Instituto Puig Castellar
Ciclo formativo: DAM - Desarrollo de aplicaciones multiplataforma
Grado: CFGS Proyecto intermodular

TheraPets

Aplicación móvil para gestión de terapia asistida con animales

Autora: Nicole Melanie Vega Bautista
Tutor: Luis Elía
Curso: 2025-2026
Fecha de entrega: 17/05/2026

CC BY

Resumen del proyecto

El origen de TheraPets parte de una experiencia personal que permitió comprobar de primera mano el impacto positivo que los animales pueden tener en el bienestar emocional de las personas.

A pesar de que la terapia asistida con animales es una práctica reconocida por sus beneficios sobre la salud mental, su acceso continúa siendo limitado y poco digitalizado. Las plataformas existentes, como Miwuki Pet Shelter o Protectora BCN, se centran principalmente en la adopción o el cuidado de mascotas, pero no ofrecen soluciones orientadas específicamente a la gestión de sesiones terapéuticas emocionales. Esta carencia dificulta el acceso organizado y sencillo a este tipo de terapias desde dispositivos móviles.

TheraPets es una aplicación móvil nativa para Android orientada a la gestión de sesiones de terapia asistida con animales. La aplicación está dirigida a personas que atraviesan problemas emocionales como ansiedad, depresión, baja autoestima o dificultades para socializar.

A través de la aplicación, el usuario puede registrarse, explorar centros terapéuticos, seleccionar el terapeuta y el animal con el que desea realizar la sesión, indicar el motivo de la cita y gestionar sus reservas, pudiendo consultarlas o cancelarlas indicando la razón correspondiente. Una vez finalizada la sesión, el usuario puede dejar un testimonio con comentario y fotografía, visible para el resto de usuarios, además de consultar el historial de sesiones realizadas.

La aplicación ha sido desarrollada de forma individual utilizando Android Studio con Java, Firebase Firestore como base de datos en la nube y Firebase Authentication para la gestión de usuarios y accesos. El control de versiones se ha realizado mediante GitHub.

El resultado es una herramienta funcional que demuestra la viabilidad de digitalizar el acceso a la terapia asistida con animales, acercando este tipo de apoyo emocional a un público más amplio de forma intuitiva, organizada y accesible.

Palabras clave:

Terapia asistida con animales, aplicación móvil Android, salud mental, bienestar emocional, Firebase, gestión de citas terapéuticas.

Abstract

Animal-assisted therapy is a recognised practice due to its positive impact on mental health and emotional well-being. However, access to these services remains limited and insufficiently digitalised. Existing platforms mainly focus on pet care or adoption, but they do not provide solutions specifically designed for the management of animal-assisted therapy sessions.

TheraPets is a native Android mobile application that allows users to register, explore therapy centres, select a therapist and an animal for the session, indicate the reason for the appointment, and manage their scheduled appointments. Once a session is completed, users can leave a review including a comment and an image, which is visible to other users, as well as consult their session history.

The application has been developed individually using Android Studio, Java, Firebase Firestore, and Firebase Authentication. The result is a functional system that demonstrates the feasibility of digitalising access to animal-assisted therapy, making this type of resource more accessible in an organised and intuitive manner.

Keywords:

Animal-assisted therapy, mobile application, Android, emotional well-being, mental health, appointment management, Firebase

Índice

Figuras

Tablas

1. Presentación del proyecto

1.1 Introducción

TheraPets es una aplicación móvil desarrollada para Android cuyo objetivo es facilitar el acceso a la terapia asistida con animales como herramienta de apoyo al bienestar emocional.

Aunque existen numerosas aplicaciones relacionadas con animales, la mayoría de ellas están orientadas a la adopción, el cuidado o el entretenimiento. En este contexto, las soluciones digitales centradas específicamente en la gestión de sesiones de terapia asistida con animales son limitadas, lo que pone en manifiesto una oportunidad de mejorar en este ámbito.

TheraPets propone una solución digital orientada a la organización y gestión de este tipo de sesiones terapéuticas, permitiendo a los usuarios acceder de forma sencilla a un servicio estructurado y accesible que contribuye potencialmente a la mejora del bienestar emocional.

1.2 Contexto

La terapia asistida con animales es una intervención complementaria utilizada en distintos entornos clínicos y socioeducativos, especialmente en el tratamiento de personas mayores, niños hospitalizados y pacientes con trastornos emocionales. Diversos estudios han demostrado su impacto positivo en la reducción del estrés, la ansiedad y el dolor, así como en la mejora del bienestar emocional y social de los pacientes.

Un ejemplo de ello es la investigación realizada por el Hospital Universitario 12 de Octubre junto con la Cátedra Animales y Sociedad de la Universidad Rey Juan Carlos [1], en la que se evaluaron intervenciones con perros de terapia en niños ingresados en unidades de cuidados intensivos. Los resultados mostraron una reducción significativa del dolor, miedo y la ansiedad [2], así como mejoras variables como la autoestima, las relaciones interpersonales, la sensación de soledad y el rendimiento cognitivo tras un programa de 10 sesiones durante 6 semanas [3].

A pesar de su eficacia demostrada en entornos clínicos, la terapia asistida con animales sigue siendo una práctica poco extendida a nivel general y con escasa presencia en el ámbito digital. Actualmente, no existen soluciones móviles específicas que permitan gestionar este tipo de servicios de forma integral. Las aplicaciones existentes en el sector animal se centran principalmente en la adopción, el cuidado o el entrenamiento de mascotas, dejando sin cubrir el ámbito de la intervención terapéutica asistida, lo que evidencia una oportunidad clara para el desarrollo de TheraPets.

1.3 Justificación

La justificación de este proyecto se basa en la necesidad de facilitar el acceso a la terapia asistida con animales mediante soluciones digitales que permitan su gestión de forma sencilla y centralizada.

Aunque este tipo de terapia ha demostrado beneficios en la mejora del bienestar emocional y la reducción de síntomas asociados a la ansiedad y depresión, actualmente no existen aplicaciones móviles específicas que permitan gestionar este tipo de sesiones de forma integral.

TheraPets se dirige a usuarios que presentan necesidades relacionadas con el bienestar emocional, como ansiedad, baja autoestima o dificultades en la socialización, ofreciendo una plataforma que permite la gestión de sesiones terapéuticas asistidas con animales de forma estructurada. Esto incluye la reserva de sesiones, la selección del terapeuta y el animal, así como el seguimiento de las mismas.

Desde el punto de vista tecnológico, el desarrollo de esta aplicación responde a la oportunidad de digitalizar un servicio que actualmente no cuenta con soluciones móviles especializadas, integrando herramientas modernas como Android y Firebase para su implementación.

1.4 Objetivos

Objetivo general

Desarrollar una aplicación móvil para Android que facilite el acceso a la terapia asistida con animales, conectando a personas con problemas emocionales con centros especializados de forma sencilla, organizada e intuitiva, fomentando así su bienestar emocional.

Objetivos específicos

- ➢ Permitir a los usuarios registrarse, iniciar sesión y gestionar su perfil de forma segura.
- ➢ Facilitar la consulta y selección de centros de terapia, especialistas y animales disponibles.
- ➢ Permitir la reserva y cancelación de sesiones terapéuticas desde la propia aplicación.
- ➢ Permitir registrar el motivo de la cita por parte del usuario.
- ➢ Ofrecer un sistema de testimonios y valoraciones tras cada sesión, incluyendo comentarios y fotografías.

- ➢ Permitir a los coordinadores de los centros gestionar la disponibilidad de animales y horarios.
- ➢ Desarrollar una interfaz de usuario accesible, intuitiva y adaptada a dispositivos móviles Android.

2. Estrategia y planificación

2.1 Estrategia de desarrollo y viabilidad

El punto de partida del proyecto fue una fase de investigación inicial en la que se analizó la existencia de una necesidad real para una aplicación como TheraPets. A partir del estudio de la terapia asistida con animales y la revisión de soluciones digitales existentes, se identificó la ausencia de aplicaciones específicas orientadas a la gestión de este tipo de terapias, lo que justificó el desarrollo del proyecto.

En cuanto a los recursos disponibles, el proyecto se desarrolló de forma individual utilizando los conocimientos adquiridos durante la formación académica, especialmente en desarrollo Android. Se utilizó código base previamente trabajado como apoyo inicial, complementándolo con investigación adicional y documentación oficial para ampliar los aspectos técnicos necesarios.

Respecto a la viabilidad del proyecto, desde el inicio se identificaron ciertas limitaciones, principalmente relacionadas con el tiempo de desarrollo, lo que obligó a priorizar las funcionalidades esenciales y descartar características adicionales. Asimismo, se tuvo en cuenta el riesgo asociado al desarrollo de nuevas funcionalidades y la necesidad de mantener una arquitectura coherente durante todo el proceso. No obstante, el proyecto se consideró viable debido al uso de tecnologías consolidadas como Android Studio, Java y Firebase, que cuentan con amplia documentación de apoyo y soporte.

2.2 Metodología de trabajo

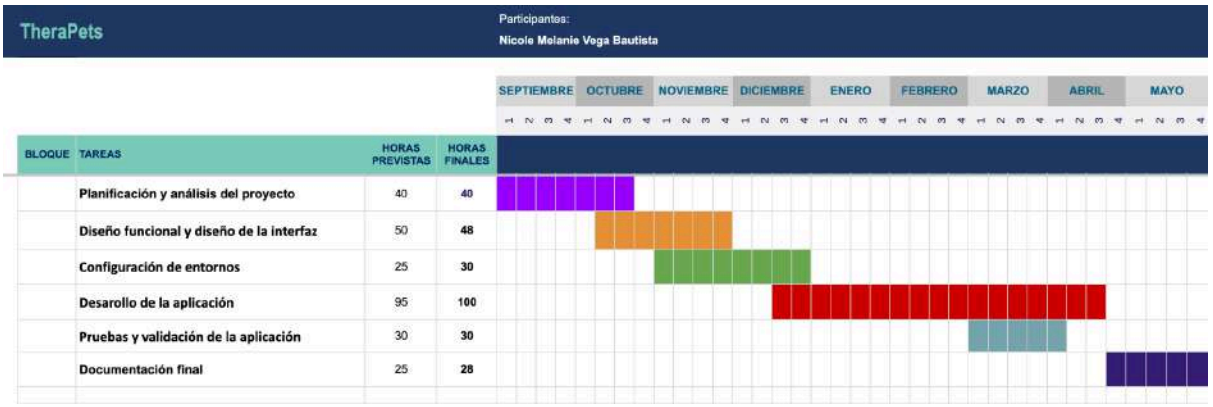
El desarrollo del proyecto se organizó por fases, siguiendo un enfoque estructurado que permitió avanzar de forma ordenada durante todo el proceso. Para gestionar las tareas se utilizó Trello [4], donde se crearon tableros diferenciados para las distintas áreas del proyecto, como documentación, diseño y programación/desarrollo. Esta organización permitió un seguimiento del estado de las tareas, clasificándolas como pendientes, en progreso y finalizadas.

Antes de comenzar la implementación, se llevó a cabo una fase de diseño de la interfaz del usuario, administrador y coordinador de centro mediante bocetos manuales. Estos bocetos sirvieron como base inicial del diseño de la aplicación y fueron refinándose progresivamente durante el desarrollo en función de las necesidades del sistema.

En cuanto a la planificación de trabajo, se inició con la fase de documentación, en la que se definieron los requisitos del sistema, los roles de usuario y las principales funcionalidades de la aplicación. Posteriormente, se pasó a la fase de desarrollo, utilizando como base el código previamente realizado en prácticas del ciclo formativo, el cual fue ampliado y adaptado a los requisitos del proyecto.

2.3 Planificación

La planificación del proyecto se estructuró en seis fases principales, cada una con una estimación de horas asignadas y distribuidas a lo largo del curso académico, desde septiembre hasta mayo. Las fechas de entrega de cada fase fueron definidas de forma autónoma y supervisadas periódicamente por el tutor del proyecto para garantizar el cumplimiento de los objetivos establecidos.



BLOQUE	TAREAS	HORAS PREVISTAS	HORAS FINALES
	Planificación y análisis del proyecto	40	40
	Diseño funcional y diseño de la interfaz	50	48
	Configuración de entornos	25	30
	Desarollo de la aplicación	95	100
	Pruebas y validación de la aplicación	30	30
	Documentación final	25	28

Figura 1. Diagrama de Gantt. Fuente: Elaboración propia.

Las fases y su dedicación:

- **Fase 1 - Planificación y análisis del proyecto (40 horas | Septiembre - Octubre)** En esta fase se llevó a cabo el análisis inicial del sistema, identificando los requisitos funcionales y no funcionales de la aplicación, así como los roles de usuario. También se inició la organización de tareas mediante la herramienta Trello

- **Fase 2 - Diseño funcional y diseño de la interfaz (48 horas | Octubre - Noviembre)** Se realizaron bocetos iniciales de la aplicación con el objetivo de definir la estructura de pantallas y funcionalidades. Posteriormente, estos diseños se refinaron mediante la herramienta Figma, obteniendo una propuesta más detallada y estructurada de la interfaz de usuario.

- **Fase 3 - Configuración de entornos (30 horas | Noviembre - Diciembre)** Se configuró Android Studio, se integraron las dependencias necesarias para la conexión con Firebase y se creó el repositorio del proyecto en GitHub [5] para el control de versiones.

- **Fase 4 - Desarrollo de la aplicación (100 horas | Diciembre - Abril)** Se implementó la lógica principal de la aplicación, incluyendo la creación, visualización y gestión de datos. Inicialmente se desarrollaron las pantallas principales y la interfaz

de usuario, y posteriormente se integró Firebase para la gestión de datos en tiempo real.

- **Fase 5 - Pruebas y validación (30 horas | Marzo - Abril)** Se realizaron pruebas funcionales tanto en dispositivo físico como en emulador de Android Studio, verificando el correcto funcionamiento de las principales funcionalidades de la aplicación.

- **Fase 6 - Documentación final (28 horas | Abril - Mayo)** Se elabora la memoria del proyecto, recopilando el proceso de desarrollo, las decisiones de diseño y los resultados obtenidos.

3. Análisis

3.1 Casos de uso

Los casos de uso describen qué hará cada tipo de usuario dentro de TheraPets, qué rol tiene y qué funciones puede realizar. Cada actor tiene acciones distintas dentro de la aplicación, por lo que identificarlos y separarlos permite entender mejor cómo funciona el sistema en su conjunto. Se identifican cuatro actores: **visitante**, **usuario**, **coordinador** y **administrador**.

Tabla 1. Casos de uso

Código	Nombre del caso de uso	Actor principal	Descripción	Resultado esperado
CU1	Registro de usuario	Visitante	El visitante introduce sus datos para crear una nueva cuenta en el sistema.	Usuario registrado correctamente.
CU2	Ver información	Visitante	El visitante consulta información general de la aplicación sin necesidad de registro.	Información mostrada correctamente.
CU3	Iniciar sesión	Usuario/ Coordinador/ Administrador	El usuario accede al sistema mediante credenciales.	Acceso concedido según rol.
CU4	Recuperar contraseña	Usuario	El usuario solicita el restablecimiento de contraseña mediante correo electrónico.	Correo de recuperación enviado correctamente.
CU5	Solicitar cita	Usuario	El usuario selecciona fecha, centro, animal con terapeuta y motivo de la cita.	Solicitud registrada en estado “pendiente”.
CU6	Ver perfil e historial	Usuario	El usuario consulta su información personal y el historial de solicitudes/citas.	Datos mostrados correctamente.
CU7	Dejar testimonio	Usuario	El usuario publica una valoración con comentario e imagen tras la sesión finalizada.	Testimonio publicado y visible.

CU8	Ver testimonios	Usuario	El usuario consulta los testimonios publicadas por otros usuarios.	Testimonios mostrados correctamente.
CU9	Cerrar sesión	Usuario / Coordinador / Administrador	El usuario finaliza la sesión de forma segura.	Sesión cerrada correctamente.
CU10	Gestionar solicitudes de cita	Coordinador	El coordinador revisa las solicitudes de cita de su centro y puede aceptarlas/rechazarlas o cancelarlas.	Solicitud actualizada a aceptada, rechazada o cancelada.
CU11	Gestionar centro	Administrador	El administrador crea y modifica centros terapéuticos.	Centro gestionado correctamente.
CU12	Gestionar coordinadores	Administrador	El administrador crea coordinadores y los asigna a centros.	Coordinador creado y asignado.
CU13	Ver todas las citas	Administrador	El administrador consulta todas las solicitudes y citas de todos los centros.	Información mostrada correctamente.
CU14	Gestionar animales	Coordinador	El coordinador añade, edita o elimina animales con terapeutas del centro.	Cambios guardados correctamente.
CU15	Gestionar horarios	Coordinador	El coordinador define la disponibilidad del centro	Horarios actualizados correctamente.
CU16	Cancelar cita	Usuario	El usuario cancela una solicitud o cita activa indicando un motivo.	Actualizada a estado “cancelada”.

Tabla 2. CU5 - Solicitar cita

Campo	Descripción
Identificador	**CU5**
Nombre	Solicitar cita
Actor	Usuario
Precondiciones	Usuario autenticado en la aplicación
Flujo principal	1. El usuario accede a la pantalla de crear solicitud. 2. Selecciona la fecha y hora disponible. 3. Selecciona el centro de terapia. 4. Selecciona el animal y terapeuta con el que desea realizar la sesión. 5. Indica el motivo de la cita entre las opciones disponibles. 6. Si lo necesita, especifica el motivo en un campo de texto libre. 7. El sistema registra la solicitud de cita. 8. Se muestra la confirmación de solicitud enviada.
Flujo alternativo	Si el usuario no ha iniciado sesión, es redirigido al login.
Postcondiciones	La solicitud queda registrada en estado pendiente de confirmación por el coordinador.

Tabla 3. CU16 - Cancelar cita

Campo	Descripción
Identificador	**CU16**
Nombre	Cancelar cita
Actor	Usuario
Precondiciones	El usuario ha iniciado sesión y dispone de solicitudes de cita activas.
Flujo principal	1. El usuario accede a la sección de próximas solicitudes de citas. 2. Selecciona la cita/solicitud que desea cancelar. 4. El usuario confirma la cancelación. 5. El usuario introduce el motivo de la cancelación. 6. El sistema actualiza el estado de la solicitud a “cancelada”.
Flujo alternativo	Si el usuario no tiene ninguna cita/solicitud reservada, el sistema muestra que en el panel de MisCitas está vacía.
Postcondiciones	La solicitud queda cancelada correctamente y el cambio se refleja en el historial del usuario.

Tabla 4. CU07 - Dejar un testimonio

Campo	Descripción
Identificador	**CU07**
Nombre	Dejar un testimonio
Actor	Usuario registrado
Precondiciones	El usuario ha iniciado sesión en la aplicación y ha finalizado al menos una sesión.
Flujo principal	1. El usuario accede a la sección de testimonios. 2. Redacta su experiencia en el campo de texto. 3. Opcionalmente, selecciona una foto de su galería para adjuntarla. 4. El usuario sube el testimonio. 5. El sistema publica el testimonio y queda visible para todos los usuarios.
Flujo alternativo	El usuario puede publicar el testimonio únicamente con texto, sin adjuntar imagen.
Postcondiciones	El testimonio queda publicado y visible para el resto de usuarios.

Tabla 5. CU10 - Gestionar solicitudes de cita

Campo	Descripción
Identificador	**CU10**
Nombre	Gestionar solicitudes de cita
Actor	Coordinador de centro
Precondiciones	El coordinador ha iniciado sesión en la aplicación y tiene acceso a las solicitudes de su centro.
Flujo principal	1. El coordinador accede a su panel de gestión. 2. Consulta las solicitudes de citas recibidas en su centro. 3. Selecciona una solicitud. 4. Revisa detalles de la solicitud. 5. Decide aceptar o rechazar solicitud. 6. El sistema actualiza el estado de la solicitud correctamente.
Flujo alternativo	Si no existen solicitudes para el centro, el sistema muestra el panel de solicitudes vacío.
Postcondiciones	La solicitud queda actualizada como **aceptada, cancelada o rechazada** en el sistema.

3.2 Requisitos funcionales

Los requisitos funcionales describen las interacciones que tendrá la aplicación, es decir, qué funciones debe cumplir y qué tiene que ocurrir dentro de ella. Se pueden considerar la base que garantiza que TheraPets funcione correctamente y tenga coherencia, ya que definen qué debe pasar en cada momento dentro del sistema.

Tabla 6. Requisitos funcionales

ID	Requisito	Prioridad
RF1	El sistema permitirá registrar nuevos usuarios.	Alta
RF2	Los usuarios podrán iniciar sesión/cerrar mediante correo electrónico y contraseña.	Alta
RF3	El sistema permitirá el restablecimiento de contraseña mediante correo electrónico.	Media
RF4	El sistema permitirá visualizar una lista básica de animales disponibles en la pantalla principal, mostrando únicamente su nombre.	Alta
RF5	El sistema permitirá consultar la disponibilidad de fechas y horarios durante el proceso de solicitud de cita.	Alta
RF6	El sistema permitirá seleccionar el motivo de la solicitud de cita.	Alta
RF7	El sistema permitirá crear solicitudes de cita en estado pendiente de validación por el coordinador.	Alta
RF8	El sistema permitirá mostrar el historial de solicitudes y citas del usuario.	Media
RF9	El sistema permitirá registrar valoraciones tras la finalización de una sesión.	Alta
RF10	El coordinador podrá gestionar la disponibilidad de animales, terapeutas y horarios de su centro.	Alta
RF11	El administrador podrá gestionar los centros terapéuticos de la aplicación.	Alta
RF12	El administrador podrá gestionar coordinadores y su asignación a centros.	Alta

Los requisitos de prioridad alta constituyen el núcleo funcional de la aplicación y deben estar implementados en la versión final del proyecto. Los de prioridad media se desarrollarán en función del tiempo disponible.

3.3 Requisitos no funcionales

Además de las funcionalidades descritas, la aplicación debe cumplir una serie de condiciones de calidad que afectan a su comportamiento general.

- **Usabilidad:** La interfaz debe ser intuitiva, de forma que el usuario pueda realizar las funciones principales como el registro, la solicitud de cita y el cierre de sesión de forma sencilla. Además, se tendrán en cuenta criterios básicos de accesibilidad visual, como contraste de colores, legibilidad y tamaño de fuente.
- **Rendimiento:** Las pantallas de la aplicación deberán cargarse en un tiempo reducido en condiciones normales de uso, garantizando una experiencia fluida para el usuario.
- **Disponibilidad:** La aplicación deberá mantener una alta disponibilidad del servicio, asegurando el acceso continuo a los usuarios, apoyándose en la infraestructura en la nube de Firebase.
- **Compatibilidad:** La aplicación será compatible con dispositivos Android, adaptándose a diferentes tamaños de pantalla y resoluciones.

3.4 Análisis de alternativas tecnológicas

Las tecnologías utilizadas en el desarrollo de TheraPets han sido seleccionadas tras un análisis de distintas alternativas, considerando criterios como la facilidad de implementación, el tiempo de desarrollo, la escalabilidad y la integración con plataformas móviles.

Entorno de desarrollo

Se evaluó la posibilidad de desarrollar la aplicación como solución web o aplicación móvil. Finalmente, se optó por una aplicación móvil Android debido a que este tipo de plataformas ofrece un acceso más rápido, directo y adecuado para el uso en movilidad, especialmente en aplicaciones orientadas a servicios personales.

En cuanto al entorno de desarrollo, Android Studio [6] con Java [7] y Kotlin. Se eligió Java debido a la experiencia previa adquirida durante la formación académica, lo que permitió agilizar el desarrollo y centrarse en la implementación de funcionalidades. Kotlin se consideró como alternativa, aunque se descartó por el menor dominio de la tecnología.

Base de datos

Se analizaron dos alternativas principales: MySQL y Firebase Firestore. MySQL es una base de datos relacional ampliamente utilizada, pero requiere la configuración y mantenimiento de un servidor propio.

Firebase Firestore [8], en cambio, ofrece una solución en la nube que reduce la complejidad de infraestructura y permite una integración directa con aplicaciones Android. Además, Firebase Authentication [9] facilita la gestión de usuarios y autenticación de forma segura y centralizada. Por estos motivos, se seleccionó Firebase Firestore como base de datos principal del proyecto.

Herramientas de diseño

Para el diseño de la interfaz se utilizó Figma [10] una herramienta de prototipado que permite diseñar interfaces de forma visual e iterativa. Su uso facilita la planificación previa de la estructura de la aplicación antes de su implementación y cuenta con extensa documentación complementaria.

Conclusión del análisis

Las tecnologías seleccionadas responden a criterios de eficiencia, simplicidad de desarrollo e integración con el entorno Android, adaptándose a las limitaciones de tiempo y al alcance del proyecto.

4. Diseño

4.1 Arquitectura del sistema

La aplicación ha sido desarrollada siguiendo una arquitectura basada en el patrón MVC (Modelo-Vista-Controlador), habitual en el desarrollo de aplicaciones Android con Java. Esta arquitectura permite separar la interfaz de usuario, la lógica de la aplicación y la gestión de datos, facilitando el mantenimiento y la escalabilidad del sistema.

Vista

La vista corresponde a la interfaz de usuario de la aplicación, implementada mediante layouts XML y Activities. Es la parte del sistema con la que el usuario interactúa directamente, permitiendo el acceso a las diferentes funcionalidades de la aplicación.

Controlador

El controlador está formado por las clases desarrolladas en Java, encargadas de gestionar la lógica de la aplicación. Estas clases controlan el flujo de datos entre la interfaz y el sistema, gestionando funcionalidades como el registro de usuarios, la solicitud de citas, la autenticación y la actualización de información.

Modelo

El modelo está representado por Firebase, que actúa como capa de persistencia y backend del sistema. Firebase Firestore se utiliza para el almacenamiento y sincronización de datos en tiempo real, mientras que Firebase Authentication gestiona el acceso de usuarios según su rol (usuario, coordinador o administrador).

Conclusión

Esta arquitectura permite una clara separación de responsabilidades, facilitando la organización del código, la detección de errores y la futura escalabilidad del sistema.

4.2 Modelo de datos

La base de datos de TheraPets está compuesta por varias entidades principales que permiten gestionar la información necesaria para el funcionamiento de la aplicación. La implementación se realiza mediante Firebase Firestore, lo que facilita el almacenamiento y sincronización de datos en la nube.

Usuario

La entidad Usuario almacena la información de las personas registradas en el sistema, incluyendo un identificador único, nombre, apellidos, correo electrónico, teléfono, fecha de nacimiento y rol (usuario, coordinador o administrador). En el caso de los coordinadores, se incluye además el identificador del centro asignado.

Centro

La entidad Centro almacena la información de los centros de terapia disponibles, incluyendo identificador, nombre, dirección, teléfono e imagen.

Animal

La entidad Animal representa los animales disponibles para las sesiones terapéuticas, incluyendo identificador, nombre, tipo, raza, edad, especialidad y el centro al que pertenece.

Terapeuta

La entidad Terapeuta almacena la información de los profesionales asociados a cada centro, incluyendo identificador, nombre, especialidad e imagen.

Solicitud de cita

La entidad Solicitud de cita constituye el núcleo del sistema. Almacena las solicitudes realizadas por los usuarios, incluyendo identificador, usuario_id, centro, animal con terapeuta, fecha, hora, motivo y estado.

Testimonio

La entidad Testimonio almacena las valoraciones realizadas por los usuarios tras las sesiones, incluyendo identificador, usuario_id, comentario, imagen opcional y fecha de publicación.

Horario

La entidad Horario representa la disponibilidad de los centros, incluyendo identificador, día, hora y centro_id asociado.

Relaciones entre entidades

Un usuario puede realizar múltiples solicitudes de cita y publicar varios testimonios. Cada solicitud está asociada a un único usuario, un centro, un animal y un terapeuta. Cada centro puede tener múltiples animales y horarios asociados.

Conclusión

Este modelo de datos permite gestionar de forma estructurada las funcionalidades principales de la aplicación, garantizando coherencia entre usuarios, centros y animales.

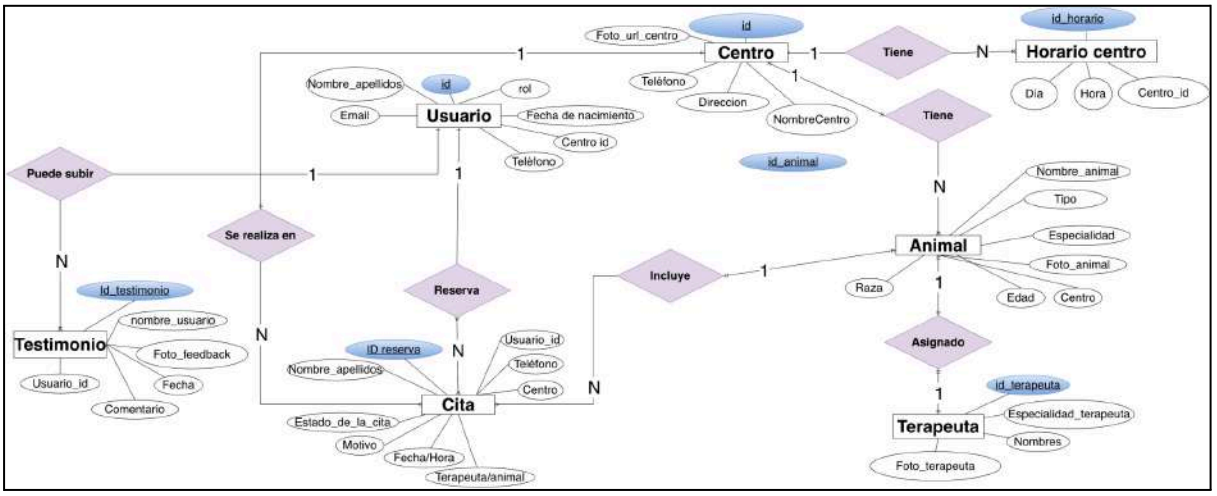


Figura 2. Diagrama entidad-relación TheraPets. Fuente: Elaboración propia mediante draw.io [11].

4.3 Diseño de interfaz

El diseño de la interfaz se ha planteado con el objetivo de ofrecer una experiencia de uso clara, intuitiva y accesible. Antes del desarrollo, se realizaron bocetos en Figma para definir la estructura visual y el flujo de navegación de la aplicación. A lo largo del desarrollo, estos diseños fueron ajustándose e implementando progresivamente en la aplicación final.

Se ha utilizado Material Design 3 [12] como guía de diseño, aplicando una paleta de colores personalizada basada en distintos tonos verdes (verde principal #4CAF82, verde oscuro #2D5F4F y verde claro #E8F5E9) que transmite calma, conexión con la naturaleza y bienestar, valores coherentes con el propósito de la aplicación.

La aplicación cuenta con cuatro flujos diferenciados según el rol del usuario:

Flujo de visitante

- **Pantalla de splash:** Pantalla inicial que muestra el nombre de TheraPets con una barra de carga progresiva antes de redirigir a la pantalla de bienvenida.
- **Pantalla de bienvenida:** Presenta la aplicación con opciones para iniciar sesión o registrarse.

- **Pantalla de inicio de sesión:** Permite el acceso mediante correo electrónico y contraseña, con opción de recuperación de contraseña.
- **Pantalla de registro:** Formulario para la creación de una nueva cuenta con datos personales.
- **Pantalla de recuperación de contraseña:** Permite solicitar un enlace de restablecimiento mediante correo electrónico.

Flujo de usuario registrado

- **Pantalla de inicio:** Muestra la información general del usuario, próxima solicitud de cita y acceso a las distintas secciones de la aplicación.
- **Flujo de agendar cita:** Proceso dividido en cuatro etapas. El usuario selecciona fecha, centro, animal con terapeuta y motivo de la solicitud.
- **Pantalla de mis citas:** Muestra las solicitudes solicitadas y las citas pasadas, permitiendo cancelar solicitudes pendientes y valorar sesiones finalizadas.
- **Pantalla de testimonios:** Permite consultar y publicar opiniones sobre la experiencia.
- **Pantalla de perfil:** Muestra los datos del usuario con opciones de edición y configuración.

Flujo de coordinador

- **Panel de coordinador:** Permite gestionar los animales del centro, los horarios disponibles y las solicitudes de cita, pudiendo aceptarlas o rechazarlas.

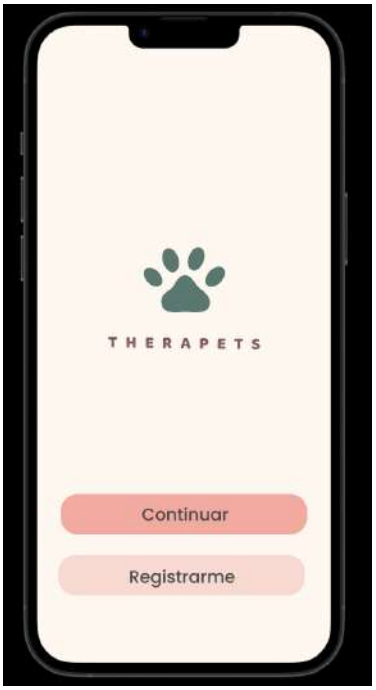
Flujo de administrador

- **Panel de administración:** Permite gestionar los centros terapéuticos, crear y eliminar coordinadores asignándoles a un centro, y supervisar las solicitudes del sistema.

Pantalla principal (administrador, coordinador de centro y usuario)

Boceto en Figma	Versión final
	

Pantalla login (administrador, coordinador de centro y usuario)

Boceto en Figma	Versión final
	

Pantalla inicio (administrador)

Boceto en Figma	Versión final
	

Pantalla inicio (coordinador de centro)

Boceto en Figma	Versión final
	

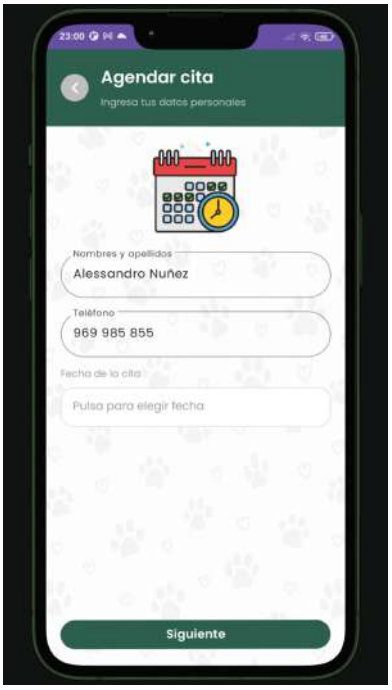
Pantalla inicio (usuario)

Boceto en Figma	Versión final
	

¿Cómo surgió TheraPets? (usuario)

Boceto en Figma	Versión final
	

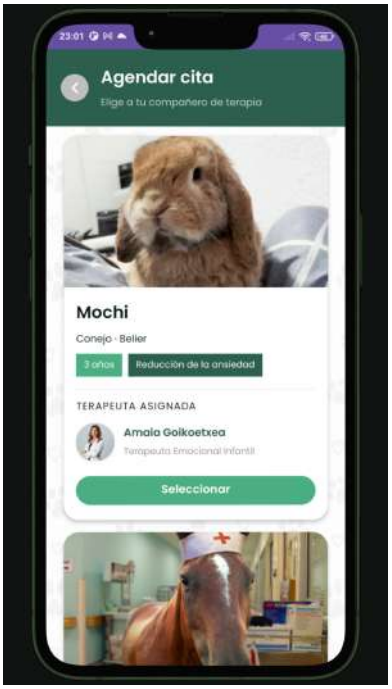
Solicitar cita (nombres, apellidos y fecha)

Boceto en Figma	Versión final
	

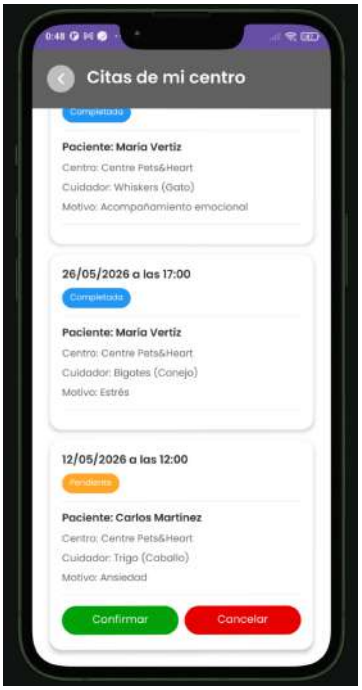
Solicitar cita (seleccionar centro)

Boceto en Figma	Versión final
	

Solicitar cita (seleccionar animal de compañía y terapeuta)

Boceto en Figma	Versión final
	

Confirmar solicitud de la cita (gestión coordinador de centro)

Boceto en Figma	Versión final
	

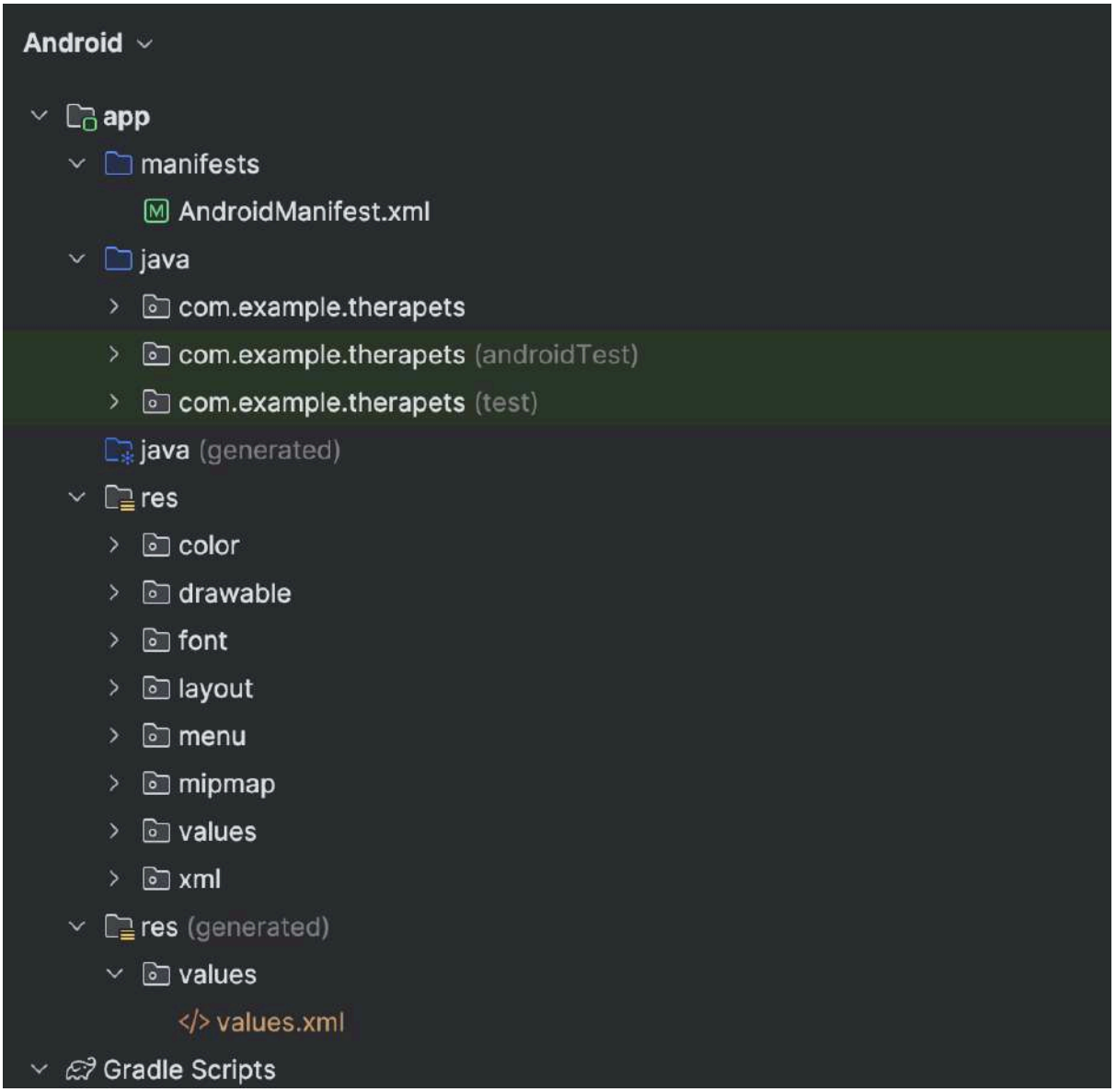
La navegación sigue un flujo coherente: el usuario accede, consulta su estado, agenda una sesión y puede gestionarla posteriormente. Los paneles de coordinador y administrador son independientes del flujo de usuario y solo son accesibles según el rol asignado en el sistema.

En el Anexo B se encuentran una serie de pantallas adicionales que conforman la aplicación para entender este flujo en su totalidad.

5. Desarrollo

5.1 Estructura del proyecto

El proyecto sigue la estructura estándar de una aplicación Android con Java, organizada dentro del paquete principal: **com.example.therapets.**



Activities y Fragments

Las Activities representan las pantallas principales de la aplicación, como el inicio de sesión, el registro o los paneles de administración y coordinación. Los Fragmentos se utilizan como componentes reutilizables dentro de las determinadas pantallas, permitiendo dividir funcionalidades complejas en diferentes secciones. Un ejemplo de ello es el flujo de la solicitud de cita, dividido en varios pasos, así como las pestañas de citas próximas y pasadas.

Modelos

Los modelos representan las entidades de datos de la aplicación y se corresponden directamente con las colecciones almacenadas en Firebase Firestore, como Usuario, Animal, Centro, Solicitud deCita, Horario y Testimonio.

Adapters

Los **Adapters** mostrarán la lista en los RecyclerView. Hay un adapter para cada caso: las citas del usuario, las citas del coordinador, los animales de la home, la selección de animal al agendar, los testimonios y las tarjetas de los centros con verificación de disponibilidad según el horario.

Utilidades

Las **Utilidades** son las clases de apoyo usado en distintas partes de la app. Entre ellas destaca CitaDraftStore, encargada de almacenar temporalmente la información introducida durante el flujo de solicitud de cita y GestorHorarios consulta Firestore y muestra las horas disponibles según el día elegido. Asimismo, la clase TheraPetsApp inicializa servicios externos necesarios para el funcionamiento de la aplicación, como Cloudinary para la gestión de imágenes.

Recursos visuales

La carpeta res/ contiene todos los recursos visuales utilizados por la aplcación, incluyendo layouts XML, imágenes e iconos, colores, estilos, menús y tipografía, que se aplica de manera global en toda la aplicación.

5.2 Desarrollo de funcionalidades

5.2.1 Autenticación y gestión de sesión

El sistema de autenticación gestiona el registro, el inicio de sesión, el cierre de sesión, la recuperación de contraseña y la eliminación de cuenta.

Al registrarse, el usuario introduce su nombre completo, correo electrónico, teléfono, fecha de nacimiento y contraseña. Firebase Authentication crea la cuenta y devuelve un UID único. Se genera un documento en la colección usuarios de Firestore con ese UID como identificador, almacenando el resto de los datos junto con el rol “usuario” por defecto.

El inicio de sesión comprueba las credenciales contra Firebase Authentication y una vez verificadas, consulta el documento del usuario en Firestore para leer su rol correspondiente. Según el rol, la aplicación redirige a la pantalla principal Home si es usuario, al panel de

administrador si es el administrador o al panel de coordinador si es el coordinador del centro. Esta lógica de redirección se centraliza en una única función dentro de la actividad del login para evitar duplicar comprobaciones en otras partes del código.

La sesión persiste automáticamente gracias a Firebase, de modo que si el usuario cierra la aplicación sin cerrar sesión, al volver a abrirla se redirige de manera automática a su pantalla principal sin necesidad de volver a iniciar sesión. Para la recuperación de contraseña se utiliza el envío de correo de restablecimiento que ofrece Firebase, evitando tener que implementar manualmente todo el flujo de tokens y validaciones.

Usar Firebase Authentication en lugar de implementar un sistema de autenticación propio fue una decisión que permitió delegar la gestión segura de credenciales en Firebase, mantener la persistencia de sesión y reducir el riesgo de errores de seguridad en una funcionalidad crítica.

5.2.2 Sistema de roles

La aplicación cuenta con tres roles diferenciados: usuario, coordinador y administrador. El rol se almacena en el campo rol del documento de cada usuario en Firestore.

El rol determina a qué pantalla accede el usuario tras iniciar sesión y qué funcionalidades tiene disponibles dentro de la app. Los usuarios tienen acceso a la pantalla principal donde pueden agendar citas, subir sus testimonios y gestionar su perfil. Los coordinadores acceden a un panel propio desde el que gestionan únicamente la información de su centro. El administrador tiene acceso total al sistema, pudiendo crear y gestionar centros como coordinadores.
Los coordinadores se crean exclusivamente desde el panel de administrador, ya que requiere un campo adicional llamado centroId que los vincula con el centro al que pertenecen. Cuando un coordinador inicia sesión, la aplicación lee este campo y filtra todas las consultas de Firestore para mostrarle únicamente la información relativa a su centro.

5.2.3 Flujo de solicitud de cita

La solicitud de cita es la funcionalidad central de la aplicación y se desarrolló como un flujo de cuatro pasos implementados mediante Fragments que se reemplazan dinámicamente dentro de una misma Activity contenedora.

En el paso 1 el usuario introduce sus datos personales (nombres, apellidos y teléfono) y selecciona la fecha de la cita mediante un MaterialDatePicker. En el paso 2 se muestran los centros disponibles en formato de tarjetas con su foto, dirección y teléfono. Al seleccionar un centro se muestra un diálogo con las horas disponibles para el día elegido. En el paso 3 se muestran los animales disponibles en el centro seleccionado, cada uno con su foto, raza, edad, especialidad y la terapeuta asignada con su foto. En el paso 4 el usuario indica el motivo de la consulta mediante un desplegable con opciones ya predefinidas y un campo que es opcional para especificar o añadir más detalles respecto a su cita.

Para mantener los datos entre los distintos pasos sin necesidad de pasar información entre Fragments a través de Bundles, se creó la clase CitaDraftStore que actúa como almacén estático temporal durante todo el flujo. Cada paso lee y escribe en este almacén, y al

finalizar el proceso o cancelarlo se vacía. **Para entender cómo se ha implementado este flujo, en el Anexo C se especifica.**

Una parte importante de este flujo es la verificación automática de disponibilidad horaria. Cuando se cargan las tarjetas de centros en el paso 2, la aplicación consulta en Firestore si existe algún horario configurado para ese centro en el día de la semana correspondiente a la fecha elegida. Si no hay horarios, el botón de selección aparecerá como desactivado automáticamente. Para determinar el día de la semana se utilizó SimpleDateFormat con localización en español, aplicando una capitalización manual de la primera letra para que coincidiera exactamente con el formato almacenado en Firestore.

Se decidió implementar el flujo como Fragments dentro de una sola Activity en lugar de Activities separadas porque permite usar addToBackStack para que el botón de volver funcione de manera natural, y mantiene una experiencia visual más fluida sin transiciones bruscas entre pantallas. También se decidió hacer la verificación de disponibilidad automática en lugar de mostrar un mensaje al solo pulsar, porque mejora notablemente la experiencia del usuario que de forma inmediata puede ver qué centros tienen disponibilidad. Esta funcionalidad está implementada completamente.

5.2.4 Gestión de citas

Las solicitudes de cita y las citas confirmadas pueden ser gestionadas tanto por el usuario como por el coordinador del centro correspondiente.

El usuario consulta sus citas desde la sección “Mis citas”, organizada en dos pestañas mediante un TabLayout con **ViewPager2**: próximas y pasadas. La pestaña de próximas muestra las solicitudes con estado “pendiente” (esperando confirmación) y “confirmada” (ya aceptada). La pestaña de pasadas muestra las citas con estado “completada” y las canceladas, tanto por el usuario como por el coordinador. Cada tarjeta muestra la fecha, hora, centro, animal, terapeuta y motivo, junto con un **badge** que indica el estado actual.

El coordinador accede a las citas de su centro desde el panel “Citas de mi centro”. La aplicación consulta filtrando por el **campo** centro de cada centro y comparándolo con el **centroId** del coordinador, de modo que solo ve las citas relacionadas con su centro. Desde su panel puede confirmar, cancelar o marcar cada cita como completada mediante botones que cambian según el estado actual.

Tanto el usuario como el coordinador deben indicar un motivo al cancelar una cita mediante un **AlertDialog**. Si cancela el usuario el estado pasa a “cancelada_usuario”, y si cancela el coordinador pasa a “cancelada_coordinador”. En ambos casos el motivo se guarda en el campo **motivoCancelacion** y queda visible para la otra parte. Esto permite que el usuario entienda por qué se canceló su cita sin necesidad de un sistema de notificaciones externo.

Diferenciar los estados “cancelada_usuario” y “cancelada_coordinador” permite identificar fácilmente quién canceló la cita y mostrar mensajes adecuados en cada contexto. Agregar un motivo al cancelar una cita refleja el flujo real de una consulta profesional y aporta información útil a la otra parte.

5.2.5 Valoración de sesiones

Cuando una cita ha sido marcada como completada por el coordinador, el usuario puede valorarla desde la pestaña de citas pasadas. La pantalla de valoración muestra cinco emojis que representan estados de ánimo de menor a mayor (mal, regular, bien, muy bien y genial) y un campo opcional para añadir una nota.

Al seleccionar un emoji, este queda resaltado visualmente para confirmar la elección. Al pulsar el botón de guardar, la valoración se almacena en la colección **valoraciones** de Firestore con el UID del usuario, el ID de la cita, el centro, la fecha, la puntuación y la nota.

Usar emojis en lugar de un sistema tradicional de estrellas encaja mejor con el carácter emocional de la aplicación y resulta más visual e intuitivo para el usuario.

5.2.6 Sistema de testimonios y likes

Los testimonios permiten a los usuarios compartir su experiencia con TheraPets. Desde la sección de testimonios, cualquier usuario puede consultar publicaciones de otros y crear las suyas propias. Al publicar un testimonio, el usuario puede añadir un comentario, una foto opcional y una valoración con estrellas mediante un RatingBar. Al guardar, la aplicación consulta el documento del usuario en Firestore para obtener su nombre y foto de perfil, garantizando que los datos mostrados estén siempre actualizados aunque el usuario los modifique posteriormente.

Cada testimonio se muestra en una tarjeta que incluye la foto del usuario, su nombre, la fecha de publicación, las estrellas de valoración, la foto opcional del testimonio, el comentario y el contador de likes. Si el testimonio es del propio usuario, aparece un botón para eliminarlo.

Las publicaciones pueden recibir “me gusta” de otros usuarios, que tienen un icono y un contador. Para evitar que un usuario pueda dar más de un like al mismo testimonio, cada documento de testimonio incluye un array **likesUsuarios** con los UIDs de los usuarios que ya han dado like. Antes de registrar un nuevo like, la aplicación comprueba si el UID del usuario actual ya está en ese array. Si lo está, muestra un mensaje indicando que ya dio me gusta. Si no, añade el UID mediante **FieldValue.arrayUnion()** e incrementa el contador en una sola operación, evitando inconsistencias.

Guardar los likes como un array de UIDs en lugar de una colección separada resulta más sencillo de consultar y evitar tener que hacer una segunda lectura de Firestore para saber si el usuario ya dio like. Mostrar la foto del usuario junto al testimonio permite generar una experiencia más cercana y reconocible para los usuarios.

5.2.7 Perfil y configuración

Desde la pantalla de perfil, el usuario puede consultar y modificar sus datos personales. La pantalla muestra la foto de perfil, el nombre, correo electrónico, teléfono y la fecha de nacimiento, leídos desde Firestore al cargar el fragmento.

El usuario puede cambiar su foto de perfil pulsando sobre ella, lo que abre el selector de imágenes del dispositivo. Tras seleccionar una imagen, esta se sube a Cloudinary y la URL resultante se guarda en el campo fotoUrl del documento del usuario en Firestore. El cambio

se refleja inmediatamente tanto en el perfil como en la pantalla de inicio. Desde el botón “Editar perfil” el usuario puede modificar su nombre y teléfono.

Desde la pantalla de configuración, accesible desde el perfil, el usuario puede cambiar su contraseña, activar o desactivar las notificaciones, cerrar sesión o eliminar su cuenta. El cambio de contraseña redirige a la pantalla de recuperación, donde Firebase envía un correo con el enlace de restablecimiento. La eliminación de cuenta primero pide confirmación mediante un **AlertDialog**, ya que es una acción irreversible. Si el usuario confirma, la aplicación elimina primero el documento del usuario en Firestore y después la cuenta en Firebase Authentication.

Se decidió incluir verificaciones con **isAdded()** en todos los callbacks asíncronos de Firestore y Cloudinary para evitar crashes cuando el usuario cambia rápidamente de pantalla mientras una operación está en curso. El switch de notificaciones está implementado visualmente pero no afecta a un sistema real de envío de notificaciones push, por lo que esta funcionalidad está implementada parcialmente, dejando el switch preparado para una futura versión.

5.2.8 Panel de administrador

El panel de administrador permite gestionar los centros de TheraPets, los coordinadores y consultar todas las citas del sistema.

La gestión de centros incluye operaciones de creación, edición y eliminación. Al crear un centro, se introduce el nombre, dirección, teléfono y una foto, que se sube a Cloudinary. Al editar, los cambios se aplican directamente al documento en Firestore. Al eliminar, se pide confirmación mediante un **AlertDialog** y se borra el documento.

La gestión de coordinadores permite crear nuevos coordinadores asignándoles un centro y eliminarlos. Cuando se crea un coordinador, se genera su cuenta en Firebase Authentication y su documento en Firestore con el rol "coordinador" y el **centroId** correspondiente.

La consulta de citas muestra todas las citas del sistema en modo solo lectura, ordenadas por fecha. El administrador no puede modificar el estado de las citas porque esa responsabilidad recae en los coordinadores de cada centro.

No permitir al administrador modificar el estado de las citas genera una clara separación de responsabilidades entre roles.

5.2.9 Panel de coordinador

El panel de coordinador permite gestionar los animales del centro, los horarios disponibles y las citas recibidas.

La gestión de animales incluye operaciones de creación, edición y eliminación. Cada animal tiene un nombre, tipo, raza, edad, especialidad, foto y datos de la terapeuta asignada (nombre, especialidad y foto). Tanto la foto del animal como la de la terapeuta se suben a Cloudinary y se guardan como URLs en el documento del animal en Firestore.

La gestión de horarios permite definir las horas disponibles para cada día de la semana. El coordinador selecciona un día desde un **Spinner** y una hora mediante un **TimePicker**, y el horario se guarda en la colección **horarios** con el **centroId** del coordinador.

La pantalla "Mi Centro" muestra únicamente la información del centro al que está asignado el coordinador, sin posibilidad de añadir o eliminar centros. Esta restricción se implementa pasando un parámetro **soloMiCentro** a la actividad de gestión de centros, que oculta el botón de añadir y filtra la lista por el **centroId** del coordinador.

Se decidió reutilizar las mismas actividades de gestión para administrador y coordinador, pasando parámetros que activan o desactivan funcionalidades según el rol, en lugar de duplicar el código en actividades separadas.

5.2.10 Organización de imágenes con Cloudinary

Todas las imágenes de la aplicación (fotos de perfil, animales, terapeutas, centros y testimonios) se almacenan en Cloudinary mediante el SDK para Android [13]. El servicio se inicializa en la clase **TheraPetsApp**, que extiende **Application**, garantizando que esté disponible desde el arranque de la aplicación.

El proceso de subida es asíncrono: el usuario selecciona una imagen de la galería, la aplicación la sube a Cloudinary y, una vez completada la subida, recibe la URL segura de la imagen, que se guarda en el documento correspondiente de Firestore.

Para cargar las imágenes desde sus URLs se utiliza la librería Glide [14], que gestiona la caché y la carga asíncrona de forma eficiente. Glide también permite aplicar transformaciones como **centerCrop** para ajustar las imágenes a los **ImageView** sin deformarlas.

El uso de Cloudinary en lugar de Firebase Storage se debe a la disponibilidad de transformaciones automáticas de imagen, un plan gratuito y limpio y una integración sencilla con Android.

5.3 Pruebas

Las pruebas realizadas se han centrado en verificar el correcto funcionamiento de las funcionalidades principales de la aplicación, prestando especial atención al flujo de agendar citas, a la gestión de roles y a la comunicación con los servicios externos utilizados en el proyecto.

Las pruebas se han llevado a cabo de forma manual, ejecutando la aplicación tanto en dispositivos Android reales como en el emulador integrado en Android Studio. No se han implementado tests automatizados debido al alcance del proyecto y al tiempo disponible, pero sí se ha seguido un proceso sistemático de verificación tras la implementación de cada funcionalidad.

5.3.1 Pruebas funcionales

Tabla 7. Pruebas funcionales

Funcionalidad	Caso probado	Resultado
Registro	Datos válidos	Cuenta creada
Registro	Correo ya existente	Mensaje de error
Inicio de sesión	Credenciales correctas	Acceso a pantalla principal
Inicio de sesión	Credenciales incorrectas	Mensaje de error
Recuperar contraseña	Correo válido	Email enviado
Eliminar cuenta	Confirmación	Cuenta eliminada
Agendar cita	Fecha con horarios disponibles	Solicitud guardada en pendiente
Agendar cita	Hora ya ocupada	Marcada como ocupada
Cancelar cita (usuario)	Con motivo	Estado actualizado a “cancelada_usuario”
Cancelar cita (coordinador)	Con motivo	Estado actualizado a “cancelada_coordinador”
Valorar cita	Cita completada	Valoración guardada
Publicar testimonio	Con foto, comentario y estrellas de valoración	Testimonio publicado
Dar me gusta	Primera vez	Like registrado
Dar me gusta	Segunda vez	Mensaje de “Ya le diste me gusta”

5.3.2 Pruebas de roles

Tabla 8. Pruebas de roles

Rol	Acceso	Resultado
Usuario	Acceso a la home, citas, testimonio, perfil	Navegar en la aplicación como usuario
Coordinador	Acceso solo a su centro	Control de la información de su centro
Administrador	Acceso total al sistema	Navegar en el panel de administrador

5.3.3 Pruebas de integración

Tabla 9. Pruebas de integración

Servicio	Operación	Resultado
Firestore	Lectura de colecciones	Datos cargados correctamente
Firestore	Escritura y actualización	Datos guardados en tiempo real
Firebase Auth	Registro y login	Cuentas gestionadas correctamente
Cloudinary	Subida de imágenes	URLs almacenadas en Firestore
Glide	Carga de imágenes	Imágenes mostradas sin errores

5.3.4 Pruebas de interfaz

La interfaz se adapta correctamente a distintos tamaños de pantalla. Todos los componentes visuales mantienen su funcionalidad y aspecto.

5.3.5 Problemas detectados y resoluciones

Tabla 10. Problemas detectados y resoluciones

Problema	Causa	Solución
Consulta de horarios sin resultados	Diferencia entre mayúsculas y minúsculas	Capitalización manual del día de la semana
Cierres inesperados al navegar rápido	Callbacks asíncronos sobre fragmentos destruidos	Verificación con **isAdded()**
Múltiples instancias del mismo fragmento	Recargas innecesarias al pulsar el mismo icono	Comprobación del fragmento actual antes de reemplazar
Listado de centros en blanco en Paso 2	Firestore responde antes de adjuntar el fragmento	Verificación con **isAdded()** antes de actualizar el adapter

6. Conclusiones

6.1 Conclusiones generales

El resultado final de TheraPets es una aplicación móvil para Android funcional que cumple el objetivo principal con el que fue desarrollada: digitalizar el acceso a la terapia asistida con animales y conectar a usuarios con centros y profesionales especializados. TheraPets permite a los usuarios agendar citas, gestionar su perfil, valorar sesiones y compartir testimonios, mientras que coordinadores y administradores disponen de paneles específicos para la gestión de centros, animales, horarios y citas.

El resultado final se valora con una puntuación de 8.9 sobre 10, ya que cumple los objetivos planteados y ofrece un sistema funcional completo, aunque existen aspectos susceptibles de mejora en futuras iteraciones.

Entre las partes más destacadas del desarrollo se encuentran el sistema de roles, que permite una separación clara de funcionalidades sin solapamientos, y el flujo de agendado de citas, que resulta intuitivo gracias a su diseño en cuatro pasos. Por otro lado, el módulo de configuración es el más limitado, al no incluir todas las funcionalidades que podría tener una sección de este tipo en una aplicación comercial.

A nivel de aprendizaje técnico, el proyecto ha permitido profundizar en el uso de Fragments, el desarrollo de interfaces en XML para Android, y el uso de Firebase tanto en autenticación como en base de datos en tiempo real. Además, se ha integrado Cloudinary como solución eficiente para la gestión de imágenes, lo que aporta escalabilidad y simplificación del almacenamiento multimedia.

6.2 Consecución de objetivos

La mayoría de los objetivos planteados al inicio del proyecto se han cumplido. Se ha desarrollado una aplicación funcional que permite la gestión completa del flujo de citas, la creación y gestión de testimonios, la diferenciación de roles con sus respectivos paneles y la integración con servicios externos como Firebase y Cloudinary.

El único objetivo no implementado ha sido el seguimiento del avance del usuario, una funcionalidad orientada a mostrar la evolución emocional a lo largo de las sesiones. Esta funcionalidad no se ha incluido debido a la complejidad de su diseño y a la dificultad de definir una estructura que aportara valor real sin afectar a la claridad y usabilidad de la aplicación. Se considera una línea de mejora futura.

6.3 Valoración de la metodología y planificación

La metodología de trabajo se basó en una planificación por sprints utilizando Trello como herramienta principal de organización. Trello ha permitido tener una visión clara del estado del proyecto en todo momento, diferenciando tareas pendientes, en curso y completadas, así como reorganizar prioridades cuando ha sido necesario. Para el control de versiones del código se ha utilizado GitHub, realizando subidas periódicas del avance del desarrollo.

La planificación inicial fue compleja debido a la falta de experiencia previa en la organización de proyectos de este tipo. Sin embargo, a medida que el desarrollo avanzó, la metodología adoptada resultó cada vez más eficaz. Durante el proceso se produjeron bloqueos puntuales, principalmente relacionados con la falta de información sobre determinadas implementaciones y la complejidad de algunas funcionalidades. Estos se resolvieron mediante la división de problemas en tareas más pequeñas y un desarrollo iterativo.

En conjunto, la metodología utilizada ha sido adecuada para el alcance del proyecto y ha permitido una organización eficiente del desarrollo, facilitando la finalización del mismo con un resultado satisfactorio.

6.4 Visión a futuro

TheraPets es un proyecto con potencial de evolución hacia una aplicación real, ya que cubre una necesidad relacionada con el bienestar emocional que actualmente no se encuentra completamente digitalizada y que presenta una demanda creciente.

De cara a futuras ampliaciones, se identifican varias funcionalidades susceptibles de incorporación. Entre ellas, la implementación de un sistema de chat entre usuarios y coordinadores, que permita una comunicación más directa para la gestión de dudas o ajustes previos a las sesiones. Otra mejora relevante sería el desarrollo de un sistema de seguimiento del progreso del usuario, permitiendo visualizar su evolución emocional a lo largo del tiempo a partir de las valoraciones de las sesiones.

Asimismo, se podrían añadir pantallas de carga dinámicas para mejorar la percepción de fluidez de la aplicación, una sección de noticias relacionada con la terapia asistida con animales y el bienestar emocional, así como la incorporación de la posibilidad de realizar llamadas directas a los centros desde la propia aplicación, facilitando el acceso a usuarios que prefieran un contacto más tradicional.

En conjunto, estas mejoras permitirían ampliar las funcionalidades actuales y acercar la aplicación a un entorno de uso real dentro del ámbito del bienestar emocional.

7. Glosario

- **Adapter**: Componente de Android encargado de gestionar la presentación de listas de datos en un RecyclerView.
- **API** (Application Programming Interface): Conjunto de funciones que permite que distintos programas se comuniquen entre sí.
- **Cloudinary**: Servicio en la nube utilizado en el proyecto para el almacenamiento y la gestión de imágenes.
- **Colección**: Conjunto de documentos en Firestore. En este proyecto se utilizan colecciones como **usuarios**, **citas** o **testimonios**.
- **Firebase**: Plataforma de Google que ofrece servicios como autenticación, base de datos en la nube y almacenamiento.
- **Firebase Authentication**: Servicio de Firebase que gestiona el registro, inicio de sesión y autenticación de los usuarios.
- **Firestore**: Base de datos NoSQL en la nube de Firebase, organizada en colecciones y documentos.
- **Fragment**: Componente de Android que representa una porción de interfaz dentro de una Activity. Permite construir vistas modulares y reutilizables.
- **Glide**: Librería de Android utilizada para cargar imágenes desde URLs de forma asíncrona.
- **IDE** (Integrated Development Environment): Entorno de desarrollo integrado. En este proyecto se ha utilizado **Android Studio**.
- **Material Design**: Sistema de diseño desarrollado por Google que define las directrices visuales y de interacción para las aplicaciones Android.
- **RecyclerView**: Componente de Android que muestra listas o cuadrículas de elementos de forma eficiente.
- **UID** (Unique Identifier): Identificador único asignado a cada usuario al registrarse en Firebase Authentication.
- **UI** (User Interface): Interfaz de usuario. Conjunto de elementos visuales con los que el usuario interactúa.
- **XML** (eXtensible Markup Language): Lenguaje de marcado utilizado en Android para definir las interfaces de usuario y otros recursos de la aplicación.
- **ViewPager2**: Componente de Android que permite deslizar entre distintas pantallas o fragments, utilizado en este proyecto en combinación con el TabLayout.

8. Bibliografía

[1] Hospital Universitario 12 de Octubre. (2022, 19 de julio). *El Hospital 12 de Octubre reanuda de forma presencial la terapia asistida con perros en niños ingresados en la Unidad de Cuidados Intensivos y Reanimación Pediátrica*. Comunidad de Madrid. https://www.comunidad.madrid/hospital/12octubre/noticia/hospital-12-octubre-reanuda-forma-presencial-terapia-asistida-perros-ninos-ingresados-unidad

[2] Cátedra Animales y Sociedad URJC. (s.f.). *'Huellas de Colores': Un estudio constata que la terapia asistida con perros puede reducir el dolor de los pacientes pediátricos críticos*. Universidad Rey Juan Carlos. https://catedraanimalesysociedad.org/huellas-de-colores-un-estudio-constata-que-la-terapia-asistida-con-perros-puede-reducir-el-dolor-de-los-pacientes-pediatricos-criticos/

[3] Real Sociedad Canina de España. (2024, 12 de enero). *Día Mundial contra la Depresión: Las terapias asistidas con perros reducen un 60% la medicación de los pacientes*. Animal's Health. https://www.animalshealth.es/animaladas/dia-mundial-contra-depresion-terapias-asistidas-perros-reducen-60-medicacion-pacientes

[4] Atlassian. (s.f.). *Trello: Manage your team's projects from anywhere*. https://trello.com/

[5] GitHub. (s.f.). *GitHub*. https://github.com/

[6] Android Developers. (s.f.). *Android Studio*. https://developer.android.com/studio

[7] Oracle. (s.f.). *Java Documentation*. https://docs.oracle.com/en/java/

[8] Firebase. (s.f.). *Cloud Firestore*. https://firebase.google.com/docs/firestore

[9] Firebase. (s.f.). *Firebase Authentication*. https://firebase.google.com/docs/auth

[10] Figma. (s.f.). *Figma: The collaborative interface design tool*. https://www.figma.com/

[11] JGraph Ltd. (s.f.). *draw.io: Flowchart Maker and Online Diagram Software*. https://www.drawio.com/

[12] Material Design. (s.f.). *Material Design 3*. https://m3.material.io/

[13] Cloudinary. (s.f.). *Android SDK | Cloudinary Documentation*. https://cloudinary.com/documentation/android_integration

[14] Bumptech. (s.f.). *Glide: An image loading and caching library for Android focused on smooth scrolling*. GitHub. https://github.com/bumptech/glide

9. Licencia y uso de la IA

Este proyecto está licenciado bajo la Creative Commons Attribution 4.0 International (CC BY 4.0). Esto significa que cualquier persona puede usar, compartir y modificar el trabajo, incluso con fines comerciales, siempre que se sigan los términos establecidos por la licencia.

Más información sobre la licencia: https://creativecommons.org/licenses/by/4.0/

En la elaboración de este proyecto se ha utilizado la inteligencia artificial únicamente como herramienta de apoyo. Su uso se ha centrado principalmente en la corrección del texto de la memoria y en la resolución de dudas o apoyo en la parte práctica de código, actuando como una especie de segunda opinión. Todas las ideas, la planificación y el desarrollo del proyecto han sido siempre responsabilidad propia, siendo el trabajo intelectual y creativo íntegramente mío.

10. Anexos

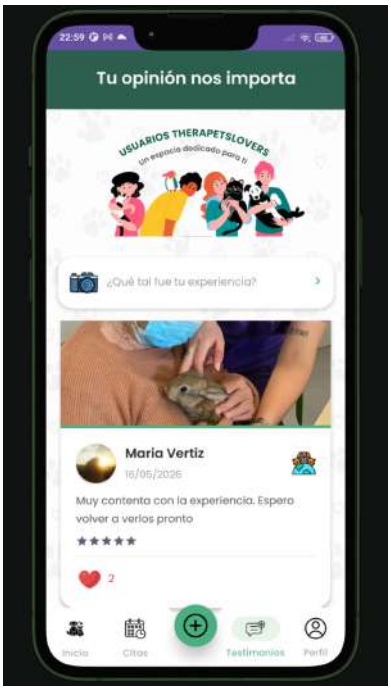
Anexo A - Credenciales de prueba

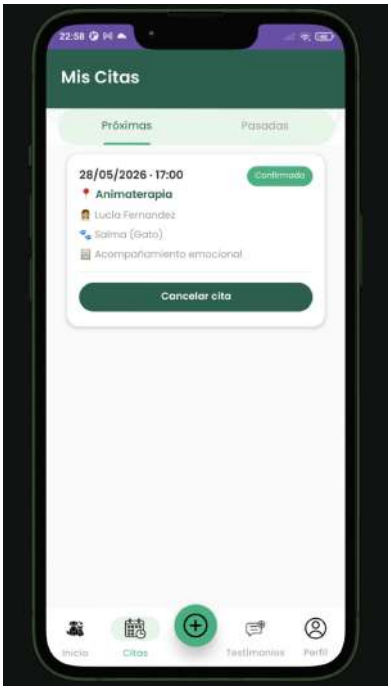
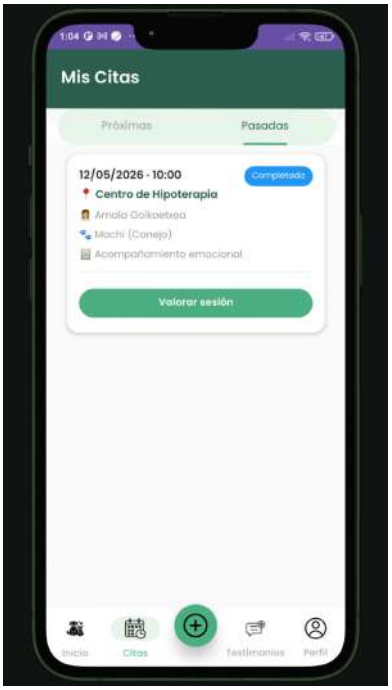
Administrador: Acceso completo al sistema: gestión de centros, coordinadores y visualización de todas las citas.

Coordinador: Acceso al panel del centro asignado: gestión de animales, horarios y citas del centro.

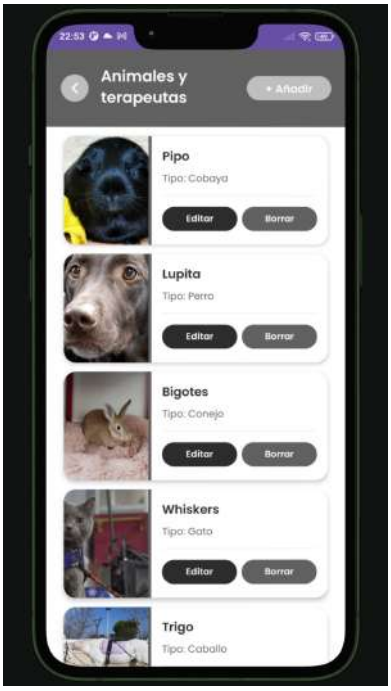
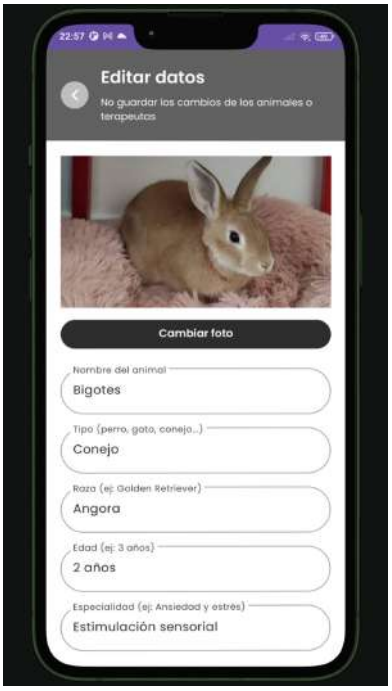
Usuario: Puede registrarse libremente desde la app desde la pantalla de registro.

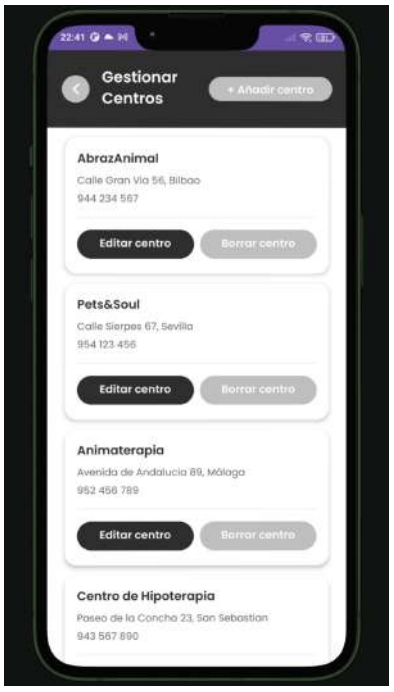
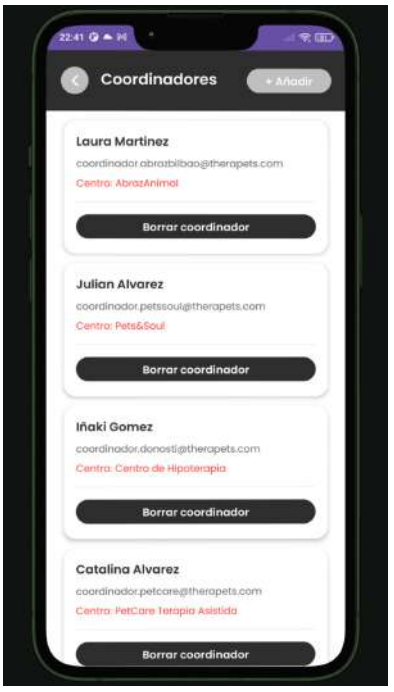
Anexo B

Pantalla de testimonios existentes	Pantalla compartir testimonio nuevo
	

Pantalla próximas citas	Pantalla citas pasadas
	

Pantalla recuperar contraseña	Pantalla correo de recuperación de contraseña
	

Pantalla de coordinador de centro - Animales y terapeutas disponibles	Pantalla de coordinador de centro - Editar datos de animales y terapeutas
	

Pantalla de administrador - Centros	Pantalla de administrador - Coordinadores de centros
	

Anexo C.1: Almacén temporal del flujo de agendado

```java
package com.example.therapets;

44 usages
public class CitaDraftStore {
    5 usages
    public static String nombres = "";
    public static String telefono = "";
    public static String fecha = "";
    public static String hora = "";
    public static String centro = "";
    4 usages
    public static String cuidador = "";
    7 usages
    public static String motivo = "";
    7 usages
    public static String otroMotivo = "";
    4 usages
    public static String nombreTerapeuta;

    public static void clear() {
        nombres = telefono = fecha = hora = "";
        centro = "";
        cuidador = "";
        motivo = "";
        otroMotivo = "";
        nombreTerapeuta = "";
    }
}
```

Anexo C.2: Verificación de disponibilidad horaria

```java
private void verificarDisponibilidad(String centroNombre, ViewHolder holder) {
    // El botón se desactiva hasta que sepamos si hay horarios
    holder.btnSeleccionar.setText("Comprobando si hay citas...");
    holder.btnSeleccionar.setEnabled(false);
    holder.btnSeleccionar.setBackgroundTintList(android.content.res.ColorStateList.valueOf(holder.itemView.getContext().getColor(android.R.color.darker_gray)));

    try {

        //Se convierte la fecha en el día de la semana (con mayúscula inicial)
        //para que coindica con el formato guardado en Firestore.
        Date date = new SimpleDateFormat( pattern: "dd/MM/yyyy", Locale.getDefault()).parse(CitaDraftStore.fecha);
        String dia = new SimpleDateFormat( pattern: "EEEE", new Locale( language: "es",  country: "ES")).format(date);
        dia = dia.substring(0, 1).toUpperCase() + dia.substring( beginIndex: 1);
        final String diaFinal = dia;

        FirebaseFirestore.getInstance().collection( collectionPath: "horarios").whereEqualTo( field: "centroId", centroNombre)
                .whereEqualTo( field: "dia", diaFinal).get().addOnSuccessListener( QuerySnapshot snap -> {
                    if (snap.isEmpty()) {
                        holder.btnSeleccionar.setText("Sin horarios");
                        holder.btnSeleccionar.setEnabled(false);
                        holder.btnSeleccionar.setBackgroundTintList(
                                android.content.res.ColorStateList.valueOf(holder.itemView.getContext().getColor(android.R.color.darker_gray)));
                    } else {
                        holder.btnSeleccionar.setText("Seleccionar");
                        holder.btnSeleccionar.setEnabled(true);
                        holder.btnSeleccionar.setBackgroundTintList(
                                android.content.res.ColorStateList.valueOf(
                                        holder.itemView.getContext().getColor(R.color.morado_principal)));
                    }});
    } catch (Exception e) {
```

Anexo C.3: Sistema de likes con FieldValue.arrayUnion()

```
boolean yaLeDioLike = t.getLikesUsuarios() != null && t.getLikesUsuarios().contains(uidActual);
holder.btnMeGusta.setImageResource(yaLeDioLike ? R.drawable.corazon : R.drawable.amor);

// Like a publicación
holder.btnMeGusta.setOnClickListener( View v -> {
    if (t.getLikesUsuarios() != null && t.getLikesUsuarios().contains(uidActual)) {
        mostrarToast(v.getContext(), mensaje: "Ya le diste me gusta");
        return;
    }

    FirebaseFirestore.getInstance().collection( collectionPath: "testimonios").document(t.getId()).update(
            field: "meGusta", value: t.getMeGusta() + 1, ...moreFieldsAndValues: "likesUsuarios", FieldValue.arrayUnion(uidActual)
        )
        .addOnSuccessListener( Void a -> {
            t.setMeGusta(t.getMeGusta() + 1);

            if (t.getLikesUsuarios() != null) {
                t.getLikesUsuarios().add(uidActual);
            }

            holder.tvContadorMeGusta.setText(String.valueOf(t.getMeGusta()));

            // Cambio del icono
            holder.btnMeGusta.setImageResource(R.drawable.corazon);
        });
});
```

Anexo C.4: Subida de imágenes a Cloudinary

```
//Subir las fotos a Cloudinary
1 usage
private void subirFotoYPublicar(String comentario) {
    MediaManager.get().upload(fotoSeleccionada).callback(new UploadCallback() {
        @Override public void onStart(String requestId) {
        }

        @Override public void onProgress(String requestId, long bytes, long totalBytes) {
        }

        @Override
        public void onSuccess(String requestId, Map resultData) {
            String fotoUrl = resultData.get("secure_url").toString();
            publicarTestimonio(comentario, fotoUrl);
        }

        @Override
        public void onError(String requestId, ErrorInfo error) {
            if (!isAdded()) return;
            mostrarToast( mensaje: "Error al subir foto");
        }
        2 usages
        @Override public void onReschedule(String requestId, ErrorInfo error) {}
    }).dispatch();
}
```

```
private void publicarTestimonio(String comentario, String fotoUrl) {
    String uid = FirebaseAuth.getInstance().getCurrentUser().getUid();
    String fecha = new SimpleDateFormat( pattern: "dd/MM/yyyy", Locale.getDefault()).format(new Date());

    FirebaseFirestore.getInstance().collection( collectionPath: "usuarios").document(uid).get().addOnSuccessListener( DocumentSnapshot document -> {
        if (!isAdded()) return;

        String nombre = document.getString( field: "nombre");
        String fotoUsuarioUrl = document.getString( field: "fotoUrl") != null ? document.getString( field: "fotoUrl") : "";

        Map<String, Object> testimonio = new HashMap<>();
        testimonio.put( k: "usuarioId", uid);
        testimonio.put( k: "nombreUsuario", nombre);
        testimonio.put( k: "comentario", comentario);
        testimonio.put( k: "fecha", fecha);
        testimonio.put( k: "meGusta",  v: 0);
        testimonio.put( k: "fotoUrl", fotoUrl);
        testimonio.put( k: "fotoUsuarioUrl", fotoUsuarioUrl);
        testimonio.put( k: "estrellas", estrellasSeleccionadas);
        testimonio.put( k: "likesUsuarios", new ArrayList<>());

        FirebaseFirestore.getInstance().collection( collectionPath: "testimonios").add(testimonio).addOnSuccessListener( DocumentReference ref -> {
            if (!isAdded()) return;
            requireActivity().getSupportFragmentManager().popBackStack();
        }).addOnFailureListener( Exception e -> {
            if (!isAdded()) return;
            mostrarToast( mensaje: "Error al publicar");
        });
    });
```