2016

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Escuela de Ingeniería Informática

GPS – PET

Aplicación Web para la geolocalización de mascotas

Proyecto de Fin de Carrera

Autor: Rebeca Escarlata Pérez Alonso
Tutores: Francisca Quintana Domínguez
Carmelo Cuenca Hernández
Titulación: Ingeniería en Informática
Julio de 2016

Proyecto de Fin de Carrera de Ingeniería en Informática de la Universidad de Las Palmas de Gran Canaria

Rebeca Escarlata Pérez Alonso

Título del Proyecto:

GPS – PET: Aplicación web para la geolocalización de mascotas

Tutores:

Francisca Quintana Domínguez
Carmelo Cuenca Hernández

DEDICATORIA

Este proyecto ha sido creado por el amor que tengo hacia las personas, los animales, las plantas, la vida. Por ello, dedico este trabajo, a mis padres. Estos seres tan especiales para mí, que me han enseñado que cada cosa, cada momento, cada vida es única y bella, que todo tiene un precio y un valor, que el esfuerzo y la humildad es el pan de cada día, donde la unión hace la fuerza, y el coraje y la fuerza rompe barreras, además de recordarme que cada día hay que luchar por lo que se quiere, que la vida no para y hay que vivir como los animales, en el presente, disfrutando de todo y de todos.

También dedico este proyecto a mi preciosa hermana y a mi precioso perro Max, mis dos grandes pilares.

Y por último, se lo quiero dedicar a Daniel Martínez, mi pareja, otra gran persona con la que tengo el privilegio de compartir mi vida.

Un honor ser vuestra hija, hermana, compañera y pareja.

AGRADECIMIENTOS

Quiero empezar los agradecimientos nombrando a mis dos tutores, quienes han sido mis guías durante todo este proyecto. Ambos me han dado ánimos, ayuda, consejo, tiempo y dedicación. Hacer un proyecto en la distancia no ha sido nada fácil, pues la comunicación no fluye de la misma manera, perdiéndose mucha información en el camino. Pero ellos me han ayudado para que esto haya sido posible. Les estaré eternamente agradecida.

Muchas gracias a mis padres, que me han dado la oportunidad de estar aquí, de realizar esta carrera, de ayudarme a emprender mi camino, mis sueños y, por apoyarme y quererme incondicionalmente.

Muchas gracias a mi hermana, que siempre está ahí, observándome, dándome ánimos y fuerzas para seguir adelante, sobre todo en los momentos en los que he sentido que el agotamiento estaba haciendo mella en mí. Gracias por tu luz hermana.

Muchas gracias a mi pareja Daniel, que está siempre a mi lado, atento a mí para que esté lo mejor posible en cada momento, ayudándome para que este proyecto fuese posible.

Agradecer a mis compañeros de trabajo y a la empresa ASPgems. Sin vuestra ayuda, comprensión y apoyo, hubiera sido un camino más duro de realizar. Sobre todo quiero agradecer a cuatro compañeros que se han preocupado por mí. A Belén por dedicar su tiempo a ayudarme a ver las cosas desde otra perspectiva; a Martín por tu ayuda en mis momentos de colapso mental; a Álvaro por darme ánimos y sabios consejos; y a Pilar, una hermosa persona que me ha cuidado, dándome aire fresco y recordándome quién soy y qué es lo quiero hacer.

Quiero agradecer a muchas personas más, a todas las que conozco, pues cada una me ha enseñado muchas cosas en el transcurso de mi vida, donde cada hora, minuto y segundo compartido con ustedes (Shay, Mario, Pablo, Abraham, Jenny, mi tía Ángela, mi tío Victor, Jose Carlos, Cristo, Eli…), han hecho que sea como soy y en quién quiero ser.

Muchísimas gracias a todos, solo sé que, soy muy afortunada de vivir y de estar rodeada de grandes seres.

ÍNDICE

1. INTRODUCCIÓN

A lo largo de los años hemos ido tomando a nuestras mascotas como uno más en la familia. Su salud y bienestar es nuestra preocupación, así que procedemos a integrarlos en la sociedad identificándolos al igual que lo estamos nosotros. Para ello, los llevamos al veterinario para que le hagan la pertinente revisión y le pongan sus respectivas vacunas, además de un identificador, el microchip.

Por otro lado, el artículo 11 de la Ley 8/1991, de 30 de abril, de protección de animales del Gobierno de Canarias, obliga que "Los propietarios de perros deberán identificarlos como reglamentariamente se establezca y censarlos en el Ayuntamiento donde habitualmente viva el animal, dentro del plazo máximo de tres meses contados a partir de la fecha de nacimiento, o de un mes después de su adquisición. El animal deberá llevar necesariamente su identificación censal de forma permanente.".

Un animal se puede identificar mediante el tatuaje de un código identificativo, que era lo que se hacía antes del 03/07/2011 o, la implantación de un microchip, que es el más utilizado y estandarizado a día de hoy.

El microchip es un pequeño dispositivo del tamaño de un grano de arroz, que se le coloca entre la parte de la oreja y el costado izquierdo del animal, y sirve para conocer, entre otras cosas, mediante el uso de un lector especial y la base datos en PC, el nombre del perro, el dueño y su historia clínica.

FIGURA 1 - MICROCHIP

FIGURA 2 – LEC RFID

La lectura de la información que tiene integrado el microchip se realiza mediante un lector de RFID (Radio Frequency IDentification), que es un sistema de almacenamiento y recuperación de datos remotos que usa dispositivos denominados etiquetas, tarjetas, transpondedores o tagsRFID. El propósito fundamental de la tecnología RFID es transmitir la identidad de un objeto (similar a un número de serie único) mediante ondas de radio. Las tecnologías RFID se agrupan dentro de las denominadas Auto ID (Automatic IDentification o identificación automática).

Identificados a nuestras mascotas según está establecido en la normativa vigente, nos preocupa que cuando los sacamos o sacan a pasear, ya sea en una zona montañosa, en un parque o en una urbanización, se puedan perder o escapar cuando rastrean, persiguen a otro perro o por otro motivo. De modo que no nos sirve que solo estén identificados con el microchip o una placa, porque esto solo sirve si alguien consigue localizarlo y cogerlo, para a continuación llevarlo a una clínica veterinaria y averiguar mediante un lector de RFID quién es su propietario. Una solución a esto es poner un dispositivo de localización, como puede ser introducir al microchip un GPS o un collar con

GPS, para que nos indique la ubicación exacta del animal en caso de extravío o pérdida.

El GPS, o comúnmente llamado Sistema de Posición Global, es un sistema que permite determinar en todo el mundo la posición de un objeto, ya sea una persona, vehículo, etc, con una precisión de hasta centímetros (si se utiliza GPS diferencial), aunque lo habitual son unos pocos metros de precisión.

FIGURA 3 – SISTEMA GPS

El GPS funciona mediante una red de 24 satélites en órbita sobre el planeta tierra, a 20.200 km de altura, con trayectorias sincronizadas para cubrir toda la superficie de la Tierra. Cuando se desea determinar la posición, el receptor que se utiliza para ello localiza automáticamente como mínimo cuatro satélites de la red, de los que recibe unas señales indicando la identificación y la hora del reloj de cada uno de ellos. Con base en estas señales, el aparato sincroniza el reloj del GPS y calcula el tiempo que tardan en llegar las señales al equipo, y de tal modo mide la distancia al satélite mediante el método de trilateración inversa, que se basa en determinar la distancia de cada satélite respecto al punto de medición. Conocidas las distancias, se determina fácilmente la propia posición relativa respecto a los satélites. Conociendo además las coordenadas o posición de cada uno de ellos por la señal que emiten, se obtiene la posición absoluta o las coordenadas reales del punto de medición.

Puestos en contexto, lo que se quiere conseguir con este proyecto es que el usuario pueda registrar y gestionar las trayectorias realizadas por su o sus mascotas. Para ello, se requiere desarrollar un sistema informático donde la aplicación a implementar estará enmarcada dentro de la gestión de rutas, que serán demandadas por la aplicación web a desarrollar, siendo el propietario y el paseador del animal el que podrá visualizar mediante su identificador, la trayectoria que está realizando o ha realizado el animal.

Sin embargo, debido a que esto supone obtener varios microchips con GPS o collares con GPS, así como los animales que harán que el proyecto sea posible, se hará una **simulación** de ello mediante fuentes de ficheros de rutas.

2. OBJETIVOS

El principal objetivo que se quiere conseguir de este proyecto, es desarrollar una aplicación web en Ruby on Rails, donde el usuario pueda visualizar en "*tiempo real*" la ubicación de su animal haciendo uso de su identificador con GPS (microchip con GPS), si existiera tal dispositivo, o un dispositivo de geolocalización GPS incrustado en el collar del perro, para hacer un seguimiento del mismo cuando otra persona que ha sido contratada lo pasea.

Al no disponer de microchips o collares con GPS, así como los respectivos animales para hacer el estudio y, por tanto, su posterior seguimiento, se realizará una simulación del mismo, de ahí que el tiempo real esté puesto entre comillas.

Cuando un paseador[1] pasea a un perro, éste, si fuera en el mundo real, accionaría el botón de activación del GPS para que el propietario pueda ver en ese preciso momento de paseo, dónde está su perro y, por tanto, qué ruta está haciendo con el paseador.

Para simular el GPS, haremos uso de ficheros .kml, donde leeremos del mismo sus coordenadas para luego ser pintadas a posteriori en el mapa.

Aparte de mostrar la ubicación de la mascota del propietario cuando está siendo paseada, el propietario también podrá ver la duración de la ruta que ha realizado su perro, así como quién lo ha paseado, el estado en el que se encuentra el paseo (en progreso si sigue siendo paseado o, finalizado, si el paseo ha sido concluido), etc.

El que hará posible la simulación del GPS va a ser el paseador, de modo que, cuando un propietario se registra en el sistema e introduce sus perros en el mismo, tendrá que contratar a un paseador.

Añadir que, el paseador tendrá a su disposición un listado de paseos que ha realizado y que está realizando actualmente con los perros que está paseando, teniendo de esta manera un histórico de paseos.

1. Paseador: rol que permite a una persona a realizar los servicios contratados. Más detalle en Axexo o Glosario de conceptos.

3. ESTADO DEL ARTE

Actualmente existen empresas a nivel internacional y nacional que comercializan collares con GPS.

Mencionaré varias de ellas, siendo la primera en exponer la que se ha desarrollado en España.

3.1. DONDO: LOCALIZADOR DE PERROS PERDIDOS POR GPS

FIGURA 4 – LOCALIZADOR GPS DONDO

DONDO es una empresa española y el nombre del propio localizador GPS que venden.

Este localizador, con diseño ergonómico y poco peso para que se adapte a cualquier tipo de mascota, cuyo tamaño real es similar a la de un mechero (34 gramos y 72 horas de autonomía), va incrustado en el collar o en el arnés del animal. Además, presenta unos indicadores luminosos que indica el estado del dispositivo.

DONDO permite rastrear a tu mascota desde el móvil, introduciendo para ello, una tarjeta SIM y conectando el dispositivo GPS.

Ofrece a los clientes una **aplicación móvil gratuita**, disponible para Android e iOS, o SMS, que proporciona la posición exacta del animal de compañía que se quiere localizar en el mapa. Puede establecer un perímetro de seguridad, avisando si éste sale del mismo y, tiene un sistema multiusuario, para que más personas o miembros de la familia, también puedan tener localizada a la mascota.

3.2. POSITION LOGIC: SOLUCIONES DE RASTREO GPS PARA EL SEGUIMIENTO DE MASCOTAS

Es una empresa líder B2B (Business to Business) en servicios de localización y rastreo, que se compromete a apoyar a clientes empresariales, suministrándoles inteligencia integrada, tecnología precisa de ubicación, y servicios y soluciones personalizadas para todos sus clientes.

Con sede en Naples, Florida, ofrecen servicios a nivel nacional e internacional, con un software y soluciones que se utilizan en todo el mundo.

Position Logic se concentra, principalmente, en el suministro de soluciones para empresas que se ocupan en las áreas de gestión y logística de activos empresariales, seguimiento de activos vía GPS, controles avanzados de seguridad y los servicios profesionales de consultoría.

Es una plataforma de seguimiento GPS, que ofrece a los propietarios de mascotas una serie de opciones de dispositivos para poder realizar un seguimiento de ellos. Al poner un pequeño dispositivo de seguimiento GPS al collar o arnés del animal, los propietarios pueden seguir los movimientos de su mascota, pudiendo incluso poner el nombre de la mascota, foto y descripción para facilitar su identificación.

Esto lo pueden hacer a través de la conexión de uno de los dispositivos GPS que mencionaré a continuación y, de la plataforma de Position Logic. La mayoría de estos dispositivos están equipados con un micrófono para que los usuarios puedan escuchar lo que su mascota esté haciendo.

Utilizan tres GPSs que mencionaré más adelante, siendo dos de ellos localizadores de personas y uno de animales.

3.2.1. MÉTODOS DE COMUNICACIÓN

Estos dispositivos presentan tres métodos de comunicación:

- **GPRS**: General Packet Radio Service (GPRS) o servicio general de paquetes vía radio, creado en la década de los 80, es una extensión del Sistema Global para Comunicaciones Móviles (Global Systemfor Mobile Communications o GSM) para la transmisión de datos mediante conmutación de paquetes. Existe un servicio similar para los teléfonos móviles, el sistema IS-136. Permite velocidades de transferencia de 56 a 114 kbps.

  Una conexión GPRS está establecida por la referencia a su nombre del punto de acceso (APN). Con GPRS se pueden utilizar servicios como Wireless Application Protocol (WAP), servicio de mensajes cortos (SMS), servicio de mensajería multimedia (MMS), Internet y para los servicios

de comunicación, como el correo electrónico y la World Wide Web (WWW).

Para fijar una conexión de GPRS para un módem inalámbrico, un usuario debe especificar un APN, opcionalmente un nombre y contraseña de usuario, y muy raramente una dirección IP, todo proporcionado por el operador de red. La transferencia de datos de GPRS se cobra por volumen de información transmitida (en kilo o megabytes), mientras que la comunicación de datos a través de conmutación de circuitos tradicionales se factura por minuto de tiempo de conexión, independientemente de si el usuario utiliza toda la capacidad del canal o está en un estado de inactividad. Por este motivo, se considera más adecuada la conexión conmutada para servicios como la voz que requieren un ancho de banda constante durante la transmisión, mientras que los servicios de paquetes como GPRS se orientan al tráfico de datos. La tecnología GPRS como bien lo indica su nombre es un servicio (Service) orientado a radio enlaces (Radio) que da mejor rendimiento a la conmutación de paquetes (Packet) en dichos radioenlaces.

- **SMS**: El servicio de mensajes cortos o servicio de mensajes simples, más conocido como SMS (por las siglas del inglés Short Message Service), es un servicio disponible en los teléfonos móviles que permite el envío de mensajes cortos, conocidos como mensajes de texto, entre teléfonos móviles.

  Este servicio fue inventado en 1985 por MattiMakkonen, junto al sistema global para las comunicaciones móviles (Global Systemfor Mobile communications, GSM). El SMS se diseñó originalmente como parte del estándar GSM de telefonía móvil digital, y actualmente está disponible en una amplia variedad de redes, incluidas las redes 4G, usándolo tanto teléfonos fijos y como los dispositivos de mano.

- **GSM**: El sistema global para las comunicaciones móviles (del inglés Global Systemfor Mobile communications, GSM, y originariamente del francés group espécial mobile) es un sistema estándar, libre de regalías, de telefonía móvil digital. Un cliente GSM puede conectarse a través de su teléfono con su computador y enviar y recibir mensajes por correo electrónico, faxes, navegar por Internet, acceder con seguridad a la red informática de una compañía (red local/Intranet), así como utilizar otras funciones digitales de transmisión de datos, incluyendo el servicio de mensajes cortos (SMS) o mensajes de texto.

El logotipo para identificar los terminales y sistemas compatibles es:

Este sistema de comunicación se considera, por su velocidad de transmisión y otras características, un estándar de segunda generación (2G). Su extensión a 3G se denomina UMTS y difiere en su mayor velocidad de transmisión, el uso de una arquitectura de red ligeramente distinta y sobre todo en el empleo de diferentes protocolos de radio (W-CDMA).

3.2.2. DISPOSITIVOS GPS

3.2.2.1. DISPOSITIVO GPS PERSONAL: INTELLITRAC SERIE P1 DE STC

FIGURA 5 - DISPOSITIVO INTELLITRAC

Es de gran utilidad para el seguimiento de personas. Está compuesto de una carcasa impermeable que garantiza que no sufra daños de fuentes externas, teniendo una batería interna que proporciona una conexión constante durante días, mientras que también rastrea los cambios de movimiento y altura.

Sus características:

Fabricante:	Systems & Technology Corporation
Modelo:	P1 Series
Tipo de Dispositivo:	Rastreador Personal
E/S	0/0
Batería:	Si – Bateria de Respaldo Interna
Métodos de comunicación:	GPRS, SMS, GSM

FIGURA 6 - CARACTERISTICAS DISPOSITIVO INTELLITRAC

3.2.2.2. DISPOSITIVO GPS PERSONAL: SANAV MU-201

FIGURA 7 - DISPOSITIVO SANAV MU-201

Está considerado como el *GPS de rastreo personal más pequeño del mercado*, siendo su máxima duración 100 horas con 100 informes de posición y 50 horas con 600 informes de posición. Esto es gracias a la conexión de baja potencia de los módems GPS y GSM, combinado con una potente batería de 650 mAh.

Este dispositivo es utilizado para la realización de seguimiento de personas mayores, niños e incluso mascotas, y su funcionalidad más preciada es la de enviar una alerta a un determinado contacto, a través de mensajes de texto o de correo electrónico, presionando para ello el botón de SOS cuando las personas tienen necesidades de salud o seguridad.

Presenta las siguientes características:

Fabricante:	Sanav
Modelo:	MU-201
Tipo de Dispositivo:	Rastreador de Personas
E/S	0/0
Batería:	Si – Bateria de Respaldo Interna
Métodos de comunicación:	GPRS, SMS, GSM

FIGURA 8 - CARACTERISTICAS DISPOSITIVO

3.2.2.3. DISPOSITIVO PARA RASTREO DE PERROS: XIRGO XT-2200

FIGURA 9 - XIRGO XT-2200

Es una unidad compacta, pequeña y sellada de gran durabilidad permite al dueño adherir la unidad XT-2200 al collar y tener localizado en todo momento a la mascota. No obstante, también es ideal para el rastreo de artículos o personas, gracias a su diseño y batería de larga durabilidad.

Las especificaciones del dispositivo son:

Fabricante:	Xirgo
Modelo:	XT-2200
Tipo de Dispositivo:	Rastreador Personal de Bienes
E/S	0/0
Batería:	Si – Bateria de Respaldo Interna
Métodos de comunicación:	GPRS, SMS, GSM

FIGURA 10 - CARACTERISTICAS DISPOSITIVO XIRGO XT-2200

Position Logic proporciona el rastreo, pero los dispositivos o unidades de rastreo, o lo que es lo mismo, el hardware, es independiente, pudiendo ser utilizado cualquier de ellos.

Se *basan en el servicio de rastreo B2B o Negocio a Negocio*, donde tienen integrados más de 400 aparatos que van desde aplicaciones para el rastreo individual, encubierto u oculto, control de vehículos y administración de flotillas, rastreo de botes, aviones y mucho más.

3.2.3. PRODUCTOS

Esta empresa tiene a disposición del cliente una demo en la que pueden realizar una demostración para que puedan ver los productos que ofrecen, que de forma resumida son:

	Dispositivo GPS	Aplicación	E/S	Batería	Tipo de Dispositivo	OTA
	ATrack AT1[E] El Rastreador GPS Principal AT1 [E] de A-Track es un rastreador compacto de vehículos que ofrece todas las características comunes requeridas para rastrear eficazmente diversas clases de vehículos.	Rastreo de Vehículos	2/2	Sí – Batería de Respaldo Interna	GPS / GSM / GPRS / SMS / DTMF	Sí
	ATrack AT5[i] Con el AT5[i] de A-Track, los usuarios obtienen más. Más conexiones, protocolos, valor y características.	Rastreo de Vehículos	5/3	Sí – Batería de Respaldo Interna	GPS / GSM / GPRS / SMS / DTMF	Sí
	Cal-Amp LMU-700 El dispositivo LMU-700 es un producto económico dirigido a rastreo de vehículos, diseñado para su instalacion fácil y segura en automóviles.	Vehículos	3/3	Ninguno	GPS / GSM / GPRS / SMS / DTMF	Sí
	Cal-Amp LMU-800 El dispositivo Cal-Amp LMU-800 ofrece una solución económica de GPS para una simple implementación en vehículos que requieren el beneficio de una reserva de batería.	Activos/Vehículos	3/3	Interna	GPS / GSM / GPRS / SMS / DTMF	Sí
	Cal-Amp LMU-900 El dispositivo Cal-Amp LMU-900 es una elección fantástica para el rastro GPS de vehículos. Cuatro puertos personalizados de entrada & salida, una batería de respaldo de 700mAh y acelerómetro de 3-vías, son solo algunas de las características tremendamente útiles.	Activos/Vehículos	3/3	Interna	GPS / GSM / GPRS / SMS / DTMF	Sí
	Cal-Amp LMU-1100 El dispositivo Cal-Amp LMU-1100 es una solución económica para vehículos recreacionales de agua y al aire libre como botes, motos acuáticas, motocicletas, ATVs y motos de nieve.	Activos/Vehículos	1/1	Interna	GPS / GSM / GPRS / SMS / DTMF	Sí
	Cal-Amp LMU-1200 El dispositivo Cal-Amp's LMU-1200 es una elección fantástica para el rastreo de automóviles GPS. La batería 700-mAh le permite al dispositivo reportar incluso cuando el vehículo esta apagado. La alternativa de usar una batería interna o externa proporciona al	Activos/Vehículos	4/4	Interna	GPS / GSM / GPRS / SMS / DTMF	Sí

FIGURA 11 - TABLA DE DISPOSITIVOS GPS

3.3. TRACTIVE: DISPOSITIVO DE LOCALIZACIÓN GPS PARA PERROS, GATOS…

Es una aplicación web que ofrece al usuario una serie de productos para el seguimiento de la mascota, además de apps móviles gratuitas para su posterior rastreo.

En esta plataforma se activará el rastreador una vez comprado e incorporado al collar del animal, y podrá registrar la o las mascotas que tenga el cliente.

3.3.1 PRODUCTOS

3.3.1.1 TRACTIVE: GPS PET TRACKING

FIGURA 12 - GPS PET TRACKING

Es un dispositivo GPS que permite localizar a las mascotas en cualquier momento y en cualquier lugar, recibiendo el propietario notificaciones de la posición en la que se encuentran.

Su **funcionamiento** es el siguiente: la información de la posición del animal se subirá en intervalos regulares a los servidores de Tractive, para que de este modo los usuarios puedan ver la ubicación actual del animal, e incluso mostrar dónde estuvo minutos previos.

De modo que el GPS se comunica con las apps y la aplicación web de Tractive, pagando esta empresa las tarifas de datos móviles y el cliente una pequeña cuota, no siendo esto un contrato de móvil, pues no cobra ningún cargo de activación o cargos adicionales.

Los **detalles técnicos** del dispositivo son:

- Vida de la batería: 2-5 días.
- La batería se recarga completamente en 2 horas.
- Recomendado para los animales domésticos por encima de 4,5 kg (9 libras).
- Tamaño: 51mm x 41mm x 15mm (2.0in x 1.6in x 0.6in).
- Peso: 35g (1.2oz).

3.3.1.2 MOTION: PET ACTIVITY TRACKING

FIGURA 13 - PET ACTIVITY TRACKING

Es un **dispositivo GPS** de 7 gramos de peso, resistente al agua y con una semana de duración de batería, que realiza el seguimiento de la actividad del animal durante las 24 horas del día.

Para ello, hay que conectar el hardware en el collar de la mascota y obtener los datos estadísticos directamente en el teléfono móvil mediante la utilización del Bluetooth, permitiendo de esta manera establecer metas diarias.

¿Qué **tipo de datos** obtenemos?

- Nos dice si la mascota es perezoso, moderadamente activo o muy activo.
- Da un promedio de la actividad que realiza mensualmente para poder realizar una comparativa.
- Nos da información de cuánta actividad hace dependiendo de influencias externas, tales como la temperatura o la luz del día.
- Y nos proporciona valores de movimientos diarios que realiza, para así poder establecer metas de sus actividades y mantenerlo sano y activo.

3.3.1.3 MOTION: PET REMOTE TRACKING

FIGURA 14 - PET-REMOTE TRACKING

Dispositivo pequeño y ligero, de 10 gramos y 1,7 por 0,9 pulgadas y batería de duración de un año, que permite mandar tres órdenes distintas al animal según el modo de vibración. Estas órdenes que se mandan a través de la app móvil que proporciona la empresa son: sentarse, tumbarse y ponerse a dos patas.

3.4. OTRAS APLICACIONES

Estas son algunas de las aplicaciones webs donde se puede comprar el producto GPS y obtener su app móvil gratuita para poder ver el rastreo de la mascota.

4. METODOLOGÍA DE DESARROLLO

El objetivo de la Ingeniería del Software es optimizar la calidad de los productos de software, para ampliar la productividad y facilitar el trabajo de los ingenieros del software, proporcionándoles las bases necesarias para construir software de alta calidad en forma eficiente.

Existen diversas etapas y procedimientos a las que se las denomina ciclo de vida, donde se definen parámetros como el tiempo y las características necesarias para el software considerado confiable y completo.

Sabiendo esto, durante la etapa de análisis y desarrollo, haré uso de las herramientas utilizadas en Ingeniería del Software, utilizando en el análisis un entorno orientado a objetos con UML (análisis de requisitos de usuario y análisis de requisitos de software). En el diseño generaré el diseño de la base de datos y el diseño de la aplicación web. Y en el desarrollo, conociendo diversos modelos para construir un producto de software, tomaré como referencia el **modelo de prototipos**.

4.1. MODELO DE PROTOTIPOS

El modelo de prototipos permite que todo el sistema, o alguna de sus partes, se construyan rápidamente para comprender con facilidad y aclarar ciertos aspectos en los que se aseguren que el desarrollador, el usuario, el cliente estén de acuerdo en lo que se necesita, así como también la solución que se propone para dicha necesidad, y de esta forma, optimizar el riesgo y la incertidumbre en el desarrollo.

Este modelo se encarga del desarrollo de diseños, para que estos sean analizados, y prescindir de ellos a medida que se adhieran nuevas especificaciones, siendo ideal para medir el alcance del producto, pero no asegurando su uso real.

Dicho modelo se aplica cuando se define un conjunto de objetivos generales para el software a desarrollarse, sin delimitar detalladamente los requisitos de entrada, procesamiento y salida. Es decir, cuando no se está seguro de la eficacia de un algoritmo, de la adaptabilidad del sistema o de la forma en que interactúa el hombre y la máquina. En definitiva, este modelo se encarga de ayudar a entender cuál será el resultado de la construcción cuando los requisitos estén satisfechos.

4.1.1. FASES DEL MODELO DE PROTOTIPOS

Las fases que comprende el método de desarrollo orientado a prototipos son:

- **Investigación preliminar**: determina el problema y su ámbito, la importancia y sus efectos potenciales sobre la organización por una parte y, por otro lado, identificar una idea general de la solución para realizar

un estudio de factibilidad que determine la factibilidad de una solución software.

- **Definición de los requerimientos del sistema**: El objetivo de esta etapa es registrar todos los requerimientos y deseos que los usuarios tienen en relación al proyecto bajo desarrollo.
- **Diseño técnico**: El sistema debe ser rediseñado y documentado según los estándares de la organización, para ayudar a las mantenciones futuras. Esta fase de diseño técnico tiene dos etapas: por un lado, la producción de una documentación de diseño que especifica y describe la estructura del software, el control de flujo, las interfaces de usuario y las funciones y, como segunda etapa, la producción de todo lo requerido para promover cualquier mantención futura del software.
- **Programación y prueba**: Es donde los cambios identificados en el diseño técnico son implementados y probados para asegurar la corrección y completitud de los mismos con respecto a los requerimientos.
- **Operación y mantención**: La instalación del sistema en ambiente de explotación.

Estas fases se resumen tal y como en la **Figura 15**.

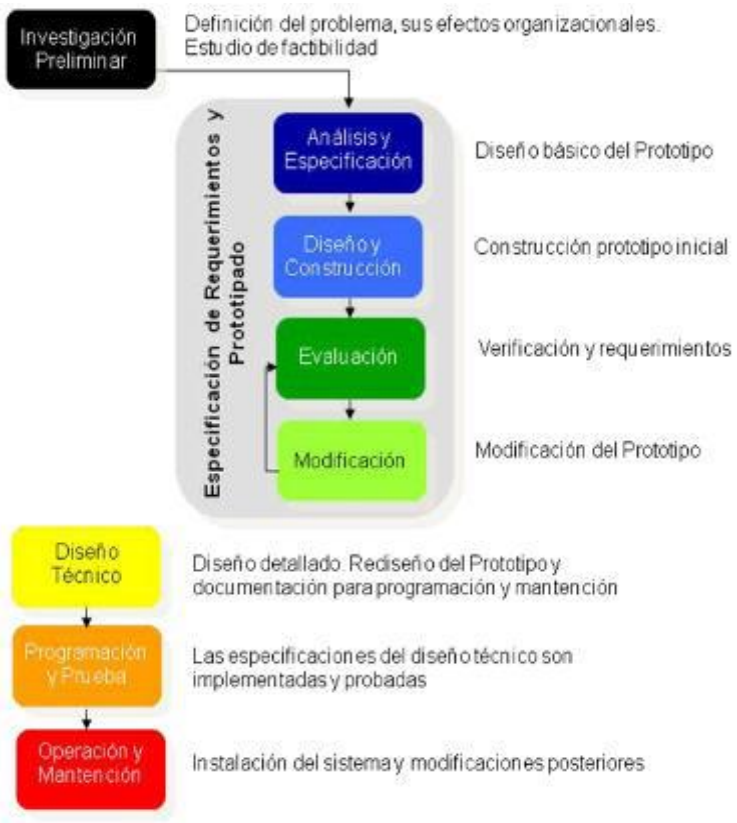


FIGURA 15 - FASES DEL MODELO DE PROTOTIPO

4.1.2. VENTAJAS DEL MODELO DE PROTOTIPOS

Las ventajas que ofrece este modelo son:

- Útil cuando se conoce los objetivos generales para el software, no identifica los requisitos detallados de entrada, procesamiento o salida.
- Proporciona un mejor enfoque cuando se está inseguro de la eficacia del algoritmo, de la adaptabilidad de un sistema operativo, o de la forma que debería tomar la interacción humano-máquina.

4.1.3. DESVENTAJAS DEL MODELO DE PROTOTIPOS

Como toda ventaja, tiene sus desventajas que mencionaré a continuación.

La visualización del funcionamiento de la primera versión del prototipo puede generar desilusión, pues el sistema aún no ha sido construido y finalizado, sino solo desarrollo, como su nombre indica, un prototipo construido con "plastilina y alambres".

El amplio margen de construcción de prototipos puede provocar descuidos en el compromiso de calidad y de mantenimiento.

4.2. LENGUAJE DE MODELADO - UML

El lenguaje de modelado utilizado a lo largo del proyecto será UML, cuyas siglas son "Unified Modeling Language" o "Lenguaje Unificado de Modelado".

Se trata de un estándar que se ha adoptado a nivel internacional por numerosos organismos y empresas para crear esquemas, diagramas y documentación relativa a los desarrollos de software (programas informáticos).

UML es ante todo un lenguaje. Un lenguaje proporciona un vocabulario y unas reglas para permitir una comunicación. En este caso, este lenguaje se centra en la representación gráfica de un sistema, que nos indica cómo crear y leer los modelos, pero no dice cómo crearlos, siendo esto último el objetivo de las metodologías de desarrollo, en este caso, el Modelo de Prototipos.

4.2.1. OBJETIVOS

Los objetivos de UML son muchos, pero se pueden sintetizar sus funciones:

- **Visualizar**: UML permite expresar de una forma gráfica un sistema de forma que otro lo puede entender.
- **Especificar**: UML permite especificar cuáles son las características de un sistema antes de su construcción.
- **Construir**: A partir de los modelos especificados se pueden construir los sistemas diseñados.

- **Documentar**: Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión.

4.2.2. CLASES DE BLOQUES DE CONSTRUCCIÓN

Un modelo UML está compuesto por tres clases de bloques de construcción:

- **Elementos**: Los elementos son abstracciones de cosas reales o ficticias (objetos, acciones, etc.).
- **Relaciones**: relacionan los elementos entre sí.
- **Diagramas**: Son colecciones de elementos con sus relaciones.

Definida cada una de las partes que se van a utilizar para poder llevar a cabo el proceso de despliegue de este proyecto, la metodología y el lenguaje de modelado, se definirán a continuación, los recursos que van a ser necesarios.

5. RECURSOS

Los recursos hardware y software que se va a necesitar para llevar a cabo este proyecto son los que se detallarán.

5.1. RECURSOS HARDWARE

El proyecto se llevará a cabo en un portátil Samsung cuyas características de composición expongo a continuación.

La aplicación que se desplegará en la nube para producción, solo se ha necesitado hacer uso de un portátil y un periférico (ratón), en este caso, de un Samsung cuyas características son:

- Memoria RAM: 5,5 GB
- Procesador: Intel® Core™ i5-3317U CPU @ 1.70GHz × 4
- Gráficos: Intel® Ivybridge Mobile
- Tipo de Sistema Operativo: 64 bits
- Sistema operativo: Ubuntu 14.04 LTS
- Disco: 124,6GB

En este apartado añadiría, además, el dispositivo GPS, pues la idea inicial y el fundamento de nuestra simulación se basa en esta tecnología.

Al no vamos a disponer de esta tecnología, haremos la simulación del Sistema de Posicionamiento Global haciendo uso de **ficheros .kml**, que son archivos que contienen datos geográficos. Los explicaré con mayor detenimiento en el siguiente apartado, Recursos Software.

5.2. RECURSOS SOFTWARE

Las herramientas software que se harán uso, estarán bajo el soporte del Sistema Operativo Ubuntu 14.05. Dichas herramientas son:

5.2.1. SISTEMA OPERATIVO

Puede ser definido como un conjunto de programas especialmente hechos para la ejecución de varias tareas, en las que sirve de intermediario entre el usuario y la computadora. Este conjunto de programas que manejan el hardware de una computadora u otro dispositivo electrónico, provee de rutinas básicas para controlar los distintos dispositivos del equipo y permite administrar, escalar y realizar interacción de tareas, teniendo además como función, administrar todos los periféricos de una computadora. Es el encargado de mantener la integridad del sistema.

Por tanto, podemos decir que el sistema operativo es el programa más importante de la computadora, y por ello, la elección del mismo ha sido la versión 14.05 de Ubuntu.

5.2.1.1. UBUNTU

Está basado en GNU/Linux y se distribuye como software libre, incluyendo su propio entorno de escritorio denominado Unity.

Su composición está constituido por un conjunto de software, normalmente distribuido bajo una licencia libre o de código abierto.

Como añadido, he trabajado con este sistema operativo durante mucho tiempo, facilitándome de este modo, el uso de otras herramientas necesarias para que pueda llevar a cabo la constante construcción del proyecto.

5.2.2. EDITOR WEB

Para la creación de la aplicación web se necesitará un editor web, que sea compatible con las diversas tecnologías que se va a utilizar para facilitar y optimizar el desarrollo.

Estudiando los distintos editores webs existentes, la elección del mismo ha sido **Sublime Text**.

Sublime Text es un **editor de código multiplataforma**, ligero y con pocas concesiones a las florituras. Es una herramienta concebida para programar sin distracciones. Su interfaz de color oscuro y la riqueza de coloreado de la sintaxis, centra la atención completamente.

Este editor permite tener varios documentos abiertos mediante pestañas, e incluso emplear varios paneles para aquellos que utilicen más de un monitor. Dispone de modo de pantalla completa, para aprovechar al máximo el espacio visual disponible de la pantalla. Y para navegar por el código cuenta con Minimap, un panel que permite moverse por el código de forma rápida.

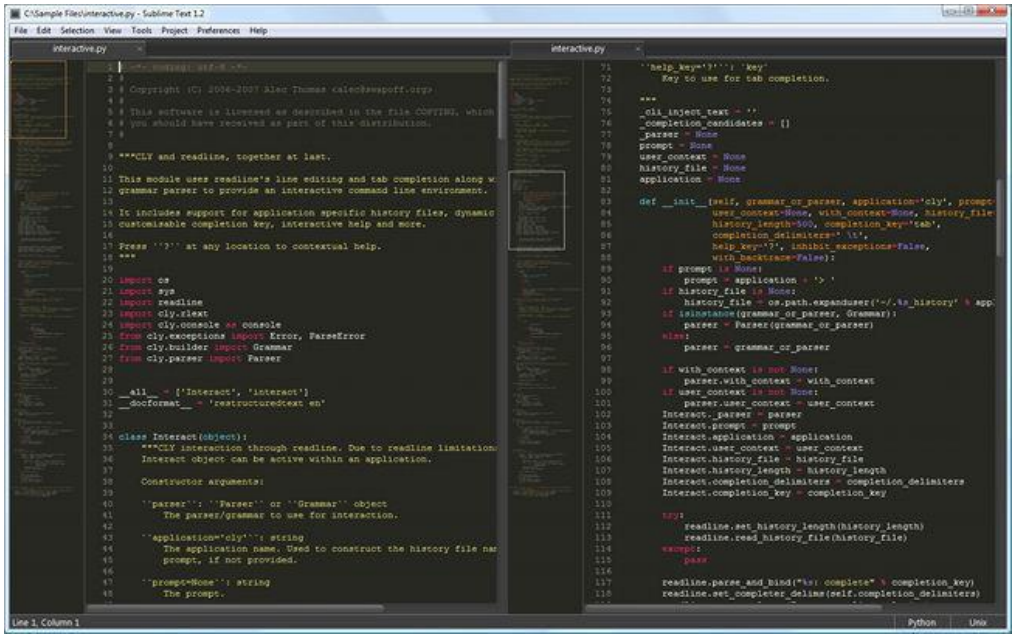


FIGURA 16 - EDITOR SUBLIME TEXT

El sistema de resaltado de sintaxis de Sublime Text soporta un gran número de lenguajes (C, C++, C#, CSS, D, Erlang, HTML, Groovy, Haskell, HTML, Java, JavaScript, LaTeX, Lisp, Lua, Markdown, Matlab, OCaml, Perl, PHP, Python, R, Ruby, SQL, TCL, Textile and XML).

El programa dispone de auto-guardado, muchas opciones de personalización, cuenta con un buen número de herramientas para la edición del código y automatización de tareas. Soporta macros, Snippets y auto completar, entre otras funcionalidades. Algunas de sus características son ampliables mediante plugins.

5.2.3. EDITOR UML

Para llevar a cabo las distintas fases del proyecto, es necesario la creación de múltiple diagramas, que ayudarán a separar y a entender cada uno de los aspectos que comprende la aplicación. StarUML permitirá que se pueda hacer esto.

5.2.3.1. STARUML

Es una herramienta para el moldeamiento en los estándares UML, que como ya he comentado con anterioridad, es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. Con él se creará:

- **Diagramas de clases**: muestran las diferentes clases que componen un sistema y que se relacionan unas con otras.
- **Diagramas de secuencia**: muestran un intercambio de mensajes en un momento dado. Dan énfasis en el orden y el momento en que se envían los mensajes a los objetos.
- **Diagramas de estado**: muestran los diferentes estados de un objeto durante su vida y lo estímulos que provocan los cambios de estado en un objeto.
- **Diagrama de Relación Entidad**: Muestran el diseño conceptual de las aplicaciones de bases de datos.

5.2.4. EDITOR DE BASES DE DATOS

La creación y la administración de la Base de Datos, se hará a través del programa MySQL Workbench.

5.2.4.1. MYSQL WORKBENCH

Es una herramienta visual de diseño de bases de datos que integra desarrollo de software, administración y diseño de bases de datos, creación y mantenimiento para el sistema de base de datos MySQL.

5.2.5. EDITOR DE TEXTO

La memoria del proyecto se ha ido elaborando haciendo uso de **Microsoft Office Word 2007**.

5.2.5.1. MICROSOFT OFFICE WORD 2007

Microsoft Office Word 2007 es uno de los programas más populares que forman parte de Microsoft Office. Consiste en un procesador de textos que incluye un corrector ortográfico, diccionario de sinónimos y la posibilidad de trabajar con diversas fuentes.

5.2.6. CONTROL DE VERSIONES

El almacenamiento de las distintas versiones que se han ido implementando durante la fase de desarrollo, se ha hecho a través del control de versiones llamado **Git**. La utilización de esta herramienta permite que quede registrado cualquier cambio que se haga en el proyecto, y por tanto, su posterior visualización, pudiendo acceder al mismo cualquier persona, en este caso los tutores, para su continua evaluación.

5.2.6.1. GIT

Es un software de control de versiones diseñado por Linus Torvalds. Y la pregunta es ¿qué es control de versiones? Pues bien, es la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo. O lo que es lo mismo, es lo que haces cuando subes y actualizas tu código en la nube, o le añades alguna parte o simplemente le editas cosas que no funcionan como deberían o al menos no como tú esperarías.

Algunas de las **características** más importantes de Git son:

- Rapidez en la gestión de ramas, debido a que Git nos dice que un cambio será fusionado mucho más frecuentemente de lo que se escribe originalmente.
- Gestión distribuida: Los cambios se importan como ramas adicionales y pueden ser fusionados de la misma manera como se hace en la rama local.
- Gestión eficiente de proyectos grandes.
- Almacenamiento periódico en paquetes.

5.2.7. HERRAMIENTAS Y TECNOLOGÍAS

A continuación se explicarán las distintas herramientas y tecnologías utilizadas en la realización de este proyecto.

5.2.7.1. FRAMEWORK

Hace referencia a una estructura software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación.

Los **objetivos principales** que persigue son: acelerar el proceso de desarrollo, reutilizar código ya existente, y promover buenas prácticas de desarrollo como el uso de patrones.

De todos los **frameworks** que existen para poder desarrollar la aplicación web:

- **Ruby on Rails**: Framework MVC basado en Ruby, orientado al desarrollo de aplicaciones web.
- **CodeIgniter**: Poderoso framework PHP liviano y rápido.
- **Kohana**: Un fork de CodeIgniter, Gracias a Samuel por mencionarlo en los comentarios.
- **Django**: Framework Python que promueve el desarrollo rápido y el diseño limpio.
- **Cake PHP**: Framework MVC para PHP de desarrollo rápido.
- **Zend Framework**: Framework para PHP 5, simple, claro y open-source.
- **Yii**: Framework PHP de alto rendimiento basado en componentes.
- **Pylons**: Framework web para Python que enfatiza la flexibilidad y el desarrollo rápido.
- **Catalyst**: Framework para aplicaciones web MVC elegante.
- **Symfony**: Framework full-stack.
- **Turbo Gears**: Próxima generación construido sobre Pylons.

Me decanté por Ruby on Rails, que como su propio nombre indica, está escrito en el lenguaje de programación Ruby. De ambos explicaré sus principales características y, por tanto, los motivos por los que lo he cogido.

5.2.7.2. RUBY

Ruby es un lenguaje de programación con licencia de software libre y creado por Yukihiro "Matz" Matsumoto. Es un lenguaje de scripts, multiplataforma y netamente orientado a objetos. La primera versión fue liberada en 1995, heredando características de lenguajes como: Perl, Smalltalk, Eiffel, Ada y Lisp. Como indica su propio autor, es un lenguaje

"aparentemente sencillo pero internamente complejo". Esto quiere decir que, mientras más nos abstraemos en el paradigma orientado a objetos, notaremos realmente la complejidad del lenguaje.

Ruby fue diseñado para un desarrollo rápido y sencillo, y entre las características del lenguaje se encuentran:

- Posibilidad de hacer llamadas directamente al sistema operativo.
- Muy potente para el manejo de cadenas y expresiones regulares.
- No se necesita declarar las variables.
- La sintaxis es simple y consistente.
- Gestión de memoria automática.
- Todo es un objeto.
- Métodos Singleton.

Ahora, ¿qué es Rails? ¿Y por qué le encantó a la comunidad de Ruby su sintaxis, ya que es legible y parecida al pseudo-código?

5.2.7.3. RUBY ON RAILS

Ruby on Rails es un **framework** que nos facilita la construcción de grandes aplicaciones web, y que a su vez, estas son de código abierto creado por David Heinemeier Hansson, liberando la primera versión en Julio del 2004.

Rails maneja el paradigma del **MVC** (Model-View-Controller), comúnmente conocido como, Modelo-Vista-Controlador, que nos permite dar ciertas configuraciones de manera libre en el controlador, para darle lógica a nuestras vistas y permitirnos manejar información de una base de datos, mediante consultas al **Active Record** en base a los atributos o campos de nuestro modelo ya creado.

Su filosofía está basada en los siguientes principios:

- **Don´t Repeat Yourself (DRY):** Nos indica que lo que ya está hecho no tiene porqué volver a hacerse.
- **Convention Over Configuration (COC):** "Convención antes que Configuración. Con esto el framework nos dice: oye he notado que tú siempre usas esto de ésta forma, o veo que siempre tienes que configurar esto de aquí, ¿por qué no hacemos una cosa? yo te doy todo esto configurado, si tú respetas esta configuración, te ahorras tiempo, y si no quieres hacerlo, no hay problema no me molesto,

Una de las cosas más interesantes de este framework, es que nos permite combinar lenguaje de Ruby con **HTML** o HTML5, mediante archivos con la extensión"_.html.erb_" en las vistas del controlador, lo que nos facilita el manejo de distintas funciones, variables o métodos dentro de nuestra aplicación.

Al igual que muchos otros frameworks, Rails nos permite instalar librerías y bibliotecas (llamadas **Gemas**) desde la consola de Rails, o desde el mismo símbolo del sistema (en Ubuntu).

ACTIVERECORD

El Active Record es uno de los temas más amplios que dispone Rails. Se podría decir que Active Record, es una solución de acceder a los datos de una base de datos, en la que cada tabla de la base de datos es una clase, por lo que cada fila es asociada con objetos. Cuando se crea un objeto, se añade una fila a la tabla de la base de datos. Cuando se modifican los atributos del objeto, se actualiza la fila de la base de datos. De modo que, se puede decir que el Active Record nos proporciona las siguientes ventajas:

- Manejo de entidades objeto.
- Las acciones del CRUD (Insertar, leer, modificar y eliminar) están encapsuladas, así que se reduce el código y se hace más fácil de comprender.
- Código fácil de entender y mantener.
- Se reduce el uso de código SQL considerablemente, lo que implica cierta independencia del manejador de la base de datos que se usa.

GEMAS

Otro punto importante que caracteriza a Rails y que presenta una ventaja a la hora de desarrollar una aplicación web son las gemas. Una gema es la manera en que Ruby permite distribuir programas, módulos o librerías que extienden funcionalidad, casi siempre específica, y que hacen **no tener que repetirnos (*Don't Repeat Yourself*)**, volviendo más fácil nuestro flujo de desarrollo. Por ejemplo, hay gemas para Rails que se encargan de la autenticación de usuarios, como es el caso de Devise. O Carrierwave, una gema que se encarga de la subida de archivos.

Para cada casi acción que se realiza repetitivamente a la hora de programar, existe una gema que lo vuelve todo más fácil.

5.2.7.4. SQLITE | POSTGRESQL

En el inicio del desarrollo de la aplicación, o lo que es lo mismo, creando el proyecto Rails, en este viene por defecto la biblioteca **SQlite**, pero he cambiado el mismo por **PostgreSQL**, pues el despliegue de la aplicación web se decidió

hacerlo a través de la plataforma **Heroku,** cuya base de datos que soporta es ésta. Además de ser la base de datos más utilizada a día de hoy.

SQLITE

SQLite es una librería escrita en lenguaje C que implementa un motor de base de datos para SQL92 empotrable, o lo que es lo mismo, implementa un manejador de base de datos SQL embebido, donde los programas que utilicen esta librería pueden tener acceso a una base de datos SQL, sin tener que ejecutar un programa de RDBMS separado.

SQLite es Software Libre y, por lo tanto, el código fuente es del dominio público y licencia GPL, cuyas **características** principales son:

- Su completo soporte para tablas e índices en un único archivo por base de datos.
- Soporte transaccional.
- Rapidez.
- Escaso tamaño.
- Y completa portabilidad

POSTGRESQL

PostgreSQL es una base de datos relacional open-source, cuyo creador, Michael Stonebraker inició este proyecto para solucionar problemas existentes en los 80's.

MySQL fue por mucho tiempo el motor más popular, pero a día de hoy, es propiedad de Oracle y esto limita su evolución. Sin embargo, PostgreSQL es *gratuito y libre*, ofreciendo además, una gran cantidad de opciones avanzadas.

Las características y, por ende, las **aportaciones** que ofrecen son:

- Control de concurrencias multiversión o MVCC: este método agrega una imagen del estado de la base de datos a cada transacción, permitiendo de este modo, hacer transacciones eventuales consistentes, lo que ofrece grandes ventajas en el rendimiento.
- Host-Standby: permite que los clientes hagan búsquedas en los servidores mientras están en modo de recuperación o espera. De esta manera, se puede hacer tareas de mantenimiento o recuperación sin bloquear completamente el sistema.
- Aporta mucha flexibilidad a los proyectos. Por ejemplo, permite definir funcionalidades personalizadas por medio de varios lenguajes, como son: PL/pgSQL, PL/Tcl, PL/Perl…
- Está disponible para muchas plataformas ofreciendo el código fuente desde el sitio oficial.
- Y permite desarrollar bases de datos relacionales robustas y eficientes

5.2.7.5. LENGUAJES DE SISTEMAS DE POSICIONAMIENTO

Como se había mencionado en el inicio de la memoria, específicamente en el apartado de Introducción, el GPS (Sistema de Posicionamiento Global) es un dispositivo que permite fijar a escala mundial la posición de un objeto, una persona, un vehículo o una nave con una precisión casi milimétrica, a cualquier hora y desde cualquier lugar.

Para conseguirlo, el receptor se comunica con una constelación de 24 satélites a unos 20.200 km y 5 estaciones repartidas en toda la superficie terrestre, de los que reciben datos de posición y hora.

De esta manera, en la pantalla se dibuja un mapa con la posición exacta del objeto indicando calles, rutas, ríos y accidentes geográficos del terreno a todo color.

Este dispositivo es un recurso que no se va a utilizar directamente debido a los costes asociados y, en su lugar, se va a realizar una simulación de dicho dispositivo mediante el uso de ficheros en formato KML como resultado de un estudio previo.

KML

KML (Keyhole Markup Language) es un lenguaje de marcado basado en XML para representar datos geográficos en tres dimensiones en un navegador terrestre, como Google Earth, Google Maps (Web y aplicación móvil). Fue desarrollado para ser manejado con Keyhole LT, precursor de Google Earth, y su gramática contiene muchas similitudes con la de GML (Geography Markup Language).

Los ficheros con extensión KML utilizan una estructura compuesta por etiquetas con atributos y elementos anidados como se detalla en la **Figura 17**.

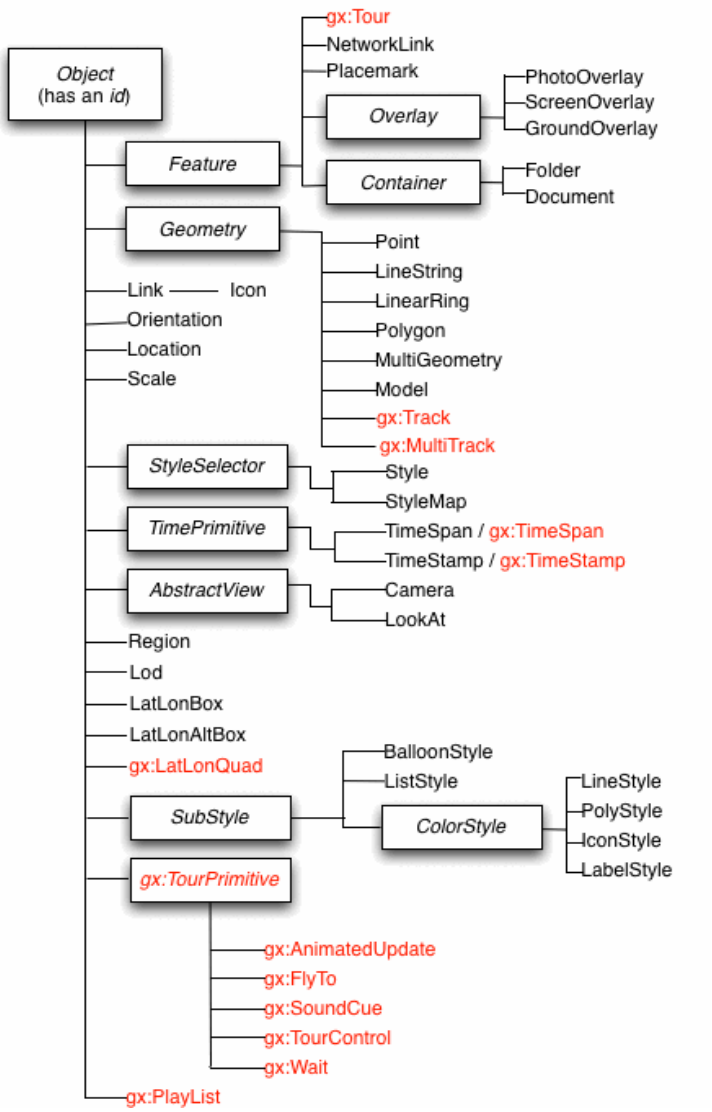


FIGURA 17 - ESTRUCTURA BÁSICA FICHERO KML

El modelo más simple de documento KML es el que se puede crear directamente en Google Earth; es decir, no es necesario editar ni crear ningún archivo KML en un editor de texto. Las marcas de posición, las superposiciones de suelo, las rutas y los polígonos se pueden crear directamente en Google Earth.

Un fichero KML especifica una característica (un lugar, una imagen o un polígono) para Google Earth. Contiene título, una descripción básica del lugar, sus coordenadas (latitud y longitud) y alguna otra información.

Un documento KML de ejemplo podría ser:

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
 <Placemark>
  <name>Oviedo</name>
  <description>Ciudad de Oviedo, capital de Asturias</description>
```

```
  <Point>
    <coordinates>-5.843868,43.360758,0</coordinates>
  </Point>
 </Placemark>
</kml>
```

POR QUÉ UTILIZAR FICHEROS KML COMO FORMATO DE ARCHIVO DE DATOS GEOGRÁFICOS, EN LUGAR DE GPX, PLT O WPT

Empezaré explicando brevemente cada uno de estos formatos y luego concluiré con la razón de por la que se determinó el uso de los ficheros con formato KML.

GPX

GPS eXchange Format (Formato de Intercambio GPS), es un esquema XML pensado para transferir datos GPS entre aplicaciones. Se puede usar para describir puntos (waypoints), recorridos (tracks), y rutas (routes).

Es el formato estándar más usado y su estructura tiene la siguiente forma:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<gpx ...>
Metadatos
 <metadata> ... </metadata>
Datos
Ejemplos: Track                    Waypoint
 <trk>                          <wpt lat="#" lon="#">
  <trkseg>                        <ele>#</ele>
   <trkpt lat="#" lon="#">            <name>...</name>
    <ele>#</ele>                     ...
   </trkpt>                        </wpt>
   <trkpt ...>                    <wpt ...>
    ...                        ...
   </trkpt>                        </wpt>
  </trkseg>
  <trkseg>
   <trkpt ...>
    ...
```

```
  </trkpt>
 </trkseg>
 ...
</trk>
Fin del fichero
</gpx>
```

PLT

Es un archivo plotter en formato de track, basado en vectores y creado por el software de dibujo de AutoCAD.

Se puede imprimir usando un plotter que imprime imágenes mediante líneas en lugar de puntos, basado en el formato HPGL (Hewlett & Packard Graphics Language), siendo este un lenguaje de descripción de gráficos, diseñado originariamente para el control de Plotters (trazadores gráficos).

La **estructura** y ejemplo de un tipo de archivo con este formato es el que veremos a continuación.

```
OziExplorer Track Point File Version 2.1
WGS84
Altitude is in Feet
0,2,16711680,ACTIVE LOG,0,0,2,8421376440
  36.448688,  -6.137883,1, -23,36786.4017477, 17-sep-00, 9:38:31
  36.448648,  -6.137863,0, -23,36786.4017593, 17-sep-00, 9:38:32
  36.448498,  -6.137813,0, -23,36786.4018056, 17-sep-00, 9:38:36
```

Se trata de un archivo escrito en formato texto que puede leerse con cualquier procesador de texto, incluso manipularse.

¿Cuál es el significado de estas líneas?
Línea 1: **OziExplorer Track Point File Version 2.1**
Información que indica que tipo de archivo es y la versión del mismo.

Línea 2 **WGS84**
Datum en el que se está expresando las coordenadas, constituyendo cada uno de los puntos del track de este archivo.

Línea 3: **Altitude is in feet**
La altitud está en pies (no en metros).

Línea 4: **0,2,16711680,ACTIVE LOG ,0,0,2,8421376440**
Muchos campos diferentes:

- Campo 1: siempre cero (0).
- Campo 2: anchura de la línea del track en la pantalla. En este caso es de 2 pixels.

- Campo 3: Color del track (expresado en valor RGB).
- Campo 4: Descripción o comentario del track (no permite comas).
- Campo 5: Valor para saltar puntos del track. Reduce el número de puntos del track que se dibujan. Normalmente tiene el valor de 1.
- Campo 6: Número de puntos de track que constituyen el track.

Datos del Track:

- Una línea por cada punto del track.
- Cada campo está separado por una coma.
- Si un campo no es necesario o no se dispone de información, se colocan varias comas seguida (por ejemplo ,,).
- Campo 1: Latitud - expresada en grados decimales. Si es negativo, está en el hemisferio Sur.
- Campo 2: Longitud expresada en grados decimales. Si es negativo es que va hacia el Oeste.
- Campo 3: Línea de código, - 0 si es normal, 1 si representa una discontinuidad en la línea de track.
- Campo 4: Altitud en pies (-777 si no es válido).
- Campo 5: Fecha y hora codificada.
- Campo 6: Fecha.
- Campo 7: Hora.

WPT

Es un fichero con formato de Waypoints. Su nombre completo es WordPerfect Template y ha sido creado por Corel Corporation (De la abreviatura "Co wpland Re búsqueda L aboratory", empresa de software canadiense que se especializa en el procesamiento de gráficos).

Un ejemplo de la estructura de este tipo de archivo es el siguiente.

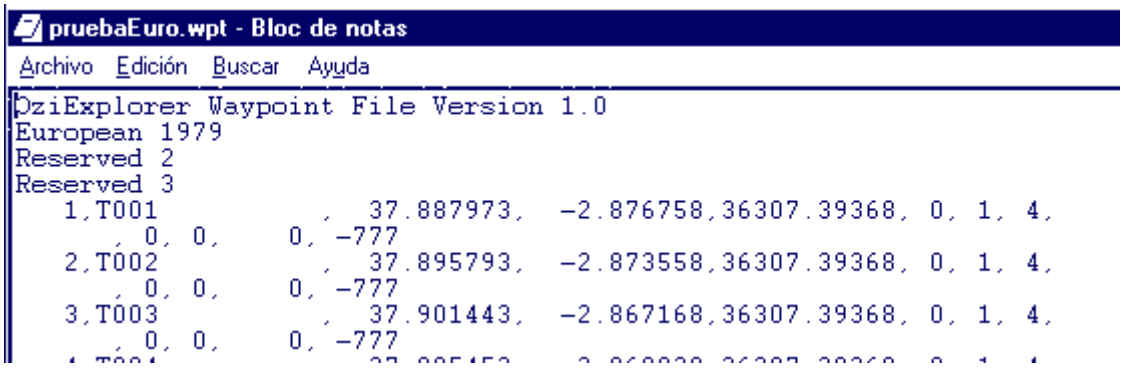


FIGURA 18 - ESTRUCTURA DE FICHERO WPT

Se ha elegido **el formato de ficheros de datos geográficos KML** debido a que, junto con el formato GPX, es uno de los más populares. Fue expresamente desarrollado para la herramienta Google Earth, herramienta utilizada en este proyecto y por lo tanto me facilita mostrar las rutas de los animales en dicho navegador terrestre y, en su defecto, en Google MAps. Otra ventaja de este formato es que estos **ficheros** suelen **distribuirse comprimidos (ficheros .kmz)**, donde se pueden incluir archivos de imágenes y otros recursos asociados al mapa.

5.2.7.6. CLOUD COMPUTING: HEROKU

El Cloud Computing o computación en la nube es un modelo de prestación de servicios de negocio y tecnología, que permite al usuario acceder a un catálogo de servicios estandarizados y responder a las necesidades de su negocio, de forma flexible y adaptativa, en caso de demandas no previsibles o de picos de trabajo, pagando únicamente por el consumo efectuado. Las ventajas de este modelo de negocio, frente a modelos mainframe de gestión interna o a través de contratos de outsourcing de servidores, son:

- Agilidad, escalabilidad y elasticidad de los servicios.
- Costes asociados a la gestión y al mantenimiento de los servicios.
- Mantenimiento y Rendimiento de los sistemas.
- Del mismo modo, existen diferentes tipos de nube, que dependen de las necesidades de los clientes. Estos Tipos son:
- Public cloud (nube pública). Es un tipo de cloud mantenida y gestionada por terceras personas no vinculadas con la organización.
- Private cloud (nube privada). Son gestionadas gestionada para un solo cliente y es la opción si se necesita alta protección de datos.

- Hybrid cloud (nube híbrida). Es un tipo de cloud que incluye los modelos explicados anteriormente.
- A continuación se detallan los servicios prestados actualmente, según las necesidades de los proyectos desarrollados:
- IaaS (Infrastructure as a Service), proporciona acceso a recursos informáticos situados en un entorno virtualizado, en particular, en hardware virtualizado.
- PaaS (Platform as a Service), proporciona funcionalidades que permite crear aplicaciones de software utilizando herramientas suministradas por el proveedor.
- SaaS (Software as a Service), se encuentra en la capa más alta y caracteriza una aplicación completa ofrecida como un servicio.

Un ejemplo de modelo cloud utilizado es durante la realización de este proyecto es Heroku. Se trata de un tipo de servicio cloud PaaS (explicado anteriormente), que proporciona una interfaz avanzada de gestión de recursos en la nube. No tiene servidores propios, sino que los contrata de la nube EC2 de Amazon y, un poco a la manera de Apple, ofrece conveniencia a cambio de restringir opciones de desarrollo. Mientras que Amazon EC2 es una plataforma totalmente abierta que sólo abstrae el hardware, donde el programador controla desde el sistema operativo hasta el lenguaje que quiere usar, Heroku decide y gestiona el sistema operativo (Debian), los servidores (Nginx, Varnish y Thin para proxy inverso, caché y servidor de aplicaciones Ruby, respectivamente), y también ofrece un menú selecto de opciones para programar en Ruby: puedes subir cualquier código que use rack, pero recomiendan sus propias "gemas" o paquetes nativos de Ruby. Además proporciona la opción de programar en otros lenguajes como Node.js, Java, Python, Clojure y Scala.

Una de las **ventajas** de **Heroku** es que al igual que Google APP Engine, es gratuito para aplicaciones de poco consumo. Esto quiere decir que puedes desarrollar y lanzar sin pagar, y sólo tienes que empezar a hacerlo cuando empieza crearse usuarios.

Por otro lado, todo se controla desde el terminal. Se puede habilitar un nuevo hosting usando una gema específica de Heroku. Los desarrolladores pueden desplegar su código con sólo usar su gestor de código fuente ($ git push master), y con ser un poco previsores, se pueden despreocupar de si su servicio aguantará los picos de demanda; en respuesta a cargas, el sistema puede desplegar procesos (que en Heroku se llaman "dynos") en menos de dos segundos, permitiendo gran escalabilidad a la aplicación, gracias a la contratación inmediata de dynos según las necesidades que se tengan.

5.2.7.7. HTML (HYPERTEXT MARKUP LANGUAGE)

Es lo que se conoce como "lenguaje de marcado", cuya función es preparar documentos escritos aplicando etiquetas de formato. Las etiquetas indican cómo se presenta el documento y cómo se vincula a otros documentos.

HTML se usa también para la lectura de documentos en Internet desde diferentes equipos gracias al protocolo HTTP, que permite a los usuarios acceder, de forma remota, a documentos almacenados en una dirección específica de la red, denominada dirección URL.

5.2.7.8. JAVASCRIPT

JavaScript es un **lenguaje de programación** que se utiliza principalmente para crear páginas web dinámicas.

Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario.

Técnicamente, JavaScript es un **lenguaje de programación interpretado**, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios.

A pesar de su nombre, JavaScript no guarda ninguna relación directa con el lenguaje de programación Java. Legalmente, JavaScript es una marca registrada de la empresa Sun Microsystems, como se puede ver en http://www.sun.com/suntrademarks/.

5.2.7.9. JQUERY

JQuery es una **librería JavaScript open-source**, que funciona en múltiples navegadores, y que es compatible con CSS3. Su objetivo principal es hacer la programación "scripting" mucho más fácil y rápida del lado del cliente. Con jQuery se pueden producir páginas dinámicas así como animaciones parecidas a Flash en relativamente corto tiempo.

Las **ventajas** que proporciona son:

- Es flexible y rápido para el desarrollo web.
- Viene con licencia MIT y es Open Source.
- Tiene una excelente comunidad de soporte.
- Tiene Plugins.
- Bugs son resueltos rápidamente.
- Excelente integración con AJAX.

5.2.7.10. AJAX (ASYNCHRONOUS JAVASCRIPT AND XML)

AJAX es una técnica que permite la comunicación **asíncrona** entre un servidor y un navegador en formato **XML** mediante programas escritos en **JavaScript**.

El **principal objetivo** del AJAX, es intercambiar información entre el servidor y el cliente (navegadores) sin la necesidad de recargar la página. De esta forma, ganamos en usabilidad, experiencia y productividad del usuario final.

Las **ventajas** que nos proporciona son:

- Rapidez en las operaciones.
- Menos carga del servidor (menos transferencia de datos cliente/servidor).
- Menos ancho de banda.
- Soportada por la mayoría de navegadores.
- Interactividad, ya que el usuario no tiene que esperar hasta que lleguen los datos del servidor.
- Portabilidad.
- Usabilidad.
- Velocidad (Debido a que no hay que recargar la página nuevamente).

Ahora, como todo, también tiene sus **desventajas** y son:

1. Se pierde el concepto de "volver a la página anterior".
2. Problemas con navegadores antiguos.
3. No funciona si el usuario tiene desactivado el JavaScript en su navegador.
4. Se requieren conocimiento sobre las tecnologías que forman AJAX.
5. Problemas SEO, los buscadores no indexan la información recibida vía AJAX.

5.2.7.11. JSON (JAVASCRIPT OBJECT NOTATION)

Es un formato para el intercambios de datos, que describe los datos con una sintaxis dedicada que se usa para identificar y gestionar los datos. JSON nació como una **alternativa a XML**, su fácil uso en JavaScript ha generado un gran número de seguidores de esta alternativa. Una de las mayores **ventajas** que tiene el uso de JSON es que puede ser **leído por cualquier lenguaje de programación**. Por lo tanto, puede ser usado para el intercambio de información entre distintas tecnologías.

5.2.7.12. CSS (CASCADING STYLE SHEETS)

Es un **lenguaje de hojas de estilos** creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y, es imprescindible para crear páginas web complejas.

Separar la definición de los contenidos y la definición de su aspecto presenta numerosas **ventajas**, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo (también llamados "documentos semánticos"). Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y, permite visualizar el mismo documento en infinidad de dispositivos diferentes.

Al crear una página web, se utiliza en primer lugar el lenguaje HTML/XHTML para marcar los contenidos, es decir, para designar la función de cada elemento dentro de la página: párrafo, titular, texto destacado, tabla, lista de elementos, etc.

Una vez creados los contenidos, se utiliza el lenguaje CSS para definir el aspecto de cada elemento: color, tamaño y tipo de letra del texto, separación horizontal y vertical entre elementos, posición de cada elemento dentro de la página, etc.

5.2.7.13. MOCKUPS

Los Mock Ups son fotomontajes que permiten a los diseñadores gráficos y web mostrar al cliente cómo quedaran sus diseños, permitiendo ahorrar en gastos de impresión y de montajes debido a que se puede mostrar al cliente final una visualización lo más aproximada del diseño final de la aplicación.

La solución utilizada durante las fases de diseño de la aplicación ha sido Balsamiq Mockups.

5.2.7.14. BOOTSTRAP

Bootstrap es un framework originalmente creado por Twitter, que permite crear interfaces web con CSS y JavaScript, cuya particularidad es la de adaptar la interfaz del sitio web al tamaño del dispositivo en que se visualice. Es decir, el sitio web se adapta automáticamente al tamaño de un PC, una Tablet u otro dispositivo. Esta técnica de diseño y desarrollo se conoce como "responsive design" o diseño adaptativo.

El beneficio de usar responsive design en un sitio web, es principalmente que el sitio web se adapta automáticamente al dispositivo desde donde se acceda. Lo que se usa con más.

Los **beneficios** que nos aporta Bootstrap son:

- El ahorro de tiempo. No tenemos que empezar una página desde cero, sino que podemos pararnos sobre el código que nos aporta y empezar a desarrollar desde ahí.
- Es fácil de aprender.
- Posee soporte para los preprocesadores Less y Sass.
- Es fácil de modificar.
- Como se ha indicado, está pensado con el diseño móvil primero, con lo cual nuestro sitio va a escalar correctamente sin importar la pantalla que esté utilizando el visitante.
- Aporta un estilo base a todos los elementos HTML.
- Posee una documentación muy detallada y abundante, cosa que no ocurre con otros frameworks.
- Incluye una lista extensa de componentes que incluye: dropdowns, botones, barras de navegación, alertas, barras de progreso, etc.
- Y, actualmente existen muchos plugins de terceros, que amplían las características de Bootstrap.

6. ANÁLISIS

En esta etapa lo que se pretende es tener un concepto claro de las necesidades, problemas y requisitos del usuario.

Para ello, se requiere tener un conocimiento claro acerca del sistema, para poder profundizar y proponer una solución de la mejor forma posible.

6.1. REQUISITOS DEL SOFTWARE

En primer lugar se apreciará la descripción de los requerimientos del sistema, seguido del modelo del dominio, actores, diagramas de casos de uso y diagramas de secuencia.

6.1.1. DESCRIPCIÓN

La aplicación estará alojada en *la nube* (servidor), haciendo uso de la plataforma **Heroku** (más, ir al Apartado 5.2.7.5.), cuya función principal será almacenar y gestionar los datos.

Realizar el seguimiento de una mascota va a ser el principal pilar del proyecto, pues es el objetivo que se quiere conseguir.

Este seguimiento permitirá al propietario del perro visualizar su ubicación cuando un paseador lo esté paseando.

Cuando un usuario se registra en el sistema, directamente pasa a tener el rol de propietario, a menos que quiera convertirse en paseador, que lo indicará en su perfil.

Convertirse en paseador no significa que deje de ser propietario, sino todo lo contrario, este último rol está implícito, pues un paseador también puede añadir sus propios perros al igual que un propietario.

El propietario, ya registrado en el sistema, podrá registrar a sus perros y contratar a un paseador, pues es éste actor quien se encargue de pasear a los perros del propietario y, por tanto, el que va a hacer "click" en el botón del dispositivo GPS si lo hubiese. En nuestro caso hará "click" en un botón de la aplicación web que simulará dicho dispositivo.

A partir de ahí, ambos, paseador y propietario, podrán visualizar dónde está su mascota, guardar la trayectoria que ha hecho mientras realizaba su seguimiento, calcular el tiempo que ha durado (aproximadamente) su trayectoria, recuperar el cualquier momento las rutas almacenadas y visualizarlas, borrar las mismas, en caso de que no se quiera tenerlas.

El paseador, una vez contratado por un propietario, además de que pueda ver a sus perros, podrá visualizar a los perros que tiene que pasear. De este modo, podrá hacer "click" en el botón que simule el GPS como se acaba de mencionar.

La **simulación del GPS se hará haciendo uso de ficheros .kml**, donde se leerá y almacenará las coordenadas del mismo en la base de datos, para luego empezar enviar a la aplicación cada coordenada para su posterior pintado en el mapa. Más información en el apartado de implementación.

6.1.2. MODELO DEL DOMINIO

El modelo del dominio es el artefacto más importante que se crea durante el análisis, pues representa las clases conceptuales del mundo real, no de componentes software y se muestra en la **Figura 19.** En ella pueden verse las cinco principales clases del proyecto: Usuario, Perro, Paseo, Ruta y Localización, con sus respectivos atributos y relaciones.

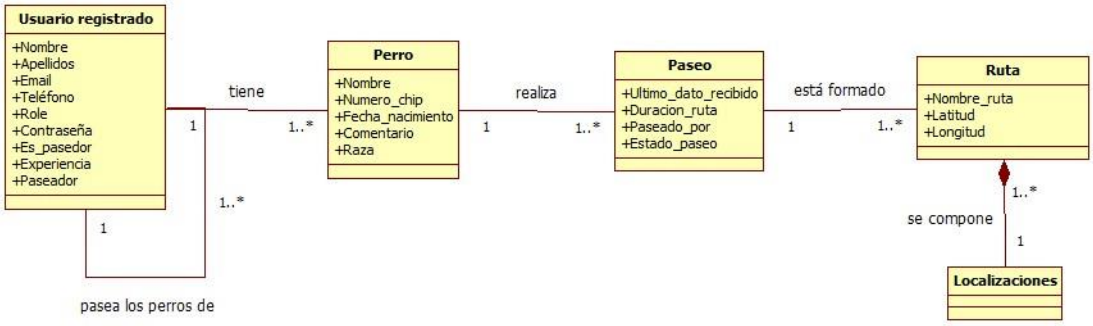


FIGURA 19 - MODELO DEL DOMINIO

6.1.3. ACTORES DEL SISTEMA

Vamos a encontrar tres tipos de actores en el sistema, cuyo rol y objetivos se especificarán a continuación: paseador, propietario y administrador, los tres son usuarios registrados.

Previo a esto, se mostrará en la **Figura 20** la relación que existe entre los diferentes actores.

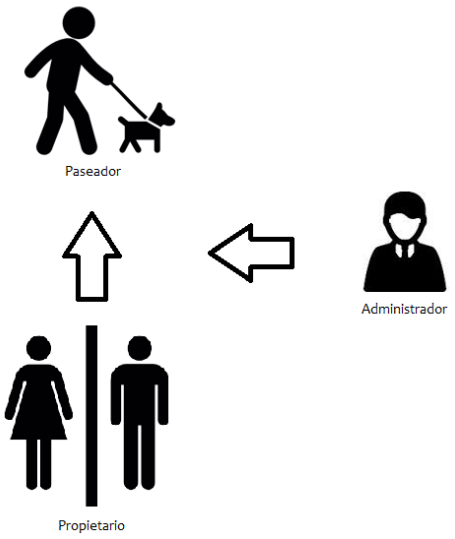


FIGURA 20 - ACTORES DEL SISTEMA

Cuando un usuario se da de alta en el sistema, su rol pasa a ser **Propietario**. Si el propietario quiere ser además **Paseador**, tendrá que especificarlo en su perfil.

La diferencia que existe entre el paseador y el propietario, es que el primero dará permiso al sistema para que se publiquen sus datos en el listado de paseadores, donde cualquier propietario podrá consultarlo cuando quiera contratar a uno de estos. Añadir que los paseadores tendrán un campo adicional que le permitirá exponer su experiencia como dicho rol. No obstante, se ha indicado, un paseador implícitamente también es un propietario.

El **Administrador** es un usuario registrado que se encargará de gestionar los usuarios que estén registrados en el sistema.

6.1.3.1. ROL DE CADA ACTOR

Actor	Tipo	Definición
Propietario	Principal	Usuario que se ha registrado y autenticado en el sistema. El propietario tiene la capacidad de insertar perros, añadiendo sus correspondientes datos; Contratar a un paseador; Visualizar la ubicación de su perro cuando está siendo paseado por un paseador; Visualizar el listado de paseadores que están registrados en el sistema, etc.
Paseador	Principal	Usuario que se ha registrado y autenticado en el sistema, da permiso para exponer su información como pública en la aplicación para que un propietario pueda contratarlo para que pasee a sus perros, haciendo uso para ello de la simulación GSP cuando va a iniciar un nueva ruta. También podrá ver un listado de paseos que ha realizado durante el transcurso de su contratación en la aplicación.
Administrador	Principal	Usuario que tendrá la capacidad de gestionar a los usuarios registrados en el sistema, pudiendo visualizarlos, borrarlos, crear nuevos y modificar sus roles.

6.1.3.2. OBJETIVOS DE CADA ROL

Actor	Objetivos	Acción
Propietario	Registrarse	Rellenando el formulario de registro, permite al usuario quedar registrado en el sistema, asignando como rol por defecto el de propietario.
	Editar perfil	Accediendo al formulario del usuario, éste puede indicar si, además de ser propietario, desea ser paseador. Si es así, rellenará un campo adicional que reportará información de su estado actual como paseador, y dará permiso al sistema para que su información sea pública.
	Gestionar perros	Capacidades **CRUD** para gestionar perros.
	Gestionar rutas de perros	Capacidades **CRUD** para gestionar las rutas de sus perros, concretamente, ver y borrar rutas.
	Visualizar paseadores	Pueden ver todos los paseadores que se encuentran registrados en el sistema para su posterior contratación del mismo.
Paseador	Visualizar perros que pasea	Contratado por un propietario, pueden ver los perros que va a pasear.
	Visualizar mis paseos	Pueden ver todos los paseos que están realizando in situ o, que han hecho con anterioridad.
	Iniciar nueva ruta	Crea nueva ruta cada vez que pasea un perro. Simula el dispositivo de geolocalización GPS.
Administrador	Gestionar usuarios registrados	Capacidades **CRUD** para gestionar los usuarios registrados en el sistema.

6.1.4. DIAGRAMAS DE CASOS DE USO

Una vez se han definido los diferentes actores con los roles y funciones que desempeñan cada uno de éstos, se detallarán a continuación el conjunto de diagramas de casos de usos (funcionalidades de la aplicación), utilizando el lenguaje UML (Unified Modeling Language) para representarlos.

PROPIETARIO

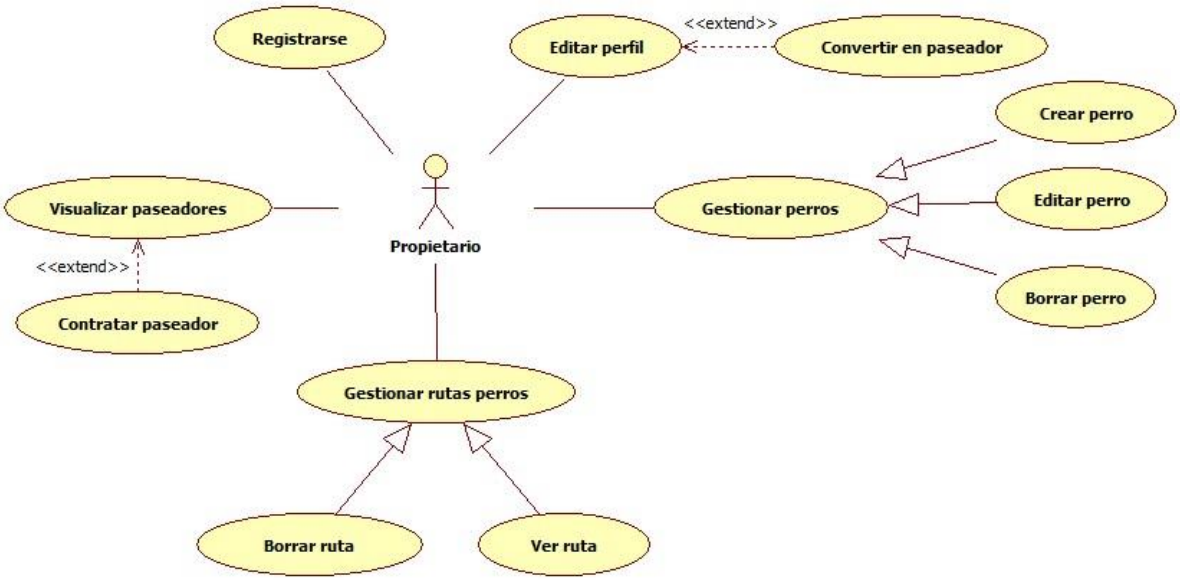


FIGURA 21 - CASOS DE USO DEL PROPIETARIO

PASEADOR

FIGURA 22 - CASOS DE USO DEL PASEADOR

ADMINISTRADOR

FIGURA 23 - CASOS DE USO DEL ADMINISTRADOR

6.1.5. LISTADO DE CASOS DE USO

En la siguiente tabla se asocia a cada caso de uso un identificar unívoco, mediante el que se puede realizar la correlación y poder ver el detalle en el ANEXO 1 de este documento.

Actor	Caso de uso	Identificador (ID)
Propietario	Registrarse	01
	Editar perfil	02
	Crear perro	03
	Editar perro	04
	Borrar perro	05
	Ver ruta	06
	Borrar ruta	07
	Visualizar paseadores	08
	Contratar paseador	09
Paseador	Ver perros que pasea	10
	Ver mis paseos	11
	Iniciar nueva ruta	12
Administrador	Modificar rol usuario	13
	Borrar usuario registrado	14

Con el objetivo de explicar las funcionalidades más críticas e importantes de la aplicación, se mostrarán a continuación cada una de las mismas detallando la siguiente información:

- Actor principal involucrado en la acción.
- Personal involucrado o intereses.
- Descripción de la funcionalidad.
- Trigger, evento que desencadena la ejecución de la funcionalidad.

- Precondición, estado previo a la ejecución de la funcionalidad.
- Postcondicion, estado posterior a la ejecución de la funcionalidad.
- Flujo normal, detalle de cada tarea llevada a cabo por la funcionalidad.
- Flujo alternativo, detalle de cada tarea de control llevada a cabo.
- Excepción.
- Includes.
- Requisitos especiales.
- Notas.

EDITAR PERFIL

Nombre	Editar perfil	ID	02
Actor principal			
Propietario			
Personal involucrado o intereses			
• Propietario: El usuario registrado quiere actualizar sus datos personales almacenados en el sistema, pudiendo modificar a su vez su rol, convirtiéndose en paseador.			
Descripción			
El usuario actualiza sus datos personales en el sistema, dando permiso para ello, a través de su contraseña. Además, el usuario registrado, teniendo asignado a priori el rol de propietario, tiene la posibilidad de convertirse en paseador para poder pasear a los perros de otros propietarios registrados en el sistema.			
Trigger			
Hacer "click" en el enlace **Perfil.**			
Precondición			
• Estar dado de alta en la aplicación.			
Postcondición			
• El perfil del cliente queda modificado a sus intereses.			
Flujo normal			
1. Hace "click" en el enlace **Perfil.** **2.** El cliente modifica sus datos personales. **3.** Introduce contraseña para confirmar los cambios a realizar y hace "click" en el botón **Enviar**. **4.** El sistema valida los datos modificados. **5.** El sistema actualiza la información en el mismo.			
Flujo alternativo			
4.1. Si al validar los datos, el sistema detecta alguno incorrecto, se lo comunica al usuario volviendo al paso 2. **4.1.1.** Vuelve al paso 2 siguiendo nuevamente el flujo de forma normal.			
Excepción			
Includes			
Requisitos especiales			
Notas			

VER RUTA

Nombre	Ver ruta	ID	06
Actor principal			
Propietario			
Personal involucrado o intereses			
• Propietario: El cliente quiere ver la ruta que está realizando su perro cuando está siendo paseado por el paseador contratado o, quiere ver una ruta que ya ha sido finalizada.			
Descripción			
El usuario accede al listado de rutas de su perro para visualizar en el mapa la ruta que está realizando en el momento que está siendo paseado por el paseador o, que ha realizado con anterioridad.			
Trigger			
Hacer "click" en el enlace **Rutas.**			
Precondición			
• El perro tiene que estar registrado en el sistema y vinculado a la cuenta del usuario.			
Postcondición			
• El usuario visualiza la ruta del perro en el mapa.			
Flujo normal			
1. Hace "click" en el enlace **Perros**. **2.** Busca el perro en el listado de peros cuya ruta quiere visualizar en el mapa. **3.** Hace "click" en el enlace **Rutas.** **4.** El cliente visualiza la lista de rutas **5.** Selecciona una de ellas visualizándola en el mapa, estando dicha ruta finalizada o en progreso.			
Flujo alternativo			
Excepción			
Includes			
Requisitos especiales			
Notas			

VER PERROS QUE PASEA

Nombre	Ver perros que pasea	ID	10
Actor principal			
Paseador			
Personal involucrado o intereses			
• Paseador: El usuario desea ver qué perros tiene que pasear.			
Descripción			
Cuando el usuario ha sido contratado por un propietario, automáticamente se le asigna sus perros, visualizándolos en un listado aparte, separando sus perros de los que pasea.			
Trigger			
Hacer "click" en el enlace **Perros.**			
Precondición			
• Estar registrado en el sistema como paseador. • Haber sido contratado por un propietario.			
Postcondición			
• Ver los perros de los propietarios que han contratado al usuario.			
Flujo normal			
1. Hace "click" en el enlace **Perros.** **2.** Visualizar los perros que pasea y los suyos propios en caso de que los tuviera registrados en el sistema.			
Flujo alternativo			
Excepción			
2.* Si no ha sido contratado por ningún propietario, la tabla saldrá vacía.			
Includes			
Requisitos especiales			
Notas			

VER MIS PASEOS

Nombre	Ver mis paseos	ID	11
Actor principal			
Paseador			
Personal involucrado o intereses			
• Paseador: El usuario registrado quiere ver todos los paseos que está haciendo o que ha hecho con anterioridad.			
Descripción			
El usuario al clicar en el enlace "Mis paseos", podrá ver el listado de paseos que ha hecho con los perros que pasea, visualizando en el mismo el nombre del paseo, el tiempo que ha durado el paseo con cada perro, la hora inicio y fin del paseo, y el nombre del perro y del propietario.			
Trigger			
Hacer "click" en el enlace **Mis paseos**.			
Precondición			
• El usuario debe haber sido dado de alta en el sistema. • El usuario tuvo que haber sido contratado por un propietario. • El usuario tiene que tener asignado a los perros del propietario que le ha contratado.			
Postcondición			
• Visualiza los paseos que está realizando actualmente como los que ha hecho con anterioridad.			
Flujo normal			
1. Hace "click" en el enlace **Mis paseos**.			
Flujo alternativo			
Excepción			
Includes			
Requisitos especiales			
Notas			

INICIAR NUEVA RUTA

Nombre	Iniciar nueva ruta	ID	12
Actor principal			
Paseador			
Personal involucrado o intereses			
• Paseador: El usuario registrado quiere iniciar una nueva ruta del perro que está paseando.			
Descripción			
El usuario acciona el botón "Iniciar nueva ruta" cuando va a realizar un paseo con el perro de un propietario que le ha contratado. La función que va a desempeñar es la de simular el GPS.			
Trigger			
Accionar el botón **Iniciar nueva ruta**.			
Precondición			
• El usuario debe haber sido dado de alta en el sistema. • El usuario tuvo que haber sido contratado por un propietario. • El usuario tiene que tener asignado a los perros del propietario que le ha contratado.			
Postcondición			
• Crea una nueva ruta del perro que está paseando, visualizando dicha ruta en el mapa.			
Flujo normal			
1. Hace "click" en el enlace **Perros**. **2.** Busca al perro que va a pasear. **3.** Hace "click" en el enlace **Rutas** del perro. **4.** Hace "click" en el botón **Iniciar nueva ruta** para crear una nueva ruta.			
Flujo alternativo			
Excepción			
Includes			
Requisitos especiales			
Notas			

7. DISEÑO

Una vez obtenida toda la información en la fase de análisis, procederemos en el diseño a modelar la estrategia a seguir para la elaboración de la aplicación web para que soporte todos los requisitos comentados con anterioridad.

Comenzaremos detallando la arquitectura que vamos a utilizar. A continuación nos centraremos en nuestra aplicación, estudiando la estructura de los mockups y el modelo de despliegue.

Acto seguido, estudiaremos el diagrama de secuencia general que tendrá nuestro sistema y, la última sección que veremos en este apartado, será referente al estudio de la estructura de la base de datos que va a poseer nuestro sistema.

7.1. ARQUITECTURA

Como se ha mencionado, el primer punto que se puede apreciar en este apartado será el referente a la arquitectura de nuestra aplicación.

Durante la realización de este proyecto se estudiará el patrón Modelo Vista Controlador, ActiveRecord y el paradigma REST.

7.1.1. PATRÓN MVC (MODELO VISTA CONTROLADOR)

El patrón de arquitectura de software que se utiliza en nuestro sistema es el de Modelo Vista Controlador (MVC). Es un patrón que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos.

Es un patrón frecuentemente utilizado en aplicaciones web, donde la vista es la página HTML que se muestra al usuario final y el código que provee los datos dinámicos a la página; el modelo es el sistema de gestión de base de datos y la lógica del negocio; y el controlador es el responsable de recibir los eventos de entrada.

A continuación se detallan de forma más precisa cada uno de los componentes del patrón MVC:

- **Modelo**: Es la representación de la información que el sistema opera, por lo que gestiona todos los accesos a dicha información (consultas, actualizaciones…) e implementa también los privilegios de acceso que se han descrito en las especificaciones de la aplicación. Además, envía a la vista aquella parte de la información que en cada momento se solicita para que sea mostrada.

  Las peticiones de acceso o manipulación de información llegan al modelo a través del controlador.

- **Vista**: La vista presenta el modelo en un formato adecuado para interactuar. Generalmente se refiere a la interfaz de usuario, lo que requiere del modelo la información que debe representar como salida.
- **Controlador**: El controlador responde a eventos (generalmente acciones de los usuarios) e invoca peticiones al modelo cuando se hace alguna solicitud sobre la información (por ejemplo, editar un documento o un registro en una base de datos).

  También puede enviar comandos a su vista asociada si se solicita un cambio en la forma en que se presenta el modelo (por ejemplo, desplazamiento por un documento o por los diferentes registros de una base de datos).

  En definitiva, se podría decir que, el controlador hace de intermediario entre la vista y el modelo.

El sistema desarrollado sigue el siguiente diagrama que muestra una representación gráfica del patrón MVC anteriormente descrito.

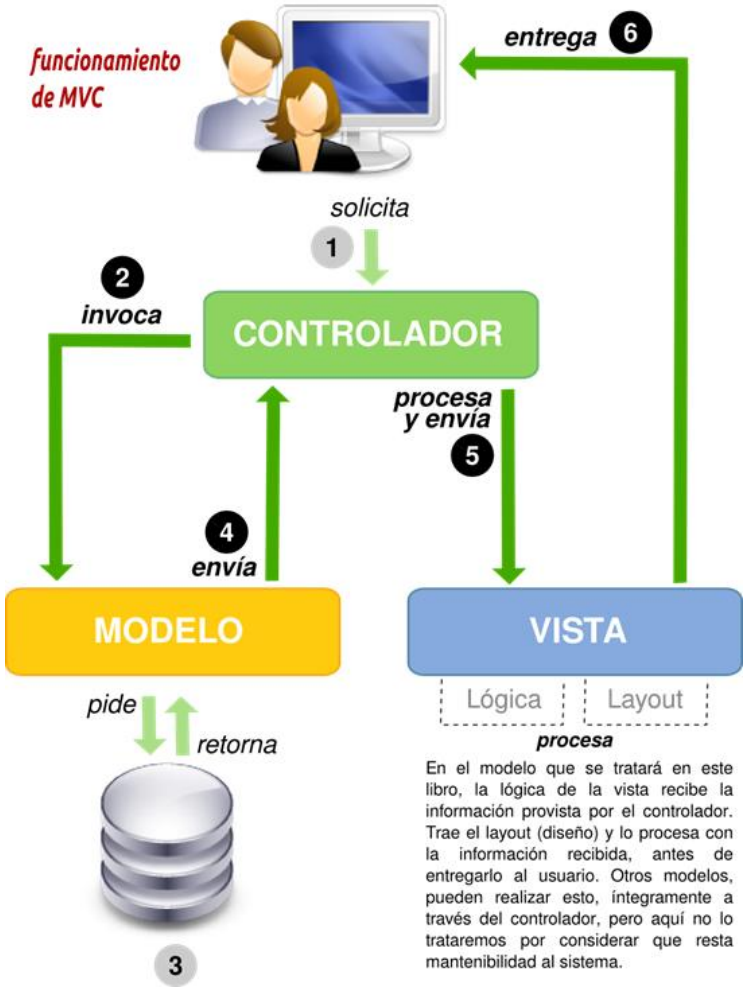


FIGURA 24 - DIAGRAMA PATRÓN MVC

El comportamiento básico del diagrama mostrado en la **Figura 24** es el siguiente:

1. El usuario interactúa con la interfaz de usuario de alguna forma (pulsa un botón, un enlace…). De esta manera el controlador recibe la notificación de la acción solicitada por el usuario y gestiona el evento que llega.
2. El controlador accede al modelo, actualizándolo (por ejemplo, mediante una llamada de retorno – callback).
3. El modelo será el encargado de interactuar con la base de datos. Puede ser de forma directa, con una capa de abstracción, un Web Service… y ésta retornará la información al controlador
4. El controlador recibe la información y la envía a la vista.
5. La vista procesa esta información, creando una capa de abstracción para la lógica (quien se encargará de procesar los datos) y otra para el diseño de la interfaz gráfica o GUI.
6. La lógica de la vista, una vez procesa los datos, los acomodará en base al diseño de la GUI – layout – y los entregará al usuario de forma legible para el usuario.

Este es el esquema general para un patrón Modelo Vista Controlador, y Rails, obedece a este esquema.

7.1.2. ACTIVE RECORD

Nuestro sistema, gobernado por el framework Ruby on Rails, se utiliza para operar con las bases de datos un Mapeo Objeto Relacional (ORM – Object-Relational Mapping).

Este mapeo es una técnica de programación utilizada para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación a objetos y, la utilización de una base de datos relacional como motor de persistencia.

En la práctica, esto crea una base de datos orientada a objetos virtual, sobre la base de datos relacional. Esto posibilita el uso de las características propias de la orientación a objetos (básicamente herencia y polimorfismo).

Resumiendo, un ORM nos permite hacer consultas a las bases de datos a través de objetos.

En el mercado existen paquetes comerciales y de uso libre que desarrollan este mapeo relacional, aunque hay programadores que prefieren crear sus propias herramientas ORM.

En nuestro caso en particular, vamos a hacer uso de la herramienta ActiveRecord de Rails.

ActiveRecord es la capa de Rails de mapeo relacional de objetos. Provee una interfaz orientada a objetos que se relacionan con una base de datos, para hacer el desarrollo más simple y amigable para el desarrollador.

El ActiveRecord de Ruby, como ya hemos mencionado, es un ORM: hace las traducciones entre objetos Ruby y la base de datos, manejando registros y relaciones.

La librería ActiveRecord de Ruby crea un modelo de negocio persistente desde los objetos de negocio y las tablas de datos, unificándose la lógica y los datos en un solo paquete.

ActiveRecord nos agregará también herencia y asociaciones al patrón ActiveRecord común, dándole más potencia.

Ni que mencionar tiene que es una característica muy útil para nosotros, ya que nos permitirá abstraernos de utilizar lenguajes específicos para acceder a las bases de datos, y centrarnos sólo en qué es lo que queremos conseguir, sin necesidad de tener que estudiar otro lenguaje específico.

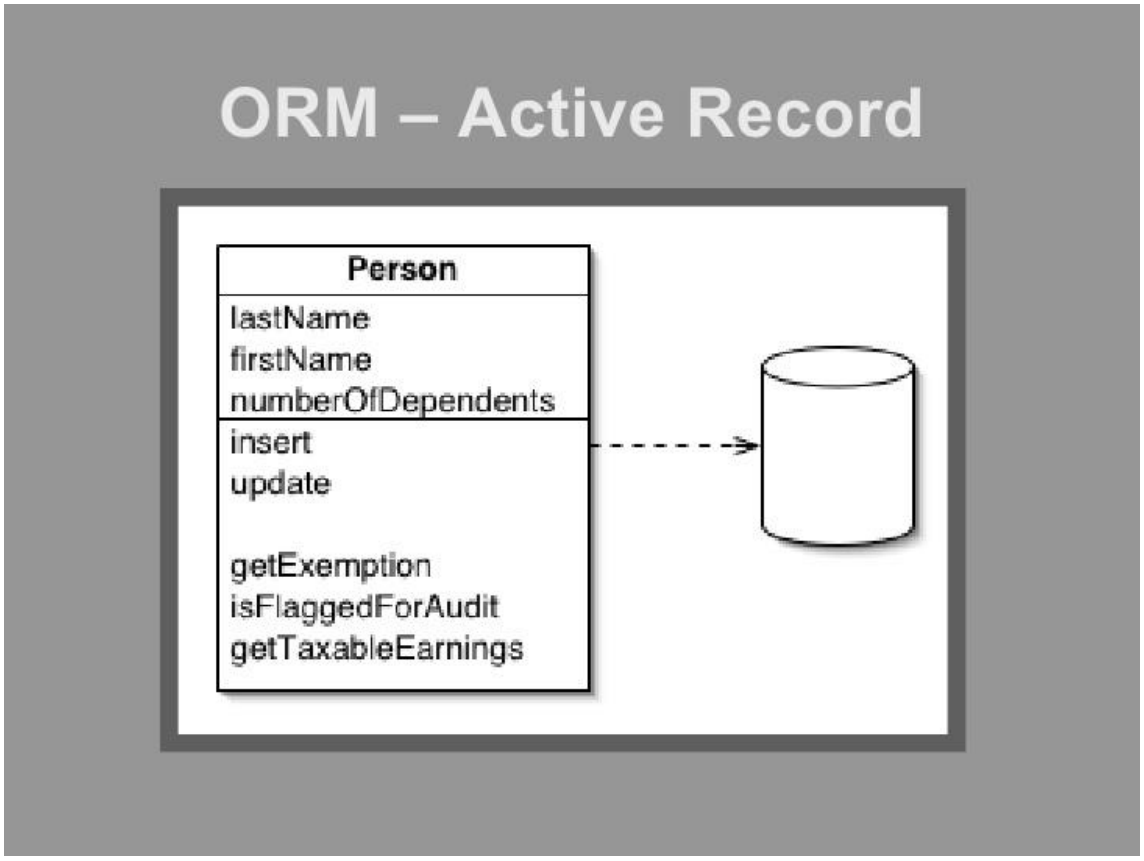


FIGURA 25 - MAPEO OBJETO RELACIONAL ORM

7.1.3. TRANSFERENCIA DE ESTADO REPRESENTACIONAL (REST)

REST (**RE**presentational **S**tate **T**ransfer) describe un paradigma de arquitectura para aplicaciones que solicitan y manipulan recursos en la web utilizando los métodos estándar de HTTP: POST, GET, PUT, PATCH y DELETE.

En REST, todo recurso es una entidad direccionable mediante una URL única con la que se puede interactuar a través del **protocolo HTTP**.

Dichos recursos pueden representarse en diversos formatos como HTML, XML o RSS, según solicite el cliente. Al contrario que en las aplicaciones Rails tradicionales, la URL de un recurso no hace referencia a un modelo y su acción correspondiente, tan sólo hace referencia al propio recurso.

Entre otras características, utilizar un modelo de aplicación **REST**, aporta los siguientes **beneficios**:

- **URLs limpias**: En REST, las URLs representan recursos y no acciones, por lo tanto siempre tienen el mismo formato, primero aparece el controlador y luego el identificador del recurso.
- **Formatos de respuesta variados**: Los controladores REST están escritos de manera que las acciones pueden devolver sus resultados fácilmente en diferentes formatos de respuesta. Una misma acción puede entregar HTML, XML, RSS…
- **Menos código**: El desarrollar acciones únicas capaces de soportar múltiples clientes distintos evita repeticiones en el sentido DRY.
- **Controladores orientados a CRUD**: Los controladores y los recursos se funden en una única cosa, cada controlador tiene como responsabilidad manipular un único tipo de recurso.
- **Diseño limpio**: El desarrollo REST produce un diseño de aplicación conceptualmente claro y más fácil de mantener.

En definitiva, el paradigma de arquitectura REST es un esquema de URLs y métodos HTTP que representan operaciones sobre un recurso (leer, crear, actualizar…).

Veamos un **ejemplo** de rutas para el **recurso Perro**:

Operación	Ruta	Método HTTP
Crear perro	/pets/new	GET
Editar perro	/pets/1/edit	GET
Listar perro	/pets/index	GET
Borrar perro	/pets/1	DELETE
Actualizar perro	/pets/1	PUT
Mostrar perro	/pets/1/show	GET

Listar rutas perro	/pets/1/routes	GET
Crear ruta perro	/pets/1/new_route	GET

7.2. APLICACIÓN WEB

7.2.1. ESTRUCTURA

En esta sección representaremos el diseño de la aplicación web desarrollada en este proyecto para la gestión de una adecuada geolocalización de mascotas. Para su desarrollo, se han tenido en cuenta los aspectos fundamentales de accesibilidad web, con especial atención a la visualización del contenido que se ha explicado a lo largo de presente documento.

Dependiendo del tipo de usuario registrado, tendrá unos contenidos u otros.

PROPIETARIO

- **Inicio**
  - Logotipo de la aplicación.
- **Perros**
  - Listado de perros registrados.
  - Listado de perros a pasear.
  - Las funcionalidades: crear, editar y borrar perros.
  - Acceso a la visualización de las rutas de cada perro.
- **Paseadores**
  - Listado de paseadores a contratar.
- **Perfil**
  - Información del usuario registrado, permitiendo modificarla si se desea.

PASEADOR

El paseador tendrá los mismos contenidos que el propietario, añadiendo además “Mis paseos”.

- **Mis paseos**
  - Listado de paseos que ha realizado el paseador con los perros que ha paseado. En él podrá ver los paseos que está haciendo actualmente como los que ha hecho con anterioridad.

ADMINISTRADOR

El administrador no contiene los mismos contenidos que los anteriores tipos de usuarios. Éste contendrá lo siguiente:

- **Usuarios**
  - Listado de todos los usuarios registrados en el sistema.
  - Control de usuarios: modificación del rol de cada usuario y el borrado del mismo.

Estas secciones son las que deberá contener y estar presentadas en el sitio web. A continuación presentamos una serie de mockups que se han realizado para reflejar cómo debe estar representado dicho portal de acuerdo al contenido que debe tener.

Los mockups de las secciones más importantes de la web son: la página principal, el registro y autenticado de un usuario, el perfil de un usuario, el registro de un nuevo perro, el listado de perros de un propietario, el listado de perros propios y que pasea un paseador, el listado de paseadores que están registrados en el sistema y el listado de rutas de un perro.

Comenzaremos mostrando el mockup de la página principal de la aplicación:

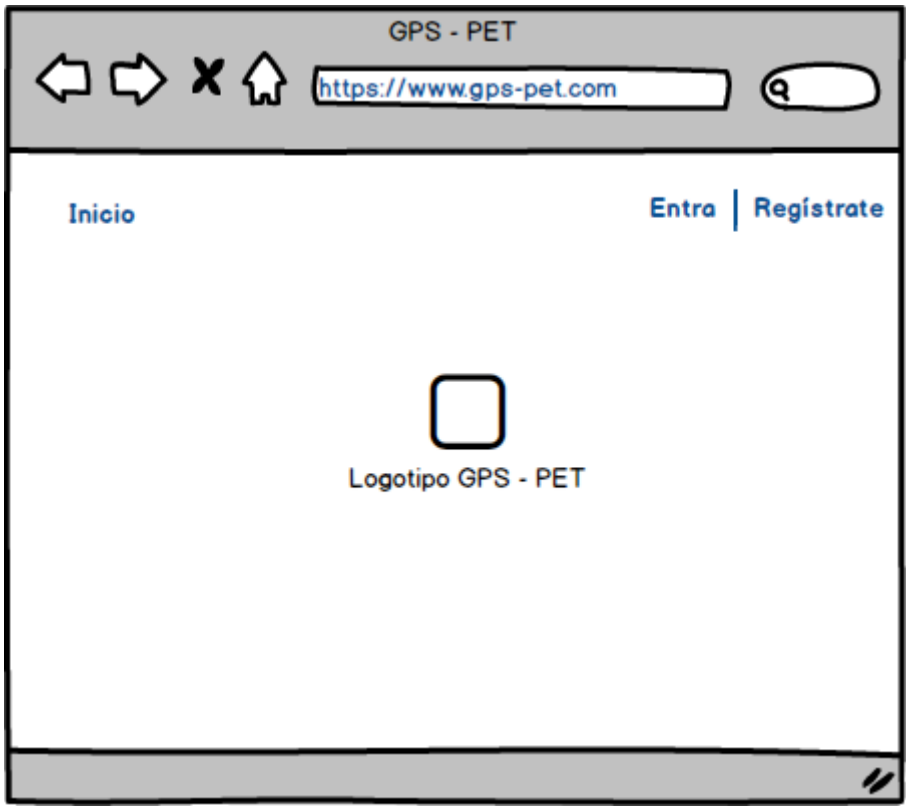


FIGURA 26 - MOCKUP INICIO APLICACIÓN GPS-PET

La página de inicio de la aplicación debería ser algo parecido a lo que muestra la **Figura 26**. En esta vista, se muestra el logotipo de la aplicación web y la posibilidad de autenticarse o registrarse en la aplicación.

Cuando el usuario esté autenticado, el menú de navegación cambiará según el rol que tenga el usuario, administrador, propietario o paseador. Estos dos últimos, tendrán el mismo menú de navegación, con la excepción de que el paseador tendrá un enlace más llamado **Mis paseos**, que permitirá a este tipo de usuario visualizar los paseos que ha hecho y que está haciendo actualmente con los perros que está paseando. Esto se podrá apreciar en el mockup de la **Figura 31**.

Los mockups que se muestran a continuación, corresponde al formulario de autenticación y el de registro de usuario respectivamente.

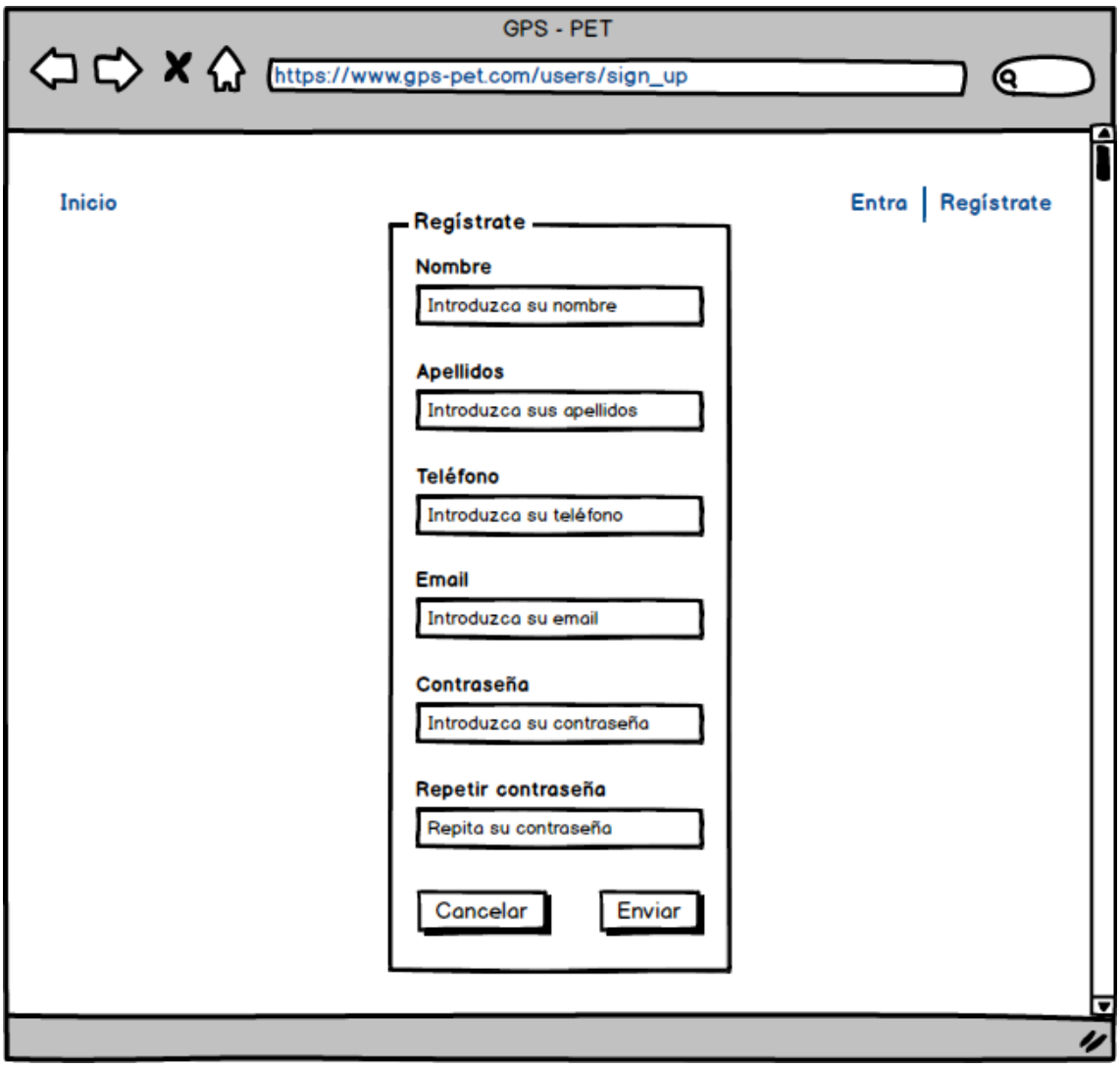


FIGURA 27 - REGISTRO DEL USUARIO

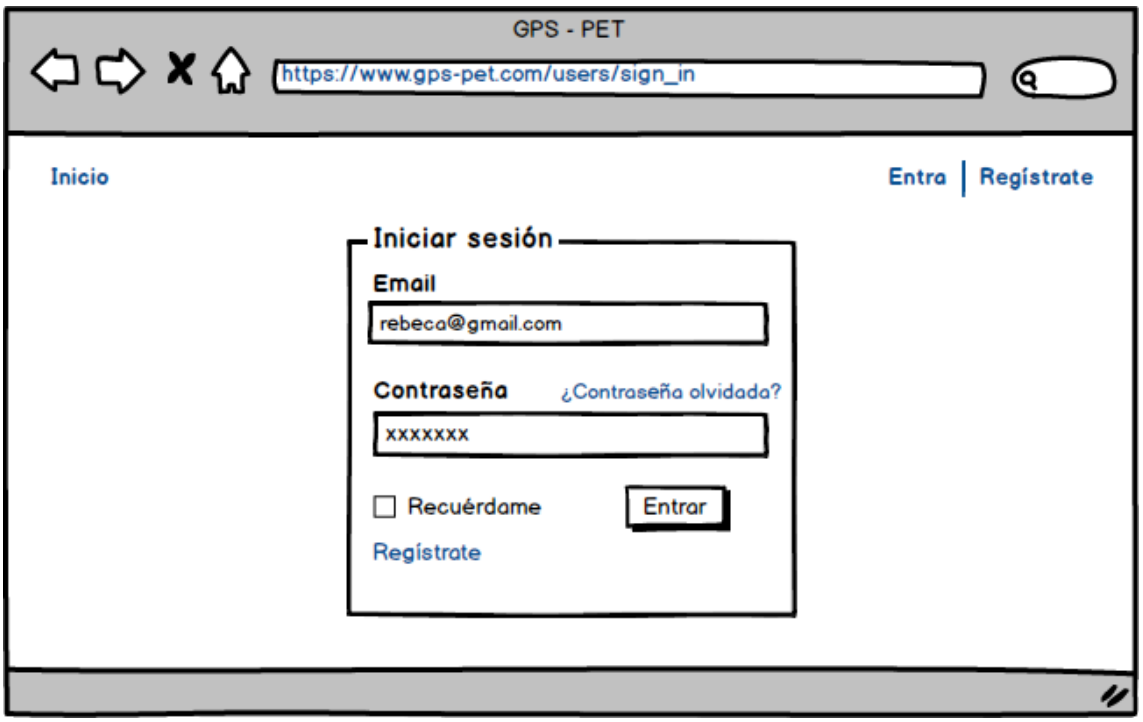


FIGURA 28 - AUTENTICACIÓN DEL USUARIO

El usuario, una vez que se ha registrado y autenticado, tendrá la posibilidad de acceder a las distintas secciones que conforman la aplicación, siendo el siguiente a mostrar el de los **Perros**.

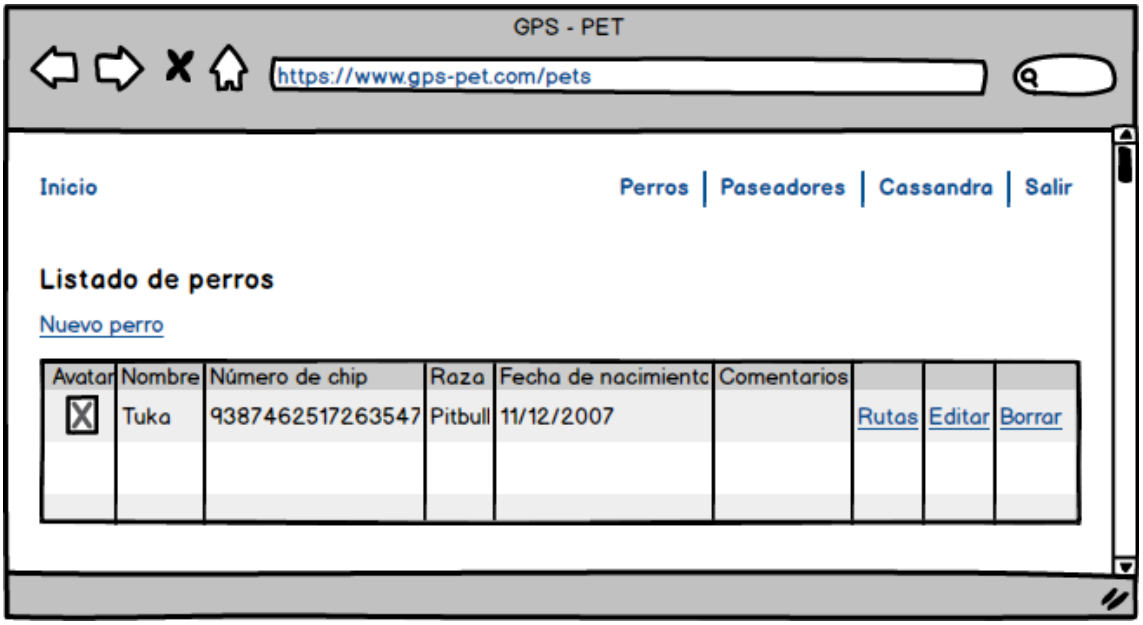


FIGURA 29 - LISTADO DE PERROS DE UN USUARIO

En este mockup, se puede ver el listado de perros que tiene la propietaria Cassandra, cuyo perro registrado se llama Tuka.

Si desea **añadir un nuevo perro** en el sistema para poder realizar su seguimiento, tendrá que rellenar un formulario como el que se muestra en la **Figura 30**.

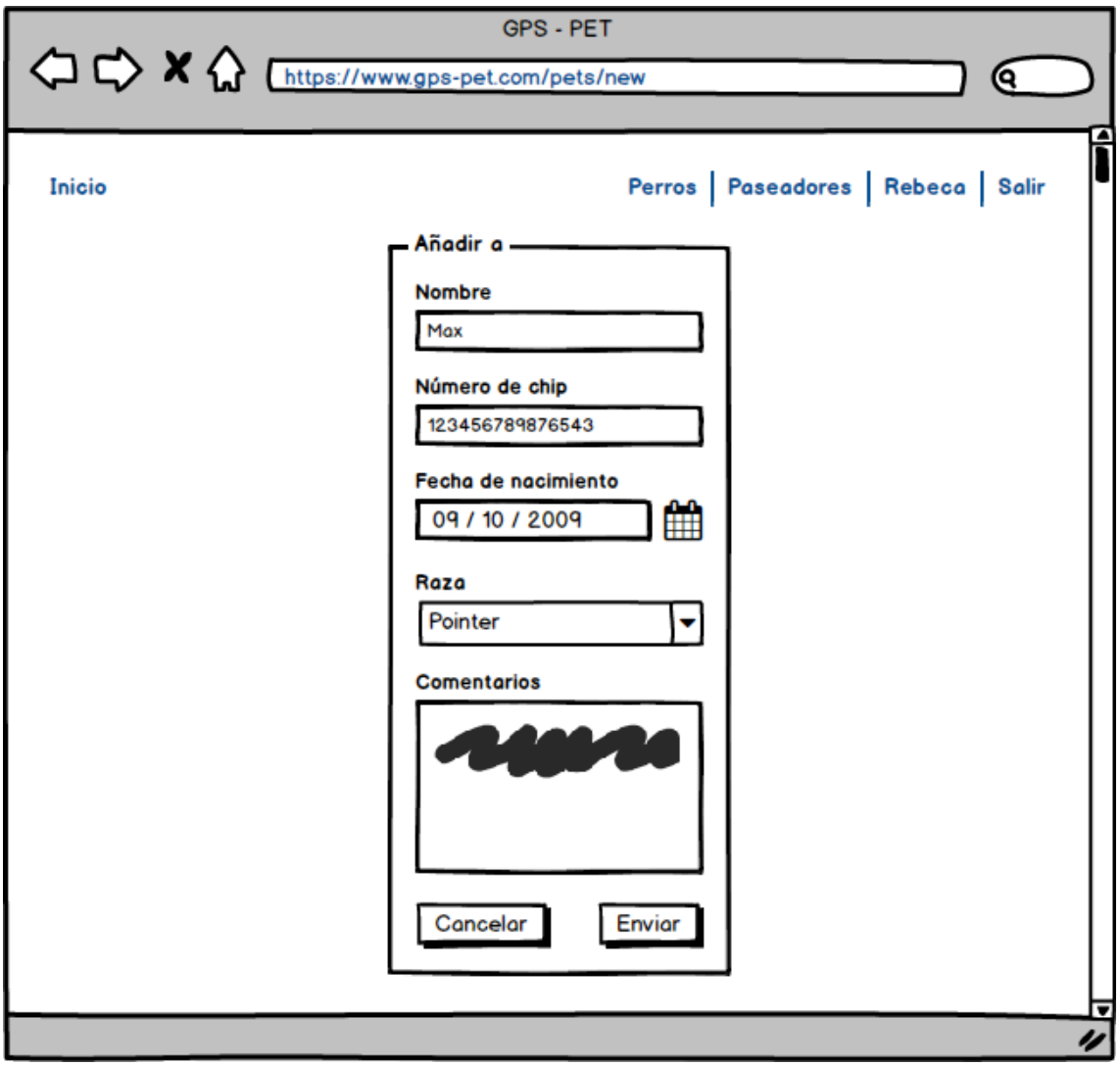


FIGURA 30 - REGISTRO DE UN PERRO

Se rellenará el formulario introduciendo los datos principales para su registro, tal como su nombre, número de chip, fecha de nacimiento, imagen y un comentario si se desea.

Registrado el perro, para poder hacer el seguimiento del mismo, es requisito contratar a un paseador. Para ello, tendríamos que ir a la sección de **Paseadores** del menú de navegación de la aplicación web.

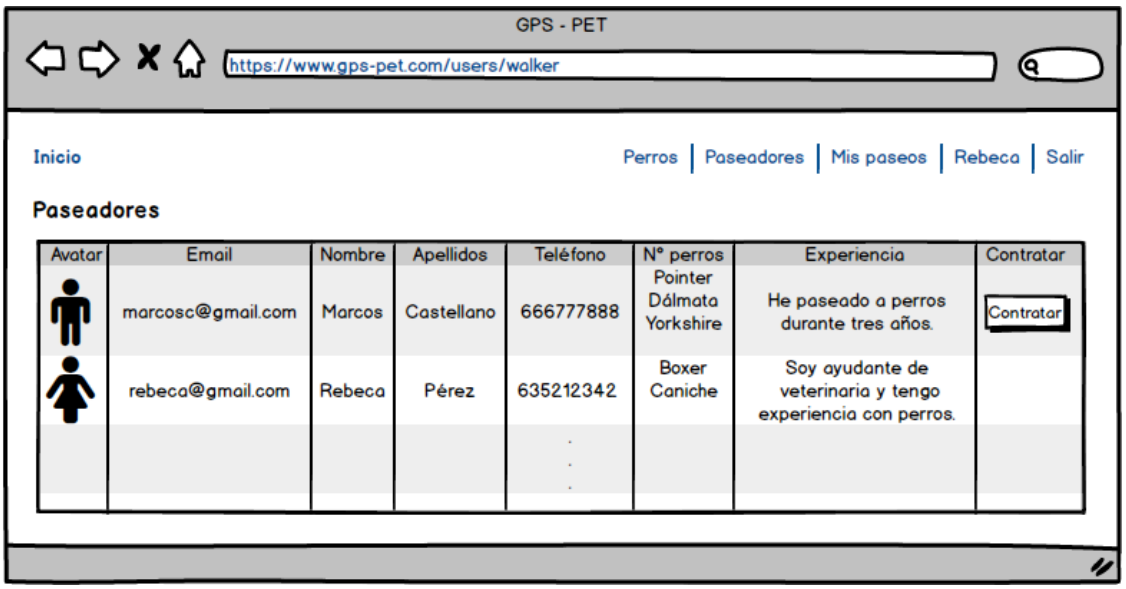


FIGURA 31 - LISTADO DE PASEADORES

Como se ha indicado con anterioridad, un propietario solo puede contratar a un paseador, siendo éste el que pasee a todos los perros del que lo ha contratado.

El paseador una vez que ha sido contratado por un propietario, automáticamente podrá apreciar en su listado de perros que pasea, a los perros del propietario propiamente dicho. De este modo, podrá iniciar el paseo y, por tanto, una nueva ruta con los perros en cuestión.

Para que un propietario pueda convertirse en paseador, tiene que ir a su **Perfil** y modificar sus datos, indicando que quiere cambiar su rol en el sistema y añadir información que le ayude a su posterior contratación.

En el siguiente mockup se podrá apreciar cómo sería el formulario del **Perfil** de un **usuario registrado**.

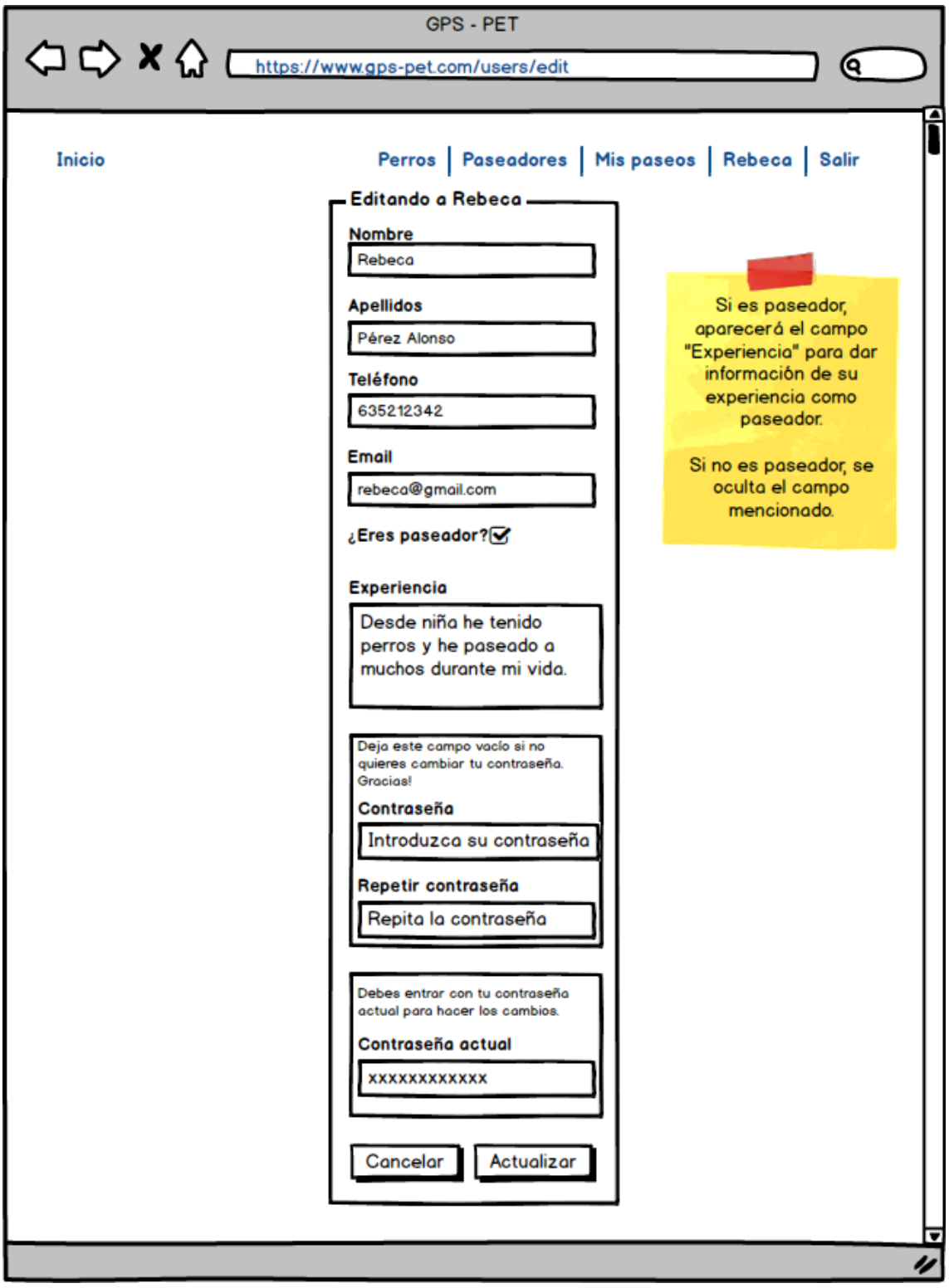


FIGURA 32 - PERFIL USUARIO REGISTRADO

Como se puede apreciar, hay un checkbox que permite al usuario modificar su rol para ser paseador, añadiendo además, otro campo de texto para que éste pueda indicar y exponer ante todos los usuarios registrados, la experiencia que tiene como paseador. Información que le permitirá a un propietario decidirse por un paseador u otro.

Siendo paseador ahora, el usuario verá que el contenido de la sección de Perros ha cambiado, pues no solo visualizará su listado de perros propios, sino también a los que va a pasear.

FIGURA 33 - LISTADO DE PERROS

El paseador en esta vista solo podrá editar y borrar sus perros; ver la ruta que está realizando su perro cuando está siendo paseado e iniciar una nueva ruta cuando está paseando al perro de un propietario.

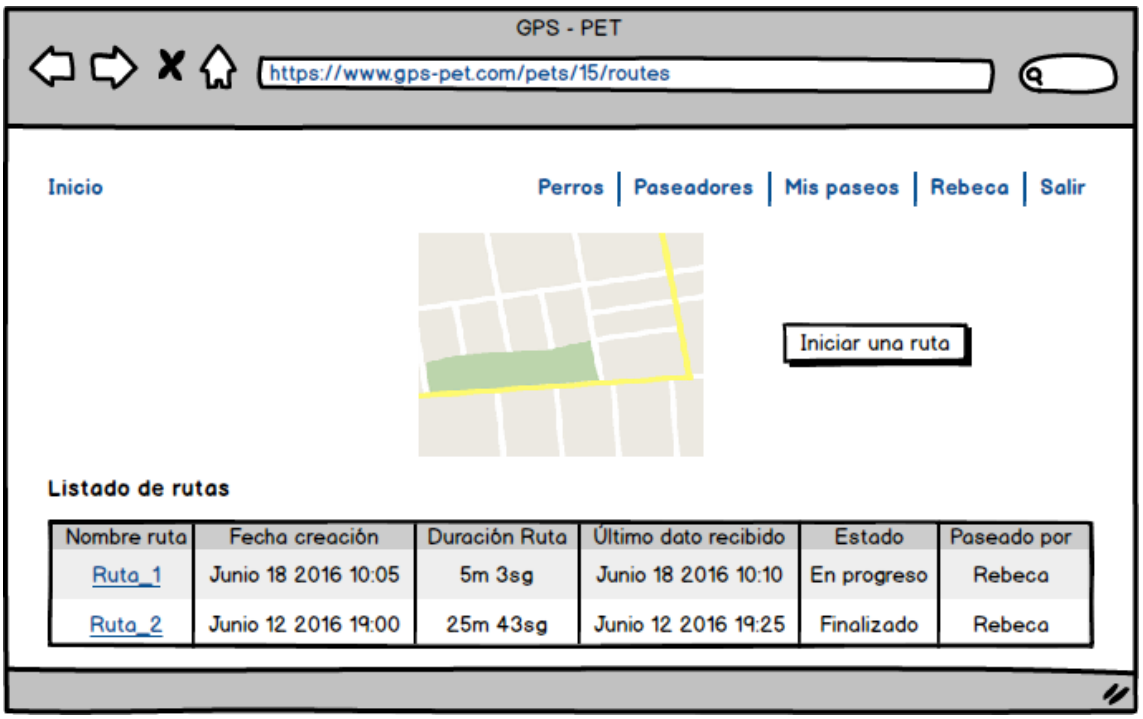


FIGURA 34 - RUTAS DE UN PERRO PASEADO POR REBECA

La paseadora Rebeca puede observar el listado de rutas del perro que pasea, pudiendo visualizar nuevamente la Ruta_1 realizada con anterioridad, así como la duración de la ruta, el último dato recibido por el simulador GPS, el estado en el que se encuentra la ruta (*Finalizado* si ya ha terminado de realizar el paseo, o e*n progreso* si el perro aún está siendo paseado), ver quién lo paseó por última vez y su fecha de creación.

Cada vez que vaya a pasear a un perro, ésta tendrá que accionar el botón de ***Iniciar nueva ruta*** para que active la simulación del GPS, enviando de este modo al mapa, las coordenadas de la posición actual del perro.

En este caso, la propietaria Cassandra una vez autenticada, podrá seguir en el mapa, desde la vista de Rutas de su perro, el paseo que le está dando la paseadora Rebeca. Observará los mismos datos que la paseadora en cualquier momento.

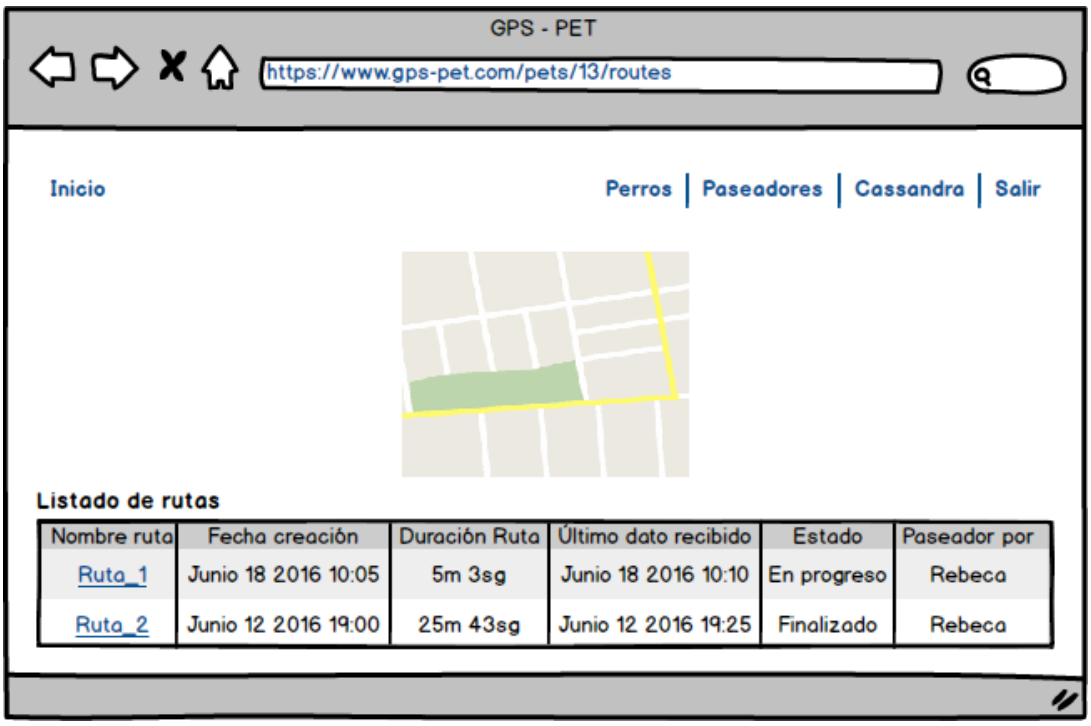


FIGURA 35 - LISTADO DE RUTAS DE UN PERRO DE LA PROPIETARIA CASSANDRA

La paseadora Rebeca podrá visualizar además, un listado de paseos que ha realizado durante su trayectoria en el registro de la aplicación.

Este listado se ha creado para que el paseador pueda tener un histórico de paseos que ha realizado con los distintos perros que ha paseado y que está paseando, pues una vez que un propietario deja los servicios del paseador, éste no podrá acceder al perro que estaba paseando para visualizar el listado de rutas que ha realizado con dicho animal.

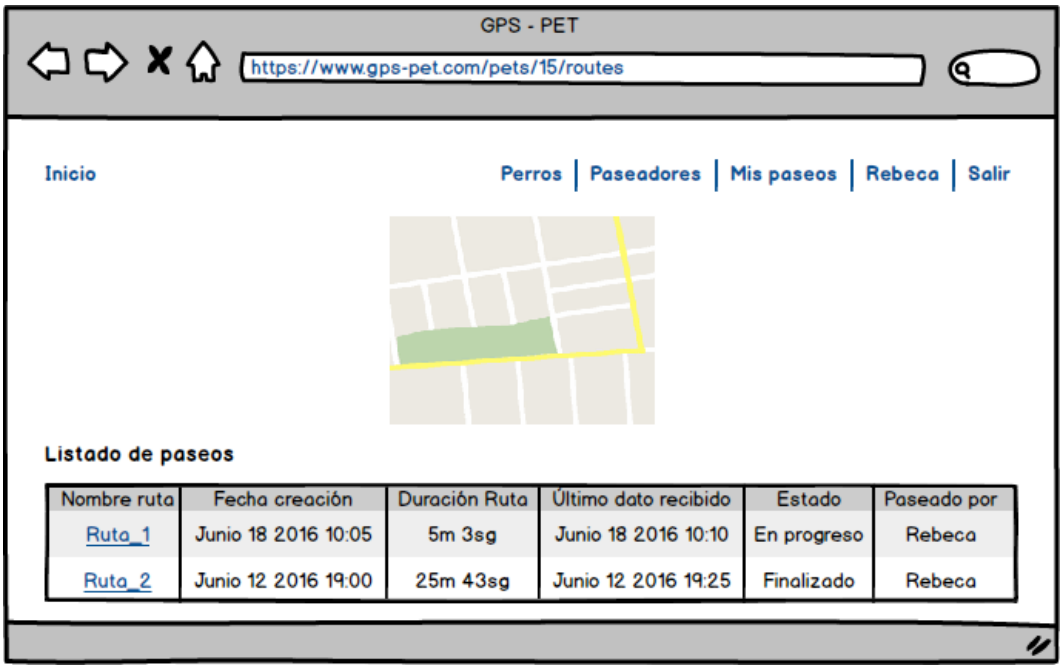


FIGURA 36 - LISTADO DE PASEOS

7.2.2. MODELO DE DESPLIEGUE

Una vez el sistema esté implementado, ha de ser desplegado para que pueda ser utilizado por los usuarios finales.

En nuestro caso en particular, se ha utilizado el Cloud Computing, técnica que permite ofrecer servicios de computación en la nube a través de internet. Pasemos a estudiar esto en detalle.

En el modelo tradicional de implementación de Tecnologías de Información, las organizaciones destinan recursos materiales, humanos y tecnológicos, los cuales agrupan en un área encargada de solucionar los problemas relacionados con la infraestructura informática y el desarrollo de aplicaciones para la organización.

La mayoría de estas áreas se ven obligadas a dedicar gran parte de su tiempo en las tareas de implementar, configurar, mantener y actualizar proyectos relacionados con la infraestructura de su organización, lo que normalmente no supone valor añadido en el balance final de la producción de la misma.

Por otra parte, se puede observar que la distribución de servicios, tales como energía eléctrica, agua potable o telefonía, dejan al proveedor la total responsabilidad de generar, organizar y administrar lo necesario para que el usuario final reciba lo acordado, pagando éste únicamente por el uso que hace de los mismo, mientras que realmente el proveedor se encarga de apreciar los mecanismos por medio de los cuales determina el consumo por el que se genera el cobro.

De esta manera, nos surge una pregunta: ¿por qué no implementar servicios o recursos de Internet bajo un esquema similar al descrito, donde el proveedor solamente proporcione lo requerido y el usuario pague únicamente por el uso que hace?

Es por esto que dirigimos nuestra mirada hacia la tecnología mencionada anteriormente, el Cloud Computing (cómputo en la nube), que es capaz de minimizar el tiempo empleado en actividades de menor valor y permitirnos centrarnos realmente en lo importante en nuestro desarrollo.

En la **Figura 37** podemos observar un diseño arquitectónico general de cómo se comporta la nube, en la que los diferentes clientes interactúan con las distintas capas en las que puede dividirse la nube.

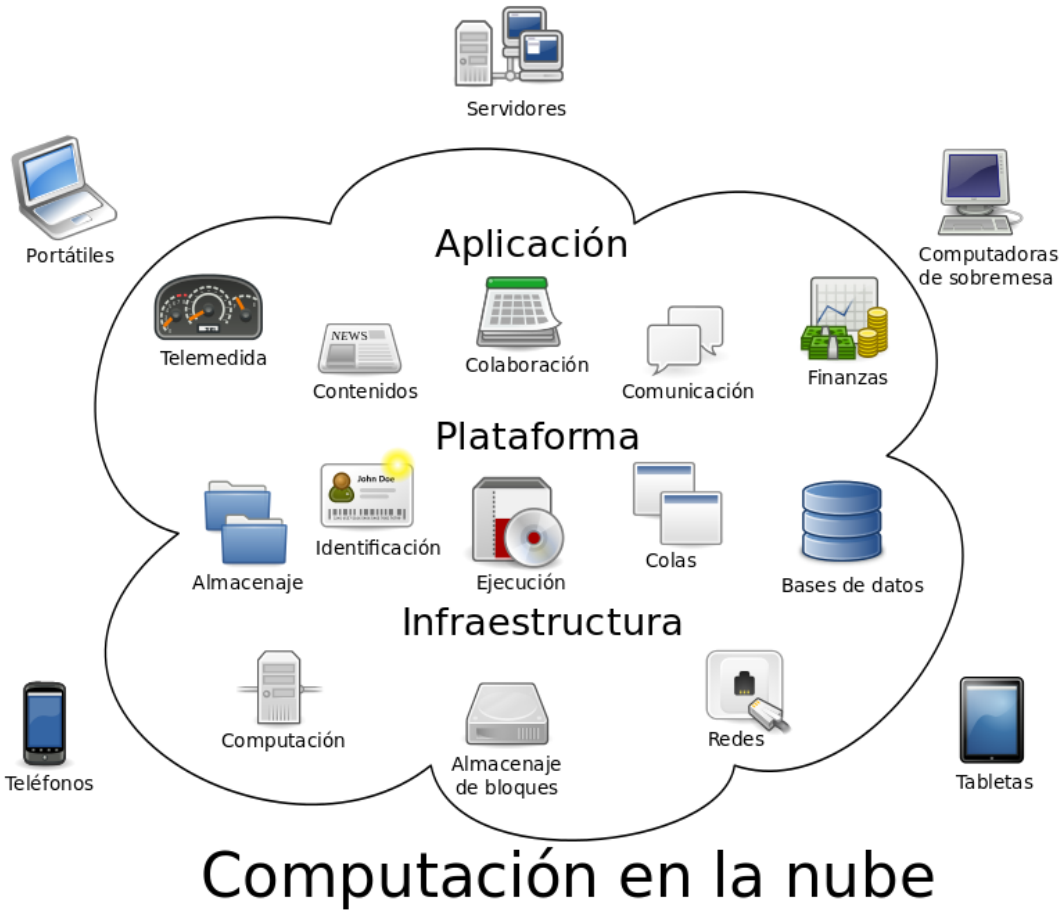


FIGURA 37 - ESQUEMA DEL COMPORTAMIENTO DE LA NUBE

Por lo general, el uso de este concepto se está extendiendo con una considerable velocidad, dando como resultados un incremento en el número de empresas que proporcionan servicios a través de esta tecnología.

Los tipos de servicios que ésta nos puede proporcionar son extensos. El cliente paga a un proveedor por un recurso determinado (memoria, almacenamiento, procesamiento, software, bases de datos…) y éste le proporciona dicho servicio a través de internet.

Las **ventajas** que nos ofrece dicha tecnología se pueden simplificar en lo siguiente:

- **Costos**: Podría ser la ventaja más atractiva que presenta el cómputo en la nube, y si no lo es, al menos es la más evidente de todas las que ofrece esta tecnología.

  Al dejar la responsabilidad de implementación de la infraestructura al proveedor, el cliente no tiene por qué preocuparse por comprar equipos de cómputo, capacitar personal, mantenimiento…

- **Competitividad**: Al no tener que adquirir costos de equipos, las pequeñas empresas pueden tener acceso a las más nuevas tecnologías a precios a su alcance, pagando únicamente por consumo.
- **Disponibilidad**: El proveedor está obligado a garantizar que el servicio esté disponible para el cliente. En este sentido, la virtualización juega un papel fundamental.
- **Abstracción de la parte técnica**: El cómputo en la nube permite al cliente la posibilidad de olvidarse de la implementación, configuración y mantenimiento de equipos.
- **Acceso desde cualquier punto geográfico**: El uso de las aplicaciones diseñadas sobre el paradigma del cómputo en la nube puede ser accesible desde cualquier equipo de cómputo en el mundo que esté conectado a internet.
- **Escalabilidad**: El cliente no tiene por qué preocuparse por actualizar el equipo de cómputo sobre el que se está corriendo la aplicación que utiliza, ni tampoco por la actualización de sistemas operativos o instalaciones de parches de seguridad, ya que es obligación del proveedor del servicio realizar este tipo de actualizaciones.
- **Concentración de esfuerzos en el negocio**: Como resultado de las ventajas antes mencionadas, el cliente puede concentrar más recursos y esfuerzos hacia un aspecto más estratégico y transcendente, que tenga un impacto directo sobre os procesos de negocio de la organización.

Sin embargo, también hay algunos puntos de **desventaja** en la utilización de esta tecnología. Detallemos algunos de ellos:

- **Privacidad**: Es comprensible la percepción de inseguridad que genera una tecnología que pone la información (sensible en muchos casos), en servidores fuera de la organización, dejando como responsable de los datos al proveedor de servicio.

  El tema a tratar aquí, es el de la privacidad, ya que para muchos es extremadamente difícil el confiar su información sensible a terceros, y consideran que lo que propone el cómputo en la nube pone en riesgo la información vital para los procesos del negocio.
- **Disponibilidad**: Si bien es cierto que en las ventajas se incluyó el concepto de disponibilidad, ésta queda como una responsabilidad que compete únicamente al proveedor del servicio, por lo que si su sistema de redundancia falla, y no logra mantener el servicio para el usuario, éste no puede realizar ninguna acción correctiva para restablecer el servicio.
- **Falta de control sobre recursos**: Al tener toda la infraestructura e incluso la aplicación corriendo sobre servidores que se encuentran en la nube, es decir, del lado del proveedor, el cliente carece por completo de control

sobre los recursos e incluso sobre su información una vez ésta es subida a la nube.

- **Dependencia**: En una solución basada en cómputo en la nube, el cliente se vuelve dependiente no sólo del proveedor del servicio, sino también de su conexión a internet.
- **Integración**: No siempre es fácil o práctica la integración de recursos disponibles a través de infraestructuras de cómputo en la nube con sistemas desarrollados de una manera tradicional, por lo que este aspecto debe ser tomado en cuenta por el cliente para ver qué tan viable resulta implementar una solución basada en la nube dentro de su organización.

En nuestro caso en particular, como nombramos anteriormente en el documento, vamos a utilizar Heroku como plataforma, y el dominio final del producto será: **http://gps-pet.herokuapp.com/**

7.3. DIAGRAMA DE SECUENCIA GENERAL

Esta sección va a estudiar el diagrama general que debe tener un sistema basado en el paradigma Modelo-Vista-Controlador, el cual mostraremos a continuación.

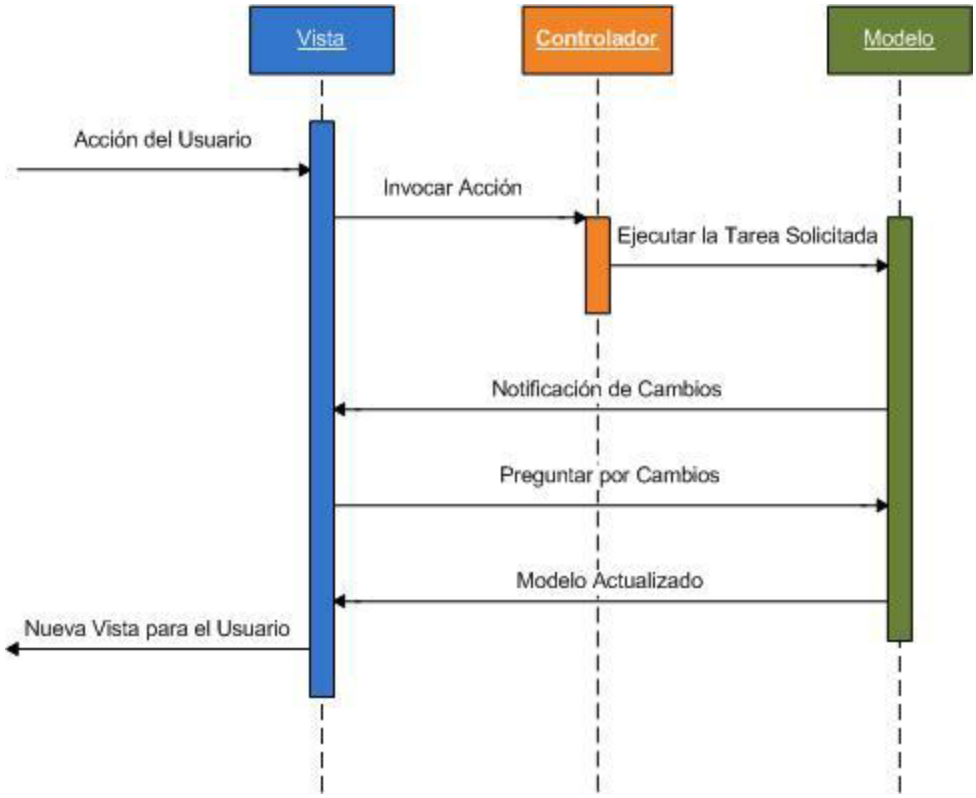


FIGURA 38 - DIAGRAMA SECUENCIAL GENERAL

Como se puede observar en la **Figura 38**, el usuario interactúa con la vista, y las acciones (como por ejemplo clicar un botón) se capturan en la vista y se reenvían al controlador.

El controlador decide qué hacer y lo hace a través de una iteración con el modelo.

El modelo es esencialmente la capa empresarial de algunas capacidades adicionales. En concreto, el modelo notifica a la vista sobre los cambios que pueden requerir una actualización de la interfaz de usuario.

El modelo no conoce los detalles de la vista, pero entre el modelo y la vista hay una relación de "observador". En otras palabras, la vista contiene una referencia al modelo y se registra con el modelo para recibir notificaciones de cambios.

Cuando se recibe la notificación, la vista obtiene datos actualizados desde el modelo y actualiza la interfaz de usuario. La figura anterior muestra el diagrama de secuencia. Hay casos de implementación de MVC en el que es el controlador el que notifica a la vista los cambios.

7.4. BASE DE DATOS

La última sección que veremos en cuanto al diseño, va a estar relacionada con la base de datos de nuestra aplicación.

El esquema de una base de datos nos va a permitir describir la estructura de la base de datos.

El lenguaje utilizado es un lenguaje relacional, y nos da la posibilidad de definir cada tabla con sus campos y las relaciones entre ellas.

Las bases de datos permiten que los sistemas sean dinámicos, y son herramientas potentes, ampliamente utilizadas en todas las aplicaciones.

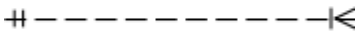
Existen varias formas de representar el modelado de una base de datos y en nuestro caso en particular, vamos a utilizar el modelo entidad-relación.

Un **modelo entidad-relación** es una herramienta muy utilizada para el modelado de datos que permite representar las entidades relevantes de un sistema de información, así como sus interrelaciones y propiedades.

El procedimiento que se ha utilizado para el modelo ha sido, primeramente, elaborar el diagrama entidad-relación que mostramos en la figura 39, y a continuación complementar el modelo con los atributos propios de cada una de las entidades, así como las restricciones que no se han podido reflejar en el diagrama.

En el diagrama de nuestro sistema, va a tener un solo tipo de relación, que es la de uno a muchos (**1:m**).

Esta relación es la más común. Cada registro de una tabla puede estar enlazado con varios registros de una segunda tabla, pero cada registro de la segunda sólo puede estar enlazado con un único registro de la primera. Se representa mediante este dibujo:



A continuación vamos a mostrar dicho diagrama y a estudiarlo, parándonos a detallar los puntos más relevantes de este diagrama.

Veremos que tenemos un total de 5 tablas donde guardarán relación directa entre ellas.

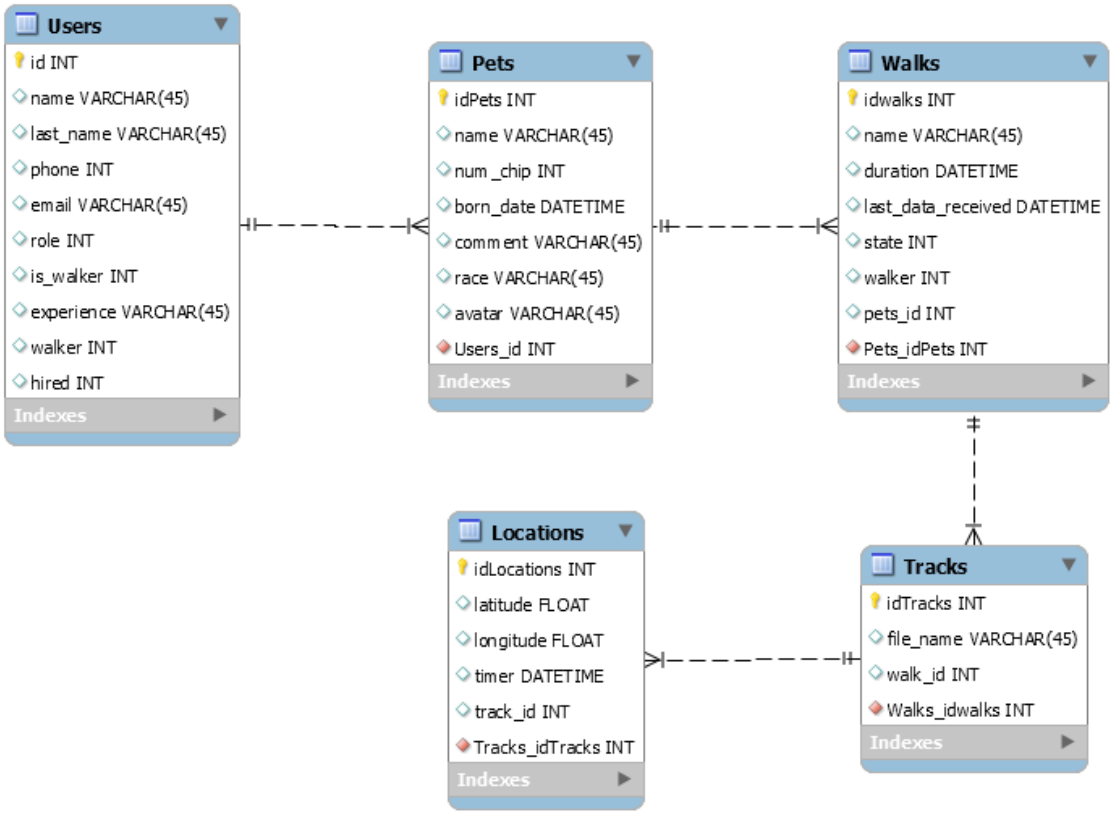


FIGURA 39 - DIAGRAMA ENTIDAD-RELACIÓN DE LA BASE DE DATOS

- **Users**: Esta tabla es la creada por la gema Devise, y lo que guarda son los datos personales de cada usuario.
- **Pets**: Se utiliza para guardar los datos referentes a los perros de los usuarios. Está relacionado 1:m con la tabla Users.
- **Walks**: Guarda la información referente al paseo de una mascota. Establece relación 1:m con la tabla Pets.
- **Tracks**: Almacena las rutas realizadas en un paseo, y al igual que las anteriores, presenta una relación de 1:m con la tabla Walks.
- **Locations**: Esta tabla almacena cada localización, coordenada, leídas del fichero .kml. Está relacionanada con la tabla Tracks, siendo su relación de 1:m.

8. IMPLEMENTACIÓN

En este apartado del documento vamos a describir los aspectos de más interés en la fase de implementación de la aplicación. Concretamente vamos a hacer especial hincapié en la estructura de ficheros de la aplicación, de las librerías y gemas que se han usado en el Front-End y Back-End de la aplicación.

8.1. ESTRUCTURA DE FICHEROS DE LA APLICACIÓN

Esta sección está destinada al estudio de la estructura de directorios del proyecto. Vamos a ver la estructura general de directorios y detallar lo que sucede dentro de cada uno de los directorios relevantes en nuestra aplicación. En la **Figura 40** mostramos dicha estructura y a continuación detallaremos el contenido de estos directorios.

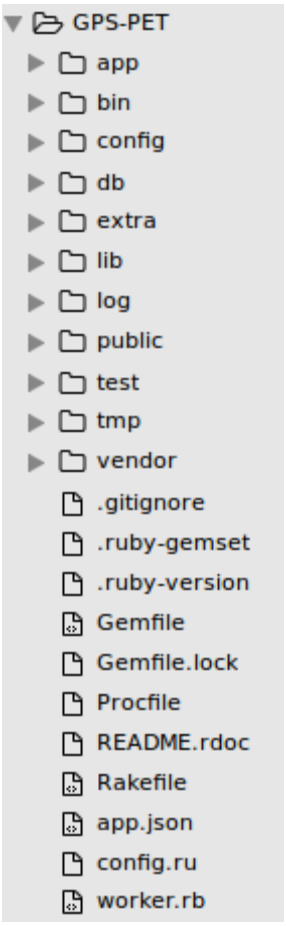
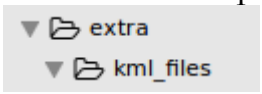


FIGURA 40 - ESTRUCTURA DE DIRECTORIOS

- **App**: En este directorio se organiza la estructura general de la aplicación. Los principales subdirectorios son: Las vistas de la aplicación (views y helpers), los controladores (controllers) y los modelos de nuestro sistema (models).

- **app/assets**: Contiene los ficheros de implementación de funciones JavaScript, CSS y las imágenes que el sistema va a mostrar.
- **app/controllers**: Es el directorio donde Rails busca los controladores de nuestro sistema.
- **app/helpers**: En este directorio nos encontramos con los helpers que apoyan a las clases de los modelos, vistas y controladores. El objetivo fundamental de los ficheros que componen este directorio es mantener un código limpio, claro y conciso.
- **app/models**: Se compone de clases de modelos.
- **app/uploaders**: Esta carpeta almacena los ficheros con información referente a las características de carga de ficheros.
- **app/views**: Directorio donde están los archivos que contienen la implementación de las vistas de la aplicación. Contiene las plantillas con formato .ERB para completar con los datos de la aplicación, convertir a HTML y devolver al navegador del usuario.
- **bin**: Contiene archivos binarios ejecutables.
- **config**: En esta carpeta se guardan archivos que contienen configuración que requiere la aplicación: configuración de la base de datos (database.yml), la estructura del enviroment de Rails (enviroment.rb), el fichero donde se encuentra todo el enrutado del sistema (routes.rb), ficheros multi-idioma (directorio locales), etc.
- **db**: En esta carpeta podremos ver el esquema de la base de datos y el fichero seed.rb, que se utiliza en la aplicación para la adición de valores predeterminados en la instalación de la aplicación. En este caso se ha hecho uso para importar las razas de los perros en la base de datos y crear el administrador del sistema. También posee una subcarpeta en la que se hayan todas las migraciones realizadas durante el desarrollo del proyecto.
- **extra**: Esta carpeta ha sido añadida para almacenar de los ficheros .kml para su posterior uso en el desarrollo de la aplicación.



- **lib**: Contiene módulos de bibliotecas específicas de la aplicación.

- **log**: Archivos de log de peticiones y errores de todos los entornos de Rails (development, production y test).
- **public**: En esta carpeta se almacenan los datos accesibles para los usuarios, por ejemplo, los accesibles vía navegador.
- **test**: Se utiliza para las pruebas realizadas con Minitest. En nuestro caso no hemos realizado pruebas con esta herramienta. Contiene subdirectorios para pruebas de controladores, helpers, mailers, etc.
- **vendor**: Códigos de terceros tales como plugins y gemas.

Además de estos directorios, hay varios ficheros muy importantes como:

- **.gitignore**: En él se especificará los archivos que no se desee que se monitorice.
- **README.rdoc**: Fichero que muestra una descripción de la aplicación, así como las indicaciones para una correcta instalación de la misma.
- **Gemfile**: Contiene todas las gemas que se va a hacer uso la aplicación.

8.2. RECURSOS EXTERNOS

Hemos utilizado una serie de recursos externos, como son librerías, gemas y servicios, para poder realizar la implementación de nuestro sistema. Los clasificaremos en dos grupos, el cliente (Front-End) y el servidor (Back-End).

8.2.1. CLIENTE / FRONT – END

En el Front – End o cliente, se ha hecho uso de la biblioteca jQuery Datepicker y del servicio de Gravatar.

JQUERY USER INTERFACE DATEPICKER

Dentro del jQuery User Interface encontramos toda una serie de elementos de programación de la interfaz de usuario preprogramados y listos para ser integrados en nuestros proyectos HTML.

Se trata de una amplia biblioteca JavaScript que abarca desde efectos dinámicos, hasta menús, calendarios, etc. En concreto, el componente Datepicker nos proporciona un calendario totalmente personalizable, en el que podremos realizar selecciones de fechas y asociarlo a elementos HTML, como entradas de formularios, como es en este caso cuando se va a registrar un nuevo perro o editar al mismo.

GRAVATAR (GLOBALLY RECOGNIZED AVATAR)

Un **avatar** es una imagen que representa online un pequeño cuadro que aparece junto a su nombre cuando interactúa con los sitios web.

Las imágenes de avatar juegan cada vez un papel más importante en la web, sobre todo en las aplicaciones sociales. De modo que como quisimos añadir imágenes de avatar en nuestra aplicación, hemos hecho uso de Gravatar, que proporciona una forma muy cómoda de hacerlo. Lo único que tenemos que gestionar es una dirección de correo para cada usuario, y no hace falta que nos preocupemos de la subida de archivos, el recorte de imágenes o que los usuarios suban imágenes inapropiadas, porque esto lo gestionará Gravatar. Tan sólo le tenemos que dar una dirección de correo y nos devolverá el avatar de dicho usuario.

Por tanto, **Gravatar** es un servicio creado por Tom Werner que ofrece un avatar único globalmente.

8.2.2. SERVIDOR / BACK - END

En esta parte del sistema, hemos hecho uso de varias gemas muy importantes para el desarrollo de la aplicación. Con la utilización de estas, se ha podido implementar de manera más fácil las distintas funcionalidades que compone la aplicación, como por ejemplo, la gema Devise, que nos permite gestionar los usuarios de una manera muy manejable y fácil.

PG

Esta gema nos permite tener acceso a la base de datos PostgreSQL haciendo de uso de su interfaz Ruby.

BOOTSTRAP

Es un framework desarrollado por Twitter que nos permite crear interfaces web con CSS y Javascript con todo tipo de diseños y siempre adaptadas al tipo de dispositivo que utilicemos para visualizar las páginas.

En Ruby On Rails, es posible además instalar este framework en forma de gema: gem 'bootstrap-sass' [x]. Las siglas Sass hacen referencia a un lenguaje de hojas de estilo creado específicamente para Ruby On Rails. Tiene un "rival" llamado LESS que podríamos integrar como alternativa.

DEVISE

La gema Devise es una solución de autenticación flexible para Rails basado en Warden:

- Está basado en Rack.
- Es una solución completa MVC basado en los Engines de Rails.
- Permite tener múltiples modelos accediendo al mismo tiempo.
- Se basa en un concepto de modularidad: Utiliza solo lo que realmente necesita.

Se compone de 10 módulos:

- **Base de datos autentificables**: encripta y almacena una contraseña en la base de datos para validar la autenticidad de un usuario mientras accede a la aplicación. La autenticación se puede realizar tanto a través de las peticiones POST o autenticación básica HTTP.
- **Omniauthable**: añade soporte OmniAuth (https://github.com/intridea/omniauth).
- **Confirmable**: envía mensajes de correo electrónico con instrucciones para confirmar y verificar si una cuenta ya se confirmó durante inicio de sesión.
- **Recuperable**: restablece la contraseña de usuario y envía instrucciones de restablecimiento.
- **Registrable**: se ocupa de suscribir a los usuarios a través de un proceso de registro, que también les permite editar y destruir su cuenta.
- **Rememberable**: gestiona la generación y la limpieza de un contador para recordar al usuario desde las cookies guardadas.
- **Rastreable**: contador de acceso a la aplicación, marcas de tiempo (timestamp) y dirección IP.
- **Timeoutable**: expira sesiones que no han estado activos en un período de tiempo determinado.
- **Validable**: proporciona validaciones de correo electrónico y contraseña. Es opcional y se puede personalizar, por lo que está en condiciones de definir sus propias validaciones.
- **Bloqueable**: bloquea una cuenta después de un número determinado de intentos fallidos de inicio de sesión. Puede desbloquear a través del correo electrónico o después de un período de tiempo específico.

GMAPS4RAILS

Gmaps4rails es una gema desarrollada para simplificar la creación de mapas de **Google Maps** con superposiciones (marcadores, ventanas de información, etc). Además, cuenta con el respaldo de una base de código muy flexible que podría ser propenso a aceptar otros proveedores de mapas.

Gracias a esta gema, hemos podido pintar en el mapa cada una de las coordenadas que hemos ido obteniendo a través de los ficheros .kml, permitiéndonos esto presentar la simulación del GPS.

KAMINARI

Con la gema de Akira Matsuda, la **paginación** de objetos se torna sencilla y elegante. Nuevamente, existen muchas otras opciones perfectamente válidas, siendo la más conocida will_paginate. Pero, esta gema aprovecha la posibilidad de definir scopes en ORMs como ActiveRecord, Mongoid o Mongomapper, de modo que su código se torna más sencillo, y la personalización del aspecto del paginador es bastante más elegante. De hecho, esta **personalización visual es extensible** en forma de THEMES, de modo que ya empiezan a aparecer algunas gemas con dichos themes para adoptar el aspecto de paginadores como los de GitHub, Google, etc.

Kaminari.paginate_array(my_array_object).page(params[:page]).per(10)

Veremos un ejemplo de paginación en el siguiente apartado cuando expliquemos las tres principales agrupaciones de funcionalidades que representa la aplicación.

PUNDIT

Es una gema que **autoriza** al usuario, ya sea invitado, registrado o administra, a hacer uso del sistema. Dependiendo de su autorización, podrá acceder a determinadas funcionalidades de la aplicación. Como por ejemplo, el administrador solo podrá acceder a los usuarios que hay registrados en el sistema, teniendo esta excepción el propietario y el paseador.

CARRIERWAVE

Esta gema escrita por Jonas Nicklas nos sirve para **añadir a nuestra aplicación la posibilidad de almacenar y procesar ficheros subidos por nuestros usuarios**. Si bien han existido numerosos plugins para realizar esta misma función anteriormente (attachment_fu, paperclip, y otros), esta gema utiliza una separación de código bastante limpia, guardando cada uploader como una clase propia donde podemos especificar reglas concretas para manipular esos ficheros que nos adjuntan.

Nosotros utilizaremos esta gema para almacenar los avatares de los perros que han sido registrados en el sistema por su propietario.

8.3. PROTOTIPO DEL SISTEMA

La aplicación se compone de una serie de funcionalidades que se clasifican en tres fundamentales grupos que explicaremos a continuación, y que son los que tratamos en este proyecto.

8.3.1. GESTIÓN DE USUARIOS

La gestión de usuarios se encarga, como su propio nombre indica, de gestionar todo lo relacionado con el usuario y esto es:

- **Darse de alta en el sistema**

Para que el usuario pueda hacer uso de los servicios que presta la aplicación, tendrá que registrarse previamente en el sistema. Su implementación se ha llevado a cabo gracias a la utilización de la gema Devise que hemos mencionado en el anterior apartado.

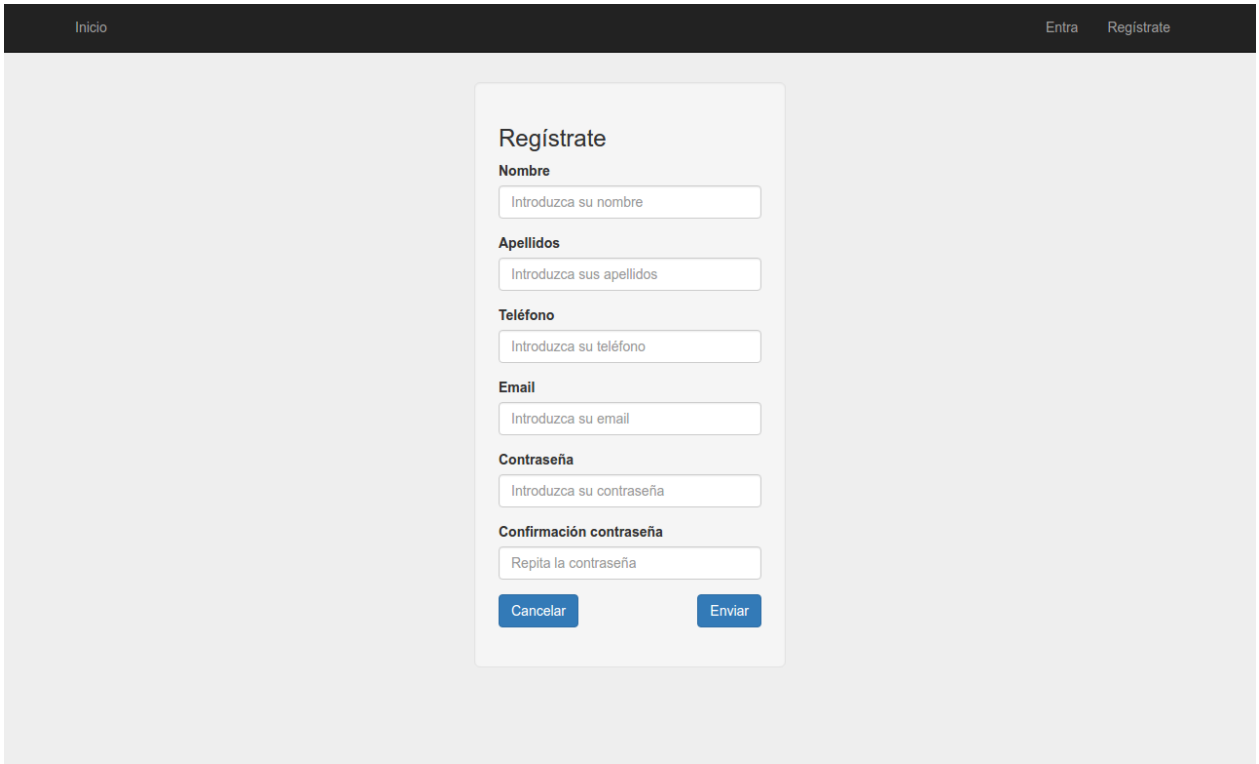


FIGURA 41 - DARSE DE ALTA EN EL SISTEMA

Y para acceder en las siguientes ocasiones al mismo, tendrá que autenticarse cada vez en la vista que se muestra en la **Figura 42**.

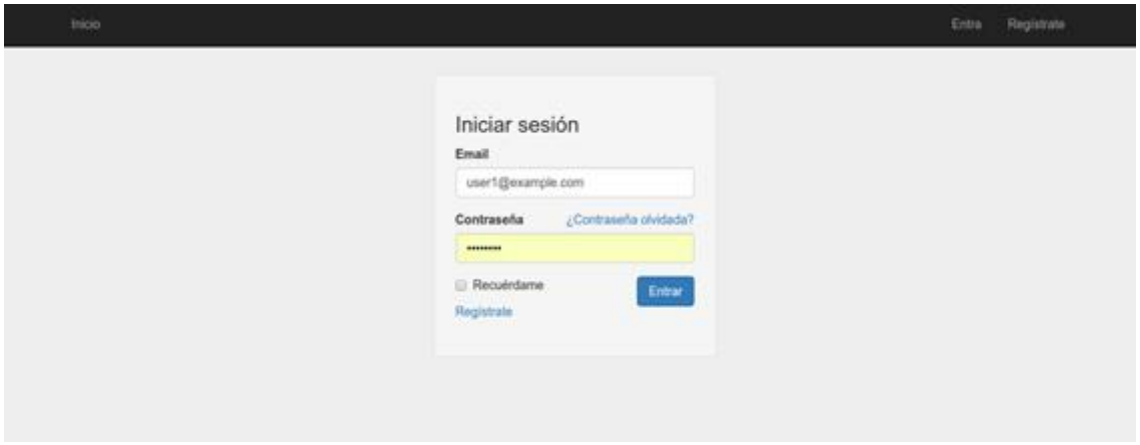


FIGURA 42 - AUTENTICACIÓN DE UN USUARIO REGISTRADO

Editar el perfil de usuario

En el perfil de usuario se podrá especificar qué tipo de usuario quiere ser, propietario, que es el rol que se le asigna por defecto una vez que se registra en el sistema, o paseador, que podrá cambiarse a dicho rol accediendo a su perfil.

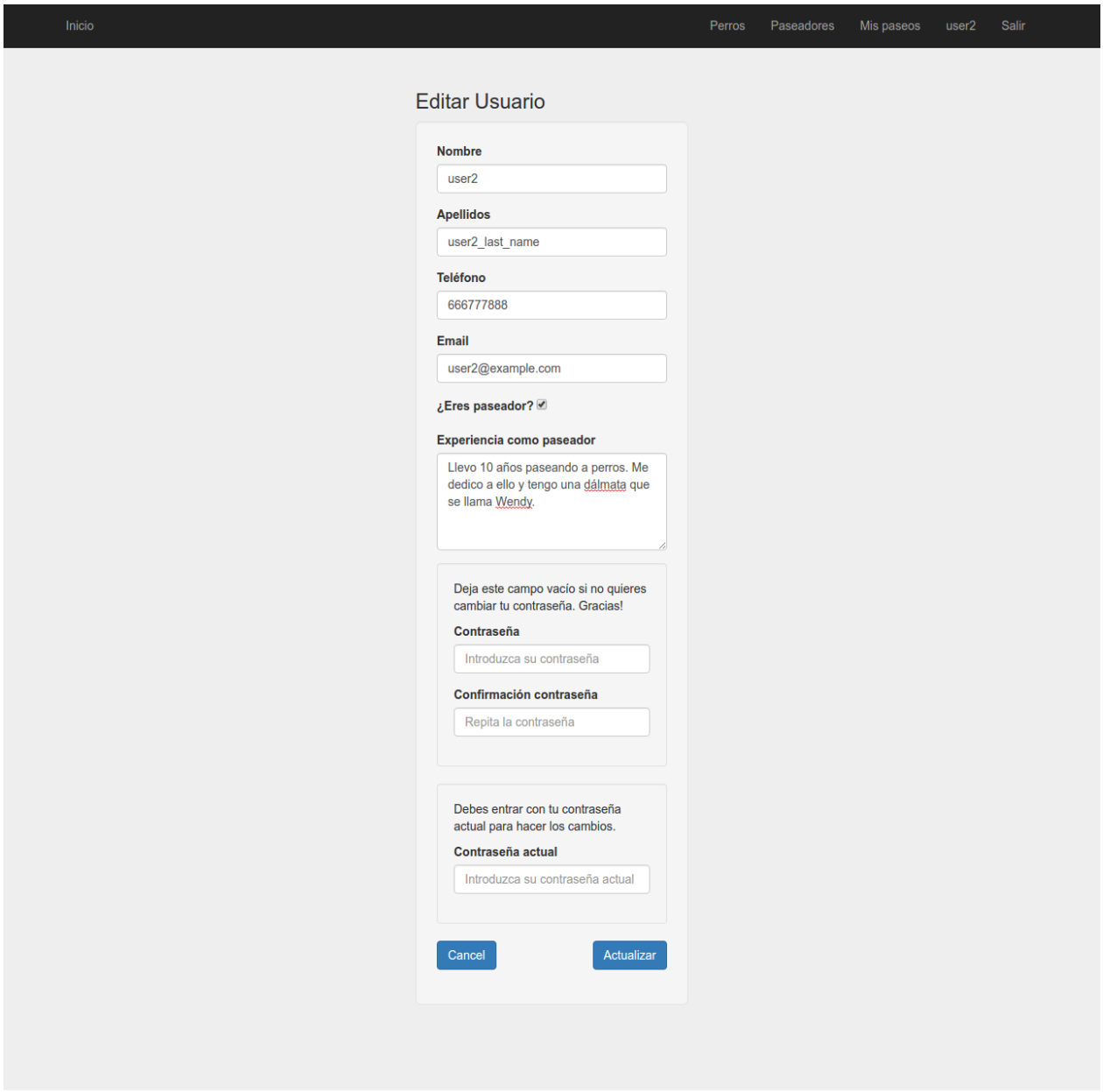


FIGURA 43 - FORMULARIO DE EDICIÓN DE UN USUARIO

- **Visualizar a los paseadores**

Esta funcionalidad proporciona un listado de paseadores para que el propietario pueda contratar a uno de ellos para que pasee a sus perros. No obstante, hay que tener en cuenta que, solo se puede contratar a un paseador si el usuario tiene registrado perros en el sistema.

Este listado facilita al usuario información referente del paseador, para que el usuario pueda decidir cuál es el paseador que desee, visualizando su experiencia con los perros.

FIGURA 44 - LISTADO DE PASEADORES

8.3.2. GESTIÓN DE MASCOTAS

La gestión de mascotas permitirá registrar a los perros de los usuarios en el sistema, así como editar su perfil cuando desee el propietario, visualizar el listado de los perros que han sido registrados y, ver y borrar las rutas que realiza el mismo cuando está siendo paseado por un paseador.

- **Registrar a un perro**

El usuario, una vez que ha sido registrado en el sistema, tiene que registrar a sus perros para que pueda hacer uso de los servicios que presta la aplicación, en este caso, de la visualización de sus perros cuando están siendo paseados, haciendo uso de la simulación GPS.

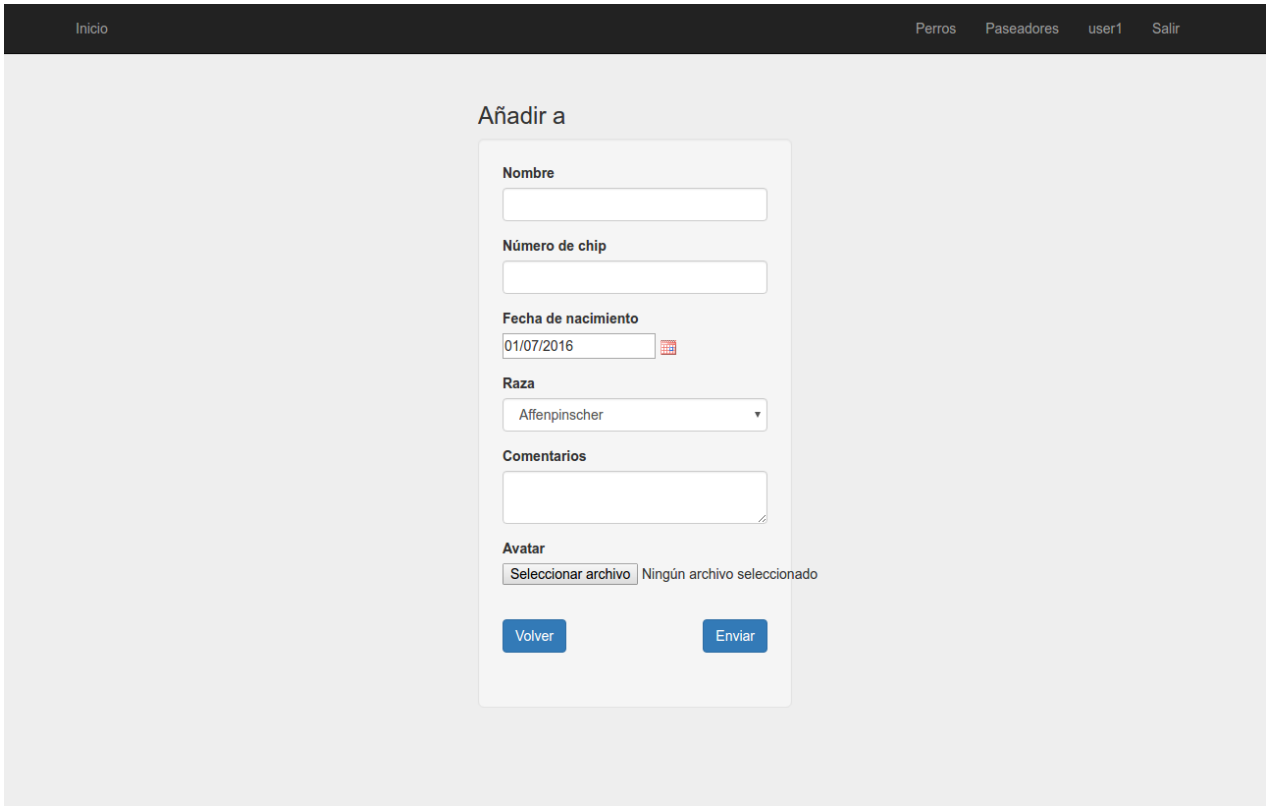


FIGURA 45 - FORMULARIO REGISTRO DE UN PERRO

Editar a un perro

El usuario en cualquier momento podrá modificar los datos de su mascota para tenerlo actualizado en el sistema.

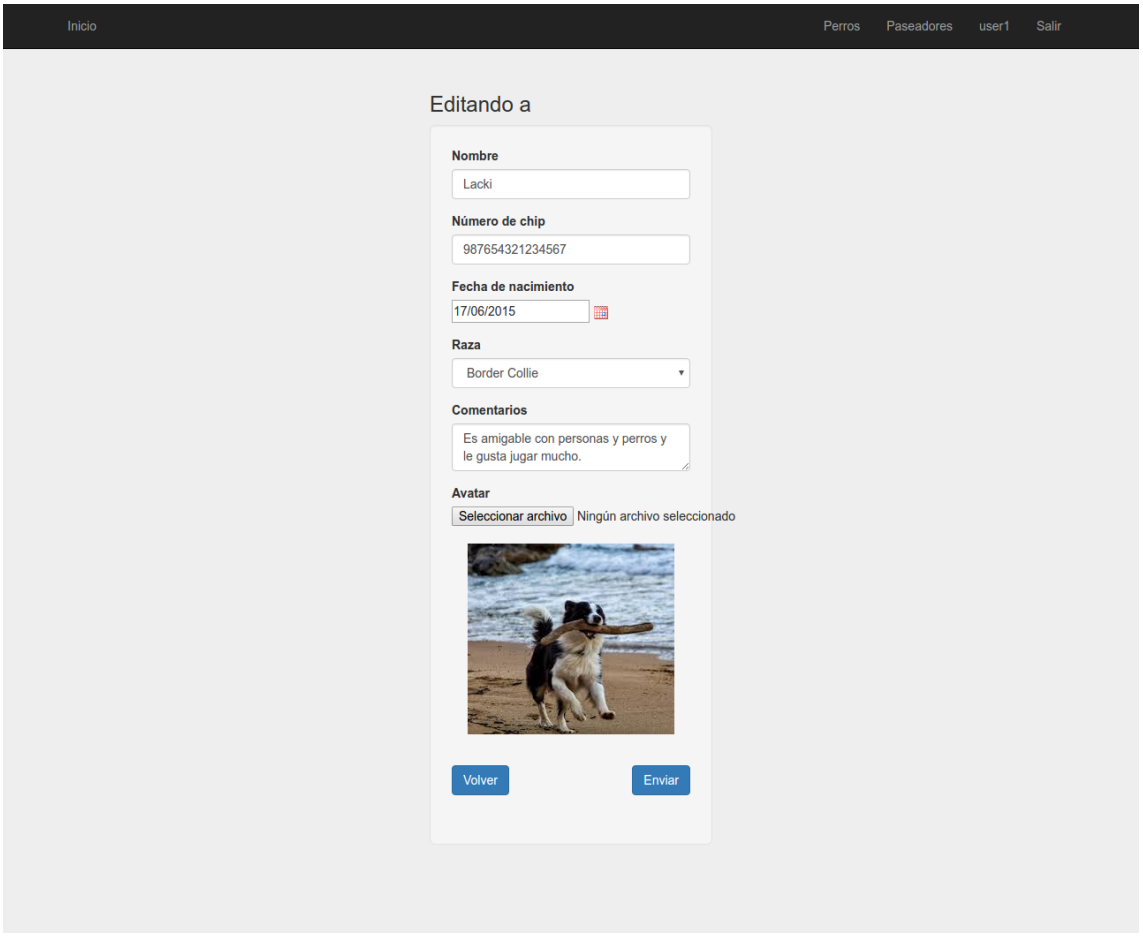


FIGURA 46 - FORMULARIO DE EDICIÓN DE UN PERRO

En la **Figura 46** se puede apreciar al lado del campo "Fecha de nacimiento" un icono representando un calendario, que lo obtenemos gracias al uso de la librería jQuery Datepicker mencionada con anterioridad.

Ver listado de perros

En esta funcionalidad e listarán los perros que han sido registrados por el usuario en el sistema y, los perros que ha de pasear un paseador una vez que ha sido contratado por un propietario.

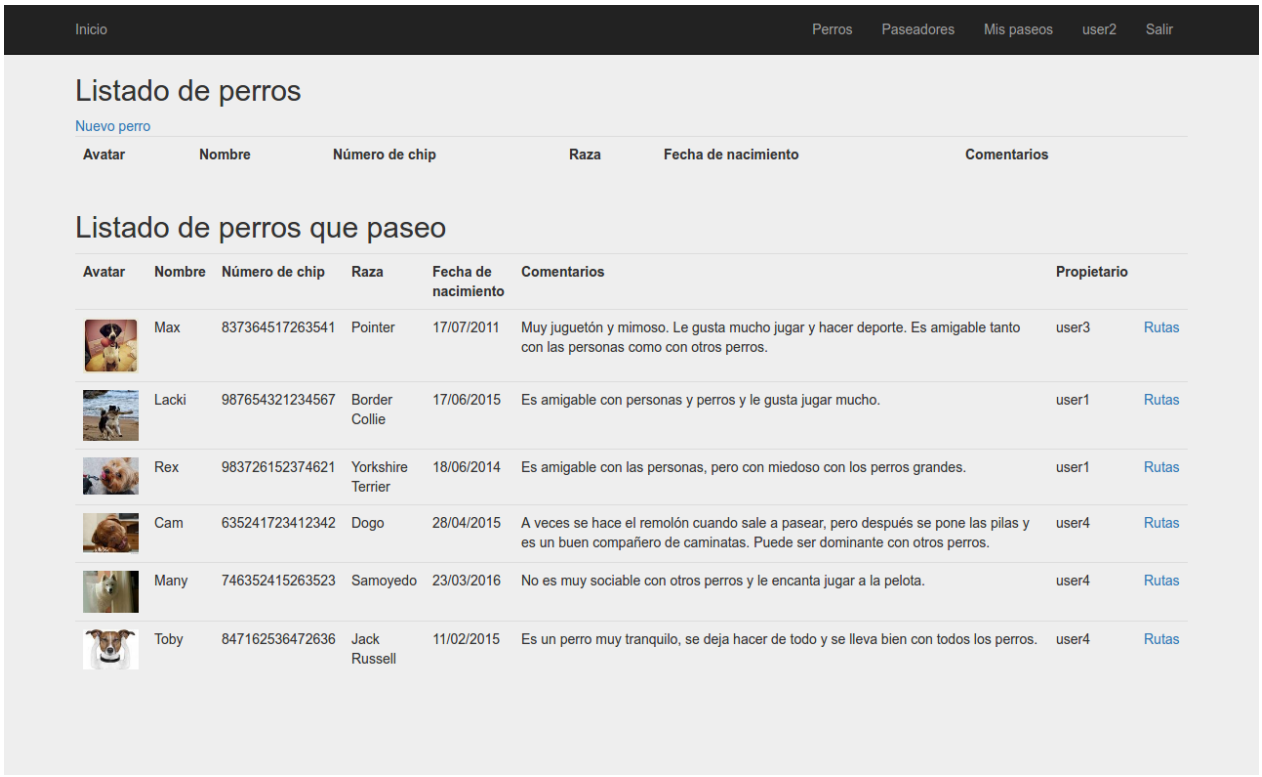


FIGURA 47 - LISTADO DE PERROS DE UN PASEADOR

8.3.3. GESTIÓN DE RUTAS

La gestión de rutas lista las rutas que ha estado y está haciendo el perro, mostrando una serie de información correspondiente a dichas rutas. Además, permite ver las rutas realizadas en el mapa y un listado de paseos que ha hecho el paseador con los perros de los propietarios que le han contratado.

Ver listado de rutas de un perro

Cuando se accede a las rutas de un perro, se podrá ver una vista como la que se muestra en la **Figura 48**.

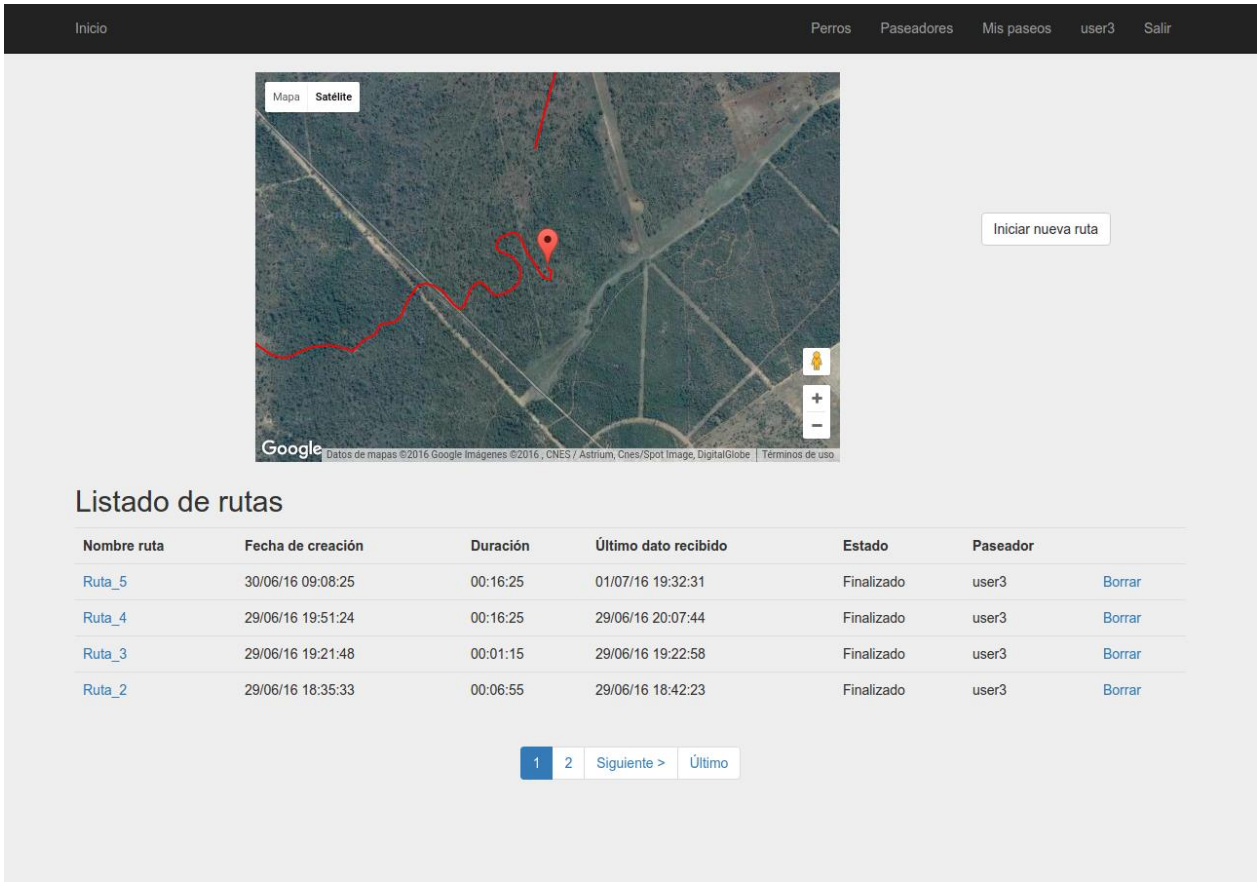


FIGURA 48 - LISTADO DE RUTAS DE UN PERRO

En esta imagen se puede apreciar la paginación que presenta la gema kaminari que se ha indicado con anterioridad. De esta manera, se mostrará un número determinado de rutas en el listado ordenando primero las más recientes en el tiempo.

Ver ruta de un perro

En cualquier momento el usuario podrá visualizar las rutas que ha hecho con anterioridad el animal.

En el caso del propietario, podrá ver todas las rutas en esta vista (/pets/pet_id/routes), y el paseador también si sigue contratado por un propietario, si no, solo podrá verlos en la vista de **Mis paseos** que se muestra en la **Figura 51**.

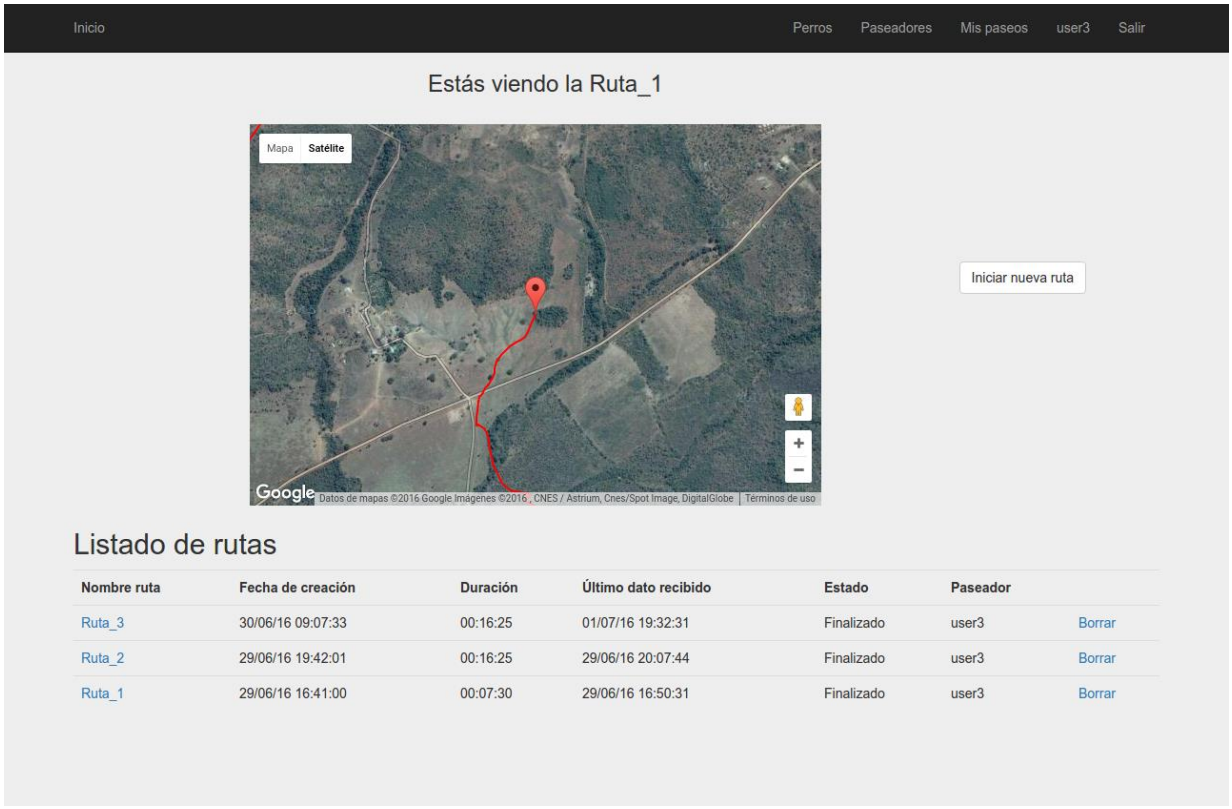


FIGURA 49 - VER RUTA DE UN PERRO REALIZADA CON ANTERIORIDAD

Iniciar nueva ruta

Solo el paseador tendrá la opción de ***Iniciar nueva ruta***, pues es él el que paseará a los perros de un usuario.

El propietario, sin embargo, podrá ver gracias a la simulación GPS, la ruta que están haciendo sus perros cuando un paseador lo está paseando, teniéndolos controlados en el momento del paseo.

La simulación GPS tratará de leer un fichero .kml guardándose las coordenadas en la base de datos para su posterior consulta y obtención de las coordenadas para mostrarlas en el mapa cada cinco segundos, que es el tiempo estimado para obtener las coordenadas del GPS en el mundo real.

A medida que se va actualizando el mapa cada cinco segundos, el usuario podrá ver además, cómo se actualiza los valores de duración, último dato recibido y el estado (en progreso o finalizado) de la ruta que está realizando in situ el perro.

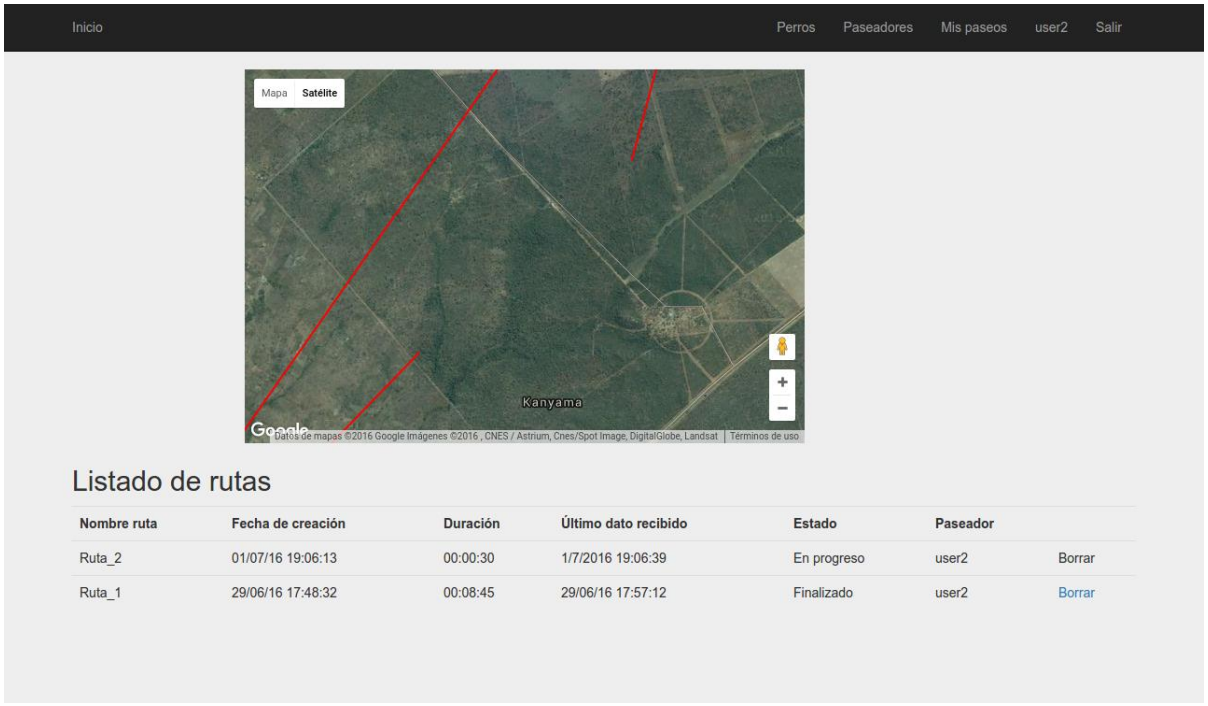


FIGURA 50 - INICIAR NUEVA RUTA DE UN PERRO

- **Ver los paseos de un paseador**

El paseador visualizará en esta vista todos los paseos que ha realizado con los perros de los usuarios que le han contratado.

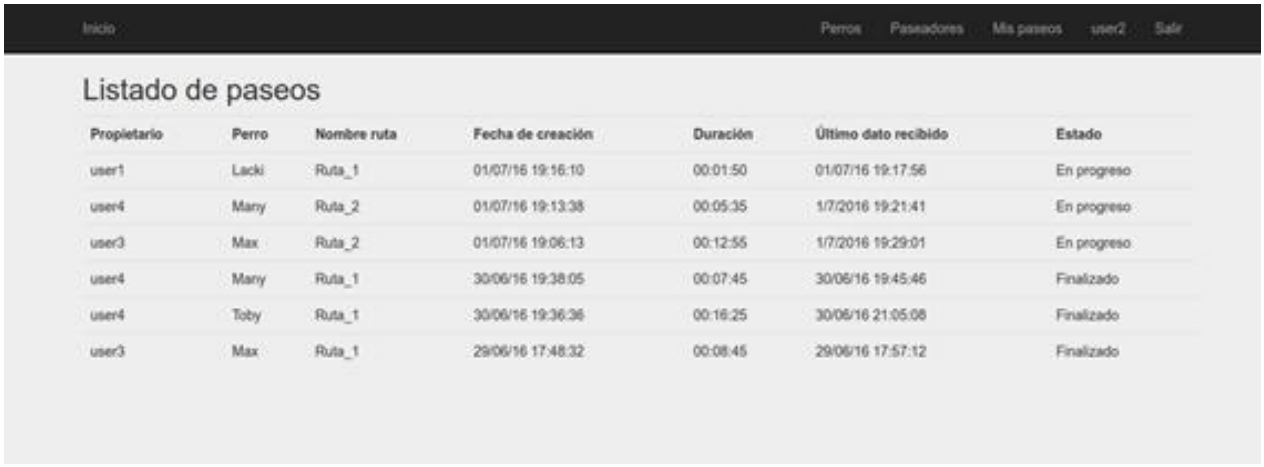


FIGURA 51 - LISTADO DE PASEOS DEL PASEADOR

9. CONCLUSIONES

En este apartado, para finalizar el presente proyecto, se va a valorar el trabajo realizado y la consecución de los objetivos gracias a ello.

El objetivo principal del presente proyecto ha sido desarrollar una aplicación para la geolocalización de mascotas. Para ello, se han utilizado metodologías de Ingeniería del Software ágil haciendo uso de prototipos, utilizadas en la mayor parte de las organizaciones a lo largo del ciclo de vida de las aplicaciones, lo que permite optimizar los riesgos y la incertidumbre asociados al desarrollo del software, debido a que los clientes pueden apreciar el avance del proyecto a edades tempranas del mismo. Por lo que el modo de desarrollo de la aplicación hará posible ampliaciones futuras y desarrollo de funcionalidades que no tienen que estar necesariamente ligadas al proyecto actual, pudiendo aportar un valor añadido a la sociedad.

Asimismo, en el presente proyecto se han unido muchos de los conocimientos adquiridos durante la carrera, como son las Redes de Computador, Ingeniería del Software, Bases de Datos o la Programación en sí mismas, sin los cuales no hubiera sido posible la realización del mismo. Y el hecho de haber fundido todos los conocimientos para la resolución de los problemas ha desembocado en un enriquecimiento directo en la forma de desarrollar el proyecto, puesto que siempre se han tenido que tener en cuenta todos ellos para desarrollar todas y cada una de las distintas funcionalidades.

Del mismo modo, se han usado un conjunto de tecnologías en base a las especificaciones y los problemas acaecidos, siendo estas tecnologías Ruby On Rails, PostgreSQL, ficheros KML, Mockups, etc. Este hecho tiene una relevancia importante puesto que se encuentran en auge actualmente y la explosión del uso de aplicaciones que funcionen a través de Internet hace que los proyectos puedan seguir manteniéndose por un conjunto de expertos mayores y los futuros desarrollos doten de una versatilidad mayor a este tipo de aplicaciones.

Finalmente, el uso particular de Heroku, ha dotado al proyecto de una mayor escalabilidad debido al uso de tecnologías cloud (Paas, Platform as a Service), debido a que ha sido diseñado y es utilizado en las organizaciones para dar servicio durante todas las fases del ciclo de desarrollo y pruebas del software.

10. TRABAJO FUTURO

Este ha sido el desarrollo del proyecto GPS - PET: Aplicación Web para la geolocalización de mascotas. Un prototipo para llevar a cabo la administración y gestión de nuestras mascotas y de sus rutas haciendo uso de la simulación del GPS.

A continuación se propone una serie de aspectos que podrían ser desarrollados de cara a mejorar y completar la aplicación web que ha sido implementada.

- Permitir realizar búsquedas en el listado de perros, rutas y paseos para cuando haya muchos datos, el usuario pueda buscar con facilidad el dato que desea.
- Poder compartir las rutas realizadas por el perro por las distintas redes sociales.
- Incrementar el número de funcionalidades del administrador para que pueda gestionar mejor los usuarios y perros registrados en la aplicación.
- Realizar un sistema de contratación adecuado para cuando el propietario vaya a contratar a un paseador.
- Añadir un chat interno para que el propietario pueda comunicarse con el paseador cuando esté paseando a sus perros.
- Añadir un blog de publicaciones para que los usuarios estén informados de cualquier información de animales publicadas en el exterior.

BIBLIOGRAFÍA

- Dondo. https://dondo.es/
- Position Logic. http://www.positionlogic.com/es/
- Tractive. https://tractive.com/es/
- Bootstrap. https://github.com/twbs/bootstrap-sass#a-ruby-on-rails
- Devise. https://github.com/plataformatec/devise
- Pundit. https://github.com/elabs/pundit
- Carrierwave. https://github.com/carrierwaveuploader/carrierwave
- Kaminari. https://github.com/amatsuda/kaminari
- Gmaps4Rails. https://github.com/apneadiving/Google-Maps-for-Rails
- Pg. https://rubygems.org/gems/pg/versions/0.18.2
- Gravatar. https://es.gravatar.com/
- Agile Web Development with Rails. 4ª Edition.
- API de Ruby on Rails. http://api.rubyonrails.org/
- Rails Guides. http://guides.rubyonrails.org/
- RailsCasts: Ruby on Rails screencasts. http://railscasts.com/
- Git Documentation. http://git-scm.com/documentation
- Stack Overflow. http://stackoverflow.com/
- Wikipedia. http://www.wikipedia.org/
- Heroku. http://www.heroku.com/
- Youtube. www.youtube.es
- Slideshare. www.slideshare.net
- MVC. https://es.wikipedia.org/wiki/Modelo-vista-controlador

ANEXO 1

CASOS DE USO DETALLADOS

PROPIETARIO

Nombre	Registrarse	ID	01
Actor principal			
Propietario			
Personal involucrado o intereses			
• Propietario: El usuario pretende introducir sus datos en el sistema para quedar registrado y poder acceder a las funcionales determinadas para el mismo.			
Descripción			
El usuario tiene la posibilidad de rellenar un formulario de registro. Dicho formulario estará constituido por: nombre, apellidos, teléfono, correo electrónico, contraseña, confirmación de contraseña y opción para indicar si desea ser paseador. A excepción del teléfono, todos estos datos son obligatorios para poder llevar a cabo el registro de usuario en el sistema.			
Trigger			
Hacer "click" en el link **Regístrate**.			
Precondición			
Postcondición			
• El usuario queda registrado en el sistema.			
Flujo normal			
1. El usuario entra en la sección de **Regístrate**. **2.** El usuario rellena el formulario de registro y hace "click" en el botón **Enviar**. **3.** El sistema valida los datos introducidos. **4.** El sistema crea un nuevo usuario en el sistema.			
Flujo alternativo			
3.1. Si al validar los datos, el sistema detecta alguno incorrecto, se lo comunica al usuario volviendo al paso 2. **3.1.1.** Vuelve al paso 2 siguiendo nuevamente el flujo de forma normal.			
Excepción			
Includes			
Requisitos especiales			
Notas			

Nombre	Editar perfil	ID	02
Actor principal			
Propietario			
Personal involucrado o intereses			
• Propietario: El usuario registrado quiere actualizar sus datos personales almacenados en el sistema, pudiendo modificar a su vez su rol, convirtiéndose en paseador.			
Descripción			
El usuario actualiza sus datos personales en el sistema, dando permiso para ello, a través de su contraseña. Además, el usuario registrado, teniendo asignado a priori el rol de propietario, tiene la posibilidad de convertirse en paseador para poder pasear a los perros de otros propietarios registrados en el sistema.			
Trigger			
Hacer "click" en el enlace **Perfil.**			
Precondición			
• Estar dado de alta en la aplicación.			
Postcondición			
• El perfil del cliente queda modificado a sus intereses.			
Flujo normal			
1. Hace "click" en el enlace **Perfil.** **2.** El cliente modifica sus datos personales. **3.** Introduce contraseña para confirmar los cambios a realizar y hace "click" en el botón **Enviar**. **4.** El sistema valida los datos modificados. **5.** El sistema actualiza la información en el mismo.			
Flujo alternativo			
4.1. Si al validar los datos, el sistema detecta alguno incorrecto, se lo comunica al usuario volviendo al paso 2. **4.1.1.** Vuelve al paso 2 siguiendo nuevamente el flujo de forma normal.			
Excepción			
Includes			
Requisitos especiales			
Notas			

Nombre	Crear perro	ID	03
Actor principal			
Propietario			
Personal involucrado o intereses			
• Propietario: Desea registrar un perro vinculado a su cuenta.			
Descripción			
El usuario registrado añade un perro a su listado de perros.			
Trigger			
Hacer "click" en **Nuevo perro.**			
Precondición			
• El usuario debe estar registrado en el sistema.			
Postcondición			
• Un nuevo perro ha sido creado y añadido a la cuenta del usuario.			
Flujo normal			
1. Hace "click" en el enlace Nuevo perro. **2.** Rellena el formulario del perro y hace "click" en el botón **Enviar**. **3.** El sistema valida los datos introducidos. **4.** El sistema almacena los datos del ejemplar vinculando éste al usuario.			
Flujo alternativo			
3.1. Si al validar los datos, el sistema detecta alguno incorrecto, se lo comunica al usuario volviendo al paso 2. **3.1.1.** Vuelve al paso 2 siguiendo nuevamente el flujo de forma normal.			
Excepción			
Includes			
Requisitos especiales			
Notas			

Nombre	Editar perro	ID	04
Actor principal			
Propietario			
Personal involucrado o intereses			
• Propietario: Quiere cambiar información respecto a algún perro registrado en su cuenta.			
Descripción			
Se cambia información referente a algún perro del usuario almacenada en el sistema.			
Trigger			
Hacer "click" en **Editar.**			
Precondición			
• Tener algún perro vinculado al usuario. • Estar visualizando la ficha del ejemplar a editar en cuestión.			
Postcondición			
• El dato del perro queda modificado a interés del usuario.			
Flujo normal			
1. Hace "click" en el enlace **Perros**. **2.** Busca el perro en el listado de perros cuya información quiere editar. **3.** Hace "click" en el enlace **Editar**. **4.** El cliente modifica los datos del perro y hace "click" en el botón **Enviar**. **5.** El sistema valida los datos introducidos. **6.** El sistema acepta los cambios y actualiza la información en el mismo.			
Flujo alternativo			
5.1. Si al validar los datos, el sistema detecta alguno incorrecto, se lo comunica al usuario volviendo al paso 4. **5.1.1.** Vuelve al paso 4 siguiendo nuevamente el flujo de forma normal.			
Excepción			
Includes			
Requisitos especiales			
Notas			

Nombre	Borrar perro	ID	05
Actor principal			
Propietario			
Personal involucrado o intereses			
• Propietario: El usuario registrado quiere borrar a su perro del sistema.			
Descripción			
El usuario registrado decide que no quiere tener el perro registrado en el sistema.			
Trigger			
Hacer "click" en el enlace **Borrar**.			
Precondición			
• El perro debe estar registrado en el sistema para ser borrado.			
Postcondición			
• El perro ha sido borrado del sistema.			
Flujo normal			
1. El usuario entra en la sección de **Perros**. **2.** El usuario busca al perro y hace "click" en el botón **Borrar**. **3.** El sistema se asegura de que el usuario desea eliminar a su perro del sistema. **4.** Se confirma el borrado.			
Flujo alternativo			
4.1. El usuario no confirma el borrado del perro. **4.1.1.** El sistema no varía respecto a cómo estaba antes de empezar la operación.			
Excepción			
Includes			
Requisitos especiales			
Notas			

Nombre	Ver ruta	ID	06
Actor principal			
Propietario			
Personal involucrado o intereses			
• Propietario: El cliente quiere ver la ruta que está realizando su perro cuando está siendo paseado por el paseador contratado o, quiere ver una ruta que ya ha sido finalizada.			
Descripción			
El usuario accede al listado de rutas de su perro para visualizar en el mapa la ruta que está realizando en el momento que está siendo paseado por el paseador o, que ha realizado con anterioridad.			
Trigger			
Hacer "click" en el enlace **Rutas.**			
Precondición			
• El perro tiene que estar registrado en el sistema y vinculado a la cuenta del usuario.			
Postcondición			
• El usuario visualiza la ruta del perro en el mapa.			
Flujo normal			
1. Hace "click" en el enlace **Perros**. **2.** Busca el perro en el listado de peros cuya ruta quiere visualizar en el mapa. **3.** Hace "click" en el enlace **Rutas.** **4.** El cliente visualiza la lista de rutas **5.** Selecciona una de ellas visualizándola en el mapa, estando dicha ruta finalizada o en progreso.			
Flujo alternativo			
Excepción			
Includes			
Requisitos especiales			
Notas			

Nombre	Borrar ruta	ID	07
Actor principal			
Propietario			
Personal involucrado o intereses			
• Propietario: El usuario registrado desea borrar una ruta que ha sido realizada por su perro cuando el paseador ha ido a pasearlo.			
Descripción			
El usuario registrado decide borrar del sistema una ruta que ha sido realizada por su perro cuando ha sido paseado por un paseador.			
Trigger			
Hacer "click" en el link **Borrar**.			
Precondición			
• La ruta debe estar almacenada en el sistema.			
Postcondición			
• La ruta del perro ha sido borrada del sistema.			
Flujo normal			
1. El usuario entra en la sección de **Perros**. **2.** El usuario busca en el listado de perros al perro que quiere borrar una ruta del mismo. **3.** El usuario busca la ruta a eliminar en la lista de rutas del perro y hace "click" en el botón **Borrar**. **4.** El sistema se asegura de que el usuario desea elimina una ruta del sistema. **5.** Se confirma el borrado.			
Flujo alternativo			
4.2. El usuario no confirma el borrado de la ruta del perro. **4.2.1.** El sistema no varía respecto a cómo estaba antes de empezar la operación.			
Excepción			
Includes			
Requisitos especiales			
Notas			

Nombre	Visualizar paseadores	ID	08
Actor principal			
Propietario			
Personal involucrado o intereses			
• Propietario: El usuario registrado desea visualizar los paseadores que han sido registrados en el sistema.			
Descripción			
El usuario puede ver a todos los paseadores que se han dado de alta en la aplicación.			
Trigger			
Hacer "click" en el enlace **Paseadores.**			
Precondición			
• El usuario debe estar registrado en el sistema.			
Postcondición			
• Visualiza todos los paseadores registrados.			
Flujo normal			
1. Hace "click" en el enlace **Paseadores.** **2.** Visualiza a los paseadores.			
Flujo alternativo			
Excepción			
3.* Si no hay paseadores registrados, la lista estará vacía.			
Includes			
Requisitos especiales			
Notas			

Nombre	Contratar paseador	ID	09
Actor principal			
Propietario			
Personal involucrado o intereses			
• Propietario: El usuario registrado quiere contratar a un paseador para que pasee a sus perros.			
Descripción			
El usuario registrado decide contratar a un paseador para que éste pasee a sus perros.			
Trigger			
Accionar el botón **Contratar**.			
Precondición			
• El usuario debe estar registrado en el sistema. • El usuario debe haber registrado a sus perros.			
Postcondición			
• El usuario ha contratado a un paseador. • Se le asignan automáticamente los perros del propietario que le ha contratado.			
Flujo normal			
1. El usuario entra en **Paseadores**. **2.** El usuario busca un paseador en el listado de paseadores. **3.** Hace "click" en el botón **Contratar** del paseador a contratar.			
Flujo alternativo			
3.1. Una vez contratado el paseador, s el propietario desea dejar de utilizar los servicios de éste, tiene que clicar en el botón **Liberar**.			
Excepción			
3.* Solo se puede contratar un paseador. Si desea contratar a otro paseador, tiene que dejar de usar los servicios del que tiene contratado actualmente.			
Includes			
Requisitos especiales			
Notas			

PASEADOR

Nombre	Ver perros que pasea	ID	10
Actor principal			
Paseador			
Personal involucrado o intereses			
• Paseador: El usuario desea ver qué perros tiene que pasear.			
Descripción			
Cuando el usuario ha sido contratado por un propietario, automáticamente se le asigna sus perros, visualizándolos en un listado aparte, separando sus perros de los que pasea.			
Trigger			
Hacer "click" en el enlace **Perros.**			
Precondición			
• Estar registrado en el sistema como paseador. • Haber sido contratado por un propietario.			
Postcondición			
• Ver los perros de los propietarios que han contratado al usuario.			
Flujo normal			
1. Hace "click" en el enlace **Perros.** **2.** Visualizar los perros que pasea y los suyos propios en caso de que los tuviera registrados en el sistema.			
Flujo alternativo			
Excepción			
2.* Si no ha sido contratado por ningún propietario, la tabla saldrá vacía.			
Includes			
Requisitos especiales			
Notas			

Nombre	Ver mis paseos	ID	11
Actor principal			
Paseador			
Personal involucrado o intereses			
• Paseador: El usuario registrado quiere ver todos los paseos que está haciendo o que ha hecho con anterioridad.			
Descripción			
El usuario al clicar en el enlace "Mis paseos", podrá ver el listado de paseos que ha hecho con los perros que pasea, visualizando en el mismo el nombre del paseo, el tiempo que ha durado el paseo con cada perro, la hora inicio y fin del paseo, y el nombre del perro y del propietario.			
Trigger			
Hacer "click" en el enlace **Mis paseos**.			
Precondición			
• El usuario debe haber sido dado de alta en el sistema. • El usuario tuvo que haber sido contratado por un propietario. • El usuario tiene que tener asignado a los perros del propietario que le ha contratado.			
Postcondición			
• Visualiza los paseos que está realizando actualmente como los que ha hecho con anterioridad.			
Flujo normal			
1. Hace "click" en el enlace **Mis paseos**.			
Flujo alternativo			
Excepción			
Includes			
Requisitos especiales			
Notas			

Nombre	Iniciar nueva ruta	ID	12
Actor principal			
Paseador			
Personal involucrado o intereses			
• Paseador: El usuario registrado quiere iniciar una nueva ruta del perro que está paseando.			
Descripción			
El usuario acciona el botón "Iniciar nueva ruta" cuando va a realizar un paseo con el perro de un propietario que le ha contratado. La función que va a desempeñar es la de simular el GPS.			
Trigger			
Accionar el botón **Iniciar nueva ruta**.			
Precondición			
• El usuario debe haber sido dado de alta en el sistema. • El usuario tuvo que haber sido contratado por un propietario. • El usuario tiene que tener asignado a los perros del propietario que le ha contratado.			
Postcondición			
• Crea una nueva ruta del perro que está paseando, visualizando dicha ruta en el mapa.			
Flujo normal			
1. Hace "click" en el enlace **Perros**. 2. Busca al perro que va a pasear. 3. Hace "click" en el enlace **Rutas** del perro. 4. Hace "click" en el botón "Iniciar nueva ruta" para crear una nueva ruta.			
Flujo alternativo			
Excepción			
Includes			
Requisitos especiales			
Notas			

ADMINISTRADOR

Nombre	Modificar rol usuario	ID	13
Actor principal			
Administrador			
Personal involucrado o intereses			
• Administrador: Quiere modificar el rol de un usuario registrado.			
Descripción			
El usuario modifica el rol de un usuario que sido dado de alta en el sistema.			
Trigger			
Clicar en el enlace **Cambiar rol.**			
Precondición			
• El usuario tiene que estar registrado en el sistema.			
Postcondición			
• El rol del cliente ha sido modificado.			
Flujo normal			
1. Hace "click" en el enlace **Usuarios.** **2.** Busca el usuario a cambiar el rol. **3.** Selecciona el nuevo rol. **4.** Acciona en el botón **Cambiar rol**. **5.** El sistema se asegura de que el administrador desea eliminar a un usuario. **6.** Se confirma el cambio de rol.			
Flujo alternativo			
5.1. El administrador no confirma el borrado del usuario. **5.1.1.** El sistema no varía respecto a cómo estaba antes de empezar la operación.			
Excepción			
Includes			
Requisitos especiales			
Notas			

Nombre	Borrar usuarios registrados	ID	14
Actor principal			
Administrador			
Personal involucrado o intereses			
• Administrador: Quiere borrar un usuario.			
Descripción			
Opción que permite al administrador eliminar la cuenta de un usuario del sistema. Una vez terminada la operación, la cuenta quedará totalmente borrada del sistema.			
Trigger			
Accionar en el botón **Borrar**.			
Precondición			
• El usuario a borrar tiene que estar registrado en el sistema.			
Postcondición			
• El usuario queda eliminado del sistema.			
Flujo normal			
1. Hace "click" en el enlace **Usuarios.** **2.** Busca el usuario a eliminar. **3.** Hace "click" en el botón **Borrar**. **4.** El sistema se asegura de que el administrador desea eliminar a un usuario. **5.** Se confirma el borrado.			
Flujo alternativo			
4.1. El administrador no confirma el borrado del usuario. **4.1.1.** El sistema no varía respecto a cómo estaba antes de empezar la operación.			
Excepción			
Includes			
Requisitos especiales			
Notas			

ANEXO 2 – GLOSARIO DE CONCEPTOS

Para que se entiendan los términos que se van a utilizar en este proyecto, se va a definir cada uno de ellos:

- **Usuario registrado**: Persona que tiene acceso a las funcionales e información proporcionada por el sistema.
- **Paseador**: Usuario registrado que se ha dado de alta en el sistema, cuyas funcionalidades son las mismas que las del propietario, con la excepción de que su perfil es público ante los demás usuarios registrados.
- **Propietario**: Usuario que está dado de alta en la aplicación y que ha accedido a su cuenta, teniendo unos permisos específicos para poder hacer uso de una serie de funciones.
- **Administrador**: Usuario registrado que tiene privilegios para acceder a determinadas funcionalidades del sistema que un usuario registrado común no tiene.
- **Perro, can**: Animal de compañía que será registrado en el sistema por su propietario.
- **Número de chip / microchip**: dispositivo del tamaño de un grano de arroz que contiene una numeración que permite identificar a un animal.
- **Raza**: especie de animal identificado por un conjunto de características comunes.
- **GPS**: Sistema de Posicionamiento Global que permite determinar en toda la Tierra la posición de un objeto (persona, cosa…) con una precisión de hasta centímetros.
- **Fichero .kml**: es un formato de archivo que se utiliza para mostrar datos geográficos en un navegador terrestre, como Google Earth, Google Maps y Google Maps para móviles.
- **Trayectoria**: Camino que ha establecido el can cuando el propietario desea visualizar cuál es su ubicación in situ.
- **Rutas**: Trayectoria finalizada por el perro, cuyo camino ha sido guardado para su posterior visualización, o borrado, en caso de que no se desee tener en el historial de rutas del perro.