Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TRABAJO DE FINAL DE GRADO

Título: Protecat: digitalización de una protectora de gatos

Titulación: Grado en Ingeniería Telemática

Autor: Àlex Moya I Sánchez

Director: Roc Meseguer Pallarès

Fecha: 31 de Enero de 2023

Título: Protecat: digitalización de una protectora de gatos

Autor: Àlex Moya I Sánchez

Director: Roc Meseguer Pallarès

Fecha: 31 de Enero de 2023

Resumen

El número de abandonos de gatos en España se encuentra entre los más altos a nivel europeo, y aunque se está percibiendo un leve descenso, más de 118.000 felinos son abandonados cada año en nuestro país. Para combatir este problema, contamos con diversas asociaciones que buscan hogar para estos animales, y cuidan de aquellos que no corren tanta suerte. Entre estas protectoras se encuentra “Amics i protectors dels gats”, una asociación sin ánimo de lucro que se centra en alimentar y proteger a los gatos callejeros de la ciudad de Vilanova i la Geltrú.

La falta de recursos y tiempo hace que su trabajo sea muy laborioso y costoso, por lo que el objetivo principal de este proyecto es la elaboración de una aplicación para facilitar la gestión administrativa de la asociación, así como la comunicación y distribución de las tareas de la protectora. Esta aplicación permitirá tener un registro de los gatos que aparecen en cada colonia, la medicación que necesitan y otras observaciones. También mantendrá conectadas a las voluntarias, a través de un chat, e implementará mejoras organizativas (lista de tareas, calendarios, inventarios, etc.)

Por otro lado, será necesaria la creación de otra aplicación para resolver los problemas de los usuarios externos a la asociación, consiguiendo así una mejor comunicación y facilitando los procesos administrativos, basada en un método llamativo para la adopción de los felinos, un muro con diversas noticias de la protectora y la facilitación de otorgar ayudas.

Las tecnologías principales que se utilizarán para llevar a cabo el siguiente proyecto serán: Flutter (para la creación de la parte visual de las dos principales aplicaciones), TypeScript, Express y Node (para la creación de la lógica de las aplicaciones) y Angular (para la web administrativa).

Finalmente los resultados obtenidos serán analizados, mirando qué objetivos iniciales se han cumplido, el rendimiento de la aplicación, la utilidad y las posibles futuras mejoras que se podrían llevar a cabo.

Por lo que este proyecto se puede resumir en el desarrollo de un conjunto de aplicaciones móviles para una protectora de gatos, así como de un análisis de las tecnologías y técnicas utilizadas, y un estudio de los objetivos cumplidos y los resultados obtenidos.

Title: Protecat: digitalisation of a cat shelter

Author: Àlex Moya I Sánchez

Director: Roc Meseguer Pallarès

Date: January 31 st 2023

Overview

The number of cat abandonment in Spain is one of the highest in Europe, although every year this situation is less critical, more than 118.000 cats are abandoned every year in our country. To reverse these circumstances, we can find some associations who search a home for these animals, and take care of the ones who remain in the street. One of these animal protector is “Amics i protectors dels gats”, a non-profit association who feed and protect the stray cats of Vilanova i la Geltrú city.

The lack of time and resources makes this labor harder and costly, so the main objective of this project is to make an application to help with administrative management of the association, the communication and the distribution of tasks. The app will register the cats of the colonies, the medicines that they need and other parameters. Also, will connect the volunteers with a chat. Furthermore, will improve the organization with other functions (such as list of tasks, calendar, inventory, etc.).

On the other hand, it will be necessary to develop another app to solve the problems with the external users of the association, improving the communication and the administrative management. The main part of this application will be the development of a flashy method to adopt cats. Other of the functions will be a feed with news and an easy way to contribute.

The main technologies that will be used in this project will be: Flutter (for the visual part of both apps); TypeScript, Express and Node (for the logical part of the software); and Angular (for the administrative web).

Finally the obtained results will be analysed, checking if the main objectives of the project are fulfilled, the performance of the software, the utility and the possible improvements that will be deployed in the future.

So, this project could be summarised in the development of a set of apps for a cat protection organisation, as well as an analysis of the chosen techniques and technologies, and a study of the fulfilled objectives and the obtained results.

Agradecimientos

Quiero agradecer al tutor del proyecto, Roc, por la ayuda ofrecida y las facilidades recibidas, así como la libertad a la hora de trabajar y el asesoramiento durante todo el proceso.

A la protectora “Amics Amics i protectors dels gats” y a todas las voluntarias que forman parte de ella, por la increíble labor que hacen y el valor que otorga a nuestra sociedad, sacrificando parte de su vida para salvar la vida de esos pequeños animales. En especial a Ka, quien ha estado siempre dispuesta a ayudar y a contribuir al proyecto.

A mi familia por apoyarme y sufrir conmigo todo este camino, en especial a mi madre, quien ha estado siempre ahí confiando en mí.

Y por último a mi pareja, quien me ha ayudado y apoyado en toda esta etapa, y que sin su compañía no habría sido lo mismo.

Gracias.

ÍNDICE

ÍNDICE TABLAS Y FIGURAS

INTRODUCCIÓN

Las tasas de abandono animal en España son de las más altas a nivel europeo, tal y como señala el estudio “Èl nunca lo haría 2022”[1], en el año 2021 se registraron más de 285.000 abandonos animales, de los cuales cerca del 41% son gatos. Aunque este registro es cada vez menor, los datos son alarmantes y desoladores. Para intentar frenar este horrible problema, múltiples asociaciones cuidan a los animales en refugios o, como en la mayoría de los casos de gatos, en colonias controladas, donde se les alimenta, médica y controla veterinariamente. Una de estas asociaciones es “Amics i protectors dels gats”[2]. Este proyecto viene motivado por las ganas de ayudar a esta asociación, de la cual formé parte y, de alguna manera, agradecer la labor que hacen y facilitar su trabajo en la medida de lo posible, ya que todas las acciones realizadas en esta asociación se hacen durante el tiempo libre de las personas que la componen.

Esta asociación sin ánimo de lucro cuenta con diversas carencias tecnológicas debido a los pocos recursos con los que cuenta. La mayoría de estas carencias están relacionadas con la falta de informatización y automatización de los procesos. Por lo que uno de los objetivos principales del proyecto es la solventación de estas carencias. Se espera cubrir estas necesidades mediante la realización de dos aplicaciones, tanto móviles como web, una centrada en el uso para las voluntarias que forman parte de la asociación, y otra para las que no. La primera aplicación se centrará en solucionar los grandes problemas de gestión, intentando unificar, automatizar y registrar todos los procesos y tareas. Para la otra aplicación se creará una red social para acercar a los posibles adoptantes y llamar la atención de estos, agilizando también los trámites necesarios para llevar a cabo la adopción. Estas aplicaciones estarán basadas en el *backend*, la inteligencia de las aplicaciones, la cual conecta la parte visible (las aplicaciones) con la base de datos y apis, otorgando inteligencia para poder llevar a cabo los procesos.

Los objetivos a cumplir en este proyecto son:

- Solventar todas, o la mayoría de las carencias tecnológicas presentadas por la asociación.
- Seleccionar las tecnologías utilizadas en el proyecto.
- Adquirir los conocimientos necesarios para el desarrollo de aplicaciones.
- Utilizar metodologías ágiles e incrementales para la gestión de proyectos.

La estructura del trabajo es algo compleja, ya que es mucho más amplia que la creación de las diferentes aplicaciones, contando con una extensa documentación en la que se detallan aspectos como la división del proyecto y los aspectos técnicos.

A la hora de iniciar un proyecto es muy importante dividirlo en etapas, para poder ejecutarlo a la perfección y tener un control total sobre él, pudiendo cerrar etapas a medida que sean completadas, y así, dosificar la carga de trabajo. Los cinco grupos de procesos en la gestión de un proyecto están definidos[3][4] como: inicio (capítulo 1 y 2) , planificación (capítulo 3), ejecución (capítulos 4, 5, 6 y 7), seguimiento y control (capítulo 8), y cierre (capítulo 9). Primero, contará con la definición de los aspectos concretos del proyecto (tecnologías a utilizar o las funcionalidades de las aplicaciones). Para esta parte del proyecto será necesaria una gran comunicación con el cliente final, ya que es donde se definirán las funcionalidades y se tomarán decisiones que marcarán el camino de la planificación; así como un contexto social para entender el proyecto. Tras este paso, se definirá la organización del proyecto, definiendo el plan de trabajo y estableciendo un calendario, que servirá para cumplir los plazos establecidos, así como poder analizar posteriormente el esfuerzo dedicado y poder mejorar la estructuración en futuros proyectos. Una vez todos estos puntos estén concretados, se procederá al desarrollo de las aplicaciones. Debido a la dependencia entre las aplicaciones, se realizarán de forma simultánea. Previsiblemente, esta será la parte más laboriosa del proyecto, ya que hay una gran cantidad de trabajo, pero a la vez será la que obtendrá unos resultados más clarificantes y tangibles sobre el proyecto. Por último, se llevará a cabo una fase de análisis del producto para determinar si se adapta a los objetivos marcados. Los datos que permitirán determinar la efectividad de las aplicaciones se recogerán a través de las opiniones ofrecidas por el cliente final, por lo que, deberán de ser analizadas para informar las conclusiones.

Resumen:

Por lo que las principales fases de este proyecto se podrían definir como:

- Estudio de las carencias de la asociación y definición de una guía de trabajo.
- Análisis de las posibles tecnologías para la realización de las aplicaciones.
- Desarrollo de aplicaciones móviles con el fin de cubrir aquellas carencias que pueda mostrar el cliente (en este caso la protectora).
- Uso de metodologías ágiles e incrementables en la gestión del proyecto.
- Obtención de resultados que posteriormente se deberán analizar.

CAPÍTULO 1: CONTEXTO

Para poder entender el porqué del proyecto, así como las funcionalidades que se llevarán a cabo, hay que conocer en profundidad las acciones que lleva a cabo la asociación a la que se pretende ayudar, así como la motivación del proyecto y la situación social en la que se encuentran los animales.

1.1. Contexto social

Tal y como indicaba el estudio “Èl nunca lo haría 2022”[1], la situación de abandono animal en España es crítica, situando a nuestro país entre los más altos de Europa. A pesar de que en los últimos dos años se ha experimentado una leve bajada, en 2021 más de 285.000 perros y gatos fueron abandonados, de los cuales 118.000 son gatos, un 2.6% del total de gatos españoles. El estudio indica que el 51% de los gatos recogidos son adoptados, aunque tal y como indican algunas protectoras, en los últimos meses el número de adopciones está bajando considerablemente, haciendo que cada vez haya más animales en los refugios o en las calles. El perfil de estos animales está compuesto en su mayoría por cachorros, debido a la facilidad de procreación de los gatos, siendo las camadas no deseadas la principal fuente de abandono. Tal y como indica el estudio, en España hay 1.591 entidades protectoras de animales censadas, de las cuales 1.257 ayudan a buscar una familia a los felinos y/o les dan acogida en los propios refugios o en casas de acogida temporal. Más de la mitad de estas asociaciones gestionan colonias controladas de gatos. Estos suelen vivir en grupos, comúnmente conocidos como colonias, divididos por toda la ciudad, formadas generalmente por gatos abandonados (mayormente cachorros), que presentan una actitud asustadiza debido al trauma al que han sido sometidos, dificultando así el trabajo de esterilización de las asociaciones, y empeorando así su situación en referencia a su calidad de vida y los riesgos que ello conlleva.

De entre estas asociaciones encontramos “Amics i protectors dels gats”[2]. Esta asociación sin ánimo de lucro inició su actividad en el año 2005 en la ciudad de Vilanova i la Geltrú, provincia de Barcelona, con el objetivo de velar por el bienestar de los gatos callejeros. Esta asociación está respaldada por muchas voluntarias que sacrifican gran parte de su tiempo libre para llevar a cabo todas las actividades necesarias para el correcto funcionamiento de la protectora. Estas acciones son muy diversas, pasando desde las más rutinarias (como la alimentación de los gatos) hasta las más excepcionales (como el transporte urgente de un animal al veterinario), algunos ejemplos de estas son:

- **Cuidado diario de las colonias:** como se ha comentado anteriormente, las colonias son grupos de gatos abandonados en la calle, y que por diversas circunstancias (como el parentesco entre diferentes animales o la obtención de comida en esa zona) se acaban agrupando en pequeñas comunidades. Las voluntarias cada día pasan por los diferentes puntos de la ciudad en los que están distribuidos y se encargan de alimentarlos (con comida húmeda para ese mismo momento, y con pienso para el

resto del día), medicarlos y ofrecerles un mínimo de caricias. Esta tarea es de las más agradecidas y sacrificadas, ya que los animales ofrecen mucho cariño, pero el tiempo que se les puede dedicar es mínimo. Además, se han de contemplar varios aspectos como la ausencia de algún integrante de la colonia o el deterioro de salud que pueda presentar alguno de ellos.

- **Gestión de la asociación:** actualmente alimentar a los gatos callejeros está regulado por las ordenanzas municipales. En el caso de la ciudad en la que actúa la asociación, está prohibido alimentar a los animales a no ser que se esté autorizado por el ayuntamiento (ordenanza municipal 347a, artículo 23 segundo parágrafo, de Vilanova i la Geltrú[6]). Por lo que una de las gestiones a realizar será la de solicitar este tipo de permisos, que cada uno de los voluntarios debe llevar consigo a la hora de realizar dicha tarea. Otra de las tareas sería la tramitación de los formularios de adopción, unión a la asociación, solicitud de permisos al ayuntamiento, etc.

- **Llevar a los animales al veterinario:** los gatos no solo deben de ir al veterinario asiduamente para controlar su estado de salud o por una urgencia, sino que, cuando se lleva a cabo una acogida, hay que realizar una serie de requerimientos recogidos en el artículo 56 de la ordenanza municipal 347a del ayuntamiento de Vilanova i la Geltrú[6]. Así como la esterilización a la cual la asociación está obligada a realizar (tal y como indica el punto 2 del artículo 58 de la ordenanza municipal 347a del ayuntamiento de Vilanova i la Geltrú[6]). Todos estos desplazamientos los ha de realizar la voluntaria con sus propios medios, incluyendo la gestión del estado que pueda presentar el animal en el momento de desplazarlo (puede llegar a sentir miedo al separarse del grupo y de su entorno habitual).

- **Promoción de la adopción y estilo de vida libre de crueldad animal:** para poder llevar a cabo tareas como las que se han mencionado anteriormente, se necesita la ayuda de las personas voluntarias, que constituyen la base de la asociación. Para esto, realizan diferentes puestos informativos en los que anuncian todas las tareas que realizan, y recaptan alimentos, medicación, donaciones y voluntarios. Además, comunican las adopciones que tienen disponibles o buscan casas de acogida para casos en los que la salud del animal está en peligro si este se encuentra en un entorno hostil como puede ser la propia exposición a la calle. Además, promueven un estilo de vida vegano y sin crueldad animal, ayudando e informando a todo aquél que tenga cualquier tipo de duda. Esta tarea se extiende por las redes sociales y la página web.

Estos son algunos ejemplos de la cantidad de tareas que realizan, que como se puede ver, es una gran carga de trabajo que las voluntarias afrontan de manera altruista.

Yo tuve el placer de formar parte de esta asociación durante un tiempo. Durante la estancia en ella pude entender el sacrificio que realizan y la gran

importancia de su existencia. Me hicieron ver las duras condiciones en las que viven los animales callejeros, y la gran cantidad de abandonos que ocurren a diario. También pude disfrutar de su compañía y ver como algunos arrastraban traumas que generaban miedo al establecer contacto con las personas por temor a ser rechazados o agredidos. Fue una época de aprendizaje muy bonita y que por incompatibilidades horarias con la universidad, y problemas personales, tuve que abandonar, por lo que se me ocurrió agradecerles la acogida que me hicieron, así como los valores que me transmitieron a través de este trabajo.

Este proyecto está enfocado al apoyo a una asociación sin ánimo de lucro, por lo que se define como un proyecto de acción social[5], el cual está relacionado con los derechos humanos básicos o con la ayuda a grupos vulnerables. Este tipo de proyectos no cuentan con un beneficio económico, ya que se realiza de manera altruista u obteniendo otros beneficios, adaptándose perfectamente a este plan, ya que el autor recibe un gran aprendizaje, mientras que la protectora recibe unos servicios con un gran coste y que de otra manera no podrían conseguir.

La utilización de los dispositivos móviles es necesaria para cualquier actividad que queramos realizar, y de esta manera la sociedad actual ha hecho de estos dispositivos unos esenciales en su vida cotidiana, por lo que el trabajo final será útil para una gran parte de la población. El número de aplicaciones dirigidas a la ayuda al bienestar de los animales es escaso, y si nos centramos en el campo de la gestión de las protectoras o al fomento de las adopciones el número es ínfimo. Además, este proyecto tiene como objetivo el de amoldarse estrictamente a las necesidades mencionadas anteriormente, haciéndolo así único en este ámbito. Aunque también se podría utilizar como base para otras asociaciones (tal y como se hace en las empresas del sector).

1.2. Definición de carencias de la protectora

Una de las primeras tareas que se deben realizar a la hora de iniciar un proyecto consiste en la elaboración de unos objetivos claros y concisos que marquen el rumbo de todos los procesos que se lleven a cabo. Esto hará que se parta desde una base sólida, para poder llevar a cabo una toma de decisiones de forma rigurosa, que permita que la idea tome forma adecuadamente, y se pueda profundizar en aspectos más concretos del trabajo, llegando así a un resultado final sin grandes complicaciones.

Los objetivos principalmente vendrán definidos por las necesidades del cliente, pero también se pueden identificar desde una visión externa las necesidades que requiere y hacer una primera propuesta al cliente, para que este pueda entender cuales son las opciones que pueden obtener. En ocasiones, la visión de una persona externa puede aportar ideas que en un primer momento el cliente no tiene en cuenta, pero que le pueden ser de gran utilidad. En este aspecto, se realizó un primer análisis de posibles necesidades y se redactó una propuesta de proyecto.

Las necesidades requeridas por la asociación fueron las siguientes:

- Mejorar y agilizar la experiencia del usuario a la hora de realizar trámites tales como la adopción u obtener información sobre la protectora y sus actividades.

- Fomentar la adopción.

- Concienciar del coste de los cuidados de los animales.

- Reducir la carga de trabajo derivada de algunos trámites administrativos como la adopción.

- Tener un método de registro y control de las colonias.

- Unificación de ciertas acciones de organización (calendario de actividades, tareas asignadas)

- Tener una mejor relación entre el usuario y la asociación.

- Controlar los recursos de la asociación.

Como se puede observar, las propuestas están relacionadas con objetivos concretos que buscan la unificación y la mejora en la experiencia del usuario, para así fomentar la adopción.

Tras el análisis de las necesidades que presenta el cliente, se le propone la creación de dos aplicaciones móviles y web; una de ellas dirigida a los usuarios y otra a los miembros de la protectora. Esta división se desarrollará con mayor detalle en futuras etapas del proyecto, pero esencialmente se origina por el tratamiento de los datos, entre otros aspectos.

Una vez definidos los objetivos principales, es debido precisar precisar aquellas necesidades que se podrán cubrir. Para llevar a cabo esta tarea, es necesaria una estrecha colaboración con la asociación, ya que serán las integrantes de dicho grupo las que deberán de confirmar si las funcionalidades propuestas se adaptan o no a su forma de trabajar y a las necesidades descritas anteriormente.

Las funcionalidades propuestas se dividieron en dos partes, cada una de ellas enfocada en el usuario que utilizará mayoritariamente la aplicación.

Las ideas para solventar los problemas propuestos en relación al usuario perteneciente a la protectora son las siguientes:

- **Registro y control de las colonias:** mapa que mediante la utilización de la ubicación de nuestro dispositivo nos indicará cada una de las colonias, pudiendo acceder a la información asociada a cada una de ellas, como los animales que se encuentran o la medicación prescrita

para cada uno de ellos. Además de contar con un registro en el que la persona encargada pueda ingresar las tareas realizadas u otras observaciones (como el comportamiento extraño de un gato, si algún gato no ha aparecido, etc.).

- **Reducir carga de trabajo administrativo:** gestión de las solicitudes de adopción. En este apartado de la aplicación la administradora podrá controlar las solicitudes de adopción recibidas por cada animal y gestionarlas directamente desde la aplicación.

- **Controlar los recursos de la asociación:** inventario de los recursos de la protectora. En esta sección se contará con un recuento de los productos obtenidos por la asociación.

- **Unificación tareas de organización:** para mejorar la organización de la asociación, se creará un calendario en el que se podrá anotar la distribución de los diferentes turnos de las colonias. Asimismo, se podrán crear tareas que se podrán asignar a cada una de las colaboradoras, pudiendo visualizarlas en forma de lista o en el propio calendario.

La posibilidad de contar con un registro de las colonias en las que trabajan, pudiendo hacer anotaciones, genera un gran beneficio y facilita considerablemente el trabajo a realizar. Por otro lado, el uso de esta aplicación ofrece una centralización de todos estos servicios, haciendo las gestiones mucho más sencillas y obteniendo una experiencia como usuario mucho más cómoda.

En cuanto a las ideas propuestas para complementar las carencias de las usuarias externas de la protectora contamos con:

- **Fomentar la adopción y reducir la carga de trabajo administrativo:** crear un método llamativo para la adopción de los gatos. El objetivo es que el usuario sea tentado a adoptar. El método a utilizar todavía no está determinado, ya que se generarán diferentes modelos y se realizará una pequeña muestra para poder analizar cuál es el método más eficiente. Las propuestas iniciales son:

    - Generar un muro en el que cada animal en adopción cuente con una tarjeta que contenga una foto y una pequeña descripción suya. De esta manera el posible adoptante podrá ver a cada uno de los gatos y en caso de que uno le llame la atención podrá visitar su perfil y ver si cumple los requisitos para adoptarlo, así como tener una visión global de los animales disponibles para adoptar.

    - Otra de las opciones que se contemplan, que se encuentra estrechamente relacionada con la anterior, se basa que cada animal cuente con un perfil (formado por una descripción, algunas características, gustos y curiosidades) que cada usuario podrá ver

de forma individual para así poder decidir si quieren iniciar el proceso de adopción o deciden descartarlo.

- Por último contamos con una opción basada en un método que se aleja a los anteriores. En este caso, el usuario deberá contestar una serie de preguntas, y según los resultados obtenidos, la aplicación hará una selección de los gatos que según sus características se adapten más a los gustos y requerimientos que presente el adoptante. Una vez realizada la selección, el usuario podrá decidir con cuál de ellos continuará en su proceso de adopción.

Es importante contemplar que para todas y cada una de estas opciones, cuando el usuario proceda al proceso de adopción, se le otorgará un formulario con una serie de preguntas básicas para considerar si el perfil se adapta a las necesidades del animal. En caso de que el resultado sea favorable, la candidatura del posible adoptante pasará a la administradora de la asociación siguiendo con el proceso de selección. En caso negativo la candidatura será rechazada y notificada al usuario. Esto genera una reducción drástica de las gestiones realizadas por la protectora.

- **Mejorar la comunicación asociación - usuario:** otra de las funciones propuestas para esta aplicación es la creación de un muro principal en el que colgar noticias. Entre ellas, se encontrarán la demanda de casas de acogida o donaciones para casos extraordinarios (gastos imprevistos, etc.). Estos casos también contarán con una sección independiente para poder consultar los casos abiertos de una manera mucho más rápida y eficiente. En el caso de las donaciones repentinas, se generará una barra con el objetivo del dinero necesario, y una progresión a medida que se vayan recibiendo donaciones. Las siguientes funciones se centran más en la obtención de la información, por lo que cada una de ellas contaría con su propia sección. La primera de ellas sería la creación de un formulario para solicitar la entrada en la protectora. Otra de ellas consiste en la creación de un chat para poder comunicarse directamente con la protectora, para así poder consultar cualquier duda que pueda surgir. La última sección contará con información de la propia asociación, para poder acercarse así a usuarios externos a esta.

- **Concienciar del coste del cuidado de los animales:** consiste en un método visual para realizar donaciones. Esta funcionalidad se basa en la integración de pequeños artículos, con sus respectivos precios, para que el usuario pueda ir añadiendo los artículos, y finalmente, realizar una donación del valor de la mercancía seleccionada. De esta manera, conseguimos que el usuario sea consciente del uso que se da a su donación, y el coste real de los artículos. Un pequeño ejemplo para clarificar la funcionalidad sería el siguiente: poner la imagen de una lata de comida con el precio (0.49€), debajo de este el usuario puede añadir la cantidad de productos que quiere donar (5), y finalmente haría una donación con el valor correspondiente (5 x 0.49€ = 2.45€). Aunque las

donaciones no se puedan realizar mediante la aplicación (debido a su complejidad), se le ofrecerá las facilidades para poder realizar la donación (usuario de paypal, número de cuenta o número de bizum).

1.3. Definición de los objetivos

Los objetivos de este proyecto son los siguientes:

- Solventar todas, o la mayoría de, las carencias tecnológicas presentadas por la asociación, dando prioridad a las siguientes necesidades:
    - Reducir la carga administrativa de las voluntarias.
    - Desarrollar un método para gestionar y controlar las colonias de una forma más eficaz.
    - Mejorar la experiencia del usuario.
- Realizar un estudio para seleccionar la tecnológica que mejor se ajuste a las necesidades de *frontend* y *backoffice*.
- Adquirir los conocimientos relacionados con el proceso completo de desarrollo de una aplicación móvil, tal y como se llevaría a cabo en un entorno laboral.
- Utilizar metodologías ágiles e incrementables para la gestión de proyectos.

1.4. Restricciones

Una de las grandes dificultades que se pueden presentar en el desarrollo del proyecto consiste en la falta de visión para detectar las limitaciones con las que se cuentan en base a las funcionalidades que se pretenden llevar a cabo.

A la hora de definir las restricciones de un proyecto[7] se pueden definir 2 grandes bloques, los cuales dependen de 3 factores que se encuentran interrelacionados entre sí.

El primer bloque es conocido como el equipo de la triple restricción (o el triángulo de la gestión de proyectos), cuya relación está compuesta por el tiempo, el alcance y el coste. En este caso se deben equilibrar estos tres elementos para garantizar el éxito del proyecto.

El concepto del tiempo está claro, se refiere al tiempo marcado para la realización del proyecto. Con el alcance se refiere al volumen de trabajo del proyecto, es decir, la cantidad de detalles y funcionalidades con las que contarán las aplicaciones. Finalmente contamos con la última restricción, el

coste, que refleja el presupuesto del proyecto; en este caso esta última restricción no juega un gran papel, ya que no se cuenta con un presupuesto, pero está estrechamente relacionado con el tiempo, que en este tipo de proyectos ocupa la mayor parte del presupuesto (siendo este el sueldo del programador).

- El escaso tiempo para realizar el proyecto es una de las mayores limitaciones, ya que de ello dependerá el número de funcionalidades que se podrán llevar a cabo, así como de la complejidad asociada a cada una de estas. Para poder realizar este ejercicio hay que tener en cuenta muchas variables, ya que dependiendo de las tecnologías utilizadas, el número de aplicaciones desarrolladas, el grado de detalle de cada una, etc., puede variar en gran medida el tiempo empleado.

- Se prevé un gran alcance del proyecto, debido a que puede acoger un gran número de funcionalidades, que se pueden desarrollar de forma muy detallada, al igual que se puede trabajar en el diseño para hacerla más atractiva para el público.

Por lo que en este primer bloque debemos de enfrentar el tiempo empleado con el alcance del proyecto, es decir, contra mayor sea el alcance que le queramos otorgar, mayor será el tiempo que deberemos dedicar. Teniendo en cuenta el reducido tiempo con el que se cuenta, el alcance no podrá ser tan grande como lo deseado, pero el objetivo es conseguir el mejor resultado posible con los recursos de los que se dispone. Con tal de optimizar los resultados, se trabajará en la realización de una programación detallada (que se encontrará más adelante), para así poder aumentar al máximo el alcance.

Por otro lado nos encontramos con el segundo bloque, que depende de los riesgos, los recursos y la calidad.

- Se puede entender como riesgo todo aquel imprevisto que surge una vez el proyecto se ha iniciado. Suelen asociarse a un hecho que perjudica al transcurso del trabajo, pero a veces los riesgos pueden representar un beneficio, como la colaboración de un nuevo compañero, que puede perjudicarnos o beneficiarnos.

- Otra de las dificultades son los recursos. Se trata de un proyecto en un ámbito académico, y aunque la institución facilite grandes herramientas, estas son limitadas, por lo que en algunos casos el proyecto puede estar afectado por esta falta de recursos.

- Por último nos encontramos con la calidad, este parámetro representará el resultado final. Pero este parámetro no solo está relacionado con los vistos anteriormente, sino que hay muchos factores que entran en juego a la hora de determinar la calidad, como la falta de conocimientos o la mala planificación.

Por lo tanto, a la hora de plantear el proyecto hay que tener en cuenta estas restricciones, y poner el foco sobre ellas, para así poder evitar un mal resultado. En la planificación se deberán de considerar todos estos parámetros y programar pequeños controles de calidad para cerciorarse de que la calidad del proyecto es la correcta. Por último se podrán aprovechar estas restricciones ideando nuevas funcionalidades a raíz de estos inconvenientes.

Resumen:

En este apartado se ha descrito el contexto en el que se enmarca el propio proyecto, para así determinar cuales son las necesidades que se pretenden abordar, así como los objetivos que marcarán el rumbo de todo el proceso. Además, se describen los principales aspectos que pueden afectar al proyecto a modo de limitaciones.

CAPÍTULO 2: ARQUITECTURA

Una vez determinado el contexto en el que se desarrolla el proyecto, y las carencias que se pretenden abordar, se procede a desarrollar toda la parte relacionada con la estructura que compondrá la solución que se ha propuesto para poderla llevar a cabo de forma satisfactoria. Es importante remarcar que el proyecto cuenta con diferentes aplicaciones, y cada una de ellas con 2 formatos característicos. Por lo tanto, primero se deberá definir una arquitectura general, apoyada en un diagrama clarificador, y después se irá desarrollando cada parte de la arquitectura.

2.1. Arquitectura general

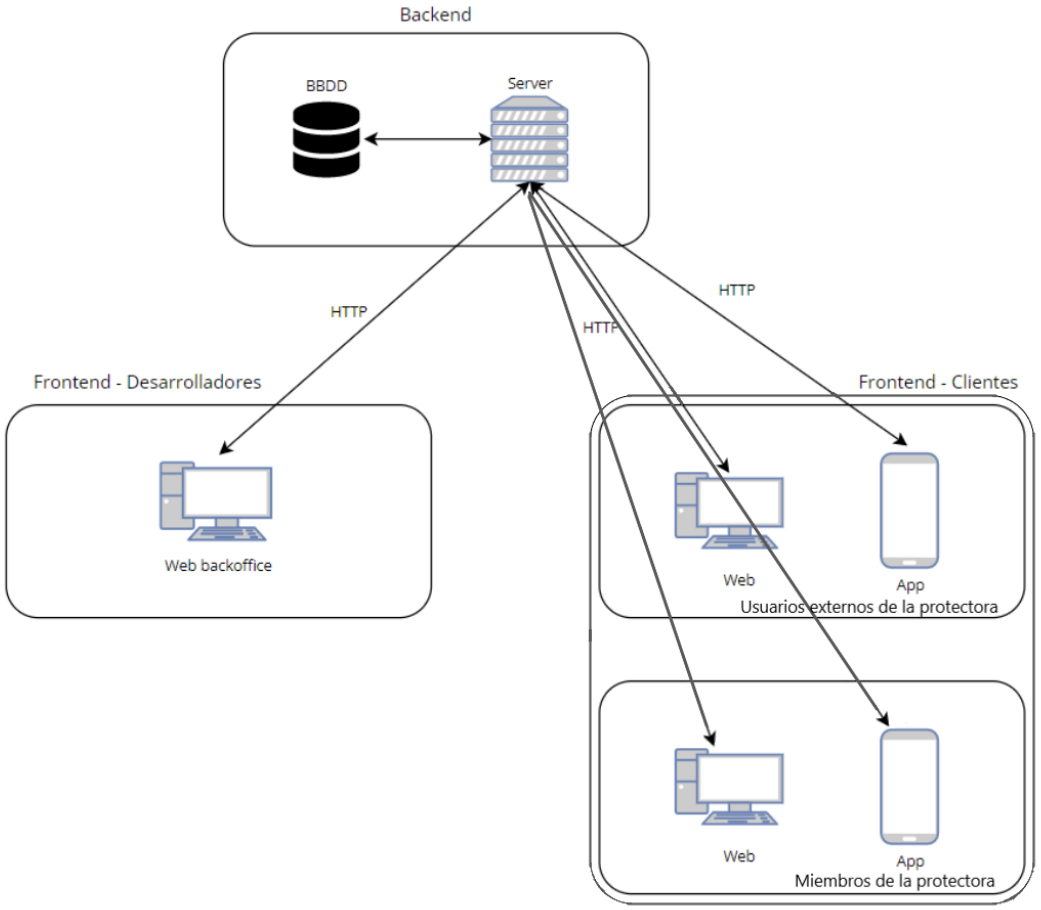

Fig 2.1 Esquema arquitectura del proyecto

La primera parte de la arquitectura es el *backend*, de esta pieza del sistema dependen directamente el resto; se podría definir como la lógica del proyecto, a raíz de esta se realizan los procesos para poder tratar y visualizar los datos en las aplicaciones. Estas aplicaciones forman parte del *frontend*, es decir, la parte

visible; esta parte no contiene una gran lógica (sí que se pueden realizar pequeñas operaciones, pero no es su finalidad), pero también es muy importante, ya que será lo que la clienta vea y trate, por lo que es muy importante crear un diseño claro e intuitivo. Tal y como se ha comentado anteriormente, esta parte está dividida en 2 aplicaciones, aunque en la práctica sean 3. Una de ellas corresponde a la aplicación para los usuarios externos de la protectora, otra para las voluntarias, y finalmente una para los desarrolladores o administradores de las cuentas. Esta última parte no debe de ser tan visual como las anteriores, ya que su uso es meramente administrativo, pero sí que deberá de cumplir algunos de los requisitos (como que sea intuitiva o cómoda para el usuario). Esta última parte se llama *backoffice* y se realizará vía web.

Una parte crucial del proyecto, que no forma parte del diagrama y que hasta ahora no ha sido mencionada, es el término de *DevOps*. Esta herramienta nos permitirá realizar modificaciones sin que el cliente lo note, ya que una de sus funciones es la de la creación de un contenedor en la que almacenar la aplicación, por lo que podemos modificar esa versión de manera local, y una vez acabadas las correcciones, subirla a producción para que el usuario se la vuelva a descargar (actualizar).

2.2. Frontend

Tal y como se ha comentado a lo largo del proyecto, el frontend es la parte que desarrolla, es decir, la parte visual de la aplicación. Pero esta parte por sí sola no tiene una gran utilidad, ya que solo se encarga de distribuir los elementos por la pantalla y llamar a las funciones creadas en el *backend* u otras APIs para recibir los datos. Pero el no presentar una gran complejidad detrás no implica que su desarrollo sencillo. La complejidad de este apartado se basa en la creación de un diseño intuitivo, atractivo al usuario, funcional y con una armonía. Es difícil conseguir unos buenos resultados en todas los parámetros, ya que para conseguirlo hay que tener en cuenta la opinión de muchos usuarios, así como ir desarrollando diferentes versiones en las que se incluyan mejoras en pequeños procesos que pueden dificultar la experiencia satisfactoria del usuario; todo esto conlleva una gran dedicación y estudio que requiere de tiempo y esfuerzo.

Actualmente se cuenta con numerosos lenguajes de programación para implementar esta parte, por lo que es importante decidir cuál de ellas es la idónea para el proyecto. Cada lenguaje ofrece diferentes características, por lo que uno de los primeros pasos a la hora de iniciar el proyecto será definir las distintas tecnologías. En este caso escogeremos entre las siguientes opciones:

- **Angular[8][9]:** Angular es un *Framework* que utiliza los lenguajes de programación *TypeScript* y *HTML*, que se basa en la creación de aplicaciones de una sola página. Estas aplicaciones se basan en módulos NG, uno base y otros complementarios. Esta tecnología ofrece diversas características, como:

- Multiplataforma: permite la creación de una aplicación compatible con la mayoría de navegadores. Al ser una aplicación web estará disponible tanto para móvil como para web. Con la combinación de otras tecnologías como Ionic se pueden crear aplicaciones nativas. Por lo que la mayor desventaja consistiría en la dependencia de otro *Framework* para la realización de una aplicación móvil.

- Gran rendimiento: el uso de plantillas agiliza la creación de código, pudiendo reutilizar una misma parte de código para funcionalidades similares. Además, permite la realización de pruebas de todo el código (o parte de él), por lo que es posible la comprobación del correcto funcionamiento de la aplicación, ahorrando tiempo en la resolución de problemas. Por otra parte, tiene implementada inteligencia artificial, con la que es posible completar partes del código de manera automática y la corrección de errores de forma instantánea.

- Llamativo visualmente: el *Framework* cuenta con preparativos para la realización de complejas coreografías y animaciones para crear un software ágil. Aunque la visualización de estos gráficos puede verse perjudicada si el usuario utiliza la aplicación en un móvil o en un ordenador. Entre los preparativos también cuenta con una parte de accesibilidad, pudiendo realizar aplicaciones accesibles de una manera sencilla.

- **React**[10][11]**:** al igual que *Angular*, está orientado al desarrollo de aplicaciones web. En este caso, se trata de una librería *JavaScript* que con la combinación de otras librerías, permite generar aplicaciones. El funcionamiento es bastante similar al de *Angular*, ya que a raíz de un componente vamos creando otros componentes independientes, que acabarán formando el software final. Las principales propiedades de esta librería son:

  - Multiplataforma: al igual que *Angular*, nos permite desarrollar una aplicación, compatible con la gran mayoría de navegadores web. Recientemente se ha implementado la opción de realizar software para *IOs* y *Android*, por lo que con esta tecnología podremos desarrollar tanto aplicaciones web como aplicaciones móviles de manera nativa.

  - Gran rendimiento: esta librería cuenta con grandes funcionalidades que permiten sacar el mayor rendimiento al código. Brinda una gran velocidad, gracias a la opción de utilizar partes individuales en el lado del cliente o en el del servidor, aumentando así la velocidad de desarrollo.

- **Flutter**[12][13][14]**:** Flutter es un *SDK* desarrollado por *Google*, especializado en la creación de aplicaciones móviles, pero que también es compatible con otros tipos. Este conjunto de herramientas de desarrollo software (*SDK*) utiliza el lenguaje de programación *Dart*, muy similar a otros (sobre todo C/C++), y con una curva de aprendizaje un poco elevada (una de sus mayores desventajas). Este lenguaje está optimizado para el desarrollo de aplicaciones. Las principales cualidades son:

    - Multiplataforma: esta es una de sus grandes ventajas. Con *Flutter* podemos generar diferentes aplicaciones desde el mismo código. Es decir, con un mismo código podemos generar una aplicación móvil, web o de escritorio. Esto ahorra tiempo, y permite ofrecer al cliente un amplio abanico de posibilidades.

    - Gran rendimiento: otra de sus características es la rapidez, *Flutter* ofrece 2 herramientas para mejorar el rendimiento. La primera de ellas es el *Stateful Hot Reload*, que permite realizar cambios en nuestro código y que estos queden reflejados instantáneamente en nuestra aplicación, aumentando así la velocidad de desarrollo. La segunda es la diferenciación de dos tipos de compilación; la primera de ellas la *JIT* (*Just In Time*) (Anexo 1), esta compilación permite compilar el software durante su ejecución (y no previamente), permitiendo así el *Stateful Hot Reload* y visualizar los cambios en tiempo real; la segunda compilación es la *OAT*, que compila el código antes de su ejecución, convirtiendo el código de *Dart* a nativo, reduciendo así el tamaño y obteniendo un mejor rendimiento en el dispositivo (se ejecuta una vez tenemos listo el desarrollo de nuestro software y la queremos subir a producción). La segunda herramienta para mejorar nuestro rendimiento es la de *Skia*, un motor gráfico que renderiza en 2D los elementos gráficos, permitiendo así realizar aplicaciones con altos fotogramas por segundo.
      Desde el punto de vista de la mejora del código, contamos con herramientas como los *lints*, que trabaja con el analizador de *Dart*, y nos permite prevenir errores y mejorar la eficiencia.

    - Llamativo visualmente: una de las características de *Flutter* es el abanico de interfaces que permite llevar a cabo (desde sencillas hasta complejas), permitiendo realizar algunas muy concretas. Una de sus excentricidades es la definición de *widget*, una herramienta para crear y describir la interfaz gráfica (*UI*). Ofreciendo así una gran flexibilidad, permitiendo crear interfaces únicas. Además, cuenta con numerosos *widgets* precreados, lo que nos permitirá implementar funcionalidades de una manera sencilla y rápida.

Tal y como se ha descrito, contamos con tres potentes utilidades, cada una con sus ventajas y desventajas, pero útiles para el proyecto a desarrollar, por lo que hay que encontrar aquella que se adecúa más a las necesidades del trabajo.

Para ello se plasmarán todas las características en una tabla para poder tomar una decisión contemplando de forma más visual las características y ventajas que ofrece cada una.

Tabla. 2.1 Tabla resumen características diferentes tecnologías

	Angular	React	Flutter
Multiplataforma	1/2	2/2	2/2
Diversos navegadores y SO	☑	☑	☑
Diversas plataformas	☐	☑	☑
Rendimiento	3/4	2/4	4/4
Reutilización de código	☑	☑	☑
Pruebas rápidas	☑	☑	☑
Inteligencia artificial	☑	☐	☑
Renderización de la versión definitiva	☐	☐	☑
Parte Gráfica	1/4	1/4	4/4
Animaciones predefinidas	☑	☐	☑
Renderizado para diversos dispositivos	☐	☑	☑
Funcionalidades predefinidas	☐	☐	☑
Diseño funcional y moderno	☐	☐	☑
Total	**5/10**	**5/10**	**10/10**

Tras el análisis de las diversas tecnologías se puede diferenciar una clara ganadora, *Flutter*. Esta tecnología nos permitirá tener una gran flexibilidad a la hora de crear la aplicación, ya que nos permite desplegar el mismo código en diversas plataformas, obteniendo un gran rendimiento y una experiencia visual moderna, intuitiva y simple.

La decisión de esta tecnología nos permite realizar las aplicaciones en diversas plataformas (web, android, IOs y escritorio), por lo que finalmente el software

será multiplataforma, dejando así a la elección del usuario la elección de la vía con la que acceder.

2.3. Backend

Tal y como se ha comentado anteriormente, *frontend* corresponde a toda la parte visual de la aplicación, pero esta necesita tener una lógica detrás que permita realizar ciertas acciones, además de conectarse con la base de datos y el servidor. El *backend*[15][16][17][18] se encarga de esa conexión, recibe las peticiones que el usuario realiza mediante el *frontend*, realiza las acciones necesarias (consultar datos, realizar acciones y/o hacer consultas a *APIs* externas) y devuelve el resultado a la aplicación. Para poder hacer estas consultas de una manera rápida y eficiente la conexión debe de ser continua, y el *backend* tiene que estar implementado de tal manera que realice las acciones necesarias lo más rápido posible para que el usuario tenga una buena experiencia.

Por lo que los principales objetivos que hay que tener en cuenta son: tener una buena comunicación entre la parte delantera y la trasera, obtener un código limpio y eficiente, y reducir la complejidad del *frontend* (entregando resultados claros y sencillos).

Para obtener un buen resultado hay que obtener un código que permita: tener escalabilidad, ya que a lo largo del desarrollo y con el transcurso del tiempo, la aplicación y el software sufrirán cambios, por lo que la estructura tendrá que ser adaptable a estos; navegar de manera segura, debido al tratamiento de datos que pueden ser sensibles, con lo que deberemos de desplegar técnicas seguras; obtener una robustez necesaria en momentos en los que el consumo de la aplicación sea muy alto y pueda perjudicar a la experiencia del usuario.

A diferencia del apartado anterior, aquí están claras las tecnologías y lenguajes a utilizar. Para el desarrollo del *Backend* utilizaremos *TypeScript, Express* y *Node*. La combinación de estos tres factores convergen en una herramienta muy potente para este desarrollo, pudiendo realizar las tareas de manera sencilla, ágil y eficiente.

- **Lenguaje de programación:** el lenguaje utilizado es *TypeScript*, muy similar a *JavaScript* pero con algunos beneficios que lo hacen más indicado para el proyecto. Ofrece un lenguaje altamente tipado, cosa que nos evitará muchos errores relacionados con las variables y las asignaciones de valores a estas; ofrece inteligencia artificial, que proporciona sugerencias activas a medida que se escribe el código; ofrece una mejor compatibilidad que *JavaScript*.

- **Entorno de ejecución:** el entorno utilizado es node.js, este entorno permite crear aplicaciones y herramientas. En este proyecto será utilizado para generar un servidor web, que escuchará y atenderá las peticiones HTTP. Una de las características de *Node* es el gran

rendimiento y escalabilidad que ofrece, ahorrando energía en el caso de que no tenga peticiones a procesar; cuenta también con miles de paquetes reutilizables, que ahorrarán tiempo en el proceso de creación. Pero este entorno cuenta con algunas limitaciones en cuanto al manejo de algunas peticiones *HTTP* (como *GET*, *POST* o *DELETE*), por lo que será necesario un *framework* para poder realizarlas.

- **Framework:** el *framework web* (marco de trabajo que facilita la organización del proyecto, ahorrando tiempo y recursos) más popular para este entorno y lenguaje es *Express*. Proporciona acceso a las acciones anteriormente mencionadas, además de otras mejoras como la integración con motores de renderización, permitiendo generar respuestas a través de la cumplimentación de una plantilla, y muchas otras funcionalidades.

En esta sección también se debe de tener en cuenta el uso y almacenaje de los datos recibidos, es decir, las bases de datos. En este caso se utilizará MongoDB.

- **Mongodb[22]:** este programa de código abierto permite crear y gestionar diversas bases de datos NoSQL (Anexo 2). Este software permite la búsqueda, consulta y análisis de los datos guardados en la base de datos. Estos datos son cargados a una nube, permitiendo así consultarlos desde cualquier dispositivo en cualquier momento. Esta herramienta será esencial para el *backend*, ya que almacenará todos los datos necesarios para la utilización de la aplicación (datos de los usuarios, variables para realizar operaciones, etc.).

2.4. Backoffice

El término de *backoffice* se refiere a la parte administrativa de la aplicación, aquella en la que un administrador podrá tratar datos importantes (cuentas de usuarios, objetos creados, etc.). Por lo que para que la administradora pueda acceder a estos datos y los pueda tratar debe de tener una aplicación. En este caso será una aplicación web, ya que son más sencillas de realizar, no se requiere ningún tipo de descarga (lo que la hace más accesible), y no tiene porque ser excesivamente visual, por lo que la guía para decidir la tecnología que se usará estará más relacionada con la funcionalidad y rapidez de programación que con la parte visual.

Tabla. 2.2. Tabla resumen características diferentes tecnologías

	Angular	React	Flutter
Multiplataforma	1/1	1/1	1/1
Diversos navegadores y SO	☑	☑	☑
Rendimiento	3/4	2/4	4/4
Reutilización de código	☑	☑	☑
Pruebas rápidas	☑	☑	☑
Inteligencia artificial	☑	☐	☑
Renderización de la versión definitiva	☐	☐	☑
Total	**4/5**	**3/5**	**5/5**

Para llevar a cabo esta decisión hay que recuperar la tabla vista con anterioridad (Tabla 2.1.), en la que se comparaban las diferentes tecnologías planteadas para el desarrollo del *frontend*, y solo teniendo en cuenta los factores "Multiplataforma" (excluyendo el subapartado de "Diversas plataformas", ya que no es necesario) y "Rendimiento". Podemos ver como las tres tecnologías están muy igualados (Tabla 2.2.). Si se tiene en cuenta que la plataforma a la que está enfocada esta aplicación es la de web, la opción indicada sería la de *Angular*, ya que es una de las tecnologías puntera en este ámbito, ofreciendo grandes ventajas como la gran cantidad de plantillas y la sencillez a la hora de programar. Entre las 3 opciones la única que sobrepasa a *Angular* es *Flutter*, si se observa que solo representa una mejora a la hora de la renderización final, no representa un gran valor añadido en todo el proceso. De esta manera se puede conocer y desarrollar una tecnología más, con los conocimientos y experiencia que otorga.

Por lo que en este apartado se realizará una página web, mediante la tecnología *Angular*, para gestionar temas administrativos de la aplicación. Para definir las funcionalidades es necesario contar con las diferentes estructuras de datos, que serán definidas más adelante.

2.5. DevOps

El término inglés de *DevOps* proviene de la combinación de *development* (desarrollo) y *operations* (operaciones). Este proceso es muy amplio, por lo que abarca casi todos los ámbitos del proyecto, mejorando la estabilidad, la adaptación y el rendimiento. En cada parte juega un papel distinto, por lo que

es importante conocer cada uno de ellos para poder obtener el máximo beneficio posible. En este proyecto serán utilizadas estas técnicas en las siguientes fases[19][20]:

- **Planificación:** tras definir las características y funcionalidades, el equipo genera unas guías para llevar a cabo el proyecto de manera organizada y eficiente. Este método de planificación será descrito posteriormente, pero básicamente nos ofrece una gran agilidad y visualización de los procesos finalizados, en curso y planeados, aunque en este proyecto no se use esta metodología estrictamente, ya que está enfocada para el desarrollo en trabajos grupales, pero sí que se utilizarán los principios básicos.

- **Seguimiento y control:** una vez el software está desarrollado hay que implementarlo en el entorno de producción, de manera constante y con fiabilidad. En este proceso las técnicas de *DevOps* permiten realizar estas entregas de una manera sencilla, segura y escalable. Esto permitirá que los usuarios no sufran errores cuando se desarrollen actualizaciones, garantizando una gran disponibilidad, realizando este proceso de manera casi transparente para el usuario. De esta manera se podrá gestionar los errores detectados durante la fase de control sin perjudicar al uso de la aplicación.

Estas técnicas son muy útiles a la hora de trabajar en equipo, ya que cada departamento puede realizar las implementaciones necesarias, y posteriormente, unirlas todas (o ir añadiendo las actualizaciones de manera individual). Este concepto es relativamente nuevo, por lo que los beneficios actualmente son algo limitados (aunque otorgan una gran calidad respecto a las técnicas utilizadas anteriormente).

2.6. Herramientas a utilizar

Para llevar a cabo todos los procesos, serán necesarias diversas herramientas de diferentes ámbitos. Las más importantes son:

- **Visual Studio Code[21]:** Este editor de código fuente permitirá desarrollar las partes de *backend, frontend* y *backoffice* (se podrán utilizar los diferentes idiomas de programación gracias a las extensiones que permiten la compatibilidad con cada una de las técnicas). Esta herramienta será utilizada para picar todo el código necesario para el desarrollo de las aplicaciones. Además, cuenta con diversas herramientas que facilitarán el trabajo como: *IntelliSense*, un modelo de inteligencia artificial que otorga funciones de autocompletado, definiciones de funciones, etc.; depuración directa desde la aplicación, pudiendo aplicar *breakpoints* para poder seguir las funciones realizadas por el código, corrigiendo así todos los posibles errores; anexión a otras herramientas como *Git*, que permitirá tener todas las plataformas utilizadas conectadas, ahorrando tiempo y posibles errores.

- **Socket.IO**[23]**:** esta biblioteca de *JavaScript* permite la comunicación bidireccional en tiempo real entre un servidor web y un usuario. En este proyecto se utilizará para comunicar el cliente con el servidor, y poder transmitir información entre estos.

- **GitHub**[24]**:** este servicio de alojamiento de repositorios, permite compartir código, teniendo disponible todas las versiones realizadas, pudiendo recuperar cualquiera de ellas. Además, permite crear diferentes ramas, consiguiendo así una diferenciación entre las distintas secciones con las que cuenta el proyecto. Será muy útil para que el director del proyecto pueda consultar el código realizado, y poder ejecutar la aplicación y poder solucionar los errores.

- **Cloudinary**[25]**:** esta empresa tecnológica ofrece servicios de almacenaje y gestión de recursos audiovisuales (como vídeos o imágenes) mediante una nube. Su funcionamiento es el de una *API*: el *backend* realiza una petición a la nube, y ésta devuelve un enlace con la respuesta. Será utilizado para almacenar todas los recursos audiovisuales (imágenes y vídeos) que los usuarios utilicen en la aplicación, pudiendo realizar transformaciones en ellas (cambio de color, forma, etc.).

- **Agora.io**[26]**:** esta *API* permite realizar llamadas y videollamadas de calidad entre dos o varias personas. Está perfectamente adaptada a la tecnología flutter, así como a todas las plataformas (android, IOs, web, escritorio), y ofrece una gran fiabilidad y escalabilidad, por lo que será perfecta para implementar llamadas y videollamadas en las aplicaciones.

Resumen:

El desarrollo del proyecto está compuesto principalmente por:

- **Frontend:** aplicaciones web y móviles. Parte visual del producto final. Desarrollado con *Flutter*.

- **Backend:** lógica del proyecto, hace las peticiones al servidor y base de datas para poder realizar las acciones en el frontend. Desarrollado en *TypeScript, Express* y *Node.* La base de datos se realizará mediante *MongoDB*.

- **Backoffice:** frontend web para poder administrar los perfiles de los usuarios. Desarrollado mediante *Angular.*

- **DevOps:** gestionar las diferentes actualizaciones de las aplicaciones de manera ágil y sin perjudicar al usuario.

CAPÍTULO 3: PLANIFICACIÓN

Tras haber definido las tecnologías y la arquitectura con la que se llevará a cabo el proyecto, se describirán las diferentes metodologías utilizadas para organizar el trabajo y un plan en el que se describen los diferentes procesos que se seguirán para llevar a cabo el desarrollo de las aplicaciones.

3.1. Metodología scrum y kanban

Las metodologías *scrum* y *kanban*, son dos populares técnicas de organización de proyectos que se basan en los principios de *agile* que permiten realizar proyectos de una manera rápida y simple. Aunque ambas metodologías son complementarias, cada una dispone de diferentes características que ofrecerán beneficios al proyecto.

3.1.1. Scrum

Tal y como se ha comentado anteriormente, en este proyecto no se ejecutará esta técnica estrictamente, ya que está enfocada en el trabajo en grupo, pero sí que se buscará utilizar las características de esta. La metodología *scrum*[27][28][29] se basa en la división de un gran proyecto en pequeños bloques (o *sprints*), pudiendo así repartir y priorizar las tareas. Dentro de estos bloques se asignan diferentes labores a los miembros del equipo, y se les otorga una dificultad (puntos de historia) que determinarán el volumen de trabajo de cada tarea, pudiendo planificar así cada *sprint* de manera equitativa.

La primera tarea a realizar consiste en la definición de los roles del equipo, ya que cada miembro contará con una responsabilidad durante todo el proyecto. Estos roles son:

- ***Product owner:*** este miembro del grupo es el nexo entre el cliente y los desarrolladores. Su trabajo se basa en escuchar las peticiones del cliente, plantear una solución y transmitir lo acordado al equipo. Este trabajo es esencial, ya que será el que decida la ruta del proyecto, y conseguirá que sea viable o no.

- ***Development team:*** compuesto por todos los desarrolladores. Son los encargados de estimar las tareas que se realizan en cada bloque, así como la manera en la que llevarlas a cabo. Una de las principales características es que son autoorganizados e iguales entre ellos, ya que todos son encargados de realizar las tareas de cada sprint, y en caso de no cumplir con todas ellas la responsabilidad es compartida.

- ***Scrum master:*** es el líder que se encarga de que la metodología *scrum* se lleve a cabo y sea entendida por todos los integrantes del equipo. Además, es el que se encarga de la organización y planificación del proyecto, con el objetivo de obtener una mejor eficiencia.

Cada bloque de trabajo se inicia con una reunión de equipo, en la que se define las tareas que se van a realizar, la duración del *sprint* y la prioridad y carga de trabajo de cada tarea. Lo primero que se debe determinar es la duración del *sprint*, esta no debe de superar el mes o mes y medio, ya que uno de los objetivos de esta técnica es la de dividir en pequeñas partes el trabajo, para así poder entregar pequeñas demostraciones de calidad al cliente. Para determinar la duración hay que concretar los objetivos para el bloque y como se conseguirán. Una vez definidos los objetivos, los desarrolladores proponen las diferentes tareas, otorgan los puntos de historia y el orden de prioridad. Para definir la puntuación de cada tarea, cada desarrollador otorga una puntuación (1, 2, 3, 5 u 8) en función de la dificultad, complejidad e incertidumbre. En caso de que las puntuaciones sean muy dispares entre ellas se hace un pequeño debate en el que cada componente razona su valoración y finalmente se llega a un consenso. El proceso se repite hasta valorar todas las tareas propuestas, al ver la puntuación final del sprint se valora si es razonable, y en caso de ser una carga de trabajo excesiva se desplazan algunas tareas al siguiente *sprint*. Todas estas tareas se ubican inicialmente en el tablón de tareas pendientes, y a medida que se inicien o se finalicen van siendo desplazadas al tablón correspondiente; esta metodología permite visualizar rápidamente las tareas pendientes para así poder evaluar el ritmo de trabajo.

Los siguientes días del *sprint*, se realiza una breve reunión (a ser posible al inicio de la jornada) en la que cada trabajador explica su avance, obteniendo así una imagen actual del proyecto. A la hora de finalizar el bloque se realiza una reunión final de *sprint*, en la que se valora si se han cumplido los objetivos, y se obtiene un *feedback* del trabajo realizado. Además, se planean los objetivos para el próximo bloque.

Al haber un único miembro en el equipo, la metodología debe de ser adaptada para poder obtener los máximos beneficios de esta técnica. Una de las adaptaciones radica en que los roles de equipo serán ejercidos por una única persona, aunque se deban de ejercer todos los roles y pasos para poder obtener un resultado satisfactorio. Esto no impide que las reuniones de inicio y fin de *sprint* no sean supervisadas por el director del proyecto, obteniendo así una instantánea del proceso, así como guiar y aconsejar en las decisiones a tomar.

3.1.1.1. Pivotal Tracker

La aplicación utilizada para llevar a cabo la metodología *scrum* es *Pivotal tracker*, esta herramienta permite dividir las tareas en los diferentes sprints, visualizando claramente las tareas que han sido iniciadas de las que no, y las que han sido finalizadas. Como se ha visto anteriormente, dentro de cada tarea se puede agregar el grado de dificultad, y a medida que vayamos completando labores, el programa hará una previsión de cuándo se completarán las diferentes tareas, y por tanto permitirán saber si pueden entrar dentro de un *sprint* o no. Estas estadísticas serán muy útiles para planificar los diferentes sprints, y a medida que se van completando las tareas, el *software* realiza previsiones más fiables.

3.1.2. Kanban

La metodología *Kanban*[30] busca visualizar el proceso del flujo de trabajo, para así poder mejorar la eficiencia y la productividad. Se busca que con un simple vistazo se conozca el estado del proyecto, para poder asignar o reprogramar tareas rápidamente. La división del trabajo se basa en pequeños bloques, en los que se describen los objetivos y tareas, que se irán desarrollando a medida que avanza el proyecto, generando nuevos bloques y flujos de trabajo. Una de las principales características de esta técnica es la de finalizar las tareas pendientes o iniciadas antes de iniciar nuevas, permitiendo así una mayor división y priorización.

Los componentes principales son el tablero, las tarjetas y flujos:

- **Tablero:** en este lugar se anotan las diferentes tareas. Normalmente se suelen dividir en tres columnas: tareas pendientes, iniciadas y finalizadas. De esta manera se pueden visualizar todas las tareas y el estado de cada una.

- **Carta:** se encuentran dentro del tablero, y es donde se describen las tareas a realizar, pudiendo dar detalles como la prioridad otorgada, fecha de entrega o una pequeña descripción.

- **Flujos:** esta herramienta es opcional, pero es de gran utilidad, ya que genera un mapa de flujos, en el que indica la fecha de entrega de cada tarea, el proceso en el que está, y lo más interesante, si de esa tarjeta depende otra. De esta manera se consigue tener un mapa conceptual del estado de las tareas, así como de las dependencias entre ellas.

3.1.2.1. Zenkit

Esta plataforma[31] compuesta por múltiples herramientas, nos permite sacar el máximo partido de la metodología *Kanban*. La división de aplicaciones hace mucho más sencilla la utilización de cada una, teniendo los procesos en interfaces separadas, pero unificadas en una única plataforma, pudiendo manejar los datos de las diferentes aplicaciones entre sí. Este programa permite gestionar los proyectos de una manera simple y obteniendo una mayor eficiencia. Tal y como se ha comentado, esta plataforma está compuesta por diferentes aplicaciones, donde cada una tiene unas funcionalidades diferentes, aunque la más potente y útil para llevar a cabo la metodología *Kanban* es la de *Zenkit projects*, que permite generar diferentes vistas, desde las cuales realizar diferentes funcionalidades, donde las más destacadas son:

- **Kanban global:** aquí se pueden generar diferentes tareas (donde se pueden incluir todo tipo de detalles como: estado, fecha de vencimiento, notas, añadir recordatorios, horas empleadas para llevar a cabo la tarea, etc.) y clasificarlas según si están pendientes, en progreso o hechas. De esta manera se cumple uno de los principales objetivos de las técnica

Kanban, poder ver de manera rápida y cómoda el estado de las tareas.

- **Planificación de recursos:** en esta pantalla se visualiza un calendario con las tareas que están programadas para cada día de la semana en función del usuario. Además, nos indicará de color verde si el total de las horas de las tareas asignadas a un usuario es menor al límite diario asignado, en naranja si es el valor del límite y en rojo si es superior (incluyendo el exceso de horas). Esta funcionalidad es muy cómoda para distribuir las tareas en horarios con poca disponibilidad, pudiendo planificar las tareas en función del tiempo que se quiere y se puede dedicar. Además se pueden consultar todos los detalles asignados a la tarea, visto en la funcionalidad anterior (*Kanban global*).

- **Semana:** en esta vista, se encuentra una lista con todas las tareas pendientes para la semana en curso. En relación a esta pantalla encontramos algunas parecidas como: **lista global de tareas** (que permite ver una lista de todas las tareas, tanto completadas como pendientes), **todas las tareas terminadas** (que lista todas las labores finalizadas), **hoy** (idéntica a la función “semana”, pero cambiando el periodo de la semana al día en curso)

- **Reporte:** la plataforma nos ofrece diversas estadísticas para poder controlar el ritmo de trabajo, las tareas realizadas y las pendientes. Estas herramientas son muy útiles para poder analizar los errores cometidos y solventarlos para trabajar de una manera más eficiente.

- **Gant:** esta pantalla ofrece un esquema en el que se pueden visualizar las dependencias entre tareas, la fecha de vencimiento de cada tarea, el proceso en el que se encuentra, los detalles, etc.

3.1.3. Scrum vs kanban

En este apartado, tal y como se observa en la tabla (Tabla 3.1.), se realizará una comparativa entre las características de ambas técnicas, y cómo es posible la conjunción entre ellas, mejorando la eficiencia de la realización del proyecto.

Tabla 3.1. Comparación entre *Scrum* y *Kanban*

	Scrum	**Kanban**
Roles	*Product owner*, *development team* y *scrum master* (ejecutados por una única persona debido a la especialidad del proyecto)	Sin roles de equipo definidos
Ritmo	Fijado	Flujo continuo
Cambios	No recomendados durante un *sprint*	Se pueden producir en cualquier momento
Métrica clave	Velocidad	Tiempo
Delegación y priorización	Se pueden empezar nuevas tareas sin haber finalizado las anteriores	No empezar una acción hasta que no se han finalizado las anteriores

3.3. Plan de trabajo

Con tal de alcanzar los objetivos determinados, es importante contar con una planificación orientativa de la distribución de las tareas a lo largo del proyecto. Para alcanzar los objetivos de tiempo, se ha dividido en cinco partes diferenciadas: la redacción de documentación, la planificación del proyecto, el desarrollo de las aplicaciones, la entrega del producto y el análisis de datos. La fecha límite de entrega se establece alrededor de finales de enero, por lo que la fase de desarrollo debería de completarse a mediados del mismo mes, aunque tal y como se muestra en la figura (Fig 3.1.), se planea llevar a cabo la fase hasta la finalización del proyecto, con tal de realizar o corregir los errores y modificaciones planteadas por el cliente. Como se puede observar en la figura, el periodo de desarrollo está segmentado en *backend, frontend* y *backoffice*, donde la primera de ellas es la que será iniciada con anterioridad, llevándose a cabo hasta los últimos días del proyecto, debido a la gran dependencia que ejerce sobre *frontend*, que será la segunda fase en iniciar, ya que conlleva un gran volumen de trabajo; por último se cuenta con el apartado de *backoffice*, que es el que menos esfuerzo y tiempo requiere. Así mismo, aunque durante todos los procesos se analicen los datos obtenidos, se establece una fase final en la que analizar en profundidad el trabajo realizado, así como los objetivos realizados; el inicio de esta sección es anterior a la entrega del producto para poder analizar las funcionalidades realizadas y cerciorarse de que el producto final es el deseado por el cliente. Con tal de realizar un control del proyecto y del adecuado progreso, se definirán reuniones cada cortos periodos de tiempo (cada dos semanas o un mes) entre el desarrollador y el director del proyecto, y un poco más espaciadas con el cliente. Las reuniones que se llevarán a cabo

con el director se denominan *sprints*, y más adelante se detallarán los detalles de estos eventos.

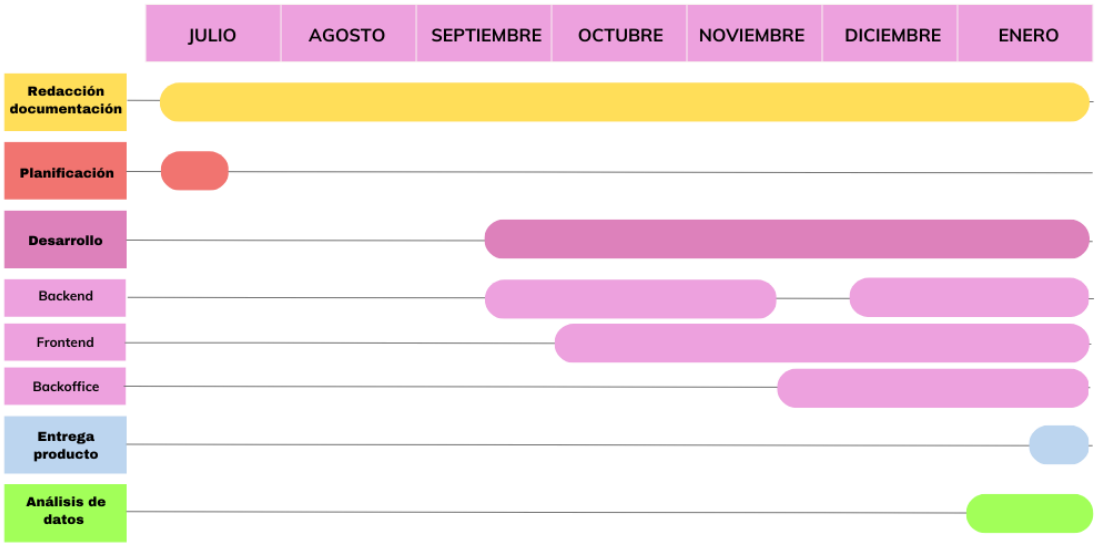

Fig 3.1. Propuesta plan de trabajo

3.3.1. Definición sprints

Tal y como indica la definición dentro del ámbito de la metodología *scrum*[32], un *sprint* consiste en un conjunto de pequeñas entregas de entre una y cuatro semanas de duración, cuyo objetivo es el de finalizar secciones del proyecto e ir comprobando la experiencia del usuario, así como contar con la seguridad de obtener un resultado satisfactorio. Aunque típicamente en esta metodología se determina el grado de dificultad de cada una de las tareas planteadas al inicio de cada *sprint,* en este proyecto el nivel de facilidad lo ha asignado el desarrollador del proyecto, al igual que las tareas asignadas a cada *sprint*. Este método, junto a las herramientas *Zenkit* y *Pivotal Tracker* ha permitido controlar el estado del proyecto en cada momento, así como calcular de manera orientativa si los objetivos planteados son factibles y realistas en el espacio de tiempo asignado. Además, esta división del proyecto, ayuda a la organización y distribución del trabajo, evitando así los picos de alta carga de trabajo.

Resumen:

En este proyecto se utilizarán metodologías ágiles e incrementales, *Kanban* y basadas en *Scrum* (ya que será realizada por una única persona), para poder gestionar el trabajo de manera visual, organizada y eficiente. Para ello se utilizarán las aplicaciones de *Zenkit* y *Pivotal Tracker*. La división de las diversas entregas y reuniones de control del proyecto serán definidas en *sprints*.

CAPÍTULO 4: DISEÑO

Una vez definida la finalidad del proyecto y la forma en que se llevará a cabo, es esencial precisar el diseño con el que contarán las diferentes aplicaciones, teniendo en cuenta las diferentes necesidades, y las herramientas necesarias para poder llevar a cabo de forma óptima las funcionalidades de las aplicaciones.

4.1. Funcionalidades

El primer paso para poder llevar a cabo un diseño eficiente consiste en definir las funcionalidades necesarias, como se usarán estas y un flujo en el que ver los posibles pasos de ejecución del usuario. Para ello se escribirá un pequeño ejemplo de uso.

Un día Fran estaba paseando por el paseo marítimo de Vilanova i la Geltrú cuando se encontró a un grupo de gatos maullando. Fran, amante de los gatos, se acercó a ellos para darles un poco del pan que llevaba cuando llegaron 3 chicas, voluntarias de la asociación “amics i protectors dels gats”. Ellas le explicaron la labor que hacían y que si estaba interesado podía descargarse la aplicación “Protecat” en la que informarse de como formar parte del equipo de la protectora, y de cómo adoptar a algún gato. Al llegar a casa Fran se descargó la aplicación y miró la información y los requisitos para formar parte de la asociación y, al cumplir con los requisitos y estar interesado, rellenó el formulario para inscribirse. Al momento recibió un correo electrónico en el que se le informaba de que su petición había sido recibida y que en unos días recibiría información sobre esta. Fran siguió investigando dentro de la aplicación, ya que al abrirla había visto un gato del que se había enamorado, por lo que se dirigió a la sección inicial de la aplicación e indicó que quería adoptar a ese gato, seguidamente vió otro gato del que también se enamoró, pero sin querer rechazó la adopción. Para solventar el error, accedió a la sección en la que se encuentran todas las solicitudes realizadas, y cambió la decisión del segundo animal. Tras esto cumplimentó los formularios requeridos para las adopciones en las que estaba interesado, y al igual que con la solicitud de ingreso en la protectora, recibió un correo informando de la recepción de sus solicitudes, aunque una de ellas fue rechazada, ya que no cumplía con las necesidades del animal; después descubrió que si hubiese completado su perfil la aplicación ya hubiese filtrado a todos aquellos gatos que eran compatibles con él. Después visualizó todos los retos económicos con los que contaba la protectora, y obtuvo el usuario de paypal para poder contribuir, además, pudo enviar el dinero equivalente a 5 latas de comida y una caja de pastillas, gracias a la calculadora gráfica de la aplicación, en la que se indican los productos que se quieren donar y calcula el importe. Finalmente, al ser rechazado del proceso de adopción, decidió formar parte de las casas de acogida, por lo que pudo acoger a la pequeña "Michi" gracias al anuncio publicado por la protectora. Tras 1 semana, Fran recibió el correo de confirmación de acceso a la protectora y el enlace de descarga de la aplicación móvil para acceder al área de voluntariado. Una vez descargada la aplicación y tras haberse registrado en esta, se dirigió al menú lateral, fue a la sección de tareas, y se apuntó en tareas relacionadas

con la colonia de la EETAC cada lunes, además de colaborar en la parada del sábado por la mañana. Estas dos tareas se le incluyeron en el calendario, donde podía ver claramente cada tarea, y así no olvidarse de ninguna. Al siguiente lunes, Fran fue a la colonia indicada para alimentar a los animales, de camino consultó la ubicación exacta de esta gracias a la opción de mapa. Además, pudo ver un pequeño resumen en el que indicaba el número de gatos que había y la necesidad de vigilar a Enzo, ya que días anteriores parecía estar resfriado. Al llegar visualizó todos los gatos que habían, y se dió cuenta de que aunque el número total era el correcto, faltaba el pequeño Noé, y había aparecido un nuevo gato, por lo que dejó todos estos datos anotados en la aplicación. Al poner el pienso, se dió cuenta de que el recipiente no estaba en su lugar normal, pero al ser nuevo no estaba seguro de si estaba en lo cierto, por lo que decidió hacer una videollamada con Dianna, quien le indicó el lugar correcto y le dió las gracias, ya que sino los animales estarían desubicados y podrían moverse del área y perder así parte de la colonia. Antes de regresar a casa, consultó el inventario de la protectora, para revisar si era necesario comprar más latas de alimento o no para la próxima semana.

4.1.1. Requerimientos funcionales

Por lo que si se hace un recuento de las funcionalidades con las que tiene que contar la aplicación utilizada por las voluntarias de la protectora serían las siguientes:

- Mapa que mediante la utilización de la ubicación de nuestro dispositivo, nos indicará cada una de las colonias, pudiendo acceder a la información asociada a cada una de ellas, como por ejemplo los animales que se encuentran o la medicación prescrita para cada uno de ellos. Además de contar con un registro en el que la persona encargada pueda ingresar las tareas realizadas u otras observaciones (como el comportamiento extraño de un gato, la desaparición de algún gato, etc.).

- En complementación con la idea anterior, se generará la opción de realizar videollamadas, para poder consultar cualquier duda al resto de voluntarias.

- Gestión de las solicitudes de adopción, mediante el cual la administradora podrá controlar las solicitudes de adopción recibidas por cada animal.

- Inventario de los recursos de la protectora, donde se contará con un recuento de los productos obtenidos por la asociación. Además, se podrá asignar un producto a una colaboradora, teniendo así un control total de donde está cada una de las mercancías.

- Para mejorar la organización de la asociación, se creará un calendario en el que anotar la distribución de los diferentes turnos de las colonias. Asimismo, se podrán crear tareas y asignarlas a cada una de las

colaboradoras, pudiendo visualizarlas en forma de lista o en el calendario.

Como se puede observar la mayoría de funcionalidades están centradas en la gestión de la protectora, cosa que quizás se podría realizar con otras aplicaciones, pero esta cuenta con una gran funcionalidad diferenciadora y muy útil para este cliente. El poder tener un registro de las colonias en las que trabajan, pudiendo hacer anotaciones, genera un gran beneficio y facilita considerablemente el trabajo a realizar. Por otro lado, el uso de esta aplicación ofrece una centralización de todos estos servicios, haciendo las gestiones mucho más sencillas y teniendo una experiencia como usuario mucho más cómoda y rápida.

En cuanto a las ideas propuestas para la aplicación enfocada a las usuarias externas de la protectora contamos con:

- Crear un método llamativo para la adopción de los gatos, con el objetivo de tentar a los usuarios para que adopten. Cuando el usuario proceda al proceso de adopción, se le otorgará un formulario con una serie de preguntas básicas para considerar si el perfil se adapta a las necesidades del animal, en caso de que el resultado sea favorable la candidatura del posible adoptante pasará a la administradora de la asociación siguiendo con el proceso de selección. En caso negativo la candidatura será rechazada y notificada al usuario. Esto genera una reducción drástica de las gestiones realizadas por la protectora.

- Otra de las funciones propuestas para esta aplicación es la creación de un muro principal en el que colgar noticias. Entre ellas se encontrarán la demanda de casas de acogida o donaciones para casos extraordinarios (gastos imprevistos, etc.). Estos casos también contarán con una sección independiente para poder consultar los casos abiertos de una manera mucho más rápida y eficiente. En el caso de las donaciones repentinas, se generará una barra con el objetivo del dinero necesario y una progresión a medida que se vayan recibiendo donaciones.

- Generar un método visual para realizar donaciones, en el que se integren pequeños artículos, con su respectivos precios, para que el usuario pueda ir añadiendo los artículos y finalmente hacer una donación del valor de la mercancía. De esta manera conseguimos que el usuario sea consciente de en que se invierte el dinero recaudado, y el coste real de los artículos. Un pequeño ejemplo para clarificar la funcionalidad sería el siguiente: poner la imagen de una lata de comida con el precio (0.49€), debajo de este el usuario puede añadir la cantidad de productos que quiere donar (5), y finalmente haría una donación con el valor correspondiente (5 x 0.49€ = 2.45€). Aunque las donaciones no se puedan realizar mediante la aplicación (debido a su complejidad), se le ofrecerá las facilidades para poder realizar la donación (usuario de paypal, número de cuenta o número de bizum).

- Una de las tareas que realiza la asociación para captar fondos es la venta de merchandising. En la aplicación habría una sección similar a la anterior en la que poder visualizar los diferentes artículos y poder comprar los deseados.

- La siguiente función se centra en la obtención de información de la asociación, acercándose así al usuario, para que éste pueda entender la labor que se lleva a cabo, y así poder decidir de que manera puede llegar a implicarse.

4.2. Frontend

Por tal de asegurar el éxito de la aplicación una vez ésta sea liberada al mercado es necesario tener en cuenta aquellos elementos que harán que los usuarios tengan una buena experiencia con las apps. Estos factores se llevarán al terreno práctico con un diseño previo de las aplicaciones para asegurar que estos requisitos se cumplen, y así evitar errores durante el proceso de programación de las apps.

4.2.1. Aspectos relevantes para el diseño

Tal y cómo indican múltiples expertos en UX[33] (User Experience), hay 7 factores claves que harán que la experiencia del usuario sea un éxito, y por lo tanto nuestro producto consiga ser usado por el máximo número de personas. Estos factores se basan en la facilidad y utilidad de uso de la aplicación, y son los siguientes:

- **Útil:** el principal objetivo es que nuestro proyecto cumpla las necesidades del usuario, una vez todos los problemas presentados por el cliente han sido solventados se tienen en cuenta el resto de factores, pero este es el punto imprescindible para que la experiencia del usuario sea satisfactoria.

- **Usable:** en este punto se busca conseguir los objetivos del usuario de manera efectiva y eficiente. Para cumplir este objetivo hay que buscar que el diseño sea sencillo y accesible a la morfología de las personas; por ejemplo que se pueda acceder con una sola mano a los controles principales, haciendo la experiencia mucho más eficiente.

- **Encontrable:** este factor trata tanto de la información que se proporciona dentro de la aplicación, como de la propia aplicación. Dentro de la aplicación, la información debe de estar organizada de tal manera que el usuario la encuentre fácilmente. Esto se logrará situando la información o los elementos que configuran la app en el mismo lugar entre diferentes pantallas y/o en lugares visibles. El segundo punto no afecta directamente a este proyecto, ya que la aplicación será entregada a los usuarios finales, al menos en la parte inicial de este, pero hay que

tener en cuenta que una vez el producto sea puesto en funcionamiento, la aplicación debe de ser visible y darse a conocer al máximo número de personas, por lo que debe de estar disponible en las principales tiendas como la *PlayStore* y la *AppleStore*.

- **Creíble:** el usuario busca que la aplicación que está utilizando sea fiable y que su información y datos no están en peligro, por lo que es muy importante dar toda la información sobre el tratamiento de sus datos, así como utilizar proveedores de confianza y utilizar la imagen de estos proveedores externos para que el usuario confíe en que la aplicación es fiable.

- **Deseable:** para que la experiencia sea buena, el usuario debe de desear la aplicación. Este elemento es complejo de alcanzar, ya que es muy subjetivo, pero se debe de buscar un diseño atractivo para la mayor parte de la población. Además, se deberá de tener en cuenta las tendencias del momento por tal de presentarse como un producto moderno, y así poder causar una buena primera impresión a la mayor cantidad de usuarios posibles. Algunas técnicas para poder lograrlo consisten en contar con una imagen de marca característica (branding) que sea potente visualmente para que el producto que se esconde detrás de esta sea recordado fácilmente. Además, esta imagen deberá configurarse de forma uniforme y armónica por tal de no causar un impacto negativo en los usuarios en un primer momento.

- **Accesible:** a la hora de diseñar una aplicación hay que tener en cuenta de que cualquier persona pueda acceder a esta, indistintamente de sus características como podrían ser una una discapacidad, por lo que se deben de utilizar técnicas para que estas personas puedan acceder y utilizar el aplicativo sin ningún problema.

- **Valioso:** el producto final debe de aportar un valor final a la sociedad, y tener un coste de producción similar o inferior al del problema a solucionar, siendo así rentable y otorgando un beneficio. En este caso al tratarse de una aplicación para una asociación y no obtener un beneficio económico, se puede medir en el tiempo y ayuda que representa para la asociación.

Otra de las normas sobre el diseño de aplicaciones es que un usuario no debe de pulsar más de tres veces para realizar una acción[34], es decir, desde cualquier pantalla el usuario debe de poder realizar todas, o casi todas, las acciones.

Si extrapolamos estos datos a los de nuestro proyecto, debemos de crear una aplicación que cumpla con todas las demandas de la protectora, así como que su uso sea sencillo e intuitivo, tanto para las voluntarias como para los usuarios externos a la protectora. Las pantallas de la aplicación serán similares entre sí, para que así se pueda encontrar fácilmente la información y los botones para realizar las diferentes acciones. Por otro lado se utilizarán proveedores externos de confianza como google, para iniciar sesión y registrarse, así como

en otras funciones (ubicación de las colonias en el mapa). Para que la aplicación sea deseable y atractiva a los clientes, se ha creado un logo y una imagen de la aplicación moderna y con colores llamativos que se han aplicado a toda la aplicación. La accesibilidad es un punto muy complejo a tratar, ya que para que una aplicación se considere accesible tiene que cumplir con unas estrictas normas, por lo que este punto será difícil de lograr debido al tiempo y recursos con los que cuenta el proyecto. Por último, el valor que ofrece este proyecto, como ya se ha mencionado anteriormente, están relacionados con el valor social, en contraposición al económico que se podría encontrar en otros productos.

4.2.2. Diseño previo

A la hora de hacer un diseño previo se tuvieron en cuenta los factores explicados anteriormente; y, aunque no sería el diseño definitivo, ya que a la hora de programar se pueden encontrar limitaciones, o el resultado puede no llegar a ser el deseado, sirve como guía para tener claros los objetivos y no perder factores claves.

Tal y como podemos observar en las siguientes imágenes, las aplicaciones están claramente diferenciadas por colores opuestos, esto facilita al usuario a diferenciar claramente en qué aplicación se encuentra, sin dejar de lado los colores corporativos. Como se puede observar esta diferenciación en el tema de las aplicaciones es completamente coherente, ya que los colores se intercambian entre sí en toda la aplicación.

Estos diseños son muy útiles para que el cliente pueda tener una imagen más o menos precisa de cómo será el producto final, y aunque este pueda sufrir cambios, se pueden definir unas líneas claras de lo que se busca y lo que no, y así poder ahorrar tiempo y confusiones a la hora de la entrega final.

Este diseño previo representa parte del flujo de uso del usuario, es decir, los pasos que debe de realizar el usuario para llegar a las diferentes secciones de la aplicación. Las primeras seis capturas (Fig 4.1.) pertenecen a la aplicación utilizada por los usuarios externos de la protectora, y las seis posteriores (Fig 4.2.) a la de la aplicación utilizada por las voluntarias de la aplicación.

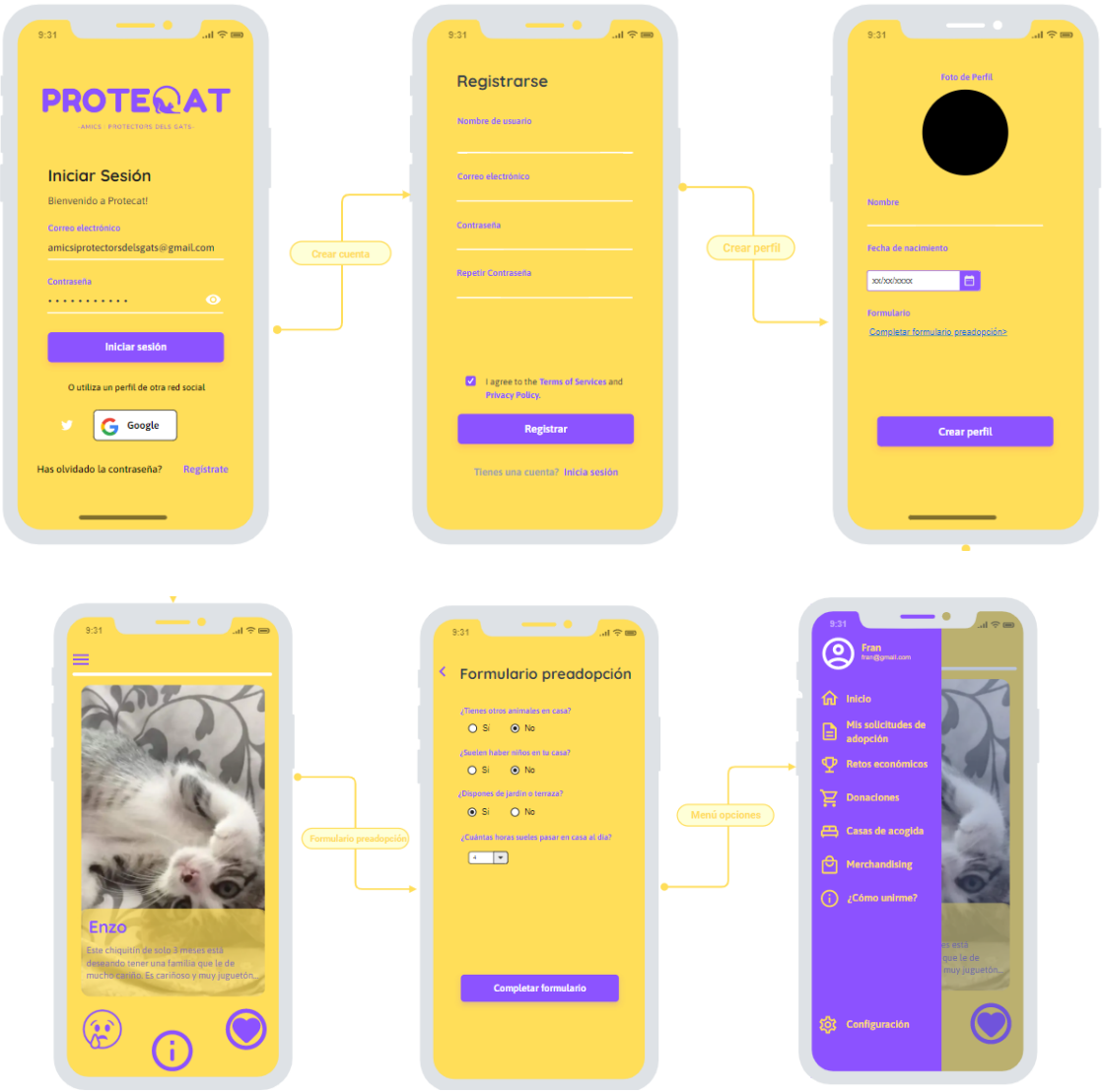

Fig 4.1. Diseño previo aplicación usuarios externos

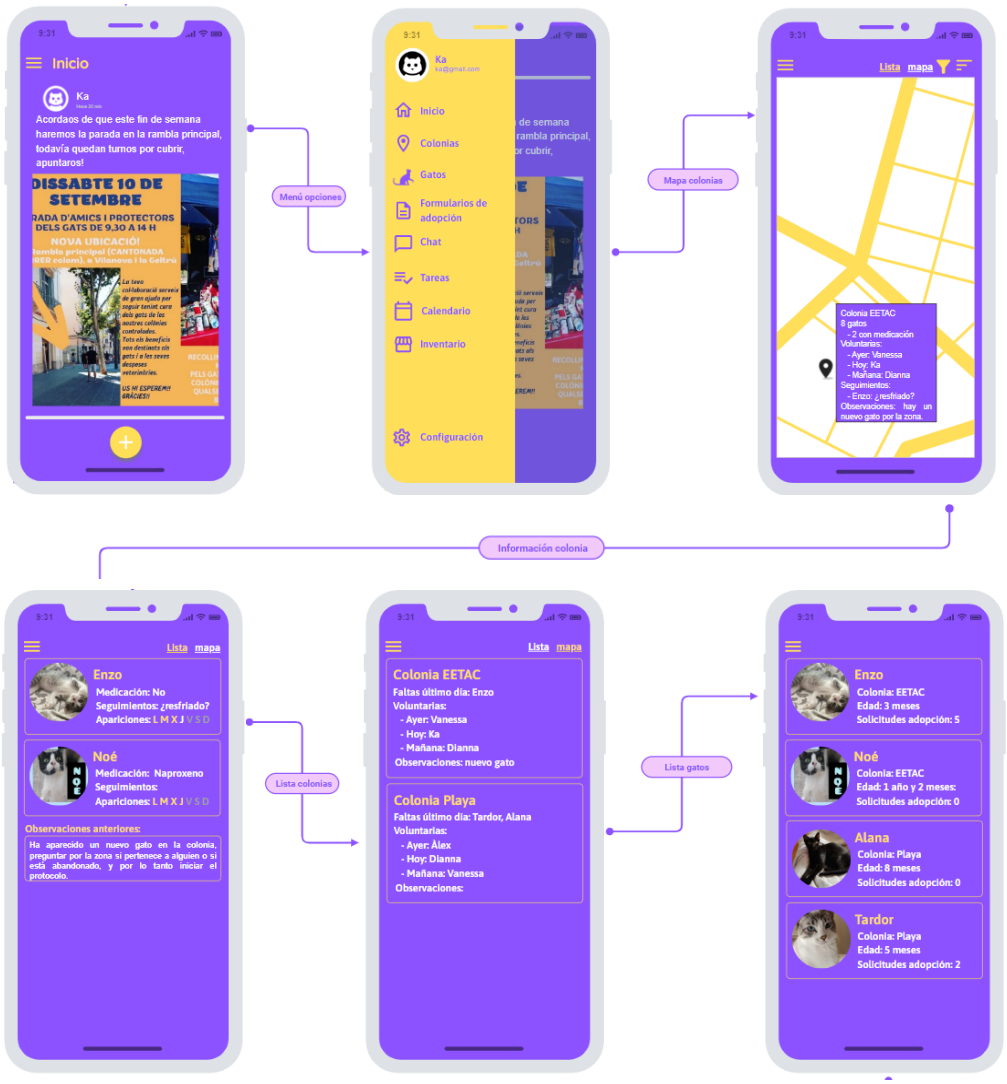

Fig 4.2. Diseño previo aplicación usuarios protectora

4.3. Backend

En cuanto al diseño del *backend*, no tenemos en cuenta aspectos visuales, ya que no cuenta con una interfaz gráfica, sino que hay que dibujar cuál será el esquema que se utilizará y qué servicios externos serán utilizados para llevar a cabo las funcionalidades de la aplicación. Posteriormente se analizará la estructura de datos de la aplicación.

4.3.1. APIs

Una API (Application Programming Interface) es una herramienta que permite conectar dos mecanismos de software, para así, transmitir información entre sí y ejecutar acciones. Es decir, una API permite invocar a servicios externos para poder obtener información o realizar acciones de esta. Hay que diferenciar dos partes, el cliente (quien hace la petición de la información) y el servidor (el que responde a la petición del cliente).

En el proyecto se encuentran dos tipos de peticiones, las que se realizan al *backend* propio del proyecto, y las que se realizan a APIs externas. Las principales APIs externas son:

- **Google:** que nos servirá para poder autenticarnos con una cuenta de esta organización, y realizar la función de ubicar las colonias en el mapa. Como se ha comentado anteriormente, el contar con un proveedor conocido otorga una fiabilidad al cliente, haciendo que se sienta seguro.

- **Cloudinary[25]:** esta API permitirá manejar todos los contenidos multimedia, por lo que permitirá ahorrar espacio local (ya que ubica todos los datos en una nube) y poder editar las imágenes de manera automática para poder adaptarlas a la aplicación.

- **JWT[35]:** JSON Web Token (JWT) es un estándar que define los mecanismos necesarios para transmitir información de manera segura. Este estándar nos permite encriptar y desencriptar la información (protegiendo así la información enviada). De esta manera los datos vulnerables de los usuarios pueden ser guardados y tratados con seguridad, así como verificar que el usuario es correcto y generar un token, pudiendo acceder a la aplicación durante un tiempo determinado sin necesidad de volver a autenticarse. Este estándar trabaja con otras dependencias como bcrypt (para encriptar y desencriptar los datos).

- **Mongoose[36]:** esta dependencia permite conectar la base de datos de *MongoDB* con el backend, de esta manera se puede solicitar la información necesaria en cada función. Esta librería también cuenta con la herramienta *populate*, que permite buscar elementos dentro de un elemento, así como filtrar los resultados de la base de datos; esta herramienta será muy útil para reducir el número de peticiones a la base de datos, aumentando así la eficiencia del código.

4.3.2. Endpoints

Tal y como se ha comentado con anterioridad, el *backend* consiste en la lógica de la aplicación, y para que estas acciones se puedan llevar a cabo deben de estar invocadas por el usuario, que puede enviar datos o no. Por lo que las funciones contendrán, o no, una entrada facilitada por el usuario, realizarán una acción, y después devolverá al usuario, o no, una respuesta (Fig 4.3.).

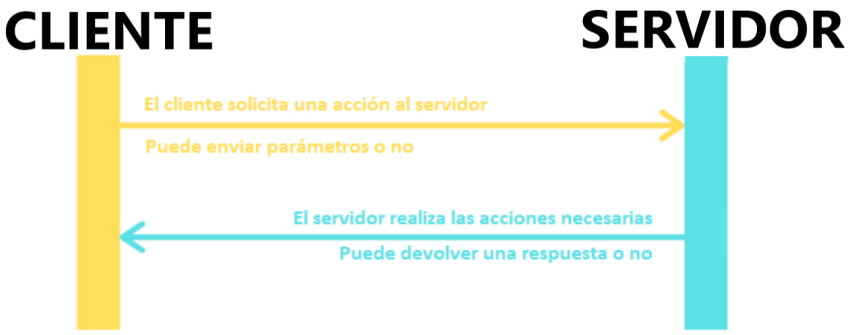

Fig 4.3. Flujo de acción llamada a servidor/*backend*.

El acrónimo *CRUD* define las funcionalidades de este apartado, las siglas se refieren a "*Create, Read, Update* and *Delate*", es decir, las cuatro funciones básicas en bases de datos. Por lo tanto, el usuario podrá crear, acceder, modificar y eliminar la información de los diferentes ítems (usuarios, gatos, colonias, etc.) de la base de datos. Estas son las funciones básicas que realizará esta parte de la aplicación para cada clase, exceptuando en algunos casos que serán detallados más adelante. Para llevar a cabo estas funcionalidades se necesitarán diferentes acciones que estarán agrupadas en *endpoints*, que son las direcciones (*URLs*) del backend a las que llamará el usuario a través del *frontend,* para así, poder acceder o modificar la información necesaria. Estos *endpoints* suelen realizar una sola acción en la base de datos, como se puede observar en la figura (Fig 4.4.), en la que se realiza una única búsqueda en la base de datos para obtener el gato que cuente con el ID que el usuario ha pasado como parámetro desde la aplicación.

```
function getCat (req:Request, res:Response): void {
    //Primero se busca el gato por el ID enviado (req.params.id)
    cat.findOne({"id":req.params.id}).then((data)=>{
        //Por defecto iguala el status a 200, que significa que la búsqueda ha sido correcta
        let status: number = 200;
        //Si no se encuentra el gato (data == null), iguala el status a 404, que indica que no lo ha encontrado
        if(data==null) status = 404;
        return res.status(status).json(data);
    }).catch((err) => {
        //Si la búsqueda ha desencadenado un error devuelve al usuario un código 500.
            return res.status(500).json(err);
    })
}
```

Fig 4.4. *Endpoint* para obtener un gato según su ID.

La mayoría de las funciones son similares a esta, pero cambiando el método, en vez de buscar (función *findOne*), actualiza el elemento indicado por el usuario con los datos facilitados (función *update*, que busca el elemento por el ID y modifica los datos facilitados), aunque hay algunas excepciones en las que se requiere realizar más de una búsqueda u otras acciones.
Una de estas excepciones se desarrolla en la función de obtener las tareas de un usuario (Fig 4.5.), en la cual se realizan dos búsquedas: primero la del usuario con el ID facilitado por el usuario, y después la de las tareas del usuario

encontrado. Esta herramienta llamada *populate* reduce el número de búsquedas y por lo tanto mejora la eficiencia del código.

```
function getTasks (req:Request, res:Response): void {
    //Buscamos al usuario por su ID, y después obtenemos las tareas de este usuario
    User.findById(req.params.id).populate('tasks').then((data)=>{
        //Por defecto se devuelve un código 200 indicando que la búsqueda ha sido correcta
        let status: number = 200;
        //En caso de que no se encuentre ninguna tarea del usuario se refiere con un código 404
        if(data==null) status = 404;
        return res.status(status).json(data);
    }).catch((err) => {
        console.log(err);
        //En caso de que suceda un error se notifica con un código 500
        return res.status(500).json(err);
    })
}
```

Fig 4.5. *Endpoint* para obtener las tareas de un usuario

Otro ejemplo en el que el *endpoint* no realiza solo una consulta a la base de datos es la función de *Login* (Fig 4.6.), la cual es llamada cuando el usuario inicia sesión en la aplicación. En este caso, la función primero comprueba que el correo facilitado por el usuario es correcto, verificando que se encuentra en la base de datos; una vez realizada esta comprobación revisa que la contraseña introducida sea correcta. Para realizar esto, primero encripta la contraseña facilitada por el usuario, y la compara con la guardada en la base de datos (ya que en el momento del registro la contraseña es encriptada para proteger la información del usuario); en caso de que las credenciales coincidan se genera un token de autenticación y se envía al usuario este token, una confirmación de que la validación ha sido realizada correctamente, y los datos del usuario. Como se puede observar, esta función es compleja y depende de otras funciones dentro del mismo backend, si cada parte de esta función se hubiese implementado con un único *endpoint*, el usuario debería de haber realizado tres llamadas al servidor, reduciendo la eficiencia.

```
function LogIn (req:Request, res:Response): void {

    let body = req.body;
    //erro y usuarioDB any(?)

    User.findOne({ email: body.email }, async (erro: any, userDB: { password: any; })=>{
        if (erro) {
            return res.status(500).json({
                ok: false,
                err: erro
            })
        }
    // Verifica que exista un usuario con el mail escrita por el usuario.
        if (!userDB) {
            return res.status(401).json({
                ok: false,
                err: {
                    message: "Usuario o contraseña incorrectos"
                }
            })
        }

        const matchPassword = await bcrypt.compare(body.password, userDB.password)
    // Valida que la contraseña escrita por el usuario, sea la almacenada en la db
        if (!matchPassword){
            return res.status(400).json({
                ok: false,
                err: {
                    message: "Usuario o contraseña incorrectos"
                }
            });
        }
    // Genera el token de autenticación
        let token = jwt.sign({
                user: userDB,
            }, process.env.SEED_AUTENTICACION, {
            expiresIn: process.env.CADUCIDAD_TOKEN
        })
        res.json({
            ok: true,
            user: userDB,
            token,
        })
    })
}
```

Fig 4.6. *Endpoint* para iniciar sesión

4.4. Backoffice

Como se ha mencionado anteriormente, *backoffice* es una herramienta de administración de la información de la aplicación que cuenta con una interfaz gráfica, pero el objetivo de esta parte del proyecto no es contar con un diseño atractivo, sino contar con un diseño funcional y simple que permita gestionar datos como las cuentas de usuarios o los elementos creados en la base de datos de una manera gráfica y rápida. Por lo tanto, aunque sea una herramienta visual que varios usuarios van a utilizar, no hay que tener en cuenta todos los parámetros definidos anteriormente, sino que hay que centrarse en que solvente el problema principal de una manera clara y concisa.

Por lo que si se recupera la lista de parámetros, se escogerían la utilidad, usabilidad, encontrabilidad y valor como objetivos principales de diseño, que como se puede observar, son todos aquellos parámetros centrados en el uso.

Al igual que en el apartado anterior (*Backend*), las funcionalidades realizadas se basan en el *CRUD*. Este apartado es muy útil para que la administradora pueda consultar todos los datos creados en la aplicación, y pueda modificarlos, así como crear algún dato de manera rápida y sencilla, o eliminar algún parámetro que no sea correcto o necesario. Aunque la mayoría de elementos se pueden crear, modificar o eliminar mediante la aplicación, esta manera es mucho más sencilla, además de poder eliminar usuarios (parte que no se puede realizar mediante la aplicación), útil para aquellos usuarios que ya no forman parte de la protectora o que no se desea que tengan acceso.

Del mismo modo que se ha hecho con el diseño de *Frontend*, se ha realizado un diseño previo con tal de poder mostrar al cliente final una posible visión del resultado final (Fig 4.7.). En el ejemplo de la figura se muestran los datos de los gatos, donde se pueden editar, eliminar o crear datos sobre los animales.

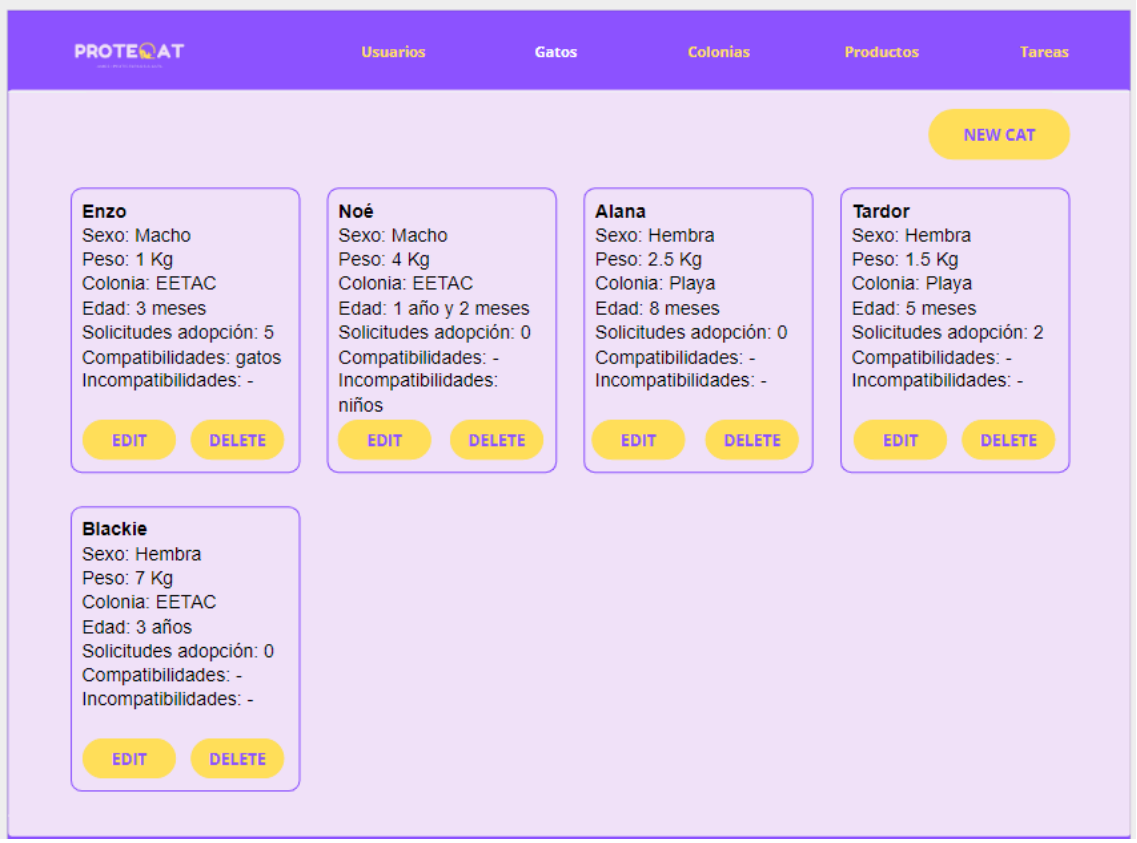

Fig 4.7. Diseño previo *backoffice*

Resumen:

En este apartado se describen los parámetros básicos para diseñar una aplicación y que sea atractiva para el usuario final, así como los criterios que se han utilizado para diseñar las diferentes secciones del proyecto (*frontend, backoffice* y *backend*), y unos diseños previos de cómo sería el producto final. Además, se explican los diferentes tipos de funcionalidades que se pueden encontrar en *backend* dependiendo de su complejidad, así como un ejemplo de uso de la aplicación y una descripción de todas las funcionalidades de las aplicaciones.

CAPÍTULO 5: MODELOS DE DATOS

Tras haber definido las funcionalidades y haber dibujado las futuras aplicaciones se debe de definir el modelo de datos y las relaciones que tendrán entre los diferentes parámetros.

5.1. Base de datos

El primer paso para definir el modelo de datos del proyecto consiste en identificar el tipo de base de datos con el que se trabajará. En este caso se tratará de una base de datos NoSQL[37], siglas de *Not Only SQL*, es decir, que utiliza el lenguaje SQL, pero no siempre. Una de las mayores características de este tipo de base de datos es que la base de datos no es relacional (los datos no están relacionados entre sí), ofreciendo una mayor escalabilidad y rendimiento. Otras de sus ventajas son:

- Capacidad de soportar un gran volumen de datos, tanto estructurados como no, en constante cambio, gracias a su carácter descentralizado.

- Flexibilidad, permitiendo adaptarse a las frecuentes entregas de la metodología ágil. Al no definir una estructura para cada tipo de dato, cada elemento cuenta con unos parámetros diferentes sin necesidad de definirlo, pudiendo así realizar cambios sin interrupciones.

- Arquitectura horizontal, permitiendo crecer y ejecutar la base de datos en diferentes máquinas, utilizando así una pequeña parte de los recursos de estas. Esto es posible gracias al *auto-sharding*, que distribuye los datos entre diversos servidores, pudiendo balancear la carga entre ellos, y teniendo una rápida respuesta ante fallos de servidores.

5.2. Diagrama entidad-relación

En este apartado se puede observar el diagrama entidad-relación de las diferentes clases de la base de datos. Este diagrama muestra los diferentes parámetros de cada clase, así como la relación que existe entre las diferentes entidades, que serán explicadas posteriormente.

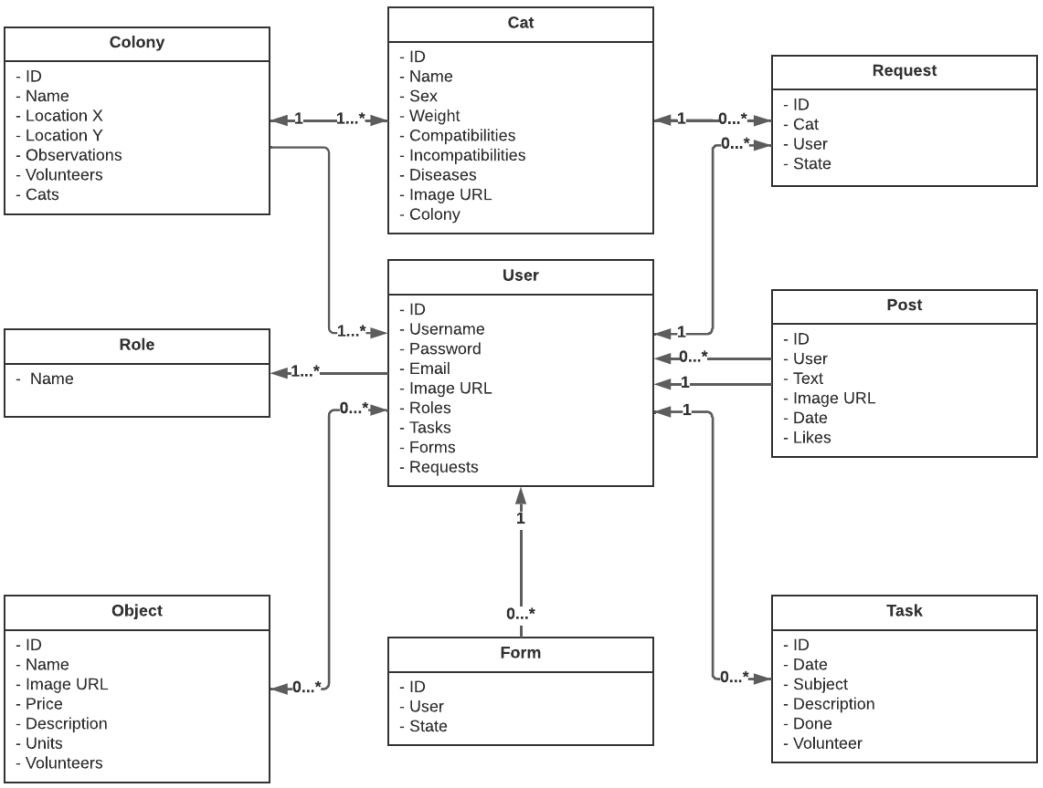

Fig 5.1. Diagrama entidad-relación

5.3. Estructura

En este punto se tratará de conocer la funcionalidad de cada una de las entidades, así como sus parámetros. Además, se explicarán las relaciones entre ellas.

- **Usuario:** esta entidad sirve para identificar a la persona que va a utilizar la aplicación. Cuenta con los siguientes parámetros:

    - ID - ObjectId: identificador único para cada usuario, es uno de los parámetros utilizados para encontrar a los diferentes usuarios. Este se genera de manera automática por la base de datos. Este valor es el que se asigna al resto de entidades que cuentan con un usuario. El valor de este parámetro es único.

    - Username - String: nombre de usuario que será visible para los diferentes trámites que realice.

    - Password - String: contraseña que utiliza el usuario para iniciar sesión y registrarse en la aplicación. Este parámetro se encripta antes de ser guardado en la base de datos para que el valor no sea visible en la base de datos, y así quede protegida la intimidad del usuario.

- Email - String: correo electrónico del usuario, que consiste en el otro parámetro utilizado por el usuario para iniciar sesión. Este valor debe de ser único, por lo que no puede haber más de un usuario con el mismo valor. Este campo también será utilizado para ponerse en contacto con el usuario a la hora de realizar cualquier trámite.

- ImageUrl - String: enlace con la imagen de perfil del usuario. El enlace corresponde a la imagen subida por el usuario en *Cloudinary*, la API utilizada para almacenar todos los contenidos multimedias.

- Roles - Rol (Object ID): como en el resto de parámetros en el que el tipo de datos es una clase, se guarda el valor del identificador generado por la base de datos (Object ID), de esta manera la base de datos solo tiene que buscar este parámetro para ver todos los datos. Este campo se utiliza para conocer el rol de cada usuario, y así se permita o prohíba el acceso a las diferentes secciones de la aplicación.

- Tasks - Tarea (Object ID): hace referencia a las tareas que el usuario tiene asignadas.

- Forms - Formulario (Object ID): hace referencia a los diferentes formularios que el usuario ha generado.

- **Colonia:** incluye las diferentes colonias de gatos que la protectora administra.

  - ID - ObjectId.

  - Name - String: nombre por el que se conoce a la colonia.

  - Location X - String: consiste en la latitud de las coordenadas de la colonia. Se utiliza para poder colocar el marcador de la localización de la colonia en el mapa.

  - Observations - String: campo que sirve para anotar cualquier hecho remarcable ocurrido en la colonia, para que la siguiente voluntaria pueda tener algo más de información, o revise algún punto concreto.

  - Volunteers - Usuario (Object ID): voluntaria/s asignada/s a la colonia.

  - Cats - Gato (Object ID): gatos que forman parte de la colonia.

- **Gato:** referencia a los animales pertenecientes a la protectora.
    - ID - ObjectId.
    - Name - String: nombre del gato.
    - Sex - String: sexo del animal, que será útil para los usuarios que quieran adoptarlo.
    - Weight - String: campo que denota el peso del animal.
    - Compatibilities - String: lista de compatibilidades con las que cuenta el animal. Este parámetro será muy útil para que el usuario que quiera adoptar al gato sepa si se adapta a su estilo de vida.
    - Incompatibilities - String: lista de incompatibilidades con las que cuenta el animal. Al igual que el anterior, este parámetro será muy útil para que el usuario que quiera adoptar al gato sepa si se adapta a su entorno. Además, sirve como filtro para que si algún usuario no se adapta a las necesidades del animal, éste no continúe en el proceso de selección, y así se puedan agilizar los trámites.
    - Diseases - String: este campo indica si el animal padece alguna enfermedad, dato que puede ser interesante para el posible adoptante, por si cuenta con otros animales con la misma patología.
    - ImageUrl - String: enlace con la imagen del gato.
    - Colony - Colonia (Object ID): hace referencia a la colonia a la que el gato pertenece.
- **Solicitud:** esta entidad guarda las solicitudes de adopción de los usuarios hacia los diferentes gatos.
    - ID - ObjectId.
    - Cats - Gato (Object ID): gato que se quiere adoptar.
    - User - Usuario (Object ID): usuario que solicita la adopción del animal.
    - State - String: estado de la solicitud. Este parámetro puede ser: “Pendiente de aprobación”, “Concedida”, “Denegada”.
- **Rol:** define el rol de cada usuario. Dependiendo del valor de esta entidad, el usuario podrá acceder a más o menos funcionalidades de la aplicación.

    - Name - String: define el rol. Puede ser: “estándar” o “administradora”.

- **Publicación:** post que las voluntarias cuelgan en el muro y que aparece tanto en la aplicación de los usuarios externos, como a los de la protectora.

    - ID - ObjectId.

    - User - Usuario (Object ID): voluntaria que ha generado el post.

    - Text - String: texto que aparece en la publicación.

    - ImageUrl - String: enlace a la posible imagen que la voluntaria haya adjuntado al post.

    - Date - String: fecha en la que se realiza la publicación.

    - Likes - Usuario (Object ID): usuarios que han dado me gusta al post.

- **Objeto:** recursos con los que cuenta la protectora. Esta clase se utiliza tanto en la aplicación de las voluntarias, para realizar un inventario de los artículos disponibles, como en la aplicación para los usuarios externos a la asociación, para poder donar el dinero equivalente a los objetos seleccionados.

    - ID - ObjectId.

    - Name - String: nombre del objeto.

    - ImageUrl - String: enlace a la imagen del producto.

    - Price - Number: precio aproximado del objeto.

    - Description - String: descripción del uso que se le da al objeto dentro de la protectora.

    - Units - Number: Número de unidades del producto con los que cuenta la protectora.

    - Volunteers - Usuario (Object ID): identificador de la/s voluntaria/s a la/s que se le ha asignado el artículo.

- **Formulario:** esta entidad guarda los formularios enviados por los usuarios externos de la protectora para unirse a esta.

    - ID - ObjectId.

    - User - Usuario (Object ID): usuario que solicita la inserción en la protectora.

    - State - String: estado del formulario. Este parámetro puede ser: “Pendiente de aprobación”, “Concedida”, “Denegada”.

- **Tarea:** entidad que indica las tareas creadas, que pueden estar asignadas o no a una voluntaria de la protectora.

    - ID - ObjectId.

    - Date- String: fecha límite o asignada para la tarea.

    - Subject - String: asunto para identificar la tarea.

    - Description - String: descripción de la tarea, en la que se puede explicar los objetivos de esta.

    - Done - Boolean: Indica si la tarea ha sido realizada o no. Este parámetro solo puede ser modificado por la persona a la que ha sido asignada la tarea.

    - Volunteers - Usuario (Object ID): identificador de la/s voluntaria/s a la/s que se le ha asignado la tarea.

Tal y como se muestra en el diagrama (Fig 5.1.), las entidades cuentan con diferentes relaciones entre ellas:

- **Colonia**
    - Gato: cada colonia puede contar con uno o más gatos.
    - Usuario: cada colonia puede contar con una o más voluntarias.

- **Gato**
    - Colonia: cada gato solo puede pertenecer a una única colonia.
    - Solicitud: cada gato puede contar con varias o ninguna solicitud de adopción.

- **Solicitud**
    - Gato: cada solicitud solo puede pedir la adopción de un gato.
    - Usuario: cada solicitud solo puede estar pedida por un usuario.

- **Usuario**
    - Rol: cada usuario puede contar con uno o más roles, dependiendo de su papel en la protectora.
    - Solicitudes: cada usuario puede realizar ninguna o varias solicitudes de adopción hacia diferentes gatos.
    - Objeto: cada usuaria (voluntaria de la protectora) puede tener diferentes objetos, y más de una unidad de cada uno de ellos.
    - Tarea: un usuario puede tener ninguna o diversas tareas.

    - Formulario: un usuario puede tener ninguna o diversos formularios asignados.

- **Post**
    - Usuario: esta entidad cuenta con dos relaciones con la clase usuario, ya que se realiza una asignación a un único usuario en referencia al parámetro *User*, que se refiere al creador del post; y una asignación de ninguno o varios usuarios en el parámetro *Likes*, referente a los usuarios que han reaccionado al post publicado.

- **Objeto**
    - Usuario: en este caso un mismo objeto puede estar asignado a uno o más usuario, ya que se puede contar con múltiples unidades de cada uno.

- **Formulario**
    - Usuario: cada formulario solo puede estar creado por un único usuario.

- **Tarea**
    - Usuario: una tarea solo puede estar asignada a un usuario o a ninguno.

Resumen:

En esta parte del proyecto se han descrito las entidades con las que cuenta la base de datos, y que por lo tanto, son necesarias para poder llevar a cabo las diferentes acciones requeridas. Además, se han explicado las relaciones entre cada una de ellas.

CAPÍTULO 6: IMPLEMENTACIÓN

Tras conocer la estructura de datos, se definirán aquellas funcionalidades que han representado un reto en su desarrollo o planteamiento, y cuáles han sido las soluciones llevadas a cabo, así como un ejemplo de uso de la aplicación.

6.1. Mapa de uso

En el siguiente apartado se representará un mapa de uso de la aplicación, es decir, los diferentes pasos que deberá llevar a cabo el usuario para realizar las funcionalidades del aplicativo. De esta manera se podrán conocer todas las acciones disponibles en el proyecto.

6.1.1. Frontend

En este punto se pueden observar las diferentes funcionalidades del proyecto, ordenadas de manera que simula los pasos que realiza el usuario para llegar a cada una de las pantallas.

Al iniciar la aplicación, el usuario dispone de una pantalla inicial hasta que se cargan los datos necesarios para inicializar la aplicación, después se encuentra con una pantalla donde se le ofrece la opción de crear una nueva cuenta o iniciar sesión mediante una cuenta ya creada (Fig 6.1.). Una vez el inicio de sesión se ha realizado correctamente, se dirige al usuario hacia un mapa ubicado en la zona donde se encuentran las colonias, desde esta pantalla el usuario puede ver los detalles de cada colonia, añadir comentarios o gatos a esa colonia (Fig 6.2.). Si el usuario pulsa sobre el icono de tres líneas situado arriba a la izquierda, puede acceder al menú lateral desde donde podrá modificar su imagen de perfil u otra información (Fig 6.3.). Al pulsar sobre la sección de “Foro”, el usuario accede a un muro donde aparecerán sus publicaciones creadas, al igual que las de sus compañeras, estas noticias son las que se verían en la aplicación para usuarios externos a la protectora (Fig 6.4.). Desde el menú lateral el usuario puede acceder a la lista de todos los gatos que cuida la protectora, así como crear o modificar el perfil del animal (Fig 6.5.). Otra de las funcionalidades a las que puede acceder la voluntaria es la lista de tareas, donde se pueden crear nuevas tareas y filtrar dependiendo de si están asignadas, si no lo están o si pertenecen al usuario que ha iniciado sesión; en caso de que las tareas no estén asignadas el usuario puede asignarlas, o marcar como realizadas en caso de que estén asignadas a su usuario. Además, puede acceder a un calendario en el que ver todas las tareas (Fig 6.6.). Finalmente la aplicación cuenta con una sección en la que visualizar los diferentes artículos con los que cuenta la protectora, así como añadir o eliminar unidades de cada producto (Fig 6.7.).

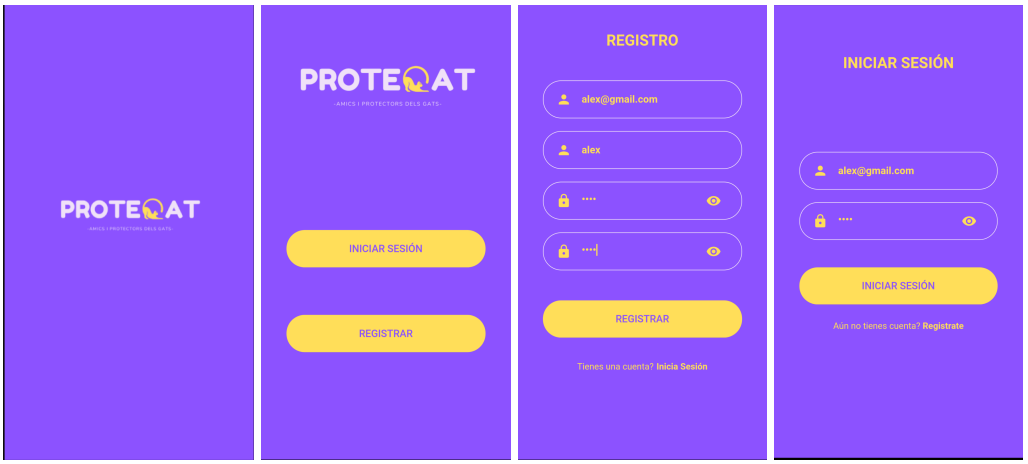

Fig 6.1. Pantalla inicial, inicio sesión y registro

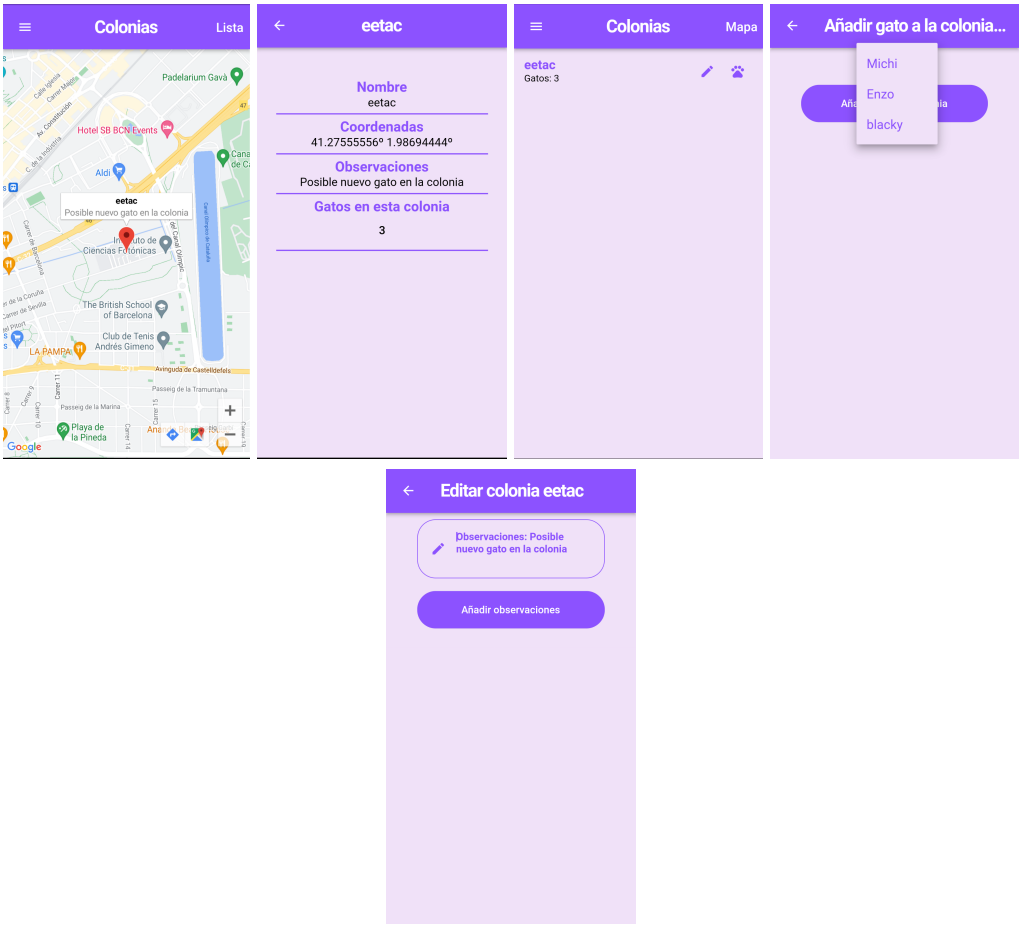

Fig 6.2. Pantallas relacionadas con las colonias

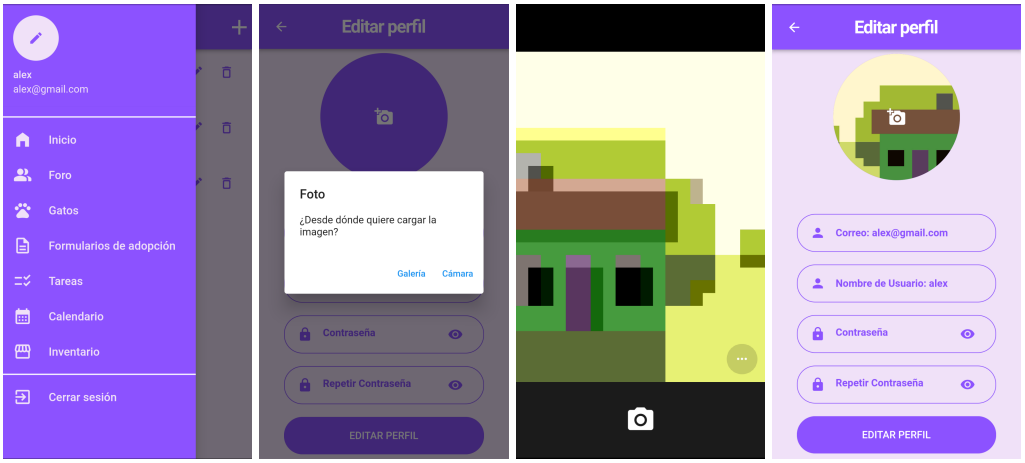

Fig 6.3. Barra lateral y modificación de perfil

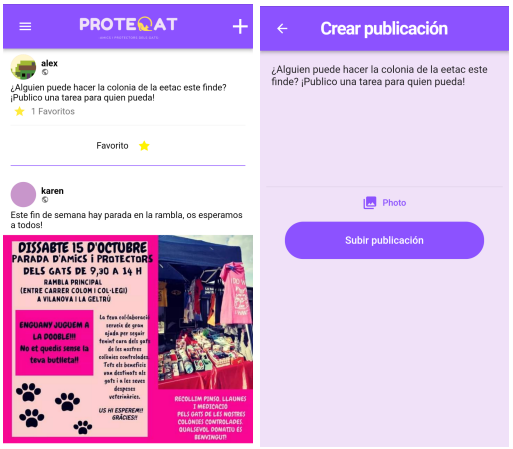

Fig 6.4. Muro noticias y creación post

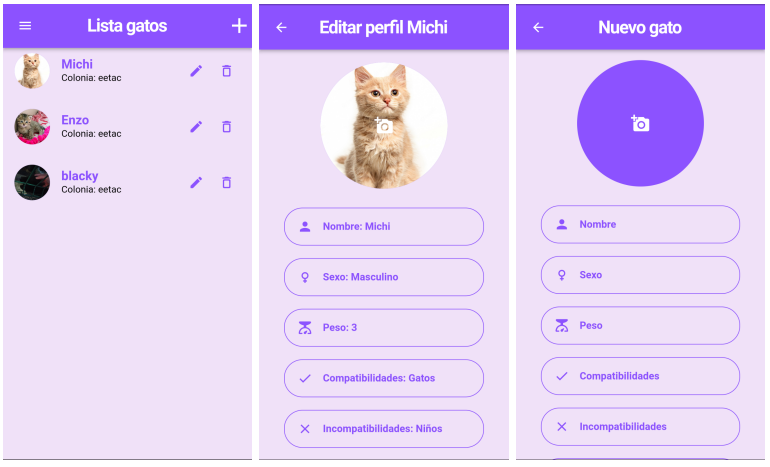

Fig 6.5. Pantallas relacionadas con los gatos

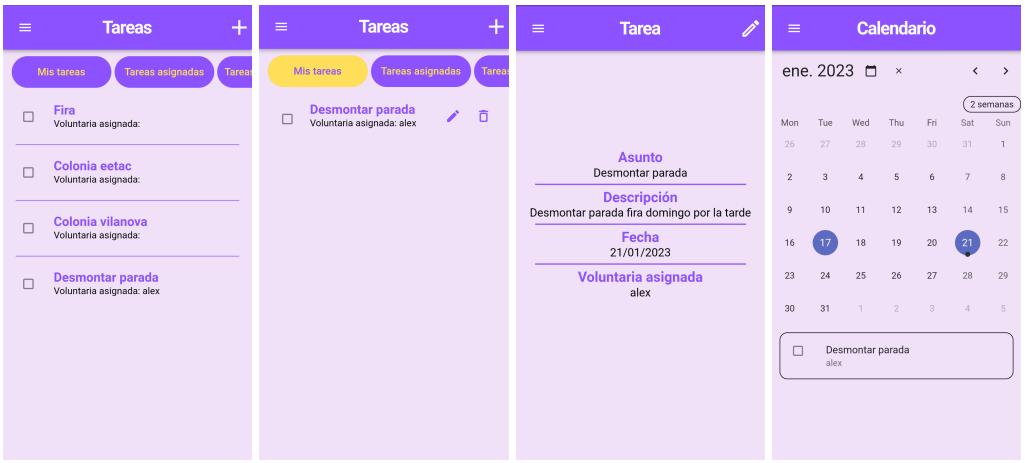

Fig 6.6. Pantallas relacionadas con las tareas

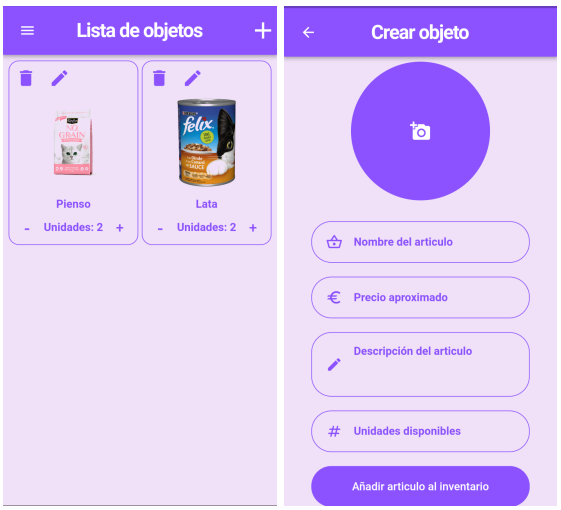

Fig 6.7. Pantallas relacionadas con los objetos

6.1.2. Backoffice

En el siguiente punto se mostrará el aplicativo correspondiente a *Backoffice*, tal y como se ha comentado en apartados anteriores, se trata de una página web para tratar los datos de las diferentes clases de una manera más visual. Tal y como se muestra en las figuras, para acceder, la administradora primero debe de iniciar sesión para corroborar que cuenta con los permisos para realizar ciertas acciones (Fig 6.8.). Si el inicio de sesión es correcto, accede a la pantalla inicial, donde se disponen las diferentes cuentas de usuarios, con el correo de este (Fig 6.9.). Desde esa pantalla el usuario puede escoger si crear un nuevo usuario (Fig 6.10.), modificar un usuario existente (Fig 6.11.) o visualizar con más detalle el perfil del usuario (Fig 6.12.).

Las imágenes corresponden a las funciones relacionadas con los usuarios, se han implementado las mismas acciones para el resto de clases (productos, colonias y gatos).

Fig 6.8. Pantalla inicio sesión

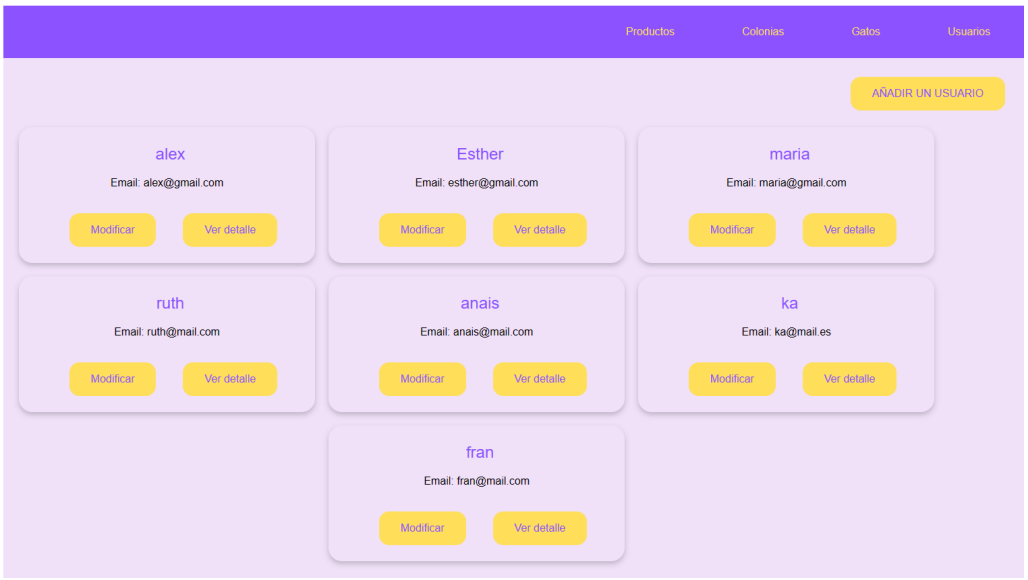

Fig 6.9. Pantalla visualización resumen datos

Fig 6.10. Pantalla creación usuario

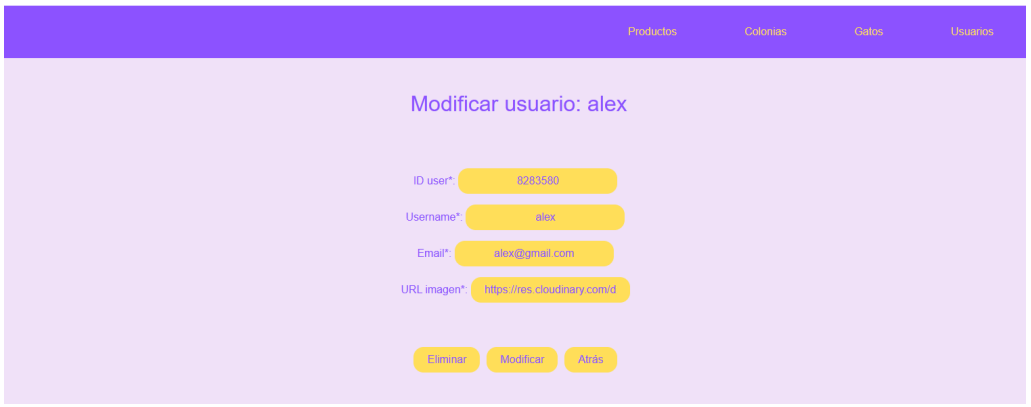

Fig 6.11. Pantalla modificación usuario

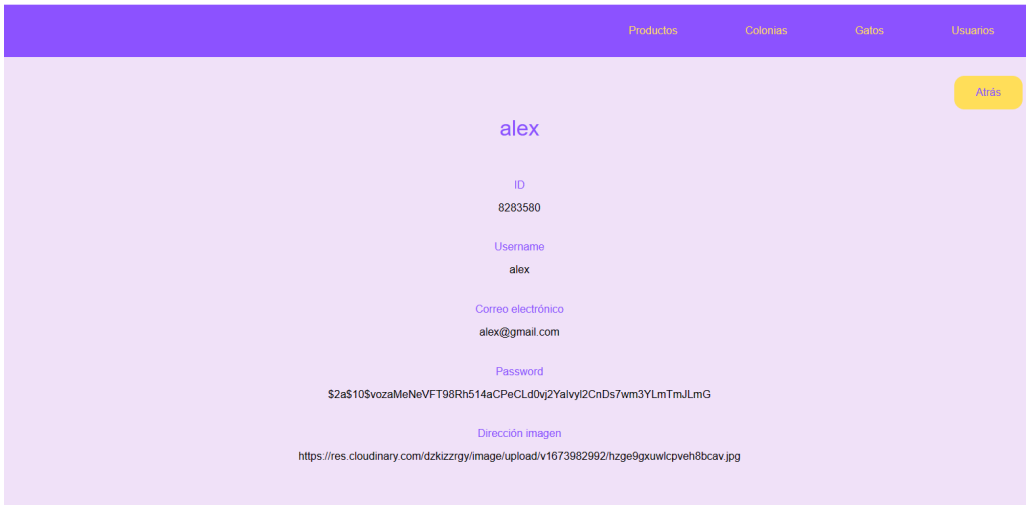

Fig 6.12. Pantalla detalle usuarios

6.2. Dificultades

En este apartado se mostrarán aquellas funcionalidades que han representado un reto a la hora de desarrollarlas y diseñarlas.

6.2.1. Frontend

Recuperando la lista de funcionalidades descritas en el apartado 4.1.1 podemos destacar las siguientes funcionalidades como complejas, o con un desarrollo algo más avanzadas que el resto.

6.2.1.1. Mapa de colonias

Una de las funcionalidades claves del proyecto consiste en el seguimiento de las colonias mediante un mapa, desde dónde se puede consultar la información relevante, o acceder al detalle de cada una de ellas. Durante el desarrollo de esta funcionalidad se han sufrido diversos problemas. El primero de ellos estaba relacionado con los permisos de ubicación.

- **Permisos de ubicación:** al intentar acceder al mapa donde localizar todas las colonias, este requería de unos permisos especiales que debían de efectuarse desde el móvil. Estos permisos eran necesarios para consultar la localización del usuario y así poder iniciar el mapa desde ese punto. Dada esta problemática, se propusieron dos opciones: cada vez que el usuario accediera a la página del mapa consultar si había aceptado los permisos de localización, y en caso de haberlo hecho, acceder al mapa; o no contar con la funcionalidad de geolocalizar al usuario. Tras tratar analizar ambas opciones, se llegó a la conclusión de que no era necesario que la aplicación contase con la ubicación exacta de la voluntaria, ya que las diferentes ubicaciones de las colonias son cercanas entre sí y las voluntarias saben exactamente la ubicación de estas. De esta manera, además, se evita dar información delicada, obteniendo así una mayor confianza del usuario.

Otro de los problemas encontrados en el desarrollo de esta funcionalidad fue la creación de colonias mediante el mapa.

- **Creación de colonias:** el proceso de crear una colonia y seleccionar la ubicación de esta desde el mapa requería una funcionalidad algo compleja y delicada. Por lo que, tras varios intentos, y tras analizar el valor aportado a la aplicación, se desestimó la realización de esta. Esta decisión fue tomada por el poco número de adhesiones de colonias que se realizan a lo largo del año, ya que el número de colonias son muy estables, prácticamente sin modificaciones desde hace años. Por lo que tras evaluar el valor obtenido con los recursos necesarios, se decidió no llevarla a cabo.

Por último, debido al *widget* requerido para realizar el mapa, no era posible pasar de pantalla cargando un contexto, que es la manera habitual de hacerlo. Por lo que se tuvo que buscar una alternativa.

- **Pasar a otra pantalla:** el *widget* y el tipo de clase utilizados, no era posible generar un contexto y pasarlo a la siguiente pantalla, ya que los parámetros debían de ser definidos antes de poder declarar el contexto. El poder pasar de pantalla era una función necesaria, ya que el usuario tenía que poder cambiar de pantalla y así poder ver el detalle de la colonia y realizar acciones sobre esta. Por lo que se intentó realizar algún cambio y poder declarar el contexto antes y enviarlo a la siguiente pantalla, pero no se consiguió, por lo que se buscó otra alternativa.
  La alternativa encontrada fue pasar de pantalla sin enviar contexto. Para poder realizar este cambio era necesario utilizar el paquete de *flutter*,

"*no_context_navigation*", este paquete permite navegar a páginas definidas al inicio de la aplicación y navegar sin crear contexto.

```
navigatorKey: NavigationService.navigationKey,
onGenerateRoute: (RouteSettings settings) {
  switch (settings.name) {
    case '/ColonyDetail':
      return MaterialPageRoute(builder: (_) => DetailPage(colonySelected));
    default:
      return null;
  }
},
```

Fig 6.13. Declaración de rutas sin contexto

Como se puede observar en la figura (Fig 6.13.), se ha definido la ruta a la página que presenta los detalles de la colonia seleccionada. Para poder presentar la información de la colonia deseada, hay que pasar como parámetro esa colonia, por lo que se ha definido una variable global (*colonySelected*) cuyo valor se establece (Fig 6.14.) cuando el usuario pulsa sobre el resumen de información de la colonia (Fig 6.15.).

```
Future<void> getColonyByName(String name) async {
  final data =
      await http.get(Uri.parse('http://192.168.56.1:3000/colonies/getColonyByName/' + name));
  var u = json.decode(data.body);
  Colony colony = Colony(
      id: u["id"],
      name: u["name"],
      locationx: u["locationx"],
      locationy: u["locationy"],
      observations: u["observations"],
      cats: u["cats"]);

  colonySelected = colony;

}
```

Fig 6.14. Función asignación colonia

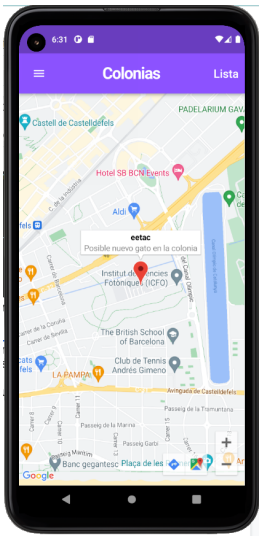

Fig 6.15. Tarjeta resumen colonia

6.2.1.2. Multilenguaje

Una funcionalidad que se desarrolló al inicio del proyecto y que no pudo persistir, fue la opción de cambiar el idioma de la aplicación. Esta funcionalidad fue pensada para participar en la causa de conciencia digital[38] impulsada por la Universidad Politécnica de Cataluña, que busca concienciar a los usuarios de los riesgos y oportunidades de las nuevas tecnologías, así como alentar a los desarrolladores a incluir estos parámetros a sus proyectos para alcanzar una digitalización segura, de calidad e inclusiva. En este marco, la funcionalidad de multilenguaje ayudaría a la inclusión de aquellas comunidades que hablen un idioma diferente. Esta herramienta iba a ser desarrollada en catalán, castellano e inglés, constituyéndose así como un gran valor a futuro, y dando acceso a personas que no hablen castellano. La funcionalidad se iba a llevar a cabo con la dependencia "i18n"[39] de *Flutter*, que permite crear diccionarios en distintos idiomas, y dependiendo del lenguaje con el que el terminal esté configurado, mostrará un idioma u otro. Esta funcionalidad se desarrolló y llegó a estar en funcionamiento, pero cuando se desarrolló el calendario en el que aparecen las tareas no eran compatibles, por lo que se decidió prescindir de la opción de disponer la aplicación en más de un idioma, ya que en la actualidad y en la historia de la protectora nunca se ha contado con voluntarias que no conozcan el español.

6.2.2. Backend

Para llevar a cabo las funcionalidades de la aplicación, es necesario crear diferentes funciones en la *API*, a las que se llamará *frontend*. En algunas ocasiones las operaciones realizadas en el *backend* requerían más de una simple acción en la base de datos.

6.2.2.1. Restringir acceso a los usuarios dependiendo de su rol

Como se ha comentado anteriormente (5.3.), para poder filtrar las acciones que puede realizar cada usuario, es necesario crear roles que indican las funcionalidades a las que tiene acceso cada usuario. Por lo tanto, en todas aquellas funciones a las que queremos blindar su acceso, es necesario consultar si el rol del usuario es “admin” (Fig 6.16.), en caso de que el rol del usuario que ha ejecutado la función, no se devolverá nada y podrá ejecutar la siguiente acción. En caso de que no cuente con permisos de administrador, se enviará un mensaje “403. Require Admin Role!”, avisando al usuario de que no cumple con los permisos necesarios. En caso de que no se encuentre el rol del usuario se enviará un código de error 500. Esta función será llamada siempre que se quiera consultar el rol del usuario. Para llamar a la función, solo hay que indicarlo en la ruta de la función (Fig 6.17.), y cada vez que el *frontend* haga una llamada http a la ruta indicada, ejecutará la función “*isAdmin*”, y en caso de que la respuesta sea satisfactoria, ejecutará la siguiente acción. El procedimiento para comprobar el rol del usuario es el mismo que para comprobar el token de autenticación del usuario, tal y como se observa en la figura (Fig 6.17.)

```
export const isAdmin = async (req, res, next) => {
  try {
    const user = await User.findOne({id: req.body.userId});
    const roles = await Role.find({ _id: { $in: user?.roles } });

    for (let i = 0; i < roles.length; i++) {
      if (roles[i].name === "admin") {
        next();
        return;
      }
    }

    return res.status(403).json({ message: "Require Admin Role!" });
  } catch (error) {
    console.log(error);
    return res.status(500).send({ message: error });
  }
};
```

Fig 6.16. Función comprobación rol usuario

```
import { Router } from "express";
import catController from "../controllers/cat.controller";
import { isAdmin, verifyToken } from "../middelware/authJWT";

const router = Router();

router.get('/',  catController.getAllCats);
router.get('/getCat/:id', [verifyToken], catController.getCat);
router.get('/getCatByName/:name', catController.getCatByName);
router.post('/new', [isAdmin], catController.newCat);
router.put('/update/:id', [isAdmin], catController.updateCat);
router.delete('/delete/:id', [isAdmin], catController.deleteCat);

export default router;
```

Fig 6.17. Llamada función comprobación rol usuario

6.2.2.2. Seguridad tratamiento datos usuarios

Recuperando la lista de aspectos relevantes para el diseño (4.2.1.), uno de los factores que más valor aporta a una aplicación, es la credibilidad, y entre los aspectos que ofrecen este valor se encuentra la seguridad. Por lo que es importante tratar con seguridad los datos del usuario. Se propuso tratar aquellos datos más sensibles de manera encriptada, y así preservar la confidencialidad. Uno de los datos más sensibles del usuario es la contraseña, por lo que el primer objetivo fue tratar este parámetro. Para ello, al crear un nuevo usuario, se guarda la contraseña de manera encriptada (Fig 6.18.). Primero se debe generar una cadena de caracteres aleatoria mediante la función “genSalt(10)”, diez es el valor por defecto para contar con un mínimo de seguridad sin representar una gran carga de computación y repercutir en el rendimiento y experiencia del usuario. Después se crea un *hash* con la contraseña y el valor aleatorio previamente creado (salt). La función hash genera un texto de una longitud y valor fijo, es decir, para una entrada siempre obtendrá la misma salida, por lo que si solo se pasa como parámetro la contraseña, el valor sería siempre el mismo, y por lo tanto, vulnerable; por eso es necesario contar con la cadena de carácteres aleatoria (salt), y así aumentar la seguridad. Una vez encriptada la contraseña. se guarda en la base de datos, y aunque alguien tenga acceso a esta, no conocerá el valor.

```
async function newUser (req:Request, res:Response): Promise<void> {

    const id = Math.floor(Math.random() * (10000000 - 1 + 1) + 1);
    const { username, password, email, imageUrl, tasks } = req.body;
    const salt = await bcrypt.genSalt(10);
    const hashed = await bcrypt.hash(password, salt);
    const newUser = new User({ id, username, password: hashed, email, imageUrl});
    const roleadded = await Role.findOne({name: "standard"});
    newUser.roles = roleadded?._id;
    await newUser.save();
    res.status(200);
}
```

Fig 6.18. Función creación nuevo usuario

Por lo que esto genera otro reto, al almacenar la contraseña del usuario guardada en la base de datos, como se verificará que la contraseña introducida por el usuario cuando inicie sesión es la correcta. Cuando el usuario trate de logearse en la aplicación, utilizaremos la función "*compare*" (Fig 6.19.) de la biblioteca *"bcrypt"*, la misma que ha sido utilizada para encriptarla. Esta función es capaz de comprobar si la contraseña pasada como parámetro por el usuario es la misma que la guardada en la base de datos; en caso de que sea así, se envía un *token* de autenticación al usuario.

```
    const matchPassword = await bcrypt.compare(body.password, userDB.password)
  // Valida que la contraseña escrita por el usuario, sea la almacenada en la db
    if (!matchPassword){
      return res.status(400).json({
        ok: false,
        err: {
          message: "Usuario o contraseña incorrectos"
        }
      });
    }
  // Genera el token de autenticación
     let token = jwt.sign({
          user: userDB,
        }, process.env.SEED_AUTENTICACION, {
        expiresIn: process.env.CADUCIDAD_TOKEN
     })
     res.json({
        ok: true,
        user: userDB,
        token,
     })
  })
```

Fig 6.19. Función creación nuevo usuario

Como se puede observar, al enviar al usuario el mensaje de confirmación de la creación del usuario en la base de datos, se envía un *token* de autenticación. Este *token* se ha generado a través del usuario de la base de datos, y con un

tiempo de validez de 48h, definido en la constante “CADUCIDAD_TOKEN”; durante los dos próximos días, el usuario podrá acceder a la aplicación sin necesidad de introducir el correo y la contraseña, ya que se ha generado este parámetro que asegura que ese usuario es válido. Este *token* se guarda en un almacenamiento local del usuario, y cada vez que se necesite, se enviará este parámetro para que el *backend* compruebe que es correcto (Fig 6.20.).

```
export const verifyToken = async (req, res, next) => {
  let token = req.headers["x-access-token"];

  if (!token) return res.status(403).json({ message: "No token provided" });

  try {
    const decoded = jwt.verify(token, SECRET);
    req.userId = decoded.id;

    const user = await User.findById(req.userId, { password: 0 });
    if (!user) return res.status(404).json({ message: "No user found" });

    next();
  } catch (error) {
    return res.status(401).json({ message: "Unauthorized!" });
  }
};
```

Fig 6.20. Función comprobación autenticación *token*

6.2.2.3. Asignar tarea al usuario cuando se crea la tarea

El último contratiempo en la parte de *backend* del proyecto se produjo a la hora de asignar una tarea a un usuario. Tal y como se vió en apartados anteriores (5.2.), los usuarios cuentan con tareas asignadas, y a la vez, las tareas tienen usuarios asignados, es decir, cuentan con una doble relación. Por lo tanto, cuando se crea o se asigna una tarea, también se deben de modificar las tareas asignadas a cada usuario. Para solventar este problema, cuando se crea (Fig 6.21.) o se asigna una tarea, se busca al usuario y, mediante la función *“push”*, se añade a la lista de tareas del usuario el identificador al que hace referencia la tarea. Este método ha sido utilizado en muchas otras funciones, tales como la creación y asignación de gatos en colonias.

```
async function newTask(req: Request, res: Response): Promise<void> {

    const task = new Task({
        "id": Math.floor(Math.random() * (10000000 - 1 + 1) + 1),
        "date": new Date(req.body.date),
        "subject": req.body.subject,
        "description": req.body.description,
        "done": false,
        "volunteer": await User.findOne({ "username": req.body.volunteer }).populate('_id'),
    });

    await User.updateOne({username : req.body.volunteer}, {$push: {tasks: task._id}})


    task.save().then((data) => {
        return res.status(201).json(data);
    }).catch((err) => {
        return res.status(500).json(err);
    })
}
```

Fig 6.21. Función creación tarea

6.2.3. Backoffice

En este último apartado del proyecto también se ha contado con dificultades, y aunque el número ha sido menor debido a la sencillez de las funcionalidades, cabe destacar la siguiente función:

6.2.3.1. Distribución de los datos en columnas

Con tal de conseguir un diseño parecido al definido en los *mockups* (Fig 4.7.), se requería dividir las tarjetas con los datos de cada clase en tres columnas, para así ahorrar espacio y conseguir un diseño más atractivo y funcional para el usuario. Para conseguir el objetivo se debía de crear tres columnas, y en cada una de ellas una lista individual con los datos a mostrar (Fig 6.22.) , es decir, tres columnas con una lista individual para cada una en vez de una tabla. Una vez diferenciadas las columnas, se debía de dividir los datos recibidos desde la llamada a *backend* en esas tres listas (Fig 6.23.). La función que distribuye todos los datos en tres listas, recorre de uno en uno los datos obtenidos, una vez ha llegado al tercer registro, se inicializa a zero y vuelve a repetir el proceso; en cada registro guarda el dato en una lista diferente, obteniendo así una distribución equitativa entre las listas. Esta función se encuentra implementada en todas las clases que se encuentran en el *backoffice*, para así mantener una estética homogénea.

```
<div id="btnContainer">
    <button (click)="newUser()" class="btn">AÑADIR UN USUARIO</button>
</div>

<div class="div1"></div>

<div class="row">
    <div class="column">
        <app-data-users *ngFor="let user of users3" [user]="user">
        </app-data-users>

    </div>

    <div class="column">
        <app-data-users *ngFor="let user of users2" [user]="user">
        </app-data-users>

    </div>

    <div class="column">
        <app-data-users *ngFor="let user of users1" [user]="user">
        </app-data-users>
    </div>
</div>
```

Fig 6.22. División de columnas

```
ngOnInit(): void {

  this.userService.getUsers().subscribe(data => {
    this.users = data;
    this.users1 = [];
    this.users2 = [];
    this.users3 = [];
    let i: number = 0;
    let y: number = this.users.length;

    while (i <= y) {
      if (i == 0) {
        this.users1.push(this.users[y - i]);
      }
      if (i == 1) {
        this.users2.push(this.users[y - i]);
      }
      if (i == 2) {
        this.users3.push(this.users[y - i]);
        y = y - 3;
        i = -1;
      }
      i++;
```

Fig 6.23. Función división datos en tres listas

Resumen:

En este apartado hemos podido ver el desarrollo final de la aplicación, así como aquellas funcionalidades más complejas de realizar, y los problemas que se han podido presentar con sus respectivas soluciones y justificaciones.

CAPÍTULO 7: SEGUIMIENTO Y CONTROL

Una vez llevado a cabo las actividades relacionadas con el proceso de desarrollo de las aplicaciones, se procede a la realización de la entrega provisional del producto, con su correspondiente prueba y posterior análisis de los datos obtenidos.

7.1. Entrega producto provisional

Para realizar la primera entrega del producto, estaba previsto publicar tanto la parte de *backend*, como la de *frontend* y *backoffice* en un *docker* (o contenedor), para así poder desplegar la aplicación y cualquier usuario pudiese acceder desde cualquier dispositivo a través de un enlace. Por falta de tiempo y recursos no se ha podido llevar a cabo esta parte del proyecto, por lo que se intentó buscar otra alternativa, subir a una plataforma en la nube la parte de *backend* para que así la aplicación (*frontend*) pudiese hacer las peticiones a esa dirección en lugar de la dirección local. De esta manera se obtiene un resultado similar al previamente planteado, ya que se consigue tener esa parte del proyecto virtualizada y accesible desde cualquier dispositivo, sin la necesidad de estar ejecutando el *backend* en el dispositivo.

Con tal de realizar el estudio de los objetivos cumplidos y las posibles futuras mejoras a efectuar en el proyecto, se realizó un simulacro de uso de la aplicación entre el cliente final y el desarrollador del proyecto. Para ello, se instaló el *software* en el dispositivo móvil de las voluntarias que ese día realizaban el circuito de las colonias, y sin dar indicaciones se les solicitó que fuesen realizando las tareas que hacían con normalidad añadiendo la ayuda que ofrece la aplicación. Durante el tiempo de prueba el desarrollador estuvo con las voluntarias con tal de brindar la ayuda necesaria, y al final del recorrido se realizó una pequeña reunión para recoger las sensaciones y los datos tomados con tal de poder realizar un análisis de las funcionalidades y su funcionamiento durante la realización de las tareas. Previamente a la realización de la entrega provisional del producto se realizó una simulación de los datos de las colonias en las cuales se iba a realizar la prueba, para que así las voluntarias pudieran testear la aplicación en unas condiciones similares a las reales. Estos datos fueron eliminados después de hacer las pruebas para proteger los datos de las colonias.

7.2. Feedback entrega provisional

Una vez los usuarios a los que iba dirigida la aplicación han podido usarla para administrar el trabajo que se realiza de forma habitual en las colonias de gatos, se ha obtenido el siguiente feedback con aquellos puntos fuertes y débiles a considerar:

- Por una parte, es importante mencionar la falta de la aplicación que iba dirigida a usuarios que no formasen parte de la protectora. La utilidad que ofrecía esta aplicación en concreto, como ya se ha comentado anteriormente, estaba relacionada con las tareas de adopción, un proceso que a menudo consideran complejo de administrar. Por ello, han considerado que una futura mejora de la aplicación consistiría en la implementación de esta función, que por problemas relacionados con la falta de tiempo, no se ha podido desarrollar de forma satisfactoria.

- Por otra parte, respecto a la función que ofrece el calendario, se encuentra un error práctico. Este consiste en la sensación que genera el hecho de que no se muestre la tarea que se crea en el calendario, enlazada a un día concreto. Para que esta se pueda visualizar en el calendario, es necesario cerrar sesión y volver a entrar para que esta aparezca, hecho que puede suscitar en el usuario una sensación de error al crear la tarea en un día concreto. Respecto al calendario, también han considerado oportuno mencionar que, al crear una tarea en la que se ha de delimitar la fecha en la que se llevará a cabo, sería más cómodo realizarlo a través de un calendario que te ofrezca la propia aplicación, en vez de tenerlo que llevar a cabo de forma manual con un formato en concreto para que esto quede bien registrado.

- Además, los usuarios han considerado que sería útil un registro de los comentarios que se van creando respecto a una colonia concreta, para así poder ver la evolución de esta, sin que los comentarios antiguos queden borrados cada vez que se genere uno nuevo.

- Respecto al empleo de la aplicación, han considerado que es de fácil uso. Esto les ha permitido llevar a cabo todas las funciones sin necesidad de preguntar cómo se llevaba a cabo alguna tarea, por lo que se considera una aplicación intuitiva y de fácil uso. Aunque se han detectado algunas posibles mejoras, como la de delimitar las opciones de marcado en los campos de “sexo”, “compatibilidades”, “incompatibilidades” y “enfermedades” de los animales, así como el formato de la fecha de expiración de la tarea previamente comentado.

- Además, el objetivo diana por el cual se creó la aplicación, que consiste en ofrecer una herramienta para poder administrar las tareas relacionadas con las colonias de gatos de forma más sencilla y eficiente, los usuarios que han podido probarlo consideran que lo cumple, y que podría considerarse un antes y un después a la forma en la que han administrado tradicionalmente estas colonias (considerando que ello les ha llevado mucho tiempo).

- Por otra parte, los usuarios han explicado que consideran positivo que no se puedan crear o borrar ciertos ítems (como podría ser borrar una colonia) desde la aplicación, y que este tipo de funcionalidades solo se puedan llevar a cabo desde la interfaz del *backoffice*. Esto se debe a que un error por parte de algún usuario podría conllevar grandes modificaciones en la gestión de ciertas tareas, que conllevarían un

perjuicio para las colonias. De este modo, sólo a través del *backoffice*, que mayoritariamente no se usaría, se podría realizar este tipo de modificaciones. Además, otras funcionalidades, menos críticas, pero que también repercuten en el funcionamiento de la protectora como la creación o eliminación de gatos, también cuentan con una limitación de permisos, permitiendo modificar este tipo de datos solo a las personas estipuladas.

- Por último, han considerado oportuno mencionar que la aplicación ofrece un estilo visual agradable, así como un carácter propio para poder hacerse conocer a través de este estilo particular. Esto permite que el uso por parte de los usuarios se lleve a cabo de forma complaciente, y que acaben adquiriendo una sensación grupal de pertenecer a un colectivo bien delimitado, con una marca concreta.

7.3. Corrección errores

Tras los comentarios recogidos por el cliente, se asume un resultado positivo, ya que se ha cumplido uno de los grandes objetivos del proyecto, mejorar la eficiencia de las tareas realizadas durante el recorrido de las colonias, así como la del resto de labores que llevan a cabo las voluntarias. Aún así se han detectado pequeños fallos en el aplicativo que deberán de ser corregidos para mejorar la experiencia del usuario, no obstante, la versión actual permite realizar todas las tareas correctamente, mejorando la eficiencia de las tareas.

Debido a la proximidad de la entrega provisional del producto y la del propio proyecto, algunas de las correcciones propuestas por las usuarias no han podido ser llevadas a cabo, por lo que deberán de ser desarrolladas en futuras versiones de la aplicación. Sin embargo, algunos errores sí que han podido ser corregidos o identificados. Algunas de las aportaciones que se han podido reemplazar han sido las de delimitación de las opciones de “sexo”, “compatibilidades”, “incompatibilidades” y “enfermedades” a la hora de crear o modificar un gato. Estos campos han sido modificados por menús que permite seleccionar una o varias opciones (Fig 7.1.).

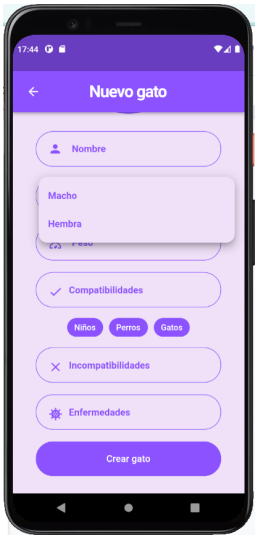

Fig 7.1. Modificación campos crear gato

Además, se han identificado los motivos de otros errores trasladados por los usuarios, como la no visualización de las tareas creadas en el calendario, esto es debido a que el *widget* de calendario cuenta con un estado, a medida que se van realizando modificaciones en la aplicación va cambiando este estado. El problema reside en que el estado del calendario es inicializado al reanudar el aplicativo, por lo que se debería de actualizar este estado cada vez que una tarea sea creada, y así reflejar esos cambios en el calendario.

Resumen:

En este apartado se ha detallado la prueba del producto, y como han sido las sensaciones del cliente en su primera toma de contacto. Además, se han trasladado aquellas funcionalidades que podrían adaptarse mejor a las necesidades de la protectora. Finalmente se han detallado los cambios realizados tras los comentarios de las voluntarias de la protectora, así como la identificación de los fallos expresados por parte del cliente.

CAPÍTULO 8: RESULTADOS DEL PROYECTO

Tras realizar la entrega provisional del producto, así como el análisis de los datos obtenidos de la entrega provisional del producto, se procederá a realizar un estudio sobre los resultados obtenidos a lo largo del proyecto, así como posibles mejoras en el trabajo realizado.

8.1. Análisis de los resultados obtenidos

Una vez finalizada la fase de desarrollo del proyecto, y tras realizar una reflexión crítica de todo el proceso, se considera que los resultados obtenidos se podrían considerar positivos mayoritariamente. Esto es debido a que consiguen suplir las principales necesidades que presentaba el cliente, y por lo tanto, logran adaptarse a los objetivos que se habían descrito inicialmente. Estos, como ya se había comentado, estaban relacionados con funcionalidades que permitían gestionar de una manera más sencilla, rápida y eficaz todas aquellas tareas que lleva a cabo la protectora de forma habitual. Todo ello se ha podido lograr gracias a la toma de decisiones focalizada hacia estos objetivos concretos, hecho que ha conllevado que se abandonasen alguna de las ideas que se habían propuesto al inicio del proyecto. Esto se puede observar en la decisión de priorizar la aplicación dirigida a las integrantes de la protectora, que estaría dirigida a dichas tareas de gestión de la propia organización, en contraposición a la aplicación para usuarios externos a la asociación, que por factores como la falta de tiempo se ha decidido abandonar para evitar entregar un producto incompleto o mal diseñado.
Además, la falta de tiempo ha ocasionado otros handicaps como el no poder realizar el desarrollo en producción mediante la técnica de *dockers*, otorgando así flexibilidad y facilidades a la hora de la instalación de la aplicación a los clientes, así como la modificación del código sin perturbar la utilización de los servicios. Tal y como se ha comentado anteriormente, la infravaloración de los objetivos respecto al tiempo y esfuerzos necesarios para desarrollarlos, ha desembocado en la no realización de una aplicación para usuarios externos a la protectora, ya que los resultados obtenidos momentos previos a la entrega provisional del producto no otorgaban un gran valor, por lo que se decidió centrar todos los esfuerzos en conseguir un buen resultado en el aplicativo destinado a las voluntarias de la protectora, que constituían el objetivo principal del proyecto. A lo largo de las reuniones de control y seguimiento realizadas con el cliente, se han ido excluyendo funcionalidades que en un inicio se creían útiles, pero que tras el replanteamiento de la aplicación y un análisis profundo de la utilización y valor que iban a aportar esas funcionalidades no se han considerado necesarias; algunas de las funcionalidad excluidas o modificadas han sido la opción de realizar videollamadas entre voluntarias, así como el chat entre las mismas y la asignación de objetos a voluntarias.
Por otra parte, el desarrollo de las diferentes aplicaciones (*frontend* y *backoffice*), me han otorgado un alto grado de aprendizaje y me han permitido profundizar en las tecnologías CSS y HTML, las cuales conocía de manera algo banal. Sin embargo, con Flutter, una tecnología con la que había trabajado de una manera más intensa, la realización de este proyecto me ha otorgado la

capacidad de conocer muchas técnicas que desconocía. Estos conocimientos también han sido desarrollados en gran parte en las tecnologías utilizadas en *backend*, ya que he llevado a cabo técnicas que desconocía y que me ofrecerán un gran valor en mi futuro profesional.
Los resultados obtenidos en el ámbito personal son más que satisfactorios, ya que este trabajo me ha otorgado los conocimientos necesarios para poder tratar con clientes, y conocer los diversos procesos que se llevan a cabo a la hora de plantear un proyecto de estas características. En este aspecto he obtenido un gran aprendizaje sobre el trato con el cliente y el enfoque necesario a la hora de determinar los objetivos del proyecto, así como determinar con claridad la prioridad de las necesidades con tal de poder generar un orden de preferencia a la hora de desarrollar el proyecto; esto ha facilitado la toma de decisiones, agilizando el proceso y mejorando la eficiencia.
Las funcionalidades pendientes de realizar, junto con la causa del proyecto y el desarrollo de este, me alienta a continuar con el proyecto y a contribuir con nuevas funcionalidades con tal de poder mejorar el funcionamiento de la protectora.

8.2. Posibles mejoras

Tras los comentarios recibidos por el cliente, y el no desarrollo del aplicativo correspondiente a los usuarios externos a la protectora, se cuenta con diversas opciones de evolución de la aplicación, así como de la terminación y corrección de algunas funcionalidades. Más concretamente se plantea la solución de la actualización de las tareas en el mapa cuando estas han sido creadas, así como el registro de observaciones realizadas por las voluntarias, o la creación de la aplicación para usuarios externos a la protectora, desarrollando todas las funcionalidades explicadas anteriormente (4.1.1.). En el ámbito del aplicativo que no se ha llegado a realizar, hay muchas operaciones que ya están implementadas o preparadas para su implantación en la aplicación desarrollada, como el muro de noticias, que se encuentra en la aplicación que se ha logrado llevar a cabo, o como la estructura de datos que está adaptada a la creación de este nuevo proyecto. Además, al contar con más tiempo y recursos, se plantea implementar una pasarela de pago para poder realizar donaciones o compra de *merchandising*, así como contribuir en retos benéficos. Otra funcionalidad que se desea implementar es la automatización de procesos tales como la contestación automática de correos electrónicos, o el procesamiento de solicitudes de adopción mediante el correo electrónico hasta que se haya desarrollado esta segunda aplicación. Otra propuesta de mejora para la aplicación sería la implementación de una herramienta de comunicación entre los usuarios externos de la protectora con los miembros de esta, implementada en la futura aplicación para aquellos usuarios externos a la protectora, así como la posibilidad de tramitar las solicitudes de adhesión a la protectora.

8.3. Conclusiones

En definitiva, este proyecto ha constituido una gran herramienta de aprendizaje personal, relacionada tanto con aquellos aspectos más teóricos y prácticos propios del carácter académico que presenta, tanto como con aquellos más relacionados con los valores éticos que ha presentado dado su origen altruista.
A través del desarrollo del proyecto he podido adquirir numerosos conocimientos, y he aprendido a administrar los tiempos de los que disponía. Esto ha implicado que tuviese que agilizar el proceso de toma de decisiones y determinar qué aspectos eran de mayor relevancia para el cliente, y cuales podían llegar a ser prescindibles en caso de ser necesario.
Además, este proceso me ha permitido acercarme a una metodología propia del trabajo que se podría desarrollar en una empresa, por lo que me ha permitido adquirir habilidades para poder desarrollarme con soltura en ciertas situaciones que pudiesen compartir características similares.
No obstante, considero que mi camino solo acaba de iniciar, y que aún me queda mucho por aprender. Aspectos como los de la administración del tiempo, los conocimientos relativos a la programación o la implementación de proyectos, constituyen pilares esenciales que me gustaría trabajar y potenciar de cara a poder ofrecer un mejor servicio.

Resumen:
En este último punto se ofrece un análisis crítico de todo el proceso de desarrollo del proyecto, así como del resultado final obtenido. Esto permite determinar que aspectos se pueden mejorar en un futuro para así crear una mejor experiencia para el usuario. Además, se desarrolla una mirada transversal en relación a todo lo que el proyecto me ha ofrecido respecto al aprendizaje.

CAPÍTULO 9: BIBLIOGRAFÍA

[1]. Infografía Él nunca lo haría. Estudio de abandono y adopción 2022 [Internet]. Fundacion-affinity.org. 2022 [citado el 18 de Julio de 2022]. Disponible en: https://www.fundacion-affinity.org/observatorio/infografia-el-nunca-lo-haria-abandono-adopcion-perros-gatos-espana-2022

[2]. Amics i protectors dels gats [Internet]. Amics i protectors dels gats. [citado el 18 de julio de 2022]. Disponible en: https://www.amicsiprotectorsdelsgats.org/

[3]. Fundamentos de la gestión de proyectos [Internet]. Wrike.com. [citado el 20 de julio de 2022]. Disponible en: https://www.wrike.com/es/project-management-guide/fundamentos-de-la-gestion-de-proyectos/

[4]. Pérez A. ¿Cuáles son las etapas de un proyecto? Te lo contamos en esta infografía [Internet]. OBS Business School. 2021 [citado el 20 de julio de 2022]. Disponible en: https://www.obsbusiness.school/blog/cuales-son-las-etapas-de-un-proyecto-te-lo-contamos-en-esta-infografia

[5]. Pérez A. Etapas de un proyecto social: un diseño que garantiza el éxito [Internet]. OBS Business School. 2015 [citado el 20 de julio de 2022]. Disponible en: https://www.obsbusiness.school/blog/etapas-de-un-proyecto-social-un-diseno-que-garantiza-el-exito

[6]. TEXT REFÓS ORDENANÇA MUNICIPAL TINENÇA D'ANIMALS Vigent a partir del 20/02/2015 [Internet]. Vilanova.cat. [citado el 20 de julio de 2022]. Disponible en: https://www.vilanova.cat/doc/doc_15450503.pdf

[7]. Asana. Las 6 restricciones de un proyecto y cómo abordarlas para tener éxito [Internet]. Asana. [citado el 20 de julio de 2022]. Disponible en: https://asana.com/es/resources/project-constraints

[8]. Angular [Internet]. Angular.io. [citado el 21 de julio de 2022]. Disponible en: https://angular.io/guide/architecture

[9]. Angular [Internet]. Angular.io. [citado el 21 de julio de 2022]. Disponible en: https://angular.io/features

[10]. React [Internet]. Reactjs.org. [citado el 21 de julio de 2022]. Disponible en: https://es.reactjs.org/

[11]. Pisuwala U. The benefits of ReactJS and reasons to choose it for your project [Internet]. Peerbits. 2022 [citado el 21 de julio de 2022]. Disponible en: https://www.peerbits.com/blog/reasons-to-choose-reactjs-for-your-web-development-project.html

[12]. Build apps for any screen [Internet]. Flutter.dev. [citado el 21 de julio de 2022]. Disponible en: https://flutter.dev/

[13]. Thomsen M. Dart 2.17: Productivity and integration - dart - medium [Internet]. Dart. 2022 [citado el 21 de julio de 2022]. Disponible en: https://medium.com/dartlang/dart-2-17-b216bfc80c5d
[14]. González DB. Flutter, el SDK de Google para desarrollar apps multiplataforma con rendimiento nativo [Internet]. Profile Software Services. 2021 [citado el 25 de julio de 2022]. Disponible en: https://profile.es/blog/que-es-flutter-sdk/

[15]. Todo lo que necesitás saber sobre backend [Internet]. MMA Global. [citado el 21 de julio de 2022]. Disponible en: https://www.mmaglobal.com/news/todo-lo-que-necesitas-saber-sobre-backend-all-you-need-know-regarding-backend

[16]. Top 8 back-end coding languages for web development [Internet]. Indeed Career Guide. [citado el 21 de julio de 2022]. Disponible en: https://www.indeed.com/career-advice/career-development/backend-languages

[17]. Back4app.com. [citado el 25 de julio de 2022]. Disponible en: https://blog.back4app.com/es/lenguajes-de-programacion-de-backend/

[18]. Chacón JL. TypeScript: qué es, diferencias con JavaScript y por qué aprenderlo [Internet]. Profile Software Services. 2021 [citado el 25 de julio de 2022]. Disponible en: https://profile.es/blog/que-es-typescript-vs-javascript/

[19]. ¿Qué es DevOps? [Internet]. Microsoft.com. [citado el 21 de julio de 2022]. Disponible en: https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-devops/

[20]. Business2community.com. [citado el 21 de julio de 2022]. Disponible en: https://www.business2community.com/business-intelligence/9-key-benefits-of-devops-02391855

[21]. Extensions LMA. Visual Studio Code - code editing. Redefined [Internet]. Visualstudio.com. [citado el 26 de julio de 2022]. Disponible en: https://code.visualstudio.com/

[22]. MongoDB: The developer data platform [Internet]. MongoDB. [citado el 26 de julio de 2022]. Disponible en: https://www.mongodb.com/

[23]. Socket.IO [Internet]. Socket.io. [citado el 26 de julio de 2022]. Disponible en: https://socket.io/

[24]. GitHub: Where the world builds software [Internet]. GitHub. [citado el 26 de julio de 2022]. Disponible en: https://github.com/

[25]. Image and video upload, storage, optimization and CDN [Internet]. Cloudinary. 2021 [citado el 26 de julio de 2022]. Disponible en: https://cloudinary.com/

[26]. Agora.Io Real-time voice and video engagement [Internet]. 33 Agora. 2020 [citado el 26 de julio de 2022]. Disponible en: https://www.agora.io/en/

[27]. Scrum: qué es y cómo funciona este marco de trabajo [Internet]. Wearemarketing.com. [citado el 23 de agosto de 2022]. Disponible en: https://www.wearemarketing.com/es/blog/metodologia-scrum-que-es-y-como-funciona.html

[28]. SCRUM y los puntos de historia, ¿Cómo funcionan? [Internet]. Incentro.com. [citado el 23 de agosto de 2022]. Disponible en: https://www.incentro.com/es-ES/blog/scrum-puntos-de-historia-como-funcionan

[29]. Asmo. Scrum 101: An introduction to scrum project management [Internet]. Zenkit. 2018 [citado el 23 de agosto de 2022]. Disponible en: https://zenkit.com/en/blog/scrum-101-an-introduction-to-scrum-project-management/

[30]. Asmo. Kanban vs Scrum [Internet]. Zenkit. 2019 [citado el 23 de agosto de 2022]. Disponible en: https://zenkit.com/en/blog/kanban-vs-scrum/

[31]. Productivity and collaboration software suite [Internet]. Zenkit. 2020 [citado el 23 de agosto de 2022]. Disponible en: https://zenkit.com/

[32]. What is a Sprint in Scrum? [Internet]. Scrum.org. [citado el 12 de diciembre de 2022]. Disponible en: https://www.scrum.org/resources/what-is-a-sprint-in-scrum

[33]. The 7 Factors that Influence User Experience [Internet]. The Interaction Design Foundation. Interaction Design Foundation; 2016 [citado el 12 de diciembre de 2022]. Disponible en: https://www.interaction-design.org/literature/article/the-7-factors-that-influence-user-experience

[34]. Daivee. The 3-click rule: myth or fact? [Internet]. IEEE Brand Experience. IEEE; 2017 [citado el 12 de diciembre de 2022]. Disponible en: https://brand-experience.ieee.org/the-3-click-rule-myth-or-fact/

[35]. JWT.IO - JSON Web Tokens introduction [Internet]. Jwt.io. Auth0; [citado el 29 de diciembre de 2022]. Disponible en: https://jwt.io/introduction

[36]. Mongoose v6.8.2: Query population [Internet]. Mongoosejs.com. [citado el 29 de diciembre de 2022]. Disponible en: https://mongoosejs.com/docs/populate.html

[37]. Explicación Sobre Las Bases De Datos NoSQL [Internet]. MongoDB. [citado el 3 de enero de 2023]. Disponible en: https://www.mongodb.com/es/nosql-explained

[38]. conciencia-digital [Internet]. Click & Safe. [citado el 3 de enero de 2023]. Disponible en: https://clickandsafe.upc.edu/es/conciencia-digital

[39]. Internationalizing Flutter apps [Internet]. Flutter.dev. [citado el 3 de enero de 2023]. Disponible en: https://docs.flutter.dev/development/accessibility-and-localization/internationalization