Escuela Politécnica Superior

Aplicación multiplataforma Tinder mascotas

Grado en Ingeniería Informática

Trabajo Fin de Grado

Autor:
Esteban Antón Sellés

Tutor/es:
Estela Saquete Boro

Universitat d'Alacant
Universidad de Alicante

Diciembre 2022

RESUMEN

Aun habiendo vivido una pandemia, la situación de abandono de los animales de compañía en España ha disminuido y el número de adopciones en refugios está en aumento desde los últimos años. Aun así, el abandono sigue constituyendo el principal problema para los animales de compañía en 2022.

Este Trabajo de Final de Grado presenta una plataforma que abarca desde los usuarios que actúan como consumidores de ella, adoptando, hasta los usuarios que proveen animales de compañía para adoptar a través de la aplicación.

WannAdopt ofrece a los usuarios que quieran adoptar, una aplicación multiplataforma que cambia la dinámica de adopción online apoyándose en la esencia de la interfaz gráfica de Tinder, aprovechando su gran atractivo y éxito. De la misma forma, a los refugios se les ofrece un portal de gestión web que les permite tener de forma sencilla y centralizada la información sobre sus animales y las adopciones.

La siguiente documentación pretende ilustrar el proceso de desarrollo de dicha plataforma utilizando Scrumban como metodología ágil para la gestión del proyecto, aportando valor de producto en las etapas tempranas del proyecto, y tecnologías actuales y robustas tanto en la parte de cliente, Ionic y Angular, y en la parte de servidor, Node.js junto con Express y Sequelize como ORM para la base de datos.

En definitiva, el objetivo de este proyecto es presentar una nueva herramienta que cambie la forma de interacción de los usuarios en los portales de adopción de animales de compañía, haciendo que pase a ser la consulta de un catálogo a la presentación de un único animal en el que se decide si se tiene interés por adoptar o no.

ABSTRACT

Despite having lived through a pandemic, the situation of pet abandonment in Spain has decreased and the number of adoptions in shelters has been on the rise in recent years. Even so, abandonment continues to be the main problem for pets in 2022.

This Final Degree Project presents a platform that ranges from users who act as consumers of the platform, adopting, to users who provide pets for adoption through the application.

WannAdopt offers users who want to adopt a cross-platform application that changes the dynamics of online adoption by relying on the essence of Tinder's graphical interface, taking advantage of its great appeal and success. In the same way, shelters are offered a web management portal that allows them to easily and centrally manage information about their animals and adoptions.

The following documentation aims to illustrate the development process of this platform using Scrumban as an agile methodology for project management, providing product value in the early stages of the project, and current and robust technologies both on the client side, Ionic and Angular, and on the server side, Node.js along with Express and Sequelize as ORM for the database.

In short, the aim of this project is to present a new tool that changes the way users interact with pet adoption portals, changing from consulting a catalogue to the presentation of a single animal in which they decide whether they are interested in adopting or not.

ÍNDICE DE CONTENIDOS

Índice de figuras

Índice de tablas

1. INTRODUCCIÓN

El abandono de animales de compañía constituyó el principal problema de bienestar, en 2021, de los animales de compañía en España. En este mismo año, se estima que llegaron a los más de 1500 centros de acogida de animales de compañía de España 167.656 perros y 117.898 gatos. Estos datos suponen un ligero aumento en comparación el año anterior, aunque continúa por debajo de las cifras recogidas en años anteriores a la pandemia. (Affinity, 2021)

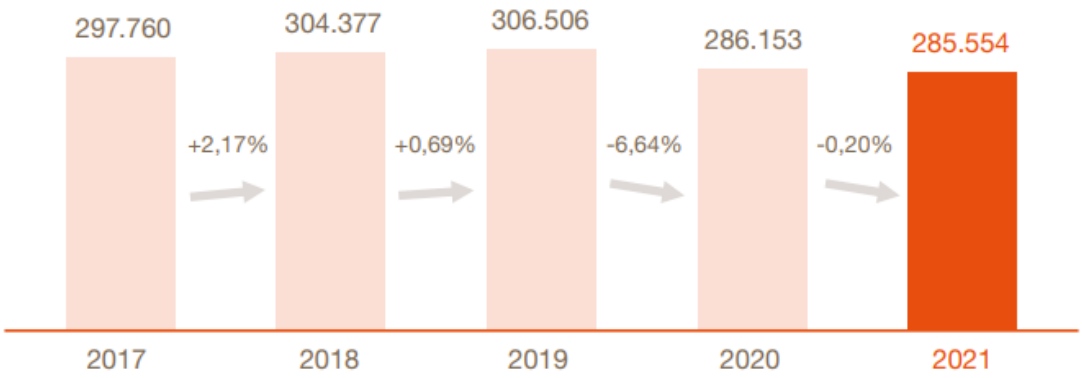


Figura 1. Evolución del número de animales que llegan cada año a refugios o protectoras de animales. (Affinity, 2021)

Los datos anteriores permiten estimar la tasa de abandono y/o pérdida de animales en 3,5 perros y 2,5 gatos por cada 1000 habitantes (Población española: 47.326.687).

Tras la vuelta a la normalidad ante un estado de alarma provocado por la pandemia de la COVID-19, el número de perros que llegan a refugios o protectoras de animales se ha incrementado en un 11,52%, pudiendo interpretarse el aumento de sus salidas y la movilidad de las personas a cargo de ellos. No así sucede con los gatos, que se ha reducido en un 5,03% las llegadas a estos centros de acogida.

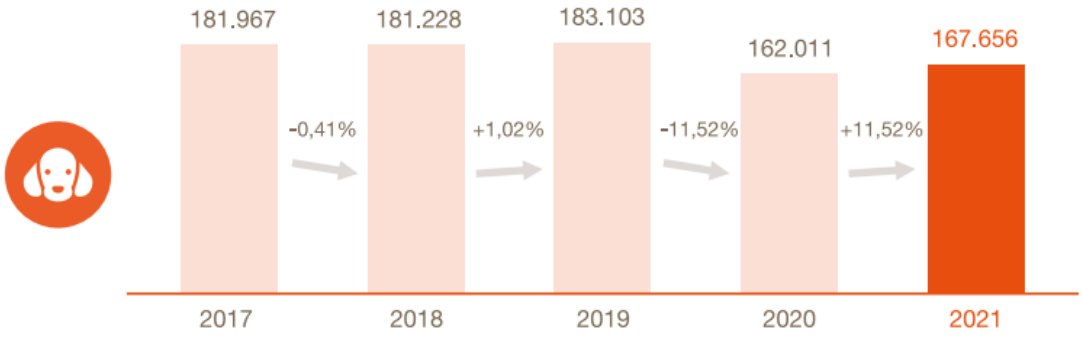


Figura 2. Evolución del número de perros que llegan cada año a refugios o protectoras de animales. (Affinity, 2021)

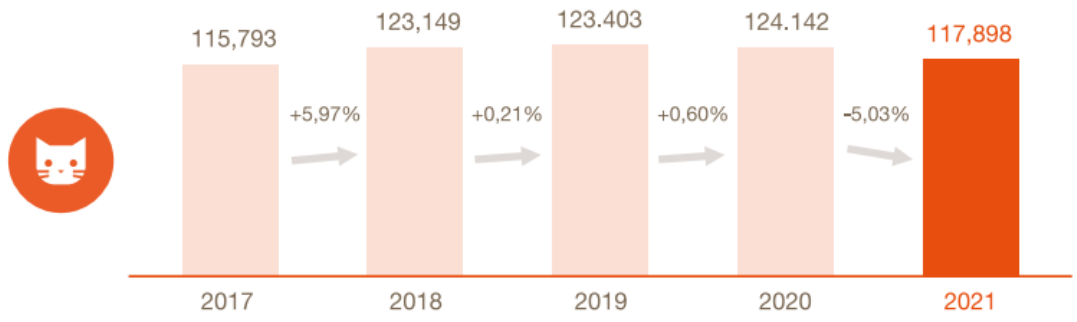


Figura 3. Evolución del número de gatos que llegan cada año a refugios o protectoras de animales. (Affinity, 2021)

En cuanto a los perfiles de los animales de compañía que son acogidos por los centros, se aprecia una considerable diferencia de edad entre perros y gatos, siendo un 26% de cachorros, un 57% de adultos y un 17% de edad avanzada en el caso de los perros, y un 54,7% de cachorros, un 38,2% de adultos y un 7,1% de edad avanzada en el caso de los gatos.

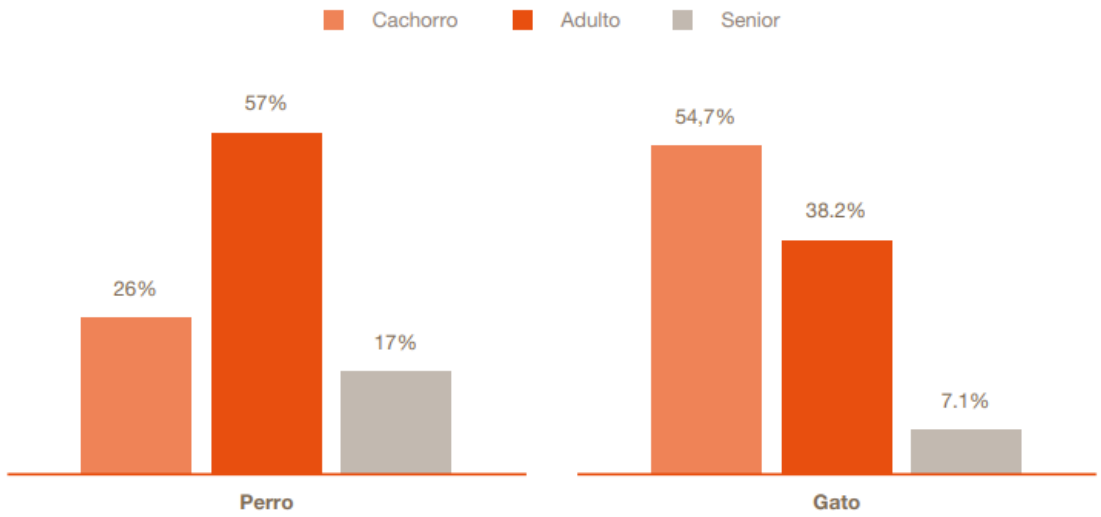


Figura 4. Porcentaje de animales recogidos por los refugios en función de su edad. (Affinity, 2021)

Si bien la adopción es una de las medidas más efectivas para disminuir el impacto negativo del abandono de animales a corto y medio plazo, no todos los animales tienen las mismas probabilidades de ser adoptados. Los cachorros de perro permanecen en los refugios una media de 2,1 meses antes de ser adoptados, mientras que en los adultos y los sénior el período medio de estancia se eleva a 13,3 meses. En el caso de los gatos, los cachorros pasan una media de 3,1 meses en los refugios, mientras que en los adultos y los sénior el período medio aumenta a los 11,5 meses.

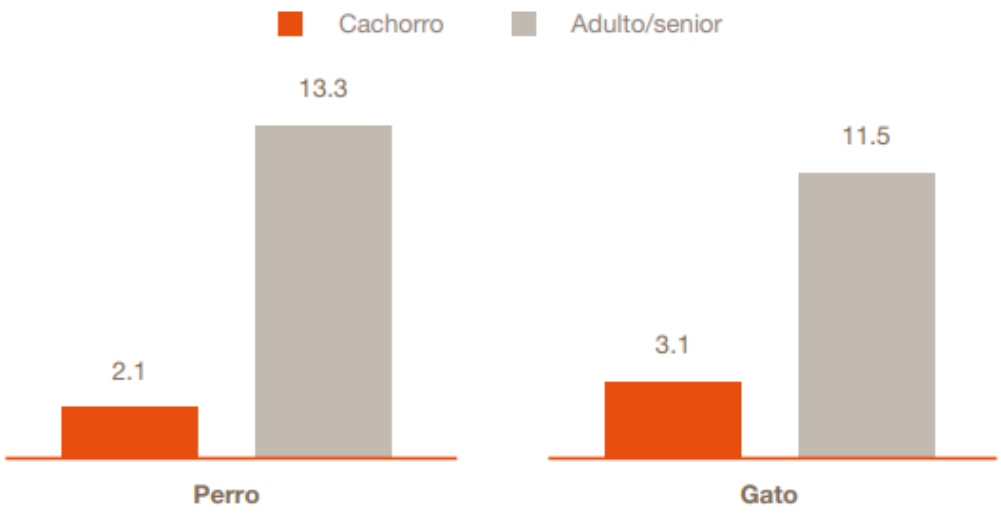


Figura 5. Duración media de la estancia de los perros y de los gatos en función de su edad en el momento. (Affinity, 2021)

El menor tiempo de permanencia de los cachorros en el refugio puede ser explicado en gran parte por el mayor atractivo que posee un animal joven para muchos adoptantes. A esto se le suma la idea equivocada de que un animal adulto no se integrará a su nueva familia o que ya no podrá ser educado de manera efectiva.

En cuanto al éxito de la adopción, el porcentaje de éxito comunicado por las entidades de acogida de animales de compañía es muy elevando, una media del 90%. Sin embargo, en una de cada diez adopciones el vínculo no llega a consolidarse debido a diferentes barreras. Coincidiendo con las cifras encontradas en otros lugares del mundo, el comportamiento del animal es referido como el principal motivo de retorno al refugio, seguido por las expectativas equivocadas acerca de la responsabilidad que supone cuidar de un animal de compañía.

Figura 6. Principales motivos de la devolución al refugio de los animales adoptados. (Affinity, 2021)

Estas cifras de retorno, aunque inevitables por el momento, sí son mejorables, y a través de la reflexión y del compromiso de los adoptantes es una de las vías consideradas. Cuestiones como el conocimiento de la duración de vida media de los perros y gatos, que es de 12 a 14 años; la alta probabilidad de haber vivido situaciones de abandono y/o maltrato y que se quiere evitar que viva de nuevo, el período de adaptación y reeducación del animal, el disponer del tiempo y el espacio necesario para su bienestar o los gastos asociados al cuidado de estos, deben ser planteadas antes de dar el paso a la adopción. A través de un conocimiento previo por parte de los centros de acogida de animales de compañía del adoptante y la posibilidad de hacer una previa valoración sobre la compatibilidad de esa persona con el animal, se puede conseguir un destino para el animal de compañía y la satisfacción de convivir con un animal de compañía adecuado por parte del adoptante. (Anerpa, 2022)

Como se ha mencionado anteriormente, el atractivo del animal es un factor decisivo, actualmente, en la selección de un animal de compañía para su adopción. Casi podría decirse que la decisión se toma de forma emocional más que racional, impulsada por la apariencia y el atractivo del animal de compañía que provoca el deseo de ofrecerle un hogar, siempre en comparación con otros de su alrededor, claro. Cayendo en el juego "sucio" de la naturaleza y nuestra psicología, animales considerados de menor atractivo o caracterizados de mayores imperfecciones se ven perjudicados y castigados con el rechazo.

No todo recae en nuestra forma de consumo. La revolución tecnológica vivida hasta el momento nos ha permitido superar innumerables barreras, tener acceso a servicios remotos de manera instantánea y optimizar la mayoría de procesos cotidianos. Sin embargo, en esta carrera por la eficiencia y la obtención de lo que queremos de forma rápida y sencilla, se han reutilizado modelos válidos y útiles en unas áreas a otras que quizá estén necesitadas de revisión a la hora de implantarse. Es inevitable que comparemos a la hora de tomar una decisión, pues nuestra mente se siente segura en lo predecible y eso es lo que perseguimos de manera inconsciente es hacia donde tendemos. Lo que sí es evitable es diferenciar el trato y la valoración ofrecida a un ser vivo y a un objeto o producto. Si esto es así, y me arriesgo a que alguien se sienta fuertemente animado a corregirme en esta afirmación, ¿por qué no cambiar la manera de hacer las cosas y evitar los catálogos y escaparates de seres vivos?

Con este proyecto se busca la forma de crear un espacio entre la persona interesada en la adopción de un animal de compañía y dicho animal, sin comparaciones directas, sin juicios superficiales. Este proyecto quiere acercar el compromiso y el respeto a las adopciones de animales de compañía desde un primer momento, evitando exponer estas amables vidas como un producto más que poder obtener. Creo firmemente que este es un gran momento para centrarse en este eje de cambio.

2. MARCO TEÓRICO Y ANÁLISIS DE MERCADO

2.1. Público objetivo

La plataforma web se centra en un público de habla hispana (peninsular) concienciado con la adopción de animales y que dispone de acceso a internet; y son tanto los usuarios de la plataforma como las protectoras de animales de ámbito nacional.

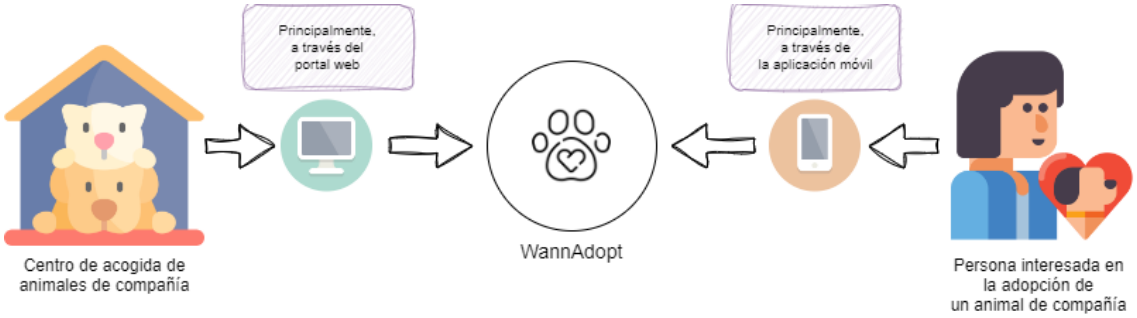


Figura 7. Diagrama de la interacción de los usuarios con las aplicaciones del sistema. (Fuente propia)

2.2. Competencia

El estudio sobre la competencia y las aplicaciones similares se ha observado desde el punto de vista de que la aplicación tiene como objetivo cubrir el servicio a nivel nacional, acotándose de primeras en la provincia de Alicante. Por lo que los valores de cobertura de algunas webs y aplicaciones, han sido estudiadas para esta región.

Para analizar la competencia, se buscó en internet plataformas web que hiciesen algo parecido. Se analizaron sus redes sociales y se utilizó *SimilarWeb* para generar informes sobre cada una de sus webs.

2.2.1. Bambú difunde

Con un total de 574 asociaciones y protectoras vinculadas, esta web se considera una de las mayores competencias directas. Ofrece cobertura en la mayoría del territorio nacional, permitiendo que otras asociaciones integren, a través de su CMS, su sistema de adopción.

Figura 8. Bambú Banner difunde (¡Enlázanos!, 2022)

En lo referente a sus cifras de visitas en internet, actualmente ocupa el puesto número 8 de la clasificación de categorías de Mascotas y animales, Mascotas (En España). Con una composición del público del 48,74% hombres y el 51,26% mujeres, el grupo más numeroso de visitantes por edad es el de 25 a 34 años, a través de ordenador.

2.2.2. Miwuki

Página web sobre adopción de mascotas bastante reconocida y con una interfaz bien trabajada. Las protectoras se dan de alta en la plataforma (a través de un CRM de la misma marca) y añaden a los animales en adopción y los detalles de estos.

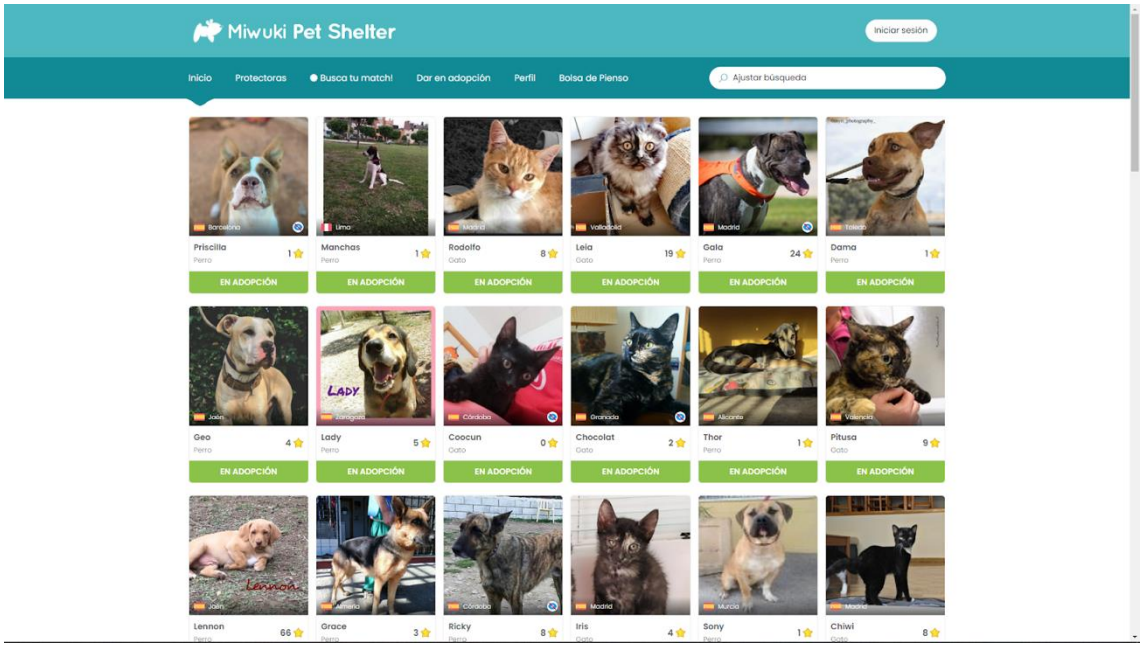


Figura 9. Captura de pantalla de la sección de Quiero adoptar perros en Madrid en Miwuki. (Perros en adopción en Madrid, 2022)

Se da información sobre el animal y sobre su estado cuando se entrega. Dónde se encuentra, el alcance que tiene la protectora para enviar al animal y la tasa de adopción de esta.

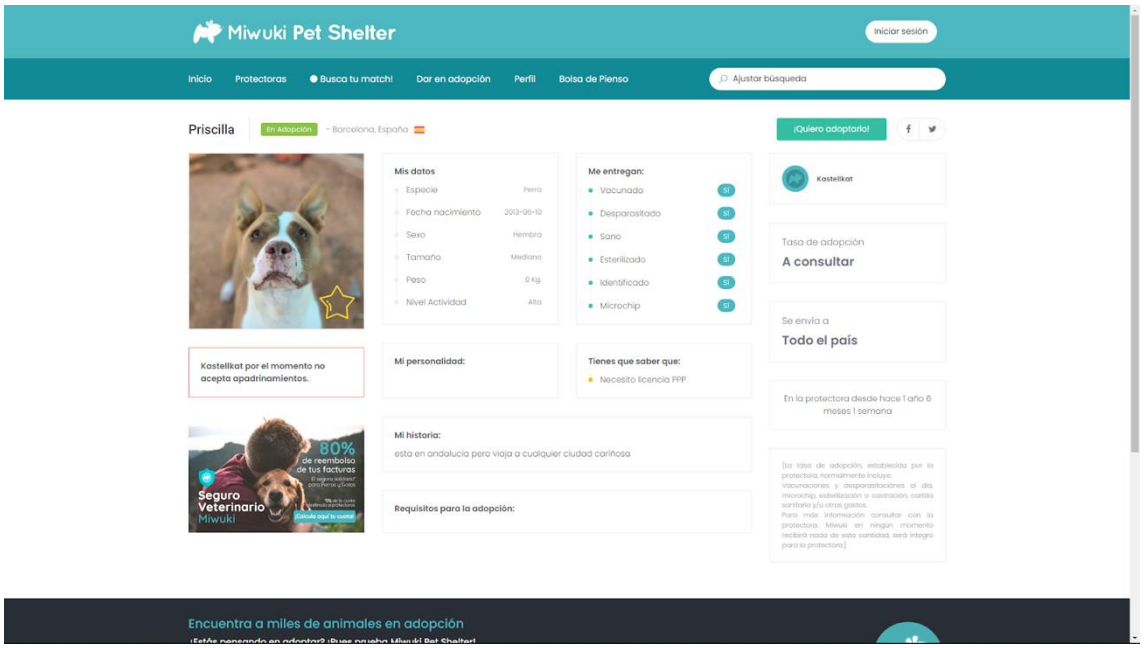


Figura 10. Perfil de un perro en adopción en Miwuki. (Perros en adopción en Madrid, 2022)

Por la parte del análisis de web, está posicionado en el puesto 15 de la clasificación de categorías de Mascotas y animales, Mascotas (En España), según los datos de SimilarWeb. Además, el 90,22% de las visitas provienen de España y su público se compone, principalmente, de un 55,29% de hombres y un 44,71% de mujeres, siendo el grupo de edad de 25 a 34 años el más numeroso entre sus visitantes.

2.2.3. Kiwoko

Se definen como un proyecto donde buscar entre perros, gatos y otros animales de todas las proyectoras de España, invitando a encontrar a la mascota ideal.

Figura 11. Landing page de Kiwoko. (Kiwoko, 2022)

Estando posicionada en el puesto número 24 de la clasificación de categorías de Mascotas y animales, Mascotas (En España), cuenta con un 89,23% de visitas desde España. Por edad, el grupo más numeroso de visitantes es el de 25 a 34 años, y la distribución del género es del 52,55% de público masculino y del 47,55% de público femenino.

Como en otras webs estudiadas, ofrece un catálogo y una serie de filtros sobre características de los animales.

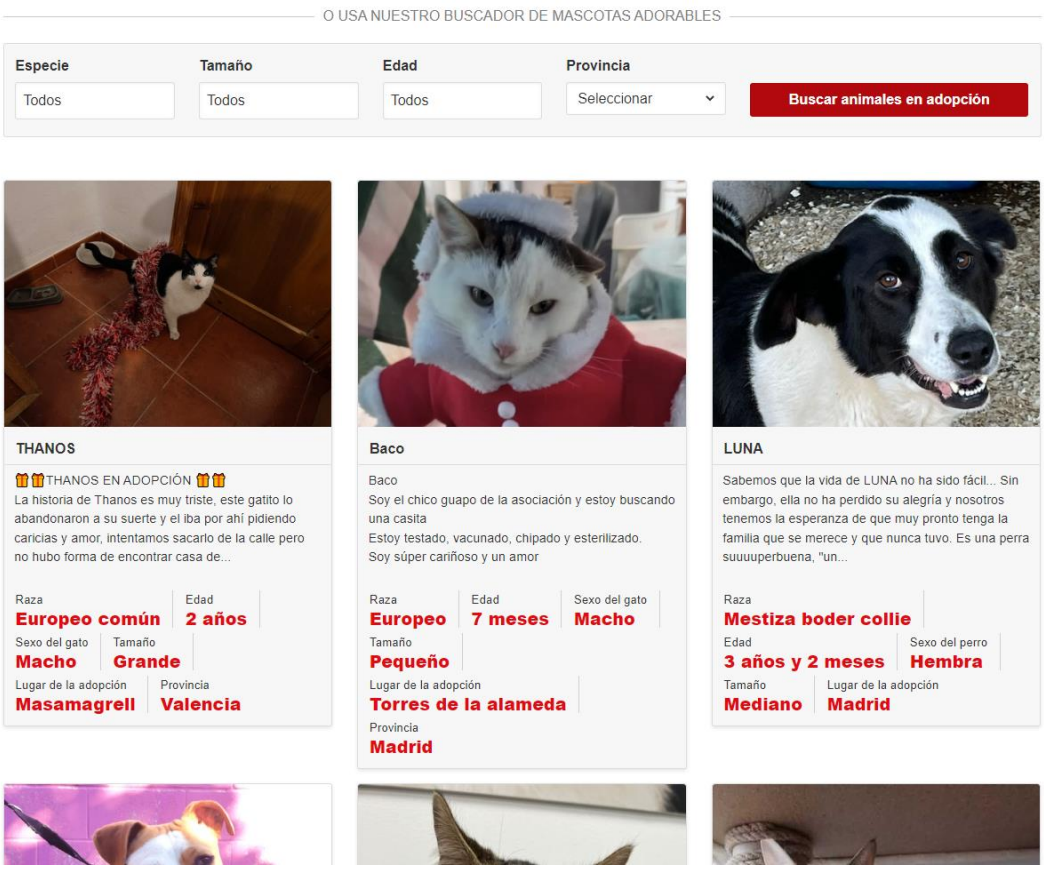


Figura 12. Listado de animales en adopción en Kiwoko. (Kiwoko, 2022)

2.2.4. Chuby

Aplicación nativa que permite buscar por filtros, subir y buscar anuncios de animales perdidos.

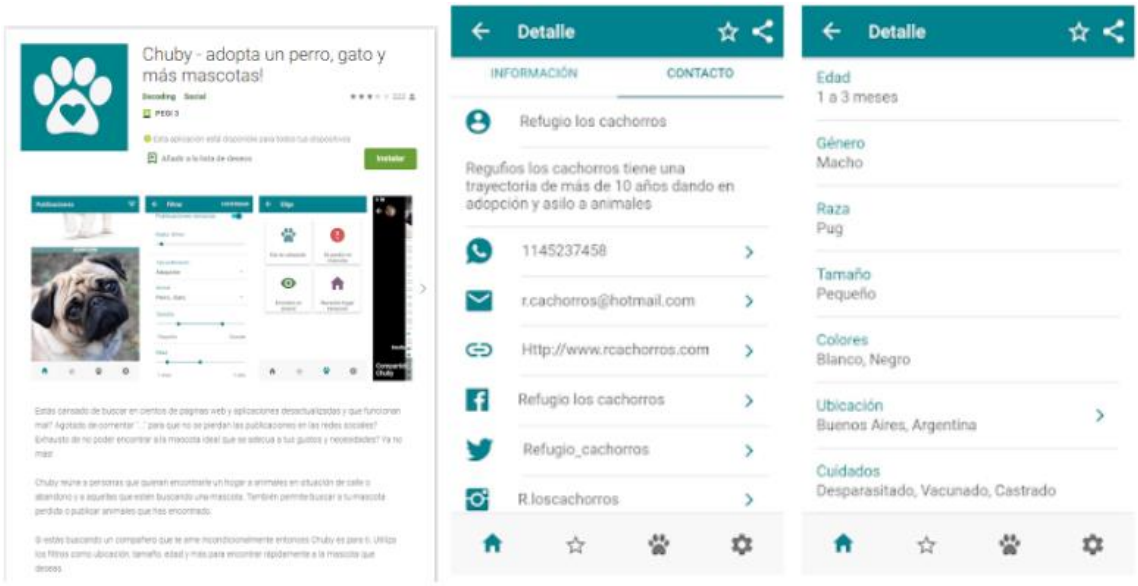


Figura 13. Aplicación Chuby en Play Store. (Chuby - Adopta un perro, gato, 2022)

Además, permite compartir por imagen las publicaciones de los animales en adopción y ponerse en contacto a través de alguno de los medios de comunicación publicados por la persona creadora de la publicación.

2.2.5. BuscaFuska

Página web de un buscador de animales de adopción con la curiosidad del filtro "perronalidad", detalle innovador y particular de esta plataforma.

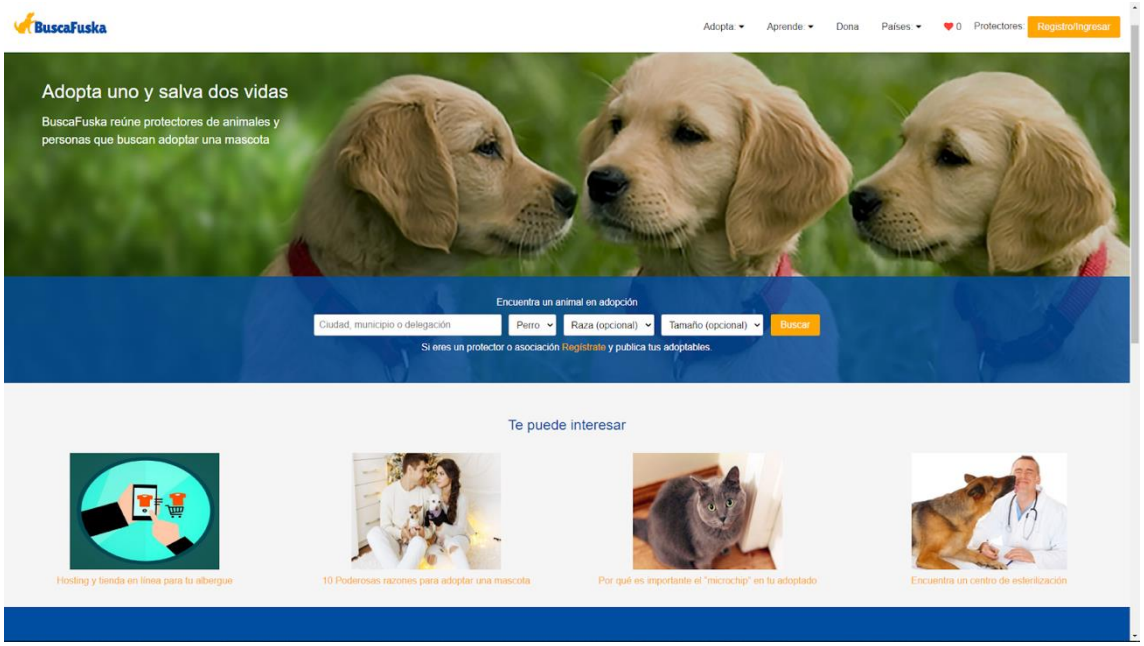


Figura 14. Filtro de búsqueda de BuscaFuska. (BuscaFuska, 2022)

Además, proporciona un test para encontrar a la mascota ideal.

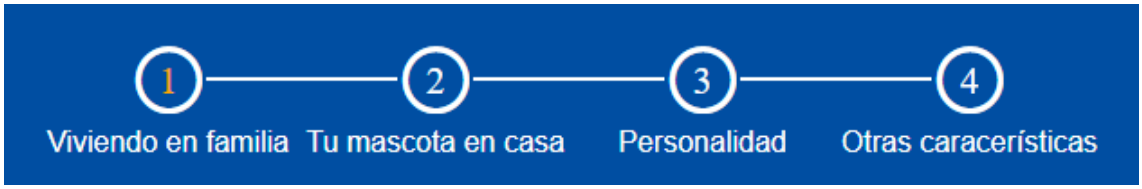


Figura 15. Pasos del test de compatibilidad con un animal. (BuscaFuska, 2022)

Geográficamente, el público de esta web se reparte más entre Latinoamérica que en España.

2.2.6. Otros

Al igual que los animales de compañía en adopción que se pueden encontrar en todas estas páginas web, existen plataformas semejantes, pero siempre con una característica diferenciadora. La lista continúa extendiéndose (kerubi.es, mascotasadopcion.com, adopcionanimal.es, seproanimal.com, AdoptMe, Amazdog, etc.), dando a entender que se pueden encontrar gran variedad de plataformas con un diseño tanto en presentación como en filtros muy similar.

Esta similitud en el diseño deja espacio a nuevas propuestas de diseño en la dinámica de la adopción de animales de compañía.

2.3. Marketing

Disponer de un buen plan de marketing, bien diseñado y desarrollado, puede suponer parte de la clave del éxito de un negocio. Con las características de este proyecto, el objetivo es ganar presencia en los espacios donde frecuente los usuarios potenciales de la aplicación y mantener esa visibilidad ganada. (2Spacios, 2015) Las opciones contempladas formarán parte de un primer plan de marketing de bajo coste y autogestionado.

Como foco principal para el posicionamiento web, se tiene en cuenta el posicionamiento orgánico en los buscadores (SEO), optimizando la web para que Google sea capaz de rastrear e indexar correctamente la estructura de la plataforma web para que se muestre en las primeras posiciones en las búsquedas sobre temas relacionados. En esto juegan un papel importante características como los tiempos de carga, detallar el contenido con palabras clave, la buena experiencia de navegación y las referencias de otras webs a las plataformas. Por ello, el hacer colaboraciones con microinfluencers relacionados con el mundo animal, o plataformas y asociaciones de rescate de animales domésticos, se tiene contemplado como una forma de conseguir esta visibilidad en la web.

Tras unos primeros resultados sobre las visitas a la web con el análisis de las estadísticas de Google Analytics, herramienta con la que conocer el tráfico que llega y el comportamiento dentro de la web, se puede ajustar la estrategia de marketing con respecto a estos parámetros y favorecer las apariciones de la plataforma a través del

posicionamiento SEM, utilizando Google Adwords, que permite hacer publicidad patrocinada dentro del buscador de Google.

2.4. Fuente de ingresos y viabilidad económica

Aunque no se trata de un proyecto sin ánimo de lucro, se entiende la complicada situación económica por la que pasan los centros de acogida de animales de compañía y los refugios. Por ello, se contempla comenzar con una oferta de 6 meses en la que los refugios tendrán acceso completamente gratuito a la plataforma. Tras este período de oferta en el que se permite probar la utilidad de la aplicación, se debería llevar a cabo un contrato de suscripción mensual, trimestral y anual, de no más de 20€ mensuales.

Desde la aplicación móvil, se añadirían anuncios, publicidad In-App, para obtener ingresos con el uso de la aplicación. (Concepto, s.f.)

Además, se intentaría hacer uso de las subvenciones y programas de ayuda para emprendedores y startups.

3. METODOLOGÍA

3.1. Herramientas

- ***VS Code*:** Editor simplificado con soporte de operaciones de desarrollo como la depuración, la ejecución de tareas y el control de versiones. Su objetivo es proporcionar sólo las herramientas que un desarrollador necesita para un ciclo rápido de construcción y depuración de código. Ha sido utilizado para el desarrollo tanto del back end como del front end.
- ***Git*:** Sistema de control de versiones distribuido, gratuito y de código abierto, diseñado para manejar todo tipo de proyectos, desde los más pequeños hasta los más grandes, con rapidez y eficiencia.
- ***GitHub*:** Plataforma de desarrollo colaborativo para alojar proyectos utilizando el sistema de control de versiones Git. Se utiliza principalmente para la creación de código fuente de programas de ordenador. Utilizada por su cómoda interfaz y los productos que ofrece para facilitar el mantenimiento y desarrollo de código.
- ***Drive*:** Servicio de alojamiento de archivos que fue introducido por la empresa estadounidense Google. Se cuenta con 15 GB de almacenamiento gratuito para almacenar sus archivos. Es accesible a través del sitio web o desde las aplicaciones para Android y iOS. Utilizado para compartir documentos con la tutora.
- ***Trello*:** Software de administración de proyectos con interfaz web y con cliente para iOS y Android para organizar proyectos. Utilizado para la organización y medición de tareas del proyecto.
- ***Meet*:** Servicio de videotelefonía desarrollado por Google. Utilizado para la comunicación con la tutora.
- ***Google Mail*:** Servicio de correo electrónico proporcionado por Google.
- ***Postman*:** Aplicación que permite realizar peticiones de una manera simple para testear APIs de tipo REST propias o de terceros.

3.2. Tecnologías y lenguajes de programación

Para comenzar a hablar de las tecnologías y lenguajes de programación que han sido utilizados en este proyecto, se comenzará el lenguaje de programación en el que se ha basado mayoritariamente toda la aplicación, **JavaScript**.

JavaScript es un lenguaje de programación o de secuencias de comandos que permite implementar funciones complejas en páginas web. Es especialmente reconocido por permitir la creación de contenido de actualización dinámica, control multimedia y animación de imágenes, entre otras muchas otras cosas, en la web. Esto le ha hecho convertirse en uno de los lenguajes más populares, en parte porque ha evolucionado y mejorado a pasos agigantados. (MDN web docs, s.f.)

3.2.1. Node.js

Como bien se ha explicado previamente, JavaScript se ha popularizado, pasando a ser un lenguaje que podía ejecutarse solo en el navegador a uno que pudiera ejecutarse también como si se tratase de aplicaciones independientes en un ordenador. (Lucas, 2019)

Esta transformación es posible gracias a Node.js o NodeJS, siendo un entorno de tiempo de ejecución JavaScript y consiguiendo ir un paso más allá en la programación con JavaScript a través de la ejecución del código en el motor de tiempo de ejecución JavaScript V8 (nombre del motor de JavaScript que alimenta Google Chrome).

En resumen, Node.js brilla en la creación de aplicaciones de red rápidas, ya que es capaz de manejar una gran cantidad de conexiones simultáneas con un alto nivel de rendimiento, lo que equivale a una alta escalabilidad.

3.2.2. AngularJS

Angular es un framework para el desarrollo del front end, escrito en **TypeScript**, que sirve para desarrollar aplicaciones web estilo Single Page Application (SPA) y Progressive Web App (PWA), sirviendo tanto para versiones móviles como de escritorio. (Gonçalves, 2021)

Utilizando AngulaJS se ha podido disfrutar de sus diversas ventajas en el desarrollo de la aplicación web (Handa, 2021):

- **Coherencia y reutilización del código:** Gracias a su estructura basada en componentes hace que los componentes sean altamente reutilizables y simplifica el proceso de desarrollo.
- **Fácil de aprender, usar y probar:** Ayuda a escribir código más conciso, ahorrando tiempo y esfuerzo a la persona que desarrolla con AngularJS.
- Soporte de Google y excelente soporte de la comunidad
  Este marco de trabajo cuenta con un excelente apoyo de la comunidad para desarrollar una aplicación única y fácil de usar.
- **Funciones orientas a SPA:** SPA pertenece al ADN de AngularJS. Este admite el desarrollo de aplicaciones de una sola página, cuyo propósito principal es obtener una transición más rápida del sitio web.
- **Enlace de datos bidireccional:** Utilizando este tipo de enlaces, la aplicación simplificará la cala de presentación, permitiendo un enfoque más simple y menos intrusivo de la visualización del DOM para construir la interfaz de usuario.
- **UI declarativa:** Su modelo de codificación declarativo se puede emplear para generar diseños de acceso frecuente que dan como resultado un código liviano con una facilidad óptima de lectura y escritura.
- **Integración perfecta y productividad de gama alta:** AngularJS se integra sin problemas con otras librerías, ofreciendo un marco inteligente y robusto que ahorra mucho tiempo en desarrollo.

TypeScript es un superset de JavaScript, es decir, los programas de JavaScript son programas válidos de TypeScript, a pesar de que TypeScript sea otro lenguaje de programación. Esto permite a TypeScript integrarse en proyectos de JavaScript sin reimplementar el código existente. Su principal característica es el tipado estático, permitiendo la implementación de herramientas de desarrollo más avanzadas. (Hernández, 2018)

Por lo tanto, TypeScript es la solución a muchos de los problemas de JavaScript, está pensado para el desarrollo de aplicaciones robustas, implementando características en el lenguaje que nos permitan desarrollar herramientas más avanzadas para el desarrollo de aplicaciones.

3.2.3. Ionic

Como tecnología para el desarrollo de la aplicación móvil y web, se ha utilizado Ionic. Este framework permite el desarrollo de aplicaciones híbridas basadas en tecnologías web (HTML, CSS y JS). Con esto, se tiene la posibilidad de crear una aplicación para iOS nativo, Android y web desde una única base de código. (Atmitim, 2021)

3.3. Planificación del proyecto

3.3.1. Estimación temporal

El proyecto ha sido planificado para el desarrollo y documentación **desde mediados de junio hasta mediados de diciembre, un total de seis meses**, aproximadamente. Para poder ser llevado a cabo en este período con la mejor calidad y compaginándose con el trabajo, se ha decidido trabajar de manera ágil bajo los cuatro valores contemplados en el Manifiesto Agile (Sentrio, 2021):

- Individuos e interacciones por encima de procesos y herramientas
- Software funcionando por encima de documentación exhaustiva
- Colaboración con el cliente por encima de negociación contractual
- Respuesta ante el cambio por encima de seguir un plan

Y bajo estos valores, siguiendo los doce principios que derivan de ellos:

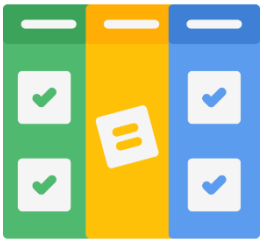
1. Satisfacer al cliente mediante la entrega temprana y continua
2. Aprovechar el cambio como ventaja competitiva
3. Entregar valor frecuentemente
4. Cooperación negocio-desarrolladores durante todo el proyecto
5. Construir proyectos en torno a individuos motivados
6. Utilizar la comunicación cara a cara
7. Software funcionando como medida de progreso
8. Promover y mantener un desarrollo sostenible
9. La excelencia técnica mejora la agilidad
10. La simplicidad es fundamental
11. Equipos autoorganizados para generar más valor
12. Reflexión y ajustes frecuentes del trabajo de los equipos

Dentro de esta metodología de trabajo, aunque sin equipo con el que trabajar, se ha trabajado dentro del marco de trabajo planteado en **Scrumban**. Esta metodología de gestión de proyectos combina dos estrategias ágiles comunes: Scrum y Kanban. (Laoyan, 2022)

Antes de explicar cómo funciona Scrumban, se comentará rápidamente en qué se basan las estrategias de Scrum y Kanban.

- Scrum: Es un marco ágil que ayuda a los equipos a colaborar y realizar un trabajo de alto impacto. Por lo general, Scrum se lleva a cabo en *sprints* de dos semanas de duración. Estos *sprints* son uno de los eventos predefinidos en Scrum. Por lo general, Scrum se divide en cinco eventos que hacen que se revise la dirección y el estado actual del proyecto de forma orgánica y se trabaje en la correcta evolución del proyecto, atendiendo a las demandas del cliente. A parte de los *sprints*, existen las reuniones de planificación del *sprint* (antes de cada *sprint*), las reuniones diarias (durante el *sprint*), la revisión del *sprint* (al final del *sprint*) y la retrospectiva del *sprint* (al final de este). (Donetonic, s.f.)

- Kanban: Es un marco ágil muy útil para que los equipos distribuyan equitativamente el trabajo según la capacidad productiva disponible de cada integrante. Esta estrategia se basa en la filosofía de la mejora continua.



Tras esta breve introducción a ambas estrategias ágiles, Scrumban es una fusión de varias características de ambas, tal y como sugiere el nombre. Como sucede en ambos marcos, las tareas son llevadas a un *backlog* tras haber realizado un estudio de las necesidades del cliente y estas tareas son representadas con tarjetas. El trabajo se prioriza según la complejidad de la tarea y la demanda del producto, como sucede en Scrum, y estas tarjetas que forman el trabajo a realizar en el *sprint* actual, avanzan a través de diferentes etapas del proceso en el tablero de Scrumban. Por parte de Kanban, se añaden límites estrictos con respecto a la cantidad de tareas en progreso, para evitar el exceso de trabajo en el equipo. Esto ha resultado ser crucial para poder llevar a cabo las tareas de forma ordenada, ya que se ha tenido que realizar las tareas desde varios puntos de vista. Simplificándolo, se han llevado a cabo dos roles: líder de equipo, mezclando las tareas de *Product Owner* y *Scrum Master*, y la de desarrollador *Full Stack*. Por ese motivo, esta metodología ha sido de gran ayuda.

Durante todo el proyecto, ha habido un total de doce *sprints*, y en cada uno se intentaba conseguir diferentes cosas de manera global:

- Sprint 1: Estudio de mercado y definición de requisitos funcionales.
- Sprint 2: Definición de la arquitectura, creación de esquemas de base de datos y diseño de primeros mockups.
- Sprint 3: Especificación de las herramientas utilizadas para el desarrollo e implementación de una prueba.
- Sprint 4: Diseño del inicio de sesión y el registro en el *front end*.
- Sprint 5: Revisión de la autenticación de la aplicación y protección de la parte privada de la aplicación.
- Sprint 6: Definición de rutas del API utilizada.
- Sprint 7: Integración de la base de datos con el API y el *front end*.
- Sprint 8: Desarrollo del CRUD de la parte de gestión de animales.
- Sprint 9: Diseño de las páginas de gestión de animales.
- Sprint 10: Diseño de las pantallas de la aplicación para el usuario.
- Sprint 11: Realización de pruebas para la comprobación del funcionamiento de la aplicación.
- Sprint 12: Revisión del proyecto, la seguridad y la documentación.

El seguimiento de las tareas realizadas en cada *sprint* se ha realizado apoyándose en Trello. Con esta herramienta se ha creado un tablero para el proyecto a partir de las columnas básicas o artefactos utilizados en el paradigma de Scrumban. Por lo que, además de la columnas de Selected, In Progress y Done, se han utilizado cuatro columnas más para detallar el estado de una tarea.

En primer lugar, como Product Backlog se ha utilizado la columna de Backlog, sirviendo como fuente principal sobre el producto. En ella se encontraban las tareas futuras que se añadirían a los *sprints* pertinentes.

La siguiente columna ha sido la de Selected, sirviendo de Sprint Backlog. En ella, al principio de cada *sprint*, se colocaban las tareas que se iban a realizar durante este. De esta forma, las tareas que continuaban en el Backlog podían ser refinadas en un futuro, dependiendo del resultado del trabajo al final del *sprint*.

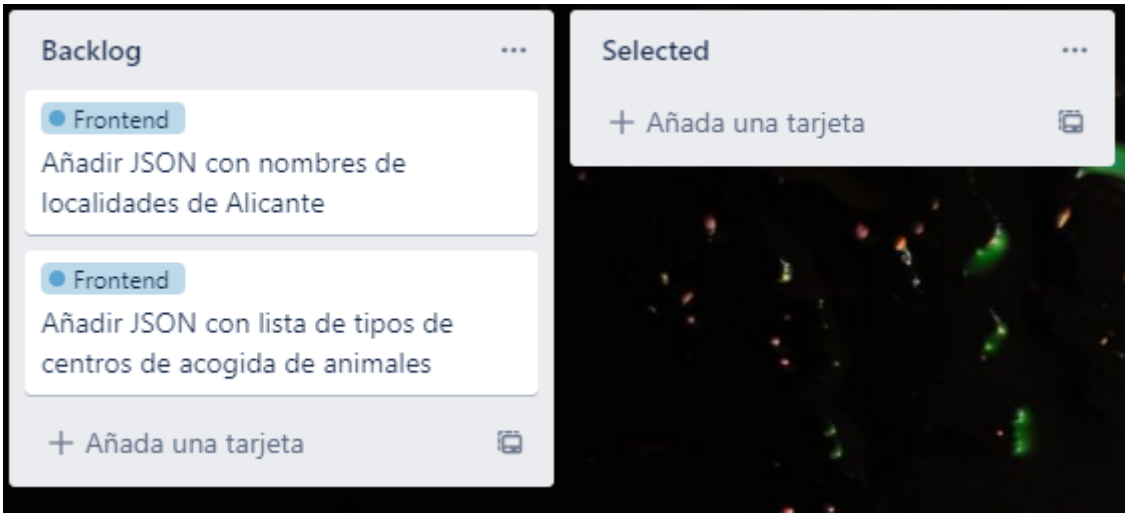


Figura 16. Columnas Backlog y Selected del tablero de Trello. (Fuente propia)

Una vez la tarea era llevada a cabo pasaba a la columna In Progress, pudiendo pasar a Blocked si por algún motivo no se podía avanzar con ella por algún imprevisto. En este caso, se centraba la atención en desbloquear la tarea. Si se conseguía finalizar la tarea correctamente, pasaba a On hold, donde se revisaría que realmente se integra correctamente con el sistema y que cumple con los requisitos para considerarse como finalizada, tal y como se había comprobado al final de su desarrollo.

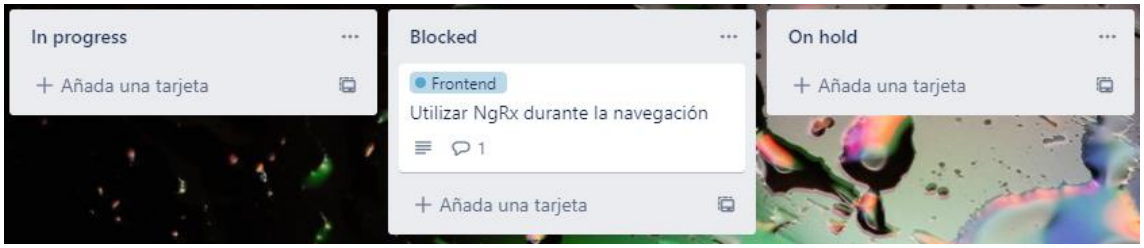


Figura 17. Columnas In Progress, Blocked y On hold del tablaro de Trello. (Fuente propia)

Por último, como paso previo al estado de "finalizada", la tarea pasaba a Review and document. En esta columna la tarea estaba a la espera de ser documentada y revisada de nuevo. Tras haberse documentado los cambios realizados en el proyecto por la tarea, la tarea se consideraba terminada, por lo que pasaba a la columna Done.

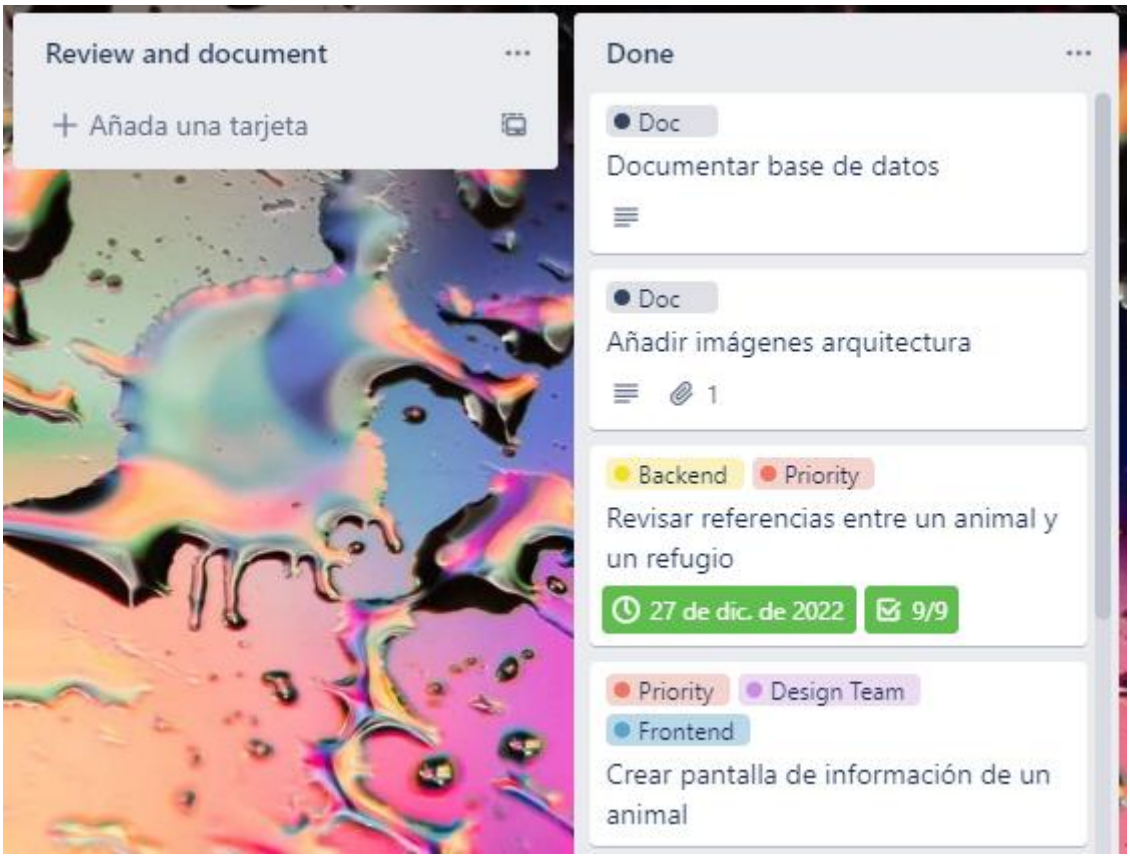


Figura 18. Columnas Review and document y Done del trablero de Trello. (Fuente propia)

Además de las columnas para el seguimiento del estado de las tareas, se han utilizado otras tres para la anotación de información relevante o reflexiones sobre el proyecto.

De esta manera, se ha podido tener algo más centralizada la información relevante que se iba adquiriendo a lo largo del proyecto, mostrándose de manera más visual.

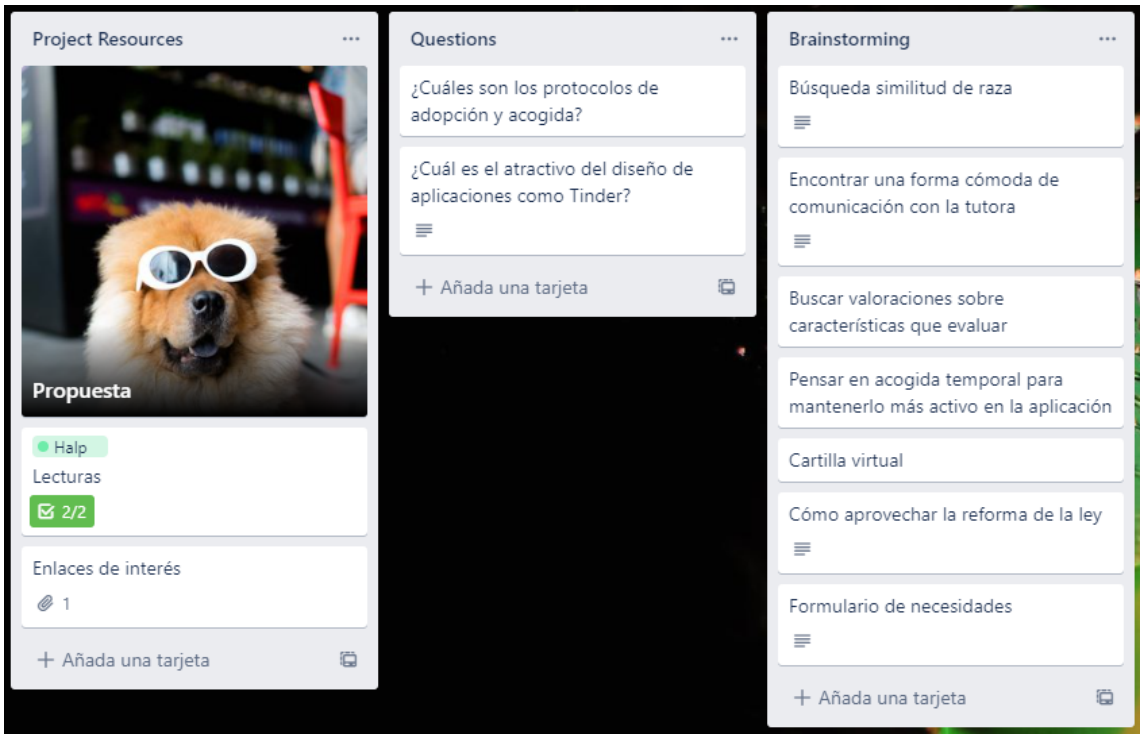


Figura 19. Columnas Project Resources, Questions, Brainstorming del tablero Trello. (Fuente propia)

4. OBJETIVOS

El presente proyecto de Trabajo de Final de Grado (a partir de ahora, TFG) tiene como objetivo desarrollar un sistema multiplataforma que permita la adopción de animales domésticos sin hogar a nivel nacional, aunque comenzando por un alcance local, como podría ser Alicante.

Además, opta por un cambio en la experiencia de usuario interesado en la adopción añadiendo un factor de incertidumbre en la presentación de los animales sin hogar. De esta manera, se cree posible un cambio de perspectiva a la hora de adoptar, más desvinculado al factor físico del animal; no mostrando el catálogo completo de opciones, si no mostrándose individualmente.

De esta forma, se ofrece una experiencia de usuario más satisfactoria, a través de un diseño que opta por la sencillez y la gamificación, y que, a diferencia de las plataformas disponibles con la misma finalidad, permite una mayor atención a los animales ofrecidos en adopción, favoreciendo la creación de un vínculo.

5. CUERPO DEL TRABAJO

5.1. Especificación

Para poder tener una idea aproximada de los objetivos del proyecto, a continuación, se detallan los requisitos del mismo. Esta especificación de requisitos es la base que permite verificar si se alcanzan o no los objetivos establecidos del proyecto, ya que son un reflejo detallado de las necesidades de los clientes.

5.1.1. Tipos de usuario

Esta herramienta ha tenido como objetivo el ofrecer servicio a dos tipos de usuario: la persona que quiere encontrar un animal que poder adoptar y la persona o equipo que gestiona o trabaja en cualquier tipo de refugio con animales domésticos. Estos dos perfiles han sido utilizados como rol dentro de la aplicación, otorgándosele a cada uno una serie de permisos y accesos a las funcionalidades del sistema.

Seguidamente, se detalla cada uno de estos roles que aparecen en la aplicación.

5.1.1.1. Usuario

Desde este tipo de perfil se reciben y visualizan las tarjetas de los animales en adopción que concuerden con los valores de preferencia que ha introducido al registrar su cuenta.

Funcionalidades

Tiene acceso a los detalles del animal que se le presente para consultarlos, además de poder mostrar interés en la adopción o no. Por supuesto, puede editar los datos personales de la cuenta y sus valores de preferencia, así como darse de baja del sistema.

5.1.1.2. Refugio

Este tipo de cuenta será desde donde se alimente el sistema y se gestionen los detalles de los animales y las solicitudes de adopción enviadas por las cuentas de tipo “usuario”.

Funcionalidades

Tiene acceso a un panel de control para la gestión de los animales en el que puede dar de alta a un nuevo animal para ofrecerlo en adopción, editar sus detalles y darlo de baja. Además, desde este panel de control se confirma, o no, a la persona que ha mostrado interés en la adopción de un animal, que su perfil puede ser válido para la adopción.

Obviamente, tendrá pleno control de sus datos, por lo que podrá editarlos o darse de baja del sistema.

5.1.2. Requisitos

Los requisitos son una parte crítica en un proyecto de desarrollo de software, ya que con ellos se definen las funcionalidades y el propósito del proyecto. Simplemente, ofrecen una descripción de lo que la aplicación debe hacer, actuando como pautas para el desarrollo y creación de un producto funcional. (Northware, 2022)

Por lo que, a continuación, se han definido tanto requisitos funcionales como no funcionales sobre el proyecto.

5.1.2.1. Requisitos funcionales

Con los siguientes requisitos funcionales se quiere definir las características que el usuario detecta en el sistema. (PMOinformatica, 2017) (Visure Solutions, s.f.)

Identificador: RF1	**Perfiles:** Usuario y Refugio
Nombre	Creación de cuenta o registro
Descripción	Para acceder a la aplicación, se deberá registrar un usuario mediante el formulario de registro al que se tiene acceso desde la pantalla de inicio de sesión. Para la creación de una cuenta de tipo "Usuario", únicamente hará falta un correo electrónico válido y una contraseña. Para obtener un usuario de tipo "Refugio" se deberá contactar con el administrador a través del correo de contacto, facilitando un correo válido, al que se le comunicará el acceso a la plataforma cuando se dé de alta en ella; el CIF y la razón social.

Tabla 1. RF1 – Creación de cuenta o registros.

Identificador: RF2	**Perfiles:** Usuario y Refugio
Nombre	Inicio de sesión
Descripción	Una vez se haya registrado la cuenta en el sistema, se podrá acceder a la plataforma a través del correo electrónico y la contraseña introducida en el registro, o la ofrecida a través de un correo electrónico por parte del administrador cuando se trate de una cuenta de "Refugio".

Tabla 2. RF2 - Inicio de sesión.

Identificador: RF3	**Perfiles:** Usuario
Nombre	Introducción de datos de preferencia
Descripción	Cuando no se tiene la información de preferencia de adopción con la que poder mostrar animales en adopción lo más semejantes a estos datos, se pide que se introduzcan estos datos para ofrecer el mejor servicio posible.

Tabla 3. RF3 – Introducción de datos de preferencia.

Identificador: RF4.1	**Perfiles:** Usuario y Refugio
Nombre	Consulta de datos de la cuenta
Descripción	Accediendo a la página de la cuenta, se pueden consultar los datos que la plataforma tiene almacenados sobre la cuenta.

Tabla 4. RF4 -Consulta de datos de la cuenta.

Identificador: RF5	**Perfiles:** Usuario
Nombre	Presentación de la tarjeta del animal
Descripción	Como principal *layout* de la aplicación, se muestra una tarjeta con la foto y la información más relevante del animal en cuestión.

Tabla 5. RF5 – Presentación de la tarjeta del animal.

Identificador: RF6	**Perfiles:** Usuario
Nombre	Consulta de la información detallada del animal presentado
Descripción	Para conocer más detalles sobre el animal que se presenta en la tarjeta, clicando sobre esta, se puede consultar toda la información proporcionada por el refugio que le ha dado de alta en el sistema.

Tabla 6. RF6 - Consulta de la información detallada del animal presentado.

Identificador: RF7	**Perfiles:** Usuario
Nombre	Confirmación de interés por la adopción del animal
Descripción	Se pulsa el botón de confirmación de interés por adoptar al animal presentado para hacer saber al refugio de esto.

Tabla 7. RF7 – Confirmación de interés por la adopción del animal.

Identificador: RF8	**Perfiles:** Usuario
Nombre	Rechazo posibilidad de muestra interés por la adopción
Descripción	Se pulsa el botón de rechazo para no mostrar interés por la adopción del animal presentado.

Tabla 8. RF8 - Rechazo posibilidad de muestra interés por la adopción.

Identificador: RF9.1	**Perfiles:** Refugio
Nombre	Creación ficha animal en adopción
Descripción	Desde el panel de control accesible para este tipo de perfiles, pulsando en el botón de crear e introduciendo los datos requeridos en el formulario de alta del animal, se guarda la información sobre el animal en la base de datos del sistema.

Tabla 9. RF9.1 - Creación ficha animal en adopción.

Identificador: RF9.2	**Perfiles:** Refugio
Nombre	Consulta de animales pertenecientes al refugio
Descripción	Como vista principal desde el panel de control, se muestra un listado con los animales que pertenecen al refugio desde el que se ha iniciado sesión.

Tabla 10. RF9.2 - Consulta de animales pertenecientes al refugio.

Identificador: RF9.3	**Perfiles:** Refugio
Nombre	Edición de la ficha de un animal
Descripción	Pulsando en el botón de editar, en el listado de animales, podrán editarse los campos rellenados en la creación del mismo.

Tabla 11. RF9.3 - Edición de la ficha de un animal.

Identificador: RF9.4	**Perfiles:** Refugio
Nombre	Dada de baja un animal
Descripción	Pulsando en el botón de dar de baja, el animal es dado de baja, dejando de aparecer en el listado de animales pertenecientes al refugio.

Tabla 12. RF9.4 - Dada de baja un animal.

Identificador: RF10	**Perfiles:** Refugio
Nombre	Consulta de solicitudes de adopción
Descripción	Desde el panel de control de los refugios, se podrá consultar qué animales tienen solicitudes de adopción por parte de los usuarios de la aplicación.

Tabla 13. RF10 - Consulta de solicitudes de adopción.

Identificador: RF11	**Perfiles:** Refugio
Nombre	Confirmación de la solicitud de adopción
Descripción	Desde la página de solicitudes de adopción, se confirma al usuario de que el refugio ha dado el visto bueno para la adopción.

Tabla 14. RF11 - Confirmación de la solicitud de adopción.

Identificador: RF12	**Perfiles:** Refugio
Nombre	Rechazo de la solicitud de adopción
Descripción	Desde la página de solicitudes de adopción, se rechaza la solicitud de adopción del usuario.

Tabla 15. RF12 - Rechazo de la solicitud de adopción.

5.1.2.2. Requisitos no funcionales

Identificador: RNF1	**Nombre:** Usabilidad
Descripción	El diseño de la aplicación, tanto para web como para móvil o tablet, seguirá patrones de diseño que prometan una navegación cómoda e intuitiva por la aplicación en cada una de las plataformas utilizables.

Tabla 16. RN1 - Usabilidad.

Identificador: RNF2	**Nombre:** Disponibilidad
Descripción	El acceso a la aplicación y a los datos de la base de datos del sistema estará disponible 24 horas al día, 7 días a la semana a través de un servicio de *hosting* y de base de datos en la nube.

Tabla 17. RNF2 - Disponibilidad.

Identificador: RNF3	**Nombre:** Recuperación
Descripción	En caso de la existencia de una o más incidencias en la plataforma, con un horario acotado a una ventana desde las 7:00 hasta las 19:00, serán resultas en el menor tiempo posible. El equipo de desarrollo y mantenimiento del servicio se compromete a que, dependiendo de su complejidad, las incidencias serán resultas en: • 1 hora si la incidencia es leve • 3 horas si la incidencia es severa • 6 horas si la incidencia es crítica

Tabla 18. RNF3 - Recuperación.

Identificador: RNF4	**Nombre:** Escalabilidad
Descripción	Para hacer frente a un aumento de demanda, primero se hará uso de un servicio escalable horizontalmente, mediante redundancia de servidores, por ejemplo. De todas formas, se contempla la posibilidad de un escalado vertical para favorecer el procesamiento de los datos y aumentar la eficiencia en la búsqueda de similitud de características de los animales con los de las preferencias de adopción del usuario.

Tabla 19. RNF4 – Escalabilidad.

Identificador: RNF5	**Nombre:** Concurrencia
Descripción	Ante las elevadas cifras de adopción, la aplicación debe soportar, en un primer momento, hasta 1000 usuarios conectados simultáneamente y haciéndose llamadas a la base de datos.

Tabla 20. RNF5 - Concurrencia.

Identificador: RNF6	**Nombre:** Integridad
Descripción	El modelo de seguridad desarrollado debe estar presente en cada una de las capas del sistema, garantizando el acceso autorizado a la información.

Tabla 21. RNF6 - Integridad.

Identificador: RNF7	**Nombre:** Seguridad
Descripción	El sistema debe exigir características especiales en las contraseñas: mínimo 8 caracteres, símbolos, números, mayúsculas y minúsculas. También, debe garantizar la confidencialidad, integridad y disponibilidad de la información, además de cifrar las comunicaciones de los servicios que puedan estar expuestos en redes públicas.

Tabla 22. RNF7 - Seguridad.

5.2. Diseño

Optimizar la experiencia de usuario debería ser una prioridad para cualquier proyecto en el que se desarrolle un producto. La incorporación del diseño UX en el desarrollo de productos digitales hace que sean más intuitivos y fáciles de usar, lo que se traduce en una experiencia de usuario más positiva, además del aumento de fidelización de clientes. (Barón, s.f.) (Bustamante, 2017) (Value Keep, s.f.)

Para ello, el diseño UX busca métodos para mejorar la satisfacción de los usuarios y lograr una mayor fidelidad de los mismos, siendo los principales:

- Utilidad: satisfacer las necesidades del usuario.
- Facilidad de uso: sin procedimientos que dificulten su uso.
- Accesibilidad: disponible en todas las plataformas y en varios idiomas.
- Atractivo: valorar la interfaz visual, para despertar más interés.

Centrarse demasiado en la funcionalidad y dejar de lado la usabilidad puede aumentar la tasa de abandono del cliente. Un producto funcional no es necesariamente un producto fácil de usar y sensible. Las funcionalidades pueden estar muy bien definidas, pero si el software es confuso y difícil de usar, la experiencia del usuario no será tan positiva como se espera.

Creo que es indiscutible que Tinder ha sido una de las aplicaciones punteras en este campo en los últimos años, el estudio que ha dedicado al desarrollo de una ergonomía en una aplicación en la que no todo el mundo se puede llegar a sentir cómodo y la manera en que ha conseguido materializarlo, le ha llevado a estar en los puestos más altos de las aplicaciones de citas durante largas temporadas.

En este proyecto, se ha buscado lo mismo, actualizar el diseño de un *grid* repleto de animales, difícilmente distinguible de casi cualquier app en la que se ofrezca otro tipo de servicio estrechamente relacionado al consumo. Se ha tenido en cuenta la comodidad de los usuarios en la navegación, respetando también el espacio de decisión ante la adopción de un animal de compañía, ofreciendo un único espacio en el que solo se encuentre el animal en cuestión y la persona interesada en la adopción.

En cuanto a la navegación, se ha planteado un esquema de navegación para asegurar que la usabilidad de la aplicación:

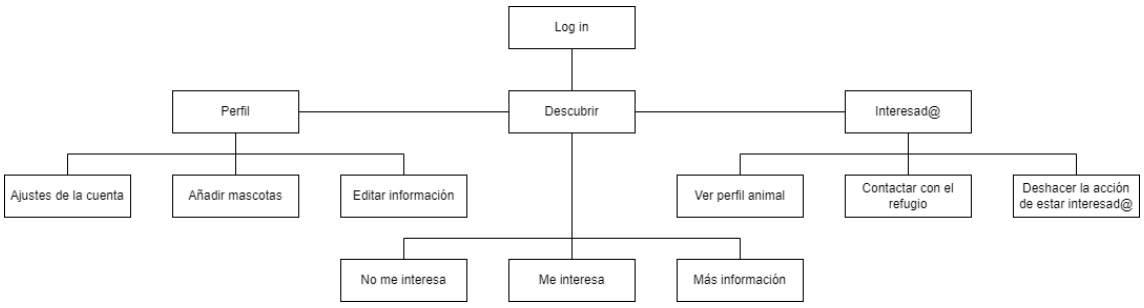


Figura 20. Diagrama de flujo de navegación en aplicación móvil wannAdopt. (Fuente propia)

En el esquema anterior se presentan las acciones que se pueden llevar a cabo por pantalla en la aplicación para los usuarios, donde se ha buscado la menor profundidad posible a través de la intuición en la navegación por las secciones de la aplicación.

De igual forma, se ha realizado un mapa de navegación para la parte del proyecto dedicada a la gestión de los animales en adopción, donde se quiere ofrecer una experiencia de ligereza y sencillez a la hora de gestionar un gran volumen de datos, que es lo que se espera manejar por parte de los refugios.

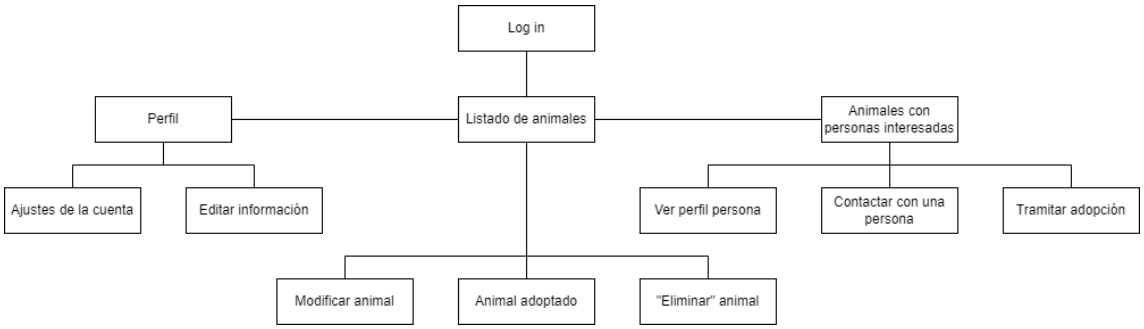


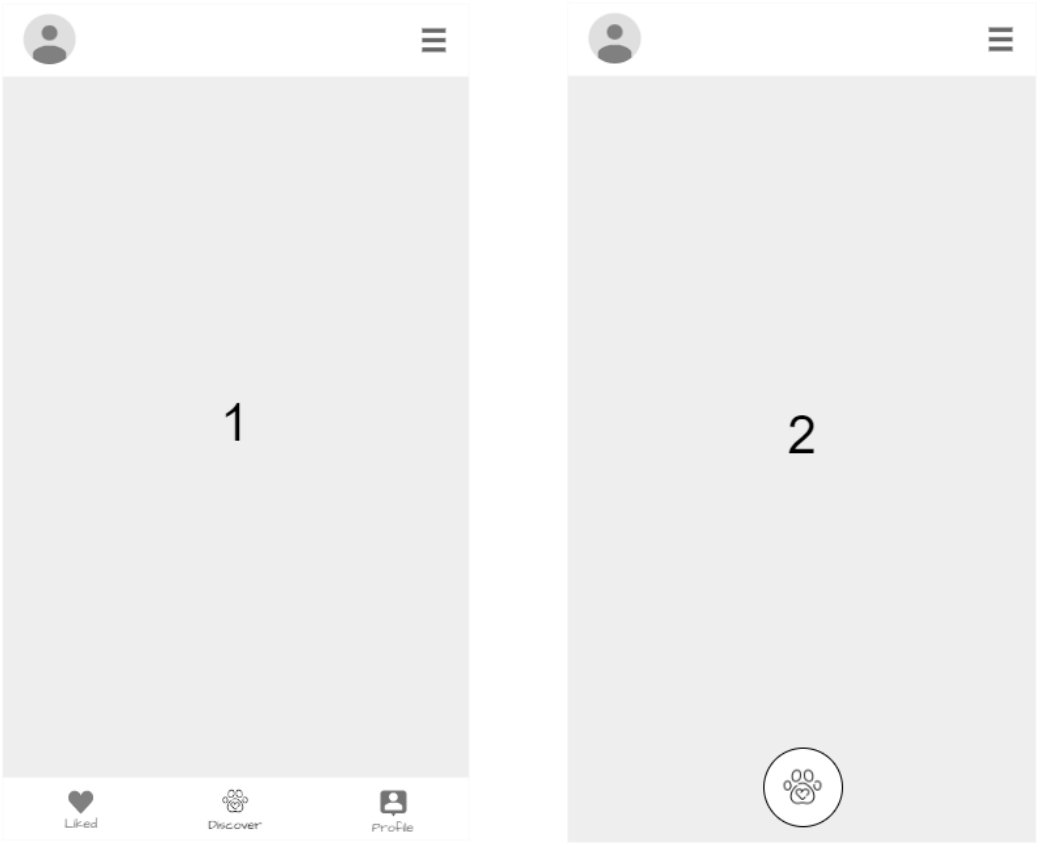
Figura 21. Diagrama de flujo de navegación en portal de gestión wannAdopt. (Fuente propia)

5.2.1. Bocetos

A continuación, se presentan los bocetos o mockups diseñados en un primer momento para alcanzar todos los objetivos propuestos en cuanto a la experiencia de usuario.

5.2.1.1. Layout

En el diseño de la disposición de las pantallas y los elementos de navegación de la aplicación ha habido tres diseños que tendían a un diseño cada vez más minimalista.



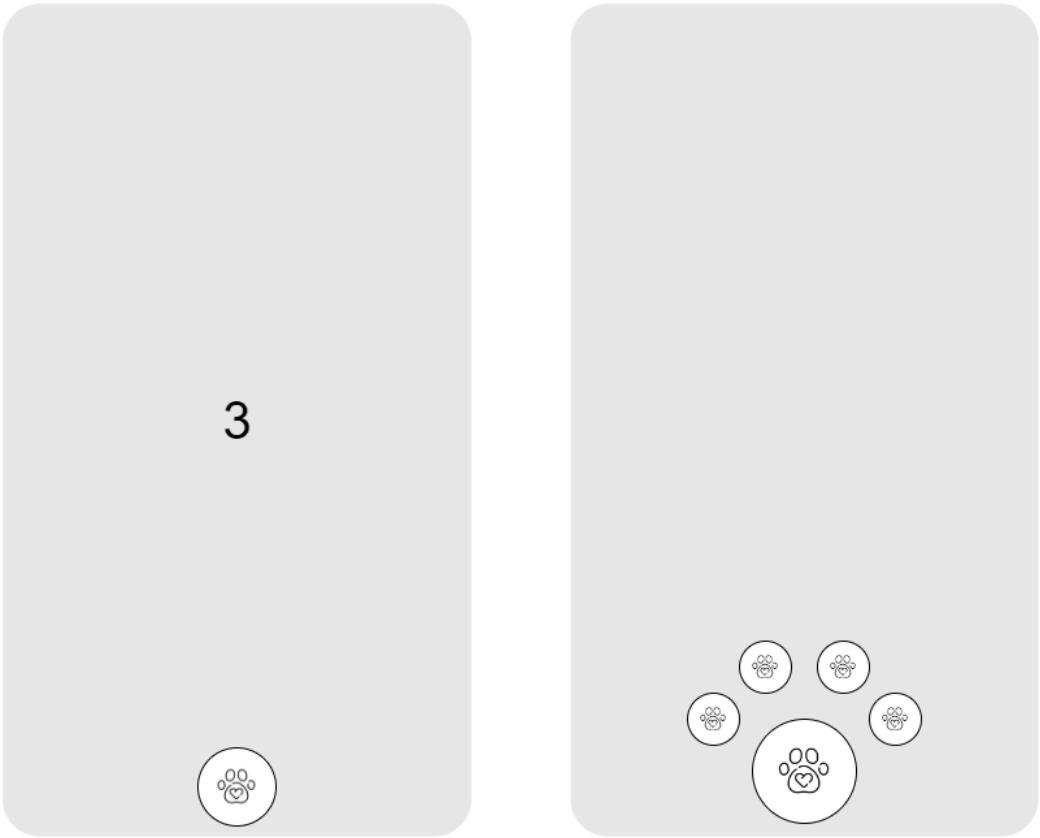


Figura 22. Bocetos layout aplicación móvil.

Tras hacer una encuesta con cinco personas pertenecientes a distintos grupos de edad, se optó por el diseño #1, ya que aparentaba ser el más convincente e intuitivo entre los diferentes participantes de la encuesta. Aunque sí que es cierto que el diseño #3 llamó especialmente la atención a las personas menores de 26 años.

5.2.1.2. Usuario – Registro e inicio de sesión

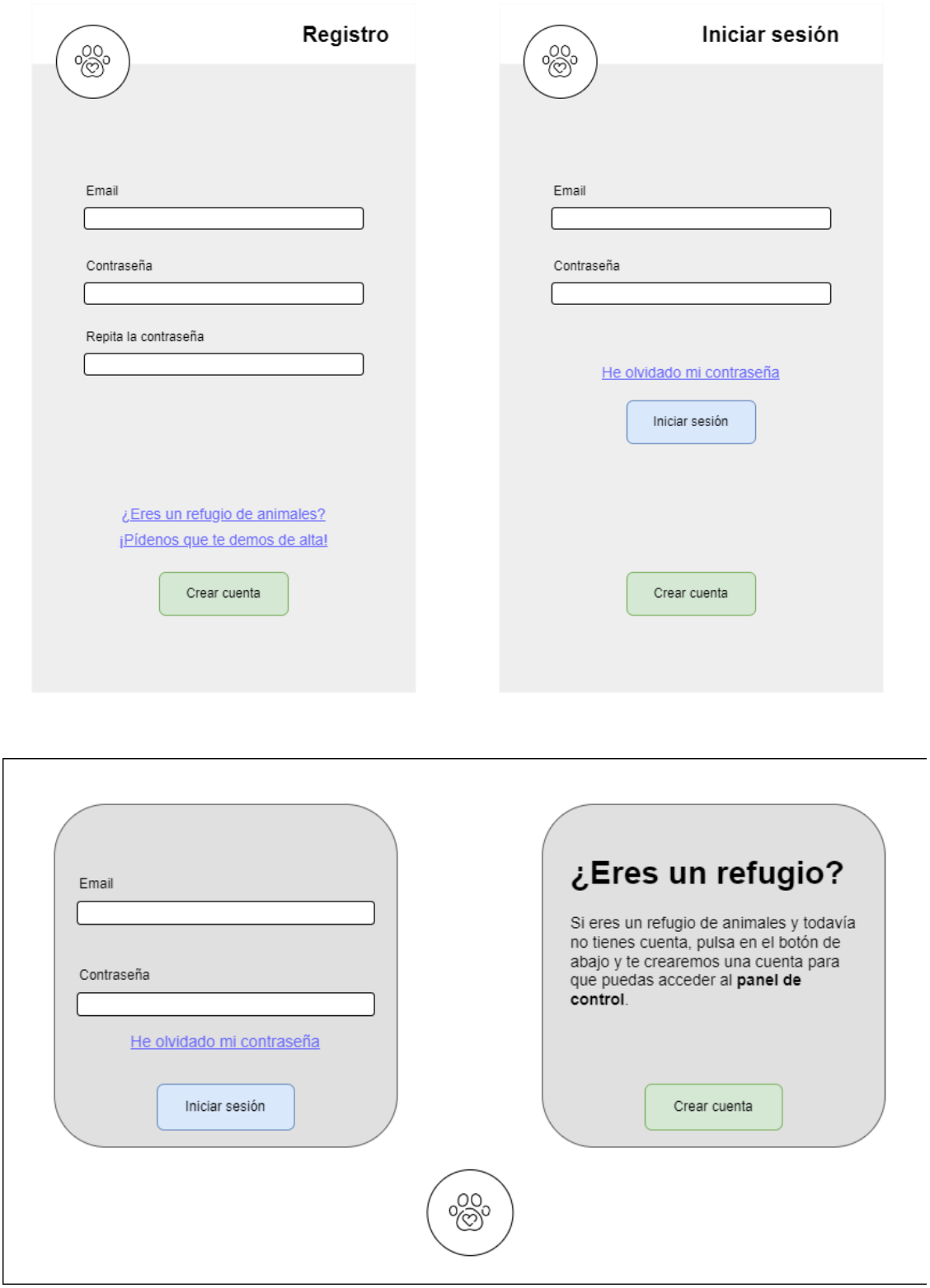


Figura 23. Boceto de pantalla de inicio de sesión y registro. (Fuente propia)

5.2.1.3. Listado de animales desde la herramienta de gestión para refugios

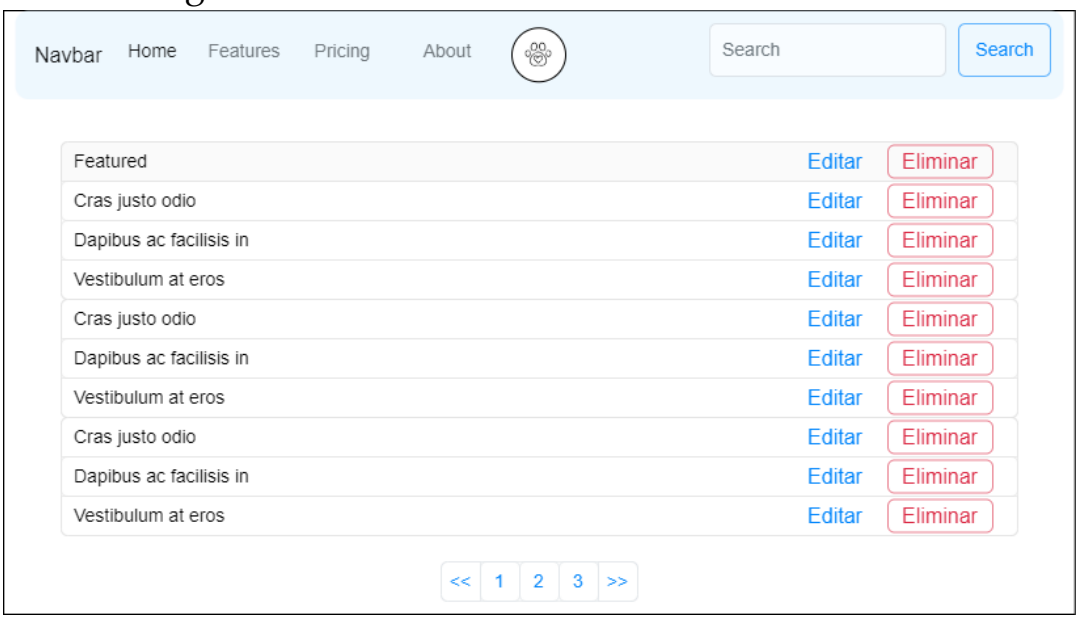


Figura 24. Boceto de listado de animales de un refugio.

5.2.1.4. Usuario – Pantalla de descubrimiento de animales y listado de los que se ha mostrado interés

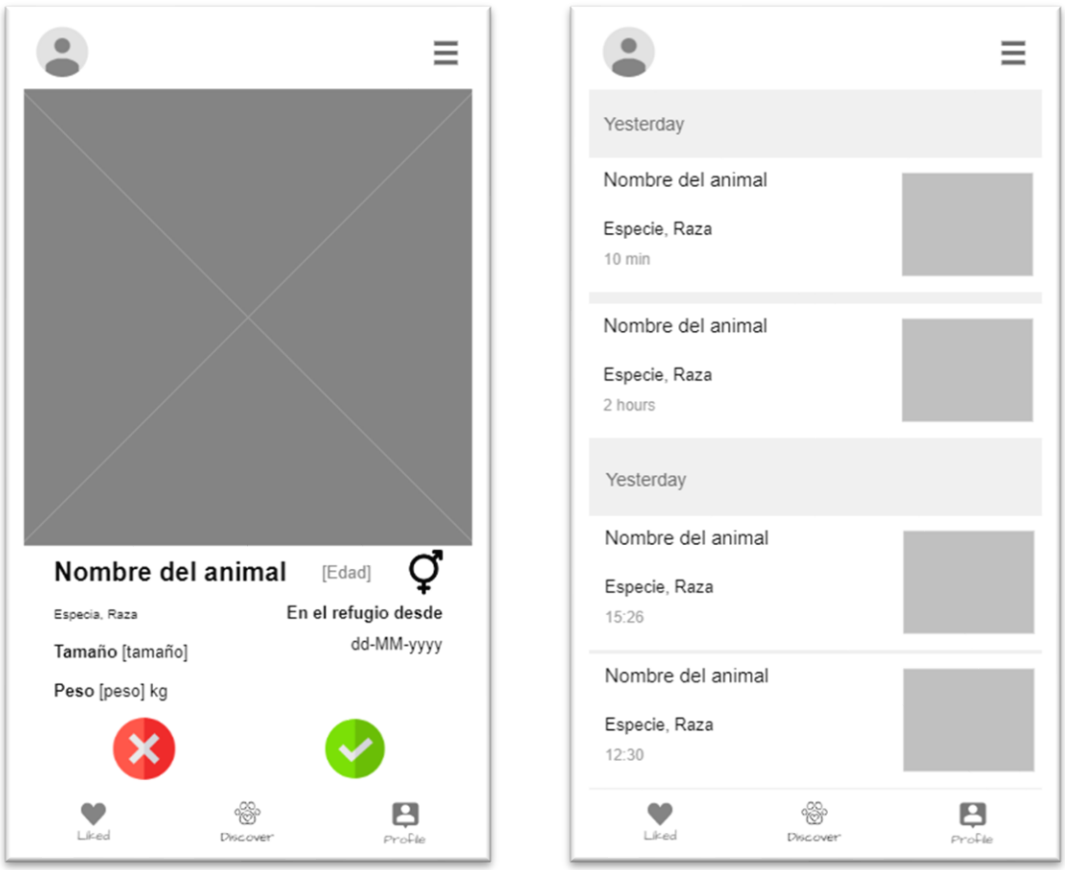


Figura 25. Bocetos de tarjeta de animal y listado de animales en los que se está interesado adoptar.

5.2.2. Navegación

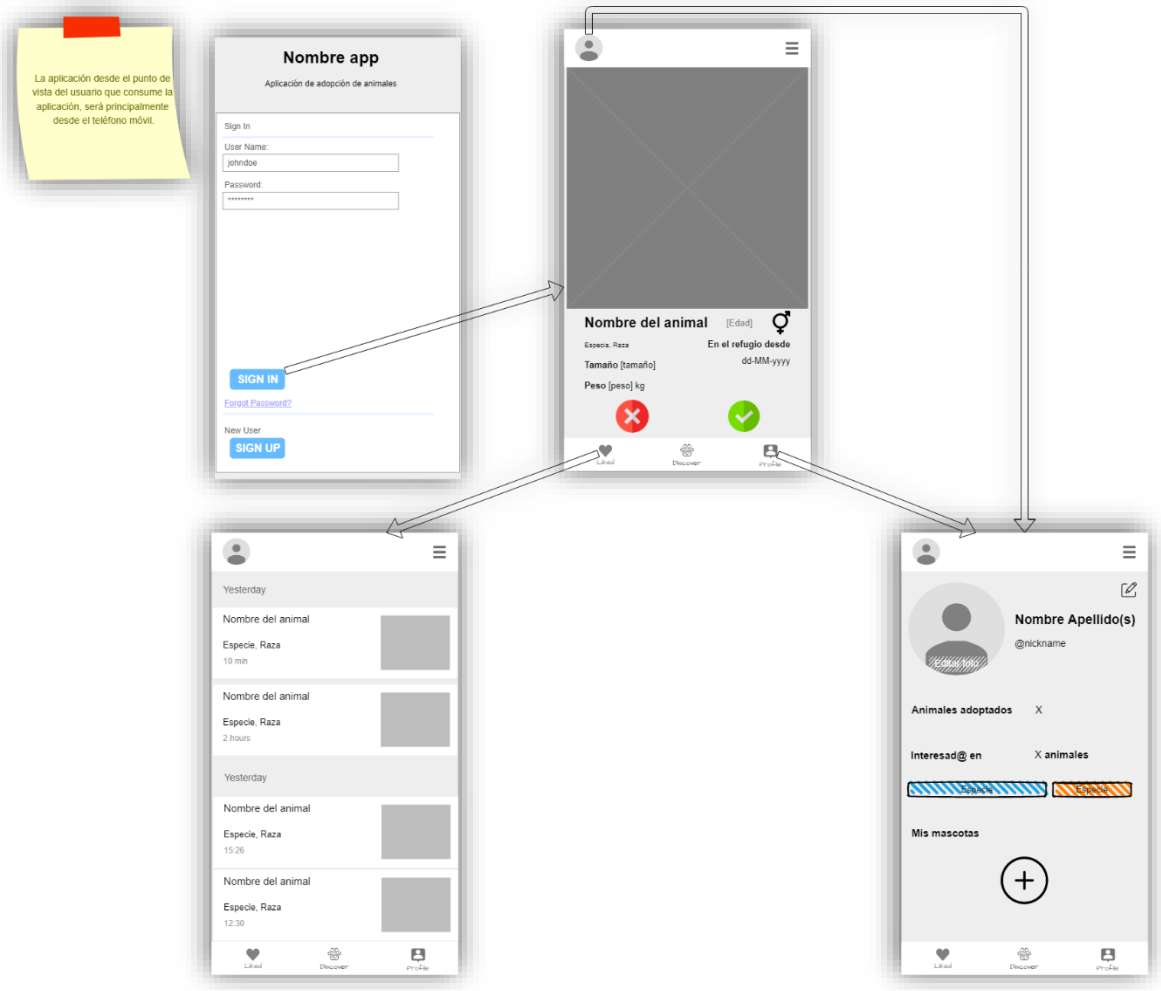


Figura 26. Boceto del flujo de navegación básico entre las pantallas de la aplicación móvil.

5.2.3. Diseño final

Haciendo referencia al primer requisito no funcional "RFN1: Usabilidad", los diseños tanto para web como para móvil, han estado enfocados en proporcionar la mayor adaptabilidad a los diferentes tipos de pantalla. Para conseguir esto, se ha hecho uso de estilos CSS propios y de frameworks CSS como Bootstrap y otras librerías de iconos.

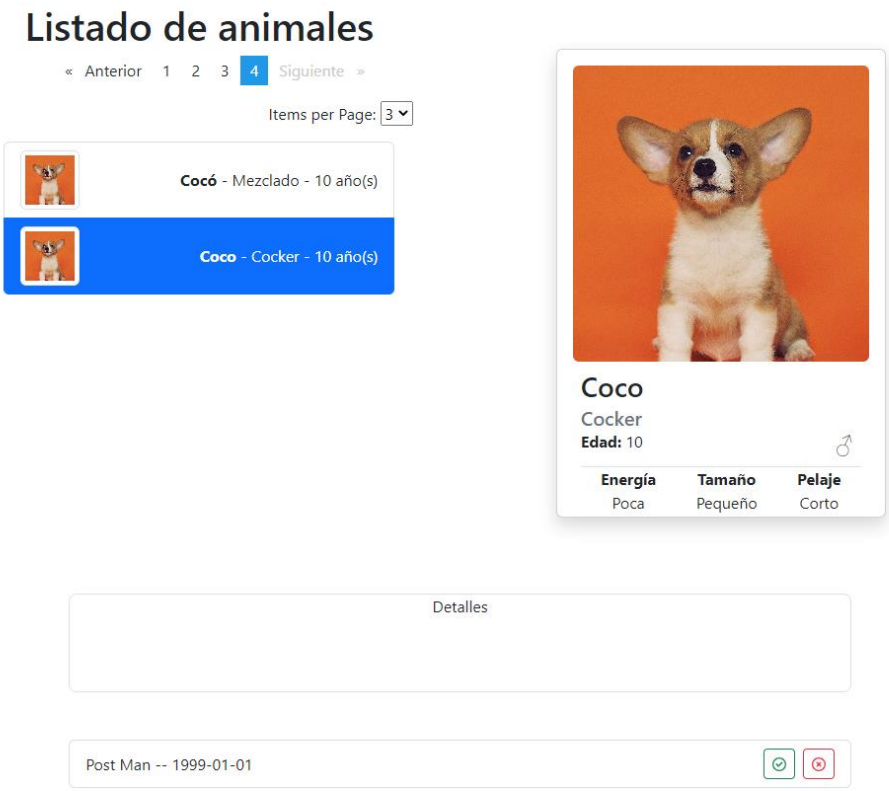


Figura 27. Pantalla de detalles de los animales en portal de gestión wannAdopt. (Fuente propia)

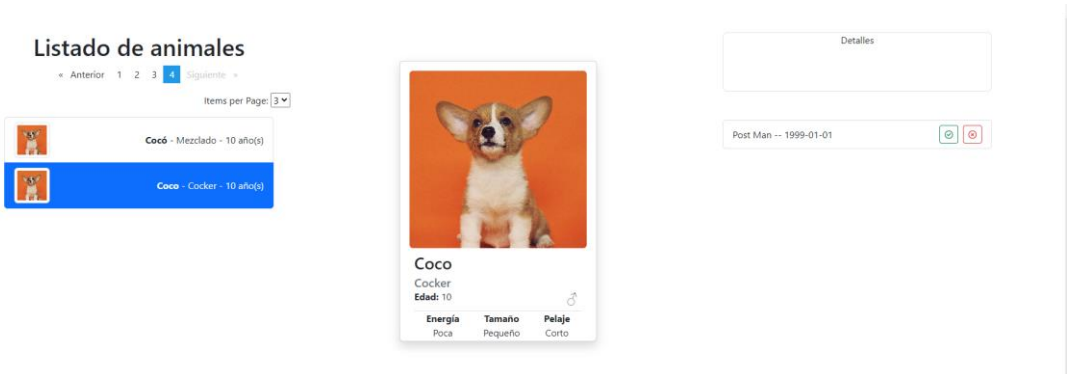


Figura 28. Pantalla de detalles de los animales en portal de gestión wannAdopt. (Fuente propia)

5.2.3.1. Portal de gestión

Este portal de gestión para refugios de animales en adopción consta de siete componentes de AngularJS que permiten la navegación completa por la aplicación.

Comenzando por la pantalla de inicio de sesión, esta pantalla es una bienvenida tanto para los clientes que todavía no pertenecen a la plataforma como para los que ya están en ella. A la izquierda, se muestra un breve resumen sobre el proyecto y la finalidad de la web. A la derecha se sitúa el formulario para introducir las credenciales y acceder a la parte privada de este. La barra de navegación se mantiene en todo momento para dar una mejor imagen en la que reina la coherencia del diseño y la simplicidad.

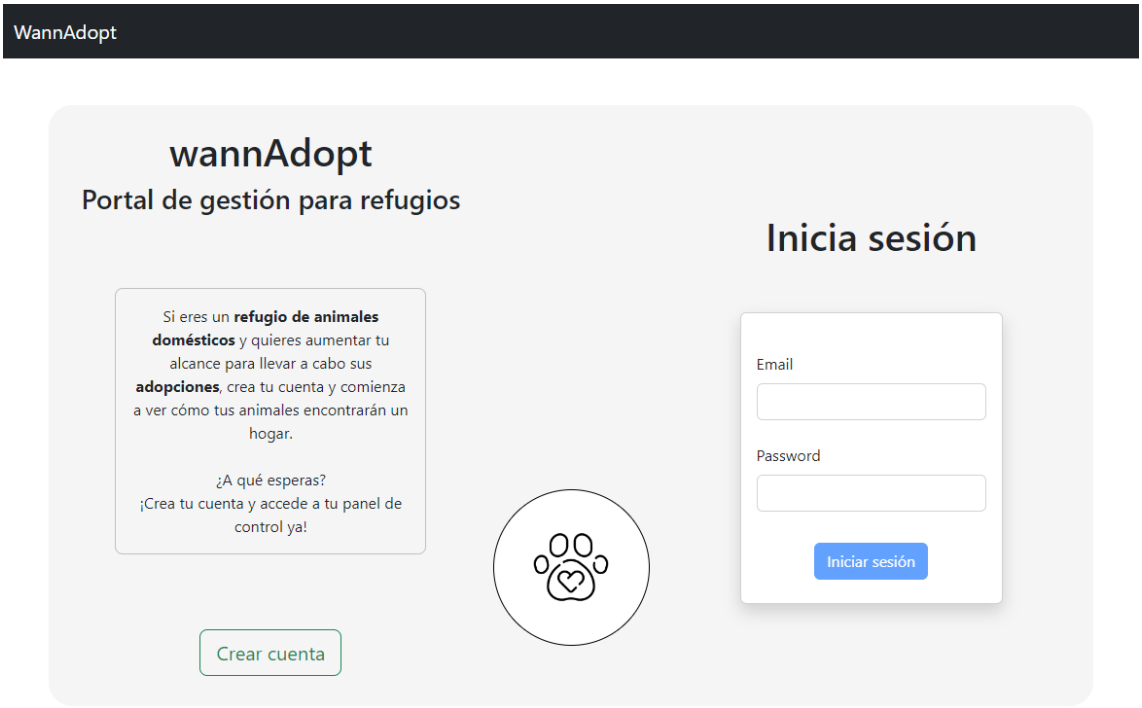


Figura 29. Pantalla de inicio de sesión al portal de gestión wannAdopt. (Fuente propia)

Pulsando al botón de "Crear cuenta", se accede a una vista del componente con el formulario para registrarse. Únicamente hace falta introducir un correo electrónico y una contraseña que serán utilizadas para el inicio de sesión a la plataforma.

Una vez en la parte privada del portal, se muestra un listado a la izquierda de los animales dados de alta por el refugio en la plataforma. Este listado es navegable a través del uso de la paginación y el tamaño de la lista es variable.

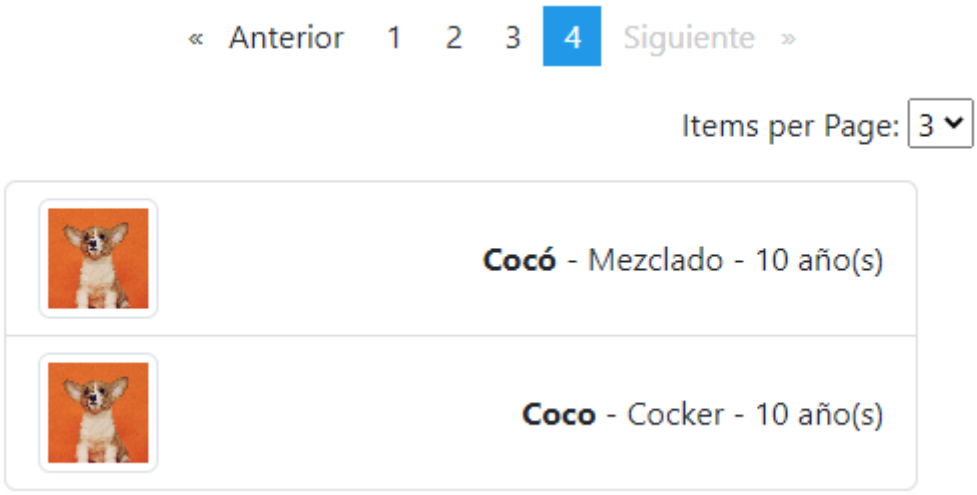


Figura 30. Listado de animales pertenecientes a un refugio. (Fuente propia)

Pulsando en cualquiera de los elementos listados, se mostrarán los detalles del animal a la derecha del listado en forma de tarjeta, la misma que se presenta desde la aplicación móvil.

Figura 31. Tarjeta con detalles del animal. (Fuente propia)

A la vez, se mostrará el listado de solicitudes de adopción vinculadas al animal para que puedan revisarse y dar respuesta a si, con los datos presentados sobre la persona interesada, se cree que puede ser alguien apto o no para la adopción de ese animal.

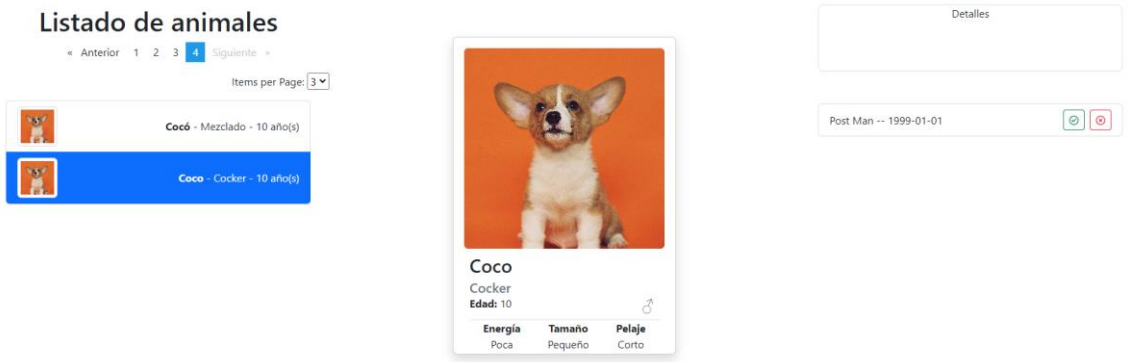


Figura 32. Pantalla de detalles de los animales en portal de gestión wannAdopt. (Fuente propia)

Estas pantallas han sido diseñadas e implementadas asegurando un diseño *responsive* en ordenadores, manteniendo una disposición correcta de los elementos al cambiar el tamaño de la ventana del navegador.

A la hora de acceder al formulario de alta de animales, a través de la barra de navegación se llega activa esta vista.

Es un formulario reactivo en el que conforme se rellenan los datos del animal, la tarjeta también cambia. De esta forma, se puede comprobar con una vista previa de la tarjeta cómo quedaría dispuesta toda la información en la aplicación móvil.

5.2.3.2. Aplicación móvil

El diseño de estas pantallas ha sido *mobile first*, ya que la aplicación espera ofrecer una experiencia de usuario rápida y cómoda, propia de las aplicaciones de móvil con las que no llevar a cabo tareas demasiado avanzadas.

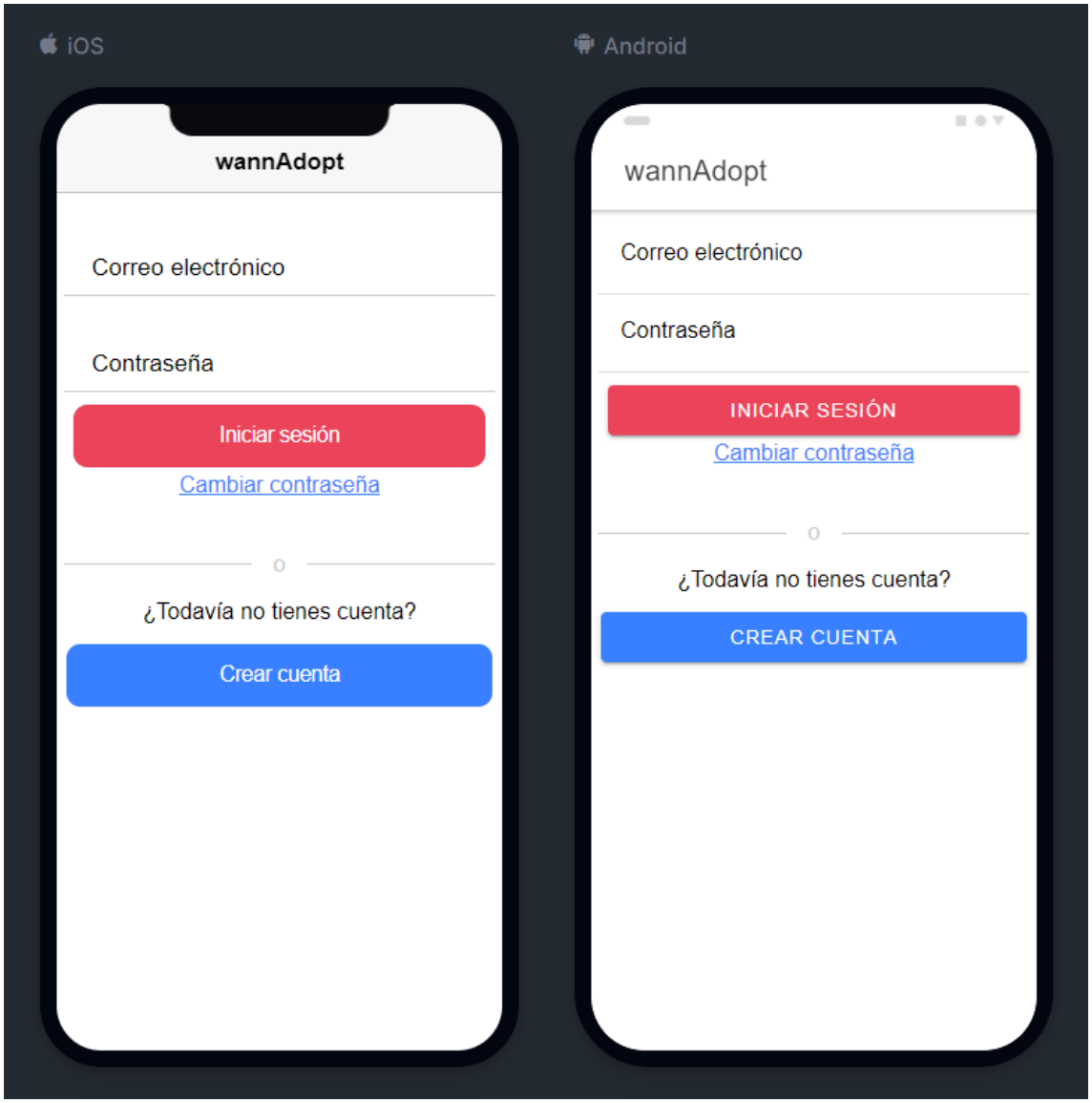


Figura 33. Pantalla de inicio de sesión de la aplicación móvil. (Fuente propia)

Arriba se ve la pantalla de inicio de sesión de la aplicación, dejando ver que las pantallas son válidas tanto para iOS como para Android. De la misma forma que antes, desde aquí se podrá navegar también a la página de registro y poder crear una cuenta introduciendo únicamente un usuario y una contraseña.

Una vez se introducen las credenciales correctamente, la aplicación aterriza en la pantalla principal, la pantalla "Descubrir". Desde esta pantalla se muestran los animales que hay en la base de datos y que puede que sean de interés para el usuario.

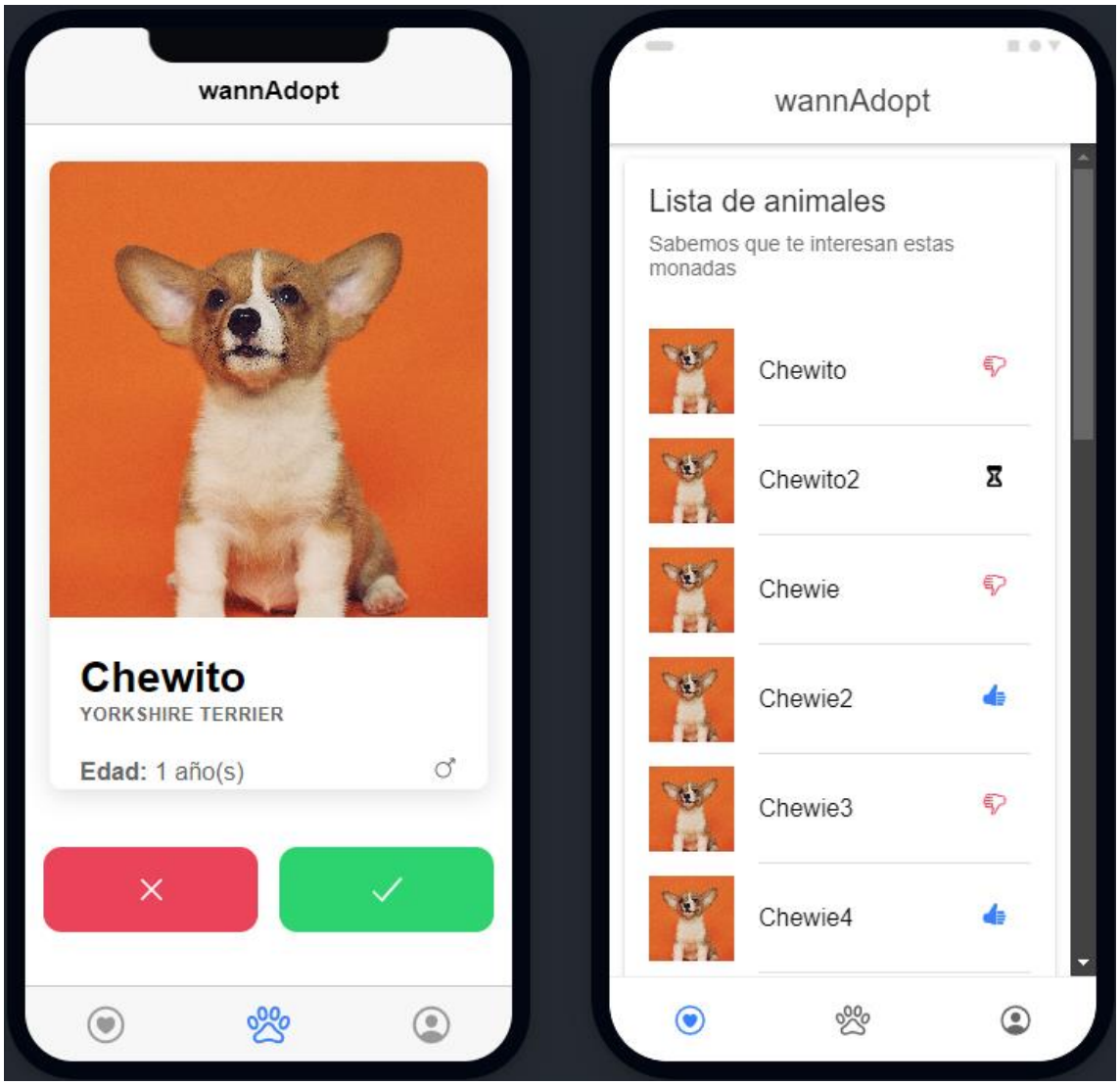


Figura 34. Pantalla Descubre e Interesados de la aplicación móvil wannAdopt. (Fuente propia)

A partir del acceso a la parte privada de la aplicación, la navegación entre las tres pantallas es a través de la barra de navegación inferior que contiene tres iconos:

- Interesados: se accede a través del botón de la izquierda y lleva a una pantalla con el listado de los animales a los que se ha pulsado el botón verde de la pantalla de "Descubrir", acción que muestra interés en la adopción del animal. Además, podrá consultarse su estado, viendo si se ha dado el visto bueno desde el refugio.
- Descubrir: se muestra como pantalla principal y es accesible desde el botón central de la barra de navegación. En esta pantalla se lleva a cabo la decisión de si mostrar interés por la adopción del animal o no.
  Esta pantalla es el factor determinante del proyecto, desde aquí se quiere permitir un espacio único al animal para presentarlo, pudiendo prestarle atención únicamente a él.
- Cuenta: en esta pantalla es posible ver los detalles de la cuenta y cerrar la sesión.

5.3. Arquitectura

En el momento de diseñar la arquitectura de este proyecto, se tenía claro que iba a basarse en una pila MEAN (MongoDB, Express, Angular, Node.js) con la ligera modificación de que los datos serían almacenados en una base de datos SQL, con el gestor de datos MySQL.

Con esta consideración de la pila MEAN, la arquitectura de microservicios es la arquitectura por definición para un proyecto de estas características. La arquitectura de microservicios es un método de desarrollo de aplicaciones que funciona como un conjunto de pequeños servicios que se ejecutan de manera independiente y autónoma, proporcionando una funcionalidad de negocio completa. Los microservicios han creado infraestructuras IT más adaptables y flexibles. Porque si se quiere modificar solamente un servicio, no es necesario alterar el resto de la infraestructura.

Y, por si fuera poco, este tipo de arquitectura ofrece ventajas como la modularidad, al tratarse de servicios autónomos; la escalabilidad, permitiendo escalar horizontalmente del módulo que sea necesario; la versatilidad, adaptándose al uso de diferentes lenguajes y tecnologías entre módulos; la rapidez de actuación, al tener el sistema dividido por módulos; el mantenimiento simple y económico, y la agilidad, al permitir la integración de funcionalidades comunes desarrolladas por terceros.

De todas formas, por motivos de eficiencia en el desarrollo del proyecto, los servicios del back end han quedado encapsulados en un mismo sistema, por lo que deja la arquitectura a medio camino entre un sistema tradicional monolítico cliente-servidor y un sistema basado únicamente en microservicios.

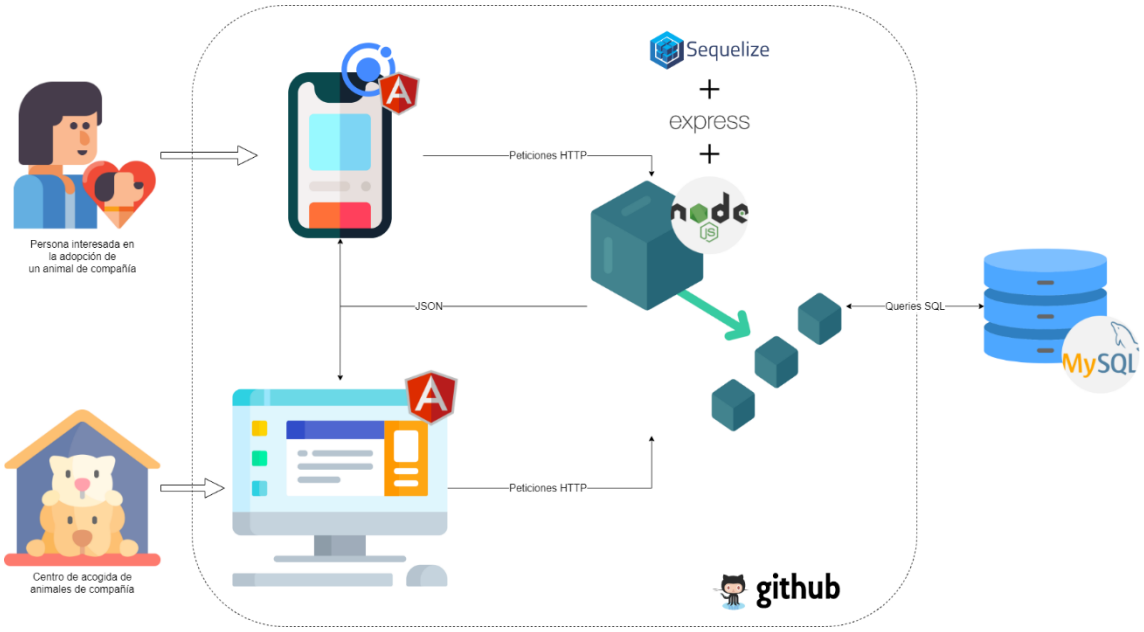


Figura 35. Diagrama de la arquitectura del sistema. (Fuente propia)

5.4. Implementación

Debido a los objetivos que persigue este proyecto, para su desarrollo, se ha visto dividido en tres proyectos que han ido creciendo en paralelo para ofrecer la interacción entre los diferentes sistemas implicados desde las primeras fases del proyecto, permitiendo que se comuniquen entre sí. Por esta razón, el código se encuentra en tres repositorios diferentes:

- Proyecto NPM con código del back end: *wannAdopt-backend*.
- Proyecto Angular con código del front end de la herramienta de gestión para refugios: *wannAdopt-BO-frontend*.
- Proyecto Ionic + Angular con código del front end de la aplicación multiplataforma para la adopción de animales de compañía: *wannAdopt-frontend*.

A través de GitHub se ha seguido el flujo de trabajo de Github Flow, en el que, a partir de una única rama de larga duración, "main", aparecen las ramas en las que se desarrollan las nuevas características para trabajar de forma independiente en ellas. Esto ha permitido asegurar una correcta integración de las nuevas funcionalidades en la versión principal de la aplicación.

5.4.1. Base de datos

5.4.1.1. Tecnología

Entre la opción de elegir el uso de una base de datos SQL, base de datos relacional basada en el almacenamiento y acceso a datos relacionados entre sí, y la de una NoSQL, almacenamiento de datos que no utiliza estructuras tan sencillas (tablas) ni se almacenan sus datos en forma de registros o campos. (Pandora FMS, 2022)

Contemplando las necesidades del proyecto y las características de cada paradigma, se optó por el uso de una base de datos SQL desde un principio.

Entre los diferentes gestores de bases de datos SQL encontramos SQLite, Microsoft SQL Server, Oracle, Microsoft Access, PostgreSQL, entre otros. A la hora de elegir un gestor u otro, se ha tenido en cuenta las ventajas que aporta cada uno a la hora de crear, gestionar y administrar la base de datos, así como de qué forma se almacena y busca la información. (Universitat Oberta de Catalunya, 2018)

Tras esta investigación, se terminó escogiendo MySQL como gestor de base de datos para el proyecto, junto a la herramienta MySQL Workbench 8.0 CE. MySQL es un sistema de base de datos relacional muy popular. Entre sus ventajas están:

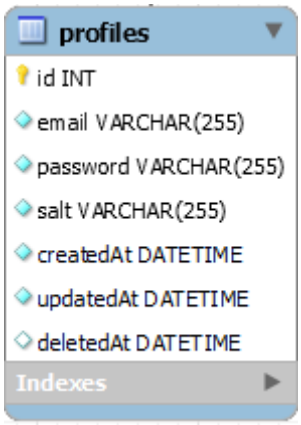
- Su gratuidad: al ser de código abierto, no tiene coste.
- Su facilidad de uso: siendo posible comenzar a usar la base de datos con pocos comandos.
- Su rapidez y rendimiento: siendo estupendo sin necesidad de funcionalidades avanzadas.

- Su seguridad: uso de contraseñas encriptadas, derechos de acceso y privilegios para los usuarios.
- Sus bajos requerimientos y alta eficiencia de memoria: teniendo una baja fuga de memoria y una necesidad de pocos recursos de CPU o RAM.
- Su compatibilidad con Linux y Microsoft.

5.4.1.2. Entidades (tablas)

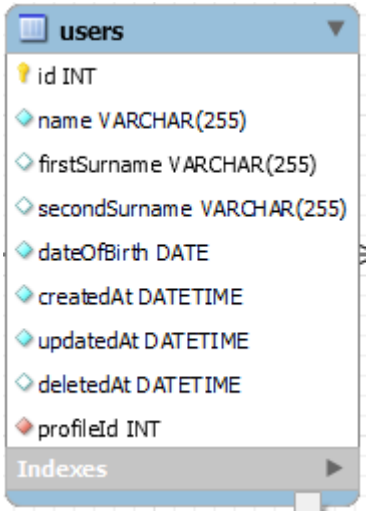
Tras el estudio del modelo de base de datos necesario para el funcionamiento de la plataforma, las entidades identificadas fueron:

- Perfil (profile): Representa la unidad mínima de información de una cuenta, por lo que todas las cuentas tendrán un perfil. Este perfil solo podrá ser para un usuario o para un refugio.



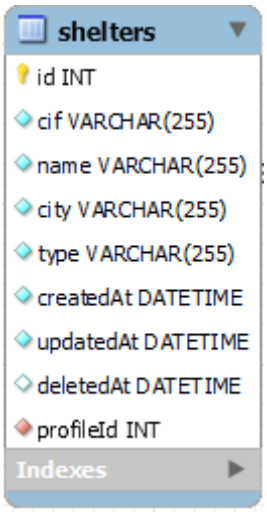
- id → Identificador interno del perfil
- email → Correo electrónico único
- password → Contraseña cifrada
- salt → Valor aleatorio para generar el hash
- createdAt → Momento en que se creó
- updatedAt → Última vez que se modificó
- deletedAt → Momento en el que fue eliminado

- Usuario (user): Información más detallada sobre una persona que se inscribe a la plataforma.



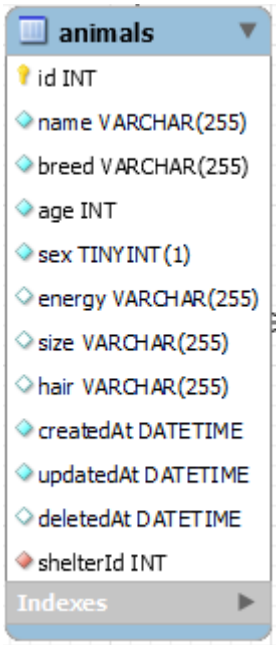
- id → Identificador interno del usuario
- name → Nombre del usuario
- firstSurname → Primer apellido
- secondSurname → Segundo apellido
- dateOfBirth → Fecha de nacimiento
- createdAt → Momento en que se creó
- updatedAt → Última vez que se modificó
- deletedAt → Momento en el que fue eliminado
- profileId → Clave ajena con el id del perfil al que pertenece

- Refugio (shelter): Información más detallada sobre un refugio que se da de alta en la plataforma.



- id → Identificador interno del refugio
- cif → Código de identificación fiscal del refugio
- name → Nombre del refugio o razón social
- city → Ciudad o localidad a la que pertenece el refugio
- type → Tipo de refugio
- createdAt → Momento en que se creó
- updatedAt → Última vez que se modificó
- deletedAt → Momento en el que fue eliminado
- profileId → Clave ajena con el id del perfil al que pertenece

- Animal (animal): Información sobre un animal en adopción.



- id → Identificador interno del perfil
- name → Nombre del animal
- breed → Raza del animal
- age → Edad del animal en años
- sex → Sexo del animal
- energy → Valor de energía del animal
- size → Tamaño del animal
- hair → Tipo de pelaje del animal
- createdAt → Momento en que se creó
- updatedAt → Última vez que se modificó
- deletedAt → Momento en el que fue eliminado
- shelterId → Clave ajena con el id del refugio al que pertenece

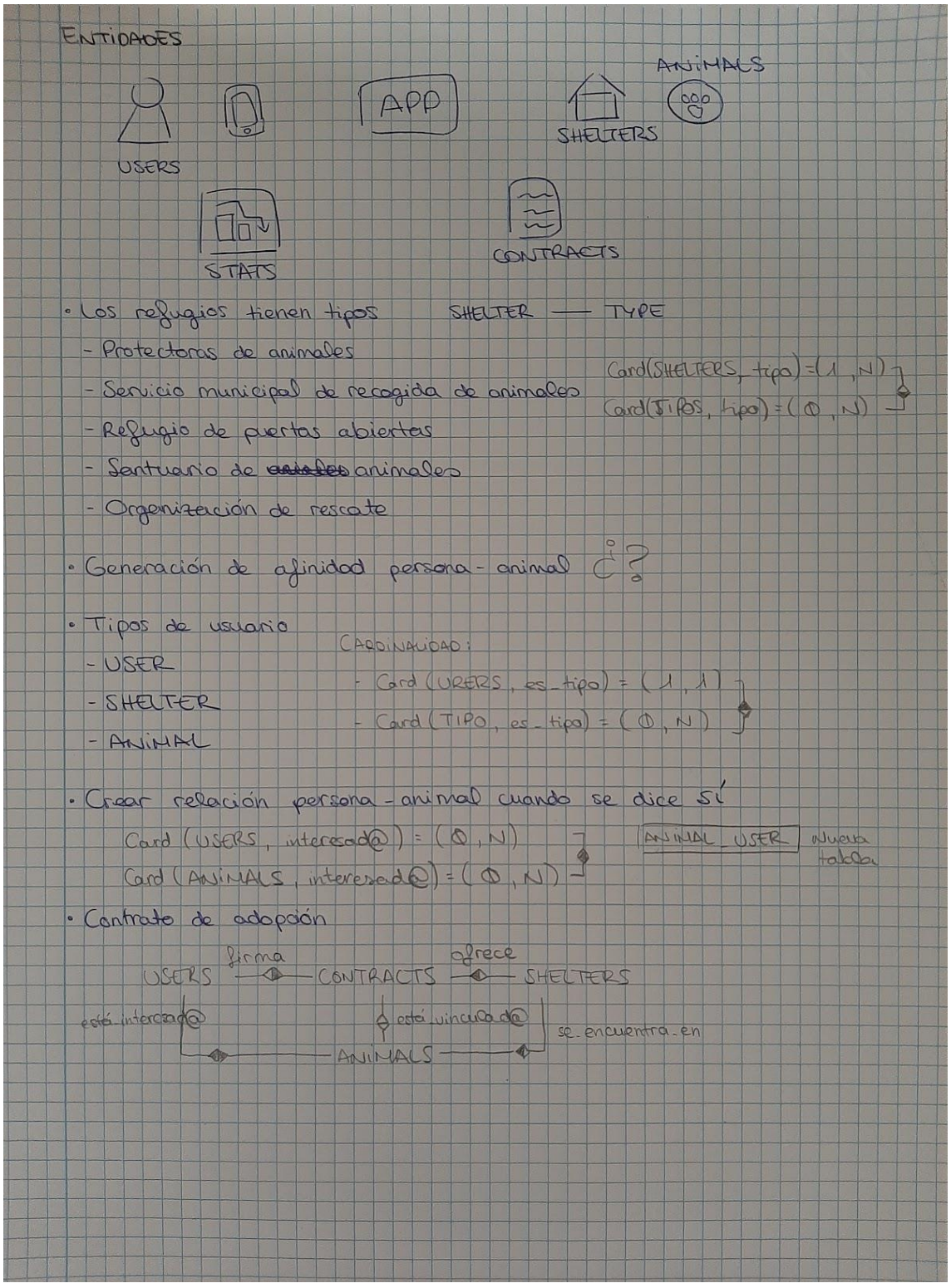
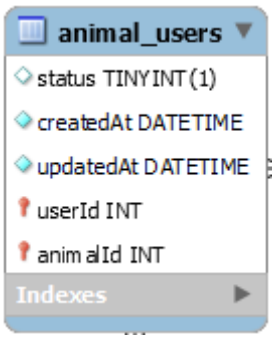


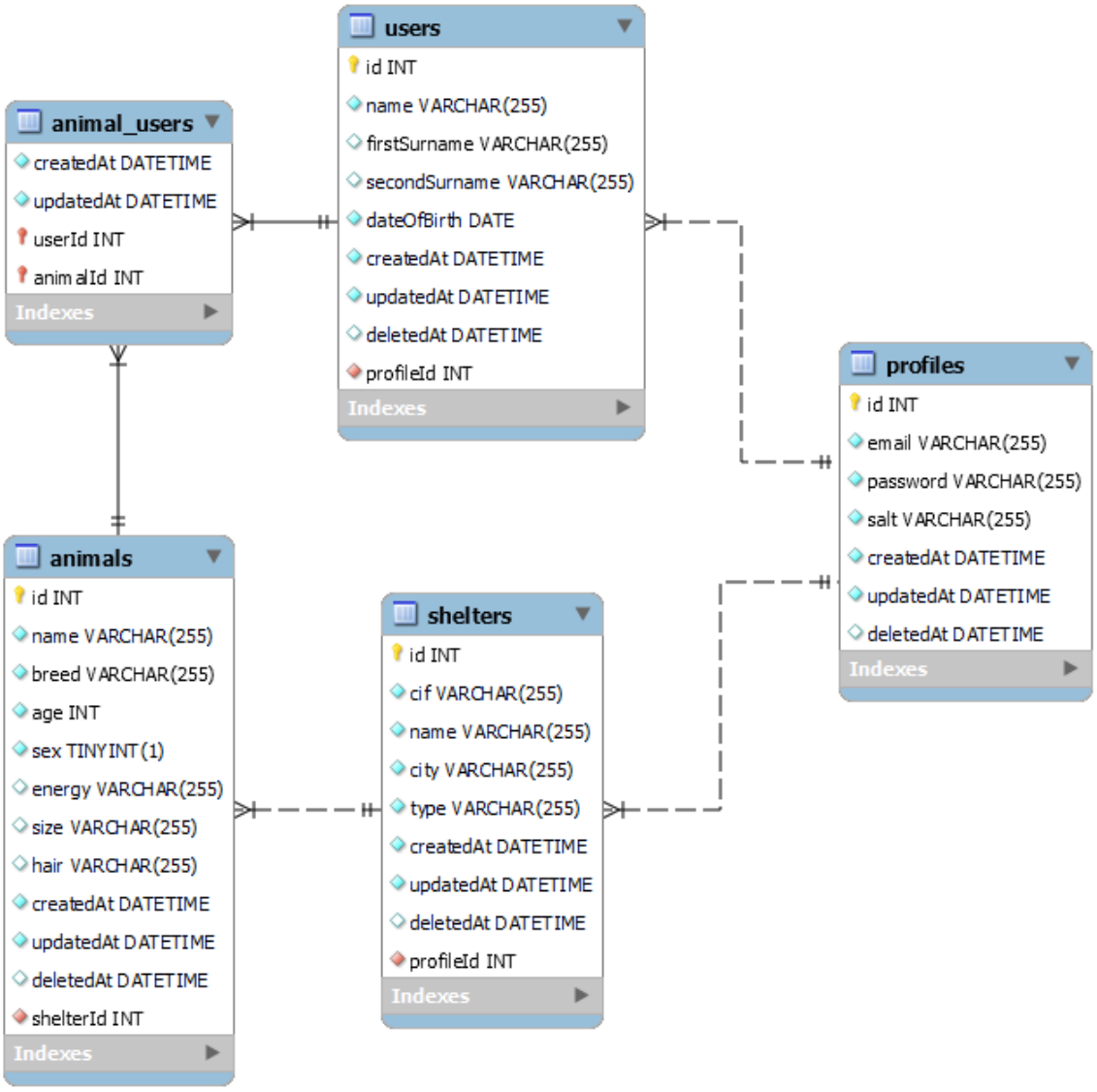
Figura 36. Bocetos hechos a mano sobre las entidades de la aplicación. /Fuente propia)

Estas entidades son las principales del sistema, por lo que siempre se verá involucrada en alguna acción del back end alguna de ellas. Sin embargo, no son las únicas tablas que componen el esquema de la base de datos.



A partir de la relación entre animales y usuarios, para mantener constancia sobre los usuarios interesados en un animal y los animales en los que está cada usuario interesado junto con el estado de la solicitud, marcada como verdadero (aceptada) o falso (rechazada) en la revisión de los refugios.

5.4.1.3. Esquema relacional



5.4.2. Back end

Como se ha comentado anteriormente, para la implementación del servidor se ha empleado Node.js en su versión 16.17.0 y Express como framework para el diseño del back end. Además, para que durante el desarrollo se viesen los cambios que se iban añadiendo al servidor, se ha hecho uso de Nodemon, que relanza la aplicación cada vez que se detecta que un fichero es guardado. Otra librería utilizada durante el desarrollo del back end ha sido Morgan v1.10.0 que muestra las peticiones que recibe el servidor. (Universitat Oberta de Catalunya, 2022) (Kinsta, 2022)

```
const morgan = require("morgan");
app.use(morgan('tiny'));
```

Figura 37. Configuración Morgan.

```
POST /api/auth/login 200 145 - 161.409 ms
Executing (default): SELECT count(*) AS `count` FROM `anim
als` AS `animals` WHERE (`animals`.`deletedAt` IS NULL);
Executing (default): SELECT `id`, `name`, `breed`, `age`,
`sex`, `energy`, `size`, `hair`, `createdAt`, `updatedAt`,
 `deletedAt`, `shelterId` FROM `animals` AS `animals` WHER
E (`animals`.`deletedAt` IS NULL) LIMIT 0, 1;
GET /api/animals?page=0&size=1 304 - - 8.647 ms
```

Figura 38. Logs de Morgan.

5.4.2.1. Integración con la base de datos

En cuanto a la integración con la base de datos, se ha utilizado Sequelize, una herramienta ORM (Object Relational Mapping o Mapeo Objeto-Relacional en castellano) Node.js fácil de usar y basada en promesas para diferentes gestores de bases de datos, entre ellos MySQL. Además, cuenta con un sólido soporte de transacciones, relaciones, carga ansiosa (eager) y perezosa (lazy), replicación de lectura y de más.

Configuración

Para configurar la conexión a la base de datos desde la aplicación Node.js, se ha creado un fichero de configuración llamado db.config.js:

```
module.exports = {
  HOST: process.env.DB_HOST,
  USER: process.env.DB_USER,
  PASSWORD: process.env.DB_PASSWORD,
  DB: process.env.DB_NAME,  // schema
  dialect: "mysql",
  pool: {
    max: 5,
    min: 0,
    acquire: 30000,
    idle: 10000
  }
};
```

Figura 39. Fichero de configuración de la base de datos.

En el fragmento de código anterior se comprueba que los valores sensibles y que deben mantenerse privados se han definido en variables de entorno. Estas variables han sido definidas en el fichero .env del proyecto y se ha utilizado Dotenv para configurar el proyecto y que utilice este fichero. De igual forma, esto nos sirve para no subir información sensible al repositorio, ya que el fichero .env es ignorado por Git.

Conexión

Desde el fichero inicial del proyecto, se ejecuta un comando para sincronizar la aplicación con la base de datos.

```
db.sequelize.sync(/* { force: true } */)
  .then(() => {
    console.log("Synced db.");
  })
  .catch((err) => {
    console.log("Failed to sync db: " + err.message);
  });
```

Figura 40. Código de sincronización de la base de datos.

Modelos

La definición de las entidades y sus modelos para que sean plasmados en la base de datos a través de Sequelize se ha llevado a cabo en los ficheros terminados en .model.js, como por ejemplo:

```
module.exports = (sequelize, Sequelize) => {
  const Animal = sequelize.define('animals', {
    name: {
      type: Sequelize.STRING,
      allowNull: false
    },
    breed: {
      type: Sequelize.STRING,
      allowNull: false
    },
    age: {
      type: Sequelize.INTEGER,
      allowNull: false
    },
    sex: {
      type: Sequelize.BOOLEAN,
      allowNull: false
    },
    energy: Sequelize.STRING,
    size: Sequelize.STRING,
    hair: Sequelize.STRING
  },
    {
      paranoid: true
    });

  return Animal;
};
```

Figura 41. Código del modelo de la entidad Animal.

Relaciones

A través de las estructuras descritas en los modelos para cada entidad, desde el fichero index.js de la carpeta models se describe las relaciones entre las entidades.

```
const db = {};
db.Sequelize = Sequelize;
db.sequelize = sequelize;
db.profiles = require('./profile.model')(sequelize, Sequelize);
db.users = require('./user.model')(sequelize, Sequelize);
db.shelters = require('./shelter.model')(sequelize, Sequelize);
db.animals = require('./animal.model')(sequelize, Sequelize);

// One To One (Profile-User)
db.profiles.hasOne(db.users, {
  foreignKey: {
    allowNull: false,
    unique: true
  },
  onDelete: 'CASCADE',
  onUpdate: 'CASCADE'
});
db.users.belongsTo(db.profiles);

// One To One (Profile-Shelter)
db.profiles.hasOne(db.shelters, {
  foreignKey: {
    allowNull: false,
    unique: true
  },
  onDelete: 'CASCADE',
  onUpdate: 'CASCADE'
});
db.shelters.belongsTo(db.profiles);

// One To Many (Shelter-Animal)
db.shelters.hasMany(db.animals, {
  foreignKey: {
    allowNull: false
  },
  onDelete: 'CASCADE',
  onUpdate: 'CASCADE'
});
db.animals.belongsTo(db.shelters);

// Many To Many (User-Animal)
db.users.belongsToMany(db.animals, { through: 'animal_users' });
db.animals.belongsToMany(db.users, { through: 'animal_users' });
module.exports = db;
```

Figura 42. Código del fichero de carga de modelos y relaciones en la base de datos.

API Sequelize

La verdadera potencia de Sequelize reside en la facilidad con la que hacer operaciones a través de su API, siempre con la oportunidad de poder lanzar nuestras propias sentencias SQL propias.

Paginación

```
// Retrieve all Animals from the database.
exports.findAll = (req, res) => {
  const { page, size } = req.query;
  const { limit, offset } = getPagination(page, size);

  Animal.findAndCountAll({ limit, offset })
    .then(data => {
      const response = getPagingData(data, page, limit);
      res.send(response);
    })
    .catch(err => {
      res.status(500).send({
        message:
          err.message || "Error obteniendo los animales."
      });
    });
};
```

Figura 43. Ejemplo de código sobre paginación.

Así de sencillo sería conseguir todos los animales de la base de datos desde el controlador de animales. Como se puede apreciar, la consulta daría un resultado mediante paginación, siendo el 'offset' el número de líneas que salta tras la consulta y el 'limit' el límite de líneas que devuelve.

Estos campos se calculan de la siguiente manera:

```
const getPagination = (page, size) => {
  const limit = size ? +size : 3;
  const offset = page ? page * limit : 0;

  return { limit, offset };
}
```

Figura 44. Código getPagination.

Otro ejemplo de cómo se han utilizado algunas de las funciones del API de Sequelize es para crear de forma sencilla tuplas relacionadas con elementos de otra tabla:

```
const shelter = shelterResponse.payload;
      // Create an Animal
      const newAnimal = {
        name: req.body.name,
        breed: req.body.breed,
        age: req.body.age,
        sex: req.body.sex,
        energy: req.body.energy,
        size: req.body.size,
        hair: req.body.hair
      };
      const createdAnimal = await
shelter.createAnimal(newAnimal);

      return res.status(201).send(createdAnimal);
```

Figura 45. Código de ejemplo sobre creación de entidad relacionada.

En este fragmento de código de una de las funciones del controlador de animales para dar de alta a uno en la base de datos, se puede comprobar que en la penúltima línea se usa una función a partir del objeto Shelter recuperado de la base de datos que se encarga de crear y relacionar al animal automáticamente.

Con ejemplos como este, se demuestra la claridad que ofrece en la lectura del código Sequelize y su API.

5.4.2.2. Enrutado

Por lo general, el servidor se mantiene a la espera de peticiones HTTP del cliente. En el momento en que se recibe una petición, se debe determinar qué acción debe tomarse según sea el método HTTP con el que se haya enviado la petición (GET, POST, PUT, DELTE, etc.), la URL y la función enlazada en el servidor. Fichero user.routes.js:

```
const { authJwt } = require('../middleware');

module.exports = app => {
  const users = require('../controllers/user.controller');
  var router = require('express').Router();

  // Create a new User
  router.post('/', users.create);

  // Retrieve all Users
  router.get('/', users.findAll);

  // Retrieve a single User with id
  router.get('/:id', users.findOne);
```

```
  // Update a User with id
  router.put('/:id', users.update);

  // Delete a User with id
  router.delete('/:id', users.delete);

  router.post('/animal', [authJwt.verifyToken, authJwt.isUser],
users.linkWithAnimal);

  app.use('/api/users', router);
}
```

Figura 46. Rutas del recurso User.

A través de métodos de Express, se identifica el método utilizado para la petición y el patrón de la URL, y busca cuál de los *endpoints* especificados coincide. De esta forma, se pueden añadir funciones que hagan de middleware para controlar el acceso a estos puntos de la aplicación, como se explicará a continuación.

5.4.2.3. Endpoints

Animal – /api/animals		
Método	**Cadena**	**Descripción**
POST	/	Crea un animal
POST	/:id/requests	Guarda el estado de una solicitud tras la revisión
GET	/	Recupera todos los animales
GET	/shelter	Recupera todos los animales que pertenecen a un refugio
GET	/:id/requests	Recupera las solicitudes de un animal
GET	/:id	Recupera un animal
PUT	/:id	Actualiza un animal
DELETE	/:id	Elimina un animal

Tabla 23. Endpoints Animal.

Auth - /api/auth		
Método	**Cadena**	**Descripción**
POST	/register	Registra un perfil
POST	/login	Inicia sesión
POST	/logout	Cierra sesión

Tabla 24. Endpoints Auth.

Profile - /api/profiles		
Método	**Cadena**	**Descripción**
POST	/	Crea un perfil
GET	/	Recupera todos los perfiles
GET	/users	Recupera todos los perfiles de usuarios
GET	/:id/user	Recupera datos del usuario vinculado al perfil
GET	/shelters	Recupera todos los perfiles de refugios

GET	/:id	Recupera un perfil
PUT	/:id	Actualiza un perfil
DELETE	/:id	Elimina un perfil
DELETE	/	Elimina todos los perfiles

Tabla 25. Endpoints Profile.

Shelter - /api/shelters		
Método	**Cadena**	**Descripción**
POST	/	Crea un refugio
GET	/	Recupera todos los refugios
GET	/:id	Recupera un refugio
PUT	/:id	Actualiza un refugio
DELETE	/:id	Elimina un refugio
DELETE	/	Elimina todos los refugios

Tabla 26. Endpoints Shelter.

User - /api/users		
Método	**Cadena**	**Descripción**
POST	/	Crea un usuario
POST	/animals/:id	Vincula un usuario con un animal como solicitud de adopción
GET	/	Recupera todos los usuarios
GET	/:id	Recupera un usuario
PUT	/:id	Actualiza un usuario
DELETE	/:id	Elimina un usuario

Tabla 27. Endpoints User.

5.4.2.4. Control de acceso a recursos

JWT

JWT (JSON Web Token) es un estándar que está dentro del documento RFC 7519. En el mismo, se define un mecanismo para poder propagar entre dos partes, y de forma segura, la identidad de un determinado usuario, además con una serie de privilegios. (Magaña, 2020)

En la práctica se trata de una cadena de texto que tiene tres partes codificadas en Base64, cada una de ellas separadas por un punto. Estas tres partes son una cabecera (header) donde se indica, al menos, el algoritmo y el tipo de token; el cuerpo (payload) donde aparecen los datos del usuario y privilegios, y demás información que se quiera añadir; y una firma (signature) que nos permite verificar si el token es válido.

Esta última parte es lo que hace a JWT tan útil y potente, dado que podemos verificar si el token ha sido manipulado.

Figura 47. Flujo de un JWT. (Open Webinars, s.f.)

En el servidor, este token es creado en el login y se devuelve al usuario para que lo utilice en las futuras peticiones que se llevarán a cabo al navegar por las aplicaciones.

```
// Create a token
function newToken(id, role) {
  const token = jwt.sign(
    {
      id, // User or Shelter ID
      role
    },
    authConfig.secret,
  );

  return token;
}
```

Figura 48. Función de creación de token.

5.4.3. Front end

Como ya ha sido mencionado anteriormente, en este proyecto quiere ofrecer herramientas tanto para las asociaciones y entidades que se hacen cargo de los animales de compañía sin hogar como de las personas dispuestas a ofrecerles un techo y cuidados estos animales. Por lo que este proyecto consta de dos aplicaciones cliente: el portal de gestión para refugios y la aplicación multiplataforma para usuarios que quieran adoptar.

5.4.3.1. Portal de gestión para refugios

A través del framework de desarrollo de aplicaciones front end, AngularJS, se ha desarrollado esta SPA orientada para su uso a través de navegador web en ordenadores,

por lo que se ha diseñado, principalmente, para este tipo de pantallas que comúnmente se utilizan con estos dispositivos, aunque siempre con una naturaleza *responsive*.

Principalmente, esta aplicación ofrece un lugar donde añadir animales en adopción al sistema, editar, dar de baja o llevar a cabo las confirmaciones de las adopciones, y se compone de diferentes vistas básicas que permiten realizar las funciones CRUD (Create, Read, Update, Delete) de manera visual y cómoda.

Servicio de eventos

Este servicio contiene funciones a las que poder suscribirse y otras con las que lanzar un evento que hará que los componentes en escucha lancen acciones. Esto permite tener una comunicación entre componentes y modificar variables de un componente a partir de acciones en otro.

Se han implementado dos usos en este servicio, el de avisar de que se ha iniciado o cerrado sesión correctamente suscribiéndose a la asignatura de *login* y el de avisar de que se ha enviado la revisión de una solicitud al servidor.

El primer uso es muy útil para controlar el contenido que se quiere mostrar en la barra de navegación. El segundo uso permite actualizar la lista de solicitudes de un animal cuando una de ellas es revisada. Esto hace que sea visible la manipulación de los datos en tiempo real de cara al usuario.

```
private loginSubject = new Subject<any>();
  private reviewSubject = new Subject<any>();
  constructor() { }

  /* LOGIN */
  publishLogin(): void {
    this.loginSubject.next(true);
  }

  publishLogout(): void {
    this.loginSubject.next(false);
  }

  getLoginObservable(): Subject<any> {
    return this.loginSubject;
  }

  /* REVIEW */
  publishReview(): void {
    this.reviewSubject.next(null);
  }

  getReviewObservable(): Subject<any> {
    return this.reviewSubject;
  }
```

Figura 49. Código TypeScript del servicio de eventos.

Servicio de almacenamiento

Este servicio es la última capa entre el almacenamiento de la sesión del navegador y la aplicación. Este almacenamiento es utilizado para guardar el token de sesión que es recibido tras iniciar sesión y con el que se valida que las peticiones son de un usuario autorizado a hacerlas.

Desde este servicio se limpia el almacenamiento, se guarda el token, se lee y se comprueba si el usuario ha iniciado sesión.

```
clean(): void {
    window.sessionStorage.clear();
  }

  public saveLoginData(loginData: any): void {
    window.sessionStorage.removeItem(TOKEN_KEY);
    window.sessionStorage.setItem(TOKEN_KEY, loginData.token);
  }

  public getToken(): any {
    const token = window.sessionStorage.getItem(TOKEN_KEY);
    if (token) {
      // return JSON.parse(token);
      return token;
    }

    return null;
  }

  public getEmail(): any {
    const email = window.sessionStorage.getItem(EMAIL_KEY);
    if (email) {
      // return JSON.parse(email);
      return email;
    }

    return {};
  }

  public isLoggedIn(): boolean {
    const token = window.sessionStorage.getItem(TOKEN_KEY);
    console.log(token);

    if (token) {
      return true;
    }

    return false;
  }
```

Figura 50. Código del servicio de almacenamiento.

Otros servicios implementados

Hasta el momento se han comentado los servicios implementados que quizá sean más particulares de la aplicación y de las decisiones de implementación tomadas, pero además de estos servicios, se tienen implementados otros que permiten el acceso a los *endpoints* relacionados con la gestión de los animales en el servidor o sobre la autenticación en el sistema.

Interceptor

Los interceptores son funciones que, como su propio nombre indican, antes de enviarse a través de HttpRequest, capturan la petición a enviar y permite manipularla. Esta clase ha sido utilizada para añadir el token de sesión en la cabecera correspondiente en cada petición, de esta forma, no se tiene que repetir esta acción en todas y cada una de las peticiones lanzadas desde la aplicación.

```
intercept(req: HttpRequest<any>, next: HttpHandler):
Observable<HttpEvent<any>> {
    const token = this.storageService.getToken();
    req = req.clone({
      // withCredentials: true,
      setHeaders: {
        authorization: `Bearer ${token}`
      }
    });
    return next.handle(req);
  }
```

Figura 51. Código de interceptor.

Guardianes

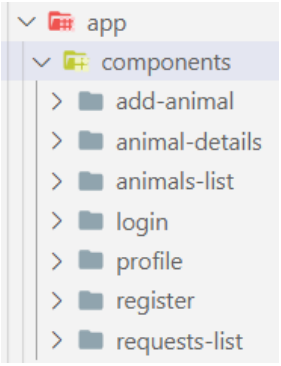
Los *Guards* en AngularJS son una clase especial que comprueba si el componente al que se quiere acceder a través del *routing* del proyecto es accesible dependiendo de las condiciones del usuario. Este tipo de clase ha sido utilizada para permitir acceso a los componentes pertenecientes a la parte privada de la aplicación únicamente si se está logueado.

```
constructor(
    private storage: StorageService,
    private router: Router) { }
  canActivate(
    route: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean | UrlTree> |
Promise<boolean | UrlTree> | boolean | UrlTree {
    const isLoggedIn = this.storage.isLoggedIn();
    if (!isLoggedIn) {
      return this.router.navigate(['/login']).then(() => false);
    }
    return true;
  }
```

Figura 52. Código de guard.

```
{
    path: 'animals',
    canActivate: [IsLoggedInGuard],
    component: AnimalsListComponent
  },
```

Figura 53. Código de uso de guard.



Componentes

A todo lo anterior, cabe añadir la implementación de siete componentes, algunos con comportamiento como páginas completas y otros como porciones de una de las páginas.

5.4.3.2. Aplicación móvil

La implementación en esta parte del proyecto ha continuado por la línea del diseño *responsive*, pero orientado a dispositivos móviles. Por lo tanto, para poder reutilizar algunos comportamientos desarrollados en el portal de gestión y viceversa, se ha utilizado Ionic v6.20.1 junto con AngularJS v14.0.0.

Servicio de eventos

De igual forma que en el proyecto anterior, se ha utilizado un servicio de eventos para la intercomunicación entre algunos componentes o páginas, haciendo que, desde ciertos componentes a través de eventos en estos, se puedan modificar variables de otros componentes.

Servicio de almacenamiento

También, igual que en el servicio anterior, se ha utilizado un servicio de almacenamiento que trabaja con el almacenamiento del dispositivo o navegador. De esta forma se puede trabajar con el token de sesión, como se ha implementado en el proyecto anteriormente comentado.

Otros servicios

Además de los servicios comentados más arriba, que han sido considerados de mayor particularidad y por eso han sido separados de los comentados a continuación, se han implementado servicios para llevar a cabo la autenticación del usuario en el sistema, para obtener información sobre el animal presentado o para enviar al servidor información sobre si se ha mostrado interés en la adopción del animal.

Interceptor

Se ha implementado esta clase para enviar el token de sesión desde cualquier petición que se le envíe al servidor. De esta manera, se ahorra el tener que añadir el token a la cabecera en cada una de las peticiones enviadas al servidor.

Guardián

El uso de *Guards* es debido al control de acceso a las pantallas de la aplicación. Así, el usuario solo puede acceder a la parte privada cuando ha iniciado sesión correctamente.

Navegación

Además de lo anterior, que ha sido implementado de igual forma que en el portal de gestión. La navegación a través de las pantallas de la aplicación ha sido implementada siguiendo el patrón Tabs que ofrece Ionic. De esta manera, se permite una navegación fluida por las pantallas de la aplicación una vez se está autenticado.

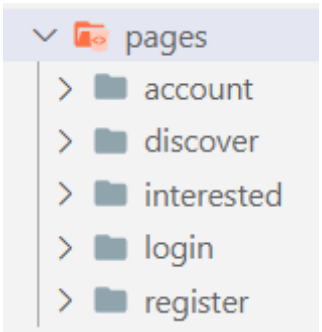
```
<ion-tabs>

  <ion-tab-bar *ngIf="isLoggedIn" slot="bottom">
    <ion-tab-button tab="interested">
      <ion-icon name="heart-circle-outline"></ion-icon>
    </ion-tab-button>

    <ion-tab-button tab="discover">
      <ion-icon name="paw-outline"></ion-icon>
    </ion-tab-button>

    <ion-tab-button tab="account">
      <ion-icon name="person-circle-outline"></ion-icon>
    </ion-tab-button>
  </ion-tab-bar>

</ion-tabs>
```

Figura 54. Uso de pestañas en Ionic.

Páginas

Se han implementado cinco páginas por las que navegar, a diferencia de el proyecto del cliente en AngularJS, estas páginas sí equivalen a pantallas completas de la aplicación, aunque siempre respetando los patrones de diseño *responsive*.

Aquí es donde se aprecia la mayor ventaja de Ionic, la facilidad con la que ofrece aplicaciones multiplataforma, ya que, con una solo implementación de la plantilla de cada pantalla, el framework se encarga de generar las pantallas para iOS e Android. Dejando la posibilidad de añadir CSS propio para describir comportamientos o diseños más específicos.



5.4.4. Resultado

Tras la implementación de estos tres proyectos, se ha conseguido desarrollar un sistema que interactúa entre sí correctamente en un entorno de desarrollo. El sistema resultante está volcado en conseguir cumplir los requisitos funcionales y no funcionales expuestos en puntos anteriores sobre la especificación del proyecto.

Al haberse comprobado el funcionamiento en un entorno de desarrollo, deja algunos puntos de los requisitos no funcionales sin poder comprobar, pero se ha tenido en cuenta en la planificación de futuras tareas. Igualmente, he querido mencionar quizá esta carencia de completitud de algunos requisitos, ya que, al haberse desarrollado diferentes proyectos, ha sido costoso acercarse al paso de la producción.

Aun así, el resultado de la implementación de ambas aplicaciones y el servidor que les proporciona información sobre los recursos ha sido muy satisfactoria, cumpliendo con la gran mayoría de objetivos propuestos en el proyecto y ofreciendo una buena proyección en cuanto a la planificación de futuras tareas.

5.5. Seguridad

Como viene siendo habitual en los últimos años, el desarrollo tecnológico considera el apartado de la seguridad como algo primordial que debe estar revisado y cuidado en todo momento. La importancia del desarrollo seguro del software al realizar aplicaciones, nos permite proteger no solo la información del usuario final, además cuidamos nuestra reputación. Con lo anterior, podemos evitar implicaciones legales, judiciales y los impactos financieros que eso conlleva. (Ospina, s.f.)

Por ello, en el desarrollo de este proyecto se ha intentado seguir pautas de desarrollo seguro para asegurar los primeros pasos para tener una aplicación segura:

- El ingreso de datos por parte de un usuario debe ser validado de manera correcta.
- La autenticación para el acceso a los registros del usuario debe darse siguiendo buenas prácticas de seguridad, por ejemplo, mediante el desarrollo por capas. Asimismo, los mensajes de error que pudieran resultar en este punto deben ser verificados y etiquetados.
- La publicación de servicios debe ser distribuida utilizando métodos de publicación que aseguren más el servicio.
- Uso de frameworks de desarrollo seguro de fuentes reconocidas por la industria, tales como: SSDF de NIST, BSA, OWASP, entre otros.

En adición a esto, se han implementado técnicas más concretas para asegurar la seguridad básica del sistema.

5.5.1. CORS

CORS o Intercambio de Recursos de Origen Cruzado, en castellano, es un mecanismo que utiliza cabeceras HTTP adicionales para permitir que un *user agent* obtenga permiso para acceder a recursos seleccionados desde un servidor, en un origen distinto (dominio) al que pertenece. (MDN web docs, s.f.)

Un ejemplo de solicitud de origen cruzado es que el código JavaScript frontend de una aplicación web utilice XMLHttpRequest o API Fetch para cargar otro recurso de un dominio distinto. Por razones de seguridad, los exploradores restringen las solicitudes HTTP de origen cruzado, siguiendo una política de mismo-origen.

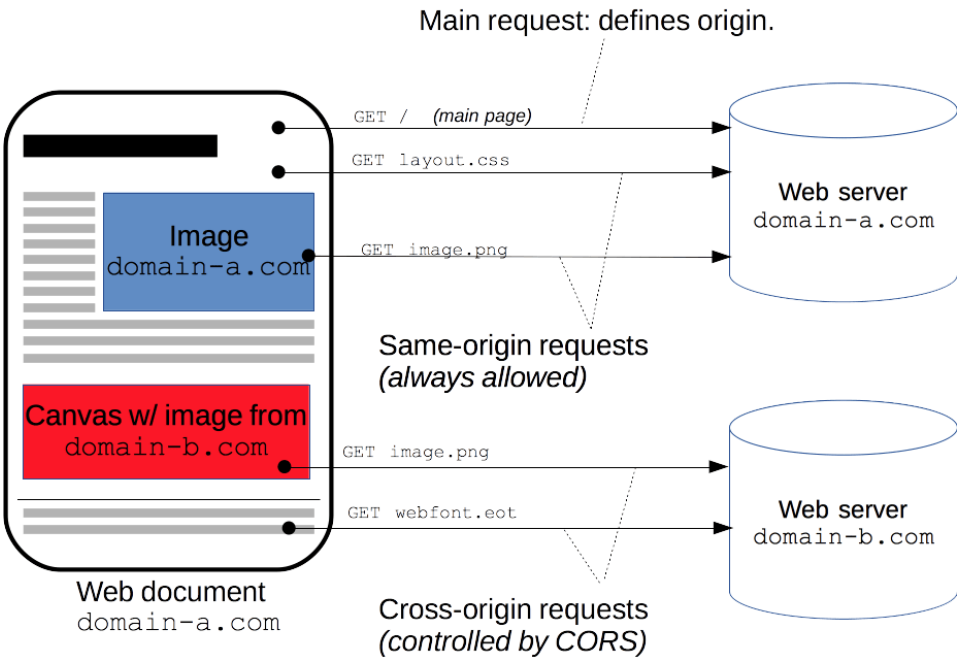


Figura 55. Flujo de CORS.

El servidor del proyecto se ha configurado para que permita las peticiones de origen cruzado de dominios concretos, ya que se sabe en qué dominios residen estas aplicaciones. Fichero server.js:

```
var corsOptions = {
  credentials: true,
  origin: [
    'http://localhost:4200',
    'http://localhost:8100'
  ]
};
```

Figura 56. Configuración CORS en el servidor.

5.5.2. Cifrado de contraseñas

Desde la parte del servidor, se ha utilizado las funciones de cifrado del paquete "bcryptjs". Este paquete es la versión optimizada del paquete "bcrypt" en JavaScript sin dependencias. (npmjs, 2017)

Además de incorporar una sal para protegerse de los ataques de tabla arcoíris, tabla de consulta para obtener claves de texto simple a partir del resultado de una función hash; bcrypt es una función adaptativa que permite aumentar el número de iteraciones para hacerla más lenta, haciéndola más resistente a los ataques de búsqueda por fuerza bruta.

Si bien es cierto que cifrados como SHA son irreversibles, el problema es que el tiempo de cifrado es demasiado corto, lo que significa que aumenta significativamente las posibilidades de que un atacante dé con la contraseña correcta. (Leiva, 2020)

SHA-512 fue diseñado específicamente para ser un algoritmo rápido, lo cual no se aconseja en el cifrado de contraseñas. La solución óptima para este caso es utilizar

algoritmos de cifrado lento, generando un hash en un tiempo lo suficientemente grande para que un ataque de fuerza bruta sea ineficaz y lo suficientemente pequeño para no ralentizar un inicio de sesión. Estos algoritmos de cifrado lento se concibieron con este propósito y son conocidos como PBKDF (Password-Based Key Derivation Function). Este es el tipo de algoritmo que utiliza Bcrypt.

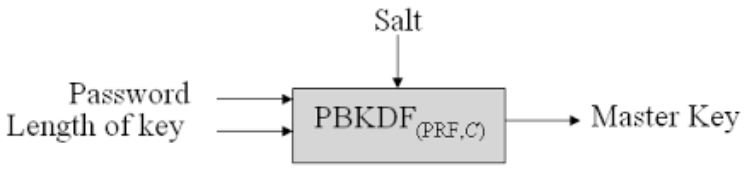


Figura 57. Esquema de PBKDF. (Meltem Sönmez Turan, Elaine Barker, William Burr, and Lily Chen, 2010)

5.5.3. Uso de Helmet en back end

Como se recomienda en el apartado de seguridad de las mejores prácticas de producción, en la documentación de Express, se ha hecho uso de Helmet. Este paquete ayuda a proteger la aplicación de algunas vulnerabilidades web conocidas mediante el establecimiento correcto de cabeceras HTTP. (Express, s.f.)

Helmet es realmente una colección de nueve funciones de middleware más populares que establecen cabeceras HTTP relacionadas con la seguridad:

- csp → establece la cabecera Content-Security-Policy para evitar ataques de scripts entre sitios y otras inyecciones entre sitios.
- hidePoweredBy → elimina la cabecera X-Powered-By.
- hsts → establece la cabecera Strict-Transport-Security que fuerza conexiones seguras (HTTP sobre SSL/TLS) con el servidor.
- ieNoOpen → establece X-Download-Options para IE8+.
- noCache → establece cabeceras Cache-Control y Pragma para inhabilitar el almacenamiento en memoria caché del lado de cliente.
- noSniff → establece X-Content-Type-Options para evitar que los navegadores rastreen mediante MIME una respuesta del tipo de contenido declarado.
- frameguard → establece la cabecera X-Frame-Options para proporcionar protección contra el clickjacking.
- xssFilter → establece X-XSS-Protection para habilitar el filtro de scripts entre sitios (XSS) en los navegadores web más recientes.

5.6. Pruebas

Las pruebas de software son el proceso de evaluación y verificación de un producto o aplicación software para saber que se hace lo que se supone que se debería hacer. En cualquier proyecto software las pruebas son necesarias para la comprobación de que se está cumpliendo con los requisitos, a alto nivel, y que las funcionalidades implementadas tienen un funcionamiento correcto, a más bajo nivel. Estas pruebas son necesarias para detectar errores o defectos en el software y probar si el producto cumple con los requisitos del cliente.

A continuación, se expone el método con el que se han realizado pruebas a lo largo del desarrollo del proyecto, las pruebas ad-hoc. Este término es utilizado para la realización de pruebas de software sin planificación ni documentación, pero puede aplicarse a las aplicaciones de una manera rápida, pero no exhaustiva. Este tipo de pruebas están destinadas a ejecutarse solo una vez, a menos que se produzca un fallo. Aunque este tipo de pruebas pueda vincularse rápidamente con una naturaleza improvisada, algunas de estas pruebas se han llevado a cabo estudiando la funcionalidad y con el objetivo de predecir el error.

Las pruebas que se acaban de exponer han estado orientadas para la parte del servidor, mediante Postman. Esta herramienta ha permitido crear colecciones de peticiones que atacan a los endpoints que se han implementando para tener un comprobante de que tanto las respuestas exitosas como las erróneas son enviadas correctamente.

Además de estas pruebas, para la parte de los clientes, se han realizado pruebas exploratorias. Estas pruebas consisten en realizar flujos de navegación a través de la aplicación, comprobando que se alcanza el objetivo de dicho flujo. Este, de igual forma que las pruebas ad-hoc, han tenido un planteamiento previo sobre lo que se quería encontrar en la aplicación dependiendo del flujo seguido en ella.

De todas formas, se es consciente de la importancia de la implementación y uso de un entorno de CI/CD (*continuous integration/continuous delivery*), el cual será propuesto en el siguiente apartado como mejora.

Por último, cabe destacar la facilidad con la que AngularJS ofrece la posibilidad de testear cualquier elemento creado desde su CLI. Esto es otra ventaja del framework propuesta como mejora para sacarle el mayor partido a esta tecnología y asegurar la máxima calidad posible.

5.7. Trabajos futuros y posibles mejoras

Una vez se tiene constancia del proyecto llevado a cabo en este tiempo, a la vista está de que todavía puede tomar diferentes direcciones y desarrollarse más en un aspecto o en otro, dependiendo de la respuesta del público objetivo y de los requisitos de los clientes o futuros inversores si los hubiese.

Volviendo a mencionar algo antes comentado, la metodología de trabajo para el desarrollo ha sido ágil y, en esta metodología, una de las características principales y factor que conforma su esencia es la mejora continua. Por lo que en ningún momento se ha perdido el foco a futuras ampliaciones y mejoras de varias de las funcionalidades implementadas.

En primer lugar, como trabajos futuros a corto plazo se plantea la revisión exhaustiva de las funcionalidades implementadas. Se plantea como un momento clave para la implementación e integración de pruebas en el proyecto. De esta manera, se conseguiría tener la certeza de que existe un producto mínimo viable de calidad y se asentarían las bases para un mejor desarrollo. Este punto, aunque propuesto a corto plazo, se

contempla con una duración de al menos la mitad de tiempo del que se ha dedicado al proyecto hasta ahora, alrededor de mes y medio. Todos los recursos se emplearían en la revisión y consulta de las aplicaciones para asegurar una base robusta y fiable.

Como trabajos futuros más a largo plazo, teniendo en cuenta que hasta ahora el desarrollo de todo lo implementado hasta el momento se ha llevado a cabo por una única persona, se propone la formación de un equipo de al menos tres personas comprometidas con el proyecto. De esta manera, el desarrollo del proyecto no se vería tan afectado por lo momentos de aprendizaje sobre las buenas prácticas e implementación de nuevas funcionalidades. De nuevo, en la esencia de las metodologías ágiles está el trabajo en equipo, permitiendo alcanzar un mayor grado de organización y calidad en el proyecto.

Como última tarea general que se plantea está la de asentar un objetivo y unas estrategias para alcanzarlo tras una revisión del plan de negocio con el nuevo equipo, si lo hubiera, o con un equipo de asesores profesionales. Este punto es clave para que el proyecto siga la dirección correcta y pueda tener una mayor probabilidad de éxito en su supuesta salida al mercado.

En lo referente a este tema, la salida al mercado o puesta en producción de la aplicación, es una tarea muy importante también a conseguir en un plazo de tiempo no demasiado amplio, pero priorizando ante todo la tarea de crear un producto mínimo de calidad, habiéndose probado y auditado.

Finalmente, como posibles mejoras se plantean muchas, pues siempre se puede hacer mejor. Las principales mejoras que podrían llevarse a cabo en un periodo de medio año aproximadamente es la de una definición de procesos y marco de trabajo dentro del proyecto. Esta estandarización aceleraría la evolución del proyecto considerablemente. Las tareas anteriormente comentadas son consideradas propuestas de mejora, pues aportan valor y, sobre todo, calidad al proyecto.

6. CONCLUSIONES

La primera perspectiva desde la que me gustaría concluir este TFG es la de proyecto. Hasta el momento, no he conseguido estar seguro de si los objetivos y el planteamiento del proyecto ha sido suficiente o excesivo para un trabajo así. Habiendo tenido que compaginarlo con clases de la universidad y prácticas de empresa, el resultado de este proyecto creo que es satisfactorio. Luchando contra la tendencia a la búsqueda de la perfección, me atrevería a decir que el resultado de este proyecto tiene una dirección y plantea unas posibilidades que al principio no era capaz de imaginar.

Soy consciente de lo mucho que queda por hacer para lograr hacer realidad que la aplicación planteada en este trabajo llegase a ser un producto real, pero de todas formas ha hecho que me vea capaz de afrontar esa parálisis al verse uno mismo frente al abismo de comenzar algo y tomar tantas decisiones que serán decisivas en el resultado de lo que se esté queriendo crear.

Otra perspectiva desde la que quería plantear la conclusión es la de estudiante. Tener la oportunidad de luchar e invertir tiempo en la realización de un proyecto así desde cero y verlo pasar por todas sus fases iniciales, necesitando buscar la respuesta a preguntas nunca antes planteadas, ha sido una gran experiencia. Durante el desarrollo de este proyecto me he acordado de muchas y muchos de mis profesoras y profesores que he tenido a lo largo del grado (para bien). He tenido en consideración todos los consejos que recuerdo que me han dado a la hora de tomar decisiones sobre las tecnologías de desarrollo utilizadas, la manera de trabajar con repositorios, el cómo plantear una arquitectura idónea, e, incluso, la importancia de la calidad en todo lo que hacemos.

Sea como fuere, siento que este periodo de estudiante llega a su fin, por el momento, y siento una sensación de alegría y tranquilidad por el considerable crecimiento personal y profesional que he experimentado, además de todo el conocimiento, y las nuevas formas de seguir ampliando conocimiento, que he adquirido.

Por último, desde mi perspectiva como persona, siento que he podido desarrollarme en un campo de la ciencia que tiene una notoria presencia en el mundo actual y que continúa abriendo nuevos frentes. Estoy satisfecho de haber podido utilizar el conocimiento sobre estas herramientas modernas para algo que, a mi parecer, puede llegar a hacer que la adopción de animales sea más justa. El haberme podido aproximado a la ética en la informática con un proyecto así hace que me sienta orgulloso de haber decidido cursar un grado como este.

Este proyecto es solo un pequeño paso hacia el planteamiento de buscar nuevas formas de tratar a los seres vivos, creo que es un gran momento para hacerlo y espero que haya servido para motivar, al menos a mi yo del futuro, a desarrollarse en el altruismo y a reflexionar sobre el valor que le damos a la vida, más allá de las nuestras.

7. REFERENCIAS

¡Enlázanos! (2022). Obtenido de Bambú Difunde: https://bambu-difunde.net/enlazanos.php

2Spacios. (2015). Obtenido de La importancia del plan de marketing: https://www.2spacios.com/noticias/la-importancia-del-plan-de-marketing

Affinity, F. (2021). *Affinity*. Obtenido de Estudio Él nunca lo haría: https://static.fundacion-affinity.org/cdn/farfuture/ZxB9BUhpCCY_I2wJmqJSY-lXO75e5joPoDKpRxNuMso/mtime:1655455698/sites/default/files/white-paper-abandono-2022.pdf

Anerpa. (2022). Obtenido de Programa Adopta 2022: https://www.anerpa.org/programa-adopta/

Atmitim, J. M. (2021). *Profile*. Obtenido de Qué es Ionic: ventajas y desventajas de usarlo para desarrollar apps móviles híbridas: https://profile.es/blog/que-es-ionic/

Barón, M. (s.f.). *Behance*. Obtenido de UI/UX Tinder: https://www.behance.net/gallery/98878253/UIUX-Tinder?locale=es_ES

BuscaFuska. (2022). Obtenido de BuscaFuska: https://buscafuska.com/

Bustamante, B. A. (2017). *LinkedIn*. Obtenido de Descubre los secretos UX destrás de Tinder : https://es.linkedin.com/pulse/descubre-los-secretos-ux-detr%C3%A1s-de-tinder-rodriguez-bustamante

Chuby - Adopta un perro, gato. (2022). Obtenido de Google Play Store: https://play.google.com/store/apps/details?id=rocks.chuby&hl=es_419&gl=US&pli=1

Concepto. (s.f.). Obtenido de https://concepto.de/viabilidad/

Donetonic. (s.f.). Obtenido de Qué son los Eventos en Scrum : https://donetonic.com/es/que-son-los-sprints-en-scrum/

Express. (s.f.). Obtenido de Mejores prácticas de producción: seguridad: https://expressjs.com/es/advanced/best-practice-security.html

Gonçalves, M. J. (2021). *Hiberus blog*. Obtenido de ¿Qué es Angular y para qué sirve?: https://www.hiberus.com/crecemos-contigo/que-es-angular-y-para-que-sirve/

Handa, U. (2021). *Cynoteck*. Obtenido de 7 razones para usar Angular para sus aplicaciones web en 2022: https://cynoteck.com/es/blog-post/reasons-to-use-angular-for-your-web-app/#:~:text=de%20las%20personas.-,El%20principal%20beneficio%20de%20AngularJS%20es%20que%20permite%20a%20los,aplicaciones%20de%20una%20sola%20página

Hernández, U. (2018). *Código Facilito* . Obtenido de Qué es TypeScript : https://codigofacilito.com/articulos/typescript

Kinsta. (2022). Obtenido de ¿Qué es Express.js? Todo lo que Debes Saber : https://kinsta.com/es/base-de-conocimiento/que-es-express/

Kiwoko. (2022). Obtenido de Kiwoko: https://www.kiwokoadopta.org/

Laoyan, S. (2022). *Asana*. Obtenido de Scrumban: lo mejor de dos metodologías ágiles: https://asana.com/es/resources/scrumban

Leiva, F. (2020). *Leiva*. Obtenido de Cómo cifrar contraseñas de forma segura: https://leiva.io/2020/04/12/como-cifrar-contrasenas-de-forma-segura/

Lucas, J. (2019). *Open Webinars*. Obtenido de Qué es NodeJS y para qué sirve: https://openwebinars.net/blog/que-es-nodejs/

Magaña, L. M. (2020). *Openwebinars*. Obtenido de Qué es Json Web Token y cómo funciona: https://openwebinars.net/blog/que-es-json-web-token-y-como-funciona/

MDN web docs. (s.f.). Obtenido de ¿Qué es JavaScript?: https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/What_is_JavaScript

MDN web docs. (s.f.). Obtenido de Control de acceso HTTP (CORS): https://developer.mozilla.org/es/docs/Web/HTTP/CORS

Meltem Sönmez Turan, Elaine Barker, William Burr, and Lily Chen. (2010). *NIST Special Publication*. Obtenido de Recommendation for Password-Based Key Derivation: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-132.pdf

Northware. (2022). Obtenido de Requerimientos en el desarrollo de software y aplicaciones: https://www.northware.mx/blog/requerimientos-en-el-desarrollo-de-software-y-aplicaciones/

npmjs. (2017). Obtenido de bcrypt.js: https://www.npmjs.com/package/bcryptjs

Open Webinars. (s.f.). Obtenido de https://dc722jrlp2zu8.cloudfront.net/media/uploads/2019/12/04/cap3-seguridad2.png

Ospina, A. M. (s.f.). *Intergrupo*. Obtenido de ¿Por qué es importante el desarrollo de software seguro?: https://intergrupo.com/por-que-es-importante-el-desarrollo-de-software-seguro/

Pandora FMS. (2022). Obtenido de NoSQL vs SQL: principales diferencias y cuándo elegir cada una de ellas: https://pandorafms.com/blog/es/nosql-vs-sql-diferencias-y-cuando-elegir-cada-una/

Perros en adopción en Madrid. (2022). Obtenido de Miwuki Pet Shelter: https://petshelter.miwuki.com/perros-en-adopcion-en-madrid?_gl=1*ow836v*_up*MQ..*_ga*MTkwOTE0OTU4NS4xNjczMjgzNzQ1*_ga_5PXGP3C97S*MTY3MzI4Mzc0NC4xLjEuMTY3MzI4Mzc0NC4wLjAuMA..&_ga=2.238001288.921295946.1673283745-1909149585.1673283745

PMOinformatica. (2017). Obtenido de Requerimientos funcionales: Ejemplos: http://www.pmoinformatica.com/2017/02/requerimientos-funcionales-ejemplos.html

Sentrio. (2021). Obtenido de Metodologías Agile: los 4 valores y 12 principios del 'Manifiesto Ágil': https://sentrio.io/blog/valores-principios-agile-manifiesto-agil/

Universitat Oberta de Catalunya. (2018). Obtenido de Por qué elegir el gestor de base de datos MySQL: https://fp.uoc.fje.edu/blog/por-que-elegir-el-gestor-de-base-de-datos-mysql/

Universitat Oberta de Catalunya. (2022). Obtenido de Qué es Node.js y cuáles son las ventajas de usar esta tecnología: https://fp.uoc.fje.edu/blog/que-es-node-js-y-cuales-son-las-ventajas-de-usar-esta-tecnologia/#:~:text=utilizar%20esta%20plataforma.-,Node.,evitando%20el%20bloqueo%20de%20procesos

Value Keep. (s.f.). Obtenido de ¿Qué es el UX y cuál es su importancia?: https://valuekeep.com/es/recursos/que-es-el-ux/#:~:text=La%20incorporaci%C3%B3n%20del%20dise%C3%B1o%20UX,la%20fidelidad%20de%20los%20clientes.

Visure Solutions. (s.f.). Obtenido de Qué son los requisitos funcionales: ejemplos, definición, guía completa: https://visuresolutions.com/es/blog/functional-requirements/