ULPGC
Universidad de
Las Palmas de
Gran Canaria

ESCUELA DE
INGENIERÍA INFORMÁTICA

Trabajo de Fin de Grado

Aplicación web para aficionados a la colombofilia

Grado en Ingeniería Informática

AUTOR: Antonio Juan Peñate Peña

TUTORIZADO POR:

Dr. Luis Miguel Hernández Acosta

Julio 2024

Agradecimientos

Quiero agradecer a toda mi familia, pareja y amigos por ayudarme en todo momento y darme los ánimos necesarios para no desistir.

A mi tutor, Dr. Luis Miguel Hernández Acosta, por ayudarme y guiarme para hacer realidad este proyecto.

Resumen

Actualmente, existen pocas aplicaciones para la administración de palomares. Además, las existentes no están en español o no tienen todas las funcionalidades que un colombófilo precisa.

Mediante la realización de este TFG, se propone el desarrollo de una aplicación web que permita al aficionado a la colombofilia llevar un registro de las aves que tiene en su palomar y poder controlar el pedigrí de sus aves. Esta aplicación también incluirá un apartado para que el aficionado pueda registrar los datos de las competiciones de sus aves. Además, incorporará también una sección donde se publicarán noticias o información de interés para el aficionado a la colombofilia.

Abstract

Nowadays, there are just a few applications for the management of pigeon lofts. What is more, the existing ones are not in Spanish or they don't have all the functionalities that a pigeon fancier needs.

This final project proposes the development of a web application that allows pigeon fanciers to keep a track of the pigeons they have in their loft and to control their pedigree. This application will also include a section for the pigeon fancier to register the competitions data of his pigeons. Additionally, it will as well include a section where news or information of interest will be published.

Índice

Índice de Figuras

Índice de Tablas

Capítulo 1 – Introducción

1.1 Contexto

1.1.1 ¿Qué es la colombofilia?

Según la Real Academia Española, la colombofilia es la *"cría y adiestramiento de palomas, especialmente mensajeras."* [1] Para un aficionado, esta definición puede quedarse corta, ya que la colombofilia representa una verdadera pasión por estos animales. Un colombófilo se dedica a entrenar a sus palomas como si fueran atletas, invirtiendo muchos recursos para asegurar que sus aves sean fuertes, rápidas y estén en buen estado de salud, para que puedan ganar las diferentes competiciones que se celebran.

La cualidad que posee la paloma para orientarse y volver a su palomar es el motivo por el que se ha usado para la entrega de mensajes. Se tiene constancia del uso de las palomas para enviar comunicaciones desde la antigüedad, y no es hasta la llegada de las comunicaciones modernas en las que cae en desuso su empleo. La entrega de mensajes tuvo una importancia relevante durante la Primera y la Segunda Guerra mundial, debido a que las comunicaciones modernas como el telégrafo o la radio sufrían sabotajes o fallos. En esos conflictos bélicos las palomas se consideraban un sistema fiable para entregar el mensaje [2].

Actualmente, la colombofilia es una práctica eminentemente deportiva. La práctica deportiva surgió a finales del siglo XIX en Bélgica y se ha ido expandiendo hacia otros lugares. Concretamente, en Canarias, la afición por las palomas es notable, lo cual se demuestra por la cantidad de clubs colombófilos que hay repartidos por todas las islas. Según la web de la Federación Canaria de Colombofilia, actualmente hay censados 43 clubes [3]. Además, si buscamos en el Anuario de Estadísticas Deportivas, creado por la Subdirección General de Estadística y Estudios de la *Secretaría General Técnica del Ministerio de Educación, Formación Profesional y Deportes*, se muestra que en el año 2023 había un total de 894 licencias federativas en este deporte en Canarias. La siguiente comunidad autónoma con más licencias es Andalucía con 568 [4]. Esto demuestra que Canarias es la comunidad autónoma donde más se practica la colombofilia en España.

La colombofilia en Canarias presenta un verdadero desafío debido a la geografía insular y su entorno marítimo, a diferencia del continente, donde las palomas pueden volar sobre tierra y detenerse para descansar antes de continuar su vuelo. En cambio, en Canarias, todas las competiciones se llevan a cabo sobre el mar, lo que implica que las palomas no tienen la opción de descansar hasta llegar a tierra firme. La mayoría de las competiciones, conocidas como "sueltas", se realizan entre las propias islas del archipiélago o, cuando las condiciones políticas y burocráticas lo permiten, desde la costa occidental de África. En ocasiones, estas sueltas desde África se sustituyen por sueltas en altamar desde un barco. Este entorno marítimo añade complejidad y enriquece aún más el desafío y la fascinación por la colombofilia en Canarias.

Precisamente, por esta singularidad, las pérdidas de palomas durante las sueltas pueden ser significativas si las aves no han sido adecuadamente preparadas para enfrentar estos desafíos

marítimos. Por ello, es importante que el colombófilo disponga de las herramientas adecuadas para asegurar una correcta selección de ejemplares que estén preparados para afrontar las condiciones de las sueltas.

1.1.2 ¿En qué consiste este deporte?

La colombofilia es un deporte que consiste en llevar a todas las palomas de los participantes a un punto específico para luego soltarlas. La paloma que regrese primero a su palomar es la ganadora de la competición.

Anteriormente, para determinar la hora a la que una paloma había llegado, se utilizaban relojes especiales. Los colombófilos debían introducir una anilla especial en la pata de la paloma antes de la suelta. Al regresar la paloma a su palomar, el colombófilo tenía que coger esa anilla e introducirla en el reloj. Al girar una rueda, se activaba un mecanismo que atrapaba la anilla y registraba la hora en la que se introdujo en el dispositivo. Posteriormente, en la sede del club se verificaban los relojes para determinar qué paloma había sido la ganadora.

Hoy en día, gracias a los nuevos modelos de relojes, se emplean anillas que incluyen un chip que registran automáticamente la hora de llegada cuando la paloma está dentro del campo de lectura de los sensores. Este método es considerablemente más preciso y fiable que el anterior, mejorando por tanto la fiabilidad al momento de determinar la llegada de una paloma.

Las competiciones se organizan en diferentes modalidades que dependen de la distancia de vuelo que deben cubrir las palomas, las cuales se especifican a continuación:

- **Velocidad**: hasta 300 Km
- **Medio Fondo**: más de 300 y menos de 500Km
- **Fondo**: más de 500 y menos de 700Km
- **Gran Fondo**: más de 700km

1.2 Motivación

La elección de esta temática para mi Trabajo de Fin de Grado surge de una experiencia personal. En su momento, mi primo y yo teníamos un pequeño palomar. Actualmente, mi primo sigue con esta afición y, en una de nuestras conversaciones, descubrí que registraba las palomas en un folio impreso, que luego transcribía a una hoja de cálculo. Esto me llamó la atención y descubrí que muchos colombófilos usan herramientas muy genéricas para llevar el control de sus palomares.

También observé que las aplicaciones disponibles no están en español o no son específicas para la colombofilia y, por lo tanto, carecen de todas las funcionalidades que un colombófilo necesita. Además, de las aplicaciones existentes, muchas solo tienen un enfoque: o bien son para el registro de competiciones o para llevar el control del palomar. Muy pocas combinan ambas funcionalidades.

Existe otro grupo de aplicaciones específicas para la gestión de clubes de colombofilia, pero, aparte de ser bastante caras, no están diseñadas para llevar el control de un palomar

individual. Por ejemplo, no permiten controlar el pedigrí de las palomas ya que su finalidad es la organización de competiciones y la presentación de resultados.

Con la realización de este trabajo, vi la oportunidad de desarrollar una aplicación propia que ayudara a los colombófilos en la gestión de sus palomares. Además, podría poner en práctica todos los conocimientos adquiridos durante la carrera y tratar de ofrecer una aplicación adaptada a las necesidades que un aficionado a la colombofilia pueda tener.

1.3 Objetivos

El objetivo de este trabajo es desarrollar el prototipo de una aplicación web adaptada para su visualización en dispositivos móviles y que permita al aficionado a la colombofilia llevar un registro detallado de las aves de su palomar. Mediante el uso de esta aplicación, se podrá:

- Tener un censo de todas las palomas que posee, además de facilitar la introducción de la información de la manera más sencilla y cómoda posible.
- Establecer el estado en el que se encuentra la paloma. De esta forma, podrá saber si está enferma, muerta, perdida, prestada o entre otros estados. Además, tendrá la oportunidad de establecer estados personalizados.
- Registrar los resultados de las competiciones en las que participan sus palomas.
- Controlar el pedigrí de las palomas, mostrando quienes son los ancestros de una determinada paloma. Este punto es de los más importantes, ya que, para mantener una buena estirpe de palomas, debe haber cierto grado de endogamia para sustentar la línea. Por lo tanto, es necesario conocer los ancestros del ave.
- Poder consultar una sección donde se pulique información y noticias relevantes sobre colombofilia. Esta sección estará disponible tanto para el usuario como para los visitantes.
- Publicar anuncios para la venta de palomas, pichones o cualquier material usado en la colombofilia como relojes, cestas o jaulas. Solo los usuarios registrados podrán llevar a cabo estas publicaciones.

Capítulo 2 Competencias

En este capítulo se expondrán las competencias específicas cubiertas durante la realización de este Trabajo de Fin de Grado

2.1 Competencias Específicas Desarrolladas

"TI3 Capacidad para emplear metodologías centradas en el usuario y la organización para el desarrollo, evaluación y gestión de aplicaciones y sistemas basados en tecnologías de la información que aseguren la accesibilidad, ergonomía y usabilidad de los sistemas." [5]

En la realización de este trabajo se han adaptado diferentes metodologías de desarrollo ágil para desarrollar una aplicación centrada en las necesidades del usuario y que, en la medida de lo posible, sea sencilla de manejar. Además, esta es una aplicación web que se ha diseñado de manera responsiva para asegurarse de que se muestre correctamente en los diferentes formatos de pantalla.

"TI6 Capacidad de concebir sistemas, aplicaciones y servicios basados en tecnologías de red, incluyendo Internet, web, comercio electrónico, multimedia, servicios interactivos y computación móvil." [5]

Al ser una aplicación web, es fundamental que utilice tecnologías de red para establecer una comunicación entre el servidor, en este caso *Firebase*, y el navegador del usuario final. Por lo tanto, es completamente necesario tener una conexión a internet para poder utilizar todas las funcionalidades de la aplicación.

"CI7 Conocimiento, diseño y utilización de forma eficiente de los tipos y estructuras de datos más adecuados a la resolución de un problema." [5]

Para el desarrollo de este trabajo se han empleado diferentes estructuras de datos estudiadas en las diferentes asignaturas de la carrera. Por ejemplo, se han utilizado *arrays* de objetos para traer las diferentes colecciones de *Firebase*, y un árbol binario para la representación del árbol genealógico de una paloma.

"TFG Ejercicio original a realizar individualmente y presentar y defender ante un tribunal universitario, consistente en un proyecto en el ámbito de las tecnologías específicas de la Ingeniería en Informática de naturaleza profesional en el que se sinteticen e integren las competencias adquiridas en las enseñanzas." [5]

El desarrollo de este trabajo es una idea genuina, en la que se emplean todos los conocimientos y técnicas aprendidas durante el transcurso de los estudios realizados.

Capítulo 3: Metodología y planificación del proyecto

En este capítulo se detalla la metodología usada para la realización de este proyecto, además de explicar la planificación del mismo.

3.1 Metodología

En un principio, para este proyecto se había considerado usar la metodología ágil *Scrum*, tal como se especificó en TFT01. Sin embargo, *Scrum* está orientado a equipos o grupos de trabajo y, dado que este proyecto es un trabajo individual, la metodología perdería gran parte de sus beneficios si se adaptara para un solo desarrollador. Por este motivo, finalmente se optó por utilizar la metodología Iterativa e Incremental, que se adapta mejor para un proyecto de este tipo.

3.1.1 Metodología Iterativa e Incremental

La metodología Iterativa e incremental surgió para subsanar las debilidades del modelo en cascada [6], tales como la poca flexibilidad para el cambio y la larga duración para la entrega del producto [7]. Esta metodología, se basa en el siguiente enfoque:

Metodología Iterativa:

El proyecto se divide en diferentes partes que se desarrollan en un periodo determinado de tiempo, conocido como iteración. En cada iteración se realizan las siguientes etapas:

- *Planificación, diseño, implementación y pruebas*: En cada iteración, se llevan a cabo todas estas actividades, que son típicas de un proyecto en cascada.
- *Evaluación del producto*: Esta es la última de las etapas de la iteración, donde se entrega el producto creado para que sea evaluado por los usuarios finales y se recogen aquellos aspectos necesarios para mejorar el producto y adaptarlo a las necesidades del usuario.
- *Nueva iteración*: Tras la validación y recogida de mejoras, se comienza con una nueva iteración. Las iteraciones se suceden hasta que se llega al producto final.

El problema de este enfoque de desarrollo es que el producto entregado no es el producto final, sino partes de él, y no estará completo. Por lo tanto, el usuario final no podrá hacer uso del producto hasta que finalice la última iteración, lo que implica que no podrá ofrecer una valoración real hasta el final.

Desarrollo Iterativo e Incremental:

Aprovechando el ciclo del modelo iterativo, en esta metodología el proyecto se divide en incrementos. Estos incrementos son partes funcionales del producto final y deben tener la capacidad de ser utilizadas por los clientes finales.

Con cada iteración, se van añadiendo mejoras y nuevas funcionalidades, así como correcciones de los incrementos anteriores, lo que permite que el producto adquiera cada vez más valor [8]. Este proceso se repite en cada iteración hasta llegar al producto final.

Con esta metodología, se persigue entregar en un periodo corto de tiempo un producto que tenga valor para las partes interesadas, que además pueda ser utilizado, y detectar de una manera temprana posibles problemas y necesidades adicionales, permitiendo así ajustes y mejoras continuas.

3.2 Planificación

En la siguiente tabla se puede ver la planificación que se entregó en el documento TFT01.

Fases	*Estimación (horas)*	*Tareas*
Estudio previo / Análisis	55	**Tarea 1.1:** Estudio de la situación actual.
		Tarea 1.2: Análisis y especificación de las tecnologías y herramientas a utilizar durante el proyecto.
		Tarea 1.3: Análisis y especificación de los requisitos finales.
Diseño / Desarrollo / Implementación	175	**Tarea 2.1:** Diseño de las vistas o “Mockups”
		Tarea 2.2: Diseño de la Base de Datos.
		Tarea 3.3: Desarrollo de la aplicación en Iteraciones.
Evaluación / Validación / Prueba	30	**Tarea 3.1:** Realización de pruebas de funcionamiento
		Tarea 3.2: Corrección de errores detectados
Documentación / Presentación	40	**Tarea 4.1:** Redacción de la Memoria del TFT
		Tarea 4.2: Redacción del manual de usuario
		Tarea 4.3: Realización de la presentación del TFT
		Tarea 4.4: Preparación de la defensa del TFT

Tabla 1 Planificación del Proyecto

En la primera fase del proyecto, las tareas se centraron en analizar las aplicaciones similares disponibles para extraer las fortalezas y debilidades que presentaban. Además, se estudiaron las tecnologías y herramientas adecuadas para el desarrollo del proyecto, y se adquirió conocimiento en aquellas tecnologías que no se dominaban, como *Angular Material* o

Firebase Storage. El último paso de esta fase fue la especificación de los requerimientos que debería cumplir la aplicación para ser competente.

En la segunda fase del proyecto, el primer paso fue crear los bocetos para visualizar cómo debería lucir la aplicación final. Además, se diseñó el logo para la aplicación. Posteriormente, se recopilaron los datos necesarios para diseñar y definir la estructura de la base de datos que serviría para almacenar los registros de usuarios, palomas, competiciones, noticias y anuncios de venta. Una vez completado el análisis y con el diseño listo, se procedió con la implementación de la aplicación.

Durante esta etapa, surgió un inconveniente al recoger y manejar diferentes formatos de fechas para el cálculo de las velocidades de las palomas. Como consecuencia fue necesaria la necesidad de invertir más horas en esta fase, afectando la planificación de la siguiente etapa del proyecto.

Cada vez que se terminaba de desarrollar una nueva funcionalidad, se realizaban pruebas funcionales para verificar el correcto funcionamiento del sistema. En caso de detectar errores, se procedía a su corrección. Esta fase se vio reducida debido a los contratiempos mencionados anteriormente, lo que llevó a restar 8 de las 30 horas planificadas, principalmente porque fue necesario corregir y refactorizar parte del código en una de las iteraciones para que la aplicación pudiera cumplir con los objetivos propuestos.

Capítulo 4: Legislación

Para la Publicación de una aplicación web como la que se ha desarrollado en este proyecto es fundamental cumplir con una serie de leyes que regulan el tratamiento de los datos personales, entre esas normativas se encuentran:

- *"Reglamento General de Protección de Datos (RGPD) o Reglamento (UE) 2016/679*" [9]
- "*Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales (LOPD-GDD)*" [10]

Estas normativas indican que el usuario debe ser debidamente informado sobre qué datos recoge la aplicación, qué se hace con sus datos y con quien se comparten esos datos. Para garantizar que la aplicación desarrollada cumpla con estas normativas, se han tomado las siguientes medidas:

Se ha añadido una página con la política de privacidad de la aplicación, donde el usuario pueda estar informado. En esta política se especifican los derechos que tiene el usuario sobre sus datos personales.

Al realizar el registro en la aplicación, el usuario debe marcar una casilla indicando que está de acuerdo con esta política de privacidad. De esta manera, se entiende que el usuario ha sido informado y ha dado su consentimiento para el tratamiento de sus datos.

Aparte de informar al usuario, se debe garantizar la seguridad de los datos. Durante el desarrollo de la aplicación, se han implementado diferentes controles tanto en la aplicación como en la base de datos para asegurar que sólo el propietario de los datos pueda acceder a su información privada.

Capítulo 5: Estado del Arte

5.1 Aplicaciones similares

En esta sección del trabajo, se analizarán algunas de las aplicaciones y herramientas disponibles que se utilizan para la administración de palomares o para gestionar las competiciones. El objetivo de este análisis es buscar aquellas características y funcionalidades que nos permitan crear un producto novedoso, así como valorar aquellos aspectos de aplicaciones existentes que nos puedan resultar útiles.

5.1.1 My Loft

Figura 1: Logotipo de "My Loft"

Autor: Kabbary

Fecha de publicación: 8 de julio de 2019.

Última Actualización: 18 de diciembre de 2020.

Número de descargas en Google Play: más de 10.000.

URL de Google Play: Actualmente ya no está disponible para su descarga.

Esta aplicación actualmente ya no está disponible para su descarga e instalación. Sin embargo, permitía llevar un registro de todas las palomas que habitaban en un palomar. Además, tenía la posibilidad de administrar las parejas e incluía notificaciones para avisar del momento aproximado en el que iban a eclosionar los huevos, lo que permitía realizar un seguimiento de los pichones. Aparte de esto, con esta aplicación se podía saber quiénes eran los padres de cada paloma, pudiendo trazar un pequeño árbol genealógico.

También disponía de un Marketplace donde poner anuncios para la venta de palomas, aunque esta funcionalidad dejó de funcionar hace bastante tiempo. Como primer aspecto negativo, debemos destacar el hecho de que solo era compatible con versiones antiguas de Android. Para el análisis de esta aplicación, se ha tenido que instalar en un *smartphone* con la versión 8 de Android, por lo que no era compatible con los móviles más modernos.

Otro de los aspectos negativos es que no permitía llevar un registro sobre las competiciones en las que participaban las palomas.

5.1.2 My Pigeon – Pedigree Management

Figura 2: Logotipo de "My Pigeon"

Autor: Dierta

Fecha de publicación: 11 de mayo de 2023.

Última Actualización: 13 de abril de 2023.

Número de descargas en Google Play: más de 1000.

URL de Google Play: https://play.google.com/store/apps/details?id=com.my_pigeon&hl=es

Esta aplicación está enfocada para la visualización del árbol genealógico de las palomas, pero también permite llevar un registro todas las palomas de un palomar.

En su versión gratuita, el número de anuncios que se muestran al usuario es elevado, lo que afecta a la usabilidad de la aplicación debido a las constantes interrupciones.

Entre los puntos débiles de esta aplicación se encuentra la incapacidad para subir fotos de las palomas en su registro. Además, no permite llevar un registro de las competiciones en las que ha participado la paloma, simplemente se puede poner una frase con el logro más importante, lo cual puede resultar insuficiente.

Otro de sus aspectos negativos es que la información mostrada en el árbol genealógico es demasiado escueta, simplemente se muestra el nombre de la paloma y su número de anilla.

5.1.3 Mis Pájaros – Gestión de aviarios

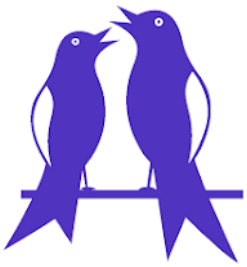

Figura 3: Logotipo de "Mis pájaros"

Autor: Daniel Medina

Fecha de publicación: 7 de octubre de 2017.

Última Actualización: 4 de agosto de 2023.

Número de descargas en Google Play: más de 100.000

URL de Google Play:
https://play.google.com/store/apps/details?id=com.danielme.mybirds&hl=es&gl=US

Esta aplicación permite llevar un registro bastante detallado de muchos tipos de aves. Algunas de funcionalidades más destacables son: registro de aves de diferentes especies, registrar parejas y crías, notificación de eclosión de huevos, seguimiento de genealogías, llevar un registro de aquellas aves que han sido compradas o vendidas y tener una galería de imágenes por cada ave.

Esta aplicación es muy completa, con una interfaz sencilla y muy fácil de usar. Pero presenta las siguientes desventajas. La aplicación no está destinada específicamente a las palomas. Por lo que al registrar al animal hay que especificar que se trata de una paloma. Además, faltaría añadir más campos específicos como el color o subespecie.

Otro punto flaco de esta aplicación es que, aunque tiene una parte para registrar concursos, este apartado es muy general y no está debidamente adaptado para registrar las competiciones en las que participan las palomas. El registro de concursos se compone de varios campos donde poner el nombre del concurso, la fecha de celebración, el premio, la puntuación y un campo de comentarios. El problema es que el campo de puntuación se basa en un campo tipo numérico que va de 0 a 100 cosa que puede resultar insuficiente para registrar la posición en la que queda una paloma en una competición.

5.1.4 Loftmanager Online

Figura 4: Logotipo de "Loftmanager Online"

Autor: Loft Manager Online

URL: https://www.loftmanageronline.com/index.php

Es una aplicación online que permite llevar un registro muy detallado de los palomares. Además, incluye una forma visual de mostrar el árbol genealógico de una paloma, así como listar a sus descendientes. También ofrece herramientas como facilitar emparejamientos para mantener el pedigrí, una calculadora de consanguinidad y la capacidad de exportar el registro de aves a diferentes formatos.

El diseño para pantallas móviles tiene bastante margen de mejora, especialmente en los formularios utilizados para registrar o editar cualquier aspecto de la aplicación.

Como identificador del animal, la aplicación utiliza el número de anilla, que es lo normal. De hecho, en la versión móvil, solo se muestra este número. Un criador de palomas puede recordar de memoria algún número de anilla, pero no todos. Por lo tanto, sería beneficioso mostrar algún dato adicional, como una miniatura de una foto del animal, para identificar el registro del animal de manera más sencilla.

5.1.4 PigeonDB.com

Figura 5: Logotipo de PigeonDB.com

Autor: AllBreed Solutions

URL: https://pigeondb.com

Esta aplicación online es muy completa y ofrece la posibilidad de llevar un registro detallado de las palomas en un palomar de colombófilo. Se asemeja mucho a "*Loft Manager Online*" y comparte muchas de sus funcionalidades, como mostrar el árbol genealógico de una paloma, listar los descendientes y herramientas para gestionar emparejamientos.

La principal diferencia con "*Loft Manager Online*" es que "*PigeonBD*" tiene un buen diseño responsivo que permite una correcta visualización en dispositivos móviles. Además, permite identificar a los animales mediante nombre e imagen, y ofrece varios modos de visualización de resultados (como tarjetas, solo foto y nombre, o en forma de tabla).

El punto negativo de esta aplicación es que, debido a la gran cantidad de opciones disponibles, su uso puede resultar algo complicado, especialmente al principio. Muchas funcionalidades no están visibles de manera inmediata y se encuentran ocultas detrás de menús y selectores que pueden no ser evidentes para todos los usuarios.

5.1.5 Pigeon Loft Manager

Figura 6: Logotipo de "Pigeon Loft Manager"

Autor: Jaithoon Venture

Fecha de publicación: 2 de marzo de 2021.

Última Actualización: 23 de octubre de 2023.

Número de descargas en Google Play: más de 10.000.

URL de Google Play:
https://play.google.com/store/apps/details?id=com.sstech.loftmanager&hl=es&gl=US

"*Pigeon Loft Manager*" es una aplicación para Android que permite llevar un registro detallado de todas las palomas de un colombófilo. Además, facilita la visualización en forma de árbol genealógico de cada ave y proporciona herramientas para gestionar los emparejamientos.

Una de las ventajas destacables de esta aplicación es su listado visual de palomas, presentado con una imagen y el número de anilla, lo cual resulta muy cómodo para la visualización en dispositivos móviles.

Otro aspecto positivo es que permite registrar competiciones y entrenamientos de manera sencilla e intuitiva. El registro de competiciones incluye detalles como la distancia, localización, tiempo de salida y llegada, y velocidad. Sin embargo, sería beneficioso añadir la posición obtenida por el animal dentro de la competición, un dato crucial en mi opinión.

La aplicación también ofrece secciones dedicadas al seguimiento de competiciones realizadas por clubes de colombofilia. Aunque es importante señalar que entre los clubes registrados en la aplicación no se incluyen clubes europeos.

A pesar de ser una herramienta útil para la gestión de palomares, algunas secciones de la aplicación no funcionan con fluidez o han dejado de funcionar, especialmente en las funcionalidades que requieren comunicación con servidores externos.

Capítulo 6: Tecnologías y Herramientas

En este capítulo, se describirán las diferentes tecnologías, herramientas y servicios que se han utilizado para el desarrollo de este proyecto.

6.1 Lenguajes

6.1.1 HTML

HTML cuyas siglas hacen referencia a "*HyperText Markup Language*", que en castellano significa "*Lenguaje de Marcado de Hipertexto*" es un lenguaje de marcado que se utiliza para estructurar el contenido de una página web [11]. HTML hace uso de etiquetas para indicar como van organizados los elementos que forman el contenido de la página web como, por ejemplo, encabezados, párrafos, imágenes, enlaces, botones, por nombrar alguno de los más utilizados [12]. También hace uso de atributos, estos son valores que complementan a las etiquetas y modifican su apariencia o configuran su comportamiento [13].

6.1.2 CSS y SASS

CSS cuyas siglas hacen referencia a "*Cascading Style Sheets*", que en castellano significa "*Hojas de Estilo en Cascada*" es un lenguaje de estilos que se utiliza para definir como tiene que visualizarse un documento estructurado como puede ser HTML o XML, entre otros [14] [15]. Su denominación de "en cascada" es porque cuando se aplica el código se hace de arriba abajo [16].

Una vez explicado qué es CSS, se podrá comprender qué es *Sass*. *Sass* cuyas siglas hacen referencia a "*Syntactically Awesome Stylesheets*" que en castellano significa "*Hojas de estilo sintácticamente increíbles*" es un preprocesador de CSS. La función de un preprocesador de CSS es tomar el código escrito, en este caso en Sass, y lo traduce a CSS [17].

Entre las ventajas del uso de Sass destacan el uso de variables, anidamiento y herencia. Lo que facilita el mantenimiento, la reusabilidad y la legibilidad del código.

Existen dos sintaxis de Sass. La original que usa una sintaxis con sangrado donde se utiliza la sangría y el carácter de '*nueva línea*' para separar las reglas. Y la sintaxis "*Sassy CSS*" que utiliza las llaves y el punto y coma para separar las reglas [18] como se hace en CSS.

Para la realización de este trabajo que ha usado la sintaxis original.

6.1.3 TypeScript

Para entender que es *TypeScript*, primero se explicará que es *JavaScript*. *JavaScript* es un lenguaje de programación interpretado y ligero que puede ejecutarse en los navegadores web [19].

TypeScript es un superconjunto de JavaScript, desarrollado y mantenido por *Microsoft* como un proyecto de código abierto. Este lenguaje añade nuevas características al lenguaje *JavaScript*, como el tipado estático, que puede ser opcional, y nuevos conceptos de Programación Orientada a Objetos que no tenía incorporado *JavaScript*. *TypeScript* utiliza un transpilador que traduce el código escrito en *TypeScript* en *JavaScript*. Una característica importante de *TypeScript* es que, al ser una extensión de *JavaScript*, el código escrito en *JavaScript* también es válido en *TypeScript* [20].

6.2 Frameworks y Librerías

6.2.1 Node.js

"*Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación JavaScript, asíncrono, con E/S de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google. Fue creado con el enfoque de ser útil en la creación de programas de red altamente escalables, como por ejemplo, servidores web."* [21] Node.js permite usar JavaScript en el lado del servidor [22].

6.2.2 Node Package Manager (npm)

Gestor de paquetes de Node.js que permite descargar, instalar y administrar paquetes de software libre desde un repositorio [24]. Estos paquetes permiten añadir nuevas funcionalidades a los proyectos desarrollados con Node.js. La forma más común de interactuar con el repositorio es a través de una interfaz de línea de comandos (CLI). NPM se instala automáticamente al realizar la instalación de Node.js.

6.2.3 Angular

Angular es un *framework* que permite diseñar aplicaciones web usando TypeScript. Este *framework* es de código abierto y fue desarrollado y está mantenido por Google. Con Angular se pueden crear aplicaciones web de una sola página (*Single Page Applications*) [25] donde la navegación en lugar de actualizar todo el contenido de la página sólo actualiza aquellas secciones que sean necesarias.

Algunas de las características más importantes de Angular son:

- **Módulos**: Las aplicaciones desarrolladas con angular son modulares. Un módulo se podría definir como "un conjunto de código dedicado a un ámbito concreto de la aplicación, o a una funcionalidad específica" [26] Un ejemplo de módulo es "*Angular Material*" que se explicará más adelante.
- **Componentes**: Un componente es una clase que se encarga de controlar una parte de la vista de la aplicación. Esta clase incluye todos los métodos y funciones necesarias para manejar los datos que se requieran en la vista. Un componente está asociado a

una plantilla creada con HTML y CSS que indica cómo debe mostrarse ese componente [27]

- **Servicios**: Es una clase que tiene una función específica. Sus funciones pueden ser muy variadas, pero normalmente se utilizan para proveer datos o la comunicación entre componentes. Los servicios serán utilizados por los componentes.
- **Angular CLI**: Herramienta de línea de comandos que automatiza muchas de las tareas del desarrollo como, por ejemplo, la creación de Componentes y Servicios, por nombrar algunas de las funciones más usadas [28].

El diseño modular de Angular permite crear aplicaciones escalables con, al mismo tiempo que simplifica su mantenimiento.

6.2.4 Angular Material

Angular Material es una librería de componentes comunes de interfaz de usuario, como, por ejemplo, botones, formularios, iconos, tablas, selectores de fecha, etcétera. La librería está basada en el lenguaje de diseño "*Material Design*" desarrollado por Google y se integra perfectamente con Angular. El uso de *Angular Material* facilita enormemente el diseño de las aplicaciones ya que proporciona muchos componentes visuales y estilos predefinidos [29].

6.2.5 Firebase

"*Firebase* es una plataforma para el desarrollo de aplicaciones web y aplicaciones móviles" [30] que actualmente pertenece a Google. Esta plataforma ofrece las herramientas y la infraestructura necesaria para desarrollar y alojar aplicaciones. Entre los servicios más destacables que ofrece se encuentran [31]:

- **Cloud Firestore**: Es un base de datos NoSQL que permite almacenar los datos que sean necesarios para el funcionamiento de las aplicaciones.
- **Firebase Authentication**: servicio de autentificación de usuarios. Proporciona diferentes posibilidades de autenticación como correo y contraseña, o usando proveedores de inicio de sesión de terceros como Google, Facebook, Microsoft, …
- **Cloud Storage**: Servicio para el almacenamiento de contenido como fotos y videos.
- **Firebase Hosting**: Permite alojar una página web o una aplicación para dispositivos móviles.
- **Realtime Database**: es una base de Datos NoSQL como "*Cloud Firestore*", pero con la diferencia de que los datos son guardados en formato JSON.
- **Google Analytics**: Crea informes sobre el comportamiento que tienen los usuarios en las aplicaciones donde se configure este servicio.

Existen librerías que facilitan la comunicación de la aplicación desarrollada con Angular y *Firebase*. En el caso de Angular la librería se llama *AngularFire*.

6.2.6 RxJS

R*xJS* (cuyas siglas hacen referencia a *"Reactive Extensions for JavaScript") "es una librería para crear programas asíncronos y basados en eventos mediante el uso de secuencias observables."* [32]. Con el uso de esta librería, se facilita enormemente el desarrollo de componentes reactivos, ya que proporciona una gran cantidad de herramientas para trabajar con flujos de datos asíncronos como las llamadas que se realizan a *Firebase* para recuperar información.

6.3 Herramientas

6.3.1 Git

Es un software que se utiliza para el control de versiones gratuito y de código abierto. [33] Git "registra los cambios realizados en un archivo o conjunto de archivos a lo largo del tiempo, de modo que se pueden recuperar versiones específicas más adelante" [34]. Esta característica permite realizar un seguimiento de las modificaciones realizadas y gestionar las diferentes etapas por las que discurre un proyecto.

6.3.2 GitHub

Es una plataforma online que permite a sus usuarios alojar proyectos utilizando Git. Desde esta plataforma se podrá tener un repositorio desde donde controlar las versiones de los proyectos, compartir los proyectos con otras personas, trabajar en equipo y tener un portafolio de todos los proyectos del usuario [35].

6.2.3 Visual Studio Code

Es un editor de código fuente desarrollado por Microsoft que "*Incluye compatibilidad integrada con JavaScript, TypeScript y Node.js, y cuenta con un amplio ecosistema de extensiones para otros lenguajes (como C++, C#, Java, Python, Go, .NET)*" [36]. Es de código abierto y multiplataforma. Visual Studio Code incluye el resaltado de sintaxis y autocompletado inteligente lo que facilita la lectura y escritura del código. Además, se puede personalizar y añadirle funcionalidades mediante la instalación de extensiones disponibles en su repositorio [37].

6.2.4 Trello

Trello es una aplicación que ayuda en la gestión de proyectos de manera muy visual mediante el uso de un tablero virtual. En los tableros se muestran diferentes listas que constituyen las distintas fases por las que puede discurrir una tarea. Para representar a las tareas se hace uso de tarjetas que incluyen la descripción del trabajo necesario a realizar y se irán moviendo entre las listas dependiendo del estado en el que se encuentre la tarea. [38]

6.2.5 Figma

Es un editor de gráficos vectorial que se utiliza para la creación de prototipos. Con esta aplicación se puede diseñar de una manera visual la interfaz de usuario [39]. Se puede usar como una plataforma online por lo que no es necesario instalarlo, aunque dispone de versión de escritorio para los sistemas operativos más usados. Con *Figma* se puede diseñar las vistas de aplicaciones móviles y páginas.

6.2.6 Draw.io

Es una plataforma online, aunque también cuenta con una versión de escritorio, que sirve para crear diagramas, mapas mentales, organigramas, y otros tipos de recursos visuales. En este proyecto, se ha usado para crear los diagramas de casos de uso.

6.2.7 Inkscape

Es un editor de gráficos vectoriales de código abierto con el que se pueden crear todo tipo de ilustraciones. Su principal función es la de proporcionar una herramienta para la creación de gráficos en formato vectorial escalable (SVG) [40]. Se ha utilizado para la creación del logotipo de la aplicación.

6.2.8 Microsoft Word

Es un procesador de textos desarrollado por Microsoft, muy utilizado para la creación de documentos. En este caso se ha usado para la redacción de esta memoria.

6.2.9 Coolors.co

Es una aplicación online que proporciona diferentes opciones para crear paletas de colores. Además, permite comprobar cómo quedaría una paleta de colores en diferentes entornos como aplicaciones web, diseño de marcas, ilustraciones, entre otros.

Capítulo 7: Análisis

En este capítulo se ofrece una visión general de la aplicación, detallando las funcionalidades, roles de usuario y casos de uso más relevantes. Además, se incluyen los diagramas de casos de uso para comprender de una manera sencilla el alcance de la aplicación.

7.1 Descripción general

Censo Palomar es el nombre de la aplicación que se va a desarrollar. Esta aplicación permitirá a los colombófilos censar las palomas de su palomar y registrar las competiciones en las que participan. Además, incorporará la funcionalidad de mostrar el árbol genealógico de una paloma, permitiendo así a los colombófilos controlar la estirpe de cada ave. La aplicación también incluirá una sección de noticias y un espacio para que los usuarios publiquen anuncios para vender palomas o material relacionado con la colombofilia.

7.2 Roles de usuario

La aplicación contará con tres tipos de roles: usuario no registrado, usuario registrado y administrador. A continuación, se detallarán que funcionalidades deberá tener cada rol.

7.2.1 Usuario no registrado

- Puede ver la página de Inicio
- Puede ver los anuncios de venta de palomas y material colombófilo
- Puede ver las noticias publicadas
- Puede iniciar sesión
- Puede registrarse en la aplicación

7.2.2 Usuario Registrado

- Puede registrar una paloma
- Puede ver un listado de palomas que le pertenecen
- Puede editar y eliminar el registro de una paloma que le pertenezca
- Puede ver el árbol genealógico de palomas que le pertenecen (hasta los abuelos)
- Puede registrar la competición de una paloma
- Puede editar y eliminar la competición de una paloma
- Puede ver su información de usuario
- Puede editar su información de usuario
- Puede publicar un anuncio para la venta de palomas o material usado en colombofilia
- Puede editar y eliminar un anuncio que haya publicado
- Puede ver un listado con los anuncios que ha publicado

- Puede ver todos los anuncios que se hayan publicado en la aplicación
- Puede ver las noticias publicadas

7.2.3 Administrador

Además de todas las funcionalidades del usuario registrado podrá:

- Puede ver un listado con los usuarios registrados
- Puede publicar noticias
- Puede editar noticias
- Puede eliminar noticias

7.3 Casos de Uso

Los casos de uso son descripciones detalladas de cómo los usuarios interactúan con un sistema para realizar alguna funcionalidad. Son fundamentales en el desarrollo software porque ayudan a comprender qué debe hacer el sistema que se va a desarrollar [41]. Cada caso de uso detalla los pasos necesarios para lograr un objetivo exitoso. Además, en la especificación de un caso de uso se incluyen rutas alternativas, manejo de excepciones y condiciones que deben cumplirse antes y después de la ejecución. Representados en diagramas, los casos de uso muestran las relaciones entre los actores (usuarios o sistemas) y el sistema en sí. Su propósito es identificar, esclarecer y organizar los requisitos del sistema [42].

7.3.1 Diagramas de Casos de Uso

A continuación, se muestran los diagramas de casos de usos realizados para el desarrollo de este proyecto.

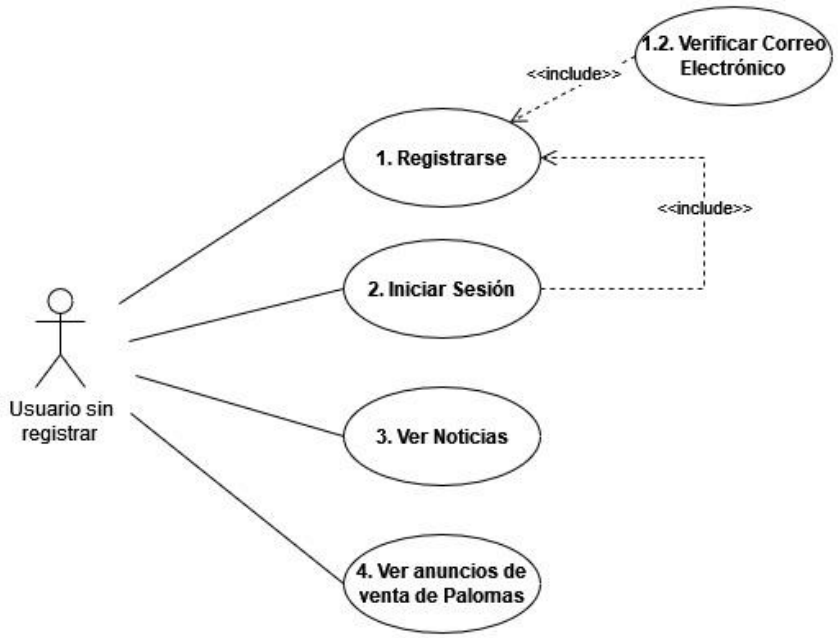

Figura 7: Diagrama de casos de uso de un usuario sin registrar

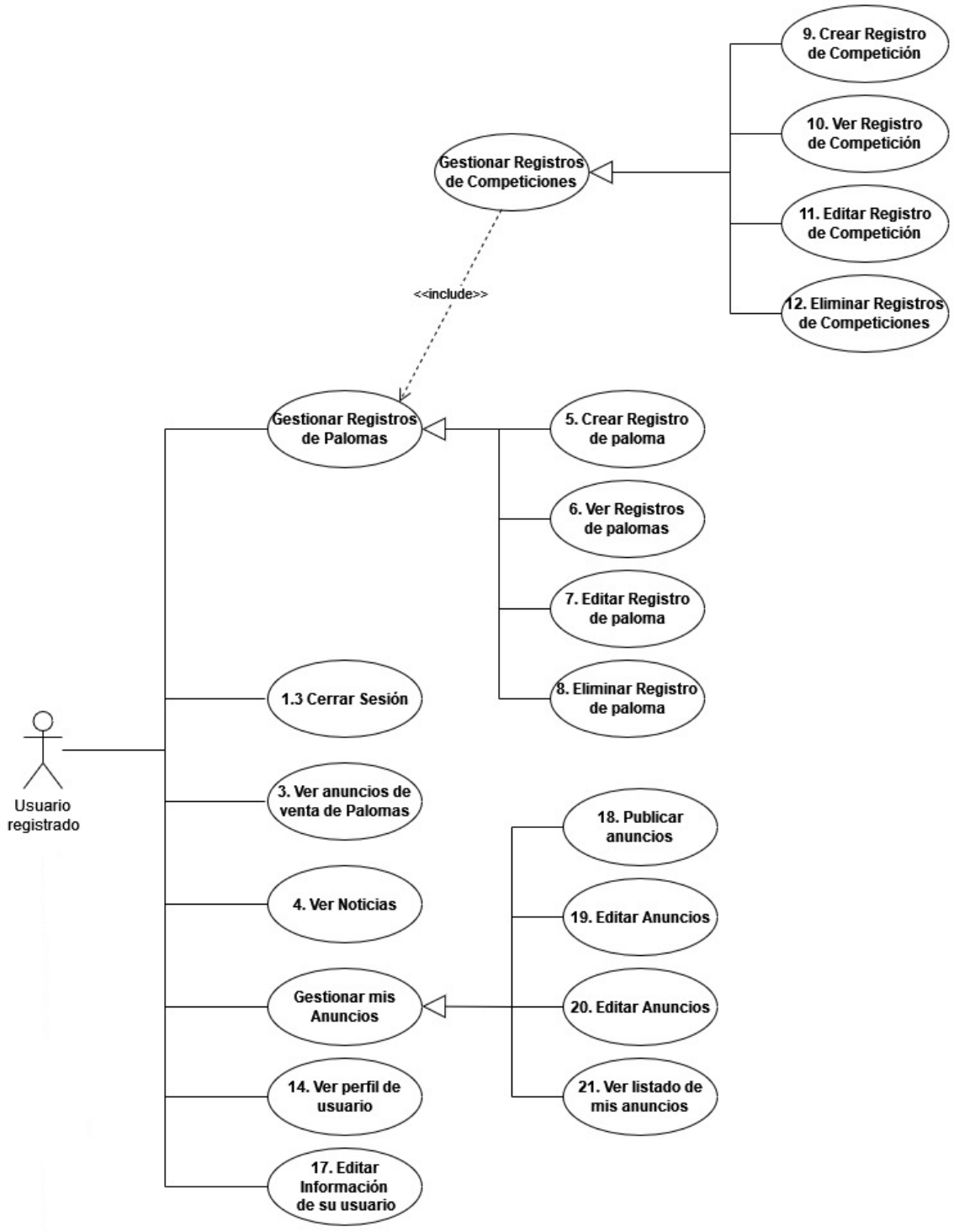

Figura 8: Diagrama de casos de uso de un usuario registrado

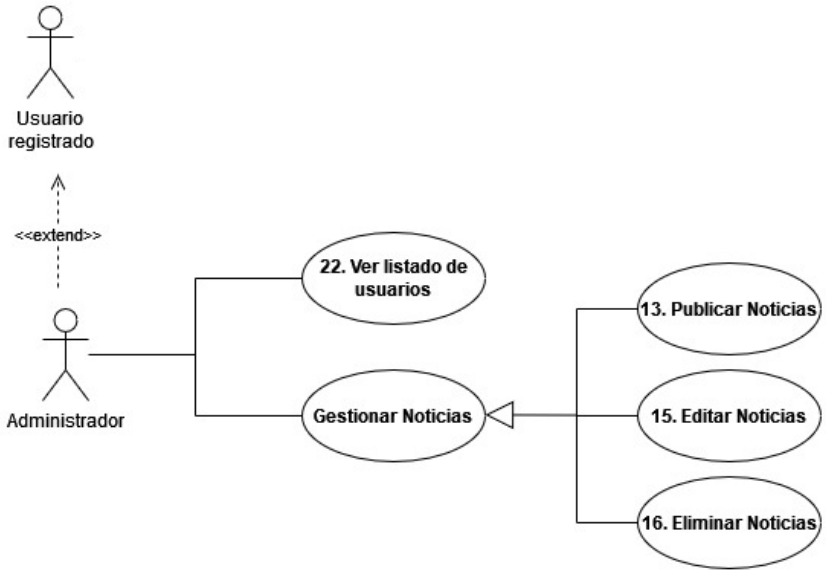

Figura 9: Diagrama de casos de uso de un Administrador

7.3.2 Especificaciones de los casos de uso

Las especificaciones de los casos de uso son muy útiles a la hora de desarrollar una aplicación. Estas especificaciones describen las acciones que debe realizar un usuario o sistema para alcanzar el objetivo de una funcionalidad, proporcionando al desarrollador una visión global de lo que debe lograr cada caso de uso.

A continuación, se muestran las especificaciones de los casos de usos más representativos de la aplicación.

Caso de Uso	**5**	**Crear registro de una paloma**
Descripción		El usuario quiere crear el registro de una paloma en la aplicación
Actores		Usuario Registrado
Precondiciones		El usuario ha iniciado sesión y se encuentra en la sección de registrar paloma. Además, debe haber conexión a internet
Flujo Normal		
Paso		Acción
1		El usuario rellena el formulario con los datos de la paloma
2		Se valida que los campos requeridos estén rellenos y con el formato correcto
3		El usuario selecciona el botón "Registrar paloma"
4		El sistema guarda la información de la paloma en la base de datos
5		Se muestra un mensaje indicando el éxito de la operación
Postcondiciones		El usuario es redirigido al listado de palomas y verá en primer lugar la paloma que acaba de registrar
Variaciones		

Los campos requeridos no han sido cumplimentados	
1	El sistema muestra un mensaje indicando el error
2	El usuario corrige los campos erróneos
3	Selecciona el botón "Registrar paloma"
4	El sistema guarda la información de la paloma en la base de datos
Excepciones	
No se puede establecer conexión con la base de datos	
1	El sistema muestra un mensaje de error indicando la situación

Tabla 2: Especificación del caso de uso "Crear Registro de una paloma"

Caso de Uso	**01**	**Iniciar Sesión**
Descripción		El usuario quiere acceder a la parte privada
Actores		Usuario sin Registrar
Precondiciones		El usuario no tiene la sesión iniciada y se encuentra en la página de Iniciar Sesión. El usuario ya tiene que estar registrado en el sistema. Debe haber conexión a internet
Flujo Normal		
Paso		Acción
1		El usuario introduce su correo electrónico y su contraseña
2		Si el correo está bien formado se activa el botón de "Iniciar Sesión"
3		El usuario selecciona el botón iniciar sesión o presiona la tecla "*enter*"
4		El sistema valida que el correo y la contraseña son válidos y están registrados en el sistema
Postcondiciones		El usuario tiene la sesión iniciada y es redirigido al listado de sus palomas
Variaciones		
Pasos		Acción
Credenciales incorrectas / Correo no existente		
1		El sistema muestra un mensaje indicando el error
2		El usuario corrige los datos
3		Si el correo está bien formado se activa el botón de "Iniciar Sesión"
4		El usuario selecciona el botón iniciar sesión o presiona la tecla "*enter*"
5		El sistema valida que el correo y la contraseña son válidos y están registrados en el sistema
Excepciones		

Tabla 3: Especificación del caso de uso "Iniciar Sesión"

Caso de Uso	2	Registro de usuario
Descripción		El usuario quiere crear una cuenta en la aplicación
Actores		Usuario sin Registrar
Precondiciones		El usuario no tiene la sesión iniciada y se encuentra en la página de registro. Debe haber conexión a internet
Flujo Normal		
Paso		Acción
1		El usuario rellena el formulario de registro con sus datos
2		El usuario selecciona el botón "*Regístrate*" o presiona la tecla "*enter*"
3		El sistema registra al usuario y guarda sus datos en la base de datos
4		El sistema le envía correo de verificación (Caso de uso 1.2)
5		Se muestra un mensaje indicando el éxito de la operación
Postcondiciones		El usuario ha creado su cuenta, tiene la sesión iniciada y es redirigido a la sección del censo de paloma.
Variaciones		
		Datos incorrectos en el formulario
1		El sistema muestra un mensaje de error indicando qué campo está mal
2		El usuario corrige los datos
3		El usuario selecciona el botón "*Regístrate*" o presiona la tecla "*enter*"
4		El sistema registra al usuario y guarda sus datos en la base de datos
5		El sistema le envía correo de verificación (Caso de uso 1.2)
Excepciones		
El correo ya está registrado		
1		El sistema muestra un mensaje indicando el error y que no puede usar ese correo

Tabla 4: Especificación del caso de uso "Registro de usuario"

Caso de Uso	3	Ver noticias
Descripción		Cualquier tipo de usuario quiere ver las noticias publicadas
Actores		Usuario no registrado, Usuario Registrado y Administrador
Precondiciones		Debe haber conexión a internet
Flujo Normal		
Paso		Acción
1		El usuario despliega el menú lateral
2		Selecciona "Noticias"
3		Se muestra un listado con las noticias publicadas
4		Selecciona una noticia
Postcondiciones		El usuario puede leer el contenido completo de la noticia
Variaciones		
1a		El usuario navega por la página de inicio hasta la sección *últimas noticias*
2a		Selecciona una de las noticias
1b		En la sección de noticias de la página de inicio selecciona el botón "Ver más noticias"
Excepciones		

Tabla 5: Especificación del caso de uso "Ver noticias"

Caso de Uso	6	Ver registro de palomas
Descripción		El usuario quiere ver la información de una paloma
Actores		Usuario Registrado
Precondiciones		El usuario ha iniciado sesión, se encuentra en la sección del listado de palomas. Además, debe haber registrado al menos una paloma. Debe haber conexión a internet
Flujo Normal		
Paso		Acción
1		El usuario selecciona la paloma que desea ver
Postcondiciones		El usuario verá la información completa de la paloma
Variaciones		
Excepciones		
No se puede establecer conexión con la base de datos		
1		El sistema muestra un mensaje de error indicando la situación

Tabla 6: Especificación del caso de uso "Ver registro de paloma"

Caso de Uso	11	Editar Registro de Competición
Descripción		El usuario quiere eliminar el registro de una paloma
Actores		Usuario Registrado
Precondiciones		El usuario ha iniciado sesión y se encuentra en el desglose de la información de una competición. Debe haber conexión a internet El usuario debe ser el mismo que creó el registro de la competición
Flujo Normal		
Paso		Acción
1		El usuario selecciona el botón "Editar"
2		El usuario es redirigido al formulario de edición
3		El sistema cargará la información actual de la paloma en el formulario
4		El usuario modificará la información que considere
6		El usuario selecciona el botón "Editar Competición"
4		Se valida que los campos requeridos estén rellenos y con el formato correcto
5		El sistema calcula la duración y la velocidad
7		El sistema guarda la información actualizada de la competición en la base de datos
8		Se muestra un mensaje indicando el éxito de la operación
Postcondiciones		El usuario es redirigido al desglose la competición y verá los datos modificados
Variaciones		
Los campos requeridos no han sido cumplimentados o no tienen el formato correcto		
1		El sistema muestra un mensaje indicando el error
2		El usuario cumplimenta los campos erróneos
3		Selecciona el botón "Editar Competición"
4		El sistema guarda la información de la competición en la base de datos
Excepciones		
No se puede establecer conexión con la base de datos		
3a		El sistema muestra un mensaje de error indicando la situación

Tabla 7: Especificación del caso de uso "Editar registro de competición"

Caso de Uso	16	Eliminar noticia
Descripción		El Administrador quiere eliminar una noticia
Actores		Administrador
Precondiciones		El Administrador ha iniciado sesión y se encuentra en la viendo noticia. Además, debe haber conexión a internet
Flujo Normal		
Paso		Acción
1		El Administrador selecciona la opción *"Eliminar*"
2		Aparecerá una ventana de confirmación
3		El Administrador pulsa en *"Aceptar"*
4		La noticia y todos sus datos son eliminados de la base de datos
5		Se muestra un mensaje indicando el éxito de la operación
Postcondiciones		El Administrador es redirigido al listado de noticias y la noticia ha sido eliminada
Variaciones		
El administrador pulsa cancelar en la ventana de confirmación		
1		El Administrador selecciona el botón "Eliminar"
2		Aparecerá una ventana de confirmación pulsa en cancelar
3		El Administrador pulsa en *"Cancelar"*
4		Este caso de uso de cancela
Excepciones		
No se puede establecer conexión con la base de datos		
1		El sistema muestra un mensaje de error indicando la situación

Tabla 8: Especificación del caso de uso "Eliminar noticia"

7.3 Requisitos

Los requisitos son la especificación de las funcionalidades y características que tiene o va tener un producto software. Estos requisitos sirven como guía para definir cómo se espera que funcione y qué características o cualidades debe cumplir. Se divide principalmente en:

- **Requisitos funcionales**: aquellos requisitos que definen las funciones que se deben desarrollar en el producto software. [43]
- **Requisitos no funcionales**: aquellas características que definen las cualidades, características y restricciones que tendrá el producto software a desarrollar, como, por ejemplo, usabilidad, seguridad, rendimiento, portabilidad y muchos más aspectos. [44]

En resumen: *"Si los requisitos funcionales especifican lo que debe hacer un sistema, los requisitos no funcionales describen cómo lo hará"* [45]

A continuación, se detallan los requisitos funcionales y no funcionales que tendrá la aplicación a desarrollar.

7.3.1 Requisitos Funcionales

- El sistema permitirá mostrar y navegar por la página de inicio a cualquier tipo de usuario.
- El sistema permitirá mostrar las noticias a cualquier tipo de usuario.
- El sistema permitirá mostrar los anuncios a cualquier tipo de usuario.
- El sistema permitirá el registro de un usuario con correo y contraseña en la aplicación.
- El sistema enviará un correo para validar el correo del usuario durante el registro.
- El sistema permitirá el inicio de sesión en la aplicación con correo y contraseña como credenciales.
- El sistema permitirá cerrar sesión.
- El sistema dispondrá de un menú para que el usuario pueda navegar por las diferentes secciones de la aplicación.
- El sistema no permitirá navegar al usuario por las partes no públicas de aplicación hasta que el correo sea validado.
- El sistema permitirá a un usuario registrado ver su información de usuario.
- El sistema permitirá que el usuario registrado pueda editar sus datos de usuario, exceptuando el correo.
- El sistema permitirá crear el registro de palomas a usuarios registrados.
- El sistema permitirá crear, mostrar, editar y eliminar el registro de una paloma únicamente al usuario que creó el registro.
- El sistema permitirá el registro de competiciones de una paloma únicamente al usuario que registró la paloma.
- El sistema permitirá crear, mostrar, editar y eliminar el registro de una competición únicamente al usuario que creó el registro.
- El sistema permitirá publicar un anuncio a los usuarios registrados.
- El sistema permitirá a un usuario registrado listar sus propios anuncios.
- El sistema permitirá editar y eliminar un anuncio únicamente al usuario que publicó el anuncio.
- El sistema permitirá al administrador ver un listado de los usuarios registrados.
- El sistema permitirá a los usuarios registrados mostrar el árbol genealógico de una paloma, como mínimo hasta los abuelos del ave.
- El sistema permitirá publicar, editar y borrar noticias sólo al usuario Administrador.

7.3.2 Requisitos No funcionales

- El sistema deberá tener acceso a Internet.
- El sistema deberá mostrar mensajes indicando tanto éxito como error cuando se registra, edita o elimina una paloma, competición, anuncio o noticia.
- Los usuarios sólo podrán ver sus palomas y competiciones.
- El sistema debe notificar al usuario cuando los datos no se pueden recuperar.
- El sistema debe mostrar un mensaje de error si los datos para iniciar sesión no son correctos, sin especificar qué dato es erróneo
- El sistema debe ser compatible con diferentes resoluciones de pantalla, especialmente para los dispositivos móviles.
- El sistema debe tener disponible un manual de usuario.
- El sistema generará un identificador para cada usuario que se registre.

Capítulo 8: Diseño

En este capítulo se describirán las decisiones tomadas para realizar el diseño de la aplicación. El objetivo es presentar cómo se le dio forma a la aplicación desde su fase inicial, una parte muy significativa del desarrollo de una aplicación. A continuación, se detallarán los aspectos clave del diseño visual, la creación de bocetos y la estructura de la base de datos.

8.1 Diseño Visual

8.1.2 Paleta de colores

La elección de la paleta de colores que se va a usar en la aplicación se inspiró en los colores de las palomas. Aunque las palomas pueden lucir una gran variedad de colores como negro, marrón, blanco o combinaciones de ellos. Se ha optado por una de las tonalidades más características de las palomas, el gris azulado.

Para generar la paleta de colores se ha hecho uso de la aplicación web coolors.co. En este proyecto se ha usado la herramienta "*Image Picker*", disponible en la aplicación, que es capaz de generar una paleta de colores a partir de una imagen.

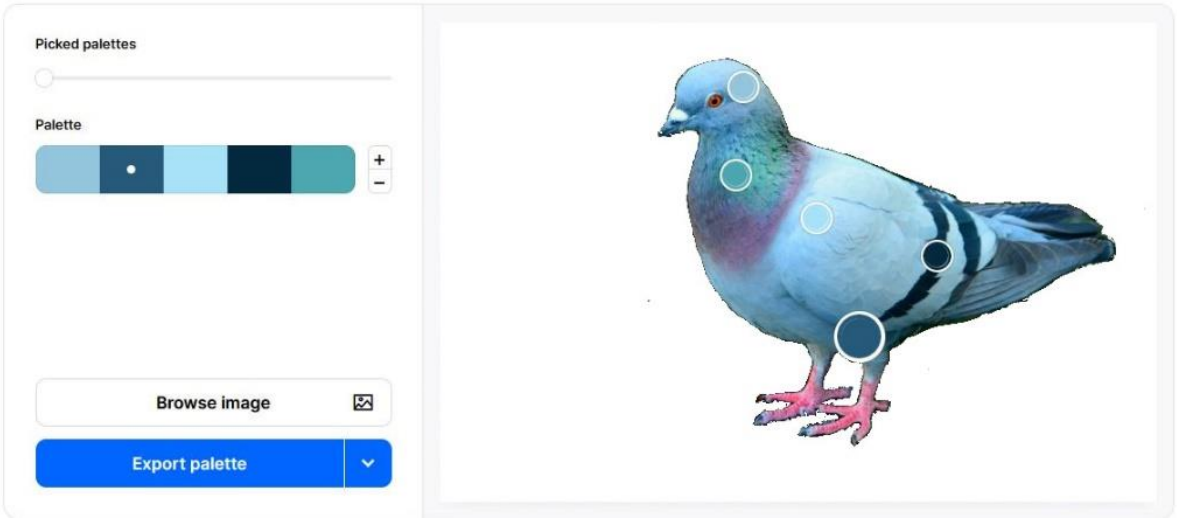

Figura 10: Imagen de una paloma para extraer la paleta de colores

La imagen utilizada, que se muestra en la figura 10, es la de una paloma con los tonos grises y azulados. El resultado de la paleta se puede ver en la figura 11.

Figura 11: Paleta de colores

Esta paleta de colores se ha utilizado para dar color a la aplicación y crear así una experiencia visual coherente para los usuarios.

8.1.2 Mockups

Uno de los primeros pasos en el desarrollo de cualquier proyecto es bocetar cómo se desea que se vea el producto final. En esta sección se mostrarán los bocetos realizados de la interfaz de usuario para proporcionar una visión clara del producto final que se quiere alcanzar. El diseño de los bocetos se ha realizado con la aplicación *Figma*, una herramienta que, como se mencionó en el apartado de *Tecnologías y Librerías,* ayuda a los diseñadores de aplicaciones a crear bocetos y prototipos. A continuación, se muestran alguno de los bocetos realizados de las secciones más importantes.

Figura 12: Boceto del listado de palomas

Figura 13: Boceto del detalle de una paloma

Figura 14: Boceto del detalle de una competición

En la figura 12 se muestran los bocetos realizados para listar las palomas registradas por el usuario. Cada fila contendrá la imagen de la paloma y la información más relevante para identificarla. Al hacer clic en una fila, se podrá ver la información detallada de la paloma, como se muestra en la figura 13. Aquí, el usuario podrá ver la información completa de la paloma, incluidas las competiciones en las que ha participado. Al hacer clic en una competición, se podrá ver la información detallada del resultado de la competición, visualizada como en la figura 14.

Figura 15: Boceto de la página de inicio

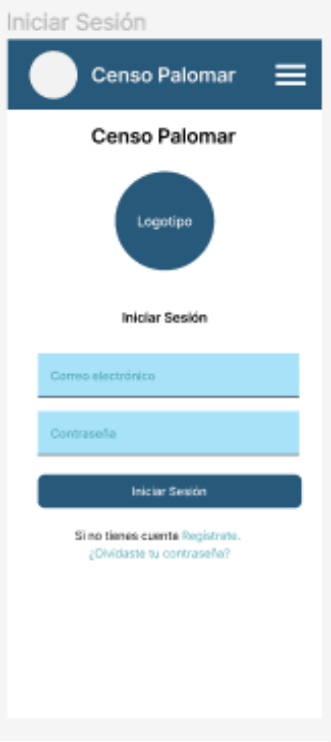

Figura 16: Boceto de la página de inicio de sesión

Figura 17: Boceto del árbol Genealógico

En la figura 15 se observa el diseño realizado para la página de inicio de la aplicación, la cual será visible independientemente de si el usuario está registrado o no. En la figura 16 se muestra la vista diseñada para el formulario de inicio de sesión. Por último, en la figura 17 se presenta el boceto inicial realizado para la vista del árbol genealógico de una paloma en un dispositivo móvil.

8.1.3 Logotipo

Para el diseño del logotipo, se utilizó el programa *Inkscape*. En la figura 18 se puede ver el proceso de creación del mismo.

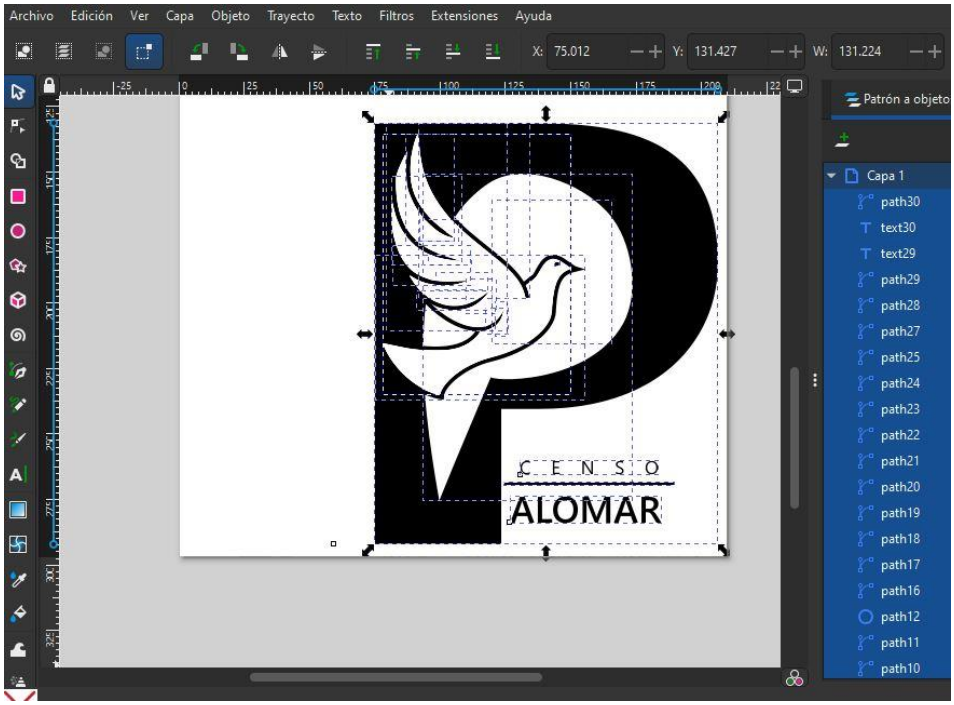

Figura 18: Creación del logotipo

Dado que la aplicación está destinada a las palomas, era fundamental que el logotipo incluyera este animal. Para crearlo, se seleccionó una ilustración de una paloma, de la que se extrajeron las líneas básicas para crear su contorno. Luego, se incrustó ese dibujo en una letra "P" mayúscula. Para evitar un diseño demasiado oscuro, se amplió el hueco de la "P" creando una forma que se asemeja a la letra, pero sin llegar a formarla completamente.

En la parte inferior derecha, se colocó el nombre de la aplicación, "Censo Palomar", en dos renglones separados por una línea horizontal en el medio, usando la letra "P" con el dibujo de la paloma como la primera letra de la segunda palabra "Palomar". Es un logotipo bastante sencillo y clásico, pero que permite identificar perfectamente que la aplicación está destinada a las palomas.

Este logotipo estará presente en varias partes de la aplicación: la cabecera, el pie de página y como imagen por defecto en los campos donde el usuario deba subir una imagen pero haya dejado ese campo en blanco. En la figura 19 se puede ver el logotipo terminado en diferentes colores:

Figura 19: Logotipo con diferentes colores para probar su contraste

8.2 Diseño de la base de datos

Para que la aplicación funcione correctamente, es necesario almacenar los datos para utilizarlos cuando sea necesario. Como se ha comentado anteriormente, se ha utilizado la plataforma *Firebase* y el servicio *Cloud Firestore* como base de datos para este proyecto.

Cloud Firestore es un sistema de base de datos no relacionales que almacena la información en forma de colecciones en lugar de en tablas, como lo hacen las bases de datos relacionales. Cada colección se compone de documentos, funcionando de manera similar a un sistema de carpetas. Cada documento contendrá la información de cada registro que queramos almacenar. Un detalle importante a mencionar es que cada colección puede, a su vez, tener subcolecciones. Esta característica se ha utilizado para almacenar el censo de cada colombófilo en su respectiva colección.

En la figura 20 se muestra el diagrama de la base de datos, donde se muestra la relación que existe entre las diferentes colecciones.

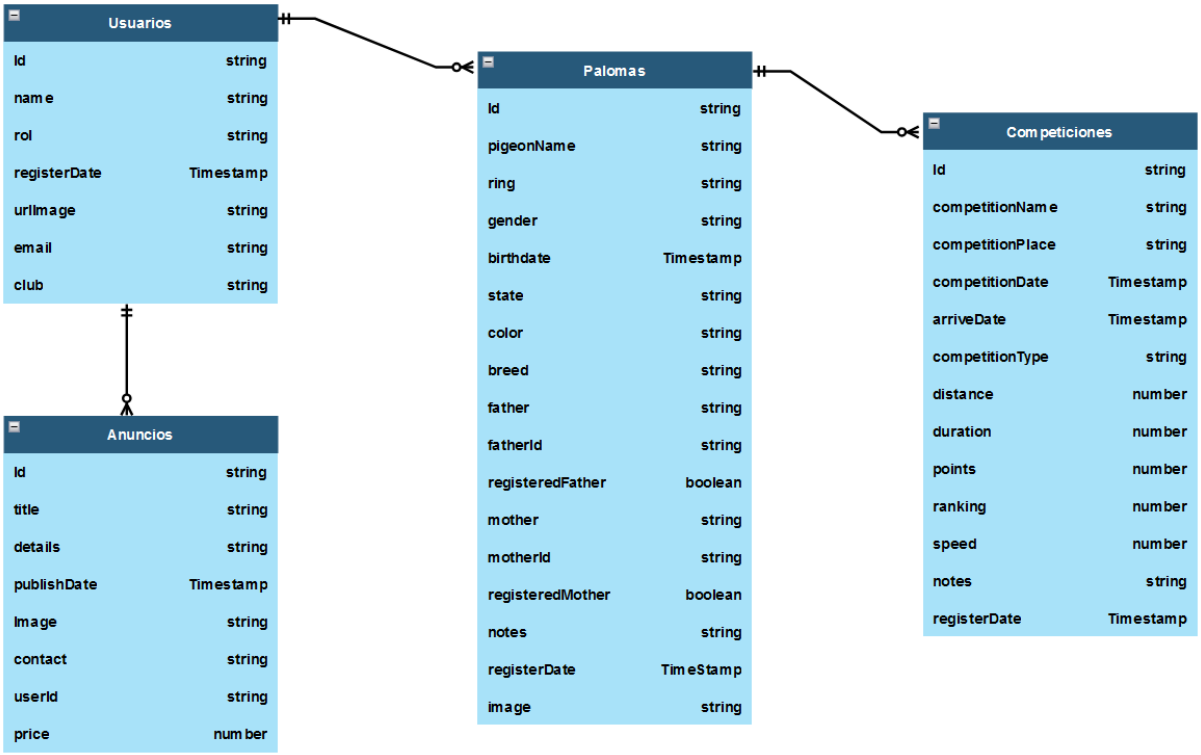

Figura 20: Diagrama de la base de datos

En los siguientes apartados se explica en detalle cada una de las colecciones creadas para este proyecto.

8.2.1 Competiciones

Esta colección de datos contiene los documentos con la información de la competición en los que ha participado una paloma, y contendrá los siguientes campos.

- **Clasificación:** puesto en el que quedó la paloma. Se almacena como un número.
- **Nombre de la competición:** nombre con el que se identifica la competición o prueba que ha realizado la paloma. Se almacena como una cadena de texto.
- **Punto de suelta:** lugar desde donde se comienza la competición. Se almacena como una cadena de texto.
- **Modalidad:** tipo de competición. Normalmente las sueltas se clasifican en alguna de estas modalidades: Velocidad, Velocidad Jóvenes, Medio Fondo, Fondo y Gran Fondo. Se almacena como una cadena de texto.
- **Fecha de la Competición:** fecha y hora en la que comienza la competición. Es el momento en el que se abren las jaulas y las palomas empiezan a volar. Se almacena como una marca de tiempo.
- **Fecha y hora de llegada:** hora en la que la paloma regresa a su palomar. Se almacena como una marca de tiempo.
- **Distancia:** distancia entre el punto de suelta y el palomar al que tiene que regresar la paloma. Este valor se suele expresar en metros. Se almacena como un valor numérico.
- **Duración:** Duración en minutos de lo que ha tardado la paloma en regresar al palomar. Se almacena como un número.
- **Velocidad:** velocidad alcanzada por la paloma en metros/minuto se calcula realizando la división entre la distancia y la diferencia de la *Hora de llegada* con la *Hora de la suelta* en minutos. Se almacena como un valor numérico.
- **Puntos:** puntuación obtenida en esa competición. Se almacena como un valor numérico.
- **Fecha de registro:** fecha y hora en la que el usuario registró la competición. Se almacena como una marca de tiempo.
- **Identificador:** valor alfanumérico único que identifica a ese documento. Se construye con la fecha de registro y el título de la competición.
- **Observaciones:** notas, comentarios u otra información que no se recoja en los campos anteriores. Se almacena como un campo de texto.

8.2.2 Palomas

Esta colección contiene toda la información relevante para poder identificar una paloma. Se compone de los siguientes campos:

- **Nombre:** cómo se identifica la paloma. Se almacena como una cadena de texto.
- **Anilla:** código alfanumérico que sirve para identificar a la paloma, es como si fuera su DNI. Es proporcionada por las diferentes federaciones y clubes colombófilos. Este código va impreso en una anilla que se le coloca a la paloma a los pocos días de nacer. Suele estar compuesto por el código del país o código de la federación seguida de

varios números y 2 o 4 dígitos indicando el año de nacimiento. Suele variar un poco entre países, pero siguen aproximadamente esa estructura. Se almacena como una cadena de texto.

- **Sexo:** determinar si el animal es macho o hembra. Se almacena como una cadena de texto.
- **Fecha de Nacimiento:** fecha en la que nació la paloma. Se almacena como una marca temporal.
- **Estado:** estado en el que se encuentra la paloma. Generalmente será alguno de estos "*Viva*", "*Perdida*", "*Muerta*", "*Vendida*", "*Enferma*" o "*Prestada*".
- **Color:** campo que se destinará a determinar el color del plumaje de la paloma. Se almacena como un campo de texto.
- **Estirpe:** este campo está destinado a guardar información sobre la raza, o línea sanguínea de la paloma. Se almacenará como un campo de texto.
- **Notas:** campo destinado a almacenar otra información de interés que no se esté englobada dentro de los otros campos. Se almacena como un campo de texto.
- **Padre**: campo para determinar al padre de la paloma. En este campo irá el nombre del padre. Se almacena como un campo de texto.
- **Identificador del padre:** campo para almacenar el identificador del padre, en el caso de que el padre esté registrado. Se almacena como una cadena de texto.
- **Tiene al padre registrado:** este campo se usa para determinar si el padre ya está registrado en la aplicación o si se ha introducido el nombre manualmente. Se almacenará como un campo booleano.
- **Madre:** campo para determinar a la madre de la paloma. Funciona de manera similar que el campo de *Padre*.
- **Identificador de la madre:** campo para almacenar el identificador de la madre, en el caso de que la madre esté registrada. Se almacena como una cadena de texto.
- **Tiene a la madre registrada:** Este campo es similar al campo "*Tiene al padre registrado*" pero en este caso para la madre de la paloma. Se almacenará como un campo booleano.
- **Competiciones:** colección de competiciones en las que ha participado la paloma. Como se explicó en el punto anterior.
- **Imagen:** URL con una imagen de la paloma para facilitar su identificación. Se almacena como campo de texto.
- **Fecha de registro:** fecha en la que el usuario creo el registro de la paloma. Se almacena como una marca de tiempo.
- **Identificador de la paloma:** valor alfanumérico único que identifica a ese usuario dentro de la base de datos. Se construye con la fecha de registro seguida del número de anilla. Se almacena como una cadena de texto.

8.2.3 Usuarios

Esta colección está destinada a almacenar datos del usuario como:

- **Nombre:** nombre del usuario. Se almacena como un campo de texto.
- **Correo:** correo electrónico del usuario. Se almacena como un campo de texto.
- **Imagen de perfil:** URL con una imagen que el usuario podrá ver en su perfil. En caso de que no suba ninguna se establecerá una genérica. Se almacena como campo de texto.
- **Club:** campo destinado para que el usuario ponga el nombre del club colombófilo al que pertenece. Este campo será opcional y se almacenará como un campo de texto.
- **Fecha de registro:** fecha en la que el usuario se creó la cuenta. Se almacena como una marca de tiempo.
- **Rol:** almacena cuál es el rol de usuario en la aplicación. Se puede diferenciar entre el rol de Administrador y el de colombófilo.
- **Palomas:** colección de las palomas que el colombófilo tiene en su palomar. Como se explicó en el punto anterior.
- **Identificador del usuario:** valor alfanumérico único que identifica a ese usuario dentro de la base de datos. Se almacena como un campo de texto.

8.2.4 Noticias

Esta colección está destinada a almacenar las noticias que se muestran en la aplicación, se compone de los siguientes campos:

- **Título:** titular de la noticia. Se almacena como un campo de texto.
- **Fecha de publicación:** fecha y hora en la que se creó la noticia. Se almacena como una marca de tiempo.
- **Cuerpo de la noticia:** contenido principal de la noticia. Se almacena como un campo de texto.
- **Categoría:** término con la que se clasificará la noticia. Se almacena como un campo de texto.
- **Imagen destacada:** URL con una imagen que ilustre a la noticia. Se almacena como campo de texto. Se almacena como un campo de texto.
- **Identificador de la publicación:** valor alfanumérico único que identifica a esa noticia dentro de la base de datos. Se almacena como un campo de texto. Está formada por la fecha concatenada con el título.

8.2.5 Anuncios de Ventas

Esta colección está destinada a almacenar los anuncios de venta de palomas que pueden publicar los usuarios. Se compone de los siguientes campos:

- **Título:** descripción breve del anuncio. Se almacenará como un campo de texto.
- **Fecha de publicación:** fecha y hora en la que se creó el anuncio. Se almacena como una marca de tiempo.

- **Descripción detallada:** toda la información relevante que el usuario considere para informar a los posibles compradores. Se almacena como un campo de texto.
- **Precio:** cantidad de dinero que el usuario determina que vale su paloma. Se almacena cómo un número.
- **Modo de contacto:** dato para que los posibles compradores puedan contactar con el vendedor. Se almacena como campo de texto.
- **Imagen destacada:** URL con una imagen de la paloma a vender. Se almacena como campo de texto.
- **Identificador del usuario:** valor alfanumérico único que identifica a ese usuario dentro de la base de datos.

8.2.6 Imágenes

Para el almacenamiento de las imágenes se ha utilizado el servicio "*Google Storage*". Este servicio permite almacenar, mantener y recuperar datos. En su versión gratuita, tiene una capacidad de 5GB, por lo que para realizar este prototipo de aplicación es suficiente. Cada vez que se suba una imagen, se obtendrá su URL y se almacenará en su campo correspondiente. A la hora de mostrar la imagen en la aplicación, obtendremos la imagen desde esa URL.

Se crearán tres carpetas diferenciadas por su uso:

- **Palomas:** esta carpeta contendrá todas las imágenes que son utilizadas en los registros de las palomas. Dentro de esta carpeta se crearán otras cuyo nombre será el correo electrónico del usuario, para separar las imágenes de cada usuario.
- **Usuarios:** aquí se almacenarán todas las imágenes que suban los usuarios para colocarla en sus perfiles.
- **Noticias:** carpeta destinada a guardar todas las imágenes utilizadas para ilustrar las noticias en la aplicación.
- **Anuncios:** carpeta destinada a guardar todas las imágenes utilizadas en los anuncios de venta. Al igual que con las palomas, las imágenes se separarán en carpetas diferentes para cada usuario.

Capítulo 9 – Desarrollo

En este capítulo, se detalla la estructura de directorios del proyecto. Además, se explican los diversos servicios y componentes desarrollados para cumplir con los objetivos establecidos. Finalmente, se describe la fase de despliegue de la aplicación en Firebase.

9.1 Estructura del Proyecto

Al ser un proyecto desarrollado en *Angular*, ya viene por defecto una estructura de directorios predefinida que ayuda enormemente a la organización y desarrollo del código. A continuación, se nombrarán los aspectos más relevantes de esta estructura de ficheros [46]:

- **`/node_modules`**: este directorio contiene todas librerías y dependencias descargadas a través de NPM.
- **`/src`**: directorio que contiene el código desarrollado en el proyecto.
  - **`/src/app`**: directorio que almacena los componentes, servicios y otros elementos implementados en el proyecto.
    - **`/app.config.ts:`** fichero que contiene toda la configuración del proyecto.
    - **`/app.component.ts:`** Este fichero junto con su correspondiente **`app.component.html`** y también **`app.component.sass`** forman el componente principal de la aplicación.
  - **`/src/assets`**: directorio que almacena el contenido estático del proyecto, como imágenes, fuentes y demás recursos necesarios.
  - **`/scr/environments`**: directorio para almacenar las variables de entorno. Como se muestra en el Anexo I es donde se almacena la configuración de *Firebase*.
  - **`/main.ts`**: punto de entrada de la aplicación, este fichero indica cuál es el componente principal con el que debe arrancar la aplicación.
  - **`/Styles.css`**: fichero con los estilos globales del proyecto.
  - **`/index.html`**: fichero principal de HTML, a partir de este fichero se construye toda la aplicación.
- **`/package.json`**: fichero de configuración que contiene la información sobre las dependencias instaladas.
- **`/angular.json`**: fichero que contiene la configuración de *Angular*.
- **`/tsconfig.json`**: fichero que contiene la configuración de *TypeScript*.

Una vez definida la estructura general de un proyecto Angular, comentaremos la estructura que se ha realizado para este proyecto. Dentro de la carpeta **`/src/app`**, se han dividido los directorios de la siguiente manera:

- **`/general-styles`**: directorio destinado a almacenar los estilos generales usados en la aplicación.
- **`/guards`**: en este directorio se encuentran los *guards*, que son servicios utilizados para controlar el acceso a las rutas de la aplicación.

- **`/models`**: directorio destinado a almacenar las diferentes interfaces que se utilizan en la aplicación.
- **`/pages`**: en este directorio se almacenan los componentes que forman las diferentes páginas que tiene la aplicación.
- **`/services`**: directorio destinado a almacenar los servicios generales de la aplicación.
- **`/shared`**: en este directorio se encuentran las clases comunes que se pueden reutilizar en la aplicación, como el menú, el pie de página o los cuadros de diálogo.

Dentro del directorio de ***`/pages`*** se ha estructurado la aplicación según funcionalidades. La organización de directorios quedaría de la siguiente forma:

- **`/admin`**: en este directorio se encuentran los componentes y servicios de las funcionalidades que puede realizar el rol administrador, como el listado de usuarios.
- **`/ads`**: en este directorio se encuentran todos los componentes relacionados con la publicación, edición, visualización y eliminación de los anuncios.
- **`/auth`**: directorio destinado a todos los componentes utilizados para la autenticación del usuario, como el inicio de sesión, el registro y la verificación del correo electrónico,
- **`/home`**: contiene el componente que se encarga de la página de inicio de la aplicación.
- **`/news`**: en este directorio se encuentran todos los componentes relacionados con la publicación, edición, visualización y eliminación de las noticias.
- **`/policies`**: contiene el componente con el texto de la política de privacidad de la aplicación.
- **`/user`**: este directorio está destinado a almacenar todos los componentes y servicios relacionados con las funcionalidades que puede realizar un usuario registrado.
    - **`/competitions`**: en este directorio se encuentran todos los componentes relacionados con la publicación, edición, visualización y eliminación de las competiciones
    - **`/familyTree`**: directorio que contiene los componentes y servicios utilizados para la generación y visualización del árbol genealógico de las palomas.
    - **`/pigeons`**: en este directorio se encuentran todos los componentes relacionados con la publicación, edición, visualización y eliminación de los registros de las palomas.
    - **`/users`**: este directorio contiene los componentes utilizados para la visualización y actualización del perfil de usuario.

Cada uno de los directorios dentro de **`/pages`** representa una sección específica de la aplicación y sus rutas asociadas. Por ejemplo:

- **`/news`**: agrupa todo lo relacionado con las noticias.
    - **`/news`**: muestra el listado de noticias.
    - **`/news/read/:id`**: muestra el detalle de una noticia específica (donde *:id* es el identificador de la noticia).

Para definir esas rutas, se han creado ficheros **.routes.ts* dentro de cada directorio principal, los cuales se encargarían de definir las rutas de la sección en la que se encuentren. Estos ficheros definen la asociación de una URL con un componente específico.

Por otro lado, se incluyen ficheros **.service.ts* que contienen los servicios comunes utilizados por los componentes de ese directorio.

El directorio **`/sections`**, presente en algunas carpetas, contiene los componentes comunes que se utilizan dentro de esa sección concreta. Por ejemplo los formularios, que pueden ser usados tanto para crear como para editar registros.

Se puede observar un ejemplo de esta estructura en la figura 21 que pertenece a la sección de noticias.

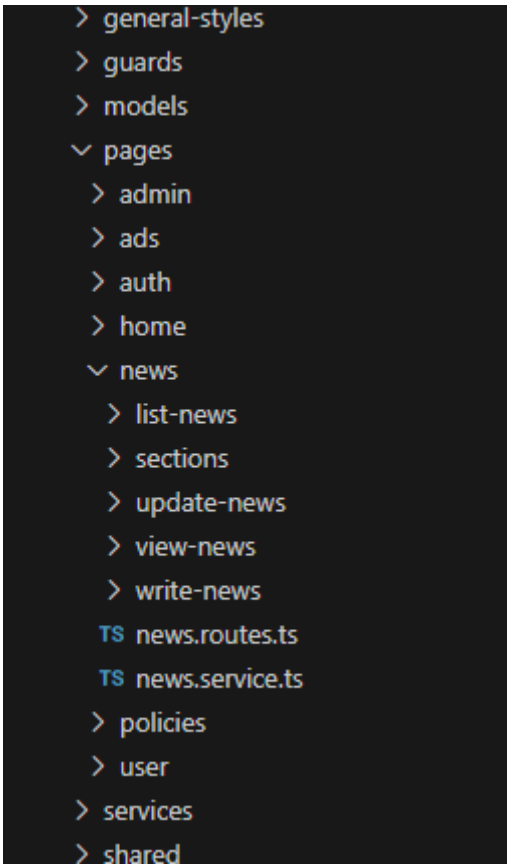

Figura 21: Estructura de directorios del proyecto

Con esta estructura se busca que el código tenga una estructura coherente y resulte sencillo el mantenimiento del mismo.

9.2 Implementación

Finalizadas las fases de análisis y diseño, y con el entorno de desarrollo preparado (como se detalla en el Anexo I) se inicia la fase de implementación de la aplicación. En esta sección, se describirán los aspectos más relevantes desarrollados para crear la aplicación.

9.2.1 Servicios

Los servicios son clases que contienen funciones reutilizables en diferentes partes de la aplicación. Estos servicios se utilizan comúnmente para la comunicación con la base de datos o para compartir datos entre diferentes componentes [47].

A continuación, se explican los servicios generales desarrollados en este proyecto.

9.2.1.1 StorageService

Este servicio se encarga de subir y borrar las imágenes en *Firebase Cloud Storage*. Cuenta con los siguientes métodos.

- ***uploadImage***: este método sube una imagen a *Cloud Storage* en la ruta especificada cómo parámetro. Devuelve la URL desde la cual se podrá visualizar la imagen.
- ***extractPathFromUrl***: esta función privada toma como entrada la URL de una imagen y extrae la ruta en la que se encuentra en *Cloud Storage*.
- ***deleteImage***: este método recibe como parámetro la URL de la imagen y se encarga de eliminarla de *Cloud Storage*. Para obtener la ruta, utiliza la función *extractPathFromUrl*.

9.2.1.2 FirebaseErrorsService

Este servicio consta sólo de un método llamado ***translateErrorCode*** que es utilizado para traducir los códigos de error que emite Firebase en mensajes claros que pueda entender el usuario, facilitando así la comprensión del motivo de error.

9.2.1.3 SpinnerService

Este servicio se utiliza para mostrar u ocultar un indicador de carga cuando se realizan consultas a Firebase. Los métodos incluidos son los siguientes:

- **showLoading**: indica que la pantalla de carga debe mostrarse emitiendo el valor *true* a través de un observable.
- **stopLoading**: indica que la pantalla de carga debe ocultarse, emitiendo el valor *false*.

9.2.1.4 SnackbarService

Este servicio se encarga de mostrar notificaciones a los usuarios utilizando el componente *SnackBar* de Angular Material, que permite presentar mensajes emergentes. Es fundamental mantener al usuario informado sobre el éxito o los posibles errores de las acciones realizadas. Este servicio incluye el siguiente método:

- **showSnackBar**: recibe como parámetros el texto del mensaje a mostrar, el texto que aparecerá en el botón para cerrar la notificación, un número que representa el tiempo en segundos que la notificación permanecerá visible, y el nombre de la clase CSS para personalizar el estilo de la notificación.

9.2.1.5 DatesService

Este servicio está diseñado para gestionar fechas. Muchos de los campos de la aplicación utilizan fechas, como la fecha de nacimiento de una paloma, la fecha y hora de una suelta, o la fecha y hora de llegada. Por esta razón, este servicio es esencial, ya que se encarga de convertir el formato *TimeStamp* de Firebase a una cadena de texto y viceversa. Los métodos incluidos en este servicio son los siguientes:

- **createFirebaseTimestamp**: este método recibe como parámetros una fecha de tipo *Date* de *Javascript*, y tres cadenas de texto con la hora, los minutos y los segundos. La

finalidad de este método es añadir la hora a ese campo de fecha y convertirlo a *Timestamp* de *Firebase*. Este método se utiliza para obtener la fecha y la hora en un solo campo a partir de la información obtenida de los formularios.

- **calculateDatesDifference**: función que tiene como parámetros dos campos *TimeStamp* de *Firebase*, y devuelve la diferencia entre las dos fechas en minutos. Este método se utiliza para el cálculo del tiempo de vuelo en las competiciones.
- **convertSecondsToHourMinutesAndSeconds**: recibe como parámetro un número que representa minutos y devuelve una cadena de texto con el formato "hh:mm:ss". Se utiliza para dar formato al tiempo de vuelo de las competiciones, de forma que resulte más cómodo de entender.
- **fillWithZeros**: esta función recibe como parámetros una cadena de texto y un número que representa el tamaño que debe tener la cadena de texto. Si la cadena de texto es menor que el tamaño especificado, se rellena con ceros a la izquierda. Esta función se utiliza, por ejemplo, para convertir una fecha como esta: 1/2/2024 en 01/02/2024.
- **getCurrentDate**: función que devuelve la fecha actual con el tipo *Timestamp* de *Firebase*.
- **calculateSpeed**: función que calcula la velocidad en metros/minutos. Tiene como parámetros dos números: uno representa la distancia en metros y el otro el tiempo en minutos.
- **separateDateComponents**: es la función inversa de *createFirebaseTimestamp*. Tiene como entrada un parámetro de tipo *Timestamp* de *Firebase* y devuelve un objeto que contiene un campo de fecha y tres cadenas de texto que representan las horas, los minutos y los segundos. Esta función se utiliza para rellenar el formulario de edición de una competición donde los datos son cadenas de texto.

9.2.1.6 AuthService

Este servicio controla todo lo que tenga que ver con la autenticación, registro y modificación de los usuarios. Cuenta con los siguientes métodos:

- **loginWithEmailAndPassword**: permite la autenticación del usuario. Recibe como parámetros el correo electrónico y la contraseña del usuario. Utiliza la función *signInWithEmailAndPassword* proporcionada por *Firebase*.
- **logOut**: finaliza la sesión activa del usuario.
- **sendVerificationEmail:** envía un correo electrónico de verificación al usuario. Utiliza la función *sendEmailVerification* de Firebase.
- **checkIfMailIsVerificated**: función privada que verifica si el usuario ha confirmado su correo electrónico al iniciar sesión. Si no lo ha hecho, lo redirige a la página de "verificación de correo" y le impide navegar por la aplicación.

- **fillUserData**: función privada que completa los datos del usuario con la fecha de registro, el rol (que siempre será "Colombófilo" para registros a través de la aplicación) y la URL de la imagen de perfil.
- **userSignup**: registra un nuevo usuario en *Firebase*. Recibe como parámetros una imagen, un objeto con la información del usuario y el rol. Si el registro es exitoso, envía un correo de verificación mediante `sendVerificationEmail` y redirige al usuario a la página de "verificación de correo".
- **sendLinkToRecoveryPassword**: envía un enlace para restablecer la contraseña al correo electrónico del usuario. Utiliza la función *sendPasswordResetEmail* de *Firebase*.
- **extractErrorCode**: extrae el código de los errores lanzados por *Firestore*. Este código es utilizado por el servicio *FirebaseErrorsService* para traducirlos a mensajes más comprensibles.
- **getUserInfoFromFirebase**: recupera la información de un usuario de *Firebase* según el identificador que se le pasa como parámetro.
- **uploadUserImageToFirestore**: almacena en *Firebase* la imagen de perfil de un usuario. Recibe como parámetros la imagen y en la ruta de almacenamiento.
- **deleteUserImage**: elimina la imagen de perfil de un usuario en *Firestore*. Recibe como parámetro la URL de la imagen.
- **updateUserData**: actualiza la información del usuario en *Firestore* con los datos proporcionados como parámetro.
- **changePassword**: cambia la contraseña del usuario desde su perfil. Recibe como parámetros la contraseña antigua y la nueva. Primero se verifica que la contraseña antigua es correcta antes de proceder con el cambio. Devuelve una promesa con un valor booleano que indica si la opresión tuvo éxito.
- **isAdminUser**: devuelve una promesa con un valor booleano que indica si el usuario actual tiene el rol administrador. Es utilizado por los *guards*.
- **isUserLogged**: devuelve un observable de tipo booleano, que verifica si el usuario actual está autenticado. Es utilizado por los *guards*.
- **IsMailverificated**: devuelve un booleano que indica el estado actual de la verificación del correo del usuario en curso. Este método también es usado por los *guards*.
- **getAllUsersFromFirebase**: recupera la información de todos los usuarios registrados en la aplicación desde *Firebase*.

9.2.1.7 FirebaseService

Este servicio está destinado al manejo de los documentos y las colecciones de Firebase, encargándose de comunicarse con la base de datos para su visualización o actualización. Los métodos que incluye son:

- **saveInFirestore**: guarda un objeto con la información del documento en una ruta específica. Ambos datos se pasan como parámetros.
- **getDocumentFromFirebase**: recupera el documento que se encuentra en la ruta pasada como parámetro con el identificador que también es pasado como parámetro.
- **getCollectionFromFirebase**: recupera la lista de todos los documentos que se encuentran en la ruta pasada como parámetro. Devuelve un observable con un *array* del tipo de datos de la colección.
- **updateDocumentInFirestore**: actualiza los datos de un documento en *Firestore*. Recibe como parámetros el identificador del documento, la ruta donde se encuentra ese documento y un objeto con los datos a actualizar.
- **deleteDocumentInFirestore**: borra el documento de *Firestore* que coincida con la ruta pasada como parámetro.
- **getLastest**: recupera de la base de datos los últimos documentos creados de la colección cuya ruta se especifica como parámetro. La cantidad de documentos a recuperar también se pasa como parámetro.
- **getDocumentsWithQuery**: recupera de la base de datos los documentos que cumplan con una determinada condición. Los parámetros de esta función para crear la consulta son:
  - *path*: ruta de la colección a consultar.
  - *field*: campo en el que se va a realizar la búsqueda.
  - *comparation*: tipo de comparación se va a realizar, por ejemplo, '==' igual a, '>' mayor que, '<' menor que, '!=' distinto a, etc.
  - *condition*: dato con el que se realiza la comparación, es el criterio de búsqueda.
  - order: campo por el que se va a ordenar la consulta.

9.2.2 Pipes

Los *Pipes* en Angular son funciones que permiten transformar los datos antes de mostrarlos en la vista [48]. Su uso es bastante sencillo: basta con colocar una barra vertical después del valor que se quiere transformar, seguida del nombre del pipe. Por ejemplo, un pipe que se ha usado mucho es el que se muestra en la figura 22. La función de ese pipe es transformar el formato de fecha *Date*, que es un número, en un texto con el formato "*dd/mm/aaaa*".

```
<span  class="info-item">{{currentUserData?.registerDate?.toMillis() | date: 'dd/MM/yyyy'}}</span>
```

Figura 22: Ejemplo de uso de un pipe

A continuación, se explica el pipe implementado para esta aplicación.

9.2.2.1 extractWords

Este *pipe* recibe como argumentos una cadena de texto y un número. Su función es extraer de la cadena de texto el número de palabras especificadas por el parámetro. En la figura 23 se muestra un ejemplo de su uso. Al utilizar ese *Pipe* se extraerán las 15 primeras palabras de la cadena de texto que serán las que finalmente se mostrarán en la vista.

```
<p class="item-body">{{ads.details | extractWords:15}}</p>
```

Figura 23: Ejemplo de uso del pipe "Extractwords"

9.2.3 Componentes principales

En esta sección, se describirán los componentes principales implementados en este proyecto.

9.2.3.1 Página de Inicio, menú de navegación, cabecera y pie de página

El primer paso para construir esta aplicación fue crear la página de inicio, que será lo primero que se muestre al acceder a la aplicación. Para conseguir un menú lateral, se utilizó el componente *sidenav* de *Angular Material*. El componente del menú se colocó en la barra lateral, dejando el cuerpo del HTML del componente para añadir la cabecera, que simplemente consta del logo y el botón para activar el menú. Debajo de la cabecera se colocó la directiva *router-outlet*. Esta directiva, reserva el lugar donde el contenido de la aplicación será renderizado según la ruta en la que se encuentre la aplicación.

Para finalizar, también se creó el componente para el pie de página que, muestra el logo de la aplicación y los enlaces que puede visitar el usuario. La figura 24 muestra el resultado final.

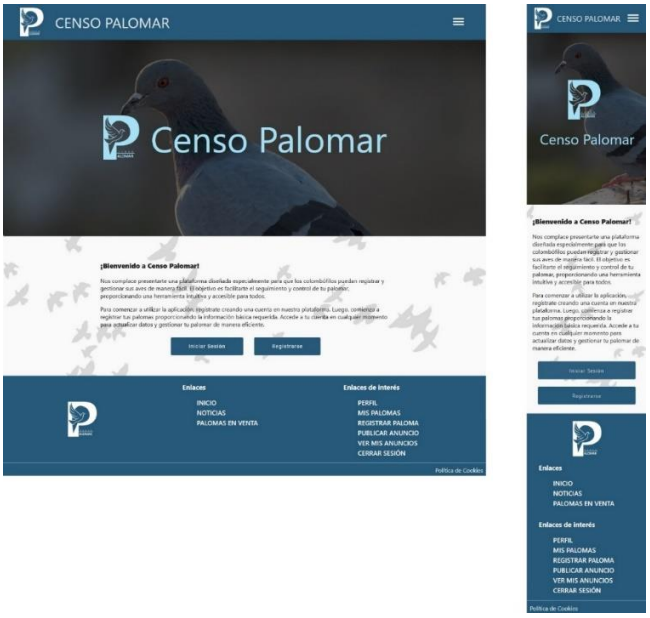

Figura 24: Página de Inicio, a la izquierda versión de escritorio, a la derecha la versión móvil

9.2.3.2 Inicio de Sesión

El inicio de sesión en la aplicación es esencial, ya que es necesario poder identificar al usuario para mostrarle únicamente su información. Para que el usuario pueda autenticarse, es

necesario implementar un pequeño formulario en el que pueda introducir sus credenciales. El resultado final del formulario se puede ver en la figura 25.

Figura 25: Inicio de sesión, a la izquierda versión de escritorio, a la derecha la versión móvil

Para la implementación del formulario, se ha utilizado la librería *@angular/forms*, que proporciona un conjunto de herramientas para la creación de formularios reactivos. Gracias a esta librería y a la clase *Validators*, se puede comprobar automáticamente los datos introducidos en el formulario. En la figura 26 se puede ver el código que inicializa el formulario de inicio de sesión, donde se observa que los dos campos, correo electrónico y contraseña, se establecen como obligatorios. Además, el campo de correo electrónico debe coincidir con un patrón específico para ser validado. Cabe mencionar que la clase *Validators* incluye una función para validar correos electrónicos, pero actualmente no funciona como debería, ya que un correo mal formado como *'usuario@dominio*' pasaría la validación, aunque no es un correo válido.

```
this.loginForm = this.formBuilder.group({
  email: ['', [Validators.required, Validators.pattern(this.emailPattern)]],
  password: ['', [Validators.required]]
});
```

Figura 26: Inicialización del formulario de Inicio de Sesión

En caso de que exista algún dato incorrecto, se muestra un mensaje debajo del campo que contiene el error para que el usuario pueda corregirlo en su caso. En la figura 27 se pueden ver los mensajes de advertencia que se muestran en tales casos.

Figura 27: Mensajes de advertencia por campos incorrectos

En el caso de que la autenticación no sea exitosa debido a una contraseña errónea, un correo no asociado a ningún usuario registrado o una falla en la conexión a internet, se le muestra al usuario un mensaje indicando el motivo del error. En la figura 28 se pueden ver algunos de los mensajes de error más comunes.

Figura 28: Mensajes de error

9.2.3.3 Registro de usuario

El primer paso de un usuario para poder acceder a la aplicación es registrarse en el sistema, y este componente es el encargado de gestionar ese registro. Se le presenta al usuario un formulario que, al igual que en el caso anterior, utiliza las ventajas de *@angular/forms* y la clase *Validators* para comprobar que los datos introducidos sean correctos.

Para el manejo de usuarios, *Firebase Authentication* proporciona la clase *User*, que contiene los datos básicos para identificar a un usuario. Sin embargo, para este proyecto se decidió recopilar también otros datos relevantes, como una imagen de perfil, un nombre de usuario, el rol y el club colombófilo al que pertenece. Por ello, fue necesario almacenar estos datos en una colección específica para poder recuperarlos posteriormente.

La finalidad de recoger datos como el club al que pertenece fue considerar las futuras posibilidades de la aplicación, como campañas de publicidad dirigidas a miembros de un club concreto, priorización de anuncios entre componentes del mismo club colombófilo o, por el contrario, prioridad para los usuarios de fuera del club. Actualmente, recoger el dato del club solo tiene fines estadísticos. En la figura 29 se muestra la vista del formulario tanto en la versión de escritorio como en la versión móvil.

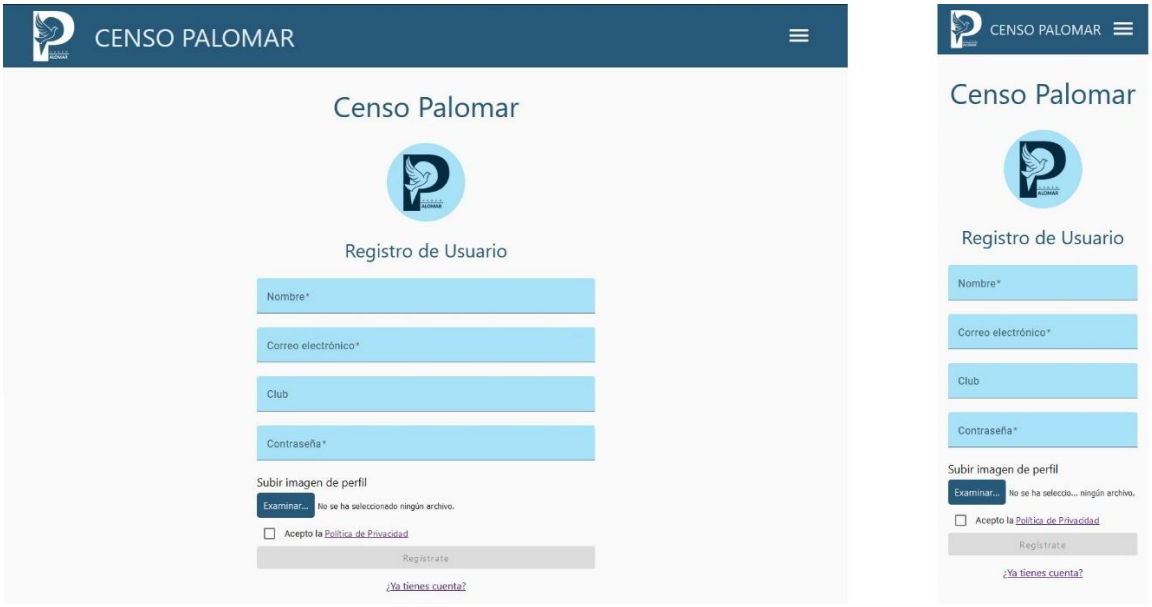

Figura 29:Página de registro de usuario, a la izquierda versión de escritorio, a la derecha la versión móvil

El registro de un usuario consta de tres fases:

- **Registro del usuario en Firebase Authentication**: mediante el uso de la función *createUserWithEmailAndPassword* que proporciona Firebase.
- **Almacenamiento de los datos del usuario**: si el registro ha finalizado con éxito, se guardan los datos del usuario recogidos en el formulario en un documento en la colección "usuarios" de *Firestore*.
- **Verificación del correo y redirección**: se le envía al usuario un correo que contiene un enlace para que valide su correo electrónico y se le redirige a la página de verificación de correo.

En la figura 30 se muestra el código del servicio *AuthService* que realiza el registro del usuario. Como se puede observar, si surge algún error es capturado y es relanzado sólo con el código de error para que el componente puede mostrar al usuario la notificación correspondiente.

```
async userSignup(imageFile: File, userData: UserInterface, role = 'Colombófilo'): Promise<void>{
  try{
    const { user } = await createUserWithEmailAndPassword( this.auth, userData.email, userData.password);
    if (user){
      await this.fillUser(user, userData, imageFile, role);
      this.firebaseService.saveInFirestore(userData, 'usuarios', user.uid);
      this.sendVerificationEmail(user);
      this.router.navigate(['/auth/email-verification']);
    }
  } catch (error) {
    throw(this.extractErrorCode(error));
  }
}
```

Figura 30: Código utilizado para el registro de un usuario

9.2.3.4 Verificación de correo

Una de las ventajas de utilizar Firebase es su funcionalidad para verificar el correo electrónico de un usuario. Esta verificación es crucial para poder establecer un canal de comunicación con los usuarios desde el principio. El proceso es bastante simple: el usuario recibe un correo electrónico que contiene un enlace, y al hacer clic en ese enlace, su correo queda validado.

La función de *Firebase* utilizada para la verificación de correo es *sendEmailVerification*, y recibe como parámetro un objeto de tipo *User*. Cada vez que un usuario se registra, en el proceso final, se llama a esta función, que envía un correo de verificación y redirige al usuario a la página mostrada en la figura 31.

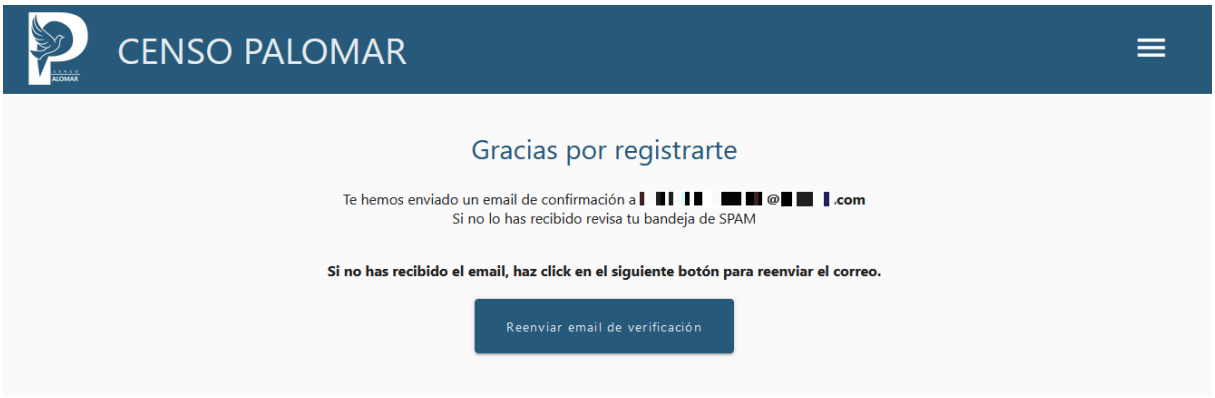

Figura 31: Página de verificación del correo electrónico

Además, cada vez que el usuario inicia sesión, se verifica si su correo electrónico está validado. Si no lo está, el usuario es redirigido a la página de verificación de correo electrónico (figura 31) y no podrá navegar por el resto de la aplicación hasta que valide su correo. Desde esta página, el usuario puede reenviar el correo de verificación.

9.2.3.5 Perfil de usuario

Este componente se encarga de mostrar la información del usuario actual. Además, en la parte inferior, se muestra un botón que lleva al usuario al formulario para modificar sus datos. La información mostrada corresponde a los datos recogidos cuando el usuario realizó el registro y que está almacenada en documento de la colección “usuarios”.

9.2.3.6 Editar Usuario

Para la edición del usuario se han creado dos componentes diferentes. Uno de ellos es exclusivo para actualizar la contraseña, y el otro es para editar los datos del usuario. El motivo de esta separación es que estos datos se encuentran en diferentes servicios de *Firebase*: la edición de los datos de usuario se hace en una colección de *Firestore*, mientras que la actualización de la contraseña se realiza en *Firebase Authentication*. Se consideró realizarlo todo desde un solo formulario, pero finalmente se optó por esta opción al ser más fácil de implementar.

Para la edición de un usuario el primer paso es rellenar el formulario con los datos del usuario. Por lo tanto, primero se recupera de la base datos la información del usuario y se rellena el formulario.

Para la edición de la información de usuario, se ha bloqueado el correo electrónico, ya que se consideró que era un dato importante y que no debía ser modificado. Todos los demás datos si pueden cambiarse. Al igual que los formularios anteriores, se utilizaron *Validators*, para verificar que los datos del formulario fueran correctos antes de que se enviarán al *Firestore*.

Un detalle importante a comentar es la actualización de la imagen de perfil. Es necesario garantizar que, si se actualiza la imagen, la anterior sea eliminada de *Cloud Storage*. Por lo tanto, antes de subir la nueva imagen, se borra la anterior.

En cuanto a la actualización de la contraseña, se le solicita al usuario que complete tres campos, la contraseña actual y la nueva contraseña, que debe repetirse dos veces, como se muestra en figura 32.

Figura 32: Formulario para la actualización de contraseña

Para cambiar la contraseña, el primer paso es verificar que la nueva contraseña coincide con su repetición, garantizando así que el usuario ha introducido la misma contraseña dos veces. A continuación, se comprueba que la contraseña actual es correcta utilizando para ello *reauthenticateWithCredentials* de *Firebase Authentication*. Si la contraseña actual no es correcta, se lanza una excepción que será capturada para mostrar un mensaje de error. En caso contrario, se procede a cambiar la contraseña.

En la figura 33 se muestra el método *changePassword* que realiza esta funcionalidad. Es importante destacar que primero se verifica que el usuario esté registrado, ya que, de no ser así, no podrá cambiar la contraseña por esta vía.

```
async changePassword(oldPassword: string, newPassword: string, user: User): Promise<boolean>{
  if(user.email == null){
    return false;
  }
  try{
    const credentials = EmailAuthProvider.credential(user.email, oldPassword);
    await reauthenticateWithCredential(user, credentials);
    await updatePassword(user, newPassword)
    return true;
  } catch (error){
    throw(this.extractErrorCode(error));

  }
}
```

Figura 33: Código del método changePassword

9.2.3.7 Gestión de Palomas

La gestión de palomas es el pilar fundamental de esta aplicación, diseñada para que los colombófilos puedan llevar un registro detallado y organizado de sus aves. Para llevar a cabo esta gestión de los registros de palomas, se han implementado los siguientes componentes:

Crear registro de una paloma

Este componente simplemente muestra un título y renderiza el formulario para registrar una paloma. Para especificar que se trata de un formulario de registro, se utiliza el decorador *@Input* para comunicárselo al componente del formulario.

Modificar el registro de una paloma

Al igual que el componente anterior, muestra un título y renderiza un formulario para editar el registro de una paloma. A través del decorador *@Input*, se le indica al componente del formulario que se trata de un formulario de edición, proporcionando el tipo de formulario y el identificador de la paloma para que se cargue su información correspondiente. En la figura 34, se puede observar cómo se le pasa esta información al componente de formulario:

```
<app-pigeon-form [typeForm]="'Editar Paloma'" [pigeonId]="id"></app-pigeon-form>
```

Figura 34: Paso de parámetros a través del decorador @Input

Formulario de edición y modificación del registro de una paloma

Para crear el registro de una nueva paloma, el usuario debe proporcionar la información correspondiente a través de un formulario que recopila los datos relevantes para identificar al ave. Al igual que en los formularios anteriores, se utiliza la clase *Validators* para asegurar que los datos ingresados sean correctos. En este caso, la información indispensable para crear un registro es el nombre de la paloma y el número de anilla; los demás campos son opcionales. Esto se debe a que, en un primer momento, puede que no se disponga de todos los datos de la paloma. Por ejemplo, a un pichón recién nacido puede que aún no se le haya identificado el sexo o definido el color de su plumaje, o en el caso de una paloma donada, es posible que no se conozca su fecha de nacimiento exacta.

Este formulario es reutilizado tanto para la creación del registro como para la edición de una paloma, por lo que para identificar el tipo de formulario, se le pasa a través del decorador @Input el tipo de formulario, como se muestra en la figura 34. De esta manera, se garantiza la flexibilidad del componente, permitiendo su adaptación a las diferentes funcionalidades sin necesidad de duplicar código. Cuando este componente se inicia se verifica su tipo para, en caso de que fuera necesario, cargar en el formulario la información de la paloma a editar.

Una característica importante de este formulario es la selección de la fecha de nacimiento utilizando el componente *MatDatePicker* de *Angular Material*. Este componente proporciona una interfaz de calendario amigable para el usuario, como se puede ver en la figura 35, y devuelve un objeto de tipo *Timestamp*, que incluye dos campos, uno para los segundos transcurridos desde el 1 de enero de 1970 y otro para los nanosegundos. Un detalle importante es que este formato es compatible con el formato de fecha de *Firestore,* lo que justifica su utilización.

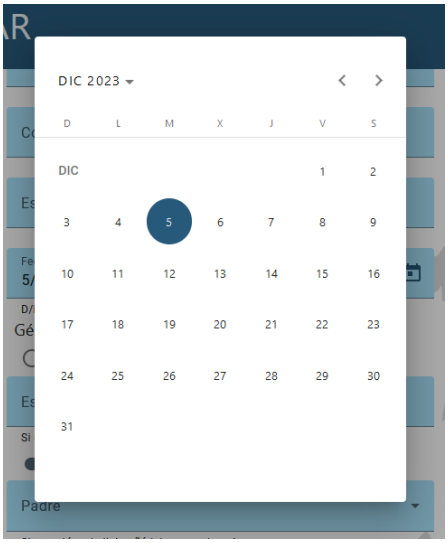

Figura 35: Vista del calendario de Angular Material

Uno de los desafíos de este formulario es permitir al usuario seleccionar a los progenitores de una paloma entre las palomas que ya tiene registradas. Para lograrlo, se utilizó el método *getDocumentsWithQuery* del servicio *FirebaseService*. Este método se emplea para implementar una consulta que recupera un listado de palomas según su sexo para mostrarlas como opciones en un campo de tipo *select*. De este modo, el usuario puede elegir a la paloma padre o madre. En la figura 36 se muestra el código del método mencionado, donde se puede observar cómo se crea la consulta.

```
async getPigeonsByGender(gender: string, userId: string): Promise<PigeonInterface[]>{
  try{
    const path = 'usuarios/'+userId+'/palomas';
    const snapshot = await this.firebaseService.getDocumentsWithQuery(path, 'gender', '==', gender, 'pigeonName');
    return snapshot.docs.map( doc => doc.data() as PigeonInterface);
  }catch (error){
    throw (error);
  }
}
```

Figura 36: Método para recuperar un listado de palomas según el género

El método anterior funciona muy bien y permite seleccionar al padre y a la madre de la paloma. Sin embargo, ¿qué pasa cuando los progenitores de esa paloma no están registrados?

Para estos casos, se creó una funcionalidad adicional mediante un campo de tipo *checkbox* que permite mostrar u ocultar un campo de texto. Así, el usuario puede cumplimentar este campo con el nombre del progenitor de la paloma o elegirlo del listado de palomas, si ya estaba registrada. Además, esta variable de tipo *checkbox* se guarda en la base de datos como un booleano, ya que será útil a la hora de cargar la información en el formulario para editar la paloma.

Antes de almacenar o actualizar la información de una paloma en *Firestore*, es necesario preparar el objeto con toda la información recogida en el formulario. Para esto, se implementó la función que se muestra en la figura 37. Esta función se encarga de preparar y rellenar el objeto con la interfaz *PigeonInterface*, que contiene todos los campos del registro de una paloma descritos en el apartado de base de datos. Este objeto es el que se enviará a *Firestore* para almacenar la información.

Para preparar este objeto, se le asignan todos los datos recogidos en el formulario. Luego, se verifica si se trata de una edición del registro existente para determinar si la imagen ha cambiado. Si la imagen ha cambiado, primero se elimina la imagen actual de *Firebase Cloud* y luego se sube la nueva. En caso de que sea un nuevo registro, simplemente se sube la imagen seleccionada. Cabe señalar que el dato que se guarda en el objeto es la URL de la imagen subida. Es importante indicar que, si no se elige ninguna imagen a la hora de un registro nuevo, se asigna la imagen por defecto establecida.

Finalmente, si es un nuevo registro, se almacena la hora actual del sistema y se le asigna un identificador. Este identificador se forma concatenando la fecha actual en segundos desde enero de 1970, los nanosegundos y el número de anilla de la paloma, asegurando así un identificador único para cada paloma. Además, esta estructura permite ordenar fácilmente las palomas según su fecha de registro, ya que está incluida en su ID.

En caso de que se trate de una actualización, se conservarán los datos del ID y la fecha de registro, ya que estos no deben modificarse.

```
private async prepareFormData(): Promise<PigeonInterface>{
  let pigeonData: PigeonInterface = this.pigeonForm.value;
  if (this.typeForm === 'Editar Paloma' && this.imageFile.name === 'null.null'){
    pigeonData.image = this.currentPigeon!.image;
  } else {
    if (this.typeForm === "Editar Paloma"){
      this.pigeonService.deletePigeonImage(this.currentPigeon!.image);
    }
    pigeonData.image = await this.pigeonService.uploadImageToFirestore( this.imageFile, 'Palomas/'+this.currentUser?.email);
  }
  if (this.typeForm === 'Editar Paloma' && this.currentPigeon != null){
    pigeonData.registerDate = this.currentPigeon.registerDate;
    pigeonData.id = this.currentPigeon.id;
  } else {
    pigeonData.registerDate = Timestamp.fromDate(new Date());
    pigeonData.id = pigeonData.registerDate + '-' + pigeonData.ring.replace(/ /g,"-");
  }
  return pigeonData;
}
```

Figura 37: Función que crea un objeto de tipo PigeonInterface con los datos recogidos del formulario

Vista de la información de la paloma

La misión de este componente es recuperar de *Firestore* la información de una paloma y presentarla de manera organizada para que el usuario pueda identificar rápidamente los datos que necesite consultar. En la figura 38, se puede ver la vista implementada para mostrar la información de una paloma.

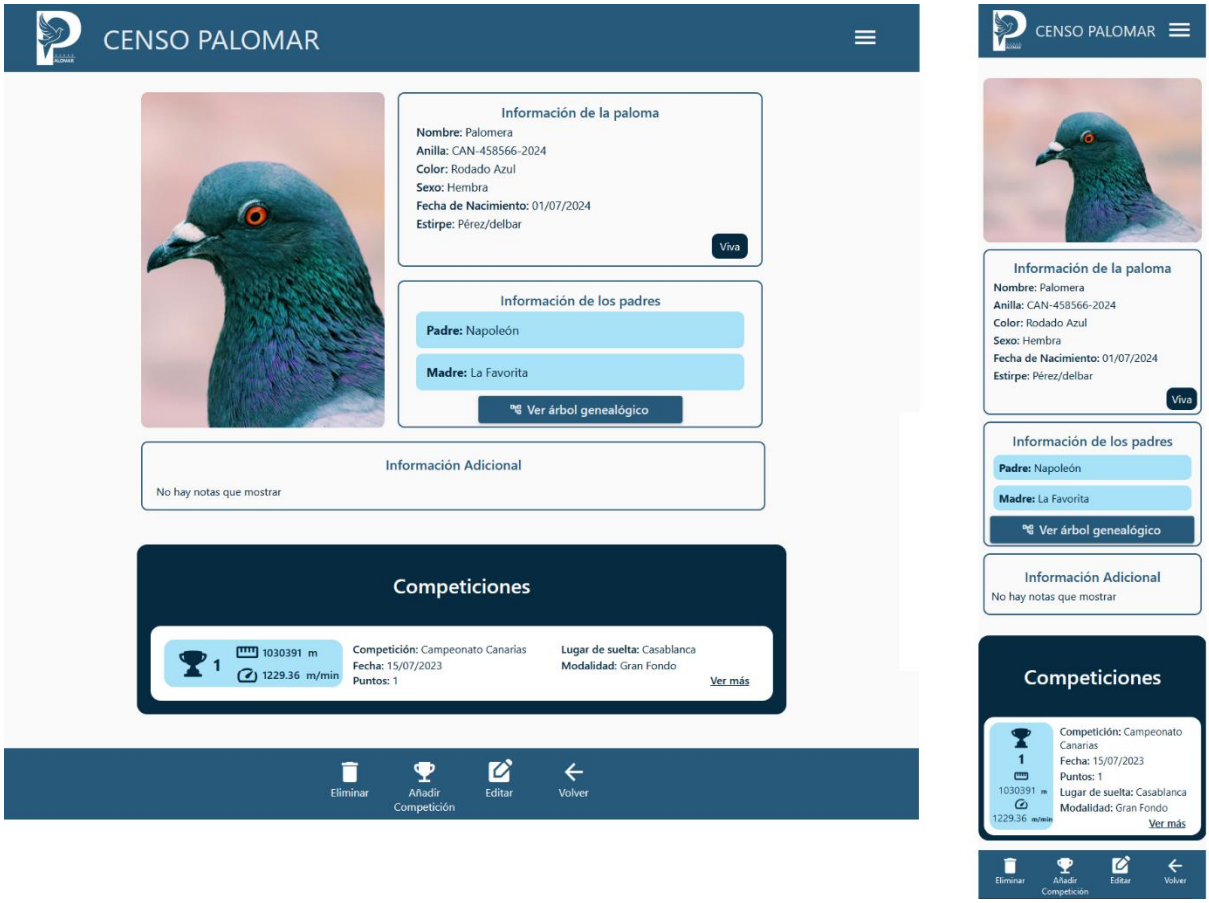

Figura 38: Página del perfil de una paloma, a la izquierda versión de escritorio, a la derecha la versión móvil

Barra de herramientas

Cuando el usuario revisa el perfil de una paloma, en la parte inferior encontrará una barra de herramientas con las acciones que puede realizar sobre el registro que está consultando. En la figura 39 se puede ver la barra de herramientas mencionada. Las acciones disponibles son:

- **Editar la paloma**: el usuario es redirigido a la página de edición del registro de la paloma, como se pudo ver en apartados anteriores.
- **Añadir una competición**: el usuario puede añadir una competición a esta paloma.
- **Borrar el registro de una paloma**: elimina de la base de datos el documento de esta paloma, incluyendo la imagen del registro. Esta eliminación también abarca las subcolecciones, como las competiciones.
- **Volver**: vuelve al listado de palomas.

Figura 39: Barra de herramientas

Para que este componente funcione correctamente, se le pasa a través del decorador *@Input* el identificador de la paloma y la ruta para editar la paloma. Este último dato es necesario para diferenciarlo del registro de una competición, que también reutiliza este componente.

Mostrar listado de las palomas

Este componente se encarga de mostrar un listado con las palomas que el usuario tiene registradas. El listado contiene la información básica de cada paloma. Si el usuario desea ver más detalles, puede seleccionar la fila de la paloma deseada y será redirigido a la página con los detalles del ave. En la figura 40 se puede observar la implementación realizada.

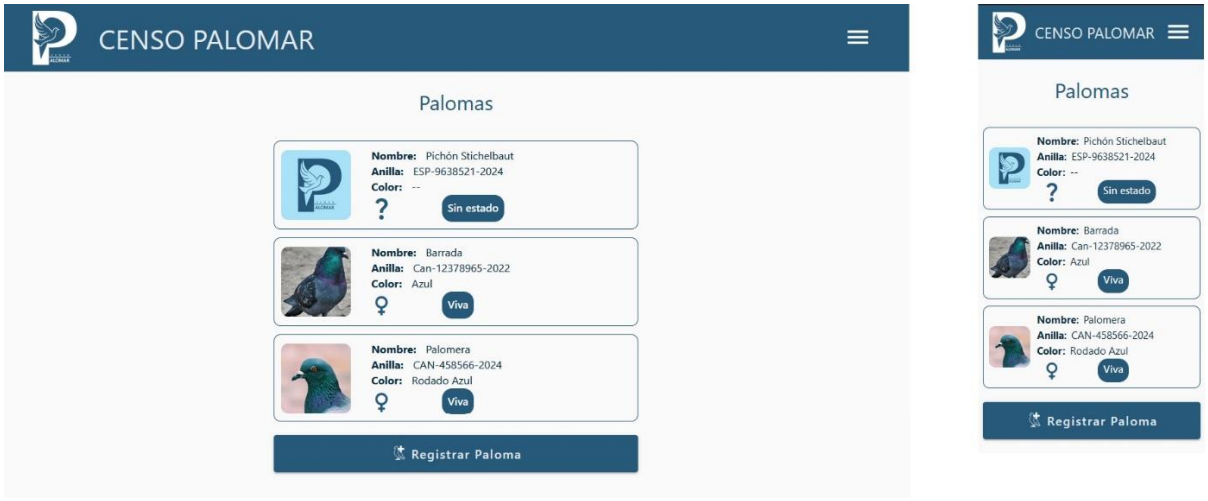

Figura 40: Página del listado de palomas, a la izquierda versión de escritorio, a la derecha la versión móvil

En caso de que el usuario no tenga ninguna paloma registrada, se muestra un texto animándolo a que comience a registrar sus palomas. Para recuperar el listado de palomas, se utiliza el servicio *FirebaseService*, explicado en la sección de servicios. Si surge algún error, se muestra un mensaje emergente indicando el motivo del error.

9.2.3.8 Gestión de competiciones

Para un colombófilo es importante conocer la valía de una paloma. Su valor se basa en varios factores como sus características físicas, la salud del animal o su estirpe, pero otra de las métricas es cómo concursa. Como se comentó en el en el apartado del diseño de la base datos, las competiciones se almacenan en una subcolección dentro del documento de una paloma.

Ver competición

La misión de este componente es recuperar de *Firestore* la información de una competición y mostrarla de manera clara y atractiva. En la figura 41, se puede ver la vista implementada para mostrar la información de una competición.

Figura 41: Página del detalle de una competición, a la izquierda versión de escritorio, a la derecha la versión móvil

Listado de competiciones

Este componente se encarga de mostrar un listado con las competiciones en las que ha participado una paloma. El listado contiene la información básica de cada competición. Si el usuario desea ver más detalles, puede seleccionar la fila con la competición deseada y será redirigido a la página con los detalles de esa competición concreta. En la figura 42 se puede observar la implementación realizada.

Figura 42: vista de la lista de competiciones, a la izquierda versión de escritorio, a la derecha la versión móvil

Añadir competición

Este componente muestra un título y renderiza el formulario para registrar una competición en la colección de competiciones de la paloma que se está viendo. Para especificar que se

trata de un formulario de registro de una competición, al igual que se hizo con el registro de una paloma, se utiliza el decorador *@Input* para comunicárselo al componente del formulario.

Editar el registro de una competición

Este componente también muestra un título y renderiza el formulario para editar el registro de una competición. A través del decorador *@Input*, se le indica al componente del formulario que se trata de un formulario de edición, proporcionando el tipo de formulario, el identificador de la paloma y el identificador de la competición para poder formar la ruta dentro de *Firestore* y cargar la información correspondiente en el formulario. En la figura 43, se puede observar cómo se le pasa esta información al componente de formulario.

```
<app-competition-form
    [pigeonId]="pigeonId"
    [competitionId]="competitionId"
    [typeForm]="'Editar Competición'">
</app-competition-form>
```

Figura 43: paso de información al formulario de edición de competiciones a través de @Input

Formulario de edición y modificación del registro de una paloma

Este componente muestra un formulario que recoge los datos para crear o modificar el registro de una competición. Al igual que el formulario para el registro de las palomas, se utiliza el decorador *@Input* para recibir las siguientes variables:

- *pigeonId*: es el identificador de la paloma, necesario para crear la ruta hacia la colección de las competiciones.
- *competitionId*: en el caso de se esté editando una competición, se recibe el identificador de la competición.
- *typeForm*: especifica si se trata de la creación de un nuevo registro de competición o de una actualización. Esta cadena también es usada para poner la etiqueta al botón de enviar del formulario.

La característica principal de este formulario es recoger la fecha y la hora de una suelta, tanto de su comienzo como de su finalización. En un primer momento, se intentó utilizar una librería *DatetimePicker*, que muestra al usuario un calendario para elegir la fecha y, debajo, muestra selectores para elegir la hora. Sería algo similar a lo que se muestra en la figura 44.

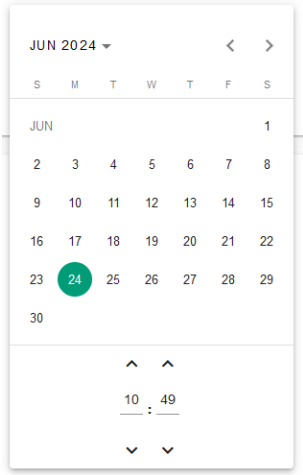

Figura 44: DatetimePicker

Para lograr esto, se instaló una librería que en principio parecía que funcionaba bien, hasta que todo lo desarrollado hasta el momento empezó a presentar fallos. Por ejemplo, las fechas recogidas no tenían sentido, no se almacenaba la hora correctamente y, constantemente, el navegador no era capaz de renderizar la página. Después de intentar ir arreglando los fallos que surgían, se optó por volver al *commit* de *git* más cercano y buscar otra solución porque se estaba perdiendo mucho tiempo en intentar que esta librería funcionara.

La solución implementada consiste en que el componente *MatDatePicker* de Angular Material, que ya se había usado para el formulario del registro de las palomas, recogiera la fecha. Sin embargo, la fecha que recoge siempre tiene la primera hora del día, es decir, las "00:00:00". Por lo tanto, se crearon en el formulario tres campos específicos para recoger la hora, los minutos y los segundos. Llegados a este punto, ya se puede obtener del formulario todos los datos para formar una fecha con la hora, por lo que solo queda unirlo todo. El código para formar una fecha con tipo *Timestamp* de Firebase se puede ver en la figura 45. Básicamente, lo que hace este método es convertir la fecha a una cadena de texto y construir una nueva con el formato ISO estándar con todos los datos. Finalmente, esa cadena de texto se convierte en *Timestamp* y se devuelve.

```
createFirebaseTimestamp(date: Date, hour: string, minutes: string, seconds: string): Timestamp | null{
  if (hour == null || minutes == null || seconds == null || date == null){
    return null;
  }
  const partsDate = date.toLocaleString('es-Es').split('/');
  let stringDate = partsDate[2].split(',')[0] + '-' + this.fillWithZeros(partsDate[1], 2) + '-' +
      this.fillWithZeros(partsDate[0], 2) + 'T' + hour + ":" + minutes + ":" + seconds + ".000";
  return Timestamp.fromDate(new Date(stringDate));
}
```

Figura 45: Método que transforma los datos recogidos del formulario en TimeStamp

El método recoge la fecha y la hora correctamente, aunque ha complicado el proceso un poco. Sin embargo, todavía queda un problema: en el formulario hay que validar que los datos sean coherentes. No tiene sentido que en las horas se introduzca un valor superior a 23, o que en los minutos se pueda poner un 90. Para controlar esto, se utilizaron expresiones regulares para que la clase *Validators* pudiera comprobar la validez de esos campos y mostrar los

correspondientes mensajes de error en caso de datos incorrectos. A continuación, se explican las expresiones regulares utilizadas:

- **Horas**: "/^0[0-9]|1[0-9]|2[0-3]$/" Esta expresión regular indica que el primer carácter puede ser un 0 o un 1 seguido de otro carácter entre 0 y 9, o un 2 seguido de un número del 0 al 3. Con esto se garantizan valores de dos dígitos del 00 al 23.
- **Minutos y segundos**: "/^0-5][0-9]$/" Esta expresión indica que el primer carácter puede ser un número del 0 al 5 y el segundo un número del 0 al 9, lo que garantiza valores desde el 00 hasta el 59.

Con todo lo implementado, se puede crear el registro de una competición sin inconvenientes. El problema ahora radica en que, para poder editar la fecha y la hora, hay que hacer el proceso inverso y, a partir de un objeto de tipo *Timestamp*, convertir la hora, los minutos y los segundos a cadena de texto. Para esto, se utiliza el método que se muestra en la figura 46. Este método convierte el objeto *Timestamp* a *Date* y, con los métodos que proporciona la clase *Date*, se pueden extraer la hora, los minutos y los segundos. Se devuelve un objeto con estos datos por separado para que la vista pueda rellenar el formulario de edición.

```
separateDateComponents(timestamp: Timestamp | null | undefined){
  if (timestamp == null || timestamp == undefined){
    return null;
  }
  let date = new Date (timestamp.toMillis());
  const objectDate = {
    date : date,
    hour: this.fillWithZeros(date.getHours().toString() as string, 2) ,
    minutes: this.fillWithZeros(date.getMinutes().toString(), 2),
    seconds: this.fillWithZeros(date.getSeconds().toString(), 2)
  };
  return objectDate;
}
```

Figura 46: Método para extraer de un timestamp los componentes de la fecha y la hora

Solventado el problema con las fechas, ahora se podían obtener los datos que se calculan automáticamente y que no es necesario que introduzca el usuario. En este caso, es la duración del vuelo, que se calcula con una resta de las fechas de salida y llegada. Con este dato y la distancia, se calcula la velocidad en metros por minuto.

Finalmente, y como se hizo con el formulario del registro de palomas, se creó una función en el componente que crea un objeto de tipo *CompetitionInterface* y lo completa con los datos obtenidos del formulario, los datos calculados y, en el caso de que se tratara de un nuevo registro, la fecha actual y el identificador. Este identificador se forma concatenando la fecha actual en segundos desde enero de 1970, los nanosegundos y el nombre de la competición. En el caso de que se tratara de una modificación, se conservarían estos datos. Este objeto es el que se envía a *Firestore* para almacenar la información de la competición.

9.2.3.9 Gestión de Anuncios

A diferencia de las competiciones y los registros de palomas donde la información se queda en el entorno privado del usuario. Los anuncios cuando se publican pueden ser leídos por

todos los usuarios. Por lo que estas no se guardaran en el documento del usuario, si no en la colección “anuncios”.

En este caso la recolección de la información se realiza con un formulario como en los casos anteriores. Y se utiliza la clase *Validators* para comprobar que el formato de los datos no sea erróneo.

Para la gestión de los anuncios se han creado los siguientes componentes:

- **ListAdsComponent**: este componente recupera de la base de datos todos los anuncios y los muestra en forma de lista.
- **AdvertisementComponent**: este componente muestra el contenido completo del anuncio seleccionado.
- **PublishAdsComponent**: muestra el formulario para publicar un nuevo anuncio.
- **EditAdsComponent**: este componente muestra el formulario con la información del anuncio para su actualización.
- **FormAdsComponent**: al igual que con los registros de palomas y las competiciones, se ha creado un formulario que sirve tanto para la publicación como la modificación de los anuncios y que para validar los campos se apoya en la clase *Validators*.
- **AdsHomeComponent**: recupera de la base de datos los tres últimos anuncios que han sido publicados y muestra sus resúmenes. Este componente es usado en la página de inicio
- **MyAdsComponent:** se muestra el listado de noticias que ha publicado el usuario. Además, en cada anuncio se muestran iconos que representan las acciones que el usuario puede realizar sobre ese anuncio. Estas acciones son visualizar los detalles del anuncio, editar el anuncio y eliminarlo. El resultado final de este componente se puede ver en la figura 47.

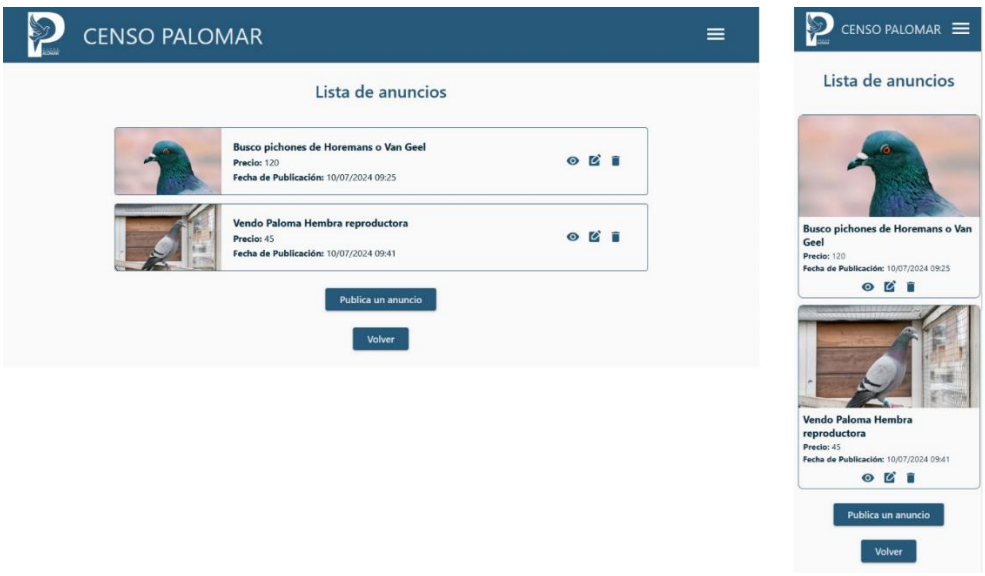

Figura 47: vista de la página my-ads, a la izquierda versión de escritorio, a la derecha la versión móvil

Para almacenar, editar, recuperar y borrar un anuncio se utiliza el servicio *FirebaseService* que se explica en la sección de servicios.

9.2.3.10 Gestión de Noticias

Las noticias solo pueden ser creadas por el usuario con rol administrador, pero deben ser leídas por todos los tipos de usuarios. Para conseguir esto, las rutas para la edición y publicación de las noticias se protegieron con un *guard* que solo permite el acceso si el rol es administrador, aunque esto se explicará en los siguientes apartados.

Para la gestión de noticias se han creado los siguientes componentes:

- **ListNewsComponent**: este componente recupera de la base de datos todas las noticias y las muestra en forma de lista.
- **NewsHomeComponent**: recupera de la base de datos las tres últimas noticias y muestra sus resúmenes. Este componente es usado en la página de inicio.
- **ViewNewsComponent**: este componente muestra el contenido completo de la noticia seleccionada.
- **WriteNewsComponent**: este componente sólo es accesible para el administrador y muestra el formulario para publicar una nueva noticia.
- **UpdateNewsComponent**: este componente muestra el formulario con la información de la noticia para su modificación.
- **FormNewsComponent**: al igual que con los anuncios, los registros de palomas y las competiciones, se ha creado un formulario que sirve tanto para la publicación como para la modificación de las noticias y que, para validar los campos, se apoya en la clase *Validators*. Este formulario también incluye un método *prepareFormData*, al igual que los formularios anteriormente vistos, para gestionar si la imagen en la edición ha sido cambiada.
- **AdminBarComponent**: menú que muestra las diferentes acciones que se pueden realizar sobre la noticia que se está consultando. Las acciones son: editar noticia, añadir nuevas noticias, borrar la noticia y acceder al panel de administrador. Esta barra sólo es visible para los usuarios administradores.

Para almacenar, editar, recuperar y borrar una noticia se utiliza el servicio *FirebaseService* que se explica en la sección de servicios.

9.2.3.11 Panel de administración

En principio, este componente no estaba en la planificación inicial, sino que se añadió en iteraciones posteriores. La función de este componente es agrupar las tareas que puede realizar el administrador. Este componente muestra un listado de botones, como se puede ver en la figura 48, que contienen los enlaces para acceder a las funcionalidades específicas del administrador.

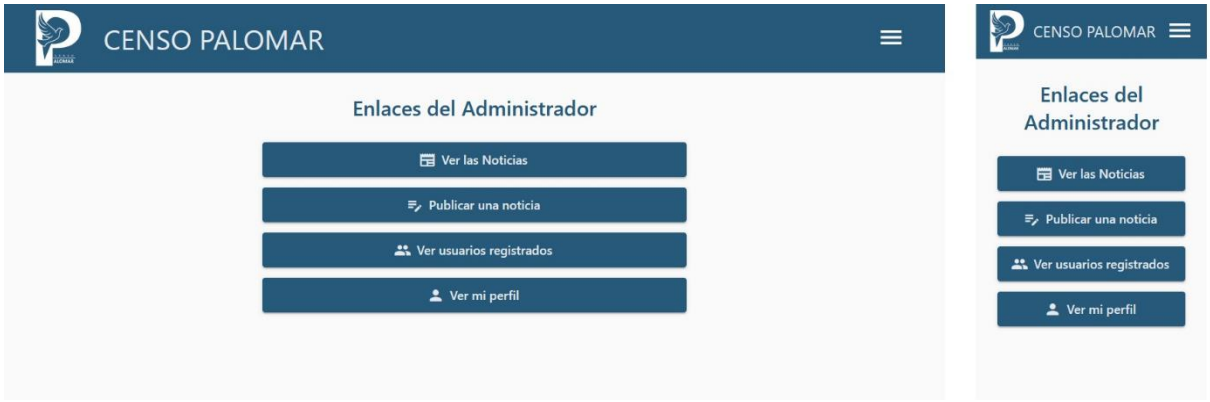

Figura 48: Panel del administrador

9.2.3.12 Árbol Genealógico

Como se explicó en la sección del diseño de la base de datos, cada paloma puede guardar en su documento la información de sus padres. Concretamente, se guardan dos campos en relación a sus padres: el nombre del progenitor, que se utiliza en el perfil para mostrar el nombre en lugar del identificador, y el identificador del progenitor. Con estos dos campos se construirá el árbol genealógico.

Al almacenar los datos de esta manera, se establece la estructura de un árbol binario para almacenar la información, ya que cada paloma guarda la referencia de sus progenitores.

Antes de continuar se explicará brevemente qué son los árboles y en concreto que es un árbol binario.

"Un árbol es un tipo de datos abstracto ampliamente utilizado que representa una estructura jerárquica de árbol con un conjunto de nodos conectados" [49]. En esta estructura se parte de un nodo principal a partir del cual se van ramificando los demás nodos. A continuación, se explican algunos conceptos clave de los árboles:

- ***Nodo***: es el elemento que contiene la información, este elemento también contiene la información de los siguientes nodos
- ***Nodo hijo***: son los nodos que tienen un padre, es decir que están referenciados por otro nodo.
- ***Nodo padre***: son todos aquellos nodos que tienen un hijo, es decir que tienen la referencia para acceder a nodos hijo.
- ***Raíz***: es el único nodo que no tiene padre dentro del árbol, es el primer nodo del árbol.
- ***Nodo hoja***: es un nodo que no tiene hijos, estos nodos indican el final de una rama.
- ***Rama***: es un subárbol dentro del árbol.

En la figura 49 se puede ver la representación de un árbol, donde el nodo raíz, apunta a sus nodos hijos y estos a su vez apuntan a sus nodos hijos, hasta llegar a los nodos hoja.

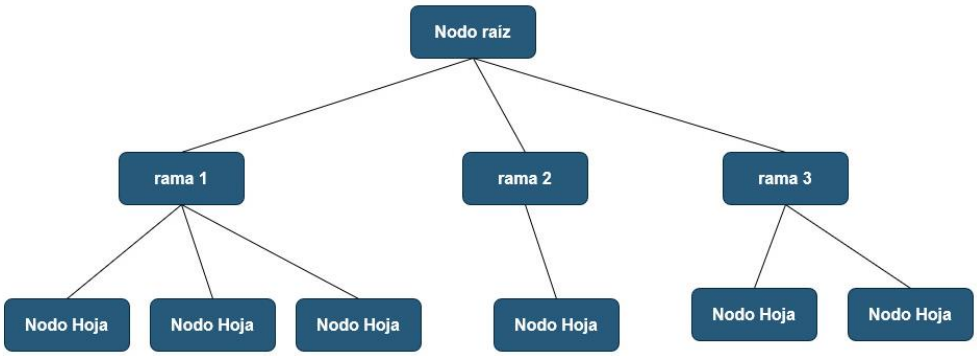

Figura 49: Representación de un árbol

Un árbol binario es un tipo de árbol donde cada nodo puede tener como máximo dos nodos hijos. En la figura 50 se puede ver la representación de un árbol binario.

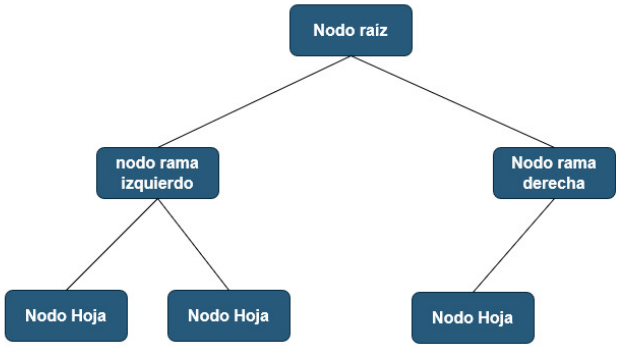

Figura 50: Representación de un árbol binario

En la aplicación desarrollada, la información de la genealogía de una paloma se distribuye como se indica en la figura 51, donde se puede ver el paralelismo con un árbol binario. De hecho, uno de los usos más comunes de un árbol binario es la representación de árboles genealógicos.

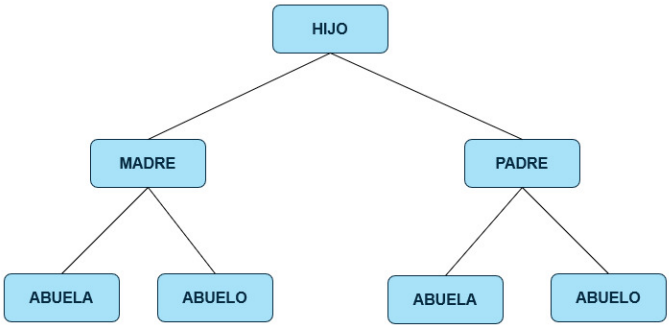

Figura 51: Representación de un árbol genealógico

Para el manejo de estas estructuras de datos y mostrar el árbol genealógico de una paloma, se han implementado los siguientes componentes y servicios que se detallan a continuación.

FamilyTreeService

Este servicio se encarga de la generación del árbol genealógico y dispone de los siguientes métodos:

levelIsValid

Función privada que comprueba si el número pasado por parámetro puede representar la cantidad de nodos que contiene un árbol binario. Realmente lo que comprueba es si el número pasado + 1 es una potencia de 2. Es importante comprobar este número porque será el tamaño del *array*.

getParents

Este método recibe por parámetros un objeto con la información de la paloma, el identificador del usuario para poder establecer la ruta donde están alojadas las palomas en *Firebase* y el número de nodos a rellenar. Luego crea un *array* con cadenas de texto vacías que la función *fillArrayWithParents* se encargará de rellenar con la información de los ascendientes de la paloma.

fillArrayWithParents

Esta función privada se encarga de crear la representación de un árbol binario en un *array*. En esta representación, el nodo raíz se almacena en la primera posición del *array*, el nodo izquierdo en la posición 2i+1 (donde la i representa el índice del *array*) y el nodo derecho en la posición 2i+2. En la figura 52 se puede ver una representación de la implementación de un árbol binario en un *array* [50].

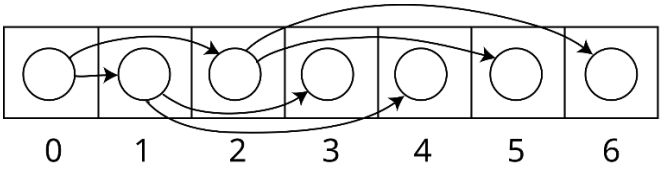

Figura 52: Representación de un árbol binario en un array, extraída de Wikipedia [50]

Para rellenar el *array*, se recorre el árbol binario utilizando recorrido en preorden, donde primero se lee la raíz del árbol, luego se continua por el lado izquierdo y finalmente por el lado derecho. Mientras se va recorriendo el árbol va rellenado el *array* que será el que se le pase al componente para que lo renderice en la vista.

Un detalle importante es que un nodo puede tener estas tres posibilidades:

- *El nodo contiene el identificador del progenitor*: el campo *motherId* o *fatherId* que almacena el identificador de los progenitores no está vacío, por lo que se puede visitar ese nodo y recuperar la toda la información de la paloma progenitora.
- *El nodo solo contiene el nombre del progenitor*: el campo *motherId* o *fatherId*, están vacíos pero el campo *mother* o *father* que almacenan el nombre sí tienen contenido. Esto significa que los padres no están registrados en la aplicación. En este caso, se almacena en el *array* la cadena de texto con el nombre en la posición que le corresponde. Para calcular la posición se utiliza la fórmula 2i+1 para las palomas hembras (árbol izquierdo) y 2i+2 para las palomas macho (árbol derecho).
- *Nodo sin información de los progenitores*: No hay información sobre los padres. Se devuelve una cadena de texto vacía.

El código descrito se puede ver en la figura 53.

```
private async fillArrayWithParents (pigeon: string,
                                    path: string,
                                    index: number,
                                    nodes: number,
                                    familyArray: (string | PigeonInterface)[])
                                    : Promise<any>{
  if (index >= nodes) return;
  if (pigeon == null || pigeon == '' || pigeon == undefined) return '';
  try{
    let currentPigeon = await this.firebase.getDocumentFromFirebase(pigeon, path).then(
      response => {return response.data() as PigeonInterface;}
    );
    familyArray[index]=currentPigeon;
    let mother = await this.fillArrayWithParents(currentPigeon.motherId, path, 2*index+1, nodes, familyArray);
    if (mother == ''){
      if (currentPigeon.mother != null && currentPigeon.mother != undefined){
        familyArray[2*index+1] = currentPigeon.mother;
      }
    }
    let father = await this.fillArrayWithParents(currentPigeon.fatherId, path, 2*index+2, nodes, familyArray);
    if (father == ''){
      if (currentPigeon.father != null && currentPigeon.father != undefined){
        familyArray[2*index+2] = currentPigeon.father;
      }
    }
  } catch (error){
    throw(error);
  }
}
```

Figura 53: Función que recorre el árbol y lo transforma en un array

Uno de los aspectos más importantes de crear la representación del árbol binario de esta manera es que, al recorrer el *array* en orden, es como si el árbol se recorriera en anchura o por niveles. Esto significa que se empieza por el nodo raíz, luego se avanza al siguiente nivel de profundidad y se leen los nodos de izquierda a derecha, continuando así hasta llegar al último nivel o hasta el nivel deseado.

Una de las ventajas de este orden en el *array* es que, a la hora de dibujarlo en el HTML, es mucho más fácil. Al estar ordenado, simplemente basta con recorrer el *array*, identificar el nivel a partir del índice y, si es un nuevo nivel, crear un nuevo espacio debajo, colocando en ese espacio las hojas del árbol de izquierda a derecha.

Además, con este orden, las palomas hembra están en las posiciones impares del *array* y los machos en las pares. Esta característica se utilizará para colorear el fondo de las hojas de diferentes colores dependiendo del sexo de la paloma.

FamilyLeaf

Este componente es un nodo del árbol genealógico. Utiliza el decorador *@Input* para recibir la información de la paloma desde el componente padre. Si la información pasada es un objeto del tipo *PigeonInterface*, se muestra la información más importante de la paloma, como el nombre, el número de anilla, la fecha de nacimiento, la estirpe y el estado. Pero si la información pasada es de tipo cadena de texto, se muestra esa cadena de texto. Al pasarse una cadena de texto, significa que los padres de esa paloma no están registrados, pero se ha puesto manualmente ese nombre.

FamilyTree

Este componente se encarga de dibujar en forma de pirámide el árbol genealógico de una paloma. En la parte superior se coloca la paloma de la que se va a generar su árbol genealógico; en la fila siguiente se colocan sus dos padres; en la siguiente, sus cuatro abuelos; y en la siguiente, sus ocho bisabuelos. Durante las fases de diseño se estableció que el límite iba a ser hasta los abuelos, pero durante la implementación se amplió este límite hasta los bisabuelos, ya que no suponía una dificultad mucho mayor.

Es importante mencionar que, para los dispositivos móviles, solo se muestra hasta los abuelos porque el diseño no terminaba de cuadrar y el contenido se volvía ilegible. De hecho, para mostrar correctamente a los abuelos, la fila se tuvo que dividir en dos. El diseño final se puede ver en la figura 54.

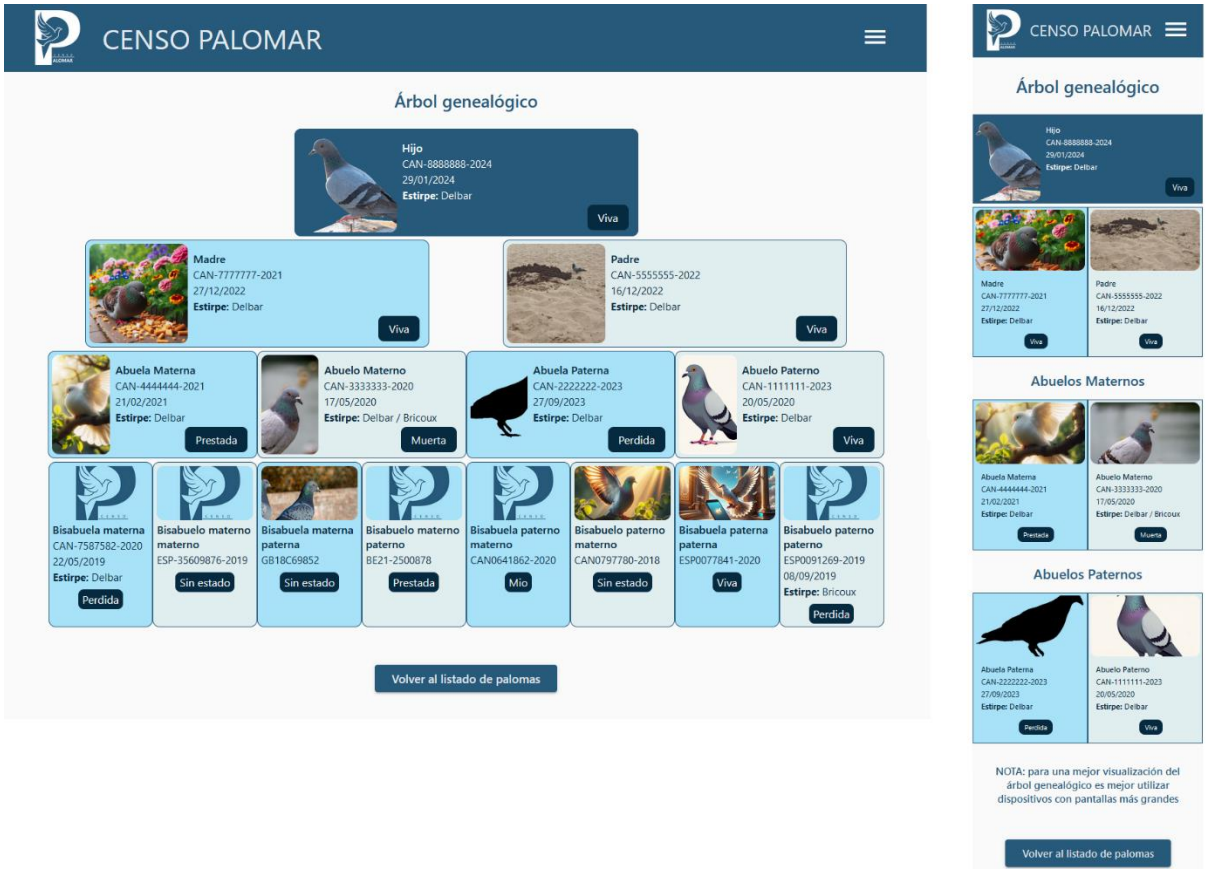

Figura 54: Diseño final del árbol genealógico, a la izquierda versión de escritorio, a la derecha la versión móvil

Para controlar si la aplicación se está ejecutando en un entorno con una pantalla reducida, se utiliza la clase `BreakPointObserver`, proporcionada por la librería `@angular/cdk`, que pertenece a *Angular Material*. Esta clase, mediante un observable, detecta el ancho de la pantalla. En el caso de que la pantalla corresponda a un entorno móvil, se establece que el tamaño máximo del árbol sea de 7 nodos, lo que significa que no se muestran los bisabuelos. En caso contrario, en pantallas grandes, el tamaño del árbol será de 15 nodos.

Para organizar la vista del árbol se utiliza la función de SASS que se muestra en la figura 55. Esta función calcula el ancho que deben tener los nodos según el nivel en el que se encuentran. El ancho de cada nodo debe ser la mitad del nivel anterior, por lo que el nivel 0 tendrá un

ancho del 100%, el nivel 1 tendrá entonces un ancho del 50%, y así sucesivamente, lo que se corresponde con la fórmula:

$$ancho(\%) = \frac{100}{2^{nivel}}$$

```
@function percentageCalc($level)
    @return math.div(100%, math.pow(2, $level))

@for $i from 0 through 3
    .container-tree
        .wrapper-tree
            .class-level-#{$i}
                width: percentageCalc($i)
                margin-bottom: 5px
```

Figura 55: Función SASS que calcula el ancho del nodo

Una vez definidas las clases, en la plantilla HTML, se asigna a cada hoja la clase correspondiente a su nivel. Para saber a qué nivel pertenece una hoja se utiliza la siguiente fórmula:

$$nivel = log_2\,(índice + 1)$$

En la figura 56 se puede ver la implementación de la función que calcula el nivel, y en la figura 57 como se le asigna la clase correspondiente a cada hoja.

```
getLevel(index: number): number{
 return Math.floor(Math.log2(index + 1))
}
```

Figura 56: Función para calcular el nivel

```
<div class="class-level-{{getLevel($index)}} center-child wrapper-component">
    <app-familiy-leaf class="module center-child" [pigeon]="paloma" [index]="$index"></app-familiy-leaf>
</div>
```

Figura 57: Asignación de la clase dependiendo del nivel

Gracias a estas clases CSS se puede conseguir mostrar en pantalla una vista como la que se puede ver en la figura 54, permitiendo al usuario visualizar de una manera muy clara y visual el linaje de su paloma.

9.2.3 Guards

Los *guards* son funciones que permiten controlar el acceso a las rutas de la aplicación dependiendo de condiciones que se le indiquen. De esta manera se asegura que sólo aquellos usuario que cumplan determinadas condiciones tengan garantizado el acceso. Para esta aplicación se han creado los siguientes guards:

- **authGuard**: permite el paso a la ruta, si el usuario ha iniciado sesión. De lo contrario muestra un mensaje utilizando el servicio *SnackbarService* y redirige al usuario a la página de inicio de sesión.
- **loggedUserGuard**: permite el acceso a la a ruta si el usuario ha iniciado sesión y ha verificado su correo electronico. Si no se cumple alguna de las condiciones, se muestra

un mensaje al usuario y se le redirige a la página de verificación de correo electrónico o la página de inicio de sesión.

- **notLoggedGuard**: es el caso contrario de *authGuard*, permite el acceso si el usuario no ha iniciado sesión. Si el usuario tiene la sesión iniciada, será redigido a la página del listado de palomas. Este guard se utiliza para evitar que un usuario autenticado acceda a páginas como el registro de usuario o el de inicio de sesión.
- **onlyAdminGuard**: permite el acceso a la ruta solo si el usuario está autenticado y tiene el rol de administrador. Si no se cumple alguna de estas condiciones, el usuario es redirigido a la página de inicio de sesión o, si ya está autenticado, a la página de listado de palomas.

9.3 Despliegue de la Aplicación

Uno de los servicios que ofrece *Firebase* es *Firebase Hosting*, el cual permite subir el prototipo desarrollado a este *hosting* para que esté disponible y los usuarios puedan comenzar a utilizarlo. Para subir el proyecto al hosting, el primer paso es instalar las herramientas de Firebase, como se detalla en el Anexo I, lo que nos permite interactuar de manera sencilla con *Firebase*.

Para subir el proyecto primeramente hay que compilar el proyecto, con el siguiente comando:

```
ng build --prod
```

Tras la ejecución de este comando, el código escrito en TypeScript se convierte a JavaScript, generando una serie de ficheros minificados que contienen todo el código del proyecto.

A continuación, estos ficheros deben subirse al hosting configurado previamente al instalar las herramientas de *Firebase*. Este paso se realiza con el siguiente comando proporcionado por Firebase:

```
Firebase deploy
```

Una vez completada la subida de los archivos, se mostrará en la consola la URL donde se ha subido el proyecto. Para este proyecto, la URL es la siguiente: https://censo-palomar.web.app/

Capítulo 10: Conclusiones

Mediante la realización de este proyecto, se ha construido una plataforma web que cumple con los objetivos iniciales estipulados. Entre sus funcionalidades se permite al colombófilo llevar un registro de las palomas que posee, además de ofrecerle la posibilidad de incluir los resultados de las competiciones en las que participa.

Otra de las funcionalidades destacadas que se han implementado es la visualización del árbol genealógico de una paloma. Para que un colombófilo pueda decidir cómo realizar cruces entre palomas con el fin de obtener ejemplares de alta calidad para la competición, es necesario que disponga de toda la información posible. La visualización del árbol genealógico puede ayudar en este aspecto, ya que permite ver claramente la estirpe de la paloma.

Por otro lado, la visualización de las noticias permite a los colombófilos estar informados sobre eventos, sueltas o consejos útiles para el cuidado y entrenamiento de las palomas, aunque actualmente en esta sección no haya contenido real. Además, la sección de anuncios ofrece a los colombófilos un lugar donde pueden dar a conocer a los visitantes de la aplicación las palomas o material colombófilo que deseen vender, o incluso, por ejemplo, publicar un anuncio en el que busquen un nuevo reloj.

En lo personal, este proyecto me ha permitido poner a prueba todos los conocimientos y habilidades adquiridos durante la carrera. Ha sido una experiencia enriquecedora y un verdadero desafío poder enfrentarme al desarrollo completo de una aplicación. Aunque durante la realización del proyecto han surgido ciertos problemas, creo que he podido resolverlos de manera satisfactoria. Debido a mi inexperiencia, reconozco que tanto el código, como el diseño tienen margen de mejora.

Para concluir, considero firmemente que la herramienta desarrollada puede facilitar a los colombófilos tener un censo y llevar el control del pedigrí de sus palomas. Disponer de esta herramienta facilitará la toma de decisiones a la hora de gestionar su palomar.

10.1 Líneas Futuras

Aunque la plataforma web finalmente desarrollada cumple con los objetivos iniciales planteados, admite la incorporación de numerosas mejoras para hacer más completa su funcionalidad. A continuación, se enumeran algunas características que se pueden incorporar:

- Asignar más responsabilidades al administrador, y que éste no solo pueda listar los usuarios y publicar noticias, sino que verdaderamente pueda gestionar los usuarios (eliminarlos, crearlos, actualizarlos y borrarlos desde la aplicación. Actualmente estas tareas se pueden hacer desde la consola de *Firebase)*.
- Mensajes directos en la sección de anuncios. Esto permitiría que la comunicación con los usuarios que tienen anuncios se realice dentro de la plataforma, sin necesidad de utilizar métodos de comunicación alternativos.

- Comentarios en las noticias. Se podría añadir la posibilidad de que los usuarios registrados puedan realizar comentarios en las noticias.
- Diseño: aunque la maquetación utilizando Angular Material ha sido en general bastante buena, en ocasiones ha sido algo limitante, además de que muchos de los estilos son bastante clásicos. En un futuro se podrían utilizar otros *frameworks* de CSS como *Tailwind*.
- Buscador: los colombófilos suelen tener un elevado número de palomas, por lo que, si quieren consultar un registro concreto, sería mucho más sencillo si dispusieran de un buscador que permitiera numerosas posibilidades de filtrado.
- Información sobre las parejas y los hijos: otra de las mejoras sería la de disponer de un árbol genealógico, pero al revés, para mostrar los descendientes en lugar de los ascendientes, además de mostrar la información de la pareja de la paloma.

Bibliografía

[1] Real Academia Española, "*colombofilia*", [En línea] Disponible en: https://www.rae.es/diccionario-estudiante/colombofilia [Accedido en 11/07/2024]

[2] I. Rosario, S. Déniz, F. Real, F. Acosta, D. Padilla, y B. Acosta, "*La Colombofilia y Canarias*," *Revista Canaria de las Ciencias Veterinarias*, Las Palmas de Gran Canaria: Servicio de Publicaciones, Universidad de Las Palmas de Gran Canaria, vol. 6-7, pp. 66, 2009. [En línea]. Disponible en: http://hdl.handle.net/10553/12402. [Accedido en 11/07/2024].

[3] Federación Canaria de Colombofilia, (s.f.) "*Clubes inscritos a la Federación Canaria de Colombofilia*" [En línea] Disponible en: http://www.fedcancol.es/club.html [Accedido en 11/07/2024]

[4] Subdirección General de Estadística y Estudios de la Secretaría General Técnica del Ministerio de Educación, Formación Profesional y Deportes, "Anuario de Estadísticas Deportivas 2024" [En línea] Disponible en: https://www.educacionfpydeportes.gob.es/dam/jcr:fbf05df0-5e3f-4b57-9d5b-6588d4ad34a9/aed-2024.pdf [Accedido en 11/07/2024]

[5] Universidad de Las Palmas de Gran Canaria (2019, mar. 05) "*Memoria del Plan de Estudios*" [En línea] Disponible en: https://www.eii.ulpgc.es/sites/default/files/2022-05/Grado%20en%20Ingenier%C3%ADa%20Inform%C3%A1tica%20-%20memoria%20del%20plan%20de%20estudios%20-%202019.pdf [Accedido 08/07/2024]

[6] Wikipedia "*Desarrollo iterativo y creciente*" [En línea] Disponible en: https://es.wikipedia.org/wiki/Desarrollo_iterativo_y_creciente [Accedido en 12/07/2024]

[7] Wikipedia, "*Desarrollo en cascada*" [En línea] Disponible en: https://es.wikipedia.org/wiki/Desarrollo_en_cascada [Accedido en 12/07/2024]

[8] J.M Beas, (2019, sept. 22) "*Desarrollo iterativo e incremental*" [En línea] Disponible en: https://blog.jmbeas.es/2019/09/22/desarrollo-iterativo-e-incremental/ cascada [Accedido en 12/07/2024]

[9] Wikipedia, "*Reglamento General de Protección de Datos*", [En línea] Disponible en: https://es.wikipedia.org/wiki/Reglamento_General_de_Protecci%C3%B3n_de_Datos [Accedido en 10/07/2024]

[10] Wikipedia, "*Reglamento General de Protección de Datos*", [En línea] Disponible en: https://es.wikipedia.org/wiki/Ley_Org%C3%A1nica_de_Protecci%C3%B3n_de_Datos_Personales_y_garant%C3%ADa_de_los_derechos_digitales [Accedido en 10/07/2024]

[11] Wikipedia, "*HTML*", [En línea] Disponible en: https://es.wikipedia.org/wiki/HTML [Accedido en 09/07/2024]

[12] J. Flores, (2015, ago. 25) "*¿Qué es HTML?*", Código Facilito, [En línea] Disponible en: https://codigofacilito.com/articulos/que-es-html [Accedido en 09/07/2024]

[13] Mozilla.org, (s.f.) "*Referencia de Atributos HTML*" [En línea] Disponible en: https://developer.mozilla.org/es/docs/Web/HTML/Attributes [Accedido en 09/07/2024]

[14] Wikipedia, "*CSS*", [En línea] Disponible en: https://es.wikipedia.org/wiki/CSS

[Accedido en 09/07/2024]

[15] Mozilla.org, (s.f.) "*CSS*", [En línea] Disponible en: https://developer.mozilla.org/es/docs/Web/CSS

[Accedido en 09/07/2024]

[16] J Román, (s.f) "*¿Qué es CSS?*", [En línea] Disponible en: https://lenguajecss.com/css/introduccion/que-es-css/ [Accedido en 09/07/2024]

[17] Onetomarket.es, (s.f) "*Diferencia entre css y scss*", [En línea] Disponible en: https://onetomarket.es/diseno/diferencia-entre-css-y-scss/ [Accedido en 09/07/2024]

[18] Wikipedia, "*Sass*", [En línea] Disponible en: https://es.wikipedia.org/wiki/Sass [Accedido en 09/07/2024]

[19] Wikipedia, "*JavasScript*", [En línea] Disponible en: https://es.wikipedia.org/wiki/JavaScript [Accedido en 09/07/2024]

[20] Wikipedia, "*Typescript*", [En línea] Disponible en: https://es.wikipedia.org/wiki/TypeScript [Accedido en 09/07/2024]

[21] S. R. Chivukula y A. Iskandar, "*Web Development with Angular and Bootstrap: Embrace responsive web design and build adaptive Angular web applications*", 3ª Edición, Birmingham, Packt Publishing, 2019.

[22] Wikipedia, "*Node.js*", [En línea] Disponible en https://es.wikipedia.org/wiki/Node.js [Accedido en 13/07/2024]

[23] Npm Docs, "*About npm*" [En línea] Disponible en: https://docs.npmjs.com/about-npm [Accedido en 09/07/2024]

[24] Wikipedia, "*npm*", [En línea] Disponible en https://es.wikipedia.org/wiki/Npm [Accedido en 09/07/2024]

[25] Wikipedia, "*Angular (framework)",* [En línea] Disponible en: https://es.wikipedia.org/wiki/Angular_(framework) [Accedido en 03/07/2024]

[26] E. Oriol, (2017, mar. 17) "*Actualizada - Introducción a Angular (parte I): Modulo, Componente, Template y Metadatos*" [En Línea] Disponible en: http://blog.enriqueoriol.com/2017/03/introduccion-angular-modulo-y-componente.html [Accedido en 09/07/2023]

[27] Angular, (s.f.) "*Introducción a los conceptos de Angular*", [En línea] Disponible en: https://docs.angular.lat/guide/architecture [Accedido en 03/07/2024].

[28] Angular, (s.f.) "*Visión general del CLI y referencia de comandos*" [En línea] Disponible en: https://docs.angular.lat/cli [Accedido en 03/07/2024]

[29] J.C. Fatjó, (2021, ener. 19) "*Introducción a Angular Material*" [En Línea] Disponible en: https://tech.tribalyte.eu/blog-introduccion-angular-material [Accedido en 03/07/2024]

[30] Wikipedia, "*Firebase*", [En línea] Disponible en: https://es.wikipedia.org/wiki/Firebase [Accedido en 03/07/2024]

[31] Firebase, (s.f.) "*Mejora tu app al máximo*", [En línea] Disponible en: https://firebase.google.com/ [Accedido en 09/07/2024]

[32] RxJS, (s.f.) "*RxJs Introduction*" [En línea] Disponible en: https://rxjs.dev/guide/overview [Accedido en 11/07/2024]

[33] Wikipedia, "*Git*", [En línea] Disponible en: https://es.wikipedia.org/wiki/Git [Accedido en 09/07/2024]

[34] Git, (s.f) "*Inicio - Sobre el Control de Versiones - Acerca del Control de Versiones*" [En línea] Disponible en: https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Acerca-del-Control-de-Versiones [Accedido en 08/07/2024]

[35] Wikipedia, "*GitHub*" [En línea] Disponible en: https://en.wikipedia.org/wiki/GitHub [Accedido en 09/07/2024]

[36] Microsoft, (s.f) "*Conozca la familia Visual Studio*", [En línea] Disponible en: https://visualstudio.microsoft.com/es/ [Accedido en 08/07/2024]

[37] Wikipedia, "*Visual Studio Code*", [En línea] Disponible en: https://es.wikipedia.org/wiki/Visual_Studio_Code [Accedido en 08/07/2024]

[38] Trello, (2023, ago. 8) "*Qué es Trello: descubre sus funciones, usos y todo lo que ofrece*", [En línea] Disponible en: https://trello.com/es/tour [Accedido en 08/07/2024]

[39] Wikipedia, "*Figma*", [En línea] Disponible en: https://es.wikipedia.org/wiki/Figma [Accedido en 03/07/2024]

[40] Wikipedia, "*Inkscape*" [En línea] Disponible en: https://es.wikipedia.org/wiki/Inkscape [Accedido en 09/07/2024]

[41] Figma, (s.f) "*What is a use case? How to write one, examples, + template*" [En línea] Disponible en: https://www.figma.com/es-es/resource-library/what-is-a-use-case/ [Accedido en 13/07/2024]

[42] Wikipedia, "Casos de uso", [en línea] Disponible en: https://es.wikipedia.org/wiki/Caso_de_uso [Accedido en 12/07/2024]

[43] Wikipedia, "*Requisito Funcional*" [En línea] Disponible en: https://es.wikipedia.org/wiki/Requisito_funcional [Accedido en 12/07/2024]

[44] Wikipedia, "*Requisito no funcional*" [En línea] Disponible en: https://es.wikipedia.org/wiki/Requisito_no_funcional [Accedido en 12/07/2024]

[45] Visure, (s.f) “Requisitos funcionales vs no funcionales” [En línea] Disponible en: https://visuresolutions.com/es/requirements-management-traceability-guide/functional-vs-non-functional-requirements/ [Accedido en 12/07/2024]

[46] Angular, (2023, oct. 24) “*Workspace and project file structure*”. [En línea] Disponible en: https://v17.angular.io/guide/file-structure [Accedido en 16/07/2024]

[47] Angular, (s.f) “*Introducción a servicios e inyección de dependencias*”. [En línea] Disponible en: https://docs.angular.lat/guide/architecture-services [Accedido en 17/07/2024]

[48] Angular, (2023, ago. 14) “*What is a pipe*”. [En línea] Disponible en: https://v17.angular.io/guide/pipes-overview [Accedido en 17/07/2024]

[49] Wikipedia, “T*ree (Data Structure)*” [En línea] Disponible en: https://en.wikipedia.org/wiki/Tree_(data_structure) [Accedido en 20/07/2024]

[50] Wikipedia, “*Binary tree*” [En línea] Disponible en: https://en.wikipedia.org/wiki/Binary_tree#Arrays [Accedido en 20/07/2024]

[51] SASS, (s.f) “*@function*” [En línea] Disponible en: https://sass-lang.com/documentation/at-rules/function/ [Accedido en 20/07/2024]

[52] SASS, (s.f) “*@for*” [En línea] Disponible en: https://sass-lang.com/documentation/at-rules/control/for/ [Accedido en 20/07/2024]

[53] Angular, (s.f) “*CLI Overview and Command Reference*”., [En línea] Disponible en: https://github.com/angular/angular-cli [Accedido en 17/05/2024]

[54] Angular, (s.f) “Getting Started with Angular Material”. [En línea] Disponible en: https://material.angular.io/guide/getting-started [Accedido en 17/05/2024]

[55] C. Axtmann, (2023, dic. 18) “Adding Firebase to Angular 17”, [En línea] Disponible en: https://medium.com/@christianaxtmn/adding-firebase-to-angular-17-5c5a7cf4aba8 [Accedido en 17/05/2024]

ANEXO I: Preparación del Entorno

En esta sección se detallará cómo se realizó la preparación del entorno necesario para iniciar el desarrollo de este proyecto. Se describirán los pasos llevados a cabo para instalar los *frameworks*, librerías y herramientas en un entorno con sistema operativo *Windows*.

Instalación de Git

Al tener un equipo con el sistema operativo *Windows*, sólo es necesaria la descarga del instalador desde la siguiente URL y ejecutarlo:

> https://git-scm.com/download/win

Hay que seguir las indicaciones del asistente de instalación para configurarlo según las necesidades que se requieran.

Finalmente, una vez termine el asistente de instalación, hay que establecer el nombre de usuario y el correo mediante los siguientes comandos, que se ejecutarán en el "*GIT Bash*", que fue añadido durante la instalación:

```
git config –global user.name "colombófilo"
git config –global user.email "appcensopalomar@gmail.com"
```

Una vez realizados estos pasos, *Git* queda instalado y configurado. La versión de *Git* instalada en este proceso ha sido la 2.45.0.

Instalación de Visual Studio Code

Al igual que se hizo con *Git*, basta con descargar el instalador y seguir los pasos del asistente de instalación. En esta ocasión, el instalador se descarga desde la siguiente dirección:

```
https://code.visualstudio.com/Download
```

Una vez descargado, se ejecuta el instalador y se siguen las instrucciones eligiendo las opciones deseadas, para configurar el editor de código.

Para este proyecto se ha instalado la versión 1.89.1 de *Visual Studio Code*.

Instalación de Node.js

La instalación de Node.js sigue un proceso similar a los casos de *Git* y *Visual Studio Code*. Para comenzar, es necesario descargar el instalador de la página oficial en la siguiente dirección:

```
https://nodejs.org/en/download/
```

Una vez descargado el fichero de instalación, al ejecutarlo se iniciará el asistente que permitirá seleccionar las diferentes opciones de configuración para personalizar la instalación.

Para la realización de este proyecto se ha instalado la versión 20.13.1 de Node.js. Con la instalación de Node.js también se incluye el administrador de paquetes *Node Package Manager* (*NPM*); en este caso, se instaló la versión 10.8.0.

Para comprobar las versiones instaladas, se pueden usar los siguientes comandos:

> ***Node.js***: `node -v`
> ***NPM***: `npm -v`

Instalación de Angular

El siguiente paso que se ha realizado para preparar el entorno es instalar el CLI (*Command Line Interface*) de Angular. Angular CLI es una herramienta que nos permite crear y gestionar aplicaciones Angular mediante comandos en la terminal [53]. A diferencia de los procedimientos anteriores, esta instalación no se realiza descargando un instalador, sino utilizando *NPM*. Para este paso, se abre una terminal de Windows y se ejecuta el siguiente comando:

```
npm install -g @angular/cli
```

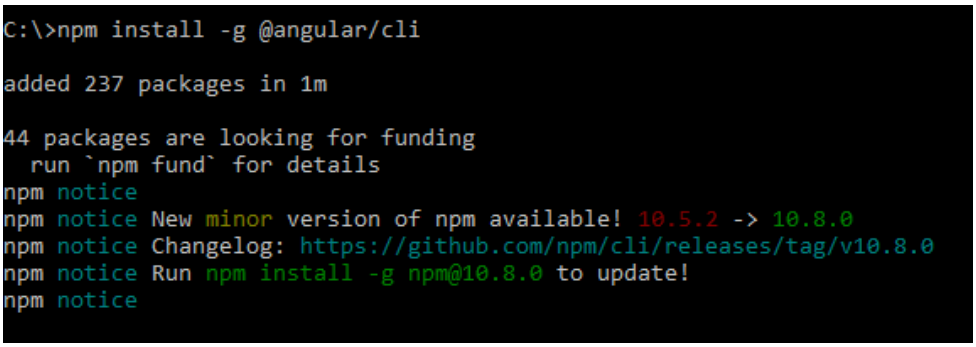

Figura A1.1: Instalación de Angular CLI

El resultado de este comando se puede ver en la figura A1.1. Este comando instala de manera global angular CLI en el equipo. En este caso, la versión instalada es la 17.1.1.

Creación del proyecto Angular

Para continuar con la instalación de los paquetes necesarios para desarrollar este proyecto, es fundamental crear la aplicación de Angular sobre la que se trabajará. El primer paso consiste en dirigirnos a la carpeta donde se va a guardar el proyecto. Una vez en la carpeta se ejecutará el comando:

```
ng new nombre del proyecto
```

En este caso concreto el comando utilizado fue:

```
ng new censo_palomar
```

Figura A1.2: Creación del proyecto de Angular

Tras la ejecución de este comando, se mostrarán opciones para elegir el tipo de formato de hojas de estilo a instalar, tal y como se muestra en la figura A1.2. En este caso particular, se ha optado por *SaSS*. Para verificar que la instalación ha sido exitosa y que el proyecto funciona correctamente, se puede ejecutar el comando

```
ng serve -o
```

Al lanzar este comando, se abrirá un navegador con la página por defecto que viene configurada cuando se crea un nuevo proyecto de angular.

Instalación de Angular Material

Para incorporar el paquete de *Angular Material* al proyecto, simplemente se ejecuta el siguiente comando desde una terminal:

```
ng add @angular/material
```

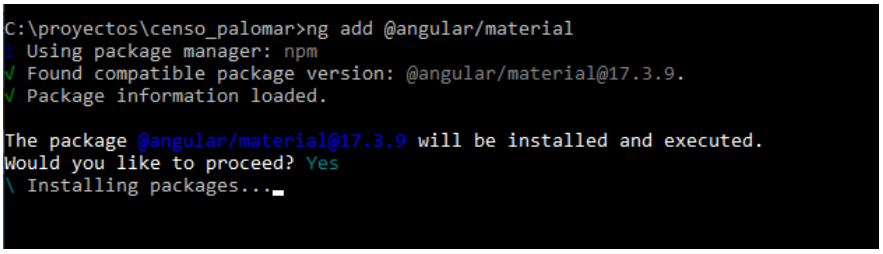

Figura A1.3: Instalación de Angular Material

Durante el proceso de instalación, el asistente de configuración permite elegir qué paleta de colores se desea utilizar y si se desea instalar "*Angular Animations*", como se muestra en la figura A1.3. Este paquete proporciona herramientas que facilitan la creación de animaciones en los elementos de una aplicación. En este caso no se ha instalado este paquete.

Además, con la instalación de Angular Material se incluye el *Component Dev Kit (CDK),* que ofrece un conjunto de componentes ampliamente utilizados en el desarrollo de aplicaciones web como acordeones, Cuadros de dialogo, tablas, entre muchos otros.

La versión instalada de Angular material en este caso has sido la 17.3.9.

Creación del proyecto en Firebase

El siguiente paso para preparar el entorno es configurar Firebase, del cual se utilizarán varios de los servicios que ofrece. Para poder utilizarlo, es imprescindible crear un proyecto en su

consola. Por lo tanto, es necesario iniciar sesión en la consola de *Firebase* en la siguiente dirección:

> https://console.firebase.google.com

Es importante destacar que para llevar a cabo este proceso es necesario contar con una cuenta de *Google*. En la página principal de Firebase, se debe seleccionar el botón "*Create a Project*". Esto conducirá a una página en la que se podrá empezar a configurar el proyecto, como se muestra en la figura A1.4. El primer paso es asignar un nombre al proyecto que en este caso es "*censo-palomar*".

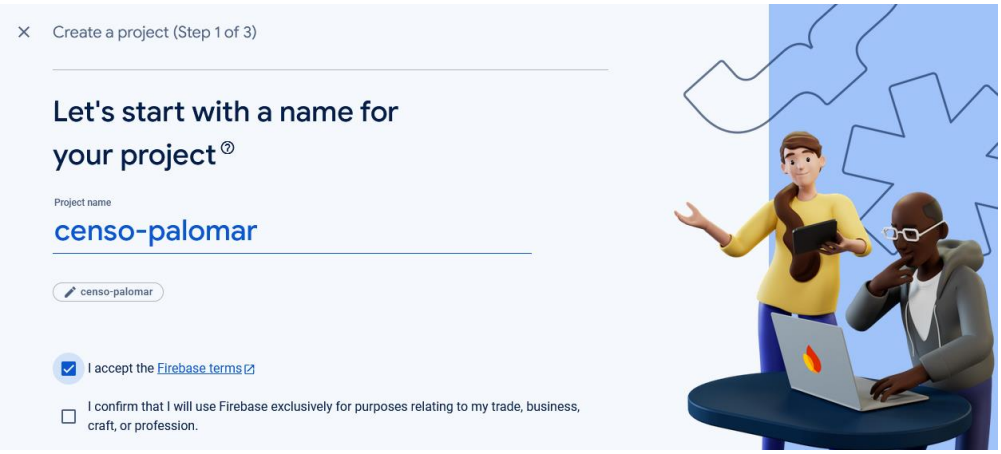

Figura A1.4: Creación del proyecto de Firebase

En el siguiente paso, se pregunta si se quiere utilizar *Google Analytics*. Para este proyecto no se utilizará por el momento; en caso de que fuera necesario, se podrá activar más adelante. Finalmente, al seleccionar "C*reate Project*", se creará el proyecto.

Una vez creado el proyecto, hay que añadirlo a una aplicación. En este caso, se trata de una aplicación web, por lo que seleccionaremos el botón que pone "*Web*". En la pantalla siguiente, que es la que se muestra en la figura A1.5, se introduce el nombre que tendrá la aplicación, que en este caso es el mismo que el del proyecto, para evitar confusiones de nombres con otros proyectos. También se puede establecer si configuramos el hosting de *Firebase*. En este caso, sí se selecciona, ya que en el futuro se irá subiendo a este hosting el proyecto. Si no se selecciona ahora, se podrá configurar más adelante sin ningún problema.

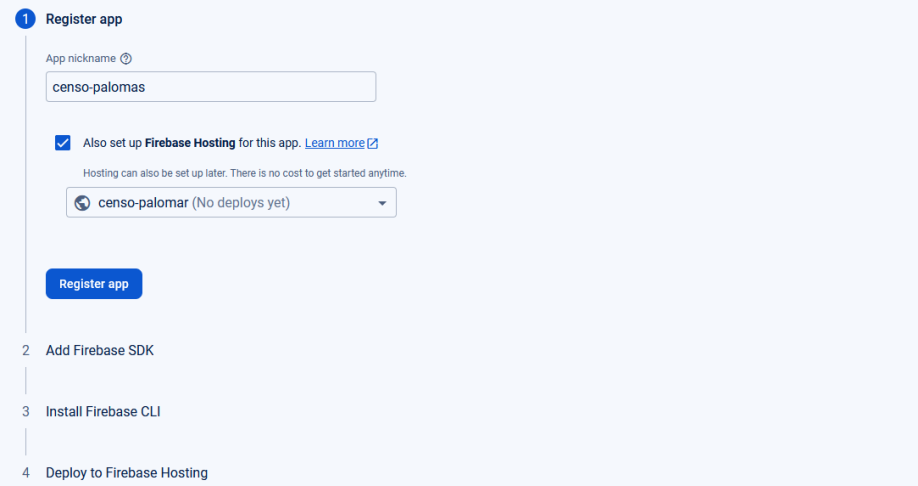

Figura A1.5: Adición de Firebase a la aplicación web

En la siguiente pantalla, se muestran los datos de configuración del proyecto. Esta información será utilizada en el futuro para configurar *Firebase*. Los siguientes pasos del asistente son indicaciones proporcionadas por *Firebase* para completar la configuración de la aplicación, lo cual se detallará en el siguiente apartado.

Para finalizar, es necesario activar y configurar los servicios que utilizarán en la aplicación. Estos servicios son "*Cloud Firestore*", "*Firebase Authentication*" y "*Cloud Storage*". Para activarlos, simplemente hay que buscar estos servicios en el menú lateral de la consola de *Firebase* y seleccionar en cada uno de los servicios el botón "*Comenzar*". Luego, seguir los pasos de cada uno de los asistentes, eligiendo las configuraciones deseadas.

Instalación de Firebase

Finalmente, ahora falta añadir al proyecto de *Angular* los paquetes necesarios para manejar las conexiones de la aplicación con *Firebase*. El primer paso es instalar *Firebase CLI*, un paquete que proporciona una serie de herramientas que permite gestionar los proyectos de *Firebase*. Para instalar este paquete, se ejecutará el siguiente comando en una terminal:

```
npm install -g firebase-tools
```

Una vez instalado el paquete, se puede iniciar sesión usando el siguiente comando:

```
firebase login
```

Una vez la sesión esté iniciada, se pueden utilizar los comandos de *Firebase* para gestionar los proyectos creados en esa cuenta.

En las últimas versiones de Angular, al crear un proyecto, ya no se generan por defecto los ficheros de entorno. Estos ficheros almacenan la configuración y las credenciales necesarias para comunicar la aplicación con *Firebase*. Para generar estos ficheros, se utiliza el siguiente comando:

```
ng generate environments
```

Este comando creará dos archivos, uno para la configuración del entorno de producción y otra para el entorno de desarrollo.

El siguiente paso es instalar los paquetes de *Firebase* con el siguiente comando:

```
ng add @angular/fire
```

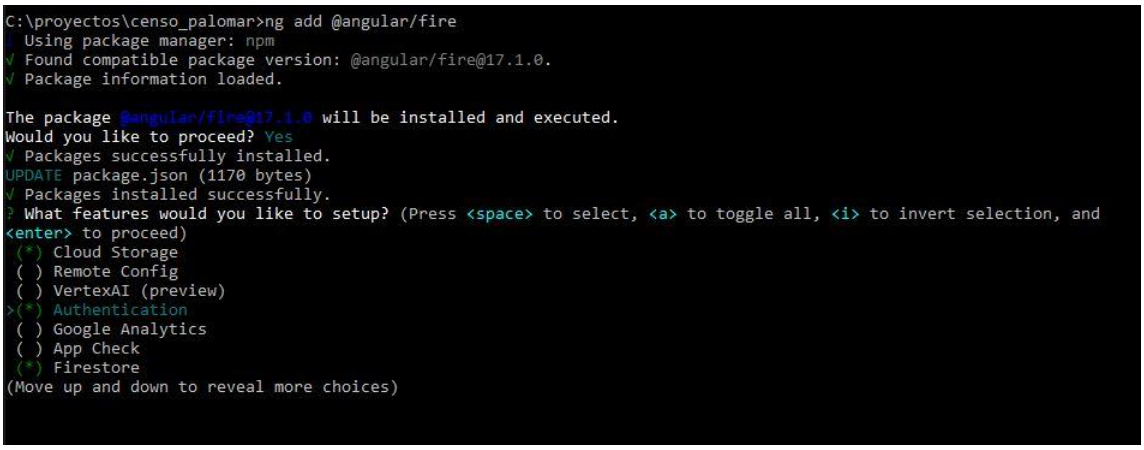

Figura A1.6: Instalación del paquete @angular/fire

Durante la ejecución de este comando, se mostrará una serie de opciones para elegir qué servicios de *Firebase* se desean instalar, como se muestra en la figura A1.6. Para esta aplicación, se han instalado los servicios *Firestore, Authentication y Cloud Storage*. Además, en este punto de la instalación, es necesario iniciar sesión con la cuenta de *Google* utilizada previamente para crear el proyecto de *Firebase*. Esto permitirá enlazar correctamente la aplicación con *Firebase*.

Una vez instalado el paquete de *Firebase*, es el momento de configurarlo, en el fichero `environments.ts` colocaremos la información de configuración que se generó cuando se creó el proyecto de *Firebase* que será similar al siguiente código:

```
export const environment = {
  firebase: {
    apiKey: " AIzaSyC6SVehePkqE-05Hl7bbgNzOVbButhvYno",
    authDomain: " censo-palomar.firebaseapp.com",
    projectId: " censo-palomar",
    storageBucket: "censo-palomar.appspot.com",
    messagingSenderId: " 664603530529",
    appId: " 1:664603530529:web:471b41db26e1b883a11a60"
  }
};
```

El último paso para terminar la configuración de *Firebase* es inyectar el servicio para que pueda ser utilizado en la aplicación. Para esto, en el archivo `app.config.ts` colocaremos el código que se muestra en la figura A1.7.

```
export const appConfig: ApplicationConfig = {
    providers: [
      provideRouter(routes),
      provideAnimationsAsync(),
      provideFirebaseApp(()=>initializeApp(environment.firebaseConfig)),
      provideAuth(() => getAuth()),
      provideFirestore(() => getFirestore()),
      provideStorage(() => getStorage())
    ]}
```

Figura A1.7: Configuración de Firebase, fichero app.config.ts

Llegados a este punto *Firebase* ya está configurado y el entorno de desarrollo está listo para comenzar con el desarrollo.

ANEXO II: Manual de Usuario

El propósito de este manual es orientar al usuario para que pueda sacar el máximo provecho de la aplicación. En esta guía se explicarán detalladamente las diversas funcionalidades disponibles, facilitando así el uso de las características que ofrece la aplicación. Todas las imágenes utilizadas se han tomado desde la versión móvil, pero son extrapolables a la versión de escritorio.

A2.1 Página de Inicio

Esta es la primera pantalla que se muestra al entrar en la aplicación. En ella se encuentra una descripción general, así como las últimas noticias y anuncios publicados. En la figura A2.1, se pueden ver varias capturas de diferentes secciones de la página de inicio.

Figura A2.1: Página de Inicio

A2.2 Menú de navegación

En la parte superior derecha de la aplicación se muestra el icono "*Hamburguer*" para mostrar u ocultar el menú de navegación. Las opciones del menú son las que se muestran a continuación:

A2.2.1 Si el usuario no ha iniciado sesión

Figura A2.2: Menú visible por usuarios sin la sesión iniciada

Si el usuario no está autenticado verá un menú como el de la figura A2.2 y verá las siguientes opciones:

Inicio: para navegar a la página de inicio.

Iniciar Sesión: conduce a la página donde el usuario puede autenticarse.

Registrarse: conduce a la página donde el usuario puede crear una cuenta en la aplicación y empezar a beneficiarse de las ventajas de la misma.

Noticias: conduce a la página con las noticias y demás información relevante sobre colombofilia publicadas en la aplicación.

Palomas en venta: conduce a la página donde el usuario podrá ver los anuncios publicados por otros usuarios.

A2.2.2 Si el usuario ha iniciado sesión

Figura A2.3: Menú visible para los usuarios autenticados

Si el usuario está autenticado, verá un menú como el de la figura A2.3 con las siguientes opciones

Inicio: para navegar a la página de inicio.

Perfil: conduce a la página donde el usuario puede ver su información de usuario.

Mis palomas: conduce a la página donde el usuario puede ver el censo de sus palomas.

Registrar Paloma: conduce a la página donde el usuario puede crear el registro de una paloma.

Noticias: conduce a la página con las noticias y demás información relevante sobre colombofilia publicadas en la aplicación.

Publicar Anuncio: al seleccionar esta opción, el usuario es llevado a la página para publicar anuncios.

Palomas en venta: conduce a la página donde el usuario podrá ver los anuncios publicados por otros usuarios.

Ver mis anuncios: conduce a una página donde el usuario podrá gestionar los anuncios que ha publicado.

Cerrar Sesión: al seleccionar esta opción el usuario deja de estar autenticado.

A2.2.3 Menú para los usuarios administradores

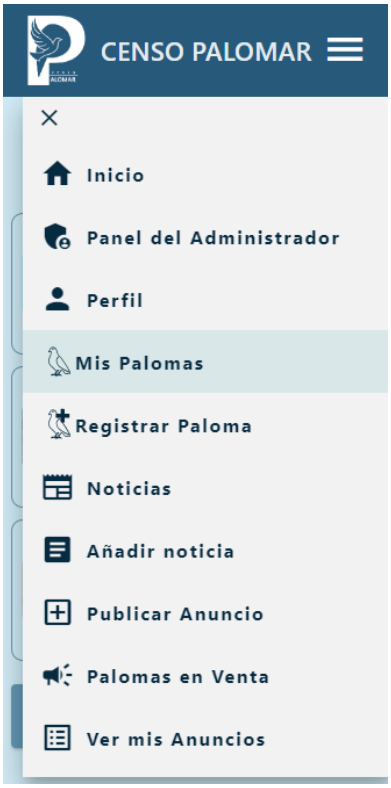

Figura A2.4: Menú visible para los usuarios administradores

Si el usuario está autenticado y tiene el rol de administrador, verá un menú como el de la figura A2.4. Tendrá las mismas opciones que en el caso anterior y además tendrá:

Panel del Administrador: lleva al administrador a una página que engloba todas las acciones que puede realizar.

Añadir noticia: conduce al administrador a la página donde puede publicar una nueva noticia.

A2.3 Inicio de sesión

Para iniciar sesión hay que seleccionar en el menú la opción *Iniciar Sesión* donde se mostrará una pantalla como la que muestra la figura A2.6.

Se introducen las credenciales en el formulario y si son correctas se podrá ver el listado de palomas. En el caso de que las credenciales no sean correctas, o surja algún error se mostrarán mensajes como el de la figura A2.5.

Figura A2.5: Mensajes de error

En el caso de que los datos introducidos no tengan el formato correcto se mostrarán avisos como los que se muestran en la figura A2.7.

Figura A2.6: Formulario de inicio de sesión

Figura A2.7: Avisos de datos incorrectos

A2.4 Registro de usuario

Para acceder al registro simplemente hay que dirigirse al menú y seleccionar la opción "Registrarse". Entonces el usuario verá una pantalla como la de la figura A2.8 que tendrá que cumplimentar para realizar el registro.

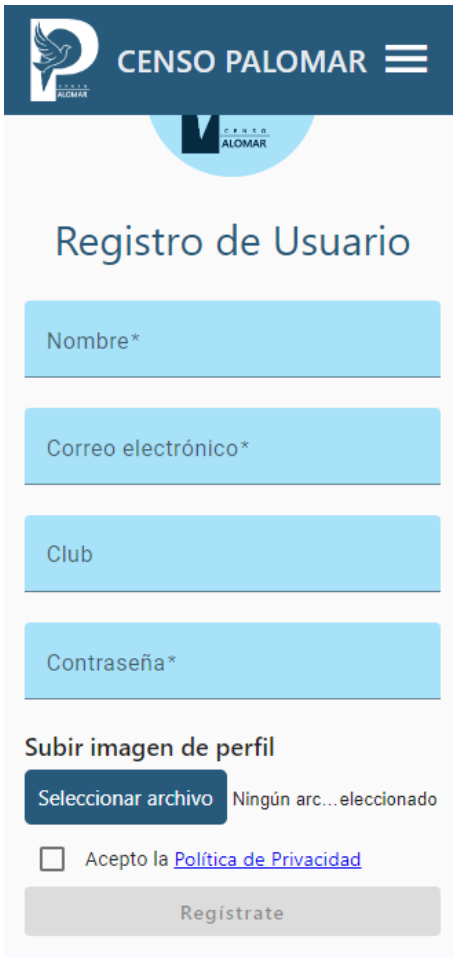

Figura A2.8: Formulario de registro

Figura A2.9: Formulario de registro con advertencias

Al igual que en el caso del inicio de sesión, si los datos no son correctos, aparecerán advertencias para ayudar a corregir los errores. Además, el botón de registrarse permanecerá desactivado hasta que los datos sean correctos.

Una vez realizado el registro, el usuario recibirá un correo similar al de la figura A2.10 y será redirigido a la página de verificación de correo. No podrá navegar por las partes privadas de la aplicación hasta que valide su correo. Si lo intenta, siempre será redirigido a la misma página de verificación de correo, similar a la figura A2.11.

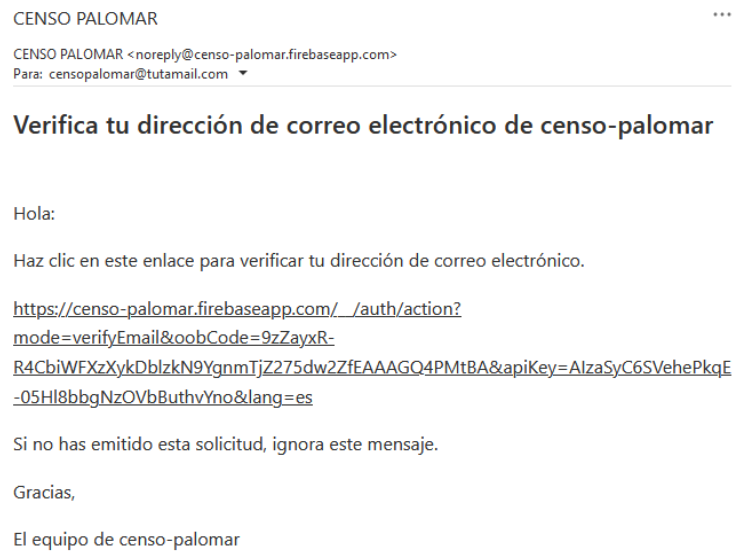

Figura A2.10: Correo de verificación de email

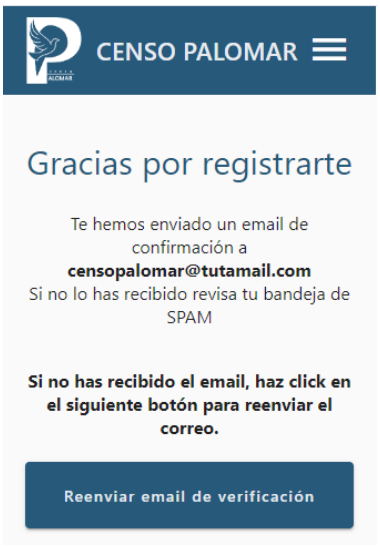

Figura A2.11: Página de verificación de correo electrónico

A2.5 Listado de palomas

Es la página principal del usuario registrado. Cuando se inicia sesión, el usuario es redirigido a esta página, que será similar a la figura A2.12. Desde esta página podrá ver todas las palomas que tiene registradas. Si hace clic sobre alguno de los registros, verá el detalle completo. En el caso de que no tenga ninguna paloma, el usuario verá una pantalla como la que se muestra en la figura A2.13.

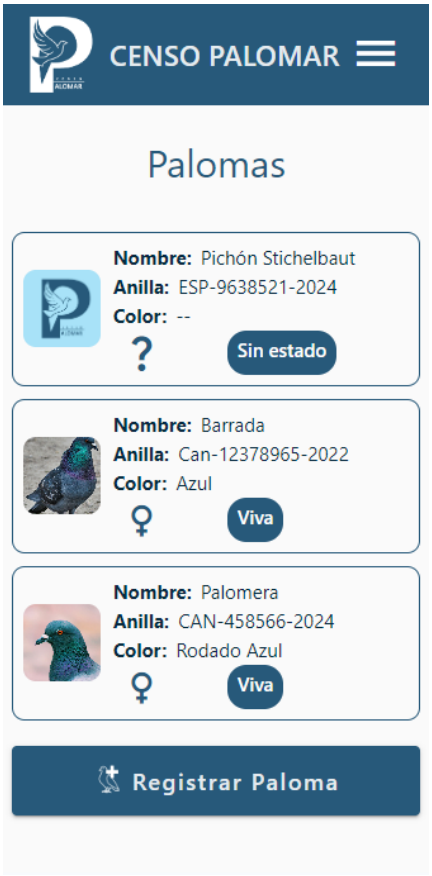

Figura A2.12: Listado de palomas

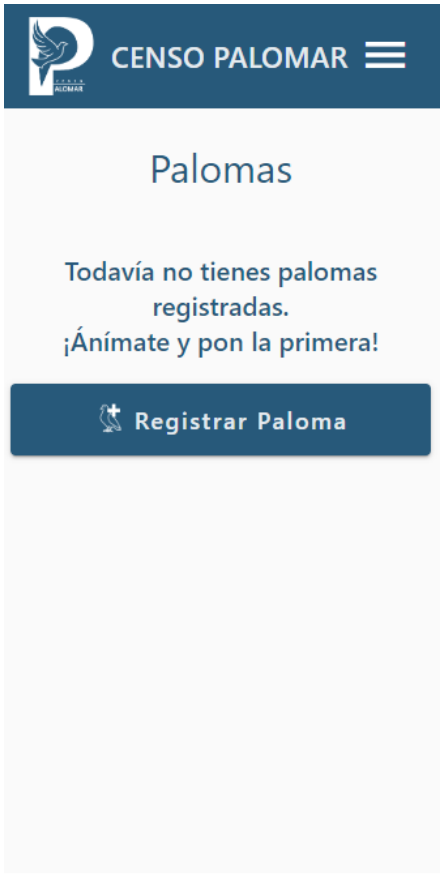

Figura A2.13: Listado de palomas vacío

A2.6 Gestión de registros palomas

En este apartado se explicará cómo se realiza el registro, la modificación y la eliminación de un registro de una paloma.

A2.6.1 Registro de una paloma

Para realizar el registro de una paloma basta con dirigirse al menú y seleccionar “Registrar paloma”, o alternativamente, ir al final del listado de palomas donde se puede encontrar un botón que también llevará a la página de registro de una paloma.

En las figuras A2.14 y A2.15 se puede ver el formulario completo (dividido en dos pantallas) que hay que cumplimentar para realizar el registro de una paloma. Los campos obligatorios a rellenar son el nombre de la paloma y el número de anilla.

En el caso de que la paloma no tenga nombre, como sugerencia, puede volver a colocar el número de la anilla, o algún dato representativo con el que se puede identificar a la paloma.

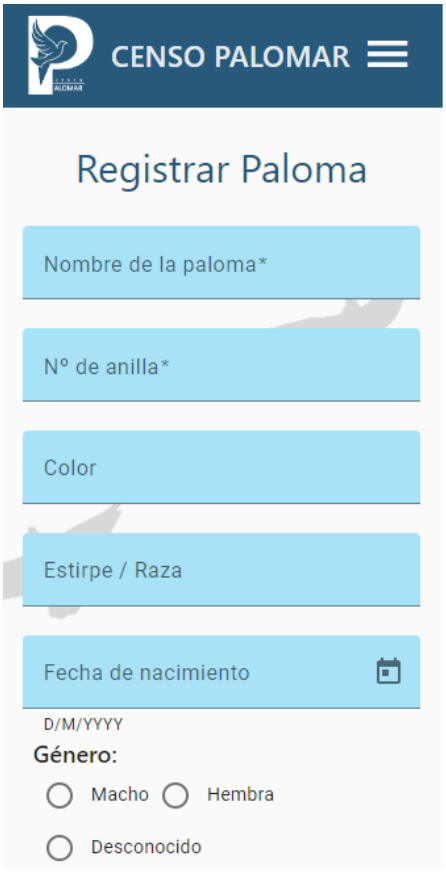

Figura A2.14: Formulario para registrar una paloma (parte 1)

Figura A2.15: Formulario para registrar una paloma (parte 2)

Selección del estado de una paloma

La aplicación tiene incorporados los estados más frecuentes de una paloma, como se muestra en la figura A2.16. Sin embargo, en el mismo campo se puede escribir un estado personalizado que no esté incluido en la lista. En la figura A2.17 se muestra un ejemplo de estado personalizado por el usuario.

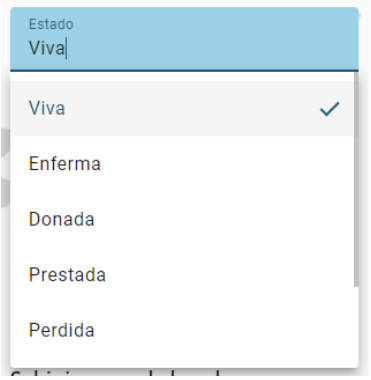

Figura A2.16: Selector de estados

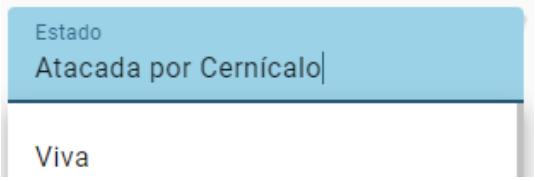

Figura A2.17: Estado personalizado

Selección de los padres de una paloma

Para establecer a los padres de una paloma se dispone de las siguientes posibilidades:

Los padres no están registrados en la aplicación

Entonces se desmarca la opción de “¿La madre ya está registrada?” o “¿El padre ya está registrado?” según sea el caso y se puede escribir el nombre o el dato identificativo de la paloma. En la figura A2.18 se muestra un ejemplo de cómo se cumplimenta este campo.

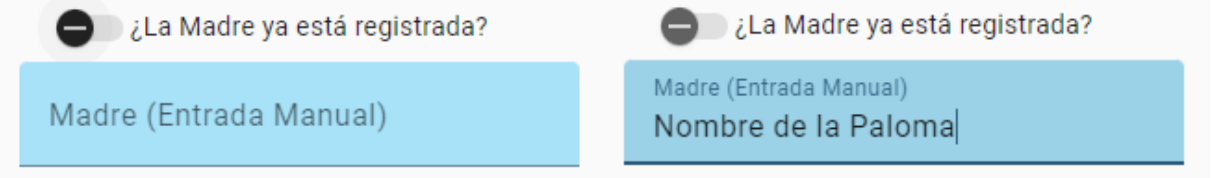

Figura A2.18: Introducción manual de los padres

Los padres ya están registrados en la aplicación

Si los padres ya están registrados, se marca la opción de “¿La madre ya está registrada?” o “¿El padre ya está registrado?” según sea el caso y entonces se obtendrá un desplegable desde el cual se puede seleccionar a la paloma. En la figura A2.19 se muestra un ejemplo del desplegable que aparece en este caso.

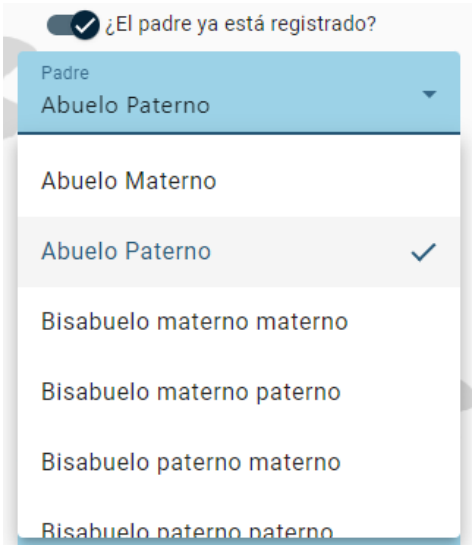

Figura A2.19: Selección de los padres de una paloma

A2.6.2 Perfil de una paloma

Desde el listado de una paloma, si se selecciona un registro, se puede obtener toda la información registrada de esa paloma concreta. La pantalla se verá aproximadamente como se muestra en las figuras A2.20 y A2.21.

Como se puede observar, la información se muestra organizada en cuatro grandes bloques:

- Información de la paloma
- Información de los padres y su genealogía
- Información Adicional
- Información de las competiciones

Figura A2.20: Perfil de una paloma (parte 1)

Figura A2.21: Perfil de una paloma (parte 2)

A2.6.3 Modificación del registro de una paloma

Para modificar el registro de una paloma, hay que dirigirse al perfil de la paloma y en la barra de herramientas inferior seleccionar la opción “Editar”. En ese momento, el usuario será redirigido a un formulario similar al de la creación del registro (ver figuras A2.14 y A2.15), pero relleno con la información de la paloma. El usuario puede modificar los datos que desee y, al seleccionar el botón “Actualizar Paloma”, se harán efectivos los cambios.

A2.6.4 Eliminación del registro de una paloma

Para eliminar un registro de una paloma, basta con dirigirse al perfil de la paloma y en la barra de herramientas inferior seleccionar la opción “Eliminar”. En ese momento, aparecerá una ventana emergente para confirmar la eliminación del registro. En la figura A2.22 se puede ver la ventana emergente. Es importante indicar que, si se elimina el registro de una paloma, las competiciones asociadas a esa paloma también se eliminarán.

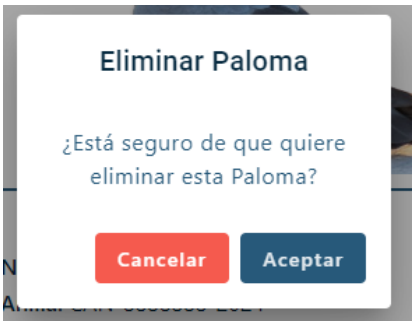

Figura A2.22: Mensaje emergente para la eliminación del registro de una paloma

A2.7 Árbol Genealógico

Desde el perfil de una paloma, debajo de la información de los padres, se encuentra un botón para mostrar el árbol genealógico de la paloma. En el caso de que el árbol esté completo, se verá como se muestra en la figura A2.23. Si no está completo, se mostrará como en la figura A2.24.

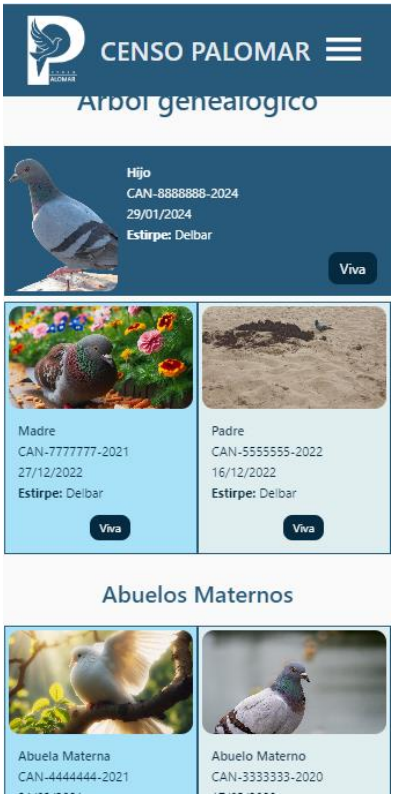

Figura A2.23: Árbol Genealógico lleno

Figura A2.24: Árbol Genealógico incompleto

A2.8 Gestión de competiciones

En este apartado se explicará cómo se realiza el registro, la modificación y la eliminación de un registro de una competición.

A2.8.1 Registro de una competición

El primer paso es dirigirse al perfil de la paloma a la que se quiere añadir una competición y en la barra de herramientas inferior seleccionar “Añadir Competición”. El usuario será redirigido a un formulario, que debe ser similar al de las figuras A2.25 y A2.26, y que deberá cumplimentar para crear el registro de la competición.

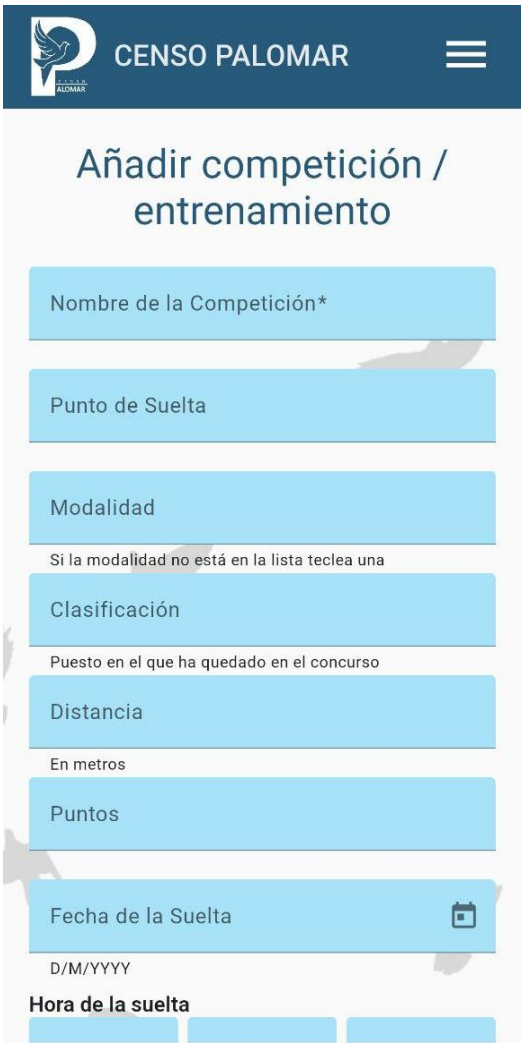

Figura A2.25: Formulario para añadir competición (Parte 1)

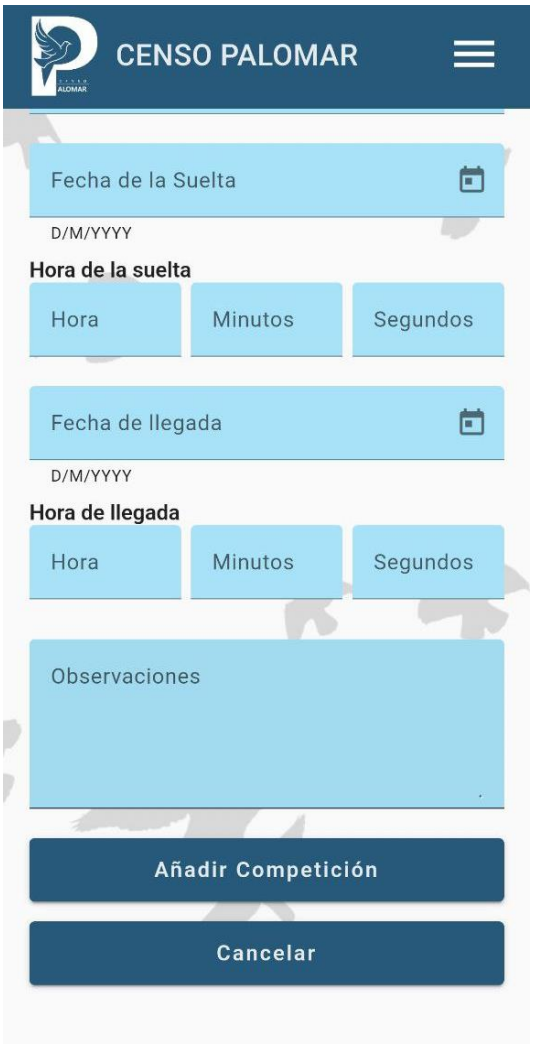

Figura A2.26: Formulario para añadir competición (Parte 2)

Selección de fechas

Para los datos de las fechas y horas de inicio y finalización de una competición, se debe seguir el siguiente procedimiento:

1. **Seleccionar la fecha:** al seleccionar el ícono en el campo de fechas, se desplegará una ventana, similar a la figura A2.27, desde donde se podrá seleccionar la fecha. También se puede introducir manualmente siguiendo el formato "mes/día/año"**.**
2. **Introducir la hora de llegada o de salida**: se introduce la hora, que debe ser un número entre 00 y 23, los minutos y los segundos, que deben ser números entre 00 y 59.

En el caso de que se introduzca mal algún dato, se mostrará un aviso para que el usuario pueda corregirlo, como se muestra en la figura A2.28.

Si los campos de las fechas y el campo de distancia están correctamente cumplimentados. Automáticamente se calculan la duración del vuelo y la velocidad que ha alcanzado la paloma.

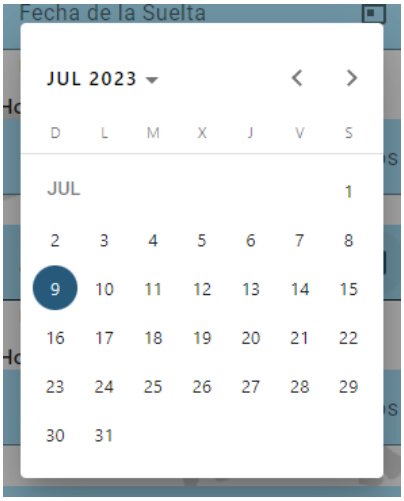

Figura A2.27; Vista del selector de fechas

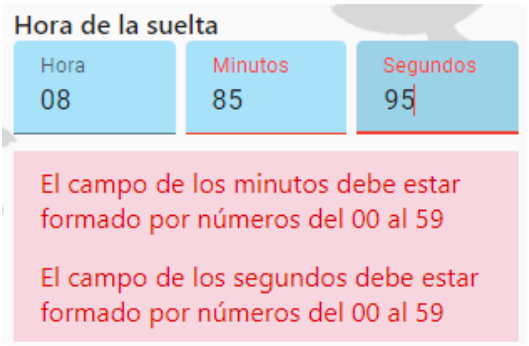

Figura A2.28: Vista de los campos para introducir la hora con advertencias de error

A2.7.2 Ver listado de competiciones

El listado de competiciones de una paloma se muestra al final del perfil de una paloma. Este listado contiene un resumen de la información más importante de cada competición y se verá de manera similar a la figura A2.29.

Debajo del ícono de la copa se puede ver la clasificación de la paloma para ese vuelo. También se podrá ver la distancia y la velocidad. Al lado, se mostrarán el resto de detalles de la competición, como el nombre, la fecha, el lugar de la suelta y la modalidad.

Figura A2.29: Listado de competiciones

Figura A2.30: Detalle de una competición

A2.7.3 Ver detalles de una competición

Si en el listado de competiciones se selecciona “ver más”, se mostrará el detalle completo de la competición que lucirá como se muestra en la figura A2.30

A2.7.4 Modificar registro de una competición

Para modificar el registro de una competición, hay que dirigirse al detalle de la competición que se quiere modificar y en la barra de herramientas inferior seleccionar la opción “Editar”. En ese momento, el usuario será redirigido a un formulario similar al de la creación del registro (ver figuras A2.25 y A2.26), pero cumplimentado con la información de la competición. El usuario puede modificar los datos que desee y, al seleccionar el botón “Actualizar Competición”, se harán efectivos los cambios.

A2.7.5 Eliminar registro de una competición

Para eliminar un registro de una competición, basta con dirigirse al detalle de la competición que se quiere eliminar y, en la barra de herramientas inferior, seleccionar la opción "Eliminar". En ese momento, aparecerá una ventana emergente, similar a la figura A2.22, para confirmar la eliminación del registro.

A2.8 Gestión de publicación de anuncios

En este apartado se explicará cómo se realiza la publicación, la modificación y la eliminación de un anuncio en la aplicación.

A2.8.1 Publicación de un anuncio

Para publicar un anuncio, en el menú se debe seleccionar la opción “Publicar Anuncio”. Una vez seleccionado se mostrará en pantalla el formulario que se debe cumplimentar. Esta pantalla será similar a la figura A2.31. Como en los anteriores formularios vistos, en caso de que los datos introducidos no sean correctos, se mostrarán advertencias.

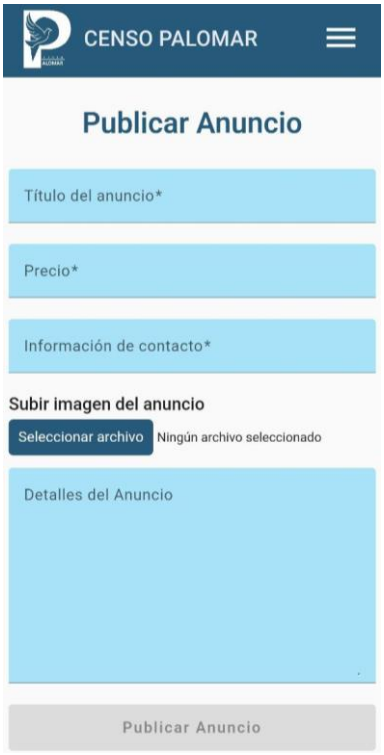

Figura A2.31: Formulario para publicar un anuncio

Figura A2.32: Listado de anuncios

A2.8.2 Listado de anuncios

Desde el menú si se selecciona la opción “Palomas en Venta” se conducirá al usuario a un listado donde puede ver los anuncios publicadas por los usuarios de la aplicación. Otra forma de ver este listado es desde la página principal, en la sección de anuncios se encuentra el botón “Anuncios” que conducirá a la misma página. El listado se verá como se muestra en la figura A2.32

A2.8.3 Listado de mis anuncios

Si el usuario está registrado y ha publicado anuncios verá un listado con los anuncios que ha publicado. Además, en cada fila encontrará una serie de iconos que permiten realizar diferentes acciones sobre esa noticia, ver el anuncio, editar el anuncio y eliminar el anuncio. El usuario verá en pantalla una vista similar a la figura A2.33

Figura A2.33: Listado de mis Anuncios

Figura A2.34: Detalle de un anuncio

A2.8.4 Vista del detalle de un anuncio

En el listado de anuncios, al seleccionar cualquier anuncio, el usuario verá toda la información publicada sobre el mismo, incluidos los datos de contacto con el vendedor. Se verá similar a como se muestra en la figura A2.34.

A2.8.5 Modificación de un anuncio

Para modificar el registro de un anuncio, hay que dirigirse al detalle de la competición que se quiere modificar y en la barra de herramientas inferior seleccionar la opción “Editar”. En ese momento, el usuario será redirigido a un formulario similar al de la creación del anuncio (ver figura A2.31), pero relleno con la información del anuncio. El usuario puede modificar los datos que desee y, al seleccionar el botón “Actualizar Anuncio”, se harán efectivos los cambios.

A2.8.6 Eliminación de un anuncio

Para eliminar un anuncio, basta con dirigirse al listado de anuncios y seleccionar el icono de la papelera. En ese momento, aparecerá una ventana emergente, similar a la figura A2.22, para confirmar la eliminación del anuncio.

A2.9 Gestión de publicación de noticias

En este apartado se detallará cómo el usuario administrador puede publicar, editar y eliminar una noticia.

A2.9.1 Publicación de una noticia

Para publicar una noticia, el administrador debe seleccionar la opción “Publicar Noticia”. Una vez seleccionado se mostrará en pantalla el formulario que se debe cumplimentar. Esta pantalla será similar a la figura A2.35. Como en los anteriores formularios vistos, en caso de que los datos introducidos no sean correctos, se mostrarán advertencias.

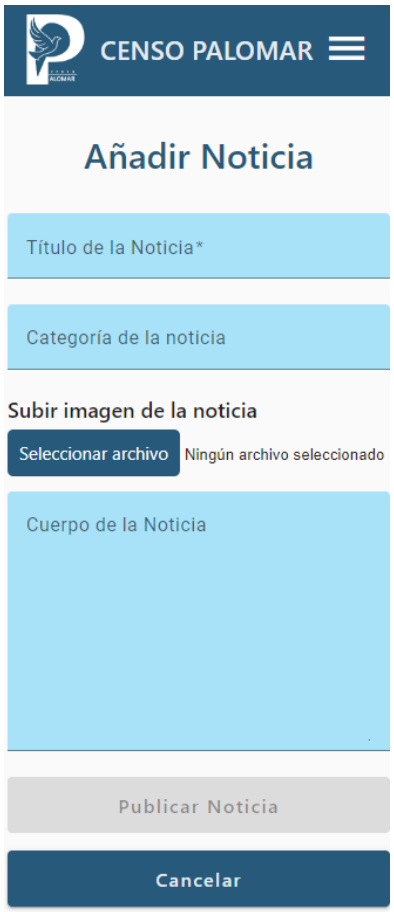

Figura A2.35: Formulario para publicar noticias

Figura A2.36: Vista del listado de noticias

A2.9.2 Listado de noticias

Desde el menú si se selecciona la opción “Noticias” se conducirá al usuario a un listado donde puede ver los anuncios publicadas por los usuarios de la aplicación. Otra forma de ver este listado es desde la página principal, en la sección de noticias se encuentra el botón “Ver todas las noticias” que conducirá a la misma página. El listado se verá como se muestra en la figura A2.36

A2.9.3 Editar una noticia

Para editar una noticia, el administrador tiene que dirigirse a la vista de la noticia que quiere modificar y en la barra de herramientas superior seleccionar la opción “Editar noticia”. La barra de herramienta se muestra en la figura A2.37. En ese momento, el administrador será redirigido a un formulario similar al de la publicación de noticias (ver figura A2.35), pero cumplimentado con los datos de la noticia. El administrador puede modificar los datos que desee y, al seleccionar el botón “Editar Noticia”, se harán efectivos los cambios.

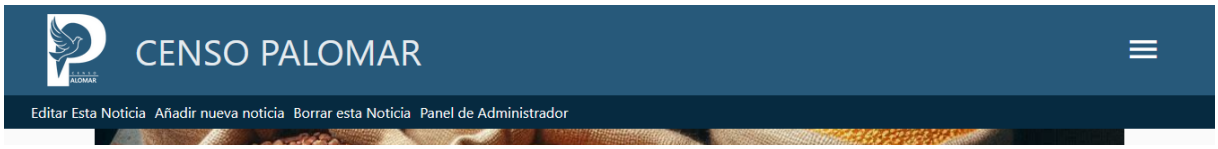

Figura A2.37: Barra de herramientas para las noticias

A2.9.4 Eliminación de una noticia

Para eliminar una noticia, el administrador debe dirigirse a la noticia que se quiere eliminar. En la parte superior, verá la barra de herramientas (ver figura A2.37) y deberá seleccionar “Eliminar Noticia”. En ese momento, aparecerá una ventana emergente para confirmar la eliminación del anuncio. En la figura A2.22 se puede ver la ventana emergente.

A2.10 Información del usuario

En esta sección se explican las acciones que puede realizar el usuario con su información.

A2.10.1 Ver perfil

Desde el menú puede consultar la información que la aplicación tiene almacenada. La vista de esta página es similar a la que se muestra en las figuras A2.38 y A2.39

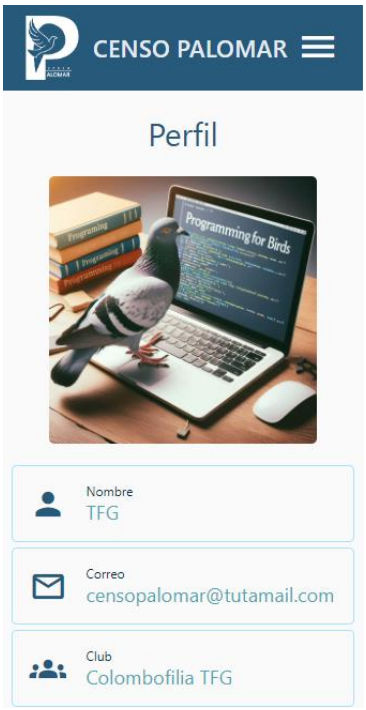

Figura A2.38: Perfil del Usuario (parte 1)

Figura A2.39: Perfil de usuario (parte 2)

A2.10.2 Editar perfil

Desde la página de perfil, el usuario puede seleccionar el botón "Editar perfil" que lo llevará a un formulario donde puede actualizar sus datos. El único dato que no puede cambiar es el correo electrónico, que aparece como deshabilitado. La vista de este formulario será similar a la mostrada en la figura A2.40.

Figura A2.40: Formulario para editar el perfil

A2.10.3 Cambiar contraseña

En la misma página de la edición del perfil, en la parte inferior, se encuentra el formulario para el cambio de contraseña. Es un formulario sencillo en el que el usuario tiene que introducir la contraseña actual, y luego, la nueva contraseña que quiere establecer, repetida para verificar que la ha puesto correctamente.

En caso de que la nueva contraseña no coincida, se mostrará una advertencia para que el usuario pueda corregirla.

En la figura A2.41 se muestra la vista de este formulario.

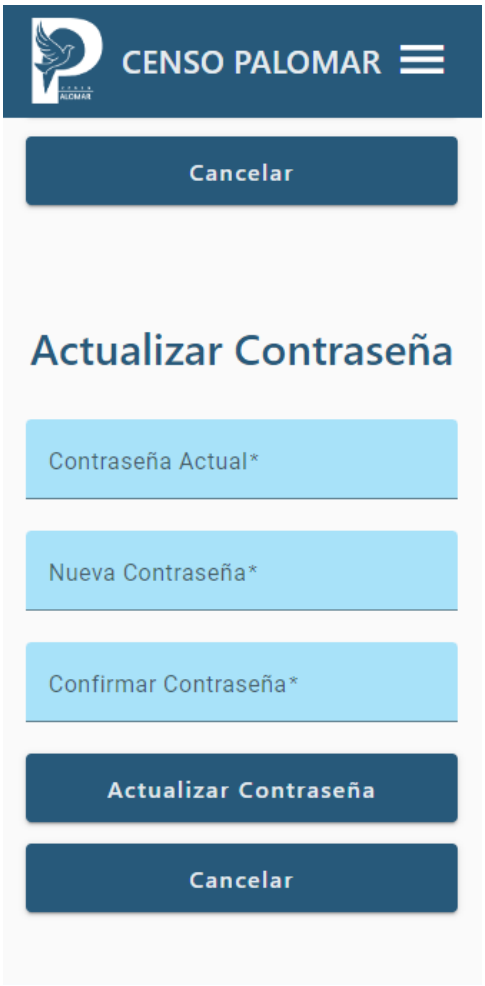

Figura A2.41: Formulario para el cambio de contraseña

A2.10 Funciones del Administrador

En este apartado se explicará las páginas creadas exclusivamente para el administrador.

A2.10.1 Panel del administrador

Desde el menú el administrador puede seleccionar un enlace desde el cual puede acceder al panel de administrador. Este panel contiene las funcionalidades que un usuario administrador puede realizar. Este panel tiene una vista como la que se muestra en la figura A2.42

Figura A2.42: Panel de administrador

A2.10.2 Listado de usuarios

El usuario administrador, desde el panel del administrador, podrá acceder a esta página donde se muestra un listado con todos los usuarios registrados en la plataforma. La vista de este listado es similar a la que se muestra en la figura A2.43

Figura A2.43: Listado de usuarios