utilizzando tecniche tabulari.

Durante il progetto oggetto di discussione, questo framework è stato utilizzato per l'addestramento di un modello per la classificazione di immagini a singola label (ad ogni immagine viene assegnata un'unica classe).

Core ML [15] è un framework sviluppato da Apple per l'integrazione e l'utilizzo di modelli di Machine Learning su dispositivi appartenenti all'ecosistema Apple. Introdotto nel WWDC del 2017, ha permesso agli sviluppatori di aggiungere nuove funzionalità basate sull'intelligenza artificiale nelle loro applicazioni. Questo framework viene utilizzato in coppia con Create ML, sfruttando i modelli di Machine Learning da esso creati e addestrati. Questi modelli hanno un formato specifico per poter essere usati da Core ML. Inoltre, è anche possibile utilizzare modelli importati da altre fonti, a patto che vengano opportunamente convertiti nel formato corretto. Core ML sfrutta tutto il potenziale hardware di Apple, fra cui il Neural Engine, per permettere di eseguire modelli di Machine Learning in maniera estremamente efficiente dal punto di vista sia energetico sia prestazionale. In aggiunta, esso supporta quattro framework per la gestione di specifiche tipologie di modelli addestrati: Vision [16] per l'analisi di immagini, Natural Language [17] per processare del testo, Speech [18] per la conversione di audio in testo, Sound Analysis [19] per l'identificazione di suoni. Nell'applicazione sviluppata durante il corso del progetto questo framework è stato utilizzato in parallelo con Create ML e Vision per l'integrazione di un modello per la classificazione di immagini.

Vision è un framework sviluppato da Apple per l'analisi e l'elaborazione di immagini. Introdotto nel WWDC del 2017, sfrutta tecnologie di machine learning e features di Swift per portare a termine attività di visione artificiale. Fra i vari obiettivi raggiungibili tramite l'utilizzo di Vision abbiamo: riconoscimento di immagini, riconoscimento e analisi del viso, tracciamento degli oggetti, riconoscimento di testo, in 18 lingue diverse, supportando l'"Optical Character Recognition" (OCR) e riconoscimento di forme e contorni. Questo framework, per lo sviluppo di applicazioni Apple, viene affiancato al framework Core ML per permettere allo sviluppatore di usufruire dei suoi modelli di Machine Learning. Nel progetto oggetto di discussione, è stata utilizzata la capacità di Vision di riconoscere e classificare in maniera efficiente le immagini.

Core Image [20] è un framework sviluppato da Apple per l'elaborazione e l'analisi ad alto livello di immagini e video. Offre un'ampia gamma di strumenti per il filtraggio di immagini e permette la concatenazione di più filtri per creare effetti visivi complessi. È possibile anche creare filtri personalizzati usufruendo di Core Image Kernel Language (CIKL), un linguaggio che permette la scrittura di kernel personalizzati in maniera tale da aggiungere routine di elaborazione personalizzate alle immagini. Core Image è progettato per sfruttare a pieno la GPU dei dispositivi Apple migliorando l'elaborazione delle immagini e garantendo un consumo minimo di energia. In aggiunta, questo framework fornisce opzioni per il rendering ad alta qualità e supporta vari formati di immagine. Riguardo lo sviluppo dell'applicazione, Core Image è stato fondamentale per le funzionalità che usufruiscono del modello addestrato di classificazione delle immagini. Infatti, fornite le immagini da dover classificare, il framework si occupa della loro ottimizzazione, per permettere un'identificazione più efficiente da parte del modello.

MapKit [21] è un framework sviluppato da Apple che permette l'implementazione all'interno delle applicazioni di mappe interattive e servizi di geolocalizzazione. Esso permette, oltre l'incorporamento di queste mappe, anche la loro personalizzazione. Inoltre, è possibile aggiungere alle mappe annotazioni come marker per segnalare punti di interesse e overlay personalizzati per disegnare percorsi. Il framework supporta anche il "geocoding", ovvero la conversione di indirizzi fisici in coordinate geografiche, e il "reverse geocoding", cioè la conversione di coordinate geografiche in indirizzi fisici. Queste funzionalità sono

particolarmente utili per la visualizzazione di informazioni dettagliate sulla posizione. MapKit, in aggiunta, consente anche la ricerca in tempo reale di punti di interesse nelle vicinanze, basandosi ovviamente sulla posizione geografica del dispositivo. Per quanto riguarda il progetto oggetto di discussione, questo framework è stato utilizzato per l'integrazione di una mappa, sfruttando le funzionalità di ricerca di specifici punti di interesse e di aggiunta di marker. Oltre ciò, è stata aggiunta anche la funzionalità di "LookAround", per permettere una visione a livello stradale dei punti di interesse disponibili.

Core Location [22] è un framework sviluppato da Apple che permette di ottenere la posizione geografica e l'orientamento di un dispositivo. Esso utilizza, per la raccolta di dati, tutti i componenti disponibili sul dispositivo, tra cui Wi-Fi, GPS, Bluetooth, magnetometro e barometro. Grazie a queste sue capacità, questo framework permette di fornire servizi di localizzazione sia in tempo reale sia dopo significativi cambiamenti, e inoltre le funzionalità sono attive anche in background. In aggiunta, con Core Location è possibile rilevare anche la direzione e la velocità del dispositivo ed eseguire "geofencing", ovvero la creazione di regioni geografiche virtuali per il monitoraggio di ingressi o uscite del dispositivo da quest'area. Un'altra funzionalità interessante è il rilevamento e la localizzazione di "beacons", dispositivi hardware che consentono il trasferimento di dati entro un raggio specifico dal dispositivo ricevente tramite l'utilizzo di segnali "Bluetooth Low Energy" (BLE). Per poter usufruire di Core Location è necessario istanziare la classe "CLLocationManager", tramite la quale è possibile richiedere agli utenti i permessi di accesso alla loro posizione. Altro aspetto da considerare è il notevole consumo di energia causato dal continuo utilizzo del GPS per aggiornamenti di posizione in tempo reale. In relazione all'applicazione sviluppata, questo framework è stato utilizzato insieme a MapKit per l'integrazione di una mappa interattiva.

Network [23] è un framework sviluppato da Apple per la gestione delle connessioni di rete tra vari dispositivi. Esso fornisce un'API moderna e potente, sostituendo il framework "CFNetwork" [24] e offrendo un'alternativa più ad alto livello. Questo framework fornisce la possibilità di utilizzare vari protocolli, fra cui TLS, TCP, UDP e QUIC per l'invio e la ricezione di dati fra le tue applicazioni. Utilizza inoltre un "URLSession" per il caricamento di risorse basate su HTTP e URL. Inoltre, supporta sia connessioni basate su IPv4, sia basate su IPv6, gestendo in maniera automatica la ricerca del miglior percorso di rete. Network permette anche il monitoraggio dello stato della rete, la gestione della sua sicurezza e consente una riduzione del consumo energetico del dispositivo per le operazioni di rete, in particolare relative alle reti mobili. Nell'applicazione sviluppata, questo framework è stato essenziale per il rilevamento della connessione alla rete e, nei casi in cui il dispositivo non sia connesso alla rete durante l'utilizzo dell'applicazione, permette la segnalazione dello stato "offline" all'utente.

PhotoKit [25] è un framework sviluppato da Apple che fornisce agli sviluppatori strumenti per l'accesso e la gestione della libreria multimediale dell'utente, inclusi foto, video e altri elementi. Questo framework offre varie funzionalità per la manipolazione degli elementi multimediali e inoltre supporta la gestione di Live Photos (combinazione di foto e video molto breve) e del formato HEIF (High Efficiency Image Format), un formato di compressione altamente efficiente utilizzato sui dispositivi Apple. Durante il progetto, questo framework è stato utilizzato, affiancato dall'istanza della classe "PHPhotoLibrary", per la gestione degli accessi da parte dell'utente alla galleria del dispositivo.

Firebase [26] è un framework sviluppato da Google che fornisce una suite completa di strumenti e servizi per progettare e migliorare applicazioni, incentrandosi in maniera particolare su "backend-as-a-service" (Baas), un modello per fornire agli sviluppatori la possibilità di collegare le loro applicazioni web o mobile a un backend cloud storage e di usufruire di API predefinite. Firebase è ampiamente utilizzato per la sua capacità di

semplificare notevolmente lo sviluppo di applicazioni, mettendo a disposizione degli sviluppatori numerose funzionalità integrate e una console online per semplificarne l'utilizzo. Le funzionalità principali offerte da questa piattaforma possono essere raccolte in quattro categorie principali: gestione dei dati, gestione dell'applicazione, analisi e testing.

Per quanto riguarda le funzionalità appartenenti alla categoria "gestione dei dati" abbiamo:

- Firestore Database [27]: un database flessibile e scalabile per lo sviluppo mobile, web e server tramite Firebase e Google Cloud. È un database NoSQL che permette il salvataggio di dati con un formato simile a JSON e consente il funzionamento dell'applicazione anche quando il dispositivo è offline, grazie al salvataggio dei dati nella cache. Inoltre, sincronizza i dati in tempo reale tra le applicazioni client e offre un'integrazione perfetta con gli altri prodotti Firebase. Ciò comporta la possibilità di sviluppo di applicazioni reattive, funzionanti indipendentemente dalla latenza di rete o dalla disponibilità di connessione ad Internet.
- Realtime Database [28]: un database NoSQL, ospitato sul cloud, che permette di archiviare i dati in formato JSON e sincronizzarli su tutti i client in tempo reale. Offre un supporto quando il dispositivo sul quale è presente l'applicazione è offline e sincronizza in tempo reale tutti i dispositivi client connessi alla rete.
- Data Connect [29]: un database relazionale che consente di gestire i servizi legati alla persistenza dei dati delle proprie applicazioni tramite l'utilizzo di un database PostgreSQL basato su Cloud SQL, un servizio che fornisce database relazionali completamente gestiti nel cloud.
- Authentication [30]: fornisce servizi di backend, SDK intuitive per un utilizzo semplificato e librerie per la creazione di interfacce utente con lo scopo di autenticare gli utenti all'interno dell'applicazione. Supporta sia l'autenticazione classica tramite numero di telefono/e-mail e password, sia tramite provider autorizzati come Google, Facebook, Apple ed altri.

Per quanto riguarda le funzionalità appartenenti alla categoria "gestione dell'applicazione" abbiamo:

- App Check [31]: strumento utile per la salvaguardia delle risorse API e backend della propria applicazione impedendone l'accesso ad utenti non autorizzati. I dispositivi che dispongono di questo tool saranno in grado di usufruire di un provider delle attestazioni in grado di attestare che le richieste di accesso provengano dall'applicazione originale e/o che le richieste di accesso provengano da un dispositivo autentico e non manomesso.
- Messaging [32]: soluzione che consente l'invio di messaggi di notifica in modo affidabile e senza costi. Per il suo utilizzo bisogna disporre di un server con il quale costruire, indirizzare e inviare il messaggio, e un'applicazione client Apple, Android o web (strutturata con Javascript) per la ricezione di questi messaggi tramite il servizio di trasporto specifico della piattaforma corrispondente.
- Remote Config [33]: servizio cloud che permette agli sviluppatori di modificare il comportamento o il design della propria applicazione senza dover obbligare gli utenti che la utilizzano a scaricare aggiornamenti. Questo è possibile creando valori predefiniti in-app che appunto controllano questi aspetti dell'applicazione. Andandoli a modificare, tramite la console di Firebase o l'API di Remote Config, verranno sovrascritti tutti i valori corrispondenti, sulle applicazioni degli utenti target scelti. Ciò avviene poiché le applicazioni scaricate dagli utenti verificano frequentemente se sono avvenute modifiche a questi valori e, inoltre, il tutto ha un impatto trascurabile sulle prestazioni.

Per quanto riguarda le funzionalità appartenenti alla categoria “analisi” abbiamo:

- Realtime Analytics [34]: strumento per l’analisi delle interazioni dell’utente con l’applicazione sviluppata. Consente analisi illimitate di al massimo 500 eventi distinti, definiti con l’SDK di Firebase. I report risultanti da queste analisi forniscono numerosi vantaggi agli sviluppatori, come ad esempio la possibilità di reperire informazioni e prendere determinate decisioni riguardanti l’aspetto di marketing legato all’applicazione oppure di focalizzarsi sull’ottimizzazione delle prestazioni. Tutti i dati raccolti possono essere visionati tramite una dashboard apposita.
- Crashlytics [35]: strumento utilizzato per la segnalazione di arresti anonimi dell’applicazione per Apple, Android, Flutter e Unity. Crashlytics aiuta a monitorare, stabilire la priorità e risolvere queste anomalie che minano la stabilità e la qualità dell’applicazione riducendo il carico di lavoro dello sviluppatore. Ciò avviene poiché questo strumento raggruppa in maniera intelligente gli arresti anomali delle applicazioni e ne evidenzia le circostanze che lo hanno generato.

Per quanto riguarda le funzionalità appartenenti alla categoria “testing” abbiamo:

- Test Lab [36]: infrastruttura di testing per le applicazioni sviluppate. Permette di effettuare test su una vasta gamma di dispositivi Android e iOS sia virtuali sia su dispositivi fisici in esecuzione in un data center di Google. In questo modo è possibile rilevare problemi che si verificano unicamente su determinati dispositivi e con specifiche configurazioni.
- A/B Testing [37]: strumento di testing incentrato sull’aspetto di marketing e fidelizzazione dell’utente. Viene utilizzato in parallelo con Messaging per consentire allo sviluppatore di testare diversi messaggi di marketing, e con Remote Config per permettere allo sviluppatore di testare l’impatto sull’utenza, monitorando parametri chiave, come entrate o fidelizzazione, derivante dal cambiamento di alcuni aspetti interni dell’applicazione. Ciò viene effettato come test prima di distribuire l’applicazione su larga scala.

In relazione all’applicazione sviluppata nel corso del progetto, questo framework è stato utilizzato per fornire un database Firestore, aggiornabile in tempo reale e facilmente integrabile con l’ambiente di sviluppo Xcode, e per implementare le funzionalità legate all’autenticazione dell’utente tramite l’inserimento di e-mail e password, usufruendo di Authentication.

Figura 1. Icone ufficiali di alcune tecnologie utilizzate

Capitolo 2 – Progettazione della soluzione

In questo capitolo sarà approfondita la fase di progettazione dell'applicazione. Si partirà con una descrizione dettagliata delle diverse funzionalità e degli altri aspetti che l'applicazione deve includere. Successivamente, si passerà a un'analisi accurata dei requisiti necessari per il suo sviluppo, accompagnata dalla presentazione di vari casi d'uso che illustrano come l'utente interagirà con l'applicazione. Saranno poi introdotti i principali mockup che hanno orientato il design finale, insieme ai wireflows, ovvero insiemi di mockup che rappresentano il flusso di esecuzione delle attività utente. Infine, sarà illustrata la progettazione del database utilizzato per la gestione della persistenza dei dati dell'applicazione.

2.1. Descrizione dell'applicazione "DoggoFinder"

L'applicazione, chiamata "DoggoFinder", nasce con lo scopo di fornire supporto a persone che smarriscono il proprio cane, dando la possibilità di pubblicare annunci di smarrimento su una bacheca condivisa fra i vari utenti e aggiornata in tempo reale. Per coloro che si registrano, viene messa a disposizione una funzionalità di precompilazione degli annunci, usufruendo delle informazioni personali che hanno deciso di fornire all'applicazione. Tuttavia, l'utente può accedere anche come ospite, non fornendo alcuna informazione personale. Per una migliore ed efficiente ricerca di annunci all'interno della bacheca, l'applicazione mette a disposizione diverse alternative di filtraggio, sia testuale, sia tramite utilizzo di un modello di intelligenza artificiale basato su immagini, per il riconoscimento della razza del cane. Inoltre, è anche possibile ordinare gli annunci per data di pubblicazione crescente o decrescente. Oltre alla bacheca, è disponibile anche una mappa interattiva, che mostra tutti gli annunci a cui è stata fornita una posizione tramite GPS e, se ce ne fosse bisogno, anche due tipologie di punti di interesse legate alla salvaguardia dell'animale. In relazione a questi punti di interesse, se disponibile, viene data la possibilità di visualizzarli in modalità "street view", per facilitarne la localizzazione. In aggiunta a ciò, è disponibile nell'applicazione anche una bacheca privata contenente i propri annunci pubblicati. Ciò facilita il loro monitoraggio e, se fosse necessario, la loro modifica o la loro rimozione. Un utente registrato può, inoltre, modificare le informazioni personali che desidera utilizzare, per facilitare la compilazione dei suoi annunci, e può anche effettuare il logout dal proprio account. Tutti i dati dell'applicazione vengono gestiti tramite un cloud database.

2.2. Analisi dei requisiti

Partendo dalla descrizione dell'applicazione è possibile estrapolare tutti i vari requisiti che essa deve possedere, per permetterne uno sviluppo corretto, in accordo con le direttive prestabilite.
Di seguito verrà presentato il "Documento dei requisiti" [38], in forma tabellare, nel quale sono contenuti tutti i requisiti identificati ed analizzati in maniera accurata, fornendo anche una breve descrizione, la tipologia e la priorità assegnata per ciascuno di essi. Successivamente verrà presentato uno "Use Case Diagram" [39], che mostra i principali casi d'uso in cui l'utente si può trovare interagendo con l'applicazione, i quali varranno descritti nel dettaglio al termine di questo paragrafo.

2.2.1. Documento dei requisiti

Tabella 1. Rappresentazione tabellare del documento dei requisiti

Requisito	Descrizione	Tipologia	Priorità
Accesso come utente ospite	L'applicazione permette all'utente di velocizzare l'accesso evitando l'inserimento di dati personali.	Funzionale	2
Accesso come utente registrato	L'applicazione consente all'utente di autenticarsi per recuperare le informazioni inserite precedentemente.	Funzionale	2
Registrazione nuovo utente	L'applicazione consente ad un nuovo utente di registrarsi, fornendo informazioni obbligatorie per l'accesso e consentendo l'inserimento di informazioni opzionali utili all'interno dell'app.	Funzionale	2
Logout	L'applicazione consente ad un utente che ha effettuato l'accesso di disconnettersi dal suo account.	Funzionale	2
Bacheca annunci di smarrimento	L'applicazione è dotata di una bacheca condivisa tra gli utenti per la visualizzazione degli annunci di smarrimento cani.	Funzionale	1
Aggiornamento bacheca	L'applicazione consente l'aggiornamento in tempo reale della bacheca.	Funzionale	1
Filtraggio annunci con ricerca testuale	L'applicazione permette il filtraggio per razza canina o città di smarrimento, con ricerca testuale, degli annunci.	Funzionale	2

Filtraggio annunci tramite intelligenza artificiale	L'applicazione consente il filtraggio degli annunci per razza canina sfruttando il riconoscimento di essa con intelligenza artificiale, dopo aver fornito un'immagine del cane smarrito.	Funzionale	2
Ordinamento annunci per data di pubblicazione	L'applicazione permette di ordinare gli annunci, presenti in bacheca principale e personale, per data di pubblicazione crescente o decrescente.	Funzionale	4
Inserimento nuovo annuncio	L'applicazione permette l'inserimento di nuovi annunci.	Funzionale	1
Modifica annuncio	L'applicazione permette la modifica di un annuncio personale.	Funzionale	1
Rimozione annuncio	L'applicazione permette la rimozione di un annuncio personale.	Funzionale	1
Accesso alla fotocamera del dispositivo	L'applicazione può richiedere l'accesso alla fotocamera del dispositivo per alcune sue funzionalità.	Funzionale	2
Accesso alla galleria del dispositivo	L'applicazione può richiedere l'accesso alla galleria del dispositivo per alcune sue funzionalità.	Funzionale	2
Accesso alla posizione del dispositivo	L'applicazione può richiedere l'accesso alla posizione del dispositivo per alcune sue funzionalità.	Funzionale	3
Mappa integrata	L'applicazione dispone di una mappa integrata.	Funzionale	3

Visualizzazione punti di interesse geolocalizzati	L'applicazione permette la visualizzazione di punti di interesse sulla mappa tramite l'uso della geolocalizzazione.	Funzionale	3
Visione street view per punti di interesse geolocalizzati	L'applicazione permette di visualizzare punti di interesse presenti sulla mappa in modalità street view, solo se il suddetto punto di interesse dispone di questa funzionalità.	Funzionale	4
Visualizzazione annunci geolocalizzati	L'applicazione consente di visualizzare sulla mappa annunci che dispongono di coordinate geografiche.	Funzionale	3
Bacheca personale	L'applicazione fornisce una bacheca personale per visualizzare unicamente gli annunci che un utente ha pubblicato.	Funzionale	1
Area utente	L'applicazione mette a disposizione dell'utente registrato un'area personale dove poter visualizzare le informazioni personali fornite.	Funzionale	2
Modifica informazioni utente	L'applicazione consente, ad un utente registrato, di poter modificare le informazioni che ha fornito, accendendo alla propria area utente.	Funzionale	2
Precompilazione annuncio	L'applicazione consente, ad utenti registrati, una funzionalità di precompilazione degli annunci da pubblicare, utilizzando le informazioni personali fornite.	Funzionale	2
Database per gestione dati	L'applicazione sfrutta un cloud database per la gestione della persistenza dei suoi dati.	Non funzionale	1

Interfaccia grafica	L'applicazione è dotata di un'interfaccia intuitiva, la quale segue le direttive imposte da Apple, che semplifica l'esperienza dell'utente nel suo utilizzo.	Non funzionale	1

Il campo "Tipologia" nella tabella definisce lo scopo di ciascun requisito, specificando se esso è relativo all'implementazione di una funzionalità dell'applicazione o se riguarda altri aspetti.

Il campo "Priorità" nella tabella indica l'importanza assegnata a ciascun requisito, determinando il loro ordine di implementazione. La priorità è espressa in modo decrescente: un numero inferiore, in questo campo, corrisponde a una maggiore importanza del requisito.

2.2.2. Use Case Diagram

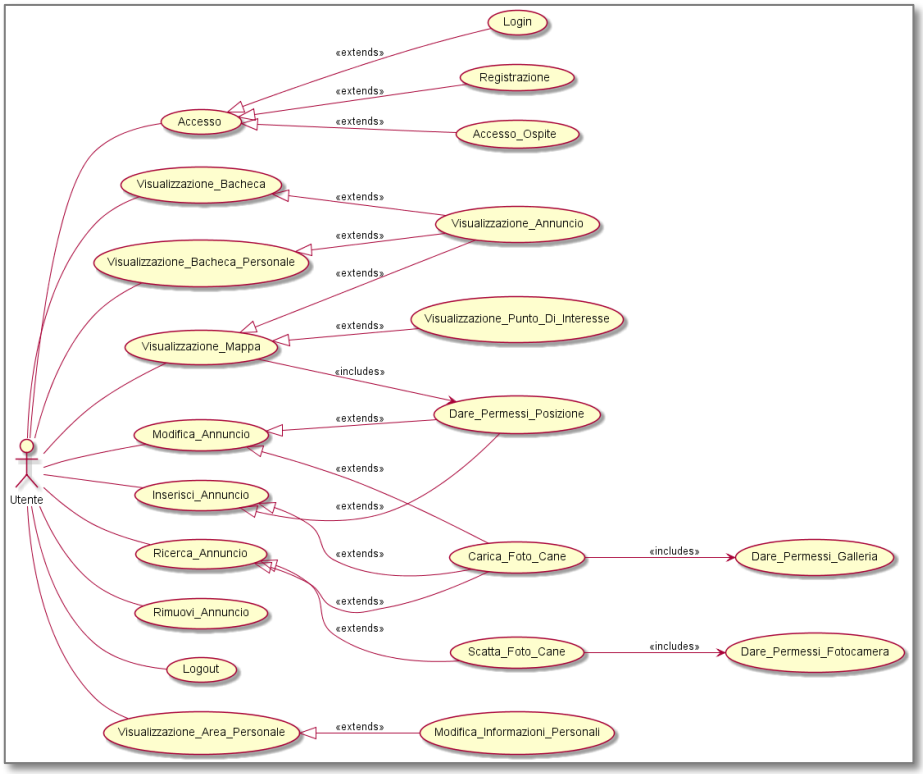

Figura 2. Use Case Diagram

Di seguito verranno spiegati gli elementi presenti nel diagramma in figura 2:

- L'ovale di colore giallo corrisponde ad un caso d'uso specifico, in cui l'utente si può trovare durante l'utilizzo dell'applicazione.
- La parola chiave "includes" viene utilizzata per indicare una relazione tra un caso d'uso chiamato "caso d'uso incluso" (use case posto alla fine della freccia), e un caso d'uso chiamato "caso d'uso base" (use case posto all'inizio della freccia). La relazione che intercorre tra essi stabilisce che, se l'utente si trova in quello base, di conseguenza eseguirà anche quello incluso.
- La parola chiave "extends" viene utilizzata per indicare una relazione tra un caso d'uso chiamato "caso d'uso esteso" (use case posto all'inizio della freccia), e un caso d'uso

chiamato “caso d’uso base” (use case posto alla fine della freccia). La relazione che intercorre tra essi stabilisce che quello esteso può ampliare opzionalmente il comportamento di quello base.

2.2.3. Descrizione Use Cases

Di seguito verranno descritti, in maniera tecnica e dettagliata, i vari casi d’uso mostrati all’interno del diagramma presente nel paragrafo 2.2.2. Per ognuno di essi verrà fornita: una breve descrizione, la precondizione per l’esecuzione del caso d’uso in esame, la sequenza di azioni che l’utente effettua e la postcondizione che si raggiunge al termine del caso d’uso.
È da tener presente che i casi d’uso verranno sviluppati assumendo che il dispositivo sia connesso alla rete internet, infatti, in assenza di essa, il comportamento dell’applicazione potrebbe variare per alcune funzionalità.

Use Case – Accesso

- **Descrizione:** l’utente, aperta l’applicazione, accede ad essa per usufruire delle sue funzionalità;
- **Precondizione:** l’applicazione è disponibile per l’apertura da parte dell’utente;
- **Sequenza:**
    1. l’utente clicca sul logo dell’applicazione;
    2. l’applicazione si avvia;
- **Postcondizione:**
    - utente con accesso effettuato: l’utente ora visualizza il contenuto dell’applicazione;
    - utente senza accesso effettuato: l’utente ora visualizza la schermata per effettuare la registrazione all’applicazione.

Use Case – Login

- **Descrizione:** l’utente effettua l’accesso con il proprio account inserendo le proprie credenziali;
- **Precondizione:** l’utente ha avviato l’applicazione e non ha già l’accesso effettuato;
- **Sequenza:**
    1. l’utente va nella schermata per l’accesso tramite account esistente;
    2. l’utente inserisce l’e-mail corrispondente al suo account;
    3. l’utente inserisce la password corrispondente al suo account;
    4. l’utente clicca sul pulsante “Accedi”;
    5. l’applicazione controlla la correttezza delle credenziali;
- **Postcondizione:**
    - Esito controllo positivo: l’utente entra nell’applicazione;
    - Esito controllo negativo: l’utente visualizza un messaggio di errore.

Use Case – Registrazione

- **Descrizione:** l’utente effettua una registrazione, inserendo e-mail e password obbligatoriamente, avendo la possibilità di inserire anche informazioni personali facoltative;
- **Precondizione:** l’utente ha avviato l’applicazione e non ha già l’accesso effettuato;
- **Sequenza:**
    1. l’utente va nella schermata per la registrazione;
    2. l’utente, se desidera, inserisce le informazioni facoltative;
    3. l’utente inserisce una e-mail valida;
    4. l’utente inserisce una password valida;
    5. l’utente inserisce la conferma della password;

6. l'utente clicca sul pulsante "Registrati";
7. l'applicazione controlla la correttezza delle nuove credenziali;

- **Postcondizione:**
  - Esito controllo positivo: le informazioni vengono salvate sul database e l'utente entra nell'applicazione;
  - Esito controllo negativo: l'utente visualizza un messaggio di errore.

Use Case – Accesso_Ospite

- **Descrizione:** l'utente accede all'applicazione come ospite, non fornendo alcuna informazione personale;
- **Precondizione:** l'utente ha avviato l'applicazione e non ha già l'accesso effettuato;
- **Sequenza:**
  1. l'utente va nella schermata per la registrazione o per l'accesso tramite account esistente;
  2. l'utente clicca su "Accedi come ospite";
- **Postcondizione:** l'utente entra nell'applicazione.

Use case – Logout

- **Descrizione:** l'utente effettua la disconnessione dal suo account;
- **Precondizione:** l'utente ha effettuato l'accesso con un account personale e si trova nella schermata corrispondente alla bacheca personale;
- **Sequenza:**
  1. L'utente clicca sull'icona con i tre punti in alto a destra;
  2. L'utente clicca sul pulsante "Logout";
  3. L'applicazione disconnette l'account dell'utente;
- **Postcondizione:** l'account dell'utente è stato disconnesso e ora l'applicazione mostra la schermata per effettuare l'accesso/registrazione.

Use Case – Visualizzazione_Bacheca

- **Descrizione:** l'utente ha a disposizione una schermata contente la bacheca dove sono presenti gli annunci di smarrimento cani pubblicati;
- **Precondizione:** l'utente ha effettuato l'accesso per entrare nell'applicazione;
- **Sequenza:**
  1. l'utente accede alla schermata contenente la bacheca aprendo l'applicazione o cliccando sull'icona corrispondente ad essa dalla barra di navigazione posizionata in basso;
  2. l'applicazione recupera tutti gli annunci pubblicati presenti sul cloud database;
- **Postcondizione:** l'utente è in grado di visualizzare l'intera bacheca contenente tutti gli annunci finora pubblicati e disponibili sul cloud database.

Use Case – Visualizzazione_Bacheca_Personale

- **Descrizione:** l'utente ha a disposizione una schermata contenete una bacheca personale con tutti gli annunci che ha pubblicato, permettendo una loro facile gestione;
- **Precondizione:** l'utente ha effettuato l'accesso per entrare nell'applicazione;
- **Sequenza:**
  1. l'utente clicca sull'icona corrispondente alla schermata per la bacheca personale posizionata nella barra di navigazione in basso;
  2. l'applicazione recupera tutti gli annunci, che l'utente ha pubblicato, presenti sul cloud database;
- **Postcondizione:** l'utente visualizza tutti i suoi annunci oppure un messaggio che notifica l'assenza di annunci da lui pubblicati.

Use Case – Visualizzazione_Annuncio

- **Descrizione:** l'utente può visualizzare gli annunci pubblicati nell'applicazione più nel dettaglio;
- **Precondizione:** l'utente si trova in una delle tre schermate principali dell'applicazione (Bacheca, Mappa integrata oppure Bacheca personale);
- **Sequenza:**
    1. l'utente clicca sull'annuncio che vuole visualizzare;
- **Postcondizione:** l'utente visualizza tutte le informazioni relative all'annuncio selezionato.

Use Case – Visualizzazione_Mappa

- **Descrizione:** l'utente ha a disposizione una schermata contenente una mappa integrata utile per la visualizzazione di elementi geolocalizzati nelle proprie vicinanze;
- **Precondizione:** l'utente ha effettuato l'accesso per entrare nell'applicazione;
- **Sequenza:**
    1. l'utente clicca sull'icona corrispondente alla schermata per la mappa integrata posizionata nella barra di navigazione in basso;
    2. l'applicazione recupera tutti gli annunci dotati di coordinate geografiche;
- **Postcondizione:** l'utente visualizza la mappa con marker di colore giallo corrispondenti agli annunci a cui è stata fornita una posizione GPS.

Use Case – Visualizzazione_Punto_Di_Interesse

- **Descrizione:** l'utente visualizza sulla mappa due tipologie di punti di interesse nelle proprie vicinanze;
- **Precondizione:** l'utente si trova nella schermata corrispondente alla mappa e ha fornito i permessi di accesso alla posizione del dispositivo;
- **Sequenza:**
    1. l'utente clicca sull'icona in alto a destra;
    2. l'utente seleziona la tipologia di punti di interesse che vuole visualizzare;
    3. l'applicazione ricerca i punti di interesse, della tipologia selezionata dall'utente, nelle vicinanze del dispositivo e li mostra sulla mappa;
    4. l'utente seleziona un punto di interesse;
- **Postcondizione:** l'applicazione mostra una schermata contenente le informazioni relative a quel luogo e, se disponibile, dà la possibilità di visionarlo in modalità street view.

Use Case – Dare_Permessi_Posizione

- **Descrizione:** l'utente, per poter usufruire a pieno di determinate funzionalità legate alla mappa o alla pubblicazione/modifica di annunci, deve fornire i permessi per l'accesso alla posizione del suo dispositivo;
- **Precondizione:** il dispositivo deve essere dotato di GPS e l'utente deve voler accedere alla schermata della mappa o deve voler aggiungere ad un suo annuncio le proprie coordinate geografiche senza aver concesso i permessi per la posizione del dispositivo;
- **Sequenza:**
    1. l'applicazione indirizza l'utente alle impostazioni del dispositivo;
    2. l'utente concede all'applicazione i permessi per l'utilizzo della posizione del dispositivo;
    3. l'utente ritorna nell'applicazione;
- **Postcondizione:** l'utente può utilizzare a pieno la mappa presente nell'applicazione e può aggiungere la posizione GPS ai suoi annunci.

Use Case – Inserisci_Annuncio

- **Descrizione:** l'utente pubblica un nuovo annuncio, fornendo tutte le informazioni necessarie per la sua compilazione;
- **Precondizione:** l'utente si trova nella schermata corrispondente alla bacheca o a quella relativa alla bacheca personale;
- **Sequenza:**
    1. l'utente clicca sull'icona con tre punti in alto a destra;
    2. l'utente clicca sul pulsante "Inserisci annuncio" presente nel menu in alto a destra;
    3. l'applicazione mostra un form per la compilazione dell'annuncio;
    4. l'utente inserisce tutti i dati obbligatori e decide se fornire anche informazioni aggiuntive;
    5. l'utente clicca il pulsante "Salva" in alto a destra;
    6. l'applicazione salva sul cloud database il nuovo annuncio;
- **Postcondizione:** l'utente ha pubblicato un nuovo annuncio, ora disponibile sia nella bacheca principale, sia nella sua bacheca personale.

Use Case – Modifica_Annuncio

- **Descrizione:** l'utente modifica un annuncio che ha pubblicato in precedenza;
- **Precondizione:** l'utente visualizza un proprio annuncio oppure lo individua all'interno della bacheca principale o personale;
- **Sequenza:**
    - Caso in cui l'utente visualizza il proprio annuncio:
        1. l'utente clicca sull'icona dei tre punti in alto a destra;
        2. l'utente clicca sul pulsante "Inserisci annuncio" dal menu in alto a destra;
    - Caso in cui l'utente individua l'annuncio nella bacheca principale o in quella personale:
        1. l'utente seleziona l'annuncio;
        2. l'utente fa swipe dell'annuncio verso destra;
    - Sequenza successiva in comune:
        3. l'applicazione mostra un form compilato con le informazioni appartenenti all'annuncio;
        4. l'utente modifica le informazioni che gli interessano;
        5. l'utente clicca sul pulsante "Salva";
        6. l'applicazione salva le modifiche effettuate all'annuncio sul cloud database;
- **Postcondizione:** le modifiche apportate dall'utente sull'annuncio ora sono visibili sia nella bacheca principale, sia in quella personale.

Use Case – Ricerca_Annuncio

- **Descrizione:** l'utente filtra gli annunci in base alla ricerca che vuole effettuare. Può farlo seguendo due metodologie differenti, la prima che sfrutta una "search bar" e un'altra che utilizza l'intelligenza artificiale;
- **Precondizione:** l'utente si trova nella schermata corrispondente alla bacheca principale;
- **Sequenza:**
    - Caso in cui l'utente utilizza la search bar:
        1. l'utente clicca sul pulsante per selezionare la tipologia di ricerca;
        2. l'applicazione mostra una schermata contenente le ricerche possibili;
        3. l'utente seleziona la ricerca che desidera effettuare;

4. l'utente clicca il pulsante "Salva" in alto a destra;
5. l'applicazione ritorna sulla schermata della bacheca principale;
6. l'utente seleziona la search bar posta in alto;
7. l'utente scrive del testo;
8. l'applicazione applica in automatico il filtraggio selezionato;

- Caso in cui l'utente utilizza l'intelligenza artificiale:
  1. l'utente clicca sull'icona dei tre puntini in alto a destra;
  2. l'utente clicca sul pulsante "Ricerca annuncio con foto" dal menu in alto a destra;
  3. l'applicazione mostra una schermata per la ricerca;
  4. l'utente carica o scatta una foto al cane cliccando sull'icona relativa;
  5. l'applicazione analizza la foto e identifica la razza del cane tramite il modello di machine learning addestrato, fornendo anche una percentuale di affidabilità;
  6. l'utente clicca il pulsante "Cerca" in alto a destra;
  7. l'applicazione torna alla schermata della bacheca applicando il filtro per razza del cane, utilizzando la razza riconosciuta;

- **Postcondizione:** l'utente visualizza la bacheca principale contenente unicamente gli annunci che rispettano il filtro applicato.

Use Case – Rimuovi_Annuncio

- **Descrizione:** l'utente rimuove un annuncio personale;
- **Precondizione:** l'utente visualizza un proprio annuncio oppure lo individua all'interno della bacheca principale o personale;
- **Sequenza:**
  1. Caso in cui l'utente visualizza il proprio annuncio:
    1. l'utente clicca sull'icona dei tre punti in alto a destra;
    2. l'utente clicca sul pulsante "Inserisci annuncio" dal menu in alto a destra;
  2. Caso in cui l'utente individua l'annuncio nella bacheca principale o in quella personale:
    1. l'utente seleziona l'annuncio;
    2. l'utente fa swipe dell'annuncio verso sinistra;
  3. Sequenza successiva in comune:
    3. l'applicazione mostra un alert che avvisa l'utente dell'imminente cancellazione dell'annuncio e lo invita a confermare;
    4. l'utente clicca il pulsante "Rimuovi" per confermare la rimozione dell'annuncio;
    5. l'applicazione rimuove l'annuncio dal cloud database;
- **Postcondizione:** l'annuncio rimosso non è più presente né nella bacheca principale né nella bacheca personale dell'utente.

Use Case – Carica_Foto_Cane

- **Descrizione:** l'utente carica una foto di un cane dalla galleria del proprio dispositivo;
- **Precondizione:** l'utente ha fornito i permessi per l'accesso alla galleria del suo dispositivo e si trova nelle schermate relative all'inserimento, alla modifica o alla ricerca di un annuncio;
- **Sequenza:**
  1. l'utente clicca sull'icona corrispondente alla galleria;
  2. l'applicazione accede alla galleria del dispositivo;
  3. l'utente seleziona la foto che vuole caricare;
  4. l'utente clicca su "Use photo";

5. l'applicazione acquisisce la foto selezionata;

- **Postcondizione:** l'utente può utilizzare la foto selezionata come informazione da inserire in un annuncio o per effettuare la ricerca tramite intelligenza artificiale.

Use Case – Scatta_Foto_Cane

- **Descrizione:** l'utente scatta una foto di un cane usando la fotocamera del proprio dispositivo;
- **Precondizione:** l'utente ha fornito i permessi per l'accesso alla fotocamera del dispositivo e si trova nella schermata corrispondente alla ricerca con intelligenza artificiale;
- **Sequenza:**
    1. l'utente clicca sull'icona corrispondente alla fotocamera;
    2. l'applicazione accede alla fotocamera del dispositivo;
    3. l'utente scatta la foto;
    4. l'utente clicca su "Use photo";
    5. l'applicazione acquisisce la foto scattata;
- **Postcondizione:** l'utente può utilizzare la foto scattata come informazione da inserire in un annuncio o per effettuare la ricerca tramite intelligenza artificiale.

Use Case – Dare_Permessi_Galleria

- **Descrizione:** l'utente, per poter usufruire della galleria del dispositivo, deve fornirne i permessi per l'accesso;
- **Precondizione:** il dispositivo deve avere una galleria multimediale disponibile e l'utente deve voler caricare una fotografia nell'applicazione;
- **Sequenza:**
    1. l'applicazione indirizza l'utente alle impostazioni del dispositivo;
    2. l'utente concede all'applicazione i permessi per l'utilizzo della galleria del dispositivo;
    3. l'utente ritorna nell'applicazione;
- **Postcondizione:** l'utente può accedere alla galleria del dispositivo per acquisire immagini da utilizzare nell'applicazione.

Use Case – Dare_Permessi_Fotocamera

- **Descrizione:** l'utente, per poter usufruire della fotocamera del dispositivo, deve fornirne i permessi per l'accesso;
- **Precondizione:** il dispositivo deve avere una fotocamera disponibile e l'utente deve voler scattare una fotografia nell'applicazione;
- **Sequenza:**
    1. l'applicazione indirizza l'utente alle impostazioni del dispositivo;
    2. l'utente concede all'applicazione i permessi per l'utilizzo della fotocamera del dispositivo;
    3. l'utente ritorna nell'applicazione;
- **Postcondizione:** l'utente può accedere alla fotocamera del dispositivo per scattare fotografie da utilizzare nell'applicazione.

Use Case – Visualizzazione_Area_Personale

- **Descrizione:** l'utente ha a disposizione una schermata contente la sua area personale, dove sono presenti tutte le informazioni che ha fornito all'applicazione;
- **Precondizione:** l'utente ha effettuato l'accesso con un account per entrare nell'applicazione e si trova nella schermata corrispondente alla bacheca personale;
- **Sequenza:**
    1. L'utente clicca sull'icona con tre punti in alto a destra;
    2. L'utente clicca sul pulsante "Account" dal menu in alto a destra;

- **Postcondizione:** l'utente visualizza una schermata contenente tutte le informazioni personali che ha fornito all'applicazione.

Use Case – Modifica_Informazioni_Personali

- **Descrizione:** l'utente modifica le informazioni personali presenti nell'applicazione o ne aggiunge di nuove;
- **Precondizione:** l'utente ha effettuato l'accesso con un account per entrare nell'applicazione e si trova nella schermata corrispondente all'area personale;
- **Sequenza:**
    1. L'utente clicca sul pulsante "Modifica" in basso;
    2. L'applicazione mostra un form dove si possono modificare le informazioni già presenti o aggiungerne di nuove non fornite in precedenza;
    3. L'utente inserisce/modifica le informazioni che desidera;
    4. L'utente clicca sul pulsante "Salva" in alto a destra;
- **Postcondizione:** l'applicazione aggiorna le informazioni personali dell'utente sul cloud database, le quali ora sono disponibili alla visualizzazione dalla schermata corrispondente all'area personale.

2.3. Progettazione del design

In questo paragrafo verrà descritta tutta la fase progettuale riguardante il design dell'applicazione.
Un aspetto particolarmente importante è stato l'utilizzo di Figma. Con l'ausilio di questo strumento è stato possibile creare i principali mockup [40] per strutturare il design delle principali schermate dell'applicazione. Inoltre, Figma mette a disposizione degli sviluppatori una funzionalità per ricreare, in assenza di codice sorgente, un prototipo interattivo dell'applicazione, andando a simulare il processo decisionale di un utente e i suoi movimenti all'interno di essa. Questo ha permesso di applicare migliorie allo scopo di rendere l'applicazione più semplice e intuitiva, andando a migliorare la "user experience" (UX) e la "user interface" (UI).
Di seguito, dopo aver presentato il possibile design per l'icona dell'applicazione in figura 3, verranno mostrati i mockup relativi alle sue schermate principali, utilizzati come base per il design finale, e, per dare l'idea dei prototipi interattivi sviluppati con Figma, verranno presentati dei wireflow [41], i quali ripercorrono i processi decisionali espressi con i prototipi.

Figura 3. Icona dell'applicazione DoggoFinder

2.3.1. Mockup

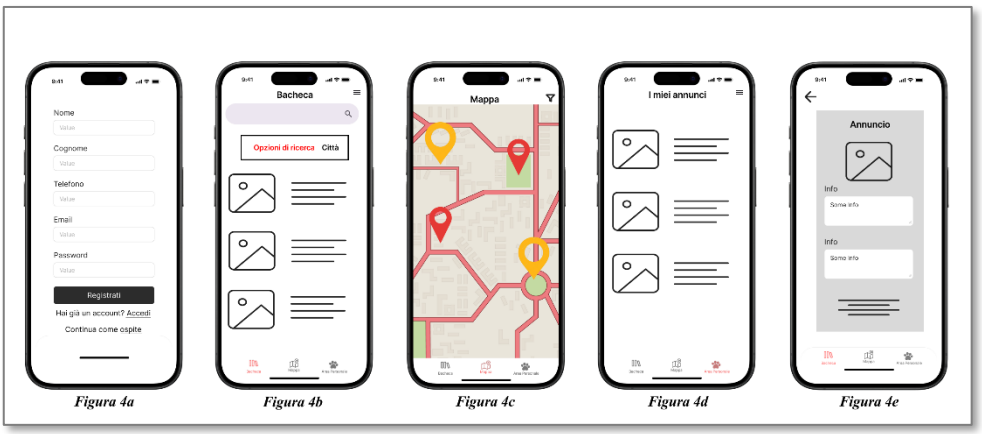

Figura 4. Mockup principali

Nella figura 4 sono presentati i mockup delle cinque schermate principali:

- la figura 4a mostra il design di base relativo alla schermata per la registrazione dell'utente;
- la figura 4b mostra il design di base relativo alla schermata per la bacheca principale, contenente gli annunci di smarrimento pubblicati dai vari utenti;
- la figura 4c mostra il design di base relativo alla schermata per la mappa integrata nell'applicazione, utilizzata per la visualizzazione di annunci geolocalizzati o punti di interesse tramite dei marker;
- la figura 4d mostra il design di base relativo alla schermata per la bacheca personale, contenente gli annunci pubblicati dall'utente che ha effettuato l'accesso;
- la figura 4e mostra il design di base relativo alla schermata per la visualizzazione dell'annuncio di smarrimento, contenente tutte le informazioni relative ad esso, ritenute importanti.

2.3.2. Wireflow

In questo paragrafo, per la presentazione dei principali processi decisionali che può compiere l'utente, verranno utilizzati cinque wireflow, rappresentanti le varie interazioni relative alle cinque schermate principali dell'applicazione. Da tener presente, questi flussi di esecuzione sono stati considerati in fase progettuale e, per alcune caratteristiche, differiscono dai flussi di esecuzione reali, con i quali è stata implementata effettivamente l'applicazione.

Wireflow per la schermata di registrazione

Nel wireflow in figura 5 vengono mostrate le varie interazioni che l'utente può eseguire a partire dalla schermata per la registrazione (figura 5a). Inserendo le informazioni obbligatorie (e-mail e password) e cliccando su "Registrati", l'utente può creare un nuovo account per utilizzare l'applicazione e accedere alla bacheca principale (figura 5c). Inoltre, durante la fase di registrazione, ha anche la possibilità di fornire altre informazioni personali, le quali semplificheranno la compilazione dei suoi annunci. A partire dalla schermata di registrazione è possibile anche accedere come ospite, cliccando su "Ospite", oppure, se si possiede già un account, è possibile navigare verso la schermata per l'accesso tramite credenziali (figura 5b), cliccando su "Accedi". In questa schermata, una volta inserite le credenziali del proprio account, se esse sono valide, permettono l'accesso all'applicazione, mostrando la bacheca principale. Dalla schermata per l'accesso è inoltre possibile ritornare a quella per la registrazione, cliccando su "Registrati" oppure effettuare l'accesso come ospite.

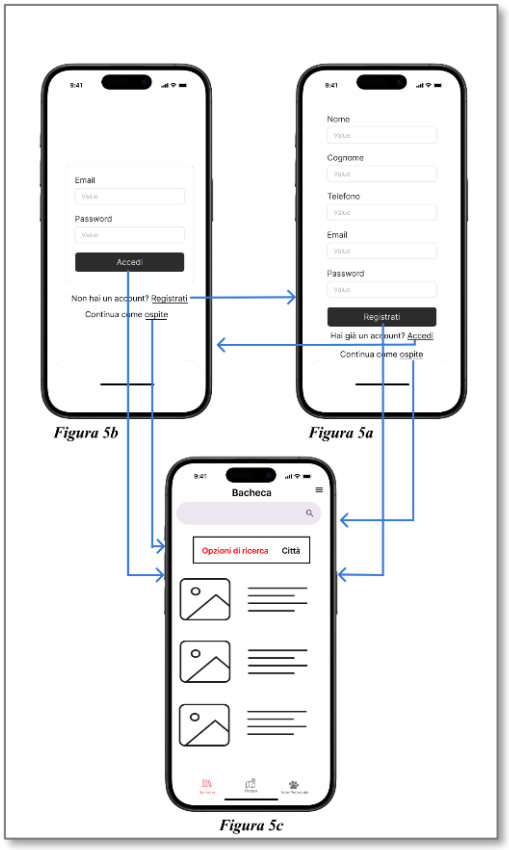

Figura 5. Wireflow per l'accesso

Wireflow per la schermata della bacheca principale

Nel wireflow in figura 6 vengono mostrate le varie interazioni che l'utente può eseguire a partire dalla schermata corrispondente alla bacheca principale (figura 6a). Cliccando sull'annuncio è possibile visualizzare più nel dettaglio le informazioni che lo riguardano (figura 6b). Invece, cliccando sull'icona in alto a destra, si visualizza un menu (figura 6c), contenente vari pulsanti. Selezionando "Inserisci annuncio" l'applicazione mostra un form con vari campi da compilare (figura 6d), per la pubblicazione di un nuovo annuncio. Dopo averlo compilato, l'utente clicca sul pulsante salva posizionato in alto a destra, l'applicazione salva l'annuncio sul database e riporta l'utente alla schermata relativa alla bacheca principale. Selezionando invece il pulsante "Cerca con foto", l'utente può fornire un'immagine di un cane all'applicazione (figura 6e), usufruendo o della fotocamera o della galleria del dispositivo, la quale viene analizzata dal modello di intelligenza artificiale presente e viene identificata, con relativa percentuale di affidabilità, la razza del cane presente nell'immagine. In seguito, cliccando sul pulsante "Cerca", posizionato in alto a destra, il risultato ottenuto dal riconoscimento viene utilizzato come filtro per la ricerca di annunci per cani di quella specifica razza, ritornando quindi alla bacheca principale.

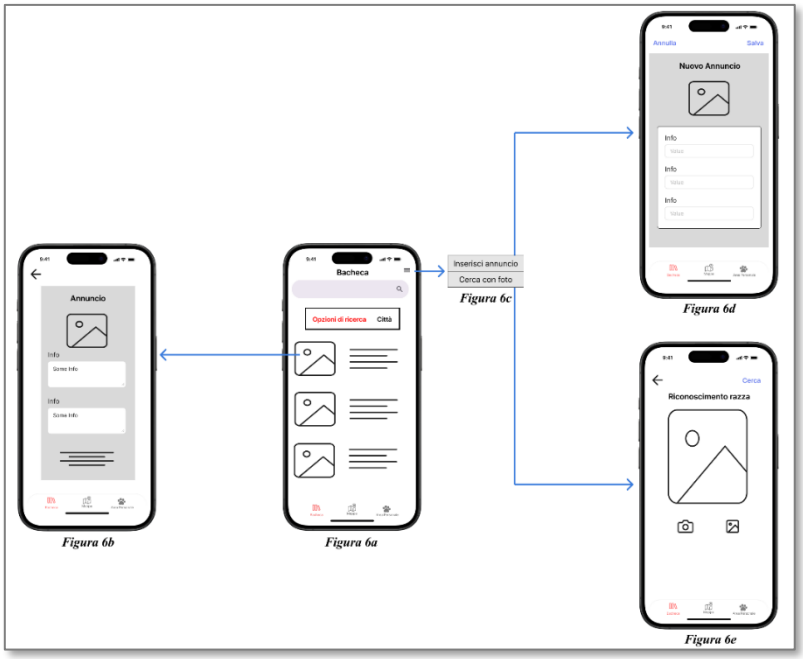

Figura 6. Wireflow per la bacheca principale

Wireflow per la schermata della mappa

Nel wireflow in figura 7 vengono mostrate le varie interazioni che l'utente può eseguire a partire dalla schermata corrispondente alla mappa (figura 7a). Cliccando sull'icona in alto a destra può scegliere da un menu (figura 7b), quali punti di interesse aggiungere come marker sulla mappa. Questi marker sono di due tipologie, per marcare la differenza tra quelli relativi agli annunci dotati di coordinate geografiche e quelli relativi ai punti di interesse selezionati. Cliccando sul marker di un annuncio, viene visualizzato l'annuncio nel dettaglio (figura 7c), mentre, cliccando sul marker di un punto di interesse, vengono visualizzate le informazioni principali relative a quel luogo (figura 7d), per facilitarne il raggiungimento.

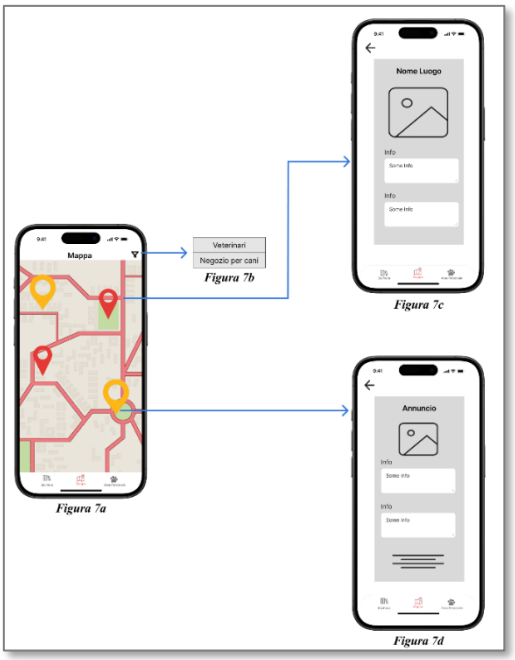

Figura 7. Wireflow per la mappa

Wireflow per la schermata della bacheca personale

Nel wireflow in figura 8 vengono mostrate le varie interazioni che l'utente può eseguire a partire dalla schermata corrispondente alla bacheca personale (figura 8a). Cliccando su uno qualsiasi dei suoi annunci, esso verrà visualizzato nel dettaglio (figura 8b), permettendone una semplice gestione. Cliccando invece sull'icona in alto a destra, l'applicazione mostra un menu contenete vari campi (figura 8c). Selezionando la voce "Inserisci annuncio" l'applicazione mostra un form con vari campi da compilare (figura 8d), per la pubblicazione di un nuovo annuncio. Dopo averlo compilato, l'utente clicca sul pulsante "salva", posizionato in alto a destra. Successivamente, l'applicazione salva l'annuncio sul database e riporta l'utente alla schermata relativa alla bacheca personale. Selezionando la voce "Account", l'applicazione mostra le informazioni personali che l'utente ha fornito (figura 8f), permettendo la loro modifica cliccando il pulsante "Modifica" (figura 8g). Selezionando la voce "Logout", l'utente viene disconnesso dal proprio account e portato alla schermata per la registrazione (figura 8e). Selezionando la voce "Registrati o accedi" l'applicazione mostra la schermata per effettuare la registrazione con un account personale. Bisogna specificare che i flussi di esecuzione relativi alla selezione delle voci "Account" e "Logout" sono disponibili unicamente quando l'utente ha effettuato l'accesso con un account personale, mentre il flusso di esecuzione corrispondente alla selezione della voce "Registrati o accedi" è disponibile unicamente quando l'utente ha effettuato l'accesso come ospite.

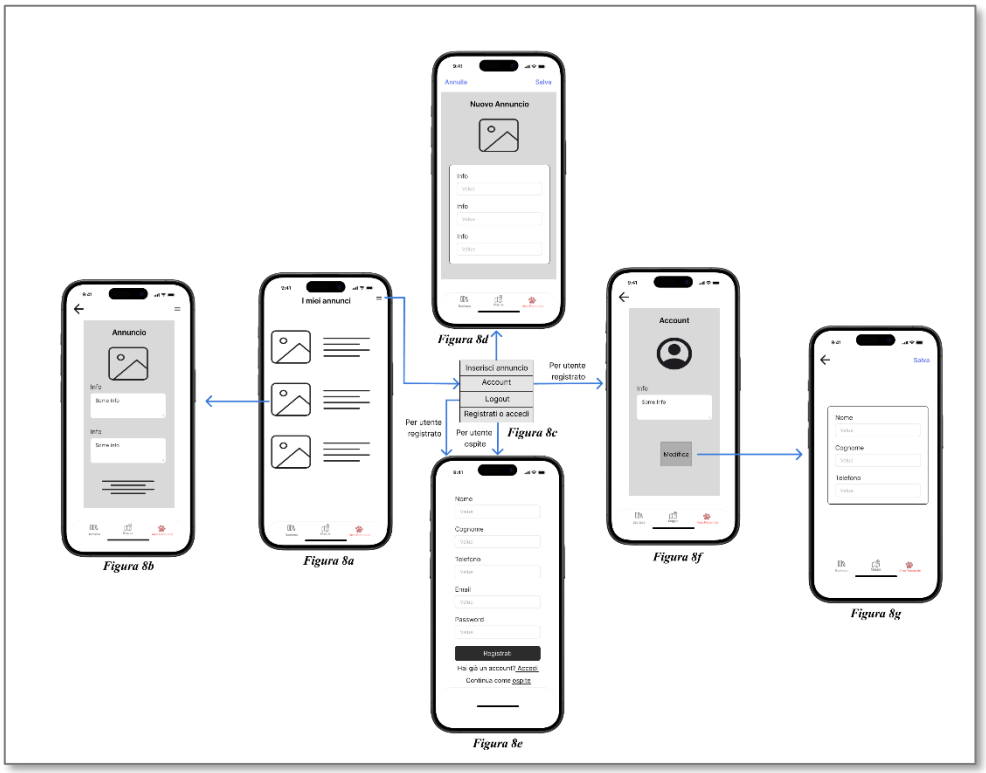

Figura 8. Wireflow per la bacheca personale

Wireflow per la schermata di visualizzazione di un annuncio

Nel wireflow in figura 9 vengono mostrate le varie interazioni che l'utente può eseguire a partire dalla schermata corrispondente ad un annuncio (figura 9a). Questo flusso può essere eseguito solo quando l'utente che sta visualizzando l'annuncio è anche colui che l'ha pubblicato. In questo caso, è disponibile un'icona in alto a destra. Cliccando questa icona compare un menu contenente due pulsanti (figura 9b). Selezionando "Modifica annuncio", l'applicazione mostra un form (figura 9c), con i campi compilati con le informazioni relative

all'annuncio. L'utente ha la possibilità, tramite questo form, di modificare le informazioni dell'annuncio e, cliccando sul pulsante "Salva" in alto a destra, di confermarle. Invece, selezionando dal menu la voce "Rimuovi annuncio", viene mostrato all'utente un'alert dove si richiede la conferma per la rimozione. Confermando, l'annuncio viene rimosso dal database.

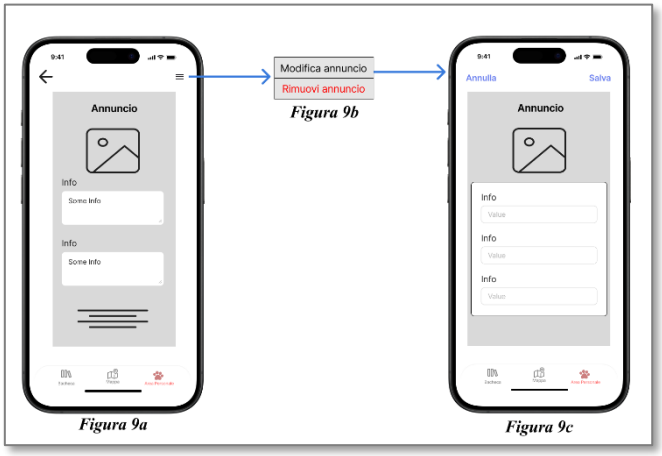

Figura 9. Wireflow per un annuncio

2.4. Progettazione del database

In questo paragrafo verrà illustrata la progettazione legata al database selezionato per la gestione dei dati dell'applicazione sviluppata.
Esso è un cloud database chiamato Firestore, disponibile tramite l'utilizzo della piattaforma Firebase. Questo database è di tipo NoSQL, non relazionale, di conseguenza non è possibile esprimere i vari vincoli e relazioni che intercorrono tra i vari elementi presenti. Esso è composto da una serie di "raccolte" che a loro volta contengono una serie di "documenti" strutturati in maniera tale da avere più "campi", i quali corrispondono alle informazioni che si vogliono salvare. Per facilitarne la gestione, la piattaforma Firebase mette a disposizione dello sviluppatore una console online, la quale permette la manipolazione diretta dei dati salvati all'interno del database Firestore.
In relazione all'applicazione sviluppata, il database è stato strutturato in maniera tale da avere due raccolte chiamate "Utenti" e "Annunci", dove rispettivamente vengono salvate le informazioni riguardanti gli utenti che utilizzano l'applicazione (sia che essi siano registrati, sia che essi utilizzino l'applicazione accedendo come ospiti) e le informazioni riguardanti gli annunci che vengono pubblicati. Di conseguenza queste raccolte contengono una lista di documenti che fanno riferimento rispettivamente agli utenti e agli annunci, i quali a loro volta dispongono di tanti campi quante sono le informazioni che si desiderano salvare per ciascuna tipologia, ritenute rilevanti per il corretto funzionamento dell'applicazione.
Una piccola parentesi va fatta per quanto riguarda il salvataggio in locale di alcuni dati specifici, tramite l'utilizzo di Userdefaults, un'interfaccia propria di Swift che permette di interagire con il defaults system e mette a disposizione un dizionario dei dati per lo storage di informazioni semplici. Ciò è utile per velocizzare l'accesso a informazioni non strutturate in maniera complessa.
Di seguito verrà mostrata una schermata contenente la console messa a disposizione da Firebase (figura 10) e verranno descritte più nel dettaglio le tipologie di documenti salvati sul database.

2.4.1. Console Firebase

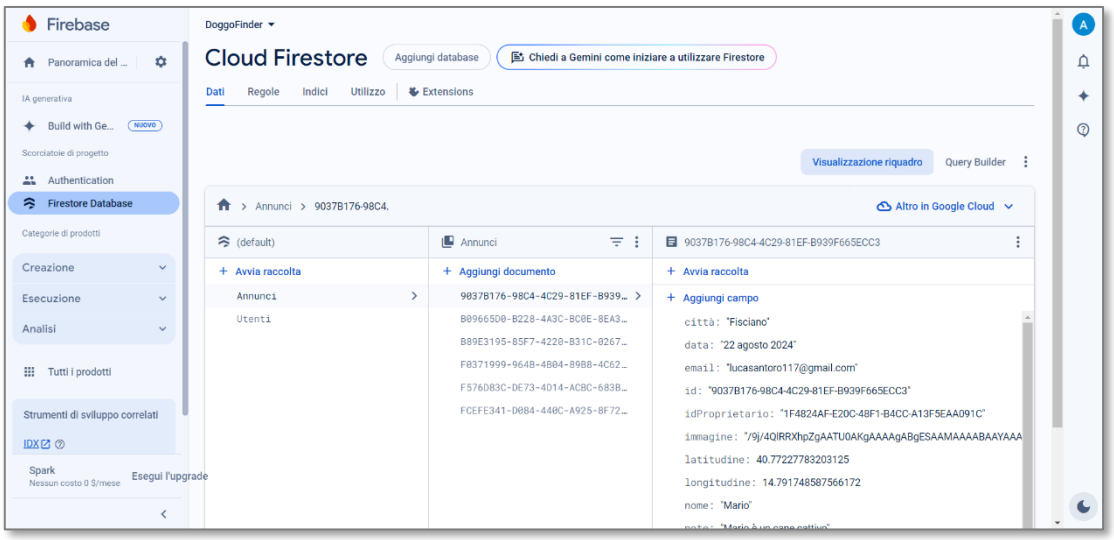

Figura 10. Console Firebase

Dalla console si possono osservare in maniera distinta le tre sezioni che compongono il database Firestore:

- le raccolte, che corrispondono alla colonna di sinistra;
- i documenti, che corrispondono alla colonna centrale;
- i campi, che corrispondono alla colonna di destra.

Per ogni sezione è possibile rimuovere o aggiungere elementi in maniera diretta e inoltre, per quanto riguarda i campi dei vari documenti, è possibile anche modificarne in tempo reale la tipologia di informazione salvata e il suo valore.
La console di Firebase mette a disposizione degli sviluppatori anche la possibilità di eseguire query sul database utilizzando un "Query Builder", uno strumento che permette la costruzione di query in maniera visiva, semplificandone il processo di creazione.

2.4.2. Descrizione tipologie documenti

Documento Utente

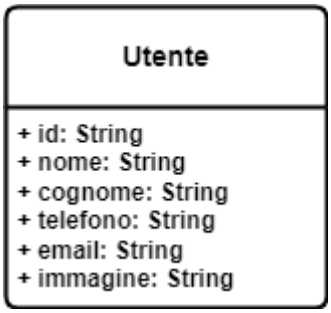

Figura 11. Schema documento Utente

Questa tipologia di documento, schematizzata in figura 11, viene utilizzata per lo storage delle informazioni riguardanti l'utenza dell'applicazione. Ciascun documento di questo tipo rappresenta un utente che ha effettuato l'accesso, sia con account sia come ospite, e ognuno di essi viene salvato nella raccolta "Utenti", sul database Firestore. Ogni documento viene riconosciuto dal sistema in maniera univoca, assegnandogli come identificativo un UUID [42], in formato stringa, generato dall'applicazione e passato al database. Inoltre, ognuno di essi possiede come campi: un id che corrisponde a quello utilizzato come identificativo del documento, usato anche per identificare direttamente l'utente nell'applicazione, il nome, il cognome, il numero di telefono, l'e-mail e l'immagine di profilo, codificata opportunamente in base 64. Tutti i campi sono di tipo stringa. Ad eccezione dell'id, le altre informazioni sono quelle che vengono utilizzate dall'applicazione per precompilare gli annunci che un utente vuole pubblicare.

Documento Annuncio

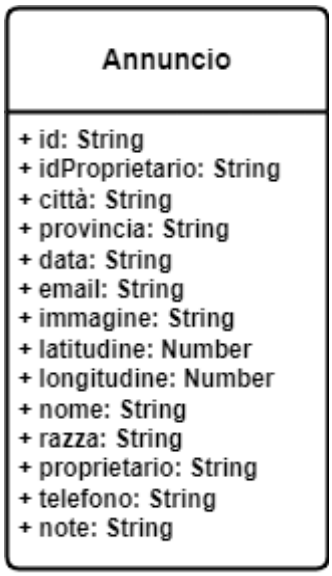

Figura 12. Schema documento Annuncio

Questa tipologia di documento, schematizzata in figura 12, viene utilizzata per lo storage delle informazioni riguardanti gli annunci pubblicati dagli utenti. Ciascun documento di questo tipo corrisponde da un annuncio pubblicato nell'applicazione e ognuno di essi viene salvato nella raccolta "Annunci", sul database Firestore. Ogni documento viene riconosciuto dal sistema in maniera univoca, assegnandogli come identificativo un UUID, in formato stringa, generato dall'applicazione e passato al database. Inoltre, ognuno di essi possiede come campi: un id che corrisponde a quello utilizzato come identificativo del documento, usato anche per identificare direttamente l'annuncio nell'applicazione, idProprietario per permettere la gestione della proprietà dell'annuncio, latitudine e longitudine per permettere l'aggiunta della posizione GPS all'annuncio, città, provincia, data, email, immagine, nome, razza, proprietario, telefono, note come informazioni rilevanti da voler condividere agli altri utenti con la pubblicazione dell'annuncio.

Capitolo 3 – Implementazione e descrizione delle funzionalità

Nel capitolo corrente verrà presentata in dettaglio la fase implementativa riguardante lo sviluppo dell'applicazione oggetto di discussione.
Elemento fondamentale da citare, per quanto riguarda l'aspetto implementativo, è il pattern progettuale utilizzato, ovvero il pattern MVVM [43] (Model−View−ViewModel). Questo pattern è un'architettura software comunemente utilizzata per lo sviluppo di applicazioni dotate di interfaccia utente. Rappresenta una variante del pattern MVC [44] (Model−View−Controller), adattata per sfruttare al meglio le potenzialità del data binding.
Di seguito saranno illustrati i vari punti fondamentali del pattern citato e, successivamente, verrà approfondita la fase implementativa avvenuta, focalizzandosi sulla descrizione delle varie funzionalità sviluppate.

3.1. Pattern MVVM

Gli elementi principali di quest'architettura sono:

- Model
- View
- ViewModel

Il Model è il componente utilizzato per rappresentare i dati e la logica dell'applicazione. Ciò comprende sia la definizione delle entità, sia la gestione della logica di accesso ai dati. Il modello è completamente indipendente dall'interfaccia utente.
La View è il componente utilizzato per la presentazione visiva dei dati all'utente. Essa deve limitarsi a definire l'interfaccia utente e a fare affidamento sul binding per la visualizzazione dei dati.
Il ViewModel è il componente che fa da intermediario tra gli altri due elementi. Contiene la logica di funzionamento per presentare i dati e gestisce il binding tra il Model e la View. Esso è anche responsabile dell'aggiornamento della View in risposta a cambiamenti dei dati del Model e viceversa.

Come già si può evincere, la caratteristica chiave del pattern MVVM è il binding dei dati. Questo permette aggiornamenti automatici in risposta a specifici cambiamenti sia per quanto riguarda i dati, sia per quanto riguarda l'interfaccia utente. Riguardo quest'ultimo aspetto, gli eventi che interessano l'interfaccia grafica, vengono spesso delegati al ViewModel, in maniera tale da mantenere la View priva di logica implementativa. Ciò permette una netta separazione delle responsabilità fra i vari componenti, migliorando la manutenibilità e la testabilità del codice.

3.2. Descrizione fase implementativa

In questo paragrafo verranno analizzati a fondo gli elementi che hanno caratterizzato l'implementazione dell'applicazione DoggoFinder. In particolar modo, si approfondiranno i modelli utilizzati per la rappresentazione dei dati e i vari aspetti implementativi legati alle varie funzionalità di cui l'applicazione è dotata.

3.2.1. Modelli dei dati

I dati utilizzati per lo sviluppo di questa applicazione corrispondono ad informazioni di vario genere, di conseguenza sono stati adottati diversi modelli per permettere la loro rappresentazione in maniera efficiente.
I due modelli cardine dell'applicazione sono "Annuncio" e "Utente", implementate nelle rispettive strutture dati. Essi rappresentano ciò su cui verte l'applicazione, ovvero gli annunci di smarrimento e l'utenza che li pubblica e li consulta.
Il modello **Annuncio** è strutturato in maniera tale da avere un attributo "id", utilizzato per generare un identificativo univoco, con lo scopo di permettere al sistema di riconoscere le varie istanze di Annuncio. Oltre ad esso, questo modello ha numerosi altri attributi di tipo stringa, i quali servono a rappresentare tutte le varie informazioni che riguardano un annuncio. Alcuni di essi sono di tipo "Optional" (opzionali), per non obbligarne l'inserimento da parte dell'utente. Tra i vari attributi riconosciamo anche due attributi di tipo Double? (questa simbologia sta ad indicare che l'attributo è di tipo numerico reale ed è opzionale) che corrispondono alle coordinate geografiche di cui può essere dotato un annuncio per permetterne la sua visualizzazione sulla mappa integrata nell'applicazione. Altro attributo di relativa importanza è "idProprietario", il quale serve a determinare l'utente che ha pubblicato l'annuncio, per permettere una corretta gestione delle proprie aree riservate.
Il modello **Utente** è strutturato anch'esso in maniera tale da avere un attributo "id" utilizzato per generare un identificativo univoco, con lo scopo di permettere al sistema di riconoscere le varie istanza di Utente. Oltre ad esso, questo modello ha vari attributi di tipo stringa. Essi sono opzionali e servono, se l'utente vuole fornire informazioni personali, ad ospitare tali dati.

Oltre a queste due rappresentazioni, ce ne sono altre due di fondamentale importanza, le quali servono a modellare i dati gestiti con la piattaforma Firebase. Questi modelli sono chiamati "DataManager" e "AuthViewModel", implementati nelle rispettive classi, e servono per l'interfacciamento con il database, permettendo la rappresentazione dei dati gestiti da esso.
Il modello **DataManager** si occupa della rappresentazione dei dati salvati nel database. Esso è strutturato in maniera tale da avere quattro attributi di tipo "@Published" [45], in maniera tale da permettere al sistema di essere al corrente di ogni modifica attuata al valore di questi attributi e modificare le varie view di conseguenza. Questi attributi fanno riferimento agli annunci pubblicati e presenti sul database, alle informazioni relative all'utente che ha effettuato l'accesso, agli annunci pubblicati da questo utente e mettono al corrente il sistema se il database è occupato a svolgere delle task oppure è libero di operare.
Invece, per quanto riguarda il modello **AuthViewModel**, esso si occupa dell'aspetto relativo all'accesso/registrazione degli utenti. È legato più in particolare alla piattaforma Firebase rispetto che al database nello specifico, e grazie, alla presenza di due attributi di tipo "@Published", permette al sistema di riconoscere se un utente ha eseguito l'accesso all'applicazione oppure no e, nel caso specifico di accesso eseguito, permette la distinzione tra coloro che l'hanno effettuato con un account personale e coloro che invece sono in modalità ospite.

Oltre a questi modelli presentati, ce ne sono altri, utilizzati per l'implementazione delle varie funzionalità dell'applicazione. Sono presenti altri tre modelli per l'integrazione della mappa e un modello per la gestione della connessione alla rete internet del dispositivo su cui è installata l'applicazione.
In relazione ai modelli legati alla mappa, chiamati "MapManager", "LocationManager" e "Placemark", implementati nelle rispettive classi, essi servono per integrare a pieno la mappa all'interno dell'applicazione e garantire il suo corretto funzionamento.
Il modello **MapManager** si occupa di gestire tutti i cambiamenti relativi alla mappa, infatti è marcato come "@MainActor" [46], garantendo che tutto ciò che lo riguarda avvenga sul thread

principale, evitando problematiche riguardanti la concorrenza o comportamenti imprevisti. Esso, implementando la logica relativa ai dati, non possiede attributi.
Il modello **LocationManager** si occupa della gestione della posizione del dispositivo, avendo come attributi quello relativo ai permessi, quello relativo all'ultima posizione reperibile e quello relativo all'istanza di un CLLocationManager, il quale permette, forniti i permessi, il recupero delle coordinate geografiche. In questo modello è presente anche un attributo di tipo CLGeocoder, il quale permette la gestione del reverse geocoding, funzionalità disponibile nell'applicazione.
Il modello **Placemark** si occupa della gestione dei dati relativi ai marker presenti sulla mappa. Per fare ciò, è dotato di un attributo "id" per garantire il riconoscimento univoco di ogni marker da parte del sistema, di coordinate geografiche di tipo Double per permettere un corretto posizionamento del marker sulla mappa, due attributi stringa per descrivere il marker e un attributo stringa opzionale chiamato "idAnnuncio" che, se è presente, collega il marker al rispettivo annuncio, se invece è assente, sta a significare che il marker non corrisponde ad un annuncio ma ad un punto di interesse scelto dall'utente.
In relazione al modello utilizzato per la gestione della connessione alla rete internet del dispositivo, esso prende il nome di **NetworkMonitor** implementato nella rispettiva classe. Grazie alla presenza di un attributo di tipo NWPathMonitor, un attributo che istanzia una DispatchQueue, per l'esecuzione di blocchi di codice in maniera asincrona, e un attributo booleano, il sistema è in grado di stabilire se il dispositivo su cui sta in esecuzione l'applicazione è connesso alla rete internet oppure no.
Ad ognuno di questi modelli sono associate numerose funzioni, le quali verranno trattate nel sottoparagrafo successivo, per dare un approfondimento riguardante l'implementazione delle funzionalità dell'applicazione.

3.2.2. Approfondimento delle funzionalità

Successivamente all'implementazione dei modelli dei dati sopracitati, è stato possibile proseguire con la fase successiva, ovvero l'implementazione delle varie funzionalità di cui deve essere dotata l'applicazione DoggoFinder. Esse possono essere raggruppate in quattro categorie principali: quelle legate all'avvio dell'applicazione, quelle legate al database, quelle legate al filtraggio degli annunci e quelle legate alla mappa. Di seguito verranno analizzate le varie categorie.

Funzionalità legate all'avvio dell'applicazione
È stata implementata una funzionalità di "launch screen" per migliorare l'esperienza utente durante l'avvio dell'applicazione. Questa schermata iniziale, mostrata in figura 20, che mostra il logo e il nome dell'app, viene visualizzata subito dopo il lancio dell'applicazione. Durante questo breve intervallo, l'applicazione esegue il caricamento dei dati dal database e completa i processi di avvio. Una volta terminato il caricamento, l'utente viene automaticamente indirizzato alla schermata principale dell'applicazione, consentendo una transizione fluida e senza interruzioni. Questa funzionalità non solo rende l'avvio più piacevole, ma comunica anche all'utente che l'applicazione sta caricando, offrendo un'esperienza più professionale e coerente.
Per implementare ciò, è necessaria la configurazione di un file di tipo storyboard, messo a disposizione dall'ambiente di sviluppo Xcode. Nel progetto oggetto di analisi, questo file prende il nome di "Launch Screen.storyboard". L'IDE permette una sua configurazione semplificata e intuitiva, grazie all'ausilio di numerosi elementi grafici che vanno a sostituire complessi script di codice. Per la sua implementazione, inoltre, bisogna configurare opportunamente anche alcune impostazioni del progetto, tra cui la corrispondente chiave presente nel file "Info.plist". In aggiunta, la schermata realizzata con questa tecnica non deve contenere elementi difficili da caricare, poiché questo allungherebbe i tempi di avvio dell'applicazione.

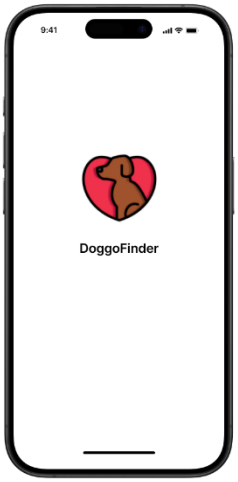

Figura 13. Launch Screen

Funzionalità legate al database

Sono state implementate diverse funzionalità che interagiscono con il database. Prima fra tutte è il recupero dei dati presenti al suo interno. Ciò avviene tramite l'utilizzo di due funzioni chiamate "fetchAnnunci()" e "fetchInfoUtente()", appartenenti alla classe "DataManager", le quali servono rispettivamente a recuperare i dati relativi agli annunci pubblicati e i dati, se presenti, dell'utente che ha effettuato l'accesso. L'utilizzo di queste due funzioni implementate, oltre a quelle per l'aggiunta specifica di informazioni dell'utente in base alla tipologia di accesso effettuato, consente all'applicazione di aggiornarsi in maniera continua e in tempo reale. Infatti, sia che si tratti di inserimento/modifica/rimozione di un annuncio, ricarica della bacheca principale o modifica delle informazioni personali, esse vengono sempre richiamate per mantenere l'applicazione costantemente aggiornata. Parlando appunto della manipolazione dei dati riguardanti gli annunci, sono state implementate anche varie funzioni per la manipolazione dei documenti sul database per appunto permettere l'inserimento, la modifica o la rimozione di un annuncio. In aggiunta a ciò, sono presenti anche funzioni per la gestione di casi particolari come il passaggio da utente ospite ad utente con account personale. Infatti, in questo caso, se da ospite ci si registra con un nuovo account oppure si accede con uno già esistente, tutti gli annunci pubblicati passano sull'account con cui si è effettuato il nuovo accesso.

Non relative propriamente al database Firestore, ma legata alla piattaforma Firebase, ci sono le funzionalità riguardanti l'autenticazione dell'utenza. Infatti, le due schermate per l'accesso e la registrazione sono strutturate in maniera tale da inviare i dati alla piattaforma, la quale verifica la loro adeguatezza, e in caso di errore, le view si aggiornano di conseguenza, avvertendo l'utente del problema. Tutto ciò è stato possibile grazie all'utilizzo di due funzioni appartenenti al framework Firebase chiamate "Auth.auth().signIn(withEmail: , password:)" e "Auth.auth().createUser(withEmail: , password:)", rispettivamente per l'accesso tramite account esistente e per la registrazione dell'utente. Legato sempre all'aspetto dell'autenticazione, è possibile ricavare l'e-mail utilizzata per effettuare l'accesso, controllare se l'utente è loggato ed effettuare il logout dal proprio account utilizzando una proprietà e una funzione appartenenti sempre allo stesso framework, ovvero "Auth.auth().currentUser" e "Auth.auth().signOut()".

Per concludere la discussione sul database, è importante precisare che, essendo il database utilizzato di tipo NoSQL, quindi non relazionale, non offre la possibilità di esprimere vincoli sui dati a livello di schema. Di conseguenza, tali vincoli sono stati implementati a livello

applicativo. In particolare, è stato garantito che un utente possa avere zero o più annunci pubblicati, i quali avranno come "idProprietario" l'identificativo univoco dell'utente che li ha creati. Inoltre, è stato imposto che ogni annuncio debba necessariamente avere un unico proprietario.

Funzionalità legate al filtraggio degli annunci

Aspetto rilevante dell'applicazione sono le funzionalità relative al filtraggio degli annunci nella bacheca principale, per permettere una ricerca più veloce da parte dell'utente.

In primo luogo, è possibile ordinare gli annunci per data di pubblicazione, dando la possibilità di visualizzare partendo dagli annunci più recenti, o da quelli più datati. Questa funzionalità può essere utilizzata in contemporanea con altre opzioni di filtraggio per facilitare la ricerca di un determinato annuncio. Per la sua implementazione è stato sfruttato l'attributo relativo alla data di pubblicazione di cui dispongono tutti gli annunci. Infatti, il vettore contenente gli annunci visualizzati in bacheca, viene ordinato, utilizzando la funzione sort(by:), per data di pubblicazione crescente o decrescente, in base alle preferenze dell'utente. Questa funzionalità è disponibile anche per la bacheca personale.

Per quanto riguarda invece le opzioni di filtraggio, esse possono essere di due tipi, manuali o tramite intelligenza artificiale. La prima tipologia sfrutta una "search bar", componente implementato utilizzando elementi già integrati nell'ambiente di Xcode e contenuti nella struttura dati "SearchBar", e un'enumerazione contenente le opzioni che l'utente può scegliere per il filtraggio, ovvero la città di smarrimento segnalata nell'annuncio e la razza del cane smarrito. L'utente, selezionata l'opzione di filtraggio, inserisce manualmente nella search bar la ricerca che vuole effettuare e la bacheca mostrerà in automatico gli annunci filtrati, sfruttando una funzione chiamata "filtro", alla quale vengono passati come parametri tutti gli annunci presenti sul database, uno per volta, e l'enumerazione contenente le opzioni di filtraggio, per stabilire se l'annuncio debba essere visualizzato oppure no, in base alle azioni compiute dall'utente. La seconda tipologia, invece, sfrutta il modello di machine learning addestrato con Create ML e implementato con Core ML e Vision. Questo modello è stato addestrato utilizzando un dataset [47] composto da 51.421 immagini, suddiviso in training set (50.921 immagini), utilizzato per l'addestramento vero e proprio del modello, e testing set (500 immagini), utilizzato per testare l'efficienza del modello addestrato. Ogni immagine inclusa in questo dataset è associata ad una delle dieci categorie di razze canine scelte. Di seguito è riportata, in tabella 2, la suddivisione delle varie immagini del dataset fra le categorie.

Tabella 2. Distribuzione del dataset tra le varie categorie scelte

Razza	Training Set	Testing Set
Beagle	4551 immagini	50 immagini
Border Collie	4730 immagini	50 immagini
Bulldog	9091 immagini	50 immagini
Chihuahua	4407 immagini	50 immagini
Husky	4458 immagini	50 immagini

Jack Russell	5231 immagini	50 immagini
Labrador	4863 immagini	50 immagini
Rottweiler	4682 immagini	50 immagini
San Bernardo	4099 immagini	50 immagini
Yorkshire	4809 immagini	50 immagini

Sono state considerate solo dieci tipologie di razze canine per dare un esempio di funzionamento. Per aumentarne il numero e garantire risultati soddisfacenti bisognerebbe utilizzare altre tipologie di reti neurali per l'addestramento di modelli più complessi.
Continuando il discorso relativo al filtraggio tramite intelligenza artificiale, questo modello è stato utilizzato grazie a Core ML e viene implementato tramite la struttura dati "Classifier" e la classe "ImageClassifier", dove appunto è presente la logica per classificare le immagini, fornendo anche una confidenza riguardo al risultato ottenuto. Ciò, aggiungendo l'image picker, implementato per il recupero e la gestione delle immagini che l'utente fornisce, e l'utilizzo delle classi integrate AVCaptureDevice e PHPhotoLibrary per la gestione dei permessi relativi rispettivamente all'accesso alla fotocamera e all'accesso alla galleria multimediale del dispositivo, è stato possibile implementare un filtraggio che, fornita un'immagine, classifica la razza del cane presente al suo interno e, settando le opzioni di filtraggio e il campo di ricerca della search bar rispettivamente con "razza del cane" e il risultato della classificazione, filtra gli annunci presenti nella bacheca principale, in maniera analoga a quella manuale sopracitata.
Questa funzionalità di riconoscimento razza tramite intelligenza artificiale è stata sfruttata anche, in relazione alla compilazione di un annuncio, per fornire, dopo aver aggiunto l'immagine del cane smarrito, in automatico la razza del cane e compilare di conseguenza il relativo campo. Ciò semplifica e velocizza la compilazione da parte dell'utente il quale però, nel caso l'identificazione non sia corretta, può sempre modificare manualmente il campo corrispondente.

Funzionalità legate alla mappa
Per quanto riguarda le funzionalità legate alla mappa integrata nell'applicazione, esse sono principalmente legate alla geolocalizzazione di annunci o punti di interesse. Infatti, dopo aver gestito i permessi grazie alla classe LocationManager, sfruttando la classe Placemark è possibile creare dei marker da posizionare sulla mappa. Con gli annunci, l'implementazione è semplificata poiché basta fornirne le coordinate geografiche durante la sua compilazione. Ciò è più complesso per i punti di interesse. Questo aspetto necessita di aggiornamenti in tempo reale, poiché l'utente deve visualizzare unicamente i punti di interesse scelti nelle sue vicinanze. Per implementare questo comportamento, nella classe MapManager, sono presenti due funzioni. La prima, chiamata "searchPlaces(modelContext: , searchText: , visibleRegion:)" , fornito il punto di interesse come parametro, richiede, tramite la classe MKLocalSearch, una ricerca sfruttando i dati di Apple Maps. I risultati vengono filtrati in base alla posizione dell'utente e mostrati sulla mappa come marker. La seconda, chiamata removeSearchResults(modelContext:), rimuove dalla mappa tutte le ricerche effettuate dall'utente. Per limitare al solo campo di interesse dell'applicazione le ricerche che l'utente

può eseguire, ne sono state consentite solo due tipologie, una relativa ai negozi per cani e una relativa ai veterinari.
Ultima funzionalità implementata, riguardante la mappa, è la possibilità di visualizzare punti di interesse apparsi sulla mappa, in seguito ad una ricerca, con la modalità street view. Questo è stato possibile grazie all'utilizzo della classe MKLookAroundScene, la quale richiede la visione di un luogo in modalità street view e, se questa è disponibile, crea una scena apposita per la sua visualizzazione.

C'è da aggiungere anche che, ad ognuna di queste categorie, sono state integrate, grazie all'implementazione della classe NetworkMonitor, piccole funzionalità per la gestione di attività dell'utente in assenza di connessione alla rete Internet. Tra queste funzionalità abbiamo vari alert per avvisare l'utente, messaggi di aggiornamento o blocco per specifiche funzionalità.

Capitolo 4 – Conclusioni e prospettive future

Nel capitolo corrente verrà fornita, come conclusione della discussione, una sintesi del progetto sostenuto, con l'aggiunta di possibili spunti per il miglioramento dell'applicazione e l'integrazione di nuove funzionalità per arricchirla e renderla sempre più funzionale e accessibile per i nuovi utenti.
Quindi, per sintetizzare il tutto, l'applicazione sviluppata durante il progetto del tirocinio Swift App Development Bootcamp, chiamata DoggoFinder, rientra tra le nuove applicazioni che sfruttano tecnologie innovative, con l'obiettivo di rendere sempre più efficienti e semplificate le attività, le quali, in passato, richiedevano notevoli impegni per la loro risoluzione, a causa di metodologie considerate al giorno d'oggi obsolete. Lo scopo con cui nasce è quello di offrire supporto a coloro che smarriscono il proprio cane. Consente la pubblicazione di annunci di smarrimento e il consulto di una bacheca centralizzata condivisa, per il reperimento di informazioni in tempo reale. L'app è dotata di funzionalità legate all'intelligenza artificiale, per ottimizzare la ricerca di annunci all'interno della bacheca condivisa, e di funzionalità legate alla geolocalizzazione, per la visualizzazione di annunci o punti di interesse nelle vicinanze dell'utente. Queste ultime funzionalità nascono con lo scopo di aiutare coloro che ritrovano un cane smarrito a ricongiungerlo al suo legittimo proprietario o ad almeno, nel caso ce ne fosse bisogno, fornirgli primo soccorso. Inoltre, l'applicazione mette a disposizione varie funzionalità di accesso e la possibilità di fornire dati personali. Ciò porta a semplificare il processo di compilazione degli annunci per gli utenti che forniscono i propri dati.
Tutto ciò sopracitato si appoggia su un cloud database, fornito dalla piattaforma Firebase, che permette la gestione della persistenza dei dati dell'applicazione.
Fra gli aspetti da migliorare abbiamo proprio la gestione del database, la quale può essere ottimizzata per gestire tipologie di dati più complessi e rendere l'applicazione più efficiente. Ciò potrebbe comportare la scelta di passare da un database NoSQL, non relazionale, ad uno di tipo SQL, relazionale, dove è possibile esprimere direttamente i vincoli sui dati. Altro aspetto che porterebbe ad un aumento sostanziale della qualità dell'applicazione riguarda il modello di intelligenza artificiale addestrato. Infatti, utilizzando reti neurali più all'avanguardia, è possibile addestrare modelli molto più efficienti e complessi, i quali riescono a classificare molte più razze canine. Ciò sarebbe un miglioramento sostanziale per la funzionalità di riconoscimento delle razze tramite immagini. In aggiunta, un altro aspetto migliorabile riguarda la mappa integrata. Infatti, le tipologie di punti di interesse mostrabili all'utente sono limitate e la ricerca non è efficiente. Queste due caratteristiche potrebbero essere migliorate, permettendo all'utente di avere un'esperienza più ampia.
Per quanto riguarda possibili sviluppi futuri, c'è, in primo luogo, l'implementazione di una funzionalità di messagistica, che permetta il contatto diretto tra vari utenti. Ciò porterebbe ad un miglioramento sostanziale relativo allo scambio di informazioni e, di conseguenza, alla risoluzione dei casi di smarrimento. Successivamente, si potrebbe ampliare il target dell'applicazione, aggiungendo al suo obiettivo oltre quello di supporto nei casi di smarrimento cani, anche quello di supporto nei casi di smarrimento gatti, aumentando in maniera esponenziale la sua utilità.
Inoltre, riprogettando il layout e adattandolo alle nuove aggiunte, rendendolo molto più intuitivo e rilassante per l'utente, permetterebbe di raggiungere netti miglioramenti riguardanti la user experience e la user interface.
In conclusione, l'applicazione, sviluppata durante il corso di questo progetto, può considerarsi un ottimo punto di partenza per lo sviluppo di sempre più sofisticati software che aiutino coloro i quali purtroppo smarriscono i propri animali domestici, membri importanti della propria famiglia, integrando tecnologie sempre più all'avanguardia, in modo tale da rendere il processo di ritrovamento sempre più semplice e con un tasso di successo sempre più elevato.

Bibliografia e sitografia

[1] Unisa, «Ingegneria informatica/Focus sulla didattica» Maggio 2024. [Online]. Available: https://corsi.unisa.it/ingegneria-informatica/focus?id=1230.

[2] Apple Inc., «Documentazione linguaggio Swift» [Online]. Available: https://developer.apple.com/documentation/swift/.

[3] Apple Inc., «Macbook Air» [Online]. Available: https://www.apple.com/it/macbook-air/.

[4] Apple Inc., «iPad Air» [Online]. Available: https://www.apple.com/it/ipad-air/.

[5] Apple Inc., «Documentazione ambiente di sviluppo Xcode» [Online]. Available: https://www.apple.com/it/ipad-air/.

[6] Figma Inc., «Figma Learn» [Online]. Available: https://help.figma.com/.

[7] GitHub Inc., «GitHub Docs» [Online]. Available: https://docs.github.com/.

[8] Apple Inc., Develop in Swift Fundamentals, Apple Inc. - Education, 2024, p. 843.

[9] Apple Inc., «Documentazione framework SwiftUI» [Online]. Available: https://developer.apple.com/documentation/swiftui/.

[10] Apple Inc., «Documentazione framework UIKit» [Online]. Available: https://developer.apple.com/documentation/uikit/.

[11] Apple Inc., «Documentazione framework AppKit» [Online]. Available: https://developer.apple.com/documentation/appkit/.

[12] Apple Inc., «Documentazione framework SwiftData» [Online]. Available: https://developer.apple.com/documentation/swiftdata.

[13] Apple Inc., «Documentazione framework Core Data» [Online]. Available: https://developer.apple.com/documentation/coredata.

[14] Apple Inc., «Documentazione framework Create ML» [Online]. Available: https://developer.apple.com/documentation/createml.

[15] Apple Inc., «Documentazione framework Core ML» [Online]. Available: https://developer.apple.com/documentation/coreml.

[16] Apple Inc., «Documentazione Framework Vision» [Online]. Available: https://developer.apple.com/documentation/vision.

[17] Apple Inc., «Documentazione framework Natural Language» [Online]. Available: https://developer.apple.com/documentation/naturallanguage.

[18] Apple Inc., «Documentazione framework Speech» [Online]. Available: https://developer.apple.com/documentation/speech.

[19] Apple Inc., «Documentazione framework Sound Analysis» [Online]. Available: https://developer.apple.com/documentation/soundanalysis.

[20] Apple Inc., «Documentazione framework Core Image» [Online]. Available: https://developer.apple.com/documentation/coreimage.

[21] Apple Inc., «Documentazione framework MapKit» [Online]. Available: https://developer.apple.com/documentation/mapkit.

[22] Apple Inc., «Documentazione framework Core Location» [Online]. Available: https://developer.apple.com/documentation/corelocation/.

[23] Apple Inc., «Documentazione framework Network» [Online]. Available: https://developer.apple.com/documentation/network.

[24] Apple Inc., «Documentazione framework CFNetwork» [Online].
Available: https://developer.apple.com/documentation/cfnetwork.

[25] Apple Inc., «Documentazione framework PhotoKit» [Online].
Available: https://developer.apple.com/documentation/photokit.

[26] Google LLC, «Documentazione Firebase» [Online].
Available: https://firebase.google.com/docs.

[27] Google LLC, «Documentazione database Firestore» [Online].
Available: https://firebase.google.com/docs/firestore.

[28] Google LLC, «Documentazione database realtime Firebase» [Online].
Available: https://firebase.google.com/docs/database.

[29] Google LLC, «Documentazione database relazionale Firebase» [Online].
Available: https://firebase.google.com/docs/data-connect.

[30] Google LLC, «Documentazione Firebase Authentication» [Online].
Available: https://firebase.google.com/docs/auth.

[31] Google LLC, «Documentazione Firebase App Check» [Online].
Available: https://firebase.google.com/docs/app-check.

[32] Google LLC, «Documentazione Firebase Cloud Messaging» [Online].
Available: https://firebase.google.com/docs/cloud-messaging/.

[33] Google LLC, «Documentazione Firebase Remote Config» [Online].
Available: https://firebase.google.com/docs/remote-config/.

[34] Google LLC, «Documentazione Google Analytics» [Online].
Available: https://firebase.google.com/docs/analytics.

[35] Google LLC, «Documentazione Firebase Crashlytics» [Online].
Available: https://firebase.google.com/docs/crashlytics.

[36] Google LLC, «Documentazione Firebase Test Lab» [Online].
Available: https://firebase.google.com/docs/test-lab.

[37] Google LLC, «Documentazione Firebase A\B Testing» [Online].
Available: https://firebase.google.com/docs/ab-testing.

[38] Glue Labs, «Documento dei Requisiti» [Online]. Available: http://bit.ly/3XxGSgY.

[39] Wikipedia, «Definizione di Use Case Diagram» [Online].
Available: https://en.wikipedia.org/wiki/Use_case_diagram.

[40] Wikipedia, «Definizione di Mockup» [Online].
Available: https://en.wikipedia.org/wiki/Mockup.

[41] UXtweak, «Definizione di Wireflow» [Online].
Available: https://www.uxtweak.com/ux-glossary/wireflow/.

[42] Apple Inc., «Documentazione struttura UUID» [Online].
Available: https://developer.apple.com/documentation/foundation/uuid.

[43] Microsoft Corporation, «Documentazione pattern MVVM» [Online].
Available: https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm.

[44] Microsoft Corporation, «Documentazione pattern MVC» [Online].
Available: https://learn.microsoft.com/it-it/aspnet/core/mvc.

[45] Apple Inc., «Documentazione struttura Published» [Online].
Available: https://developer.apple.com/documentation/combine/published.

[46] Apple Inc., «Documentazione classe MainActor» [Online].
Available: https://developer.apple.com/documentation/swift/mainactor.

[47] Kaggle, «Find Open Datasets» [Online]. Available: https://www.kaggle.com/datasets.

Indice delle figure

Indice delle tabelle