



HAPPYPAWS

Alumnos

David Menéndez Nuñez

Irene Pérez Cano

Raúl Moreno Rodrigo

Tutor

Félix De Pablo

Ciclo formativo

DAW

Instituto

UNIR FP (EDIX)



1. Memoria del Proyecto.....	5
2. Objetivo del proyecto.....	6
3. Módulos formativos aplicados en el TFG.....	7
3.1. Detalle de los módulos.....	7
4. Herramientas y lenguajes utilizados.....	9
4.1. Visual Studio Code.....	9
4.2. MySQL WorkBench.....	9
4.3. Figma.....	9
4.4. Spring boot.....	10
4.5. Angular.....	10
4.6. Jira.....	10
4.7. Git.....	11
4.8. GitHub.....	11
4.9. Java.....	11
4.10. TypeScript.....	12
4.11. Html.....	12
4.12. Css.....	12
4.13. Bootstrap.....	13
4.14. Bing Image Creator.....	13
5. Componentes del Equipo y aportación.....	14
5.1. Integrantes.....	14
5.2. Aportación.....	14
6. Fases del proyecto.....	16
6.1. Modelo de Datos.....	16
6.2. Casos de Uso.....	17
6.3. Design Thinking.....	18
6.3.1. Empatizar.....	19
6.3.1.1. Investigación documental.....	20
6.3.1.2. Investigación de la competencia.....	20
6.3.1.3. Investigación del usuario.....	21
6.3.2. Definir.....	21
6.3.3. Idear.....	21
6.3.4. Prototipar.....	22
6.3.5. Testear.....	22
6.4. Design System.....	23
6.4.1. Guía de estilos.....	23
6.4.2. Mockups.....	26
6.5. Visual Studio Code. Configuración del proyecto.....	28
6.6. Angular FrontEnd.....	28
7. Funcionalidades e Implementación.....	30
7.1. Funcionalidad Usuarios.....	30

1. Tipos de Usuarios:.....	30
a. Administrador (Admin):.....	30
b. Adoptante:.....	30
c. Protectora:.....	31
2. Autenticación y autorización.....	31
a. JWT (JSON Web Token):.....	31
b. Proceso de Registro:.....	32
7.2. Funcionalidad Protectoras.....	35
7.3. Funcionalidad Animales.....	38
7.4. Funcionalidad Admin.....	48
7.5. Funcionalidad Adopciones.....	50
7.6. Funcionalidad Formulario.....	52
7.7. Funcionalidad Subir Fotos.....	53
7.8. Funcionalidad Envío de Email.....	55
8. Conclusiones y mejoras del proyecto.....	60
8.1. Mejoras.....	60
8.2. Conclusiones.....	60
9. Agradecimientos.....	62
10. Bibliografía.....	63
11. Anexos.....	64
11.1. Scripts BBDD.....	64
11.2. Imágenes.....	64
11.3. Código fuente aplicación GitHub.....	64
11.4. Diseño de la web - Figma.....	64
11.5. Manuales de uso.....	64

1. Memoria del Proyecto

Antes de iniciar este proyecto, cada integrante del equipo presenta una idea de proyecto. Aquí se detallan las propuestas:

- Aplicación para venta de suplementación deportiva
- Aplicación de recetas
- Aplicación de gestión de adopciones

Tras debatirlo, se optó por una aplicación web para la adopción de animales y la gestión de las adopciones por las protectoras.

Para este trabajo del Ciclo Formativo de Grado Superior de Aplicaciones Web, realizamos el análisis del procedimiento que siguen actualmente las protectoras para las adopciones y se comprueba que cada protectora de cada municipio y provincia de España actúa de distintas formas. Muchas de ellas coinciden en que::

- Muchas no disponen de página web ni tienen opción de publicar los animales que tienen en adopción. Usan las redes sociales y cuesta contactar con ellas o ver qué animales tienen en adopción.
- El control de los cuestionarios de adopción y las adopciones realizadas las llevan en papel, cuadernos o como mucho por email.

Por otro lado, desde la parte del adoptante, contemplamos que cuando alguien está interesado en adoptar tiene que buscar en internet protectoras y contactar con cada una de ellas para tener más información.

2. Objetivo del proyecto

El objetivo de nuestro proyecto es tener una aplicación que facilite a las personas interesadas en adoptar un animal, acceder a una base de datos centralizada de protectoras y animales en adopción.

La aplicación debe permitir a las protectoras de cualquier parte de España, darse de alta en la plataforma, y tener un portal de visualización con un listado de animales en adopción, así como poder gestionar las peticiones de adopción, los cuestionarios de adopción y llevar la gestión de las solicitudes de adopción.

Por otro lado debe permitir a una persona interesada en adoptar, poder ver un listado de animales que pueda ser filtrado por Especie, Raza, Sexo, Tamaño, Provincia y si se Envía. También debe poder ponerse en contacto con una protectora, llenar el cuestionario de adopción y solicitar la adopción de un animal.

3. Módulos formativos aplicados en el TFG

3.1. Detalle de los módulos

- Programación (Java)

Para esta asignatura nos hemos apoyado en los Algoritmos y estructuras de datos, con lenguajes de Programación Java y JavaScript. Hemos usado los conceptos básicos de lenguaje, sintaxis y semántica. Hemos usado los paradigmas de la programación orientada a objetos, aplicando los principales principios como la encapsulación, herencia y polimorfismo. Así mismo hemos manejado las excepciones para controlar los errores de la aplicación.

- Bases de Datos (MySQL)

Hemos utilizado el sistema de gestión de bases de datos MySql Workbench y se hemos creado el diagrama de datos relacionados utilizando la herramienta LucidChart para hacer el gráfico entidad-relación (ER)

- Entornos de Desarrollo:

La aplicación se ha creado en Entorno Maven. Para el seguimiento de versiones se ha utilizado Git y GitHub para el repositorio. Se han usado ramas y pull request para el seguimiento de la implementación de las funcionalidades. El seguimiento del desarrollo se ha realizado con Jira.

- Lenguajes de marcas:

Se ha usado HTML 5, CSS, manipulación del DOM, transmisión de información por Json

- Sistemas informáticos

Se han realizado pruebas en máquinas con sistemas operativos Ubuntu y Windows para comprobar el funcionamiento de la aplicación.



- DWE Cliente

Para la parte cliente se ha utilizado Angular, Typescript y Bootstrap.

- DWE Servidor

Creación de la aplicación en Java con Spring Boot con conexión a base de datos con implementación de seguridad JWT

- Diseño de Interfaces Web

Se han diseñado los Mockups y los casos de uso con Figma.

4. Herramientas y lenguajes utilizados

4.1. Visual Studio Code

Desarrollado por Microsoft, Visual Studio Code es un editor de código fuente versátil y potente que se ejecuta en el escritorio y está disponible para Windows, macOS y Linux. Es gratuito y de código abierto, y está diseñado para ser ligero, rápido y extensible. Viene con soporte integrado para JavaScript, TypeScript y Node.js y tiene un rico ecosistema de extensiones para otros lenguajes y tiempos de ejecución como C++, C#, Java, Python, Go, .Net. Además, cuenta con depuración integrada, control de versiones y una terminal incorporada.

Ha sido nuestra herramienta principal para la generación del código fuente de nuestra aplicación.

4.2. MySQL WorkBench

Desarrollada por Oracle, MySQL Workbench es una herramienta visual que combina características de diseño, modelaje, desarrollo y administración de bases de datos que se utiliza específicamente con el sistema de gestión (DBMS) MySQL. Es una herramienta gratuita que además del diseño y la administración, también nos permite hacer consultas SQL, depurar procedimientos almacenados, administrar usuarios y privilegios, importar y exportar datos y monitoreo y optimización del rendimiento.

Se ha utilizado para el desarrollo de nuestra Base de Datos y consultas SQL, además de servirnos como muestra de la funcionalidad o no de nuestros métodos y funciones desarrolladas en la app cuando aún no teníamos generado el frontend, proporcionandonos una vista actualizada de nuestra base de datos según íbamos trabajando sobre ella.

4.3. Figma

Fundada por Dylan Field y Evan Wallace, Figma es una herramienta de diseño colaborativa en tiempo real de interfaces de usuario (UI) y experiencia de usuario (UX). Además permite la creación y reutilización de componentes,

revisión y comentarios sobre el diseño, versionado, compatibilidad multiplataforma y la exportación de recursos.

Nos ha permitido colaborar en tiempo real en la creación y prototipado del diseño de nuestro proyecto.

4.4. Spring boot

Desarrollado por Pivotal Software, que ahora forma parte de VMware, Spring Boot es un proyecto de código abierto que forma parte del ecosistema Spring, un marco de desarrollo para la construcción de aplicaciones en el lenguaje de programación java. Se utiliza para simplificar el proceso de desarrollo de aplicaciones basadas en el framework Spring al proporcionar una configuración predeterminada y simplificar las tareas comunes.

En nuestro proyecto ha sido una parte muy importante, ya que lo hemos utilizado para implementar integralmente la lógica del lado del servidor.

4.5. Angular

Desarrollado por Google, Angular es un framework de código abierto para el desarrollo de aplicaciones web modernas y escalables. Está escrito en TypeScript y proporciona una estructura robusta utilizando tecnologías como TypeScript, HTML y CSS para construir aplicaciones de una sola página (SPA) sin necesidad de recargar una y otra vez. Algunas de sus características más importantes son el enlace de datos bidireccional, inyección de dependencias, arquitectura basada en componentes, directivas, enrutamientos, manejo de formularios, etc.

En nuestro proyecto ha sido otra de las partes más importantes, ya que la parte del lado del cliente ha sido implementada con esta tecnología.

4.6. Jira

Desarrollada por Atlassian, Jira es una herramienta de gestión de proyectos. Se utiliza en entornos empresariales y de desarrollo de software para gestionar proyectos, realizar seguimiento de tareas y facilitar la colaboración entre

equipos. Así mismo, nos ha servido para marcarnos unos objetivos de desarrollo, ir siguiendo su evolución y completar cada uno de ellos.

4.7. Git

Desarrollado por Linus Torvalds, Git es un sistema de control de versiones distribuido, gratuito y de código abierto diseñado para manejar todo, desde proyectos pequeños hasta proyectos muy grandes con velocidad y eficiencia. Supera a herramientas como SCM, CVS, Perfore y ClearCase con características como ramificaciones locales, áreas de preparación convenientes y múltiples flujos de trabajo.

Su propósito principal es de gestionar el control de versiones del código fuente de un proyecto, lo cual nos ha sido muy útil a la hora de trabajar en nuestro proyecto y detectar errores de código daban como resultado el fallo de algunos componentes que funcionaban con anterioridad.

4.8. GitHub

Desarrollado por Chris Wanstrath, PJ Hyett y Tom Preston-Werner, GitHub es una plataforma de desarrollo colaborativo basada en la web que utiliza el sistema de control de versiones Git anteriormente descrita. En los últimos años se ha convertido en una de las plataformas más populares para alojar y colaborar en proyectos de software, ya que proporciona herramientas y funcionalidades adicionales más allá del control de versiones.

Para nuestro proyecto también ha sido una parte especialmente importante ya que en todo momento ha sido donde hemos tenido alojado nuestro repositorio con todo nuestro código fuente.

4.9. Java

Desarrollado por James Gosling y creado por Sun Microsystems, Java es un lenguaje de programación y una plataforma informática. Es utilizado para una variedad de propósitos y en diversos contextos, gracias a su portabilidad, versatilidad y su plataforma que permite ejecutar código en múltiples entornos.

Para este proyecto ha sido el lenguaje utilizado para la parte del servidor en Spring Boot.

4.10. TypeScript

Desarrollado por Microsoft, TypeScript es un lenguaje de programación de código abierto, tipado estático, compatible con JavaScript, orientado a objetos, muy útil para grandes proyectos gracias a su desarrollo escalable por su fuerte tipado que proporciona claridad y estructura al código. Además se integra muy bien con diversas herramientas y entornos de desarrollo, como en nuestro caso, Visual Studio Code.

Ha sido otro de los lenguajes más importantes de nuestro proyecto, ya que se ha desarrollado con este gran parte de la implementación del cliente mediante Angular.

4.11. Html

No tiene un único desarrollador pero fue propuesto por Tim Berners-Lee, Html es el lenguaje fundamental utilizado para estructurar y organizar el contenido de las páginas web. Algunas de las características más importantes que nos aporta es la estructura de documentos, permitiéndonos definir títulos, párrafos, listas, etc., creación de hipertexto e hipervínculos, incorporación de imágenes, videos, audios, formularios interactivos, etc. Además es compatible con la mayoría de navegadores. En nuestro proyecto nos ha sido muy útil para la formación de las páginas web de nuestra aplicación.

4.12. Css

Desarrollado por el World Wide Web Consortium (W3C), Css o Cascading Style Sheets, es un lenguaje de diseño propuesto como solución para separar la estructura del contenido de un documento Html de su presentación visual. Permite aplicar estilos y diseños a páginas Html de manera consistente y eficiente. En este proyecto hemos utilizado Bootstrap como para el estilo, pero este lenguaje de diseño nos ayudó a retocar ciertos detalles que bootstrap no nos permitía.

4.13. Bootstrap

Desarrollado por Mark Otto y Jacob Thornton de Twitter, Bootstrap es un framework de desarrollo frontend de código abierto. Se utiliza para facilitar el diseño y desarrollo de sitios web y aplicaciones web responsivas y estéticamente atractivas. Además, cuenta con una gran galería de componentes ya creados y listos para usarse, jQuerys y plugins personalizados que mejoran la interactividad y funcionalidad de las páginas web y además permite su personalización. Básicamente ha sido para eso para lo que lo hemos utilizado en nuestro proyecto.

4.14. Bing Image Creator

Creada por Microsoft Designer, Bing Image Creator es básicamente una herramienta que te permite crear imágenes a partir de palabras con la ayuda de inteligencia artificial. En nuestro proyecto la hemos utilizado para generar imágenes originales y no reales para nuestra base de datos.

5. Componentes del Equipo y aportación.

5.1. Integrantes

Los integrantes del equipo somos:

- David Menéndez Nuñez
- Irene Pérez Cano
- Raúl Moreno Rodrigo

5.2. Aportación

En esta tabla se detallan las acciones principales, aunque muchas de ellas fueron creadas por un componente en algunos casos, el resto ha realizado modificaciones sobre el mismo para ir adaptándolo a las necesidades del proyecto:

Acción	Componente/s
BBDD. Script creación.	David y Raúl
BBDD Script inserts	Raúl y David
Creación proyecto base, entidades, etc, en Spring Boot con conexión a bbdd en VS Code.	David
Creación del mapa ER en lucidchart	David
Creación repositorio GitHub	Irene
Creación del Design Thinking	Raúl
Creación de los Mockups y estilos	Irene
Securización JWT	Irene
Protectoras, listado, modificación, formulario contacto, estados.	David
Usuarios/Protectoras/Admin registro y login.	Irene
Formulario cuestionario de Adopción	David e Irene
Servicio de envío de email	David
Formulario de contacto web.	David

Acción	Componente/s
Panel Admin General y secciones Listado protectoras y animales, con botones modificar y formulario para modificar.	David
Perfil Usuario	Irene
Maquetación : Home, Acerca, footer, menú y FAQ	Irene
Panel Admin : Usuarios > Editar y Adopciones > ver	Irene
Alta adopción - Aprobar - Rechazar - Ver	Irene
Creación, modificación y eliminación de un Animal	Raúl
Listado de animales tanto para usuarios como para protectoras	Raúl
Filtro múltiple y sencillo para los diferentes listados	Raúl
Cambio automático de atributo “enabled” de animal a la vez que es aceptada la adopción	Raúl
Manuales de uso: Como crear protectora, panel admin protectora y panel admin administrador.	David
Manual de uso: Como crear un animal, modificar y borrar	Raúl
Manual de uso: Cómo solicitar una adopción	Irene

6. Fases del proyecto

6.1. Modelo de Datos

Se realiza el diseño de la base de datos con LucidChart. En el diseño se detallan las entidades, atributos y relaciones. Este es el esquema inicial de la base de datos:

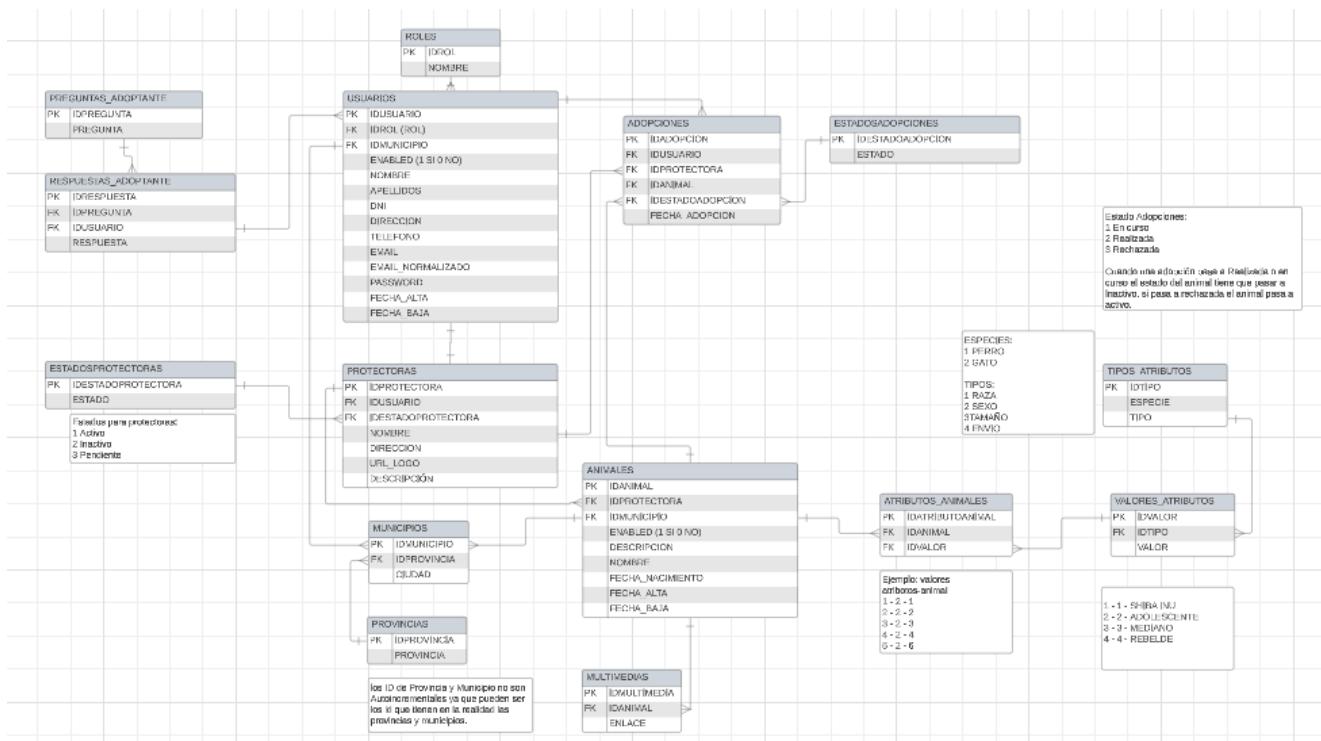


Imagen-01: Prototipo diseño ER

Después de varios análisis se realizan algunos cambios en los atributos de animales y añaden varios atributos a la clase protectora para mostrar la foto, email, etc.

Aquí se encuentra el diseño final de la base de datos y sus relaciones:

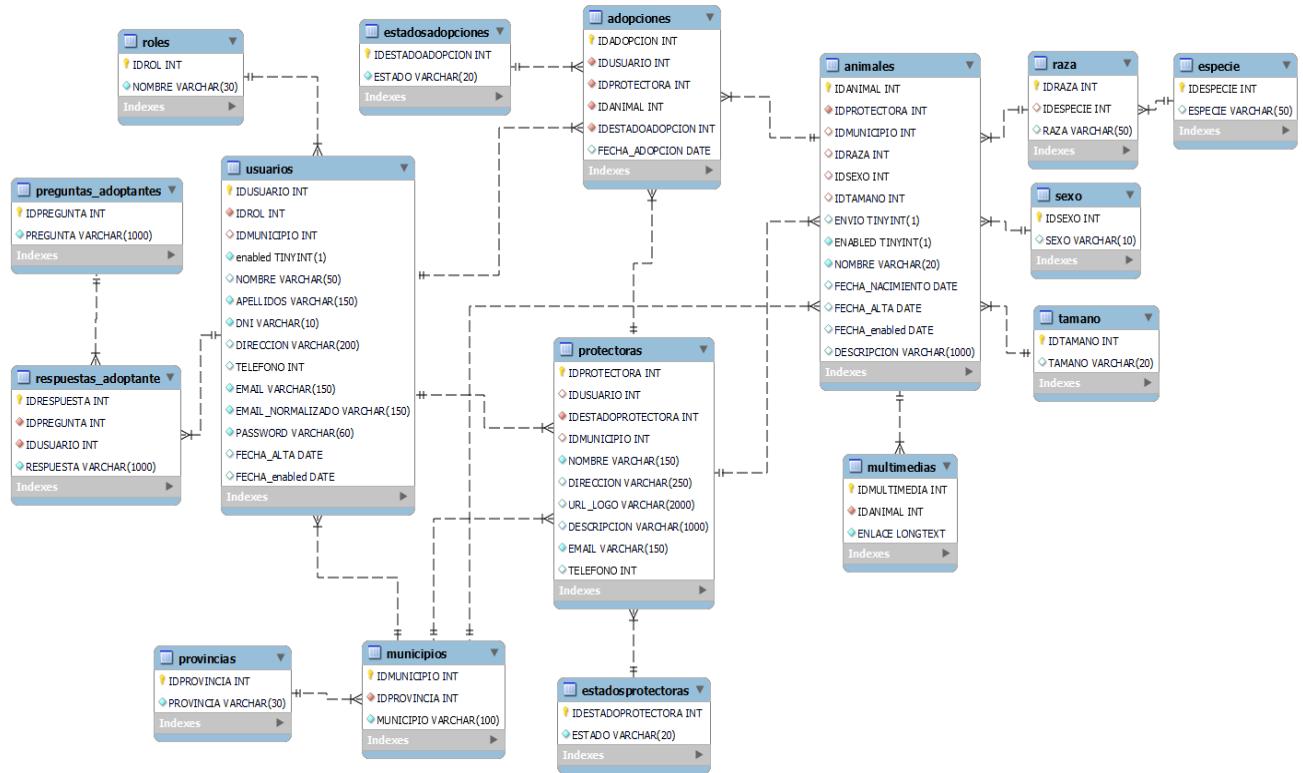


Imagen-02: [Diseño final ER](#)

6.2. Casos de Uso

A continuación se realiza el diagrama de clases de uso que muestra lo que cada actor puede realizar y aquellas tareas dependientes u opcionales. Para nuestra aplicación tenemos 4 posibles actores:

- Usuario anónimo

- Usuario adoptante
- Usuario Protectora
- Usuario Administrador

Cada usuario tiene accesos y funcionalidades definidas. Las funcionalidades e implementación de se encuentran detalladas en el punto 7.

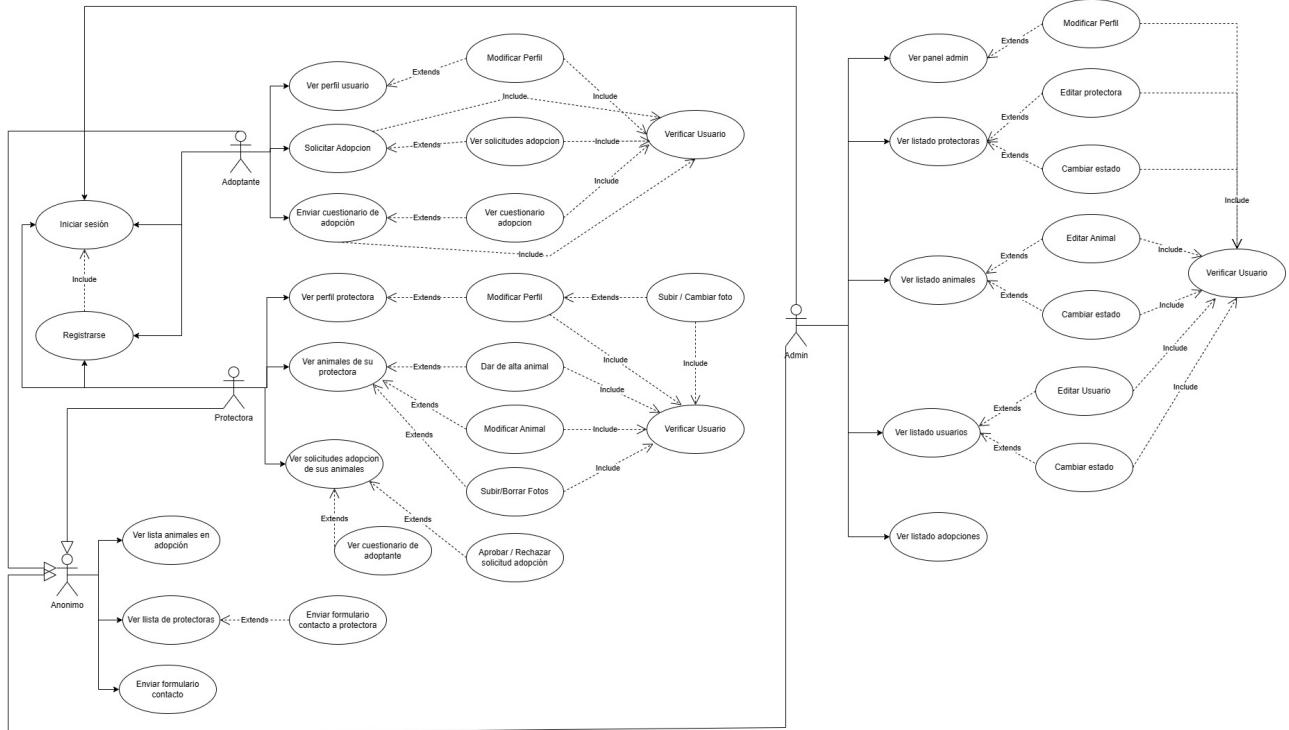


Imagen-03: Casos de uso

6.3. Design Thinking

El Design Thinking o “Pensamiento de Diseño”, es el método más difundido para conseguir un buen diseño UX. Es una metodología orientada a la generación de soluciones a partir de un reto, un proceso ágil e iterativo que se centra en el usuario, sus necesidades, emociones, expectativas, motivaciones y capacidades. Cuanto más se conozca al usuario final, más probable será que nuestro producto le proporcione una experiencia positiva.

El objetivo es generar soluciones de acuerdo a problemas detectados de un determinado marco de trabajo. Se trata de pensar “de otra forma” y construir prototipos sin temor a equivocarse, probar y fallar para de los fallos, aprender y tener una experimentación continua.

El auge y la popularidad en continuo crecimiento de esta metodología viene dada por su capacidad de generar en muy poco tiempo soluciones innovadoras, creando una cultura creativa e innovadora allá donde se utiliza.

Este proceso se divide en distintas etapas según diferentes variaciones y escuelas de pensamiento. En España se parte de la división clásica de 5, que son las siguientes;

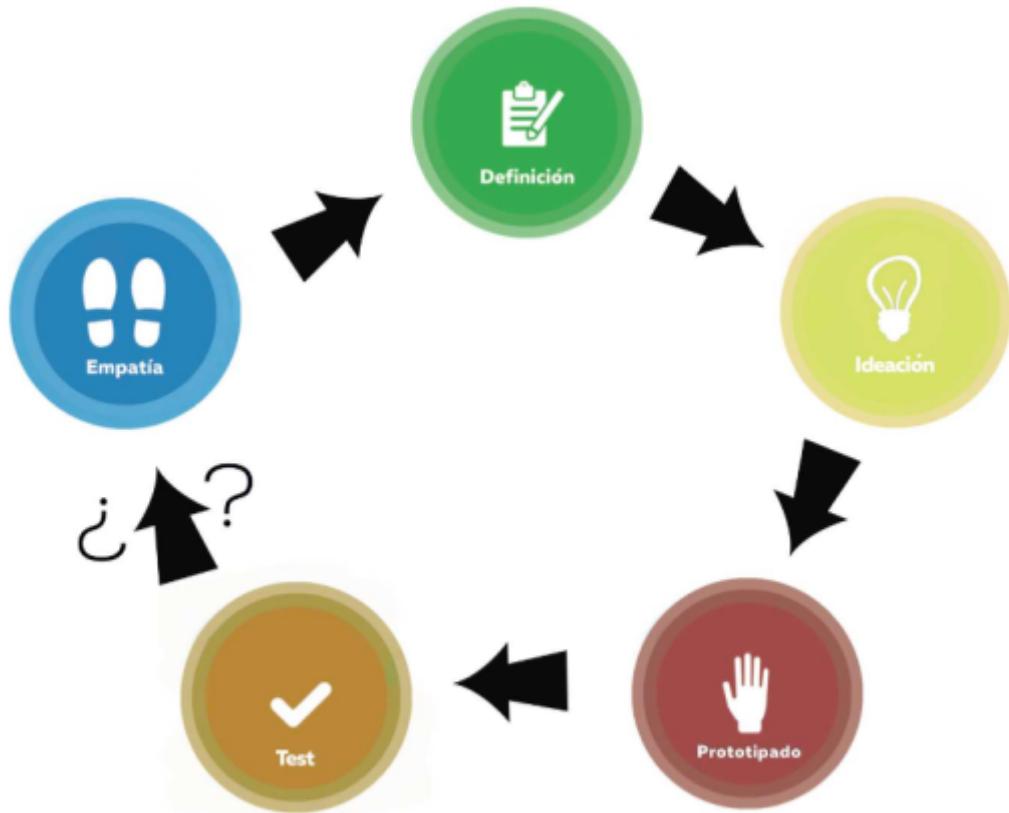


Imagen: proceso Design Thinking

Se puede encontrar el **proyecto** con estos pasos en el siguiente enlace;

<https://www.figma.com/file/3OWbJBBSObNQpdrpnUbVTn/HappyPaws?type=whiteboard&node-id=0%3A1&t=kI4PXsB6c8mrHhBu-1>

6.3.1. Empatizar

Es la primera etapa del proceso y la base del mismo. En esta fase se define el arquetipo de usuario al que vamos a dirigirnos, establecemos nuestros objetivos de investigación y a partir de ellos, decidimos qué técnicas de recopilación de información utilizaremos.

El objetivo de esta fase es comprender las necesidades de los usuarios, ponernos en su lugar teniendo en cuenta sus deseos e investigar todo lo

posible sobre su situación y sus problemas actuales para establecer una base teórica sólida. Esta fase se divide en 3 secciones más, que son las siguientes;

6.3.1.1. Investigación documental.

Esta sección se centra en ampliar nuestro conocimientos con la recopilación y revisión de las noticias sobre la temática del proyecto, contexto y situación actual del mercado. Para ello se recopilaron diferentes tipos de noticias, siendo las más abundantes las referidas a la nueva Ley de protección animal y sus repercusiones, y las que hablaban sobre el aumento del abandono de las mascotas en diferentes épocas del año, tras el COVID, tras la nueva ley, etc.

6.3.1.2. Investigación de la competencia

En esta sección se examinarán y evaluarán las actividades y estrategias de nuestra futura competencia. Para ello se estableció el servicio que nosotros queríamos darle a nuestro producto y los competidores que nos encontraríamos. Usar sus webs y ver cómo mejorarlas para así generar la nuestra con una usabilidad mucho más sencilla, cuantos menos clicks, mucho mejor. Se analizaron las web de;



- Wallamascotas

Imagen: logo

<https://www.wallamascotas.com/web3/>



- PROA

Imagen: logo

<https://www.proaweb.org/>

Además, tanto esta sección como la anterior nos ayudó a descubrir la tendencia de la temática investigada.

6.3.1.3. Investigación del usuario

En esta última sección se realizará un estudio profundo sobre los usuarios finales o al menos relevantes para el proyecto. Se busca comprender sus expectativas, necesidades, comportamientos y experiencias para poder diseñar soluciones que estén alineadas con sus expectativas y realidades.

En esta parte seleccionamos 4 posibles perfiles de usuarios; un adulto soltero, una pareja sin hijos, una familia con hijos y una protectora.

6.3.2. Definir

En esta segunda etapa, organizaremos toda la información recopilada para identificar todas las áreas de oportunidad desde la que podamos ofrecer soluciones relevantes para los deseos y necesidades del usuario.

Para ello se ha creado un User Persona donde podremos observar la vida y personalidad de nuestro usuario elegido para estudio. Unos apartados con los objetivos, frustraciones y motivaciones que puede tener el usuario para utilizar nuestro producto y a partir de las frustraciones, generar unos POV “Point Of View” o “Puntos De Vista” donde se analiza las necesidades que nuestro producto debería de cubrir.

6.3.3. Idear

Una vez establecido el reto, pasaremos a la parte de generar ideas, en forma de preguntas y soluciones, a través de actividades creativas como “lluvias de ideas”. El objetivo es generar el mayor número de ideas y que estas puedan llegar a generar otras nuevas permitiendo al equipo poder enfocarse en las más prácticas e innovadoras.

Para ello se generaron 4 “How Might we” o “¿Cómo podríamos nosotros.....? y buscar soluciones a esos problemas del usuario. Todas estas ideas son pasadas por una evaluación conjunta donde se votan las más válidas y se les da una importancia dentro de un diagrama

que relacione el impacto de esa idea en el usuario con el esfuerzo que tenga que realizar el desarrollador para llevarla a cabo.

6.3.4. Prototipar

En esta cuarta fase se pasará al prototipado de las ideas seleccionadas, donde se le dará forma a estas. Se busca producir una versión temprana, económica y reducida del producto, un prototipo para fallar rápido y barato, buscando que el usuario, cuanto antes, nos indique si el camino que estamos tomando en el diseño de la solución es adecuado o no.

Asimismo, esta parte se podría dividir en 2, una donde se analiza la funcionalidad del servicio a ofrecer, los pasos a seguir para llegar a una meta o la diferente funcionalidad que se pueda ofrecer y otra parte más referida al diseño, a lo visual, de la cual se habla más adelante en el punto 6.4 Design System.

6.3.5. Testear

Última fase del Design Thinking y el momento en el que mostramos nuestro prototipo de solución al arquetipo para el que se está diseñando. Pruebas donde los comentarios de los usuarios son la base de todo. Se utilizan varios métodos para poder testear nuestro producto, donde las más importantes son;

- Análisis heurístico; revisión del producto por parte de un experto en usabilidad tratando de analizar cómo responde a las 10 heurísticas de usabilidad de Jakob Nielsen.(Visibilidad, coincidencia entre sistema y mundo real, usuario con control y libertad, consistencia, prevención de errores, reconocimiento, flexibilidad y eficiencia, estética, ayuda en errores y ayuda y documentación).
- Pruebas de usabilidad; metodología donde un usuario utiliza la interfaz para realizar tareas. Permitirá observar el comportamiento del participante, sus expresiones faciales y sus comentarios. Este análisis debe aportar soluciones y

recomendaciones de mejora. Una vez realizados los cambios, habrá que volver a testar. Estas pruebas de mejora son; definición del alcance del test, definición del tipo de test a realizar, definición del perfil de los participantes, definición del guión y las tareas, realizar las sesiones de prueba, su grabación y para finalizar el análisis de los resultados obtenidos.

6.4. Design System

Para esta fase del proyecto hemos hecho uso de la herramienta Figma para diseñar la página web. En el momento de diseñar el proyecto hemos cogido como referencia los ejemplos que hicimos en clase de Diseño de Interfaces, así como también hemos intentado tener en cuenta las buenas prácticas UX para crear un diseño acorde a las necesidades de nuestros usuarios.

Aunque nos hubiera gustado crear cada una de las vistas que teníamos planeado hacer, con cada unos de sus detalles, debido al tiempo ajustado, hemos decidido desarrollar únicamente los mockups más importantes, que reflejaran gran parte de las funcionalidades que queríamos desarrollar. Esto incluye, las páginas como la Home, los listados, los perfiles de usuario, y los elementos comunes principales, como tarjetas, tablas, botones y formularios.

Link al diseño en figma:

<https://www.figma.com/file/rSAJl6dwHfhr5zegtUcaaF/Web-Adopciones?type=design&node-id=530-515&mode=design&t=lHk7x8MVzDhPguve-0>

6.4.1. Guía de estilos

En lo que a los estilos se refiere, hemos optado por un estilo sencillo y elegante. Hemos elegido el color blanco para el fondo de la web, como color principal porque transmite limpieza y simplicidad, y contrasta con el color rojo vino que hemos escogido como color de contraste, que a su vez, son los colores del logo de Happy Paws.



Paleta:



Imagen: Paleta de colores

Logo: contiene los colores corporativos de Happy Paws

Rojo: #8E0B0B

Negro: #1A191F



Botones principales:



Imagen: botones principales

Botones de acciones:



Imagen: botones de acciones

Tarjetas:

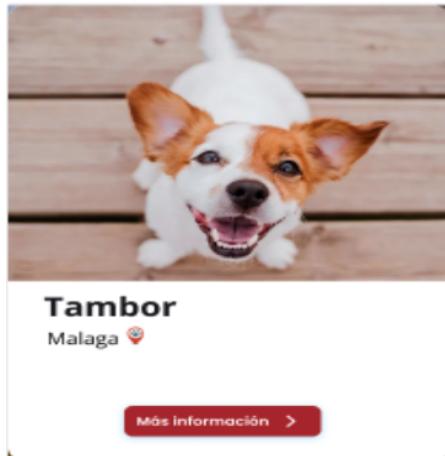


Imagen: tarjetas

Tablas:

ID Adopción	Animal	Adoptante	Fecha	ID Animal	Formulario	Aceptar
#20462	Neo	Juan González	13/05/2022	0013	<button>Ver</button>	<button>Aceptar</button>
#18933	Tambor	Irene Pérez	22/05/2022	0014	<button>Ver</button>	<button>Aceptar</button>
#45169	Rex	David Menéndez	15/06/2022	0015	<button>Ver</button>	<button>Aceptar</button>
#34304	Nala	Raul Moreno	06/09/2022	0016	<button>Ver</button>	<button>Aceptar</button>

Imagen: Tablas

Formularios:

Nombre

Especie

Imagen: formularios

6.4.2. Mockups

Imagen: [Mockups de Figma](#)

Imagen: [Mockups de Figma](#)

Perfil Protectora

Perfil Protectora

Arca de Noe

Últimos animales

Última información

Últimas adopciones

Últimas adopciones completadas

Últimas adopciones

Última información

Últimas adopciones

Últimas adopciones completadas

Nombre	Raza	Sexo	Edad	Adoptado	Aprobado	Rechazado
Arca	Perro	Hembra	1 año	Si	Verificado	No
Noe	Gato	macho	1 año	Si	Verificado	No
Sam	Perro	macho	1 año	Si	Verificado	No
Lucas	Gato	macho	1 año	Si	Verificado	No

Imagen: [Mockups de Figma](#)

Añadir animal

Añadir animal

Nombre

Especie

Sexo

Tamaño

Raza

Código disponible

Descripción

Login Pop-UP

Bienvenido

¡Desechame como favorito!

Último animal

Última información

Última adopción

Última adopción completa

Última adopción

Última información

Últimas adopciones

Últimas adopciones completadas

Imagen: [Mockups de Figma](#)

6.5. Visual Studio Code. Configuración del proyecto.

Extensiones / Dependencias:

- Extension pack for Java
- Maven for Java
- Spring Boot Extension Pack (VMware)
- Material Icon Theme
- MySQL Driver
- Spring Data JPA
- Spring Web Web
- Spring Boot DevTools
- Spring Security
- Json Web Token

Proyecto: SpringBoot Inizializer>Maven 3.2.0>Java>Jar>17.0

https://www.youtube.com/watch?v=uG_3lSaa2lc

6.6. Angular FrontEnd.

Primero de todo, necesitamos instalar angular CLI para poder interactuar con el proyecto. Ejemplo de comando en linux:

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

-La versión de Angular que usamos en el proyecto es la 16.2.

-Para inicializar el proyecto ejecutamos: `ng serve —open`

Módulos usados en el proyecto:

- [Axios](#) - Para realizar requests HTTP
- [angular/router](#) - Para gestionar las rutas en el frontend
- [auth0/angular-jwt](#) - Para interactuar con los tokens JWT
- [sweetalert2](#) - Para mostrar alertas en formato pop-up
- [angular/material](#) - Para usar componentes de material design
- [bootstrap](#) - Para el uso de componentes y del grid system
- [angular/forms](#) - Para crear y manejar formularios

Estructura:

Dentro del proyecto de angular encontramos tres tipos de carpetas:

- **Componentes:** Html de las distintas páginas y partes de nuestra web, así como la lógica referente a los mismos.
- **Servicios:** Clases para interactuar con por ejemplo el backend o diferentes lógicas específicas de nuestras entidades.
- **Entidades:** Las clases que definen conceptualmente nuestros objetos y los datos que contienen.

7. Funcionalidades e Implementación

7.1. Funcionalidad Usuarios

1. Tipos de Usuarios:

a. Administrador (Admin):

Características:

- Acceso a funciones administrativas.
- Puede gestionar protectoras, adoptantes y animales.
- Puede visualizar las adopciones.
- Comparte características de almacenamiento de datos con los adoptantes.

Base de Datos (BDD):

- Se almacena en la tabla USUARIOS.
- Rol de administrador asignado con valor 1.

b. Adoptante:

Características:

- Puede buscar y solicitar adopciones de animales, además puede llenar el cuestionario de preadopción, así como ver el estado de sus adopciones solicitadas.
- Comparte características de almacenamiento de datos con el administrador.

Base de Datos (BDD):

- Se almacena en la tabla USUARIOS.
- Rol de adoptante asignado con valor 3.



c. Protectora:

Características:

- Acceso a funciones específicas de las protectoras como:
 - Dar de alta animales (editar y borrar).
 - Ver solicitudes de adopción en curso, aprobar, rechazar adopciones y visualizar el histórico.
 - Visualizar el formulario de pre-adopción del solicitante.

Base de Datos (BDD):

- Se almacena en la tabla USUARIOS, los datos del gestor de la protectora.
- Se almacenan los datos de la protectora en sí, en la tabla PROTECTORAS.
- Rol de protectora asignado con valor 2.
- Relación mediante el ID de usuario.

2. Autenticación y autorización

a. JWT (JSON Web Token):

- Utilizado para la autenticación de usuarios.
- Al realizar el login/registro, se genera un token JWT que contiene información del usuario (ID, rol, etc.).
- Este token se utiliza en las solicitudes posteriores para verificar la identidad del usuario.
- El token del usuario se guarda en el local storage del navegador y caduca en 1 hora desde la expedición del token.

b. Proceso de Registro:

Usuarios Comunes (Admin y Adoptante):

- Rellenan los datos de usuario común.
- Al usuario adoptante se le asigna el rol 3.
- Al usuario admin, se le asignará el rol 1 directamente en la BBDD, ya que no hemos implementado un registro específico para estos usuarios.

Protectoras:

- Rellenan los datos de usuario común.
- Rellenan los datos adicionales de protectora.
- Se les asigna el rol de protectora (2).

3. Implementación

Tal y como hemos indicado en el punto anterior, tanto el registro como el inicio de sesión han sido implementados usando JWT (Json Web Token). Para aplicar JWT hemos usado la librería de JWT en auth0, tanto en el backend como en el frontend. Los tokens se generan tanto en el login como en el registro, para que el usuario no tenga que hacer login después de registrarse. A continuación explicamos en un diagrama, como funciona JWT dentro de nuestro proyecto, [aquí](#).

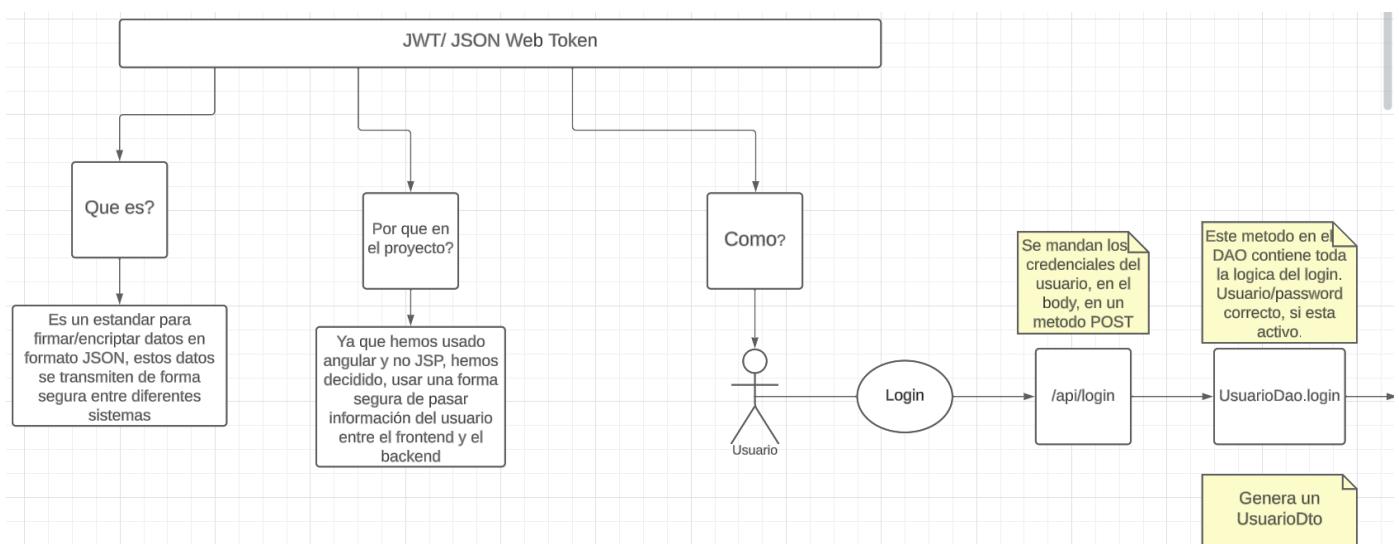
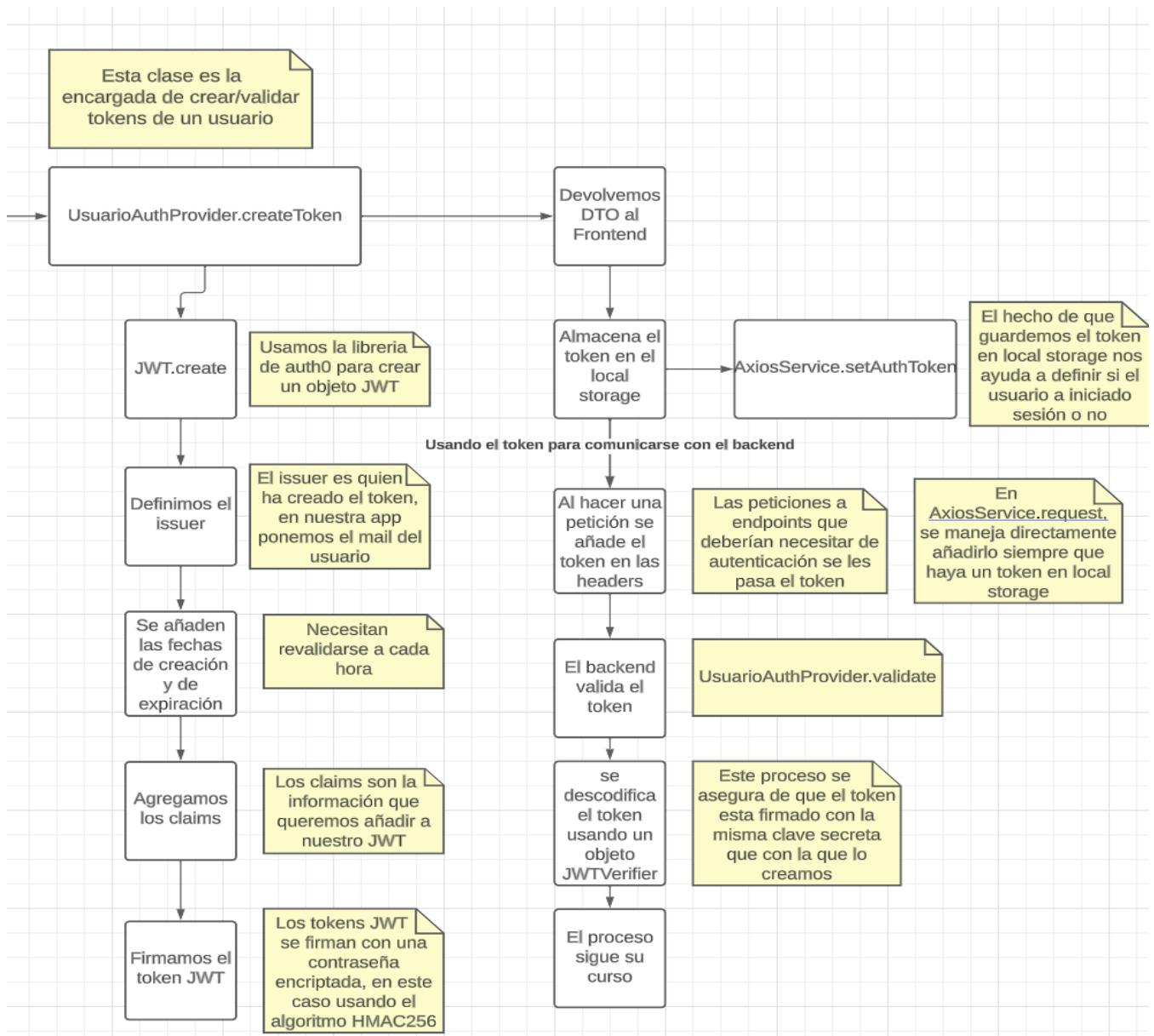


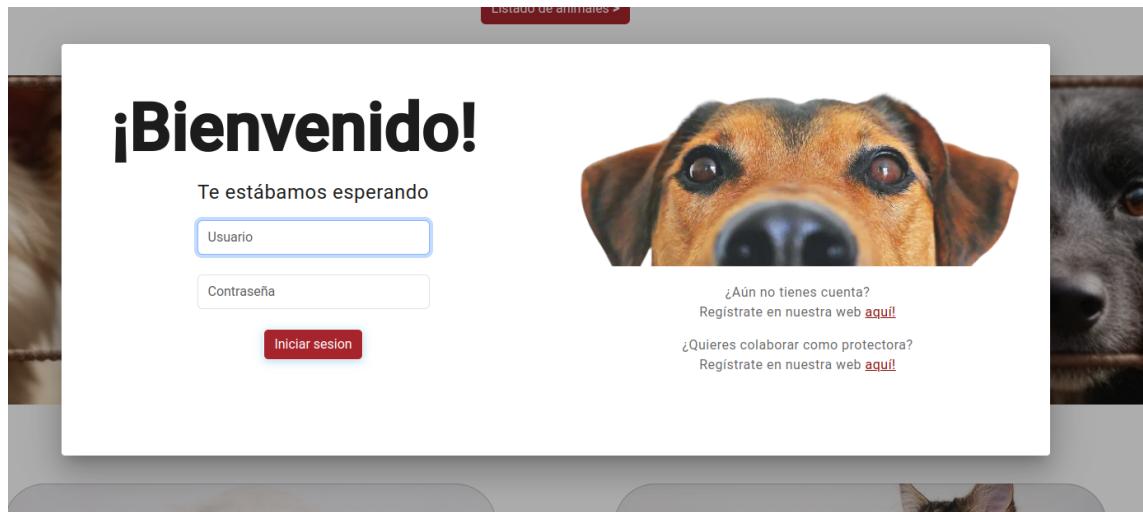
Imagen: [Diagrama explicación JWT](#)

Imagen: [Diagrama explicación JWT](#)

Login:

Para implementar el login en el frontend, hemos creado un modal usando la librería de Material Angular. Para ello, hemos creado el componente login y en los servicios `usuarioService`, `axiosService` y `authService`, los diferentes métodos para extraer datos del token, comunicarnos con el backend y saber si un usuario ha hecho login o no.

Imagen: Login



Validaciones: [OBJETO]

- El método login busca un usuario por su email, y comprueba que existe y que la contraseña es correcta.
- Se comprueba que el usuario/protectora está activo, si no está activo se muestra un error.
- De los datos que se extraen en el token, se hacen las validaciones y condiciones en el resto de componentes de la web, para saber si un usuario tiene permisos para acceder a una página o realizar diferentes acciones.

```
Bienvenido, {{user?.nombre}}
</button>
<ul class="dropdown-menu dropdown-menu-start" [ngswitch]="user?.rol">
  <li *ngSwitchCase="'Adoptante'"><a class="dropdown-item" href="adoptante/gestion">Mi Perfil</a></li>
  <li *ngSwitchCase="'Protectora'"><a class="dropdown-item" href="protectora/gestion">Mi Perfil</a></li>
  <li *ngSwitchCase="'Administrador'"><a class="dropdown-item" href="admin/gestion">Mi Panel</a></li>
  <li><a class="dropdown-item" (click)="logout()">Logout</a></li>
</ul>
</div>
</li>
```

Imagen: código fuente

Registro:

Para implementar el registro en el frontend, hemos creado dos componentes, uno para registro usuario y otro para registro Protectora. Mediante el uso de la librería de formularios de angular, hemos construido el formulario y gestionado las validaciones. Cada uno de los

componentes tiene un método para controlar el evento submit del formulario, recoger los datos y mandarlos al backend usando el servicio de Axios.

Registro de Protectora

Detalles de usuario

i Rellene la información de la persona responsable de la protectora.

Nombre
Irene

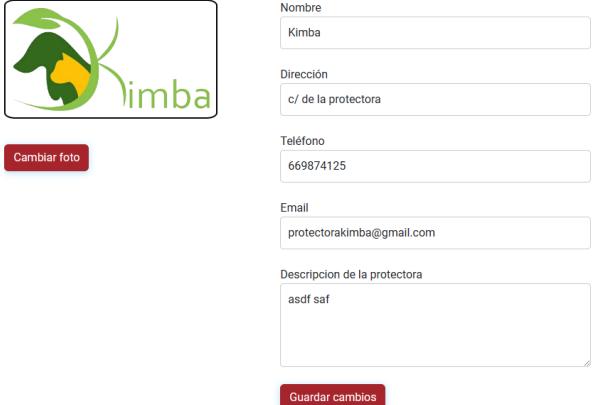
Apellidos
Introduce tus Apellidos

Apellidos es obligatorio.

Imagen: Registro usuario

7.2. Funcionalidad Protectoras

Cuando una protectora se da de alta, puede acceder a su perfil donde verá el panel de administración. Desde esta zona puede: modificar los datos, ver sus animales, dar de alta, baja, modificarlos, subir y borrar fotos. También puede gestionar las peticiones de adopción para aceptarlas o rechazarlas, y ver el historial de las adopciones realizadas o rechazadas.

Funcionalidad Protectoras - Modificar Datos		
 <p>Kimba</p> <p>Dirección: c/ de la protectora Email: protectorakimba@gmail.com Teléfono: 669874125</p> <p>Modificar Datos</p> <p>Imagen: Perfil Protectora</p>	<p>El perfil muestra los datos básicos de la protectora y permite ir al formulario para modificar los datos.</p>	
 <p>Imagen: Formulario modificar protectora</p>	<p>Al pulsar en modificar datos, tenemos un formulario que recoge los datos de la protectora actual y permite cambiarlos.</p> <p>También contiene un botón que permite cambiar la foto cuya función se detalla más adelante..</p>	
<p>Dar de baja esta protectora Activar protectora</p> <p>Imagen: Botones modificar protectora</p>	<p>Los botones inferiores permiten dar de baja la Protectora. Para ello se cambia el estado a Activo o Inactivo.</p> <p>Una protectora en estado inactivo no aparece en el listado de protectoras de la web.</p>	

Funcionalidad Protectoras - Animales en adopción

Animales	Peticiones	Adopciones
Busqueda: <input type="text" value="Nombre del animal"/> <input type="button" value="Buscar"/> <input type="button" value="Ver todos"/>  <p>Chemdo</p> <p>Disponibilidad: Animal disponible.</p> <p>Sexo: Macho Raza: Mestizo Tamaño: Mediano Envío: NO se puede enviar</p> <p><input type="button" value="Modificar"/> <input type="button" value="Borrar"/></p>	<input type="button" value="Dar de alta un animal"/>	<p>Imagen: Busqueda animal en adopción Protectora</p> <p>Desde esta pestaña la protectora puede:</p> <ul style="list-style-type: none"> ● Ver lista de sus animales ● Buscar animales por nombre. ● Dar de alta un animal ● Modificar un animal y añadir o borrar fotos. ● Borrar un Animal
<p>Imagen: Animales de la protectora</p>		

Funcionalidad Protectoras - Peticiones de adopción

ID Adopción	Animal	Adoptante	Email Adoptante	ID Animal	Formulario	Acciones
13	Mia	Juan Perez	usuarioNuevo@usuario.com	6	<input type="button" value="Ver"/>	<input type="button" value="Aceptar"/> <input type="button" value="Rechazar"/>

Imagen: Peticiones de adopción de una protectora

Obtenemos un listado de todas las peticiones que un usuario adoptante haya realizado a un animal de esa protectora y permite:

- Ver lista de peticiones con email del adoptante y animal al que se ha solicitado adopción.
- Ver el cuestionario de adopción llenado por el usuario.
- Aceptar o rechazar la adopción.

7.3. Funcionalidad Animales

En la parte de Animal se ha desarrollado todo el código referente a la manipulación de los animales en la aplicación, desde la búsqueda por su id como el filtrado dependiendo de sus atributos, pasando por creación, modificación, borrado, etc.

Hay 2 partes fundamentales en esta aplicación además de la base de datos, la parte servidor y la parte cliente.

Comenzando por la parte del servidor bajo el marco de desarrollo Spring Boot, Back-end, y partiendo de que la entity (clase) Animal ya estaba creada, se procedió a crear los siguientes componentes para el uso y manipulación de esos datos importados desde la base de datos en esta;

- Interface IAnimalDao, donde se definen los métodos de acceso a datos para la entidad

```
public interface IAnimalDao {
    boolean altaAnimal (Animal animal);
    boolean modificarAnimal (Animal animal);
    boolean enabledAnimal (Animal animal);
    boolean borrarAnimal (int idanimal);

    Animal buscarAnimalId (int idanimal);
    List<Animal> buscarTodos();
    List<Animal> buscarPorNombreContiene(String nombre);

    List<Animal> buscarPorIdMunicipio (int idmunicipio);
    List<Animal> buscarPorIdProvincia (int idprovincia);
    List<Animal> buscarPorIdProtectora (int idprotectora);
```

Imagen: código fuente IAnimalDao

- Clase AnimalDao, donde se implementa la interfaz anteriormente creada y se codifica la lógica concreta para realizar la función requerida.

```
@Service
public class AnimalDao implements IAnimalDao{
    @Autowired
    AnimalRepository aniRepo;
    @Autowired
    MultimediaRepository multiRepo;

    @Override
    public boolean altaAnimal(Animal animal) {
        try{
            aniRepo.save(animal);
            return true;
        } catch (Exception e){
            e.printStackTrace();
            return false;
        }
    }
}
```

Imagen: código fuente AnimalDao

- Interface AnimalRepository, extendida de JpaRepository y donde se han personalizado las diferentes consultas a utilizar; buscar por el Id del municipio, por su raza así como el filtrado dependiendo de diferentes atributos.

```
public interface AnimalRepository extends JpaRepository<Animal, Integer>{

    // Query para buscar por el Id del municipio
    @Query(value = "SELECT a FROM Animal a WHERE a.municipio.id = :idmunicipio")
    List<Animal> buscarPorIdMunicipio(int idmunicipio);

    //Query para buscar por el Id de la provincia
    @Query(value = "SELECT a FROM Animal a INNER JOIN a.municipio m INNER JOIN m.provincia p WHERE p.provincia.idprovincia = :idprovincia")
    List<Animal> buscarPorIdProvincia(int idprovincia);

    // Query para buscar todos los animales de una protectora por ID
    @Query(value = "SELECT a FROM Animal a WHERE a.protectora.idprotectora = :idprotectora")
    List<Animal> buscarPorIdProtectora(int idprotectora);
```

Imagen: código fuente AnimalRepository

- Clase AnimalRestController, donde se manejan las solicitudes HTTP relacionadas con la entidad y donde se inyectan la clase AnimalDao y MultimediaDao para así interactuar con los datos.

```
@RestController
@RequestMapping("/animales")
@CrossOrigin(origins = "*")
public class AnimalRestController {

    @Autowired
    private AnimalDao aniDao;

    @Autowired
    MultimediaDao multiDao;

    // Controlador para el listado de animales
    @GetMapping(path = "/listado", produces = "application/json")
    public ResponseEntity<List<Animal>> listadoAnimales() {

        List<Animal> listado = aniDao.buscarTodos();
        if (listado != null && !listado.isEmpty()) {
            return ResponseEntity.ok(listado);
        } else {
            return ResponseEntity.notFound().build();
        }
    }
}
```

Imagen: código fuente AnimalRestController

Al igual que con la clase Animal, también se desarrollaron las clases e interfaces correspondientes a las clases Especie, Multimedia, Municipio, Provincia, Raza, Sexo y Tamaño ya que son tablas relacionadas directamente con la clase Animal y nos seria util para un futuro.

Una vez implementadas todas estas clases e interfaces, siendo modificados algunos de estos métodos y funciones con posterioridad para adecuarlo a la funcionalidad de la aplicación web, se comprobó su correcto funcionamiento y se pasó al desarrollo de la parte de cliente, el Front-end.

Para el desarrollo del Front-end se optó por emplear el marco de desarrollo Angular, que nos proporcionará las herramientas necesarias para crear una interfaz de usuario dinámica e interactiva.

Lo primero de todo fue la creación de la clase Animal.ts donde representamos la estructura de datos de la aplicación y se define su formato. Así mismo, con la clase Especie, Multimedia, Municipio, Provincia, Raza, Sexo y Tamaño.

```
export class Animal {  
  
    idanimal: number;  
    descripcion: string;  
    enabled: boolean;  
    envio: boolean;  
    fechaAlta: Date;  
    fechaNacimiento: Date;  
    municipio: Municipio | any;  
    protectora: Protectora;  
    nombre: string;  
    raza: Raza | any;  
    sexo: Sexo | any;  
    tamano: Tamano | any;  
    fecha_enabled: Date | any;  
}
```

Imagen: código fuente Animal.ts

Una vez creados las clases anteriores, se pasó a crear el servicio, animal.service.ts, la cual nos ayudará a compartir datos, lógica de negocio y funcionalidades entre los diferentes componentes de la aplicación. De igual manera, se crearon los servicios para la clase Especie, Multimedia, Municipio, Provincia, Raza, Sexo y Tamaño.

```

@ Injectable({
  providedIn: 'root'
})

export class AnimalService {

  //URL del servicio Rest
  readonly endpoint = axios.defaults.baseURL;
  //readonly endpoint = 'http://localhost:8087';

  /**
   * Encargado de hacer las peticiones HTTP a nuestro servicio REST
   * @param _httpClient
   */
  //constructor(private _httpClient : HttpClient, private axiosService: AxiosService) { }
  constructor(private _httpClient : HttpClient) { }

  // Método que lista todos los animales del servicio Rest
  public listarAnimales(): Observable<any> {
    return this._httpClient.get(`${this.endpoint}/animales/listado`)
      .pipe(catchError(this.manejarError));
  }
}

```

Imagen: código fuente animal.service.ts

Con las entidades y los servicios (aún sin terminar totalmente, ya que se fueron modificando según las necesidades de la funcionalidad de la aplicación), fue la hora de empezar a crear los componentes.

Los componentes son como etiquetas Html creadas por nosotros mismos, son bloques que encapsulan la lógica y la presentación relacionada con una parte específica de la interfaz del usuario, dándole así más dinamismo a la aplicación y evitando recargar toda la página de nuevo. Cada componente tiene su propio ciclo de vida y puede comunicarse con otros mediante eventos o servicios.

Cada componentes consta de 4 archivos;

- Archivo css, para la edición de su estilo
- Archivo html, para la estructura del componente
- Archivo spect.ts para las pruebas
- Archivo ts, para desarrollar la lógica del componente.

Los componentes creados fueron;

- Animal

Este componente fue creado básicamente para mostrar una foto del animal en cuestión, como sus datos como a que protectora pertenece, provincia y otros detalles.



Perla
Protectora: Refugio
Provincia: Soria
Sexo: Hembra
Raza: Caniche
Tamaño: Pequeño
Envío: Sí
Ver más información

Imagen: componente animal

Además, cuenta con un botón al final que nos dirigirá al siguiente componente;

- Animal detallado

Este componente, a diferencia del anterior, muestra una información más amplia y detallada de cada animal además de todas las fotos que puedan estar relacionadas con su identidad. Además se le añade un botón para volver atrás y volver a esa lista de componentes de animal.

[← Volver al listado de animales](#)



Conoce a ... Perla
 Fecha nacimiento: Jun 11, 2022
 Provincia: Soria
 Protectora: Refugio
 Envío: Puede viajar

DETALLES

Especie: Perro
 Raza: Caniche
 Sexo: Hembra
 Tamaño: Pequeño
 Edad: 1

DESCRIPCIÓN

Perla es una perita muy simpática a la que le encanta jugar y disfrutar de la naturaleza. En casa es todo una señorita muy limpia y ordenada pero cuando sale suelta toda la energía que lleva dentro.






Imagen: componente animal detallado

- Lista animales

El componente lista de animales tiene como función principal iterar todos los animales que tengamos en la base de datos pero con ciertas particularidades.

```
<div *ngFor="let animal of listaAnimalesHabilitados" class="col-4 mb-4">
  <app-animal [animal]="animal" [fotos]="listaFotos[animal.idanimal]"></app-animal>
</div>
```

Imagen : iteración componente animal

En este caso, la particularidad es referida a si su atributo “enabled” es true o false, ya que si un animal tiene este atributo en “false” significa que no está disponible para adoptar, con lo cual no tendría sentido que apareciese en este listado. Esto se consigue creando de una lista de animales original y partiendo de ella se crea otra lista con un filtro aplicado para solo los animales con la propiedad “enabled” en true.

```
public listaAnimalesHabilitados = this.listaAnimales.filter(animal => animal.enabled);
```

Imagen: lista animales habilitados en lista-animales.component.ts

Además, este componente cuenta con otro filtro pero independiente por diferente atributos, los cuales puedes seleccionar uno, dos o todos. Todo ello conseguido gracias a métodos que buscan, filtran y actualizan los datos del animal

```
// Método para filtrar animales según sus atributos
public filtrarAnimales(
  especie: string,
  raza: string,
  sexo: string,
  tamano: string,
  provincia: string,
  envio: boolean
): Observable<any> {
  let params = new HttpParams();
  if (especie) params = params.set('especie', especie);
  if (raza) params = params.set('raza', raza);
  if (sexo) params = params.set('sexo', sexo);
  if (tamano) params = params.set('tamano', tamano);
  if (provincia) params = params.set('provincia', provincia);
  if (envio != null && envio != undefined) params = params.set('envio', envio ? 'true' : 'false');
  return this._httpClient.get<any>(`${this.endpoint}/animales/filtrar`, { params });
}
```

Imagen: método filtrar en animal.service

```
// Método del filtro
public buscar(): void {
  this._animalService
    .filtrarAnimales(this.especie, this.raza, this.sexo, this.tamano, this.provincia, this.envio)
    .subscribe({
      next: (animales) => {
        this.listaAnimales = animales;
        this.actualizarListaAnimales();
      },
      error: (err) => console.error(err)
    });
}

// Métodos de seteo

public actualizarEspecie(event: any) {
  this.especie = event.target.value;
}

public actualizarRaza(event: any) {
  this.raza = event.target.value;
}
```

Imagen: métodos en componente lista animales

- Lista solo Perros

Este componente es una lista similar a la anterior pero en este caso, en animal.service.ts se ha creado un método que llame al back donde tenemos el controlador y las querys para solo buscar perros.

```
// Método que lista todos los animales que son Perros
public listarPerros(): Observable<any> {
    return this._httpClient.get(` ${this.endpoint}/animales/buscar/soloperros`)
        .pipe( catchError(this.manejarError));
}
```

Imagen: metro listar perros en animal.service

```
// Controlador para buscar Perros
@GetMapping(path = "/buscar/soloperros", produces = "application/json")
public ResponseEntity<List<Animal>> buscarSoloPerros() {
    List<Animal> listado = aniDao.buscarSoloPerros();
    if (listado != null && !listado.isEmpty()) {
        return ResponseEntity.ok(listado);
    } else {
        return ResponseEntity.notFound().build();
    }
}
```

Imagen: controlador buscar solo perros en AnimalRestController

- Lista solo Gatos

Este componente es exactamente igual que el anterior pero referido a animales de la especie Gato.

- Gestión animal

Este componente es otro listado de animales pero en este caso, solo se muestran los animales que pertenecen al usuario/protectora que está registrado. Para ello, se ha tenido que generar otra lógica diferente para primero obtener los datos de usuario y así poder incluir esta variable a otro nuevo método que filtre animales dependiendo del Id de la protectora.

```

// Init

ngOnInit():void {
  this.obtenerIdUsuario();
  this.obtenerIdProtectora(this.idUsuario);
}

// Métodos de Datos

public obtenerIdUsuario():void{
  const user: any = this._usuarioService.getUserData();
  this.idUsuario = user.id;
}

public obtenerIdProtectora(idUsuario: number): void {
  this._protectoraService.obtenerProtectoraPorIdUsuario(idUsuario).pipe(
    switchMap(data => {
      this.idProtectora = data.idprotectora;
      return this._animalService.listarAnimalPorIdProtectora(this.idProtectora);
    })
  ).subscribe({
    next: (res) => { this.listaAnimales = res; },
    error: (err) => { console.error(err); },
    complete: () => { this.obtenerFotosAnimales(); }
  });
}

public obtenerAnimalesIdProtectora(idProtectora:number):void{
  this._animalService.listarAnimalPorIdProtectora(idProtectora).subscribe(dato => {
    this.listaAnimales = dato;
  });
}

```

Imagen: métodos de obtención de datos de usuario

Además, este componente cuenta con un filtro sencillo por nombre de animal y un botón para crear un nuevo animal, que será el siguiente componente.

- Alta animal

Este componente ha sido creado con la finalidad de poder dar de alta animales en nuestra base de datos. Es simplemente un formulario donde se introducen datos (todos obligatorios) y con ello dar de alta un nuevo animal. Se disponía de un componente que se podría adaptar para una vez creado el animal se le pudieran agregar fotos pero tras conversarlo, se decidió que se añadiera al menos una foto en el momento del alta. Para ello se creó un atributo fotos que seria un `FileList` y junto al input del cuestionario y el método para seleccionarlas, se diera de alta todo a la vez.

```

public seleccionarFotos(event: any): void {
  this.fotos = event.target.files;
}

private guardarFotos(idAnimal: string): void {
  if (this.fotos) {
    const formData = new FormData();
    // Añado las fotos.
    for (let i = 0; i < this.fotos.length; i++) { formData.append('files', this.fotos[i]); }
    // Añado el ID del animal.
    formData.append("id", idAnimal);
    // Guardo las fotos en la bbdd.
    this._multimediaService.subirFotosAnimal(formData).subscribe({
      error: (err) => { console.error('error:' + err); },
      complete: () => {
        Swal.fire({
          icon: 'success',
          title: '¡¡ Perfecto !!',
          text: 'La mascota ha sido creada satisfactoriamente',
          showConfirmButton: false,
          timer: 2000
        })
        .then(()=> {
          this.router.navigateByUrl('/', {skipLocationChange: true}).then(() => {
            this.router.navigate(['/protectora/gestion']);
          })
        })
      }
    });
  }
}

```

Imagen: métodos para seleccionar fotos y guardarlas en alta animal.ts

A su vez, se añadió una alerta que al finalizar el proceso completo nos informará del resultado.

- Animal gestión

Este componente es muy similar al primero pero en vez de mostrar unos datos interesantes para el usuario adoptante, se ha rediseñado para que muestre datos interesantes para la protectora, como por ejemplo su estado o añadir al final 2 botones.



Thorvellino

Disponibilidad:
Animal disponible.

Sexo: Macho

Raza: Pitbull

Tamaño: Mediano

Envío: NO se puede enviar

[Modificar](#)

[Borrar](#)

Imagen: componente animal gestión

Uno de estos botones tiene una función de borrado del animal, el cual primero elimina las fotos que tenga asociadas ese animal y después borra el animal y el otro tiene la función de poder modificar el animal que es el siguiente componente.

```
@Override
public boolean borrarAnimal(int id) {
    try {
        Optional<Animal> optionalAnimal = aniRepo.findById(id);

        if (optionalAnimal.isPresent()) {
            Animal animal = optionalAnimal.get();
            // Elimina las fotos asociadas antes de eliminar el animal
            eliminarFotosAsociadas(animal);
            // Ahora puede eliminar el animal
            aniRepo.deleteById(id);
            return true;
        }
        return false;
    } catch (Exception e) {
        e.printStackTrace();
        return false;
    }
}

// Método para eliminar las fotos asociadas de cada animal
private void eliminarFotosAsociadas(Animal animal) {
    List<Multimedia> fotos = multiRepo.todosMultimediasAnimal(animal.getIdanimal());
    for (Multimedia foto : fotos) {
        multiRepo.delete(foto);
    }
}
```

Imagen: métodos para borrar animal y fotos asociadas

- Modificar animal

Este componente lo que hace básicamente es ofrecer un cuestionario con todos los atributos que tiene ese animal y nos da la posibilidad de cambiar cualquiera y guardar cambios.

```
private actualizarAnimal(): void {
    if (this.altaForm.valid) {
        // Crear instancia de Animal y asignar valores
        const animal: Animal = {
            idanimal: this.animal.idanimal,
            descripcion: this.altaForm.get('descripcion')?.value,
            enabled: this.altaForm.get('enabled')?.value,
            envio: this.altaForm.get('envio')?.value,
            fechaAlta: new Date(),
            fechaNacimiento: this.animal.fechaNacimiento,
            municipio: this.obtenerMunicipio(this.altaForm.get('municipio')?.value),
            protectora: this.animal.protectora,
            nombre: this.altaForm.get('nombre')?.value,
            raza: this.obtenerRaza(this.altaForm.get('raza')?.value),
            sexo: this.obtenerSexo(this.altaForm.get('sexo')?.value),
            tamano: this.obtenerTanyo(this.altaForm.get('tamano')?.value),
            fecha_enabled: new Date(),
        };
        this.guardarAnimal(animal);
    }
}
```

Imagen: método que modifica animal en modificar-animal.ts

7.4. Funcionalidad Admin

Cuando un usuario tipo Administrador accede al perfil puede ver y gestionar las Protectoras, Animales, Usuarios y Adopciones. El panel admin tiene una ruta endpoint diferente.



Funcionalidad Admin - Protectoras

Buscar:	<input type="text" value="nombre protectora"/>	<button>Buscar</button>	<button>Ver todas</button>				
ID Protectora	Nombre	Provincia	Estado	Edicion	Cambiar Estado		
9	Propatas	Lugo	Activo	<button>Editar</button>	<button>Activar</button>	<button>Desactivar</button>	Pendiente
1	Protectora 1	Madrid	Inactivo	<button>Editar</button>	<button>Activar</button>	<button>Desactivar</button>	Pendiente
3	Sociedade protectora de animais e plantas de Lugo	Madrid	Activo	<button>Editar</button>	<button>Activar</button>	<button>Desactivar</button>	Pendiente
5	Protectora de Animales de Alcoy	Madrid	Activo	<button>Editar</button>	<button>Activar</button>	<button>Desactivar</button>	Pendiente

Imagen: Listado de protectoras

Obtenemos un listado de todas las protectoras en el que se puede:

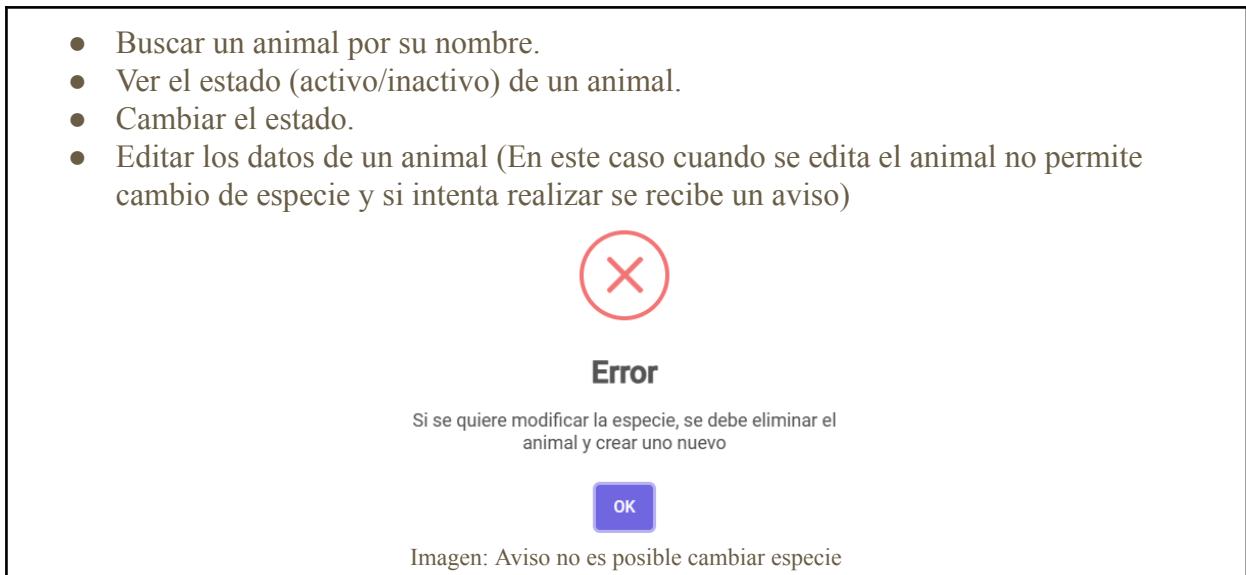
- Buscar una protectora por su nombre.
 - Ver el estado (activo/inactivo/pendiente) de una protectora.
 - Cambiar el estado.
 - Editar los datos de una protectora.

Funcionalidad Admin - Animales

Buscar:		<input type="text" value="nombre animal"/>	<button>Buscar</button>	<button>Ver todos</button>			
ID Animal	Nombre	Provincia	Estado	Protectora	Estado	Cambiar Estado	Editar
6	Valkiriaaaa	Madrid	true	Propatas	Activo	<button>Desactivar</button>	<button>Editar</button>
7	Sol	Castellón/Castelló	false	Propatas	Inactivo	<button>Activar</button>	<button>Editar</button>
8	Chemdo	Gipuzkoa	true	Kimba	Activo	<button>Desactivar</button>	<button>Editar</button>

Imagen: Listado de animales

Obtenemos un listado de todos los animales en el que se puede:



Funcionalidad Admin - Usuarios

ID Usuario	Nombre	Apellidos	Email	Rol	Estado	Editar	Cambiar Estado
1	David	Menéndez	menendez.david@gmail.com	Administrador	Activo	<button>Editar</button>	<button>Desactivar</button>
2	Irene	Pérez	irenperezcano95@gmail.com	Administrador	Inactivo	<button>Editar</button>	<button>Activar</button>
3	Raúl	Moreno	raulmro@hotmail.es	Administrador	Inactivo	<button>Editar</button>	<button>Activar</button>

Imagen: Listado de Usuarios

Obtenemos un listado de todos los usuarios en el que se puede:

- Buscar un usuario por su email..
- Ver el estado (activo/inactivo) de un usuario.
- Cambiar el estado.
- Editar los datos de un usuario.

Funcionalidad Admin - Adopciones

ID Adopción	Animal	Adoptante	ID Animal	Estado
10	Kira	Usuario Gomez	4	Aprobada
11	Neo	Usuario Gomez	5	Rechazada
12	Mia	Usuario Gomez	6	En curso

Imagen: Listado de adopciones

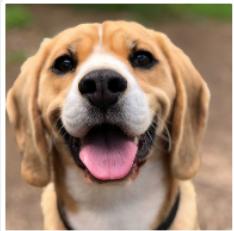
Obtenemos un listado de todas las adopciones y su estado. Se puede buscar por ID

7.5. Funcionalidad Adopciones

Funcionalidad Adopciones

DESCRIPCIÓN

Kira es una perrita encantadora, dispuesta a encontrar un hogar.



[Solicitar Adopción](#)

Imagen: Detalle Animal

El botón solicitar adopción está disponible únicamente para los usuarios de tipo adoptante. Lo comprobamos en el frontend con una sencilla función:

```
<!-- Botón de Solicitar Adopción -->
<div class="text-center" *ngIf="usuario.isAdoptante()">
| <button type="submit" class="btn btn-success" (click)="altaAdopcion()>Solicitar Adopción</button>
</div>
```

Imagen: código fuente

En cuanto se solicita la adopción, se crea una petición en estado en curso. La adopción aparece en el perfil de la protectora, donde esta, podrá aceptarla o rechazarla:



Imagen: Perfil Protectora

ID Adopción	Animal	Adoptante	Email Adoptante	ID Animal	Formulario	Acciones
12	Mia	Usuario Gomez	usuario@mail.com	6	Ver	Aceptar Rechazar



Imagen: Perfil Protectora

ID Adopción	Animal	Adoptante	ID Animal	Estado
10	Kira	Usuario Gomez	4	Aprobada
11	Neo	Usuario Gomez	5	Rechazada
12	Mia	Usuario Gomez	6	En curso

El usuario adoptante puede comprobar el estado de sus adopciones, en su perfil:

En caso de que ya exista una adopción para ese animal, por parte de ese usuario, saltará un error:

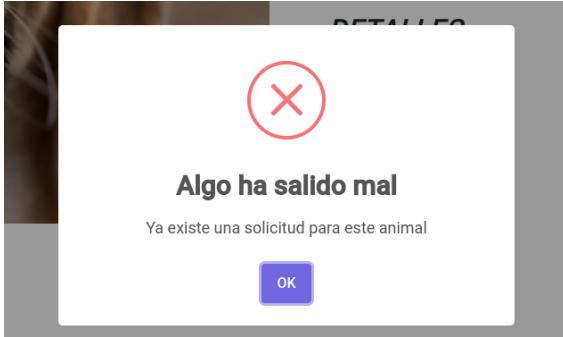


Imagen: Alerta - Ya existe una adopción

7.6. Funcionalidad Formulario

Funcionalidad Formulario

Formulario de adopción

Para poder solicitar la adopción de un animal, primero debes llenar el cuestionario de adopción. Las respuestas las recibirá la protectora responsable del animal, para que puedan valorar si eres apto para adoptar a uno de sus animales. Una vez hayas completado el cuestionario, podrás hacer solicitudes de adopciones para cualquier animal

Puede consultar el estado de sus solicitudes de adopción en el apartado "Mis solicitudes" de su perfil de usuario.

¿Has tenido animales antes? ¿De qué raza, tamaño...? ¿Qué ocurrió con ellos? ¿De qué murieron y a qué edad? ¿Qué relación tuviste con ellos: compañía, guardián...? ¿Los compraste o los adoptaste? ¿Dónde los adoptaste?

test1

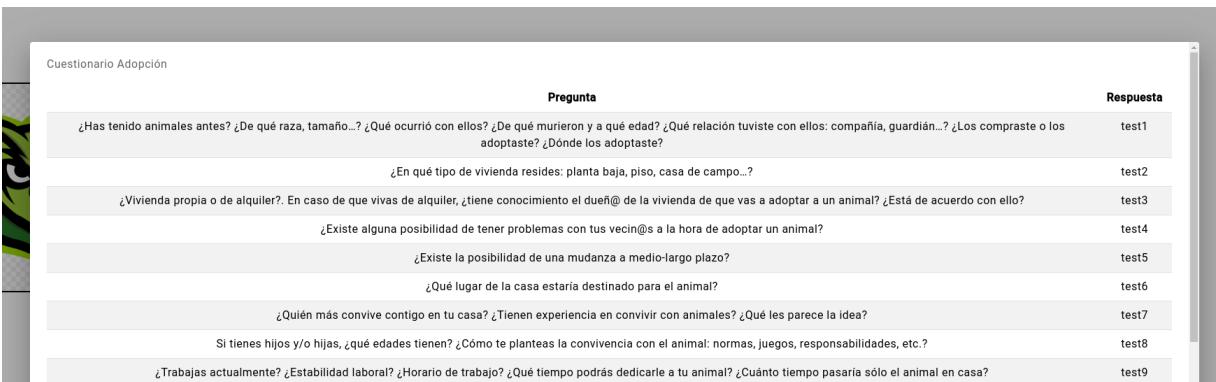
Mis solicitudes

Imagen: Perfil Usuario - Formulario

Dentro del perfil de usuario adoptante, el usuario debe llenar el formulario de adopción, que se mostrará a la protectora, cuando el usuario solicite una petición de adopción.

Las respuestas del formulario, se guardan, y el usuario puede actualizar las respuestas siempre que quiera.

Dentro de la solicitud de adopción, la protectora tiene la opción de ver el formulario del adoptante. Cuando hacemos click en el botón “ver” se abre un modal que trae las preguntas del formulario, con las respuestas del adoptante, a cada pregunta.



Pregunta	Respuesta
¿Has tenido animales antes? ¿De qué raza, tamaño...? ¿Qué ocurrió con ellos? ¿De qué murieron y a qué edad? ¿Qué relación tuviste con ellos: compañía, guardián...? ¿Los compraste o los adoptaste? ¿Dónde los adoptaste?	test1
¿En qué tipo de vivienda resides: planta baja, piso, casa de campo...	test2
¿Vivienda propia o de alquiler?. En caso de que vivas de alquiler, ¿tiene conocimiento el dueñ@ de la vivienda de que vas a adoptar a un animal? ¿Está de acuerdo con ello?	test3
¿Existe alguna posibilidad de tener problemas con tus vecin@s a la hora de adoptar un animal?	test4
¿Existe la posibilidad de una mudanza a medio-largo plazo?	test5
¿Qué lugar de la casa estaría destinado para el animal?	test6
¿Quién más convive contigo en tu casa? ¿Tienen experiencia en convivir con animales? ¿Qué les parece la idea?	test7
Si tienes hijos y/o hijas, ¿qué edades tienen? ¿Cómo te planteas la convivencia con el animal: normas, juegos, responsabilidades, etc.?	test8
¿Trabajas actualmente? ¿Estabilidad laboral? ¿Horario de trabajo? ¿Qué tiempo podrás dedicarle a tu animal? ¿Cuánto tiempo pasaría sólo el animal en casa?	test9

Imagen: Perfil Protectora - Peticiones - Ver formulario

7.7. Funcionalidad Subir Fotos

La aplicación permite subir fotos a dos componentes;

Funcionalidad Subir Fotos - Protectoras	
 <input type="button" value="Cambiar foto"/> Imagen: cambiar foto	Cambiar foto Protectora  <input type="button" value="Subir foto"/> Imagen: Subir foto Protectora
 Foto subida La foto se ha subido con éxito <input type="button" value="OK"/> Imagen: Confirmación foto subida	
Una vez cambiada la foto, se devuelve al panel de administración de la protectora.	

Funcionalidad Subir Fotos - Animales	
 Imagen: botón modificar fotos	<p style="text-align: right;">Mantenimiento de fotos de Valkiria</p> <div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;">  Borrar foto </div> <div style="text-align: center;">  Borrar foto </div> </div> <p style="text-align: center;"> Elegir archivos No se ha seleccionado ningún archivo Subir foto </p> <p style="text-align: right;">Imagen: Mantenimiento fotos Animal</p>
<p>Permite que una protectora, añadir y borrar las fotos de un animal, desde su panel de administración,</p> <p>Cuando se pulsa en borrar una foto, se pide confirmación:</p> <div style="text-align: center;">  <p>¿Estás seguro?</p> <p>No podrás revertir esto!</p> <p>Sí, borrarlo! Cancel</p> </div> <p style="text-align: center;">Imagen: confirmación borrar foto</p> <p>Cuando se sube una foto, una ventana modal confirma la acción.</p> <div style="text-align: center;">  <p>Foto subida</p> <p>La foto se ha subido con éxito</p> <p>OK</p> </div> <p style="text-align: center;">Imagen: confirmación foto subida/borrada</p>	

Para esta implementación, las fotos de los animales se guardan en la tabla multimedia que almacena el Id del animal y el enlace a la imagen.

Para evitar que se puedan subir archivos con el mismo nombre, al subir una imagen el sistema genera un código aleatorio y guarda la imagen en una carpeta con el Id del animal y el código generado + el nombre del archivo.

```
String nombreArchivo = UUID.randomUUID().toString() + "_"  
+ archivo.getOriginalFilename().replace(target:" ", replacement:"");
```

Imagen: extracto código generar código aleatorio nombre foto

Cuando una foto se elimina de un animal, se elimina también la imagen del disco del servidor y la entrada a la base de datos.

```
if (multimedia != null) {  
    String nombreArchivo = multimedia.getEnlace();  
    Path rutaArchivo = Paths.get("../frontend//src//" + nombreArchivo);  
    if (Files.exists(rutaArchivo)) {  
        Files.delete(rutaArchivo);  
    }  
}
```

Imagen: extracto código borrar foto

7.8. Funcionalidad Envío de Email.

Para el envío de email desde la aplicación se ha creado un email de gmail, happypawsunir@gmail.com Para que esta funcionalidad esté operativa se realizan varias configuraciones:

- Utiliza JavaMailSender para el envío de email a través de la dirección de gmail.
- El controlador HomeController se encarga de enviar el email del formulario general de la web.

El envío del email de contacto usa dos métodos, el que maneja el formulario recibido y lo manda al método sendEmail.

```
@PostMapping("/contacto")  
public ResponseEntity<String> manejoEnvioformulario(@RequestBody ContactForm form) {  
    // Aquí podemos añadir la validación del back del formulario.  
    try {  
        sendEmail(form);  
        return new ResponseEntity<>(body:"Tu email ha sido enviado correctamente!", HttpStatus.OK);  
    } catch (MailException e) {  
        return new ResponseEntity<>(body:"Error al enviar el mensaje", HttpStatus.INTERNAL_SERVER_ERROR);  
    }  
}
```

Imagen: extracto código manejo del formulario de contacto

y el método sendEmail que envía el email a la dirección web definida:

```
//Método para enviar el formulario al email de HappyPaws
private void sendEmail(ContactForm form) {
    SimpleMailMessage message = new SimpleMailMessage();
    message.setTo("happypawsunir@gmail.com");
    message.setSubject("Nuevo mensaje de contacto de " + form.getName());
    message.setText("Correo electrónico: " + form.getEmail() + "\n\n" + form.getMessage());
    emailSender.send(message);
}
```

Imagen: extracto código envío email formulario general

- El controlador ProtectoraController el que se encarga del envío del formulario de contacto a la protectora que se quiere contactar.

En el formulario de contacto de una protectora se busca la Protectora por su id y se asigna el email de la protectora como destinatario.

```
//Método para enviar el formulario al email de HappyPaws
private void sendEmail(ContactForm form, int idProtectora) {
    Protectora protectora = protdao.buscarProtectoraId(idProtectora);
    SimpleMailMessage message = new SimpleMailMessage();
    message.setTo(protectora.getEmail());
    message.setSubject("Mensaje " + form.getName());
    message.setText("Correo electrónico: " + form.getEmail() + "\n\n" + form.getMessage());
    emailSender.send(message);
}
```

Imagen: extracto código envío email contacto protectora

- En Application Properties está la configuración de conexión a la cuenta. En vez de usar la contraseña del email se ha usado una contraseña de aplicación definida en la cuenta de gmail en verificación de dos pasos.

```
# para el envio de correos
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=happypawsunir@gmail.com
spring.mail.password=kwxklo vxer iuvy
spring.mail.properties.mail.debug=true
spring.mail.properties.mail.transport.protocol=smtp
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true

spring.servlet.multipart.max-file-size=10MB
spring.servlet.multipart.max-request-size=10MB
```

Imagen: extracto código configuracion envio email

En Pom.xml se añade la dependencia:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-mail</artifactId>
</dependency>
```

Imagen: extracto código inserción dependencia envío mail.

Funcionalidad Envío de Email - Contacto web	
<p>Contacta con HappyPaws</p> <p>Nombre: El nombre es requerido.</p> <p>Correo electrónico: Introduce tu correo electrónico.</p> <p>Mensaje:</p> <p>¿Cómo podemos ayudarte?</p> <p><input type="button" value="Enviar"/></p>	<p>Contacta con HappyPaws</p> <p>Nombre: prueba 07/12/23 09:38</p> <p>Correo electrónico: esunapruебadehappyform@gmail.com</p> <p>Mensaje: Es un mensaje de prueba desde la página. prueba 07/12/23 09:38</p> <p><input type="button" value="Enviar"/></p>
Img: formulario contacto web	Img: formulario contacto relleno web
<ul style="list-style-type: none"> • El formulario no se puede enviar si no se llenan todos los campos y el botón enviar está desactivado hasta que se rellena. • Si no se pone un correo electrónico válido, el botón tampoco se activa. • Cuando se pulsa en enviar un texto junto al botón nos indica que se está enviando. <p style="text-align: center;"><input type="button" value="Enviar"/> Enviando correo electrónico...</p> <p>Imagen: estado enviando email</p> <ul style="list-style-type: none"> • Una vez enviado, una ventana modal nos informa si se ha enviado correctamente. <p style="text-align: center;"> Enviado El formulario se ha enviado correctamente <input type="button" value="OK"/></p> <p>Imagen: confirmación envío formulario</p>	

Funcionalidad Envío de Email - Contacto con una protectora

Contacta con Propatas

Nombre:
 El nombre es requerido.

Correo electrónico:
 Introduce tu correo electrónico.

Mensaje:

¿Qué quieres preguntar a la protectora Propatas?

Imagen: extracto código borrar foto

- En este caso el email se envía al email que tiene la protectora como contacto.
- El formulario no se puede enviar si no se rellenan todos los campos y el botón enviar está desactivado hasta que se rellena.
- Si no se pone un correo electrónico válido, el botón tampoco se activa.
- Cuando se pulsa en enviar un texto junto al botón nos indica que se está enviando.

Enviando correo electrónico...

Imagen: estado enviando email

- Una vez enviado, una ventana modal nos informa si se ha enviado correctamente.


Enviado

El formulario se ha enviado correctamente

Imagen: confirmación envío formulario

8. Conclusiones y mejoras del proyecto

8.1. Mejoras

- Enviar un email al usuario, notificando el cambio de estado de su solicitud de adopción.
- El listado de protectoras y animales se pagine para mostrar de 10 en 10.
- Un usuario no puede solicitar la adopción si no ha llenado antes el cuestionario de adopción.
- Cuando una protectora se da de baja, todos los animales se deben poner en estado disabled.
- Cuando un adoptante pulsa en solicitar adopción debe solicitar confirmación.
- Permitir a un usuario subir animales en adopción.
- Cuando una protectora se da de alta, se pone en estado pendiente para comprobar documentación y desde el panel de admin poder activarla.
- Cambiar donde se almacenan las fotos para que lo haga en el backend.
- Arreglar algunos detalles de CSS
- Añadir aviso de cookies.
- Añadir Google Search Tools
- Filtrar por estado adopcion, estado animal y estado protectoras en paneles de administración de la protectora y general.
- Realizar despliegue de la aplicación.

8.2. Conclusiones

Concluyendo nuestro proyecto de implementación de la página web, queremos resaltar las algunas lecciones aprendidas y los logros alcanzados durante este proceso. Hemos aprendido sobre nuevas tecnologías, como JWT, y hemos descubierto nuevas herramientas de gestión de proyectos, como por ejemplo JIRA. Así mismo hemos ampliado conocimientos de tecnologías como Angular y Bootstrap.



Además, esta experiencia nos ha dado la oportunidad de ampliar nuestra experiencia en el desarrollo de aplicaciones web. A pesar de los desafíos, hemos ampliado nuestras habilidades de trabajo en equipo y hemos demostrado una notable capacidad de adaptación.

Es destacable mencionar que, a pesar de la limitación de tiempo debido a compromisos laborales y prácticas, hemos logrado llevar a cabo un proyecto que nos ha permitido crecer profesionalmente. La investigación constante, tanto dentro como fuera del marco del curso, nos ha permitido realizar un proyecto funcional.

Este proyecto ha sido una experiencia enriquecedora que va más allá de la simple implementación técnica. Hemos fortalecido la cohesión del equipo, adquirido nuevas habilidades y sentado las bases para futuros proyectos. Estamos orgullosos del trabajo realizado y confiamos en que esta experiencia contribuirá significativamente a nuestro crecimiento profesional.



9. Agradecimientos

Queremos agradecer a todos nuestros familiares y amigos su apoyo durante todo este proceso ya que sin ellos no lo habríamos podido realizar..

Gracias por vuestra paciencia, comprensión, apoyo y ánimo que nos habéis transmitido.

10. Bibliografía

- <https://angular.io/>
- <https://code.visualstudio.com/>
- <https://github.com/>
- <https://git-scm.com/>
- <https://lenguajecss.com/css/>
- <https://looka.com/>
- <https://material.angular.io/>
- <https://pixabay.com/>
- <https://spring.io/projects/spring-boot>
- <https://www.adobe.com/es/>
- <https://www.atlassian.com/es/software/jira>
- <https://www.bing.com/images/create>
- <https://www.figma.com/>
- <https://www.flaticon.es/>
- <https://www.java.com/es/>
- <https://www.mysql.com/products/workbench/>
- <https://www.typescriptlang.org/>
- https://www.w3schools.com/html/html_intro.asp
- <https://www.youtube.com/watch?v=YUqi1IjLX8I>
- <https://xn--designthinkingespaa-d4b.com/#:~:text=Qu%C3%A9%20es%20el%20Design%20Thinking,que%20empezar%20una%20nueva%20ite,raci%C3%B3n.>
- [INEbase / Demografía y población /Padrón /Relación de municipios y sus códigos por provincias / Últimos datos](#)
- [Introduction · Bootstrap v5.0 \(getbootstrap.com\)](#)
- [Spring Boot - Sending Email via SMTP - GeeksforGeeks](#)
- <https://www.tango.us/> (creación de manuales)
- <https://www.lucidchart.com/pages/> (para hacer diagramas)
- [Volcado Mysql de municipios y provincias españolas y territorios UE · Harecoded](#)

11. Anexos

11.1. Scripts BBDD

- [Script creación Base de datos](#)
- [Script insert datos base.](#)
- [Diagrama ER BBDD](#)

11.2. Imágenes

- [Imagen 1: Proyecto ER](#)
- [Imagen-02: Diagrama final ER](#)
- [Imagen-03: Casos de uso](#)

11.3. Código fuente aplicación GitHub

- [Repositorio GitHub](#)

11.4. Diseño de la web - Figma

- [Figma - Design Thinking](#)
- [Figma - Design Thinking Prototipar II](#)

11.5. Manuales de uso

- [Vídeo uso completo de la aplicación.](#)
- [¿Cómo crear una protectora?](#)
- [¿Cómo crear una solicitud de adopción?](#)
- [Panel de Administración de una Protectora.](#)
- [¿Cómo dar de alta, modificar y borrar un animal?](#)
- [Panel de gestión Administrador](#)