

# Programmazione di Dispositivi Mobili - Progetto Beagle

404 Error Not Founders

Leuzzi Nicoletta 893975

Tauriello Gianmarco 866479

Zheng Lei Maurizio 866251

Mandelli Marco 890078



A.A. 2024/2025

<b>1. Introduzione.....</b>	<b>2</b>
<b>2. Tecnologie utilizzate.....</b>	<b>2</b>
<b>3. Design.....</b>	<b>3</b>
3.1 Progettazione UI.....	3
3.3 Palette colori.....	3
3.4 Componenti UI principali.....	4
<b>4. Architettura.....</b>	<b>5</b>
4.1 Pattern architetturale.....	6
4.2 Design pattern utilizzati.....	8
<b>5. Funzionalità.....</b>	<b>8</b>
5.1 Autenticazione.....	8
5.2 Login.....	8
5.3 Registrazione.....	10
Validazioni e controlli.....	10
5.4 Reset Password.....	11
Funzionalità principali.....	11
5.5 Home (Chat con AI).....	12
5.5 Cronologia chat.....	13
5.6 Profili animali.....	14
5.7 Impostazioni.....	15
<b>6. Sviluppi futuri.....</b>	<b>16</b>

# 1. Introduzione

Beagle è un'app mobile sviluppata nativamente per Android. È pensata per i proprietari e le proprietarie di animali domestici, per ora cani e gatti, che cercano un supporto immediato e personalizzato sulla salute dei loro amici a quattro zampe.

L'app è incentrata su una chat con intelligenza artificiale, che offre risposte a domande e curiosità, oltre a suggerimenti personalizzati.

Con Beagle, puoi:

- Accedere facilmente tramite registrazione, login o autenticazione con Google.
- Creare un profilo dedicato al tuo animale, completo di informazioni anagrafiche e dati personalizzati.
- Consultare la cronologia delle chat per ritrovare senza sforzo le conversazioni passate.

L'obiettivo di questo progetto è fornire un supporto a 360 gradi per gestione dei propri pet.

---

## 2. Tecnologie utilizzate

- **Android Studio:** IDE utilizzato per sviluppare l'applicazione Android in **Java**.
- **GitHub:** strumento di versionamento e collaborazione tra sviluppatori. La repository è stata strutturata nel seguente modo:
  - **Branch main:** branch principale che contiene le versioni stabili dell'app prodotte durante lo sviluppo, contrassegnate con un tag.
  - **Branch dev:** branch di lavoro principale, costantemente aggiornato al termine dei task completati sui branch personali. Viene utilizzato anche per gli ultimi cambiamenti prima di una release su master.
  - **Branch personali (Login / chat / profilo):** branch personali, impiegati per lo sviluppo delle feature assegnate ad ognuno. Il branch di chat è stato successivamente utilizzato da due persone per semplificare la collaborazione. Una volta completato il lavoro, il contenuto viene unito (merge) in dev, risolvendo eventuali conflitti.
- **Firebase:** piattaforma di Google che gestisce sia l'autenticazione (login, registrazione, Google Sign-In) sia la memorizzazione profili degli animali nel **Realtime Database**.

- **Room:** libreria di gestione del database locale.
  - **Google Sign-In:** servizio esterno fornito da Google che permette agli utenti di autenticarsi in maniera semplice e sicura con il proprio account Google.
- 

## 3. Design

### 3.1 Progettazione UI

Per la progettazione delle interfacce utente di **Beagle** ci siamo confrontati principalmente in **videochiamata**, realizzando soltanto un **disegno a schermo** per abbozzare degli elementi nelle schermate e la logica di navigazione.

Per definire lo stile visivo ci siamo ispirati a interfacce di **app già esistenti**. Lo stile grafico dell'app si ispira alle linee guida di **Material Design 3**, riadattando i componenti messi a disposizione (Buttons, Cards, TextFields, NavigationBar).

Abbiamo implementato due varianti di tema (Light e Dark Mode), la scelta avviene in automatico sulla base delle impostazioni di sistema ed è modificabile nelle impostazioni dell'app.

---

### 3.3 Palette colori

I colori principali dell'applicazione sono stati scelti riprendendo le tonalità del **logo di Beagle**, così da mantenere identità e riconoscibilità visiva. In particolare:

- **Blue Stone (#00625B):** colore primario elegante e distintivo, utilizzato per elementi principali dell'interfaccia. Garantisce una buona accessibilità visiva anche per utenti con diverse forme di daltonismo.
- **Gray Hue (#DDE4E2):** tonalità neutra impiegata come colore di supporto per sfondi secondari, highlight e separatori, contribuendo a mantenere un aspetto armonioso e bilanciato.
- **Black Squeeze (#F4FBF9) e Nero (#000000):** utilizzati rispettivamente per sfondi chiari e per testi o contrasti principali, così da assicurare leggibilità e chiarezza in ogni contesto.

Questa palette è stata scelta per offrire un'interfaccia **chiara, piacevole** e per trasmettere **tranquillità**, mantenendo sempre alta la leggibilità in tutte le modalità d'uso.

---

## 3.4 Componenti UI principali

- **Buttons**: progettati seguendo le linee guida di Material Design per garantire reattività e intuitività.
  - **RecyclerView**: utilizzata per gestire elenchi dinamici (es. cronologia chat), con supporto a inserimento, rimozione e aggiornamento in tempo reale.
  - **TextView**: ogni elemento visualizzato in lista è incapsulato in una TextView.
  - **SnackBar**: impiegate per mostrare messaggi di feedback immediati (es. conferma di login, errore di autenticazione) senza interrompere l'esperienza utente.
  - **Dialogs e AppBar**: usati per migliorare la navigazione e rendere più omogenee le diverse sezioni dell'app.
- 

## 4. Architettura

L'applicazione **Beagle** segue l'approccio *single-activity multiple-fragments*, dove le diverse schermate sono gestite da **fragment** e racchiuse in due activity principali:

- **WelcomeActivity**: gestisce i fragment dedicati all'autenticazione (login, registrazione, reset password).
- **ChatActivity**: gestisce i fragment dedicati alla chat con l'AI, la cronologia, il profilo dei pet e le impostazioni

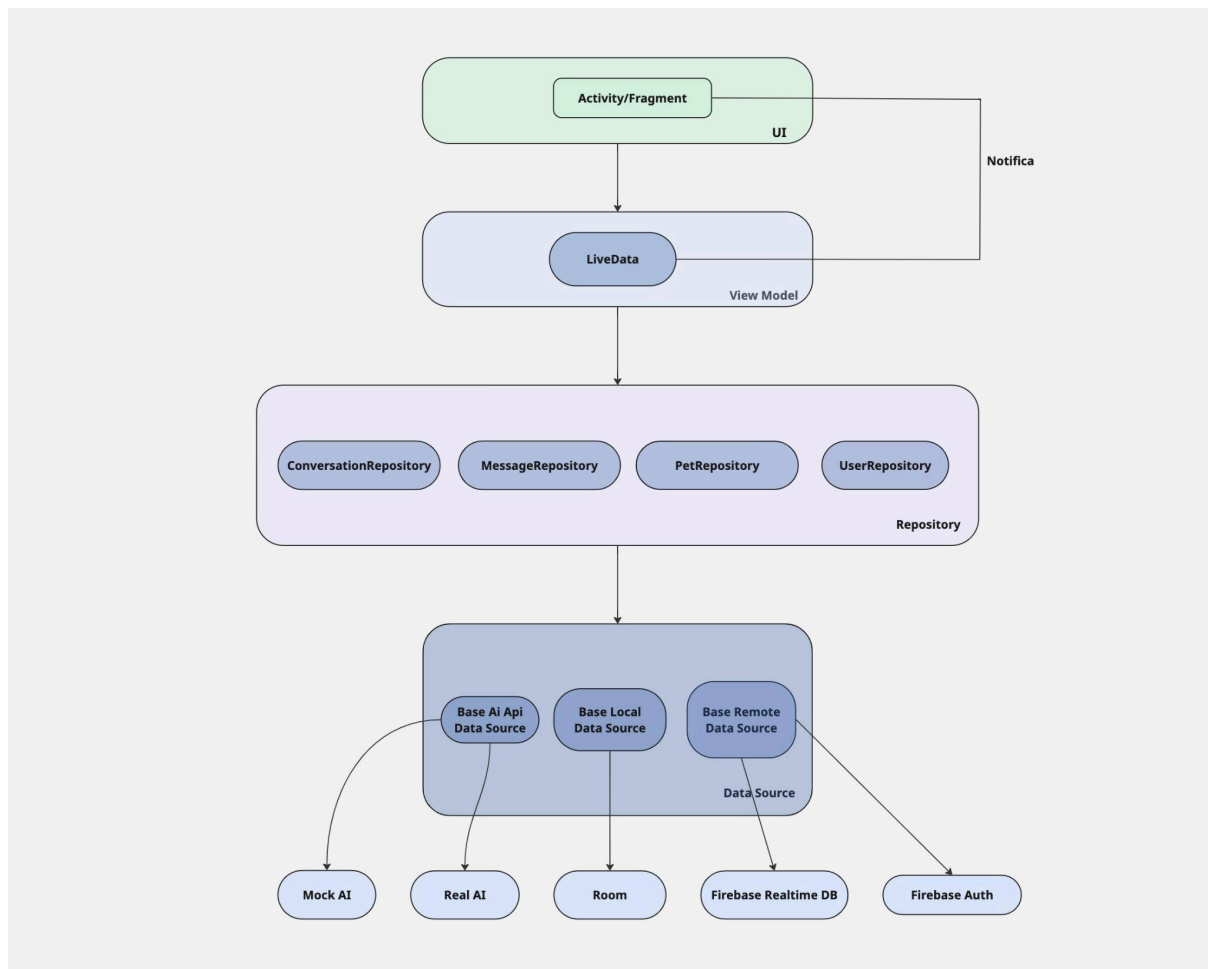
L'architettura principale dell'app è basata sul **Model-View-ViewModel** pattern (MVVM), ovvero un principio secondo cui si vuole separare il livello UI dal livello *data*, questo mediante ViewModels, che svolgono il ruolo di intermediari.

Partendo dalla UI, si chiama un metodo del ViewModel e si osserva il **Mutable Live Data** risultato quando si aggiornerà. Il Viewmodel, in modo simile, chiama una funzione della Repository e ne attende il risultato.

La Repository chiama delle funzioni sui DataSource e ne attende il callback con le funzioni implementate dall'interfaccia **ResponseCallback**.

I DataSource sono le uniche classi che interagiscono direttamente con il database, il quale potrà essere locale, remoto oppure una risposta AI proveniente da un'API. In base all'esito (success/failure) della chiamata, i DS informano la Repository tramite callback dedicata.

Infine, tutte le chiamate tornano indietro, aggiornando i LiveData, fino al livello UI. Qui, un observer che nota asincronicamente il cambiamento del LiveData, farà partire una funzione apposta per aggiornare l'interfaccia con i nuovi dati del database.



Il database **Room** è strutturato con un database chiamato **DataRoomDatabase**, ed è diviso in 3 tabelle relative ai Pet, Conversation, e Message. Gli oggetti sono in relazione tra di loro attraverso l'uso di **Foreign Keys** per permettere sia una connessione robusta tra loro, sia una netta separazione durante l'accesso ai dati.

## 4.1 Pattern architetturale

Il progetto adotta il pattern **MVVM (Model–View–ViewModel)**, in linea con le best practice Android moderne. Questa scelta permette di mantenere separati i vari livelli dell'applicazione, migliorandone leggibilità, manutenibilità e testabilità.

Le principali componenti sono:

- **UI (View)**: composta da Activity e Fragment, si occupa della gestione degli input dell'utente e dell'aggiornamento grafico. Le viste osservano i dati esposti dal ViewModel tramite LiveData.
- **ViewModel**: mantiene lo stato dell'interfaccia, gestisce la logica di presentazione e fornisce i dati alla View. Comunica con i Repository per ottenere o aggiornare le

informazioni.

- **Repository**: rappresenta il livello di astrazione che collega la logica di business alle fonti dati. Espone metodi chiari al ViewModel e decide da quale Data Source recuperare le informazioni.
- **Data Source**: componenti responsabili del recupero e della persistenza dei dati. Nel nostro caso includono:
  - **Firestore Authentication** → gestione login, registrazione e Google Sign-In.
  - **Firestore Realtime Database** → salvataggio e recupero delle informazioni sugli utenti e sui profili degli animali.
  - **Future Data Source** → predisposizione per possibili integrazioni (ad esempio API esterne per veterinari o servizi di tracciamento salute).

Si utilizza il pattern del **Single Source of Truth** tenendo il database locale come la unica fonte di verità, rispettando anche il pattern del **Offline First**.

I **Model** rappresentano le entità principali dell'app, come **User**, **Conversation**, **Message** e **Pet**, vengono visualizzati dai vari layer, e la memorizzazione è gestita interamente attraverso i ViewModel osservando dei **MutableLiveData**, in più, seguendo il pattern **Model-View-ViewModel** si applica anche un'elevata **Separation of concerns**.

## 4.2 Design pattern utilizzati

Durante lo sviluppo sono stati adottati diversi **design pattern** a supporto dell'architettura:

- **Observer**: implementato attraverso LiveData, per aggiornare automaticamente la UI al cambiare dei dati.
  - **Factory**: per l'inizializzazione di alcune componenti legate ai repository.
  - **Adapter**: per la gestione di liste dinamiche (es. cronologia chat) tramite RecyclerView.
- 

## 5. Funzionalità

### 5.1 Autenticazione

Schermata iniziale con le opzioni:

- Login
- Registrazione
- Login con Google
- Se l'utente ha già effettuato l'accesso, viene eseguito automaticamente il login con l'account precedentemente utilizzato.
- Reset password

### 5.2 Login

La schermata di login consente l'accesso al profilo tramite due campi compilabili:

- **E-mail**
- **Password**
- **Google sign-in**

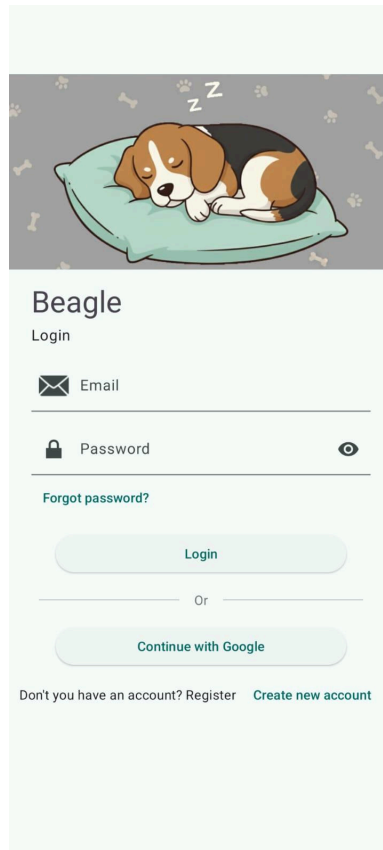
Se l'email ha un formato non conforme, appare un pallino rosso sulla destra del campo di testo che segnala tramite una piccola snack bar annessa che l'email non ha un formato corretto, idem succede con il campo password se questa ha un numero di caratteri minore di 6.

Se i dati inseriti non sono validi e si preme il bottone 'Login', l'app segnala l'errore con una **Snackbar** contenente un messaggio dedicato.



In caso di credenziali corrette o riconoscimento del profilo google (se si accede tramite bottone “accedi con google”, l’utente viene reindirizzato alla schermata principale (**Home**).

Dalla schermata di Login si può anche accedere alla schermata di reset password premendo il bottone “Hai dimenticato la password?” e alla schermata di registrazione premendo il bottone “Crea nuovo account”



The image shows a login form for an application named "Beagle". At the top, there is a header image of a beagle dog sleeping on a green pillow with the letter "Z" above it, set against a grey background with small paw prints and bones. Below the header, the word "Beagle" is displayed in a large, bold font, followed by the word "Login" in a smaller font. The form contains two input fields: "Email" with an envelope icon and "Password" with a lock icon and a toggle eye icon. Below the password field is a link "Forgot password?". A "Login" button is positioned below the "Forgot password?" link. Below the "Login" button is a horizontal line with the word "Or" in the center. Below this line is a "Continue with Google" button. At the bottom of the form, there is a link "Don't you have an account? Register" followed by a link "Create new account".

## 5.3 Registrazione

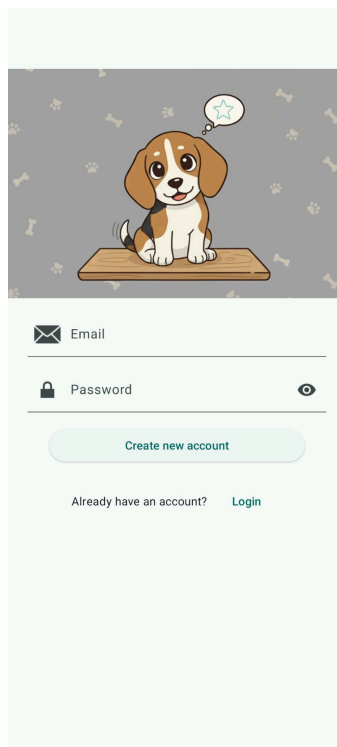
La schermata di registrazione viene visualizzata quando l'utente decide di creare un nuovo profilo nell'app *Beagle*.

In alto è presente il **logo dell'applicazione**, seguito dai campi da compilare per l'inserimento delle credenziali personali:

- **E-mail**
- **Password**

### Validazioni e controlli

- La **password** deve avere una lunghezza minima di **8 caratteri**.
- La password può essere mostrata o nascosta tramite apposita icona, così da garantire maggiore privacy.
- Se la **password è troppo corta**, l'app segnala che non rispetta i requisiti minimi di sicurezza.
- Se l'**e-mail è in un formato non valido**, l'utente viene avvisato con un messaggio di errore.
- Se è già stato creato un profilo con la stessa e-mail, la registrazione non viene completata, appare un messaggio di errore che comunica "questo utente è già stato registrato" e l'utente può tornare al **login** tramite apposito pulsante.

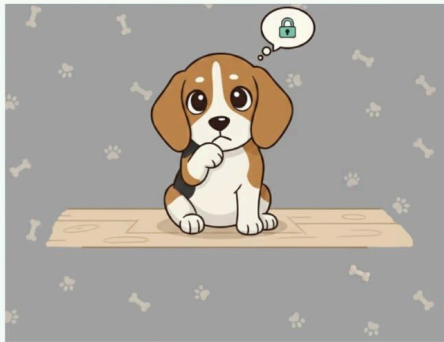


## 5.4 Reset Password


Dalla **schermata di login**, un utente già registrato può accedere alla funzionalità dedicata al recupero password tramite il link/bottone “**Password dimenticata?**”.

### Funzionalità principali

- L'utente inserisce l'**indirizzo e-mail** associato al proprio account.
- Premendo il pulsante “**RESET**”, l'applicazione avvia la procedura di recupero.
- Se l'e-mail è valida, **Firebase Authentication** invia un messaggio all'indirizzo fornito contenente le istruzioni per reimpostare la password manifestando nell'app un messaggio di invio email effettuato.
- Se l'indirizzo inserito **non è corretto** o non è associato a nessun account, viene mostrato un messaggio di errore.



**Forgot password?**

 Email

Reset

[BACK TO LOGIN](#)

## 5.5 Home (Chat con AI)

La schermata di home corrisponde alla chat con l'assistente AI.

L'utente può porre domande relative alla salute e al benessere del proprio animale.

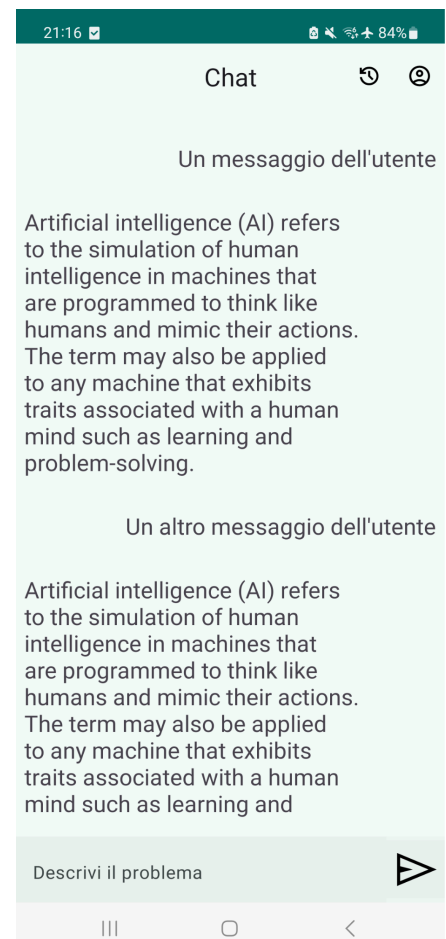
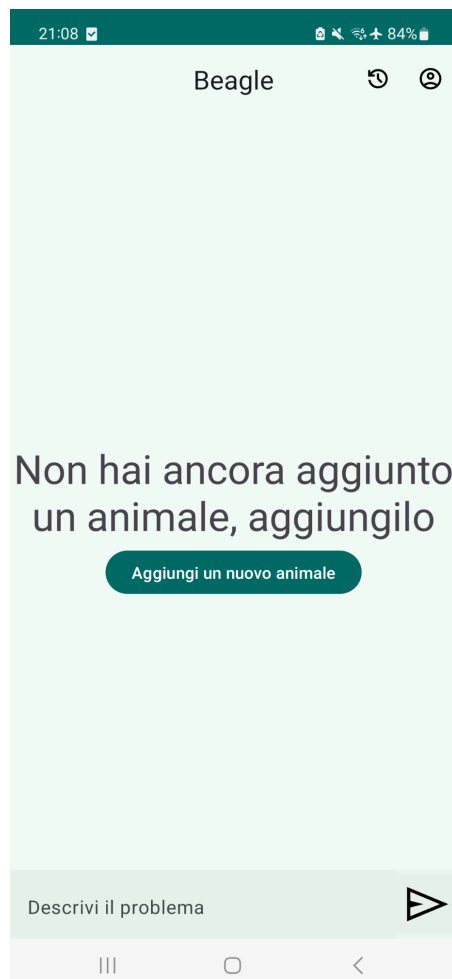
Ogni messaggio è associato ad una conversazione, ed ogni conversazione è legato al profilo del pet selezionato.

La schermata controlla se è presente o no un animale salvato, in caso contrario, disattiverà l'opzione di poter inviare domande all'AI e proporrà invece di aggiungere un animale nuovo, con un pulsante che porterà l'utente alla schermata adeguata.

Se invece è presente un animale salvato, si potrà invece chattare con l'AI; i messaggi sono salvati e mostrati in una recyclerView, è stato impostato il limite di views ad infinito perchè con le impostazioni di default, i messaggi sparivano se ce n'erano troppo.

Il fragment che ospita questa view è chiamato ChatFragment, e anche se all'apparenza sembra pieno, e quindi non rispettare la separation of concerns, in verità la maggior parte del codice è solo la gestione observers su diversi mutableLiveData per diversi eventi, e non staccabili dalla classe.

Infine, è presente una app bar, con il pulsante cronologia, che porta alla schermata di tutte le conversazioni passate riferite all'animale in questione, e il pulsante profilo, che presenta i dati dell'animale, e la possibilità di sceglierne un altro.

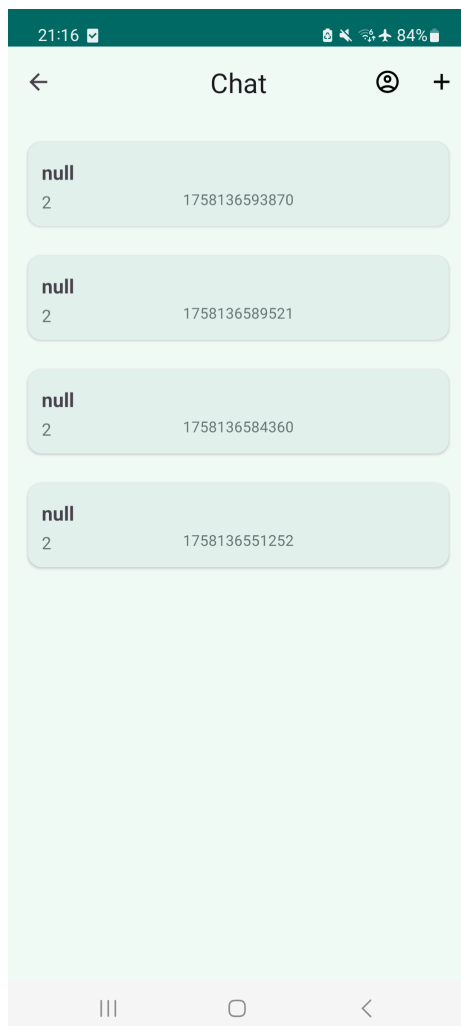


## 5.5 Cronologia chat

In questa sezione si trova una lista, ovvero la cronologia, delle conversazioni create per il Pet attualmente selezionato. Le conversazioni sono mostrate in una `recyclerView` e ogni *Conversation* è accessibile tramite una view che ne mostra titolo, nome del Pet associato e data di ultima modifica. Questi ultimi dati non sono ancora settati nella loro forma definitiva. Premendo su una delle conversazioni, questa viene aperta nel `ChatFragment`.

Il fragment che ospita questa view è chiamato `ConversationsHistoryFragment`.

Infine, nella schermata è visibile la stessa app bar condivisa in tutta l'activity. Contiene pulsante Profilo, che porta alla schermata del Pet attualmente attivo e dalla quale può esserne creato o selezionato un altro. Il pulsante più a destra consente di creare una nuova conversazione, portando l'utente sul `ChatFragment`. A sinistra il pulsante indietro che riporta su `ChatFragment`.



## 5.6 Profili animali

Ogni animale dell'utente ha un profilo dedicato che raccoglie:

- Dati anagrafici (nome, specie, razza, età)
- Collegamento diretto alla cronologia delle chat relative al pet

La schermata è raggiungibile dal menù principale e dal menù cronologia.

La razza si può scegliere da un menù a tendina che permette di selezionare cane o gatto.

L'età viene calcolata mentre viene scelta la data di nascita dal calendario popup.

/

The screenshot shows a mobile app interface for adding a new animal profile. At the top, there's a header with a back arrow and the title 'profile\_title'. Below the header is a dropdown menu labeled 'I tuoi anim...'. The main form consists of several fields: a text input for 'Nome' with the value 'Fido', a 'Specie' dropdown menu set to 'Cane', a 'Razza' text input with the value 'Pointer inglese', a 'Data di nascita' field with a calendar icon and the date '09/06/2018', and an 'Età' field with the value '7 Years 3 Months'. At the bottom, there are two buttons: 'Annulla' and 'Salva'.

The screenshot shows the same 'profile\_title' screen but for editing an existing profile. The 'I tuoi animali' dropdown is expanded, showing a list with 'Luna' selected. Below the list, the name 'Fido' is visible. The 'Specie' dropdown is set to 'Cane' and the 'Razza' field is empty. The 'Data di nascita' field has a calendar icon and the text 'Data di nas...', and the 'Età' field is empty. At the bottom, there is a red button labeled 'Elimina'.

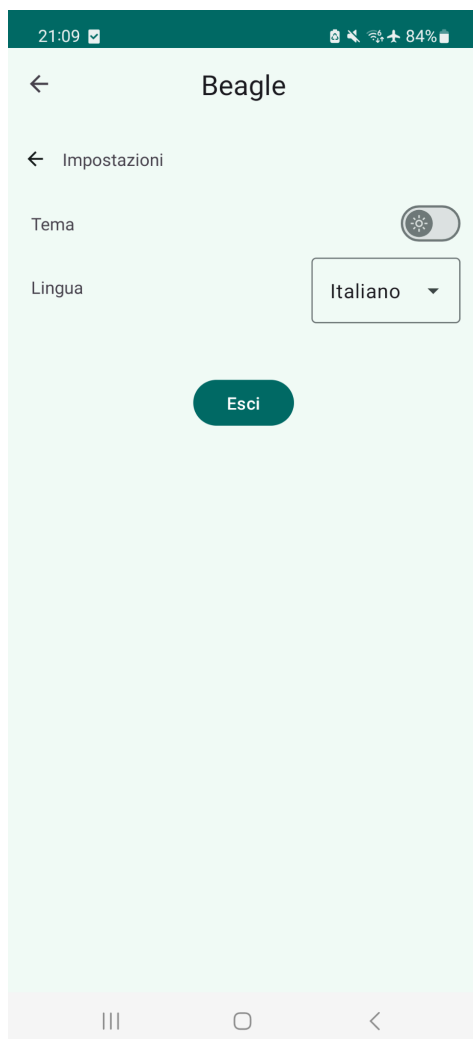
## 5.7 Impostazioni

La schermata delle **Impostazioni** consente all'utente di personalizzare alcuni aspetti dell'app e di gestire il proprio account.

Gli elementi principali presenti sono:

- **Tema:** uno switch che permette di attivare o disattivare il tema scuro/chiaro.
- **Lingua:** un menù a tendina che consente di selezionare la lingua preferita per l'applicazione.
- **Logout:** un pulsante dedicato per effettuare il logout e terminare la sessione utente.

La schermata è accessibile tramite il menù del profilo animale e offre funzionalità di personalizzazione e gestione dell'account in modo semplice e immediato.



---

## 6. Sviluppi futuri

- **Tracking salute pet:** possibilità di monitorare parametri sanitari (peso, vaccinazioni, visite).
- **Documentazione sanitaria:** caricamento e archiviazione digitale di certificati, referti e ricette.
- **Consigli veterinari in zona:** sistema di geolocalizzazione per trovare veterinari vicini, con recensioni e valutazioni degli utenti.
- **Foto profilo pet:** aggiunta della possibilità di caricare un'immagine per ogni animale registrato.

Tuttavia, la comunicazione tra umano e animale è estremamente limitata, per questo motivo uno degli obiettivi futuri è quello di offrire un prodotto che possa aiutare a discernere comportamenti innocui da quelle situazioni è effettivamente necessaria una visita specialistica.

Per raggiungere questi ed altri obiettivi, Beagle si arricchirà di molte funzionalità:

- Raccolta e conservazione della documentazione sanitaria.
- Un nuovo modello più performante, finetunato per questo specifico compito.
- Un sistema RAG per arricchire il contesto del modello, contenente letteratura veterinaria e protocolli di anamnesi.
- Supporto per nuove specie.
- Statistiche anonimizzate e aggregate sulla popolazione di pet dell'app, per riconoscere comportamenti e caratteristiche specifici di singole specie e razze.
- Collaborazioni con enti, università e aziende per coinvolgere la community in studi e ricerche sperimentali

Infine, il vero potenziale di Beagle sta nel poter riconvertire facilmente la piattaforma per creare un prodotto in tutto e per tutto analogo a disposizione del benessere e della salute umana.