



F O R M A C I Ó N

Grado Superior  
Desarrollo Aplicaciones Web



F O R M A C I Ó N

Trabajo fin de Grado

---

**DogGo!**

*Para ti, por ellos.*

Plataforma para conectar paseadores con dueños.

---

**Autor:** Raúl Navarro Serra

**Tutores:** Fortunato Yekue / Luís García

Mayo 2025



F O R M A C I Ó N

Trabajo fin de Grado

---

**DogGo!**

*Para ti, por ellos.*

Plataforma para conectar paseadores con dueños.

---

**Autor:** Raúl Navarro Serra

**Tutores:** Fortunato Yekue / Luís García

---

## Declaración de la autoría:

*Yo, Raúl Navarro Serra, declaro la autoría del Trabajo Fin de Grado titulado “DogGo! Para ti, por ellos.” y que el citado trabajo no infringe las leyes en vigor sobre la propiedad intelectual. El material no original que figura en este trabajo ha sido atribuido a sus legítimos autores.*

Valencia, mayo de 2025.

Fdo: Raúl Navarro Serra.

---

## Resumen

**DogGo!** - “**Para ti, por ellos**” es una aplicación cuyo objetivo principal es facilitar el encuentro entre personas interesadas en ofrecer y contratar servicios de paseo para perros. A través de un sistema de registro por roles, se distinguen dos tipos de usuarios: **dueños de perros** y **paseadores**. La plataforma actúa como un punto de conexión entre ambos, permitiendo que se llegue fácilmente a acuerdos para el cuidado temporal de las mascotas.

En el proceso de registro, cada usuario selecciona su rol, lo que condiciona las funcionalidades a las que podrá acceder dentro de la aplicación.

Los **dueños de perros** pueden:

- Editar su información personal (correo electrónico, nombre, apellidos y contraseña), así como eliminar su cuenta si lo desean.
- Registrar uno o varios perros, sin límite establecido por el momento. Cada perro puede tener asignado un nombre, raza, edad y, próximamente, una imagen. También es posible editar o eliminar esta información en cualquier momento.
- Visualizar los paseos activos que ha solicitado o que se encuentran en curso.
- Solicitar un servicio de paseo consultando los anuncios disponibles creados por los paseadores.

Por su parte, los **paseadores** disponen de las siguientes funcionalidades:

- Modificar sus datos personales (correo electrónico, nombre, apellidos y contraseña) y eliminar su cuenta.
  - Crear un único anuncio personal donde pueden presentarse, indicando su nombre, localidad, una breve biografía y, próximamente, una fotografía. Este anuncio puede ser editado, eliminado o desactivado temporalmente (en cuyo caso no será visible para los dueños).
  - Recibir solicitudes de paseo por parte de los dueños, que pueden aceptar o rechazar según su disponibilidad o interés.
-

## **Agradecimientos:**

En primer lugar, me gustaría expresar mi más sincero agradecimiento a mis profesores, **Luis** y **Fortu**, por su constante disposición, apoyo y orientación a lo largo de todo este proceso. Su implicación y cercanía han sido clave para poder superar los retos que han ido surgiendo.

También quiero agradecer a mis compañeros de clase, por su ayuda incondicional y su presencia diaria. Gracias a ellos, estos dos años han sido mucho más llevaderos, amenos y enriquecedores. Su apoyo ha sido una fuente constante de motivación.

Por último, pero no por ello menos importante, deseo dedicar unas palabras especiales a mi familia, y muy especialmente a mis padres. Ellos han sido siempre mi mayor pilar, brindándome su apoyo incondicional en todo momento. A los 23 años tomé la decisión de volver a estudiar y cursar un ciclo superior, y sin dudarlo ni un segundo, me respaldaron completamente. Sin su aliento, confianza y esfuerzo, este logro no habría sido posible. Este trabajo también es, en parte, suyo.

---

## Índice general

1. Introducción	08
1.1. Introducción .....	09
1.2. Motivación .....	10
1.3. Objetivos .....	11
1.4. Organización de la memoria .....	12
2. Estado del arte	13
2.1. Análisis de aplicaciones similares .....	14
2.2. Tecnologías .....	15
3. Requisitos, especificaciones, coste, riesgos, viabilidad	17
3.1. Requisitos .....	18
3.2. Especificaciones .....	19
3.3. Costes .....	27
3.4. Riesgos .....	28
3.5. Viabilidad .....	28
4. Análisis	29
5. Diseño	33
6. Implementación y pruebas	36
6.1. Implementación .....	36
6.2. Pruebas funcionales .....	37
6.3. Pruebas de rendimiento .....	37
6.4. Pruebas de usabilidad .....	37
7. Conclusiones	38
Apéndice	39
Bibliografía	40

# Capítulo 1

## Introducción

### 1.1. Introducción

### 1.2. Motivación

### 1.3. Objetivos

### 1.4. Organización de la memoria



# 1. Introducción

## 1.1 Introducción

**DogGo!** – “**Para ti, por ellos**” surge como respuesta a una necesidad cada vez más presente en la sociedad actual: la falta de tiempo para dedicar al cuidado de las mascotas, en concreto al paseo diario de los perros. Muchos dueños, debido a sus rutinas laborales, responsabilidades familiares u otras circunstancias, no disponen del tiempo suficiente para garantizar este aspecto fundamental del bienestar animal.

La plataforma propuesta, desarrollada como una aplicación web, tiene como objetivo facilitar la conexión entre **dueños de perros** que necesitan ayuda y **paseadores** dispuestos a ofrecer este servicio de forma segura y organizada. A través de una interfaz sencilla e intuitiva, **DogGo!** permite gestionar perfiles de usuario, registrar mascotas, publicar y consultar anuncios, así como gestionar solicitudes de paseo y establecer una comunicación efectiva entre ambas partes.

Esta solución digital no solo pretende optimizar la gestión del tiempo para los dueños, sino también ofrecer una oportunidad laboral flexible para quienes deseen trabajar como paseadores, fomentando así una red colaborativa centrada en el respeto y el cuidado de los animales.

## 1.2. Motivación

La motivación principal de este proyecto nace del vínculo especial con mi perro **Rocky**, que, con sus 18 años de vida, continúa ofreciendo amor y compañía como el primer día que llegó a casa. Esta experiencia personal ha reforzado mi convicción de que todos los perros merecen disfrutar de paseos diarios, momentos de aire fresco, ejercicio y felicidad que son esenciales para su bienestar físico y emocional.

Sin embargo, la realidad es que muchos dueños, por motivos laborales o personales, no disponen del tiempo suficiente para cumplir con esta necesidad básica. Esta situación fue el punto de partida para idear **DogGo!**, una plataforma que sirviera como puente entre propietarios de mascotas y paseadores responsables, facilitando una solución práctica, segura y accesible para ambas partes. Este proyecto nace, por tanto, del amor hacia los animales y del deseo de contribuir a su bienestar, ofreciendo al mismo tiempo una oportunidad de colaboración y beneficio mutuo.

### 1.3. Objetivos

El desarrollo de este proyecto persigue una serie de objetivos tanto técnicos como funcionales, orientados a ofrecer una solución práctica y accesible para la gestión de servicios de paseo de perros. Los objetivos son:

- **Desarrollar una plataforma web** que permita establecer una conexión directa entre **dueños de perros** y **paseadores**, fomentando así una red de colaboración centrada en el bienestar animal.
- **Implementar funcionalidades clave**, como el sistema de **registro e inicio de sesión**, la **gestión de perfiles de usuario**, el **registro y administración de perros**, y la **creación y visualización de anuncios de paseo**.
- **Facilitar la gestión de solicitudes de paseo**, permitiendo a los dueños enviar peticiones a paseadores, y a estos últimos aceptarlas o rechazarlas de forma sencilla, garantizando una comunicación eficaz entre ambas partes.
- **Diseñar una interfaz intuitiva y funcional**, que asegure una experiencia de usuario clara, accesible y cómoda para cualquier tipo de usuario, independientemente de sus conocimientos técnicos.

## 1.4. Organización de la memoria

En primer lugar, se expone el **estado del arte**, donde se analizan aplicaciones similares existentes en el mercado, así como las tecnologías utilizadas para el desarrollo de la plataforma.

A continuación, se detallan los **requisitos y especificaciones** de la aplicación web, junto con un análisis de los **costes estimados**, los **riesgos asociados** y la **viabilidad técnica** del proyecto.

Posteriormente, se describe el proceso de **análisis y diseño**, donde se explica la arquitectura de la solución, las decisiones técnicas adoptadas y la estructuración de las funcionalidades.

El siguiente bloque está dedicado a la **implementación y pruebas**, mostrando las distintas fases de desarrollo, los resultados obtenidos y la validación del correcto funcionamiento del sistema.

Finalmente, se presentan las **conclusiones**, una **revisión de los costes** finales y una propuesta de posibles **mejoras o líneas de trabajo futuro**.

La memoria concluye con la correspondiente **bibliografía**, que recoge todas las fuentes consultadas durante el desarrollo del proyecto.

---

# Capítulo 2

## Estado del arte

### 2.1. Análisis de aplicaciones similares

### 2.2. Tecnologías

## 2. Estado del arte.

Antes de comenzar el desarrollo de la plataforma **DogGo!**, se ha llevado a cabo una investigación previa sobre soluciones similares ya existentes en el mercado. Este análisis tiene como objetivo comprender cómo otros sistemas resuelven la misma problemática, identificar buenas prácticas, detectar posibles carencias y establecer una base sólida sobre la que construir un producto diferencial.

### 2.1. Análisis de aplicaciones similares

**DogGo!** – “**Para ti, por ellos**” toma como referencia el funcionamiento de plataformas generalistas como **Wallapop** o **Milanuncios**, ampliamente reconocidas por permitir la publicación de anuncios clasificados y facilitar la comunicación directa entre usuarios. Estas aplicaciones han demostrado ser altamente efectivas en la creación de redes de intercambio y prestación de servicios entre particulares, gracias a su simplicidad, alcance y modelo de interacción descentralizado.

Sin embargo, a diferencia de estas plataformas, **DogGo!** se especializa en un servicio muy concreto: el **paseo de perros**. Este enfoque permite adaptar la experiencia de usuario a las necesidades específicas tanto de los dueños como de los paseadores, ofreciendo una solución más personalizada, clara y eficaz.

Mientras que en Wallapop o Milanuncios los usuarios deben navegar entre múltiples categorías, filtros genéricos y una oferta muy variada, **DogGo!** simplifica el proceso centrándose exclusivamente en el bienestar animal y en facilitar la conexión entre personas con intereses compatibles: quienes necesitan ayuda para cuidar de sus mascotas, y quienes desean ofrecer ese servicio.

Además, DogGo! apuesta por una estructura más guiada, en la que se diferencian claramente los perfiles de usuario, se gestionan las solicitudes de forma directa desde la propia aplicación, y se prioriza una experiencia de uso enfocada en la confianza, la sencillez y la funcionalidad.

## 2.2. Tecnologías

### **Backend:**

Se ha utilizado **Flask**, de Python, para la construcción de la **API REST** que gestiona la lógica de negocio, el enrutamiento y la conexión con la base de datos. Ideal para aplicaciones web de tamaño medio como DogGo!.

Para el almacenamiento de datos, se ha optado por **MongoDB**, una base de datos **NoSQL** orientada a documentos, que permite una estructura dinámica y adaptable, especialmente útil para gestionar colecciones como usuarios, perros y anuncios.

### **Frontend:**

La interfaz de usuario está desarrollada con **Vue.js** con JavaScript, que facilita la creación de interfaces interactivas y reactivas. Para la gestión del estado de la aplicación, se ha integrado **Pinia**, una alternativa moderna a Vuex, que permite mantener de forma centralizada y eficiente la información compartida entre componentes.

La comunicación con el backend se realiza mediante **Axios**, una librería ligera que facilita el envío de solicitudes HTTP, especialmente útil para la integración con la API REST desarrollada.

**Autenticación:**

La gestión de sesiones y seguridad se ha implementado mediante **JWT (JSON Web Tokens)**. Esta tecnología permite validar la identidad de los usuarios de forma segura y sin necesidad de almacenar sesiones en el servidor, lo que mejora el rendimiento y escalabilidad del sistema.

**Diseño y prototipado:**

Para el diseño visual de la plataforma y la elaboración de prototipos previos al desarrollo, se ha utilizado **Figma**, una herramienta de diseño colaborativo basada en la nube. Gracias a ella, ha sido posible definir la estructura visual de las pantallas, los flujos de navegación y la experiencia de usuario antes de pasar a la implementación final.

---



# Capítulo 3

## Requisitos, especificaciones, coste, riesgos, viabilidad

### 3.1. Requisitos

### 3.2. Especificaciones

### 3.3. Costes

### 3.4. Riesgos

### 3.5. Viabilidad

### 3. Requisitos, especificaciones, coste, riesgos, viabilidad

#### 3.1 Requisitos

##### **Sistema de registro e inicio de sesión:**

La plataforma debe permitir que los usuarios, tanto **dueños** como **paseadores**, puedan **crear una cuenta** de forma segura e iniciar sesión con sus credenciales. El sistema debe garantizar la **protección de los datos personales** y una correcta **gestión de sesiones** mediante autenticación segura.

##### **Gestión de perros:**

Los usuarios con rol de **dueño** deben poder **registrar perfiles de sus perros**, incluyendo información como el nombre, la raza y la edad. Además, deben poder **editar** o **eliminar** estos perfiles en cualquier momento, de forma sencilla e intuitiva.

##### **Gestión de anuncios:**

Los usuarios con rol de **paseador** deben tener la posibilidad de **publicar un anuncio** en el que se presenten, indiquen su ubicación y una breve biografía. Los **dueños**, por su parte, deben poder **consultar** estos anuncios para encontrar un paseador adecuado, siguiendo un modelo similar al de plataformas como Wallapop.

##### **Sistema de solicitud de paseos:**

La aplicación debe permitir que los **dueños puedan solicitar un paseo** a partir de un anuncio. El **paseador correspondiente podrá aceptar o rechazar** dicha solicitud desde su panel de usuario. Esta funcionalidad es clave para facilitar la coordinación entre ambas partes.

### **Interfaz de usuario accesible y amigable:**

El diseño de la plataforma debe ser **intuitivo y fácil de usar**, incluso para usuarios sin conocimientos técnicos avanzados. La navegación debe ser clara, con menús y botones accesibles desde cualquier dispositivo.

## **3.2 Especificaciones**

### **Backend:**

El servidor está implementado con Flask. El sistema se organiza en distintos módulos o blueprints, cada uno de ellos asociado a una funcionalidad específica:

#### **/auth**

- POST /auth/register: Registro de nuevos usuarios (owner o walker) con cifrado de contraseñas mediante bcrypt.
- POST /auth/login: Autenticación de usuarios y generación de token JWT.
- POST /auth/logout: Cierre de sesión mediante invalidación de cookies.
- DELETE /auth/user: Eliminación de cuenta del usuario autenticado junto con sus perros o anuncio.
- PUT|POST /auth/update: Modificación de datos personales (nombre, apellidos, contraseña).
- POST /auth/change-password: Cambio seguro de contraseña previa verificación.

## **/dogs**

Exclusivo para usuarios con rol owner:

- POST /dogs: Registro de un perro con nombre, raza y edad.
- GET /dogs: Listado de todos los perros del dueño autenticado.
- GET /dogs/<dog\_id>: Consulta de un perro concreto.
- PUT /dogs/<dog\_id>: Edición de los datos del perro.
- DELETE /dogs/<dog\_id>: Eliminación del perro.

## **/advertisements**

Exclusivo para usuarios con rol walker:

- POST /advertisements/create: Creación de un único anuncio de paseo.
- GET /advertisements/walker: Consulta del anuncio propio.
- PUT /advertisements/update: Edición del anuncio.
- PATCH /advertisements/toggle-pause: Activación o pausa del anuncio.
- DELETE /advertisements/delete: Eliminación del anuncio.

Usuarios con rol owner pueden consultar:

- GET /advertisements/all: Listado de anuncios activos, con filtro opcional por localidad.

## **/requests**

Sistema que conecta a dueños y paseadores a través de solicitudes:

- POST /requests: Creación de solicitud de paseo a partir de un anuncio, incluyendo la fecha y los perros.
- GET /requests/owner: Visualización de solicitudes enviadas por el dueño.
- GET /requests/walker: Visualización de solicitudes recibidas por el paseador.
- PATCH /requests/<id>/accept: Aceptar solicitud (solo si está pendiente).
- PATCH /requests/<id>/reject: Rechazar solicitud.
- PATCH /requests/<id>/cancel: Cancelar solicitud (desde cualquiera de las partes).
- DELETE /requests/<id>: Eliminar una solicitud si ha sido cancelada o rechazada.

Todos los endpoints están protegidos mediante autenticación con JWT y validación del rol correspondiente.

## Base de datos:

La persistencia se gestiona mediante **MongoDB**, una base de datos NoSQL orientada a documentos. Las principales colecciones son:

### **users:**

Almacena los datos de los usuarios:

```
_id: ObjectId('6838b9cc95a5f219e2e25e4e')
name : "Pepito"
last_name : "Grillo"
email : "test@test.com"
password : "$2b$04$LNnHMGHHdZPAuMo2mbQuY0kdgaZ94FY5vnmufvnDEZtSfjAKIfyRq"
role : "owner"
```

### **dogs:**

Asociados a usuarios con rol **owner** mediante **owner\_id**:

```
_id: ObjectId('682b5e293f58d0e527cc29e3')
name : "willy"
breed : "Boxer"
age : 10
owner_id : ObjectId('682877aed7037e3b50ce01b5')
```

### **advertisements:**

Vinculados a paseadores por **walker\_id**.

```
_id: ObjectId('682cb94cbc2f90f4b245e4cc')
biography : "Holaaa, soy Paco y me encanta pasear perritos!!"
maxDogs : 5
locality : "Valencia"
walker_id : ObjectId('682b667c3f58d0e527cc29e8')
paused : false
```

## requests:

Incluyen relaciones cruzadas entre dueños, paseadores, perros y anuncios.

```
_id: ObjectId('683a305b0df70ac6055cd028')
owner_id: ObjectId('682877aed7037e3b50ce01b5')
walker_id: ObjectId('682b667c3f58d0e527cc29e8')
ad_id: ObjectId('682cb94cbc2f90f4b245e4cc')
date: "2025-06-04"
▸ dogs: Array (2)
▸ dogs_info: Array (2)
▸ owner_info: Object
▸ walker_info: Object
status: "rechazada"
created_at: 2025-05-30T22:25:31.253+00:00
```

## Frontend:

Ha sido desarrollado utilizando **Vue.js 3** con la ayuda de la herramienta **Pinia** para la gestión del estado, **Vue Router** para la navegación, y **Axios** (a través de un wrapper en `api.js`) para la comunicación con el backend.

El proyecto se estructura de la siguiente manera:

/src	
├── api/	-Lógica para llamadas HTTP (axios wrapper)
├── assets/	-Recursos estáticos como imágenes
├── components/	-Componentes Vue reutilizables
├── data/	-Archivos estáticos como JSON con provincias
├── router/	-Configuración de Vue Router
├── stores/	-Gestión de estado con Pinia
├── views/	-Vistas principales del sistema (páginas)
└── App.vue	-Componente raíz

## Gestión de rutas (router/index.js)

### Rutas públicas:

/: Página principal (IndexView)

/register: Registro de usuario

/login: Inicio de sesión

### Rutas privadas:

/owner-profile: Perfil del dueño de perro

/walker-profile: Perfil del paseador de perros

/active-walks: Vista de anuncios activos para dueños

### Protección de rutas:

- Uso de meta.requiresAuth para restringir el acceso
- Validación del rol (owner o walker)

### Gestión de estado con Pinia

- **auth.js**: Maneja el estado de autenticación, usuario actual, perros registrados, formularios de gestión de perros y acciones comunes como login, logout, registro, etc.
- **dog.js**: Manejo del listado de razas (por ahora solo breeds)
- **walkerAd.js**: Gestión del anuncio del paseador (crear, actualizar, pausar y eliminar)

### Principales vistas implementadas

#### **RegisterView.vue**

- Formulario de registro con validación básica.
- Registro con campos: nombre, apellido, email, contraseña, rol (owner o walker).
- Uso de Pinia (authStore) para realizar la acción registerUser.



### **LoginView.vue**

- Formulario de login.
- Uso de `authStore.loginUser()` para autenticar y redirigir según el rol.
- Mensaje de error en caso de fallo.

### **OwnerProfileView.vue**

- Panel de control del dueño de perro.
- Gestión de datos personales.
- Añadir, editar o eliminar perros.
- Mostrar perros registrados.

### **ActiveWalksView.vue**

- Filtrado por provincia de anuncios activos publicados por paseadores.
- Muestra datos del paseador y permite enviar una solicitud de paseo.
- Verificación de perros registrados antes de poder solicitar.

### **Estilos y experiencia de usuario**

- Estilos implementados con CSS scoped en cada componente.
- Tema oscuro predominante en vistas públicas (login, register).
- Botones y formularios accesibles y adaptativos.

### **Comunicación con backend**

- Las llamadas al backend se hacen mediante funciones importadas desde `/api/api.js`.
- Se utilizan funciones `authPost`, `dogsPost`, `dogsGet`, `adsGet`, `adsPost`, `requestsPost`, etc.

### **Persistencia del usuario**

- Se guarda `accessToken`, `userRole` y `user` en `localStorage`.
- En cada carga, `authStore.initializeAuth()` comprueba y restablece el estado.

## **Seguridad:**

### **Seguridad en el backend**

- **Autenticación mediante JWT (JSON Web Tokens):** Todos los usuarios reciben un token seguro tras iniciar sesión. Este token debe incluirse en cada solicitud protegida para verificar la identidad del usuario.
- **Middleware de protección de rutas:** Solo los usuarios autenticados pueden acceder a las rutas protegidas, y en algunas rutas se aplica además un control de rol (owner o walker).
- **Cifrado de contraseñas:** Las contraseñas de los usuarios se almacenan en la base de datos de forma segura usando Flask-Bcrypt, que aplica un hash con sal.
- **Validación de entradas:** Se valida la estructura de los datos recibidos en los endpoints (register, login, update, etc.) para evitar inyecciones y datos maliciosos.
- **Control de errores:** Se manejan de forma controlada los errores del servidor, evitando exponer información sensible en los mensajes de respuesta.

### **Seguridad en la base de datos**

- **Separación de roles:** Las operaciones disponibles para el usuario dependen de su rol (dueño o paseador), lo que limita el acceso a información sensible.
- **Restricciones en las consultas:** Las rutas de obtención de perros, anuncios o solicitudes filtran los datos según el usuario autenticado.

### **Seguridad en el frontend**

- **Control de rutas por rol y autenticación:** Se usa un router.beforeEach que impide que un usuario sin token acceda a rutas protegidas, y redirige según el rol (owner-profile, walker-profile, etc.).

- **Gestión segura del token:** El token JWT se guarda en localStorage y se elimina completamente al cerrar sesión, junto con el rol y los datos del usuario.
- **Ocultación de funcionalidades no autorizadas:** En el interfaz se ocultan botones, formularios o acciones si el usuario no tiene permiso o si no ha iniciado sesión.

### 3.3 Costes

El desarrollo completo ha requerido aproximadamente 80 horas de trabajo, distribuidas en 2 horas diarias entre semana, 6 horas los fines de semana. Lo cual hace una duración total de 4 semanas.

#### Coste en herramientas

Todas las tecnologías utilizadas son de libre acceso:

**Backend:** Flask (framework gratuito de Python).

**Frontend:** Vue.js (framework JavaScript gratuito).

**Base de datos:** MongoDB Community Edition (gratuita).

**Entorno de desarrollo:** Visual Studio Code, mongosh y otras utilidades de código abierto.

**Hardware:** Ordenador personal propio.

#### Costes previstos de despliegue

**Servidor:** Un plan de pago en Vercel (para mejorar tiempos de respuesta y capacidad) rondaría los 20 € mensuales.

**Dominio web:** El registro de un dominio personalizado podría costar entre 5 € y 20 € anuales, según la extensión elegida (.com, .es, etc.).

### 3.4 Riesgos

- **Técnicos:** El chat no muestra mensajes (problema actual). Puede haber fallos en el polling o en MongoDB si hay muchos usuarios.
- **Adopción:** Que los dueños o paseadores no usen la app porque prefieren Wallapop o no confían.
- **Tiempo:** No completar el chat o las reservas antes de la entrega del TFG.
- **Seguridad:** Algún fallo en JWT o MongoDB que exponga datos.

### 3.5 Viabilidad

#### Viabilidad económica

El desarrollo se ha realizado íntegramente con herramientas de código abierto y recursos personales, sin incurrir en costes económicos directos. A medio plazo, el coste de mantener la aplicación desplegada (incluyendo un plan básico de servidor y un dominio web) sería reducido y asumible, lo que respalda la viabilidad económica del proyecto.

#### Viabilidad social

El mercado de servicios relacionados con el cuidado y paseo de perros, ha experimentado un crecimiento sostenido. Plataformas como Rover han validado el modelo de negocio. DogGo! se presenta como una solución útil tanto para dueños con poco tiempo como para paseadores que buscan oportunidades laborales.

---

# Capítulo 4

## Análisis

### 4. Análisis

DogGo! surge como respuesta a una necesidad concreta: muchos propietarios de perros no disponen del tiempo necesario para sacarlos a pasear con regularidad, y, por otro lado, existen paseadores interesados en ofrecer este servicio de forma sencilla y confiable. Este capítulo expone el problema detectado, el análisis de soluciones existentes en el mercado, las decisiones tomadas en las primeras fases del proyecto y el modelo de datos definido para dar soporte a las funcionalidades clave.

#### Definición del problema

El problema principal se centra en la dificultad que tienen muchos dueños de perros para encontrar paseadores de confianza, especialmente debido a horarios laborales exigentes o falta de tiempo. En paralelo, los paseadores utilizan actualmente métodos poco eficientes para ofrecer sus servicios, como redes sociales, aplicaciones genéricas o anuncios impresos, lo que dificulta la conexión con potenciales clientes.

DogGo! propone una solución digital específica: una plataforma web que permita a los dueños contactar con paseadores de manera rápida, segura y directa, organizando paseos de forma clara y gestionando el proceso desde un solo lugar.

### Análisis de soluciones existentes

Para definir el enfoque de DogGo!, se analizaron diversas soluciones existentes:

Wallapop y Milanuncios: Aunque permiten publicar y buscar anuncios, su estructura no está diseñada para servicios concretos como los paseos de perros. La falta de categorías específicas, filtros adecuados o perfiles orientados a servicios, limita su utilidad en este contexto.

Rover: Esta plataforma está enfocada en el cuidado de perros y ofrece una solución completa. Sin embargo, su uso está más extendido en países como Estados Unidos, y cobra comisiones elevadas por cada reserva. Esto evidenció la necesidad de una alternativa más local, sencilla y sin intermediarios.

### Decisiones iniciales

Tras el estudio del problema y del mercado, tomé las siguientes decisiones:

- Desarrollar una aplicación web, ya que es accesible desde cualquier dispositivo con navegador, requiere menos tiempo de desarrollo inicial que una app móvil y permite iterar rápidamente.
- Definir funcionalidades clave desde el inicio, como el registro y autenticación de usuarios, gestión de perros, publicación de anuncios, y un sistema de solicitudes.

Seleccioné tecnologías probadas y eficientes:

- Flask como framework de backend, por su ligereza y facilidad de integración con MongoDB.
- MongoDB como base de datos, por su flexibilidad para trabajar con documentos.

- Vue.js con Pinia para el frontend, por su rapidez de desarrollo y la capacidad de construir interfaces reactivas.

### Modelo de datos

Las colecciones definidas en MongoDB son:

- Usuarios (users):

{user\_id, email, password, role} (role puede ser 'owner' o 'walker').

- Perros (dogs):

{dog\_id, owner\_id, name, breed, description}.

- Anuncios (advertisements):

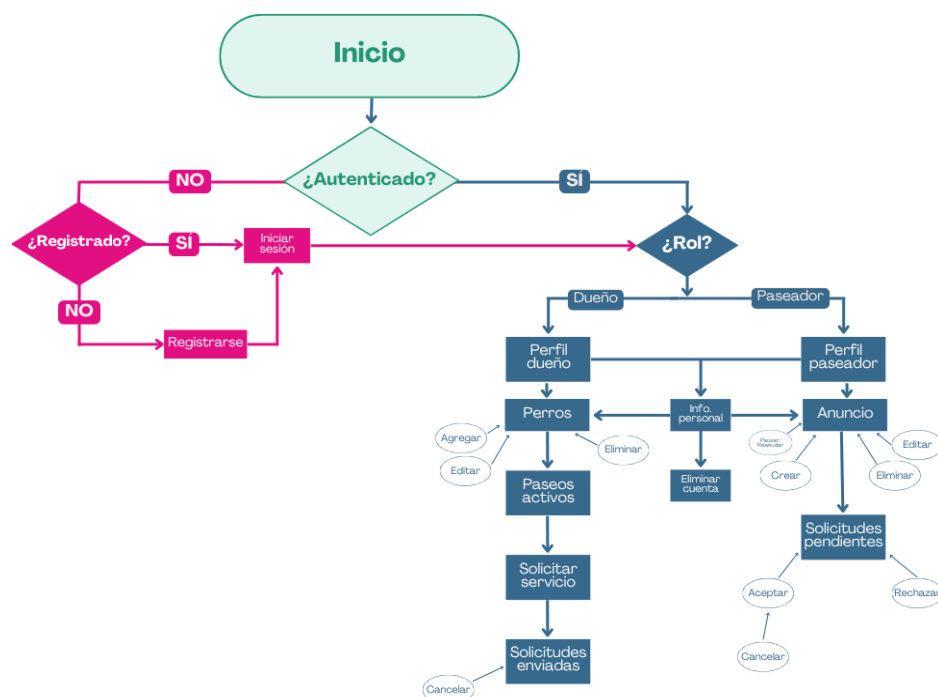
{ad\_id, walker\_id, title, description, price}.

- Solicitudes (requests):

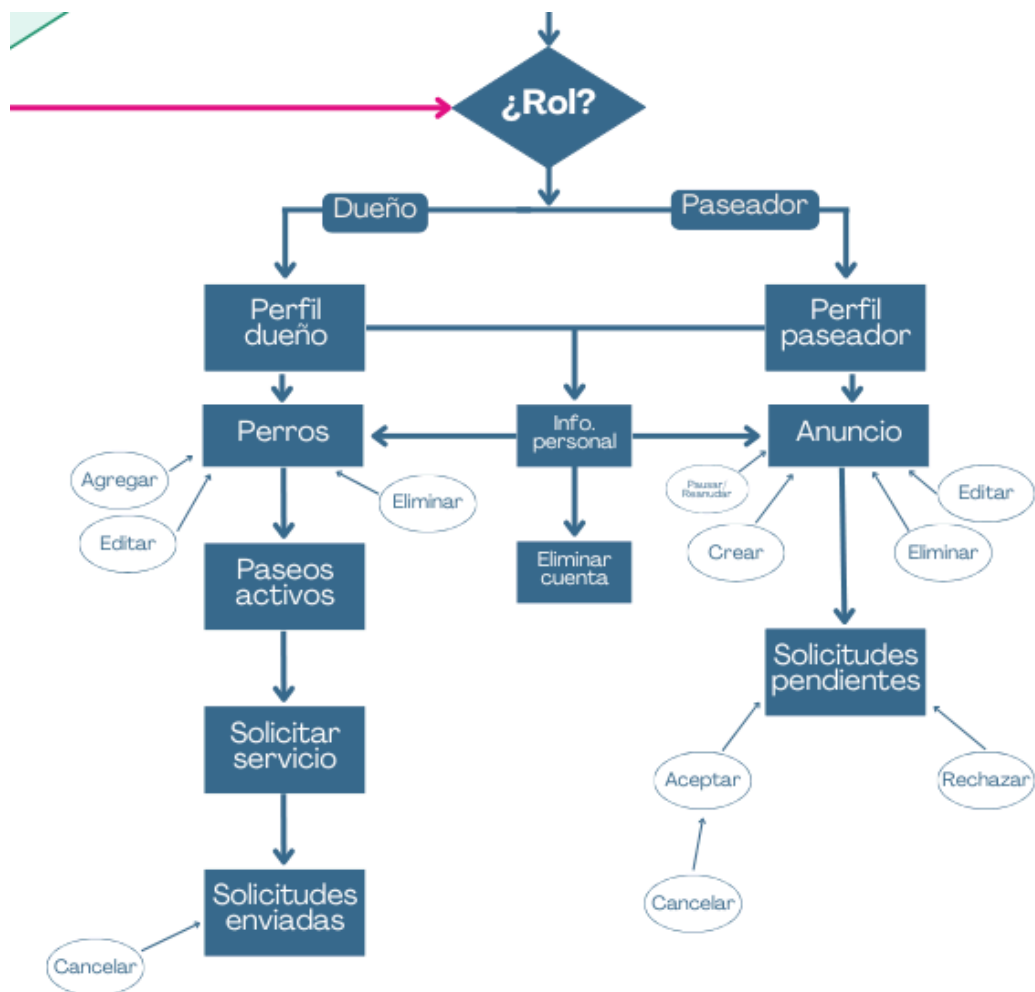
{owner\_id, walker\_id, ad\_id, date, dogs, dogs\_info, owner\_info, walker\_info, status, create\_at}.

### Diagrama de flujo de uso

A continuación se presenta un diagrama de flujo detallado que refleja el comportamiento esperado de los dos perfiles principales de usuario en DogGo! (dueño y paseador).



Como no se distingue muy bien, lo pongo por partes con más zoom.






# Capítulo 5

## Diseño

### 5. Diseño


#### - Registro:

[← Volver](#)



# DogGo!

Para ti, por ellos.



## Regístrate

Registrarse

[¿Ya tienes cuenta? Iniciar sesión](#)


Selecciona tu rol

Dueño

Paseador

Al registrarse, aceptas nuestros [términos y condiciones](#)

Register-Mobile



← Volver

## Regístrate

Registrarse

[¿Ya tienes cuenta? Iniciar sesión](#)

Selecciona tu rol

Dueño

Paseador

Al registrarse, aceptas nuestros [términos y condiciones](#)

#### - Inicio de sesión:


[← Volver](#)

## Iniciar sesión

[¿Has olvidado la contraseña?](#)


Entrar

[¿No tienes cuenta? Registrarse](#)




# DogGo!

Para ti, por ellos.



Login-Mobile



← Volver

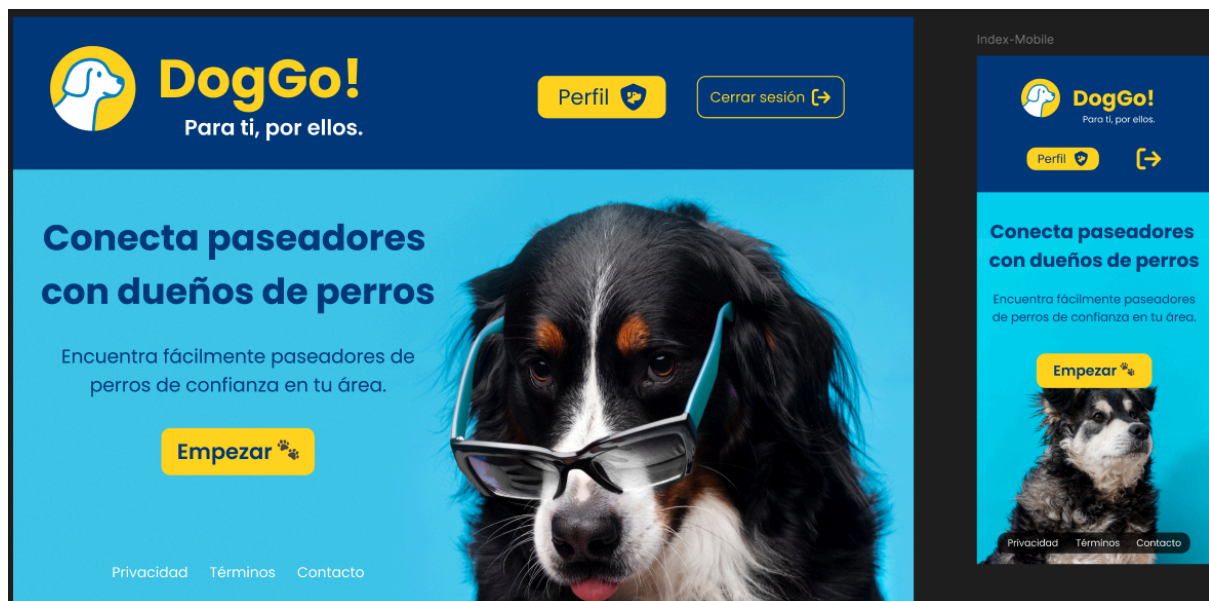
## Iniciar sesión

[¿Has olvidado la contraseña?](#)

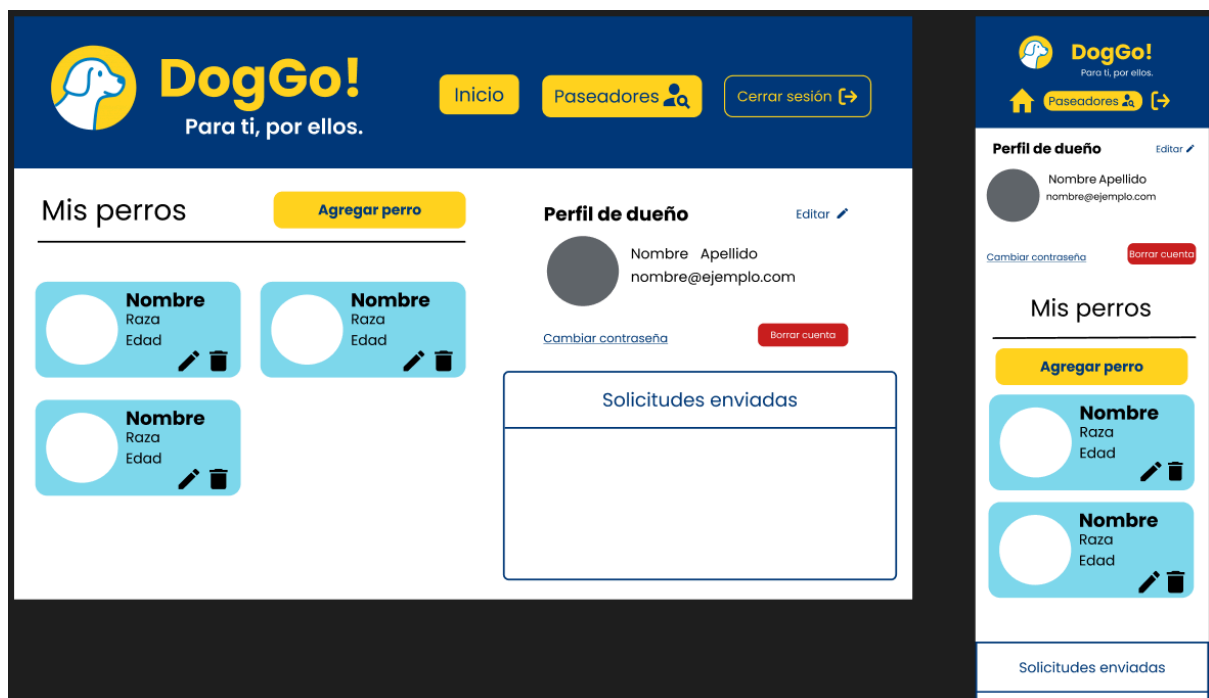
Entrar

[¿No tienes cuenta? Registrarse](#)

- Inicio:



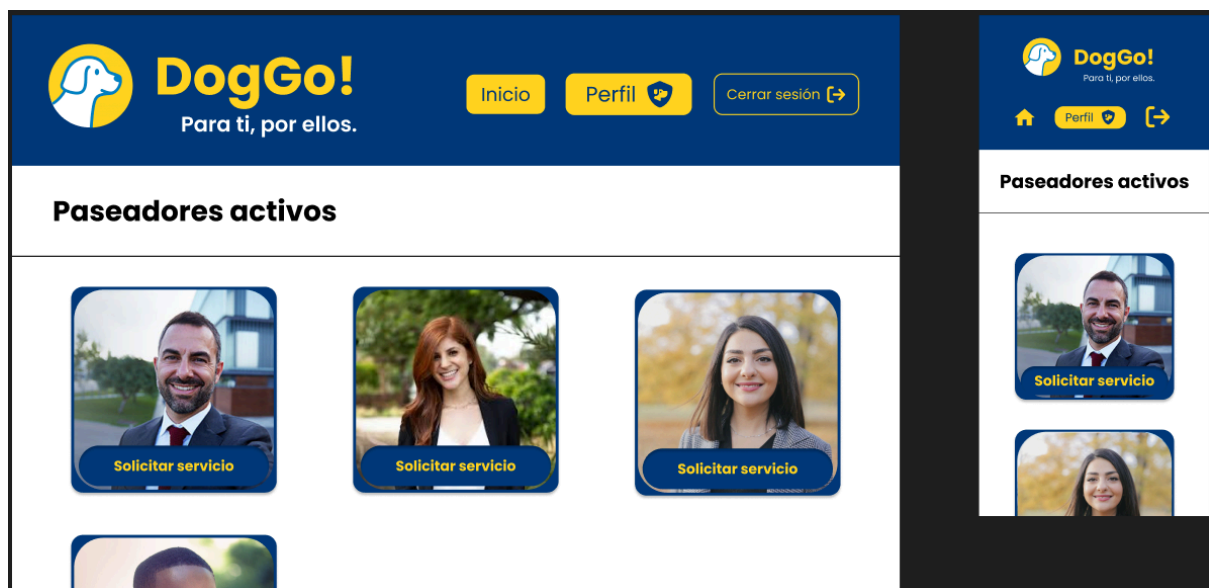
- Perfil de dueño:



- Perfil de paseador:



- Paseadores activos (solo accesible desde el rol de dueño):



# Capítulo 6

## Implementación y pruebas

### 6.1. Implementación

### 6.2. Pruebas funcionales

### 6.3. Pruebas de rendimiento

### 6.4. Pruebas de usabilidad

#### 6.1 Implementación

El desarrollo de DogGo! se ha realizado dividiendo el proyecto en frontend, backend y base de datos.

- El backend se ha implementado en Flask, estructurado en módulos REST para autenticación, gestión de perros, anuncios y solicitudes.
- El frontend se ha desarrollado en Vue.js, utilizando Pinia como gestor de estado y Vue Router para la navegación.
- Se ha usado MongoDB como base de datos, gestionada mediante Flask-PyMongo.
- El flujo básico incluye registro/login, gestión de perfil, anuncios visibles para dueños y solicitud de paseos.

#### 6.2 Pruebas funcionales

Se realizaron pruebas manuales para validar los flujos principales:

- Registro y login de usuarios (dueños y paseadores)
- Creación y edición de perros
- Publicación y visualización de anuncios
- Envío de solicitudes desde dueños hacia paseadores
- Acceso restringido según rol (comprobado en rutas protegidas del frontend)

### 6.3 Pruebas de rendimiento

Las pruebas se centraron en el tiempo de respuesta del backend y la carga de anuncios en el frontend.

- Las peticiones GET/POST mostraron tiempos de respuesta inferiores a 500ms en local.
- La carga de anuncios y navegación entre vistas fueron fluidas sin retrasos apreciables.
- No se observaron cuellos de botella ni errores bajo uso normal con varios anuncios y usuarios.

### 6.4 Pruebas de usabilidad

Se pidió a 3 personas externas que probaran la plataforma como usuarios dueños y paseadores.

- Comentaron que la interfaz era clara y fácil de usar.
  - El formulario de anuncios y el proceso de solicitud resultaron intuitivos.
  - Se detectaron pequeños fallos (como errores no mostrados al fallar el login).
-

# Capítulo 7

## Conclusiones

### 7. Conclusiones

DogGo! demuestra ser una solución viable para conectar dueños de perros con paseadores de forma sencilla y eficaz. Durante el desarrollo del proyecto, se han cumplido los objetivos principales:

- Se ha construido una aplicación web funcional con backend en Flask y frontend en Vue.js.
- Se ha implementado una base de datos MongoDB adaptada al modelo necesario para gestionar usuarios, perros, anuncios y solicitudes.
- La interfaz es intuitiva y accesible desde cualquier dispositivo, lo que mejora su usabilidad.

Además, el proceso ha permitido adquirir experiencia práctica en desarrollo web full-stack, gestión de usuarios con autenticación JWT, diseño de APIs REST y separación por roles. También se han abordado aspectos importantes como la seguridad, el rendimiento y la experiencia de usuario.

Como mejora futura, se contempla implementar WebSockets para habilitar el chat en tiempo real, finalizar la funcionalidad de reservas y desplegar la aplicación en un entorno de producción.

DogGo! sienta una base sólida para evolucionar hacia una plataforma real, útil y con potencial de crecimiento en el ámbito local.

---

# Apéndice A

## Apéndice

- El código completo del proyecto DogGo! está en el siguiente repositorio:

<https://github.com/Raulnavase/TFG-DogGo.git>

- El diseño visual de la aplicación fue realizado en Figma. Puede consultarse en el siguiente enlace:

<https://www.figma.com/design/aw3NvnWE5iYXJoKsJ5X6XI/TFG---DogGo-?node-id=0-1&t=pjkeXlqLxdmn3v7W-1>

# Bibliografía

Documentación de Flask - <https://flask.palletsprojects.com/>

Documentación de Vue.js - <https://vuejs.org/>

Documentación de MongoDB - <https://www.mongodb.com/docs/>

Documentación de Pinia - <https://pinia.vuejs.org/>

Documentación de Axios - <https://axios-http.com/docs/intro>

Documentación de JWT - <https://jwt.io/introduction>

Stack Overflow - <https://stackoverflow.com/>

Manz.dev - <https://manz.dev/>

uiverse.io - <https://uiverse.io/>

W3Schools - <https://www.w3schools.com/>

Grok, xAI

GitHub Copilot - <https://copilot.github.com/>

---