Trabajo de Fin de Grado

Aplicación web para la gestión de objetos perdidos

TITULACIÓN: Grado en Ingeniería Informática

AUTOR: Álvaro Lettieri Acosta

TUTORIZADO POR:
Francisco Alexis Quesada Arencibia

Fecha [6/2025]

ULPGC Universidad de Las Palmas de Gran Canaria | eii ESCUELA DE INGENIERÍA INFORMÁTICA

TFT04

SOLICITUD DE DEFENSA DE TRABAJO DE FIN DE TÍTULO

D./Dª Álvaro Lettieri Acosta, autor del trabajo Aplicación web para la gestión de objetos perdidos, correspondiente a la titulación Grado en Ingeniería Informática.

SOLICITA

que se inicie el procedimiento de defensa del mismo, para lo que se adjunta la documentación requerida, haciendo constar que

[x] se autoriza / [ ] no se autoriza la grabación en audio de la exposición y turno de preguntas.

Asimismo, con respecto al registro de la propiedad intelectual/industrial del TFT, declara que:

[ ] Se ha iniciado o hay intención de iniciarlo (defensa no pública).

[x] No está previsto.

Y para que así conste firma la presente. (fecha en firma electrónica)

El/La estudiante

Fdo.————————

A rellenar y firmar **obligatoriamente** por el/la/los/las tutores

En relación con la presente solicitud, se informa:(firmar donde corresponda)

Positivamente (en caso de detección de copia, esta firma quedará invalidada)	Negativamente (justificación en TFT05)
Fdo.————————	Fdo.————————

DIRECTOR DE LA ESCUELA DE INGENIERÍA INFORMÁTICA

Agradecimientos

Ante todo, a mis padres y a mí hermana, quienes han confiado en mi en cada uno de los dificultosos pasos de esta aventura.

A mi maravillosa gata, Wok, quien me ha ayudado a apreciar las pequeñas cosas de la vida.

A mi hermano del alma, Jota, quien me ha inspirado a mostrar lo mejor de mí al mundo.

A mis amigos de la península, que cada día me recuerdan lo pequeño que somos, y lo mucho que logramos.

A mis amigos de Teror, que han estado siempre ahí para ayudarme a despejar mi mente.

A mi tutor Alexis, quien me ha ayudado a dar este último gran paso.

Y finalmente, a mi mismo, por nunca rendirme ante la adversidad.

Resumen

Aplicación web escalable que cubre, facilita y mejora los procesos de búsqueda y entrega de objetos perdidos, ofreciendo una herramienta a los usuarios que permita una mayor precisión a la hora de recolectar datos sobre los objetos extraviados y facilite el proceso de comunicación entre oficinas de objetos perdidos y los individuos involucrados.

Esta aplicación usa tecnología de geolocalización en tiempo real y ofrece una interfaz simple y atractiva en la que se pueden observar y filtrar de forma minuciosa los objetos perdidos que otros usuarios hayan publicado. A su vez, los dueños de los objetos perdidos pueden ofrecer recompensas de búsqueda para incentivar a otros a cooperar.

Todo esto con el objetivo de construir una comunidad de usuarios que activamente preste su ayuda para la búsqueda de objetos perdidos.

Abstract

A scalable web application that covers, facilitates, and improves the processes of searching for and returning lost items, offering users a tool that allows for greater accuracy when collecting data about lost objects and streamlines communication between lost and found offices and the individuals involved.

This application uses real-time geolocation technology and provides a simple and attractive interface where users can view and filter lost items that others have posted in a detailed manner. Additionally, the owners of lost items can offer search rewards to encourage others to cooperate.

All of this aims to build a community of users who actively help in the search for lost belongings.

Índice general

Índice de figuras

Índice de cuadros

Índice de códigos

Capítulo 1

Introducción

En este capítulo se expondrá una ligera introducción a lo desarrollado, desde la motivación principal hasta los objetivos fundamentales del trabajo.

1.1. Motivación

Hay una gran cantidad de objetos que se extravían en Las Palmas de Gran Canaria, y más aún en el país entero, por lo que es insólito escuchar que muchas de las empresas, municipios y organizaciones que nos rodean no posean una plataforma para administrar dichos objetos extraviados, así como para facilitar el proceso de notificación/comunicación con los usuarios.

Tan solo en este mes de abril de 2025 más de 6700 objetos perdidos llegaron a la Oficina Municipal de Madrid [1], un surtido de objetos que en muchos casos no son recuperados por sus dueños y se pierden entre las estanterías de la oficina.

Por un lado, y en líneas generales, cuando un individuo descubre un objeto perdido y lo entrega en la oficina correspondiente de la organización en cuestión, se le exige que diga cuándo y dónde lo ha encontrado, su identificación y alguna manera para comunicarse con él. A continuación, el objeto es fotografiado y almacenado. Por otro lado, cuando un individuo se presenta en la oficina asegurando haber perdido algo, se le pide que identifique el objeto, cuándo y dónde lo ha perdido y que muestre una fotografía, si es que posee alguna. En el caso de que el objeto no se halle en la oficina, se le pide su identificación y alguna manera para comunicarse con él.

Siguiendo estos principios, se desarrolla una aplicación web escalable que cubra, facilite y mejore estos procesos, ofreciendo una herramienta a los usuarios que permita una mayor precisión a la hora de recolectar datos sobre los objetos extraviados y facilite el proceso de comunicación entre oficinas de objetos perdidos y los individuos involucrados.

1.2. Objetivos

Se pretende desarrollar una aplicación web escalable para la gestión de objetos perdidos, que facilite la entrada y salida de estos y ofrezca herramientas a los usuarios externos que pudieran haber perdido o encontrado algo. En definitiva, se propone el desarrollo de un portal web que pueda ser usado por distintos organismos y/o les ayude de manera indirecta gracias a la cooperación de los usuarios.

Dicho portal web constará de un sistema de sesión de usuario, una base de datos donde almacenar la información asociada a los objetos perdidos y hará uso de geolocalización y filtros para permitir al usuario encontrar objetos perdidos en su área o buscar en la base de datos según sus preferencias.

Capítulo 2

Estado del arte

En este apartado se presenta las herramientas web similares a la aplicación desarrollada, así como la situación actual de la Policía Local de Las Palmas de Gran Canaria en cuanto a la gestión de objetos perdidos.

2.1. Herramientas web y móvil

Se analizarán las siguientes herramientas web similares a la propuesta:

- FindMyLost
- FoundSpot
- Missing Pets

2.1.1. FindMyLost

FindMyLost es una empresa formada por entre 2 y 10 personas, fundada en 2016 [2]. Se trata de una alternativa web utilizada tanto por personas de a pie como por empresas como Adif o AENA [3]. Se debe crear una cuenta de forma obligatoria para usar sus servicios.

En la sección de perfil, el usuario puede ver los objetos encontrados, así como acceder a una bandeja de entrada con alertas y mensajes relacionados con objetos perdidos o encontrados, como se observa en la Figura 2.1 y la Figura 2.2.

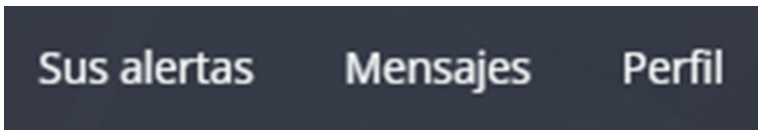

Ilustración 2.1: Sección de alertas y mensajes - FindMyLost

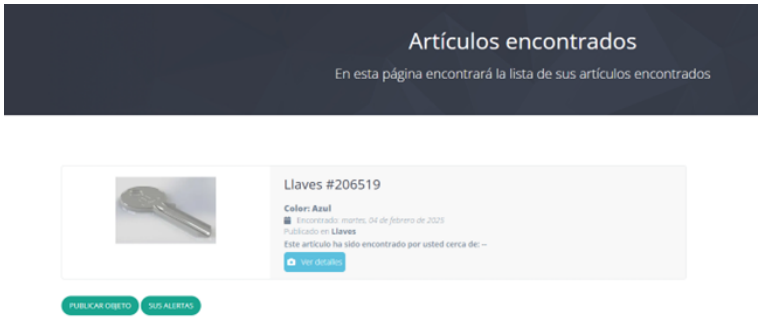

Ilustración 2.2: Lista de objetos encontrados - FindMyLost

En la Figura 2.3 se observa cómo el usuario puede seleccionar si ha perdido o encontrado algo.

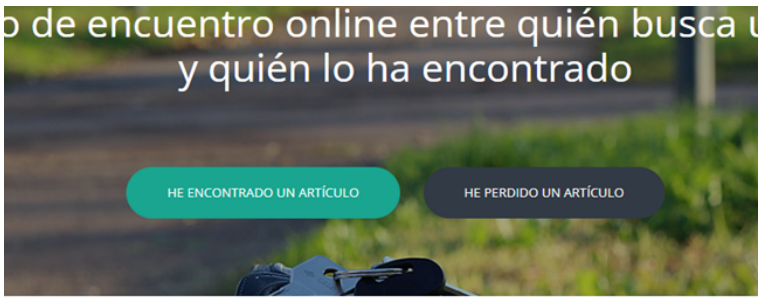

Ilustración 2.3: Selección de proceso - FindMyLost

Al hacer clic en el botón de la izquierda, se abre un formulario dividido en distintos pasos. La Figura 2.4 muestra los iconos que simbolizan cada uno de los pasos.

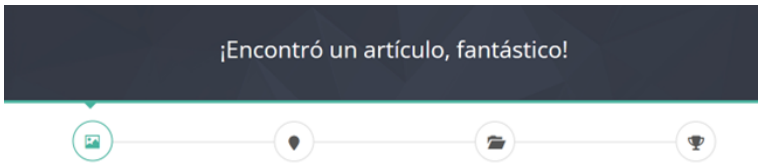

Ilustración 2.4: Proceso de objeto encontrado - FindMyLost

Este proceso se compone de 4 pasos:

1. Rellenar los datos básicos del objeto: subir una foto, asignar un nombre, seleccionar una categoría y establecer la fecha exacta en que se encontró.

2. Escribir la ubicación donde fue encontrado, lo que muestra dicha ubicación con un mapa de Google Maps, aunque no te permite interactuar con el marcador de ninguna manera.

3. Añadir información adicional: marca, color, método de entrega, dirección si se desea utilizar mensajería, y una descripción complementaria.

4. Elegir si se desea recibir una recompensa económica por parte de quien haya perdido el objeto.

Una vez completado el proceso, el objeto queda a la espera de la aprobación por parte del personal encargado.

En caso de que el usuario haya perdido un objeto, hará clic en el botón derecho mostrado en la Figura 2.3, lo que abre un formulario de un solo paso. En el que se deberá especificar qué se ha perdido, la marca, el color, cuándo y dónde, aunque en este caso no se hace uso de Google Maps ni de ninguna otra herramienta que facilite al usuario establecer la última ubicación conocida de manera más específica 2.5. Además, se muestra una serie de iconos que redirigen a los portales web de las distintas empresas clientes de FindMyLost, por si ha sido extraviado en lugares relacionados a dichos organismos.

Ilustración 2.5: Proceso de objeto perdido - FindMyLost

Al enviar el formulario, muestra si se ha encontrado o no el objeto y da la posibilidad de activar una alerta en caso de que no.

Respecto al panel de administración, FindMyLost indica lo siguiente:

> "Los operadores del Servicio de Atención al Cliente tienen acceso a un tablero detallado y a un Sistema de Gestión de Contenidos, a través del cual obtienen una visión más completa de todos los objetos encontrados (clasificados según ubicaciones y sub-ubicaciones) y de las interacciones de los Usuarios. Además, el cuadro de mandos muestra el historial de búsqueda global, las alertas activadas, los usuarios registrados, estadísticas detalladas sobre los objetos, las solicitudes de envío, los recibos de pago y los comentarios de los usuarios." - Página oficial de FindMyLost [3]

Los administradores también son capaces de registrar objetos perdidos en la propia plataforma centralizada y el portal web usa inteligencia artificial para la identificación de objetos de forma interna.

Finalmente, cabe destacar que tiene integrado un sistema de pago en línea y servicios de envío urgente.

2.1.2. FoundSpot

Otro ejemplo similar a FindMyLost. FoundSpot es una empresa formada por 2-10 personas fundada en 2016 [4]. Ha sido utilizada por clientes como GLOBAL, entre otros, y se puede considerar significantemente parecida a la anteriormente analizada [5].

Con solo algunas ligeras diferencias, sus funciones, procesos y aspectos básicos son casi idénticos a los de FindMyLost. Esta alternativa no posee un sistema de recompensas monetarias y no está claro si hacen uso de tecnología de reconocimiento de imágenes, aunque sí cuenta con la capacidad de no solo notificar sobre objetos perdidos/encontrados sino también mascotas y animales, como se puede ver en la Figura 2.6.

Ilustración 2.6: Proceso de objeto perdido - FoundSpot

No hay información pública detallada sobre el panel de administración ni existe una demo pública de la herramienta. No obstante, los procesos de notificación de pérdida y hallazgo de objetos son similares a los de FindMyLost. Con algunas diferencias, puesto que FoundSpot no tiene srevicio de mensajería ni de recompensas, sin embargo, si posee un sistema de geolocalización de usuario y un mapa de Google maps interactuable para elegir el lugar donde se ha perdido o encontrado un objeto, como se puede observar en la Figura 2.7.

Ilustración 2.7: Mapa interactuable - FoundSpot

Las diferencias más relevantes se resumen posteriormente en la Tabla comparativa final 2.1.

2.1.3. Missing Pets

Missing Pets es una aplicación móvil desarrollada por Revinade AB, que tiene como objetivo ofrecer una lista de mascotas perdidas o encontradas cerca de la ubicación del usuario [6][7]. A diferencia de las anteriores, no colabora con organismos públicos o privados, siendo completamente libre y comunitaria.

En la pantalla principal se puede ver un listado deslizable de animales perdidos, como puede verse en la Figura 2.8.

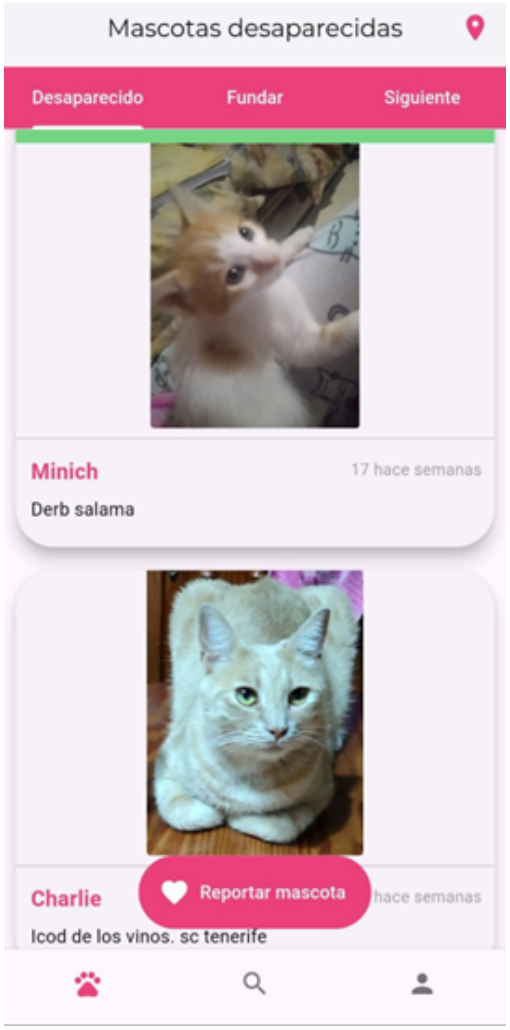

Ilustración 2.8: Menú de animales perdidos - Missing Pets

Existe una opción para indicar si se ha perdido o encontrado una mascota, visible también en la Figura 2.9.

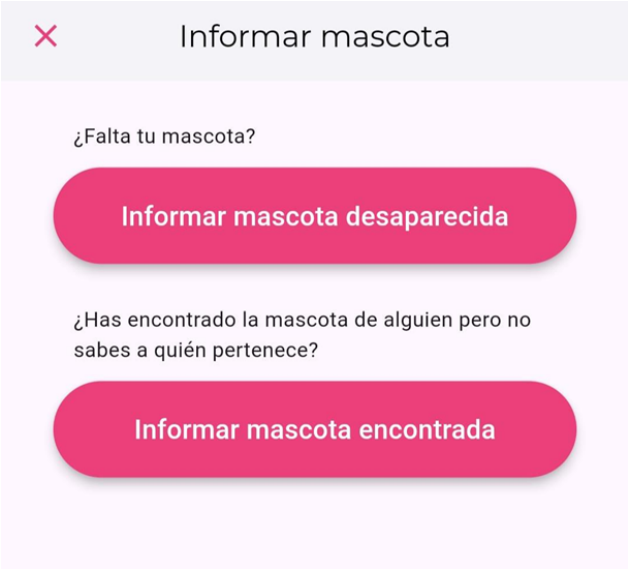

Ilustración 2.9: Elección de procesos - Missing Pets

Para subir una publicación, se deben completar los apartados mostrados en la Figura 2.10, que son similares a los formularios de las otras plataformas. No obstante, esta aplicación hace uso de tecnología de geolocalización para mostrar en la página principal de mascotas aquellas cuyo último paradero esté cerca de la posición del usuario. Permite a los usuarios usar tecnología de Google Maps para especificar el lugar donde la mascota última vez fue vista y dónde finalmente se encontró, en su caso.

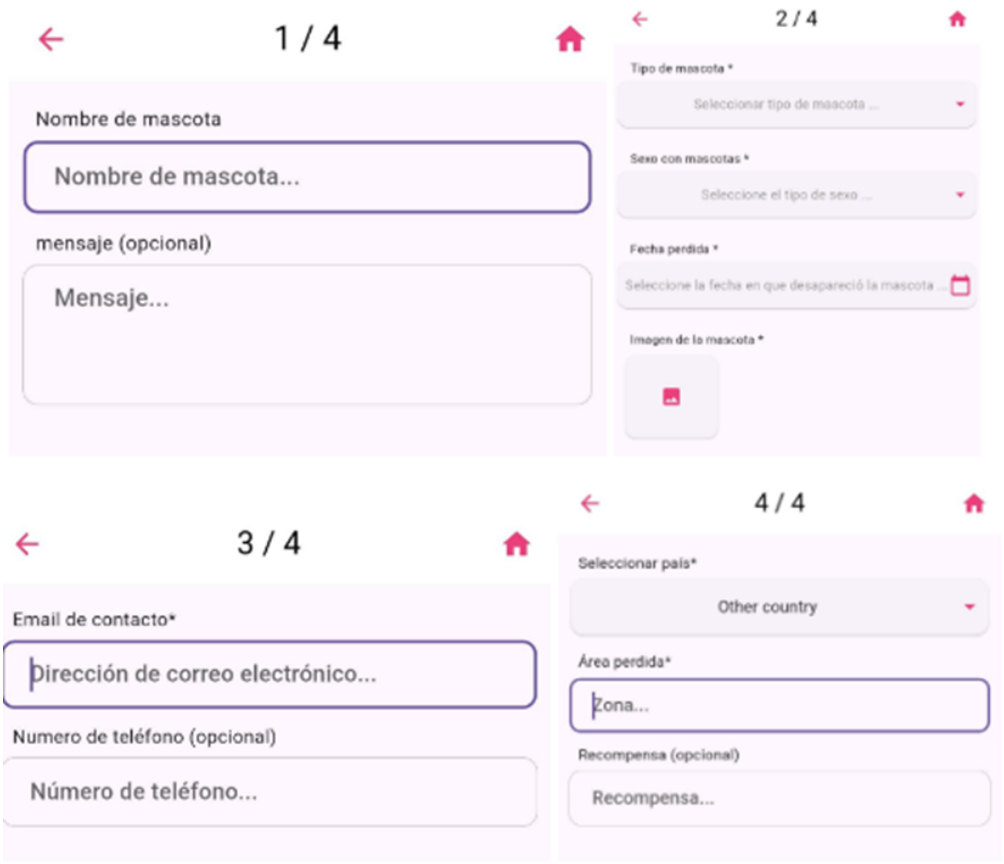

Ilustración 2.10: Pasos para subir publicación de animal perdido - Missing Pets

No se ha encontrado información sobre un panel de control de administrador en esta aplicación.

2.1.4. Comparativa

A continuación, se muestra una tabla comparativa 2.1 entre las características clave de las alternativas con las de la propuesta desarrollada.

Cuadro 2.1: Tabla comparativa entre herramientas web y móvil analizadas.

Características	**FoundSpot**	**FindMyLost**	**Missing Pets**	**Propuesta**
Seleccionar si ha perdido o encontrado algo.	✓	✓	✓	✓
Crear una cuenta es opcional.	✓	✗	✗	✓
Establecer lapso de fechas en el que se ha perdido/encontrado el objeto.	✗	✗	✗	✓
Servicio de entrega pagado por el destinatario.	✗	✓	✗	?
Sistema de recompensa monetaria.	✗	✓	✓	✓
Tecnología de reconocimiento de imágenes.	?	✓	?	?
Gestión de pérdida/encuentro de mascota.	✓	✗	✓	✓
Indicar si se ha perdido en medio de transporte.	✓	✗	✗	✓
Subida de varias fotos/videos al perder/encontrar algo	✓	✗	✗	✓
Intercambio de contactos por si el objeto ha sido dañado, o le falta algo	✗	✗	✓	✓
Lista con solicitudes de búsqueda de objetos perdidos	✗	✗	✓	✓
Mapa interactuable para especificar dónde se ha perdido/encontrado	✓	✗	✓	✓
Sistema de geolocalización para mostrar objetos perdidos cercanos	✗	✗	✓	✓
Filtros de objetos/mascotas	✗	✗	✗	✓

2.2. Policía Local de Las Palmas de Gran Canaria

La oficina de objetos perdidos de la Policía Local de Las Palmas de Gran Canaria está a cargo de una única persona, la cual organiza todas las entradas y salidas de objetos extraviados usando Excel, bases de datos y registros físicos. Existe un gran volumen mensual de objetos entrantes, que acaban almacenados en cajas fuertes, almacenes, estanterías e incluso armarios empotrados.

Hace unos años decidieron usar Facebook para conectar con las personas que pudieran haber perdido algo [8]. En la Figura 2.11 se muestra una de sus últimas publicaciones.

Ilustración 2.11: Post de Facebook Objetos Perdidos LPA

Sin embargo, dicha iniciativa fue finalizada en 2021, puesto que dificultó el trabajo de los

encargados. Hoy en día es posible contactar con la oficina en persona, mediante correo, teléfono o una solicitud en el portal web de la policía local. El formulario de la solicitud se puede observar en la Figura 2.12 [9].

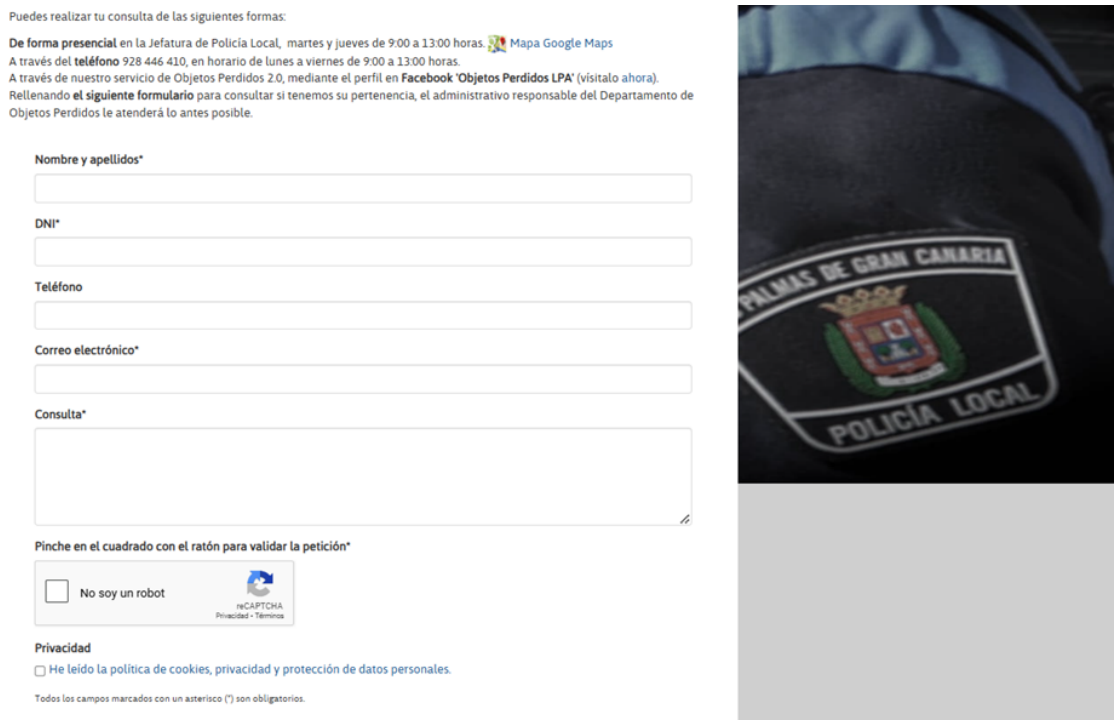

Ilustración 2.12: Contacto con Oficina de objetos perdidos de la Policía Local

2.3. Conclusión

Aunque es difícil que los objetos paren de ser perdidos, es posible ayudar para volver a encontrarlos. Es ideal, pues, fomentar una comunidad que ayude a encontrar y devolver lo que se extravía a diario. Las distintas iniciativas tecnológicas internacionales mostradas en este análisis son de por sí herramientas potentes por lo que para aportar e impulsar una propuesta innovadora en este campo se propone unificar y mejorar las características positivas clave de cada aplicación en la propuesta.

El resultado será un portal web de gestión de objetos perdidos en la que usuarios pueden realizar publicaciones de los objetos o mascotas que hayan perdido o encontrado. Los usuarios podrán navegar a través de publicaciones de objetos perdidos de otros usuarios y notificar, en su caso, si han encontrado lo que otros han perdido. En cuanto a las publicaciones de objetos encontrados, serán los administradores quienes se harán cargo, estableciendo coincidencias entre publicaciones de objetos perdidos y encontrados, y notificando a los usuarios implicados.

Capítulo 3

Metodología de trabajo y herramientas utilizadas

En este capítulo se habla sobre la metodología de trabajo aplicada, la planificación temporal establecida y las tecnologías y herramientas utilizadas en el desarrollo.

3.1. Metodología de trabajo

Se sigue una metodología iterativa/incremental, un desarrollo basado en cinco incrementos de similar longitud y trabajo. En cada incremento se prioriza la realización de las historias de usuario de mayor importancia y se itera sobre el anterior incremento para terminar lo incompleto o mejorar la implementación de lo terminado. Los incrementos inicialmente especificados han sido los siguientes:

1. Diseño de la arquitectura de la web.
2. Diseño e implementación de base de datos.
3. Diseño e implementación de vistas del usuario.
4. Diseño e implementación de vistas del administrador.
5. Diseño e implementación de sistema de notificación entre usuario y administrador.

Aunque inicialmente se ha seguido esta planificación, se ha modificado a lo largo del desarrollo del proyecto. La planificación final de los incrementos es la siguiente:

1. Diseño arquitectónico de la web.
2. Diseño e implementación de base de datos.
3. Diseño e implementación de vistas del usuario.
4. Diseño e implementación de sistema de notificación entre usuarios.

Como se puede observar, se ha omitido el incremento de las vistas de administrador y se ha modificado el último incremento para incluir notificaciones entre usuarios. Esto se debe, en primer lugar, a que la relevancia que tiene el panel de notificaciones y su uso es considerada mayor que el de las vistas de administrador, por lo que se ha priorizado su desarrollo, dejando el concepto de administrador para un futuro desarrollo. Esta decisión se ha tomado con el objetivo de priorizar la funcionalidad básica de la aplicación, que es la gestión de objetos perdidos y la comunicación entre usuarios.

En segundo lugar, tras un análisis de la aplicación y retroalimentación por parte de usuarios externos, se llega a la conclusión de que es más relevante y cómodo la existencia de notificaciones entre usuarios de forma directa, sin un administrador de por medio. Esto permite una comunicación más fluida y directa entre los usuarios, lo que puede mejorar la experiencia general de la aplicación y facilita la resolución de problemas.

3.2. Planificación temporal

Antes de comenzar el desarrollo del proyecto, y tras el análisis del estado del arte, se realizó la siguiente planificación temporal, ilustrada en los Cuadros 3.1 y 3.2. Como se puede observar, se ha realizado en base a los incrementos inicialmente especificados.

Cuadro 3.1: Planificación temporal

Nº	**Nombre**	**Horas**	**Incrementos**				
			1º	**2º**	**3º**	**4º**	**5º**
1T	Crear entorno de trabajo.	4	✗				
1	Barra lateral de navegación.	8	✗				
2	Portal de objetos perdidos/encontrados.	6	✗				
3	Tarjeta de objeto perdido/encontrado.	6	✗				
4	Detalles de objeto perdido/encontrado.	6	✗				
2T	Crear bases de datos Supabase.	4		✗			
3T	Implementar Pinia para conexión entre back y front.	8		✗			
5	Formulario Identificación de Usuarios.	4		✗			
6	Formulario de Registro de Usuarios.	8		✗			
7	Portal de perfil personal.	4		✗			
8	Portal de perfil ajeno.	4		✗			
9	Proceso de publicación de objeto olvidado.	16			✗		
10	Proceso de publicación de objeto encontrado.	4			✗		
11	Proceso de coincidencia entre objetos.	8			✗		
4T	Implementar API de Google Maps.	8			✗		
12	Filtro de objetos en las vistas de Usuario.	8			✗		
13	Crear rol de Administrador.	8				✗	
14	Proceso de Administrador de coincidencia entre objetos.	16				✗	
15	Filtro de objetos en las vistas de Administrador.	4				✗	
16	Notificaciones de Administrador.	16					✗
17	Notificaciones de Usuario.	4					✗
18	Portal de notificaciones.	16					✗

Cuadro 3.2: Estimación total de horas de trabajo.

Incremento	**Horas**	**Historias de usuario**
1º	30	1T - 1 - 2 - 3 - 4
2º	32	2T - 3T - 5 - 6 - 7 - 8
3º	52	9 - 10 - 11 - 4T - 12
4º	28	13 - 14 - 15
5º	36	16 - 17 - 18
TOTAL	170	

Tras la realización del proyecto se llega a la conclusión que, aunque la planificación de las características ha desarrollar ha sido en su mayor parte adecuada, ha habido desviaciones en el tiempo estimado y en las características desarrolladas. Como se explicó en el apartado anterior, se ha omitido las características relacionadas con el rol de administrador.

En lineas generales, la planificación temporal ha sido demasiado optimista, puesto que se ha tardado más de lo esperado en la implementación general de la aplicación. Siendo también este uno de los motivos por los que se ha priorizado la implementación de las notificaciones entre usuarios, antes que el rol de administrador.

La razón por la que esto aconteció es que, al hacer uso de gran cantidad de tecnologías y herramientas no utilizadas anteriormente por el autor de esta memoria, se ha tardado más de lo esperado en la comprensión y aprendizaje de las mismas. Por lo que, siguiendo las correctas prácticas de desarrollo de software, se ha priorizado el entendimiento de las herramientas y en su correcta implementación, así como el empleo de código limpio y técnicas para mantener y mejorar la escalabilidad.

Como dato más relevante no mencionado en cuanto a los cambios en la planificación de características, se ha utilizado la librería de `Leaflet` junto con la API de `Nominatim` para la implementación del mapa y las características asociadas, en lugar de la API de `Google Maps`.

Cabe destacar que, a su vez, se han añadido características no contempladas en la planificación inicial. Se ha desarrollado una página de inicio, una página con información sobre el proyecto, implementado mayor cantidad de funciones para el filtrado de objetos perdidos/encontrados, así como un componente para mostrar las publicaciones personales del usuario, entre otras. Las adiciones más relevantes se comentarán en el apartado de desarrollo del proyecto.

3.3. Tecnologías y herramientas utilizadas

- Framework de trabajo `Vue` (HTML, CSS, JavaScript, TypeScript) [10]. Una herramienta especializada para el desarrollo del apartado front-end de aplicaciones web. Conocida por sus múltiples compatibilidades con librerías de estilo y validación, así como su capacidad modular y su enfoque en la reactividad de los datos.

- En un inicio se planeó implementar la API de `Google Maps` [32], al final se hizo uso de la librería `LeafLet` [12]. Es una librería que se apoya en la API de `OpenStreetMap` [11], que permite la creación de mapas interactivos y personalizables, completamente compatibles con entornos móviles y de licencia creativa más flexible.

- API de `Nominatim` para geolocalización inversa [13]. Servicio apoyado en `OpenStreetMap` que permite la conversión de coordenadas geográficas a direcciones legibles.

- `Pinia` como gestor de estados [14]. Con una gran compatibilidad con `Vue`, permite gestionar el estado de la aplicación de manera ordenada y eficiente. Permite construir estados globales que pueden ser compartidos entre diferentes componentes de la aplicación.

- Base de datos `Supabase` [15]. Una plataforma que permite la creación de bases de datos, basada en `PostgreSQL` haciendo uso de una interfaz de usuario intuitiva y permitiendo el uso de comandos SQL para la gestión de tablas y datos. Se implementa con su librería `supabase-js`.

- `Figma`, con el objetivo de realizar bocetos básicos del desarrollo [16]. Es una herramienta de diseño destinada a la creación de prototipos y/o mockups de aplicaciones web y móviles, entre otras funcionalidades.

- `Git` y `GitHub` como control de versiones [17]. Plataforma de desarrollo que permite el almacenamiento y gestión de proyectos de software, lo que facilita el mantenimiento del código, la documentación del proyecto y aporta una copia de seguridad en la nube, accesible desde cualquier lugar.

- `Visual Studio Code` para el desarrollo del código [18]. Un editor de código ligero con una amplia gama de extensiones y características diseñadas para facilitar el desarrollo de aplicaciones de distinta índole.

- `NPM` como gestor de paquetes [19]. Diseñado para facilitar la instalación de librerías y dependencias necesarias para el desarrollo de software.

- `Coolors` para la elección de paleta de colores [20]. Es una herramienta gratuita que permite generar paletas de colores siguiendo patrones y combinaciones armónicas.

- `Canva` para la creación del logotipo y el diseño del fondo [21]. Una herramienta de diseño gráfico que permite crear presentaciones, gráficos, logotipos y fondos, entre otros, con gran variedad de plantillas y recursos gráficos.

- `Google Fonts` para la elección e importación de fuentes de texto [22] [23]. Una biblioteca de fuentes de código abierto que permite la inclusión de tipografías en aplicaciones de todo tipo.

- `Vercel`, para el despliegue de la aplicación [24]. Una plataforma de despliegue, gestión y alojamiento de aplicaciones web. Posee una integración directa con `GitHub` y facilita el despliegue de distintas ramas del proyecto.

- Otras librerías: `Vee-validate` [25] y `Vuetify`.

Capítulo 4

Normativa y legislación

En este capítulo se comentará la normativa y legislación que afecta al desarrollo de la aplicación. Las licencias del software que se han utilizado y las limitaciones sobre las que se ha desarrollado la aplicación.

Se ha priorizado el uso de uso de herramientas de código abierto y todas ellas han sido gratuitas.

4.1. Licencias de las herramientas utilizadas

- **Vue.js:** Licenciado bajo la MIT License, que permite su uso, modificación, distribución y reutilización tanto en proyectos personales como comerciales. Fuente: [10]

- **Supabase:** Licenciado bajo la licencia Apache 2.0, que permite el uso, modificación y distribución, incluso para fines comerciales, siempre que se mantenga la atribución y la licencia. Fuente: [15]

- **Figma:** Al ser un proyecto personal y solo se ha usado para la creación de bocetos básicos de la aplicación web, se permite su uso libre de cargo. Fuente: [16]

- **Canva:** Al ser un proyecto personal y solo se ha usado para la creación de fondos básicos de la aplicación web, se permite su uso libre de cargo. Fuente: [21]

- **Visual Studio Code:** Licenciado bajo la MIT License, aunque cabe destacar que el incluye componentes adicionales bajo licencia propietaria de Microsoft, aunque . Fuente: [18]

- **Coolors:** Se permite el uso de las paletas de colores de forma comercial y no comercial sin necesidad de dar atribuciones, pero si bajo unos términos de uso. Fuente: [20]

- **Google Fonts (Inter y Poppins):** Las fuentes están licenciadas bajo la SIL Open Font License (OFL), que permite su uso, modificación y distribución, incluso para fines comerciales. Fuente: [22] y [23]

- **Vercel:** Plataforma bajo licencia propietaria, con planes gratuitos y de pago. Al ser un proyecto personal, se está usando un plan gratuito. Fuente: [24]

- **Vee-Validate:** Licenciado bajo la MIT License, permitiendo su uso, modificación y distribución en proyectos personales y comerciales. Fuente: [25]

- **Vuetify:** Licenciado bajo la MIT License, permitiendo su uso, modificación y distribución en proyectos personales y comerciales. Fuente: [26]

- **Draw.io:** Licenciado bajo la licencia Apache 2.0, permitiendo su uso, modificación y distribución. Fuente: [27]

- **Responsively App:** Licenciado bajo la licencia GNU General Public License v3 (GPLv3), permitiendo su uso, modificación y distribución en proyectos personales y comerciales, bajo ciertas condiciones. Fuente: [28]

- **OpenStreetMap:** Todos los datos aportados por OpenStreetMap están bajo la Open Database License (ODbL), que permite copiar, distribuir, transmitir y modificar los datos, siempre que se atribuya de manera correcta a OpenStreetMap. Fuente: [11]

- **Leaflet:** Licenciado bajo la BSD 2-Clause "Simplified"License, que permite su uso, modificación y distribución, incluso para fines comerciales, siempre que se mantenga el aviso de copyright. Fuente: [12]

- **Nominatim:** Al ser una API que hace uso de OpenStreetMaps, está bajo su misma licencia. Especifica en sus términos de uso que puede ser usado libremente, pero al mantenerse gracias a donaciones se especifican ciertas limitaciones. Fuente: [13]

- **Pinia:** Licenciado bajo la MIT License, permitiendo su uso, modificación y distribución tanto en proyectos personales como comerciales. Fuente: [14]

4.2. Ley de Protección de Datos Personales y Garantía de los Derechos Digitales (LOPDGDD)

En cuanto a la Ley Orgánica de Protección de Datos y Garantía de los Derechos Digitales, la aplicación a realizar esta sujeta a la normativa vigente en España. Por tanto, se ha tenido en cuenta que, antes de recopilar datos personales, se debe informar al usuario de manera clara y concisa sobre el tratamiento de sus datos para luego recoger su posterior consentimiento haciendo uso de un *checkbox*. Además, los datos se protegerán al hacer uso de la herramienta `Supabase`. Entre varias características, cabe destacar que ni siquiera los creadores y administradores del sistema podrán acceder a la contraseña de los usuarios, puesto que están completamente encriptadas.

Los datos personales que se recopilan en la aplicación son:

- Pseudónimo del usuario.
- Correo electrónico del usuario.
- Contraseña del usuario (encriptada).
- Teléfono del usuario, de forma opcional.
- Localidad del usuario, de forma opcional.
- Datos sobre objetos extraviados o encontrados.

Los usuarios podrán exigir que sus datos sean eliminados por completo, y en cualquier momento, o que no se muestren públicamente en la web.

Capítulo 5

Competencias y aportaciones

En este apartado se especifica las aportaciones y competencias específicas que cubre el desarrollo de este portal web, así como su alineamiento con los Objetivos de Desarrollo Sostenible (ODS) de la ONU.

5.1. Aportación

El portal web busca crear una comunidad solidaria de usuarios que ayuden en la búsqueda de objetos perdidos, reduciendo así la cantidad de aquellos que nunca llegan a su propietario original y son muchas veces almacenados y/o mal clasificados en oficinas o departamentos de distinta índole de manera indefinida.

Esta aplicación pretende cooperar con las distintas organizaciones y empresas que gestionan objetos perdidos, como pueden ser aeropuertos, estaciones de tren, autobuses, etc., para facilitar la búsqueda de objetos perdidos y su posterior devolución a sus propietarios, reduciendo así la carga trabajo y el número de objetos almacenados que no llegan a su propietario original.

Esto se logra gracias a la implementación de un muro de publicaciones de objetos perdidos que todos los usuarios, registrados o no, pueden consultar en caso de haber encontrado algo que pueda pertenecer a otra persona en algún lugar. El usuario puede hacer uso de filtros avanzados de búsqueda para rápidamente comprobar si dicho objeto está en la base de datos y, en su caso, notificar a su propietario original de una manera rápida y sencilla.

Se prevé que el portal web pueda alcanzar un gran número de usuarios registrados con el marketing adecuado, puesto que el sistema de recompensas incentiva a los usuarios a participar en la búsqueda de objetos perdidos y la facilidad de uso del portal web permite a cualquier persona publicar un objeto perdido o encontrado de manera rápida y sencilla.

5.2. Competencias específicas

Como competencias específicas relacionadas de forma más directa, se ha especificado que el desarrollo de esta tecnología cumple las siguientes:

- **CI5:** Conocimiento, administración y mantenimiento de sistemas, servicios y aplicaciones informáticas.

- **CI16:** Conocimiento y aplicación de los principios, metodologías y ciclos de vida de la ingeniería del software.

- **TI1:** Capacidad para comprender el entorno de una organización y sus necesidades en el ámbito de las tecnologías de la información y las comunicaciones.

Se han cubierto de la siguiente manera:

- **CI5:** Este portal web hace uso de Supabase, una base de datos en la nube basada en PostgreSQL, Pinia como gestor de estados, Vue como framework de trabajo y Vercel para desplegar la aplicación, entre otras tecnologías. Se ha tenido que crear, conectar, administrar y mantener estos servicios, dando como resultado un sistema robusto y escalable disponible para todos los usuarios interesados.

- **CI16:** El desarrollo de esta aplicación ha seguido un ciclo de vida basada en una metodología iterativa e incremental. Se ha usado git y github como control de versiones para documentar cada una de las adiciones y cambios realizados en el código. A lo largo de los distintos incrementos se ha iterado sobre lo desarrollado en los anteriores, añadiendo o mejorando funcionalidades, corrigiendo errores y refactorizando código.

- **TI1:** Para el desarrollo de esta aplicación se han analizado otras similares, así como la situación en un departamento gubernamental, el de la Policía Local de Las Palmas de Gran Canaria. En base a las necesidades de los usuarios, y las carencias de la gestión existente, se ha desarrollado este portal web.

5.3. Alineamiento con los ODS

Cuadro 5.1: Grado de relación del TFT con los objetivos de desarrollo sostenible.

ODS	Grado de relación			
	0 No procede	1 Bajo	2 Medio	3 Alto
1 Fin de la Pobreza	x			
2 Hambre cero	x			
3 Salud y Bienestar	x			
4 Educación de calidad	x			
5 Igualdad de género	x			
6 Agua limpia y saneamiento	x			
7 Energía Asequible y no contaminante	x			
8 Trabajo decente y crecimiento económico	x			
9 Industria, Innovación e Infraestructuras			x	
10 Reducción de las desigualdades	x			
11 Ciudades y comunidades sostenibles	x			
12 Producción y consumo sostenibles	x			
13 Acción por el clima	x			
14 Vida submarina	x			
15 Vida de ecosistemas terrestres	x			
16 Paz, justicia e instituciones sólidas				x
17 Alianzas para lograr objetivos				x

5.3.1. Justificación del alineamiento con los ODS

- **ODS 9 - Industria, Innovación e Infraestructuras:** Con este proyecto se busca dar mejores y más fáciles soluciones a un problema grave, como es la pérdida de objetos personales, con tecnología que innova con respecto a sus similares. Pretende ayudar al crecimiento sostenible de la industria moderna, al empujar un progeso tecnológico en la búsqueda de objetos perdidos, creando una comunidad de usuarios sobre una infraestructura de datos robusta que facilite e incite la búsqueda de objetos extraviados de otros.

- **ODS 16 - Paz, justicia e instituciones sólidas:** El portal web busca proteger la privacidad de los usuarios al mantener una base sólida sobre la que evoluciona. Pretende construir una comunidad que se ayude entre sí a encontrar objetos perdidos, fomentando la paz, buena fe y la justicia social, devolviendo a sus propietarios los objetos que una vez extraviaron. Además, el sistema de recompensas apoya dicha iniciativa al incentivar un mayor grado de implicación, confianza y compromiso de cada uno de los miembros de la comunidad.

- **ODS 17 - Alianzas para lograr objetivos:** La aplicación se basa en gran medida en la colaboración y la buena fe entre los usuarios. Al crear una comunidad rígida de personas, se busca reducir la cantidad de objetos perdidos que nunca llegan a su propietario original y son muchas veces almacenados y/o mal clasificados en oficinas o departamentos de distinta indole de manera indefinida.

Capítulo 6

Análisis y diseño

A lo largo de este capítulo se hablará sobre la funcionalidad principal, roles y el estilo de la aplicación web desarrollada. Además, se presentará un diagrama de los casos de uso clave de la aplicación.

6.1. Funcionalidad principal, roles y estilo

Los roles son los siguientes:

- **Invitado:** Persona no registrada.
- **Usuario:** Persona registrada.
- **Administrador:** Persona encargada de la gestión interna de la web.

6.1.1. Invitados

1. Navegar por una lista de objetos perdidos.
2. Notificar que un objeto de la lista ha sido encontrado por el propio usuario.

6.1.2. Usuarios registrados

1. Hacer todo lo anterior.
2. Publicar si ha encontrado/perdido un objeto.
3. Borrar publicaciones propias.
4. Recibir notificaciones en la sección de notificaciones de la web.
5. Tener una página de perfil personal.

6.1.3. Administradores

1. Hacer todo lo anterior
2. Navegar por una lista de objetos encontrados.
3. Notificar a usuarios registrados de coincidencias entre objetos perdido y encontrados de dentro de la base de datos.
4. Vetar cuentas de usuarios.
5. Borrar publicaciones de usuarios.

6.1.4. Funcionalidad principal

La funcionalidad ideal reducida a lo importante sería la siguiente:

Cuando un usuario entra en el portal web, le da la bienvenida el portal principal de objetos perdidos, donde puede observar una lista de tarjetas de objetos que han sido perdidos y una barra lateral simple que puede ser usada para navegar entre las distintas secciones de la web, además de un sistema de filtros para personalizar su búsqueda. Vería una lista algo similar a la mostrada en la Figura 6.1.

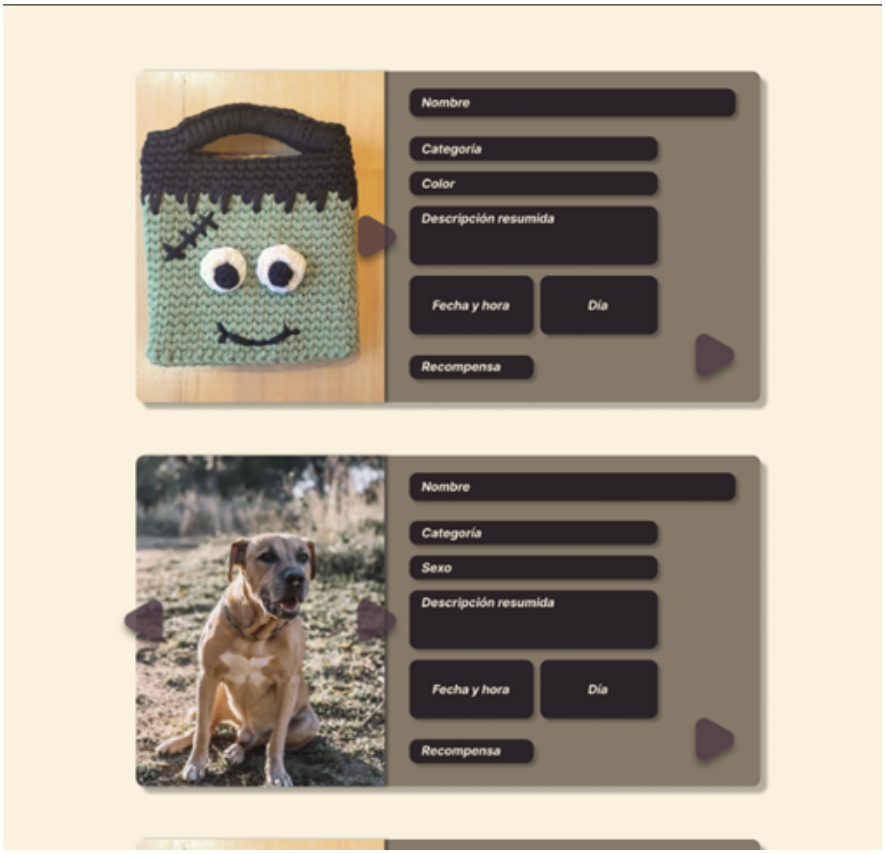

Ilustración 6.1: Boceto de portal de publicaciones.

El usuario podrá acceder a los detalles de una publicación e indicar, en su caso, que poseen el objeto en cuestión, enviando una notificación de coincidencia al la persona dueña del objeto.

En la barra lateral estará ubicado una serie de botones para iniciar sesión, cerrar sesión, acceder al perfil personal y otro para realizar una publicación, en caso de que el usuario hubiera perdido o encontrado algo que deseara publicar en el portal web.

En la Figura 6.2 se observa un boceto del proceso que sigue un usuario registrado a la hora de indicar que, por ejemplo, ha perdido algo. Al terminar el formulario de distintos pasos, la publicación se subirá a la base de datos y pasará a ser visible para todo el mundo en el portal principal.

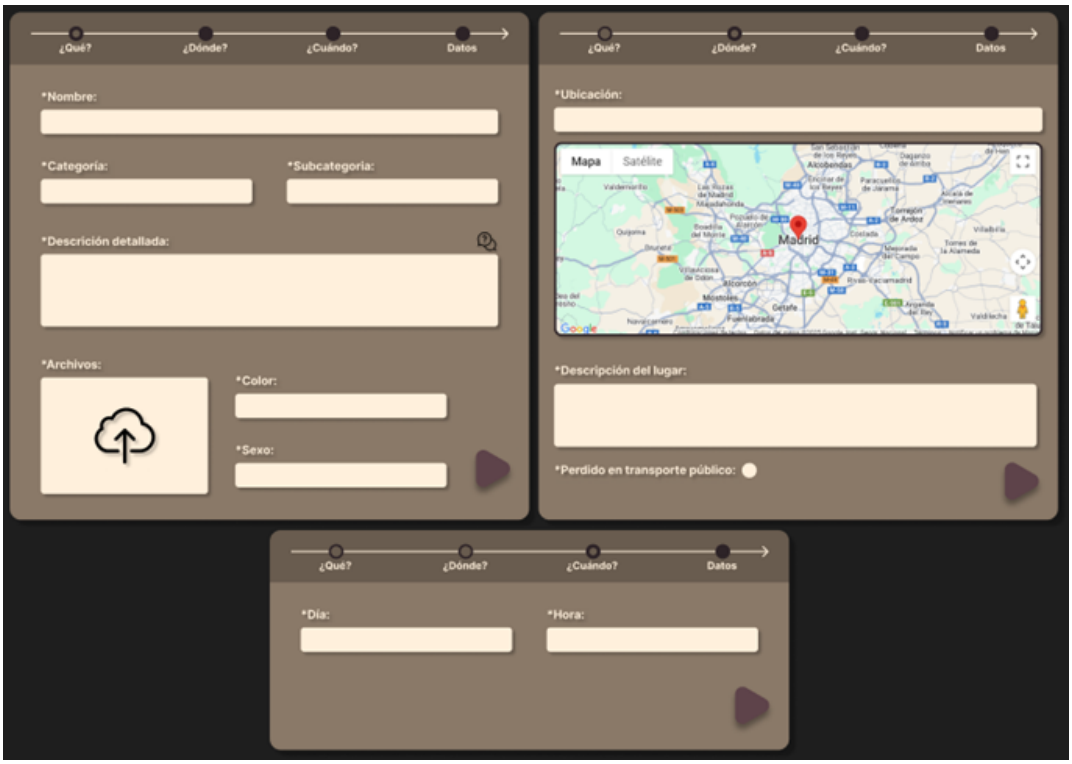

Ilustración 6.2: Boceto de proceso de publicación.

Los usuarios, además, podrá recibir notificaciones de coincidencia en el caso de que uno de los objetos que perdiera, haya sido encontrado. Tras la confirmación del mismo, el objeto es eliminado de la base de datos. El boceto de la sección de notificaciones puede ser vista en la Figura 6.3.

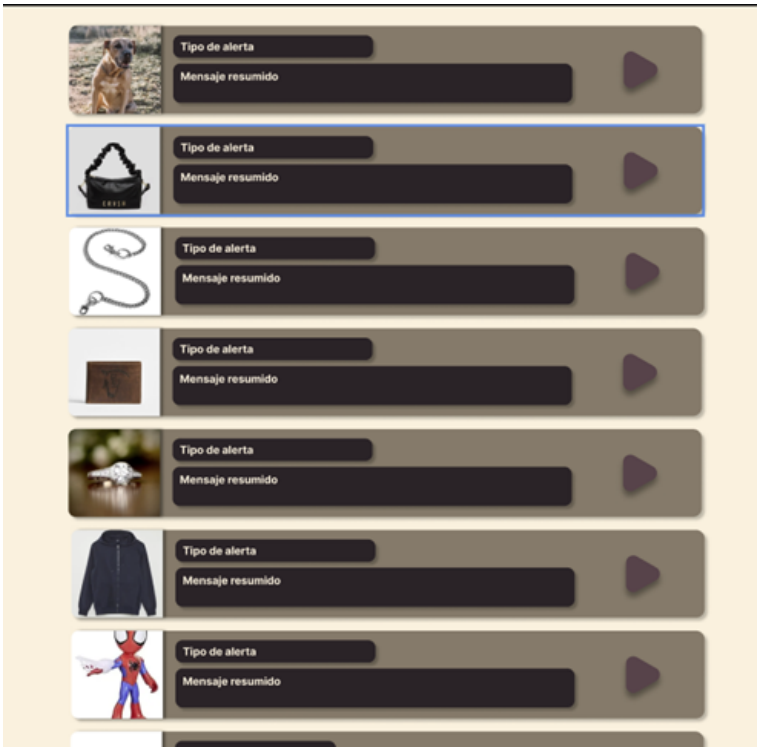

Ilustración 6.3: Boceto de portal de notificaciones.

Por último, los Administradores pueden buscar coincidencias entre objetos perdidos y encontrados que ya están dentro de la propia base de datos, enviando notificaciones a los implicados para que se pongan en contacto.

6.1.5. Estilo

En cuanto al estilo, se ha elegido una paleta de colores teniendo en mente un ambiente agradable y familiar con colores que atraigan por su simpleza. Para la elección de esta se ha hecho uso de la herramienta online Coolors [20]. La paleta de colores está disponible en la Figura 6.4.

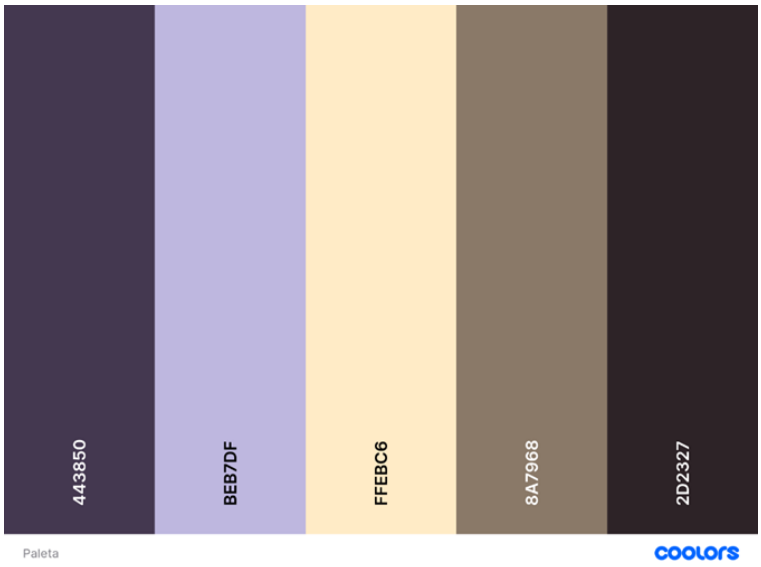

Ilustración 6.4: Paleta de colores.

Finalmente, se decide usar las fuentes Inter [22] y Poppins [23], ambas libres de cargo bajo la licencia de SIL Open Font [29]. Se pretende desarrollar una interfaz lo más limpia y simple posible, con el objetivo de atraer usuarios de todas las edades.

6.2. Casos de uso

A continuación, se muestra el diagrama de casos de uso clave del portal web, realizado antes del comienzo del desarrollo de la aplicación. Se ha usado la herramienta web Draw.io [27] siguiendo el estándar UML.

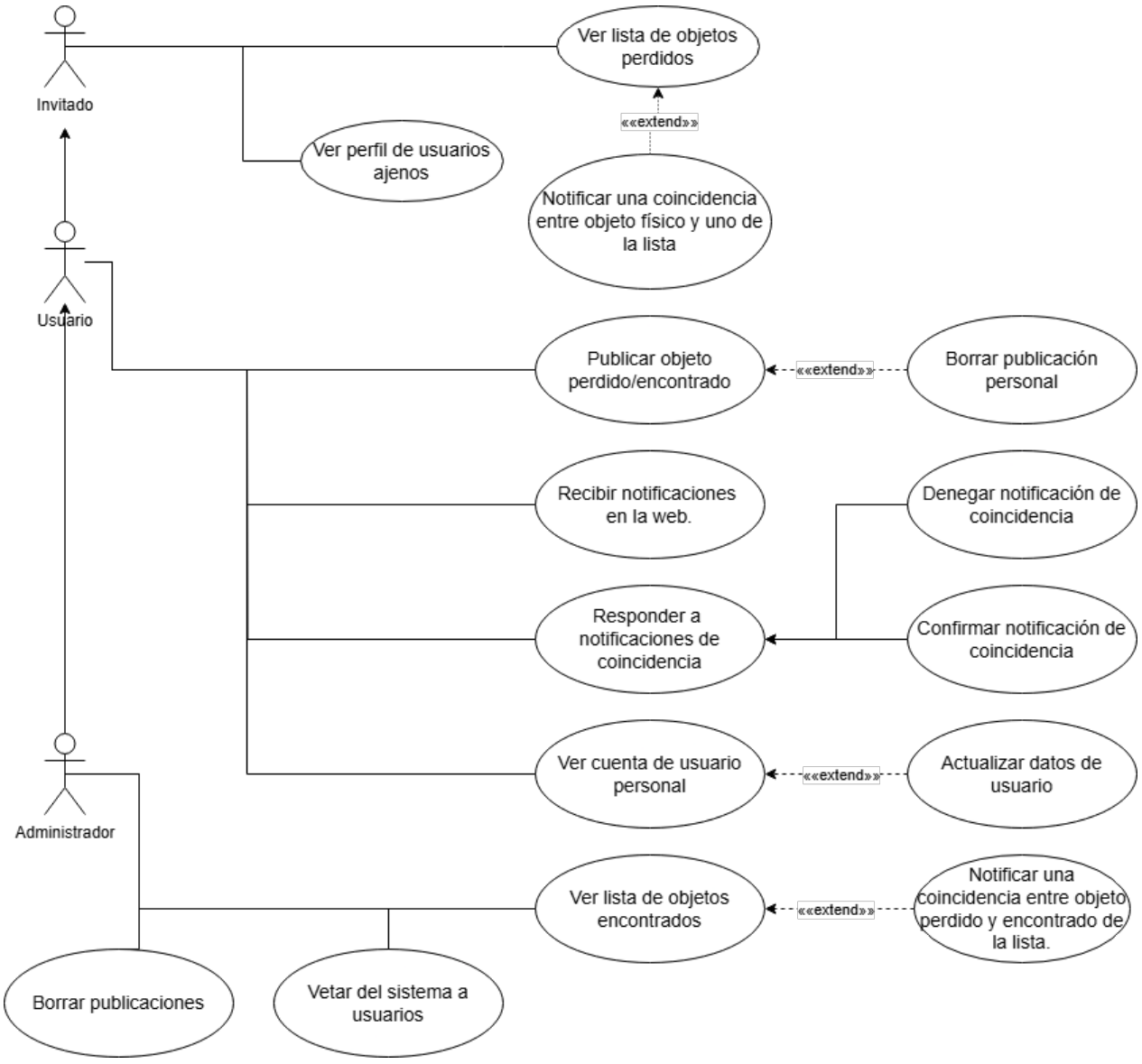

Ilustración 6.5: Casos de uso clave.

Capítulo 7

Desarrollo

En esta sección se describe los pasos seguidos para desarrollar la aplicación web a lo largo de los distintos incrementos, así como las decisiones tomadas durante el proceso, los cambios que ha sufrido el portal web y los fragmentos de código relevantes para entender el funcionamiento del mismo. Como la aplicación web ha sido producto de un trabajo de gran envergadura, se prioriza comentar y mostrar los fragmentos más relevantes del proceso para ajustar el contenido a la extensión exigida.

7.1. Diseño arquitectónico de la web

Los primeros pasos para el desarrollo de la aplicación fue descargar e instalar el entorno de desarrollo, `visual studio code` y `npm`, e inicializar el proyecto de `Vue`, haciendo uso de *Vue CLI* 7.1:

```
npm install −g @vue/cli
vue create nombre_proyecto
```

Código 7.1: Crear proyecto con Vue CLI

Se toma la decisión de usar el framework front-end de `Vue` por sus integraciones, librerías disponibles y por la gran cantidad de documentación de fácil acceso que existe en internet. Este framework ofrece la capacidad de usar variables reactivas que permiten la actualización dinámica de información entre el DOM y el código que compone la aplicación, y ofrece librerías que permiten la creación de componentes atractivos, escalables y reutilizables. Todo esto, junto con su integración con `TypeScript`, lo convierte en la opción ideal para el desarrollo de este portal web moderno.

Al crear el proyecto, se elige las opciones predeterminadas mínimas necesarias, solo especificando el uso de `TypeScript` en vez de `JavaScript`. Se prioriza la elección de TypeScrypt por su tipado estático, que permite detectar errores con mayor facilidad y mantener un código más limpio, mantenible y escalable.

Al tener listo el proyecto, ya es posible lanzar el servidor de desarrollo o compilarlo para producción. Para ello, se utilizan los siguientes comandos 7.2 7.3:

```
vue−cli−service serve
```

Código 7.2: Lanzar servidor de desarrollo

```
vue−cli−service build
serve −s dist
```

Código 7.3: Lanzar producción en local

Al tener disponible el proyecto predeterminado que ofrece Vue CLI, se comienza la adaptación del mismo siguiendo una determinada organización de carpetas y archivos, como se puede observar en la Figura 7.1

```
src/
├── assets/
├── components/
├── containers/
├── enums/
├── interfaces/
├── views/
├── stores/            # Pinia
├── json/
├── utils/             # Utilidades matemáticas, de texto y API Externas
├── router/
├── App.vue
├── main.ts
├── main.css/
└── supabase.ts        # Supabase
.env
```

Ilustración 7.1: Organización de carpetas del proyecto

Aunque el portal web está pensado para ser usado por personas de habla hispana, se hace uso de la lengua inglés en la organización interna y el código del proyecto. Considerada la lengua internacional, es la más utilizada en el mundo a día de hoy y, por lo tanto, hace que el código esté al alcance de un mayor número de personas, haciendo el proyecto más sostenible y escalable.

Cada una de las carpetas y ficheros especificados en la Figura 7.1 cumplen una función específica dentro del proyecto. Se explican brevemente a continuación:

- **assets**: Imágenes, íconos, etc.
- **components**: Componentes reutilizables y libres de lógica externa. Se encargan de presentar los datos e interactuar con el usuario.
- **containers**: Contenedores que agrupan y gestionan componentes. Encargados de comunicarse con el gestor de estado, `Pinia`.
- **enums**: Enumerados utilizadas en la aplicación.
- **interfaces**: Definiciones de interfaces `TypeScript`.
- **views**: Vistas de la aplicación. Agrupa contenedores y/o componentes con el único objetivo de mostrarlo todo al usuario.
- **stores**: `Pinia`, para el manejo del estado global de la aplicación. Se encarga de comunicarse con la base de datos, las APIs externas, etc. En definitiva, todo lo relacionado con el back-end de la aplicación.
- **json**: Archivos `JSOn` utilizados en la aplicación.
- **utils**: Utilidades matemáticas, de texto y funciones para llamar a APIs externas no principales.
- **router**: Configuración del enrutamiento de Vue. Se encarga de gestionar las rutas de la aplicación y la navegación entre ellas.
- **App.vue**: Componente padre de la aplicación. Aquí se enrutan las vistas creadas según la navegación del usuario y se muestran los componentes principales de la aplicación.

- **main.ts**: Archivo de configuración principal. Se encarga de inicializar la aplicación y cargar los plugins y librerías necesarias.

- **main.css**: Archivo encargado de los estilos globales. Aquí se define la paleta de colores y la fuente, entre otros estilos de carácter general.

- **supabase.ts**: Configuración de `Supabase`.

- **.env**: Variable de entorno. Necesaria para la configuración de `Supabase`. Debe contener la URL y la clave de la API del proyecto de `Supabase`.

Se ha decidido organizar la aplicación web de esta manera con el objetivo de mantener un código limpio, escalable y modular. Esto se logra al separar la lógica de negocio de la presentación, encapsulando las llamadas a las APIs externas y a la base de datos en `stores` de `Pinia` de distinto tipo. Estos `stores` solo son accedidas por lo contenedores, que se encargan unicamente de pedir la información deseada y pasarla a los componentes con el tipado necesario. De esta manera, si fuera necesario realizar cambios en el back-end, solo sería necesario modificar el código del `store` correspondiente. Así mismo, es posible cambiar, por ejemplo, la base de datos sin necesidad de modificar ningún componente. Todo esto se apoya en el uso de interfaces de `TypeScript` para definir la estructura de los datos que se van a manejar en la aplicación y mantener una cohesión en todo el proyecto.

Por otro lado, el uso de `Pinia` permite crear un estado global de la aplicación, es decir, permite almacenar datos que luego pueden ser usados en distintos componentes. Esto reduce la cantidad de llamadas a la base de datos y a las APIs externas, con el objetivo de mejorar el rendimiento de la aplicación así como reducir los posibles futuros costes asociados al uso del back-end.

Tras la reorganización del proyecto, se inicializa el repositorio de `Git`, se define el archivo de las variables de entorno junto a todos los paquetes de npm en el archivo `.gitignore` y se sube lo demás a `GitHub`. Se ha decidido utilizar `GitHub` como plataforma de control de versiones por su facilidad de uso, su integración con otras tecnologías como `Vercel` y la facilidad a la hora de documentar el código tras cada sesión de trabajo. El enlace al mismo es el siguiente: https://github.com/alvarosacosta/LostHub.

Al tener preparado el entorno y la organización del proyecto, se comienza a desarrollar la aplicación. En este primer incremento se comienza el desarrollo de los componentes fundamentales de la aplicación, en este caso la barra de navegación, la página principal de objetos, la tarjeta de objeto y el componente de los detalles del objeto. Se consideran fundamentales porque el resto de la aplicación gira en torno a ellos.

A lo largo del incremento, se añade un nuevo componente, una sección de *Quiénes somos*, para dar más información al usuario sobre el proyecto y su funcionamiento, de no tanta relevancia.

A su vez, para realizar estos componentes, y al no tener aún un back-end funcional, se decide crear un `JSOn` con datos de prueba para simular el funcionamiento de la aplicación.

Comenzando por la barra de navegación, inicialmente se realiza el siguiente diseño, disponible en la Figura 7.2:

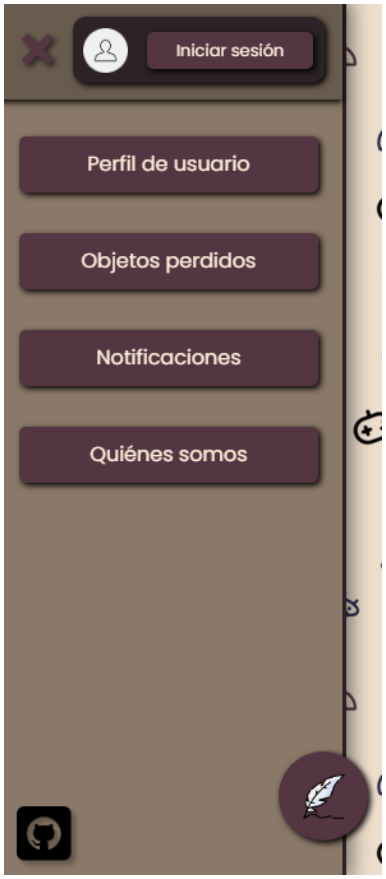

Ilustración 7.2: Diseño de la barra de navegación

Se hace uso de la paleta de colores definida, y se decide realizar un diseño sencillo y minimalista con botones que sobresalen para llamar la atención del usuario. Esta barra de navegación puede colapsarse al hacer click en el icono *X* de la parte superior izquierda, y se puede volver a desplegar al hacer click en el icono de *hamburguesa*, disponible en el mismo lugar, como se puede observar en la Figura 7.3. La vista principal se adapta a la interacción con el componente.

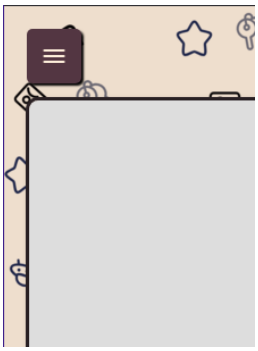

Ilustración 7.3: Botón para volver a desplegar la barra de navegación

Todos los botones visibles aún no tienen funcionalidad, a excepción del botón de *Objetos perdidos*, que lleva a la página donde se alojará la lista de objetos extraviados, haciendo uso del router de `Vue`. Aunque la funcionalidad de la mayoría de botones se explica por si misma, cabe mencionar que el botón con el icono de pluma redirigirá al usuario en futuros incrementos a la página de publicación de objetos y el botón con símbolo de `Github` permite al usuario navegar a la página del repositorio de `GitHub` del proyecto.

A continuación, se realiza el diseño de la carta/tarjeta de objeto, disponible en la Figura 7.4:

Ilustración 7.4: Diseño de la tarjeta de objeto

Este componente se crea con el objetivo de mostrar un pequeño resumen de las características del objeto. Este elemnto contiene un botón en forma de flecha que permite al usuario acceder a la página de detalles del mismo. Como se puede ver en la Figura 7.5, el componente posee un carrusel de imágenes interactuable que permite ver de una manera atractiva las imágenes asociadas al objeto, mostrando las flechas de navegación al pasar el ratón por encima del componente. Así mismo, también existe una versión vacía por si el usuario no ha subido imágenes de ningún tipo.

Ilustración 7.5: Diseño del carrusel de imágenes en la tarjeta de objeto

Inicialmente, para la página principal de objetos perdidos, se decide mostrar un listado vertical simple de elementos, que permite la fácil interacción del usuario 7.6.

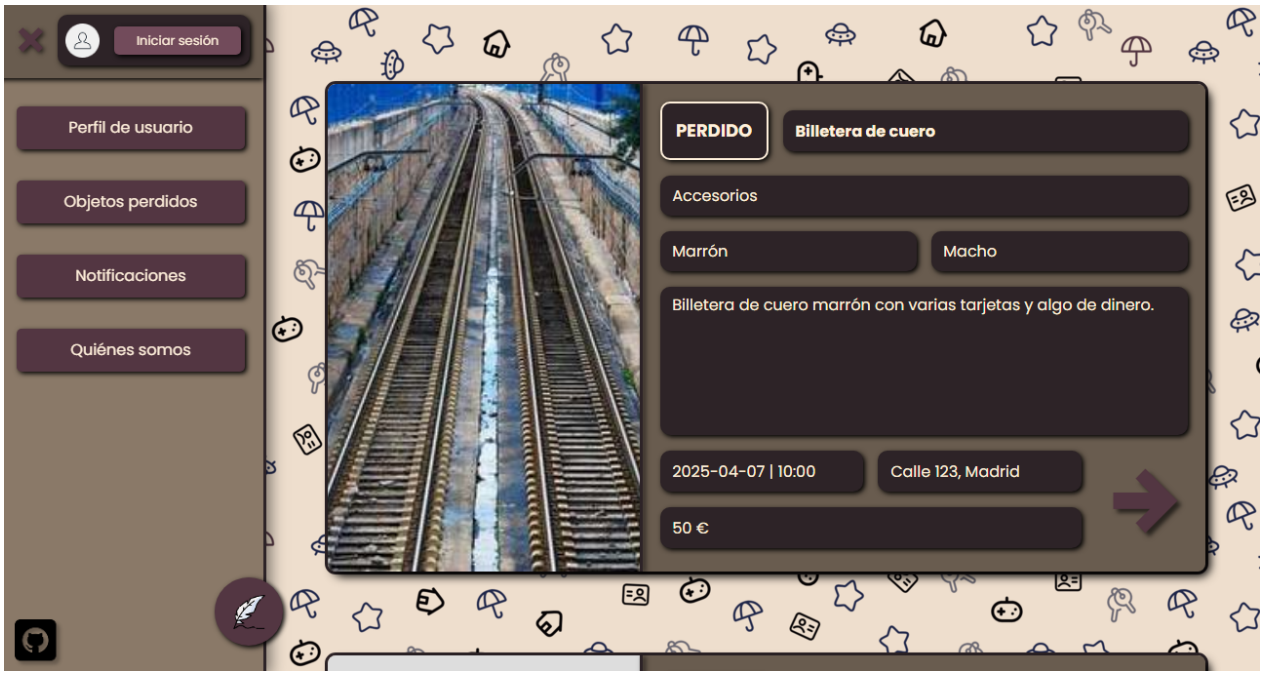

Ilustración 7.6: Diseño de la página principal de objetos perdidos

Para la página de detalles del objeto, inicialmente se realiza el siguiente diseño, disponible en la Figura 7.7:

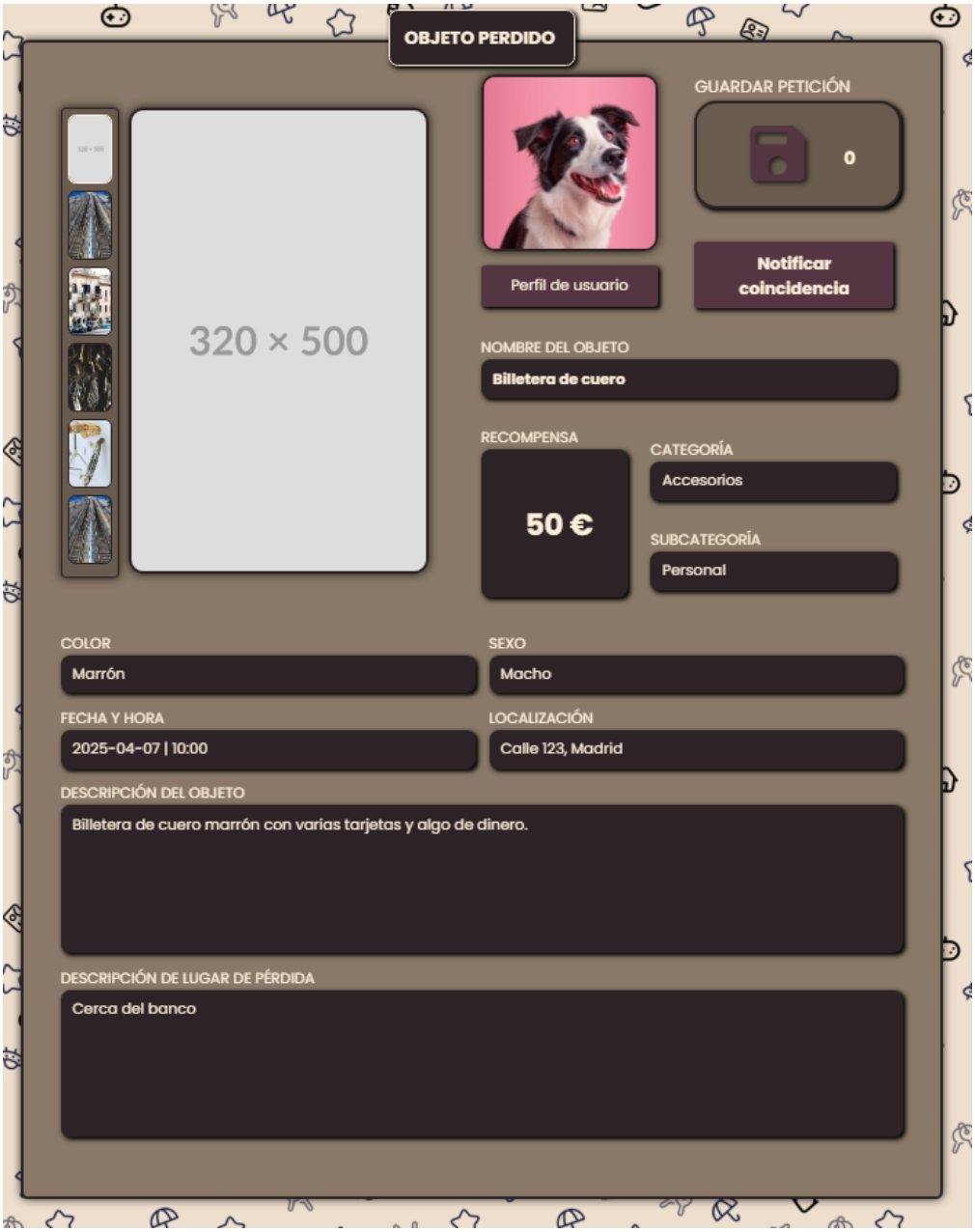

Ilustración 7.7: Diseño de la página de detalles del objeto

Es en este componente donde se muestra toda la información del objeto, incluidos los detalles del propietario y el botón para poder notificar que el objeto ha sido encontrado.

Mientras se desarrollaba este componente, se ha ideado una nueva característica no contemplada con anterioridad: permitir que el usuario pueda guardar peticiones de búsqueda del objeto en una lista, a modo de *favoritos*. Así el usuario tiene un más rápido acceso a los objetos que le han interesado en el pasado, por si en el proceso de búsqueda del mismo quisiera volver a consultar cualquier dato. Para lograr esto se ha añadido un botón todavía no funcional y se implementa un contador de las veces que el objeto ha sido guardado. En futuras versiones este contador es eliminado debido a que no añade valor a la aplicación, al no ser relevante para el usuario dueño del objeto ni para aquel que esté en su búsqueda.

Esta característica vuelve a ser comentada en el apartado de trabajo futuro, pues no fue implementada en su totalidad a lo largo del proyecto.

Por otro lado, se desarrolla de nuevo un carrusel de imágenes un poco más elaborado, con un sistema interactuable de *preview* de las imágenes a la izquierda del mismo, pero manteniendo también la funcionalidad principal visible en la Figura 7.5.

Finalmente, cabe destacar que el fondo inicial de la aplicación fue diseñado por el autor de la memoria, haciendo uso de `Canva` e imágenes libres de uso. Visible en detalle en la Figura 7.8.

Ilustración 7.8: Fondo de la aplicación

Con respecto a la parte interna de la aplicación, se instala la librería de `Supabase` junto con la de `Pinia` 7.4.

```
npm install @supabase/supabase−js
npm install pinia
```

Código 7.4: Instalar librería de Supabase y Pinia

Tras ello, se prepara el archivo *supabase.ts* y *main.ts*. Además, se crea la versión preliminar del store de autentificación dentro de la carpeta *stores* y, por último, se deja preparada la versión inicial de la interfaz de los objetos perdidos, disponible en la carpeta *interfaces*. Todo esto con el objetivo de crear las bases sobre las que construir la aplicación en las siguientes iteraciones y cumplir así con el objetivo de este incremento.

Como colofón, cabe destacar que todas las partes que componen esta aplicación se desarrollan teniendo en cuenta un diseño *responsive* que permite su uso en dispositivos móviles, tablets y ordenadores. Esto se logra haciendo uso de `CSS` y en ciertas ocasiones de `TypeScript`. Se han realizado multitud de pruebas en el buscador de Brave, Microsoft Edge y haciendo uso de la herramienta `Responsively App`, que permite ver la aplicación en diferentes dispositivos y resoluciones al mismo tiempo.

El rango de anchura de pantalla que la aplicación soporta sin errores es de *2560px* a *375px*. Esto con el objetivo de aumentar la calidad general de la aplicación, así como su usabilidad y la cohesión de lo que este servicio ofrece entres dispositivos de todo tipo.

7.2. Diseño e implementación de base de datos.

7.2.1. Incremento

El objetivo de este incremento es el de crear e implementar en la aplicación la base de datos `Supabase` de forma básica. Para ello, se realiza principalmente lo siguiente:

1. Crear la base de datos en `Supabase`.

2. Crear las tablas de los detalles del usuario y de los objetos.

3. Implementar la conexión básica entre la aplicación y la base de datos creando los stores necesarios.

4. Crear los componentes adecuados para el registro, inicio de sesión, modificación de perfil y cierre de sesión del usuario.

5. Eliminar el JSOn con objetos de prueba e implementar la carga de objetos desde la base de datos en su lugar.

De forma secundaria y no planeada se desarrolla lo siguiente:

1. Componente de carga de pantalla, que se muestra al navegar a rutas en las que hay que cargar datos.

2. Componente de muestra de errores y mensajes de éxito.

3. Componente de bienvenida al usuario, que se muestra al entrar en la aplicación por primera vez.

Estas últimas características desarrolladas con el objetivo de mejorar la experiencia de usuario y la usabilidad de la aplicación.

Algunas de las razones por la que se ha elegido `Supabase` como base de datos es por su facilidad de uso, la comodidad que aporta una base de datos relacional, sus servicios en la nube y su integración con el framework de `Vue` y `Pinia`. No obstante, la razón principal es que `Supabase` permite descargar la base de datos creada en formato SQL, lo que facilita su transporte a otros entornos, por si se quisiera escalar el proyecto haciendo uso de servidores propios o de otros servicios en la nube.

Para crear la base de datos en `Supabase` es necesario registrarse en la plataforma y crear un nuevo proyecto. Una vez creado, se accede a la sección de base de datos y se construyen las tablas necesarias. En este caso, se crean dos tablas: una para los detalles del usuario y otra para los objetos. Además, se crean dos *buckets*: uno para almacenar las imágenes asociadas a los objetos perdidos y otro para las imágenes de perfil de los usuarios. Una vez realizado lo anterior, se modifica el fichero *.env*, añadiendo las variables de entorno necesarias para hacer uso de `Supabase` en la aplicación, en este caso la *URL* y la *KEY* proporcionada por los servicios de `Supabase`.

`Supabase` está basado en PostgreSQL, por lo que se utilizan comandos SQL para crear las tablas y definir sus relaciones con mayor precisión, así como las políticas de lectura, escritura y eliminación de los datos. En este caso, se crean dos tablas, *user_details* y *item_details*.

A continuación, se procede a crear los stores necesarios para la conexión entre la aplicación y la base de datos:

- **authStore**:*Store*para gestionar la información del usuario, incluyendo su registro, inicio de sesión y cierre de sesión.

- **itemStore**:*Store*para gestionar la información de los objetos perdidos y encontrados.

Aunque se desarrollan multiples métodos para la gestión tanto del usuario como de los objetos, en las Figura 7.9 y 7.10 se muestran los fragmentos de las funciones más relevantes, ideales para explicar la lógica seguida en todos ellos.

```
async function signUp(user: User, userDetails: UserDetails, userProfileImage: UserProfileImage) : Promise<void>{
  var filePath = `user-profiles-images/no-image.png`;

  try {
    const { data: signUpData, error: signUpError } = await supabase.auth.signUp({
      email: user.email,
      password: user.password,
    });
    if (signUpError) throw signUpError;      You, last month via  PR #2 • feature: Added SignUp and LogIn options

    const userID = signUpData.user?.id;
    if (!userID) throw new Error('Could not get user ID.');

    await logIn(user.email, user.password);

    if (userProfileImage.profilePicture){
      filePath = `user-profiles-images/${userID}_${Date.now()}.jpg`;

      const { error: uploadError } = await supabase.storage
        .from('user-profiles-images')
        .upload(filePath, userProfileImage.profilePicture);

      if (uploadError) throw uploadError;
    }
```

Ilustración 7.9: Registro de usuario, ejemplo de **authStore**.

La función mostrada en la Figura 7.9 es la encargada de registrar al usuario en la base de datos con su contraseña y email, luego iniciar sesión y guardar los datos secundarios en la tabla de los detalles del usuario. Como se puede observar, se hace uso de interfaces de `TypeScript` para definir los tipos que se van a almacenar en la base de datos y se hace uso de la librería de `Supabase` para realizar las llamadas a la base de datos pertinente.

```
async function fetchAllItems() : Promise<void>{
  try {
      const { data, error } = await supabase
          .from('item_details')
          .select('*')

      if (error) throw error;

      if (data) {
          allItems.value = data.map((item: any): MixedItem => {
              const baseItem = {
                  id: item.id,
                  name: item.name,
                  category: item.category,
                  subcategory: item.subcategory,
                  gender: item.gender,
                  color: item.color,
                  dateTime: item.datetime,
                  location: item.location,
                  detailedDescription: item.description,
                  locationDescription: item.location_description,
                  isPublic: item.is_public,
                  isLostInPublicTransport: item.is_lost_in_public_transport,
                  url_images: item.url_images,
                  type: item.type,
              };
```

Ilustración 7.10: Carga de objetos, ejemplo de **itemStore**.

En el caso mostrado en la Figura 7.10, se observa la función encargada de descargar los objetos disponibles de la base de datos, almacenándolos en el store. Como se puede ver, se hace uso de la interfaz *MixedItem*, creada en el anterior incremento, para recoger los datos de los objetos cargados desde la base de datos y luego almacenarlos en el*store*para su posterior uso en la aplicación.

A continuación, en los contenedores de la aplicación se llama a las funciones creadas en los stores, y se recoge los datos almacenados en el mismo. Un ejemplo simple visible en la Figura 7.11.

```vue
<template>
    <main class="ItemsContainer">
        <ItemsComponent
            :items
        />
    </main>
</template>

<script setup lang="ts">
import ItemsComponent from '@/components/ItemsComponent.vue';
import { onMounted, Ref, ref } from 'vue';
import { useItemsStore } from '@/stores/database';
import { MixedItem } from '@/interfaces/items';

    const ItemsStore = useItemsStore()
    var items : Ref<MixedItem[] | undefined> = ref(undefined)

    onMounted(async () => {
        await ItemsStore.fetchAllItems();
        items.value = ItemsStore.allItems;
    });

</script>
```

Ilustración 7.11: Contenedor de lista de objetos.

Se puede observar como el trabajo del contenedor es el de llamar a las funciones del store, recoger los datos y pasarlos a los componentes de la aplicación. Estos componentes hacen uso de los datos recogidos sin necesidad de conocer la lógica del back-end, como se ha explicado anteriormente. En otros casos el contenedor recoge los datos del componente y llama al *store* para que este realice la acción pertinente, como puede ser el registro de un nuevo usuario.

Cabe mencionar que el código mostrado hasta ahora es lo que se desarrolló a lo largo del segundo incremento y que mucho del mismo ha sido refactorizado, mejorado y ampliado en futuras iteraciones.

Con respecto a los componentes necesarios para el registro, inicio de sesión, modificación de perfil y cierre de sesión del usuario, se observan en las Figuras 7.12, 7.13 y 7.14, respectivamente. Todos ellos completamente funcionales y con los posibles errores y avisos de validación implementados. Creados con el apoyo de la librería `Vuetify`.

Ilustración 7.12: Componente de registro de usuario.

Ilustración 7.13: Componente de inicio de sesión.

Ilustración 7.14: Componente de sección de usuario.

Al terminar este incremento, la base de datos está complemente conectada a la aplicación y se pueden realizar las acciones de registro, inicio de sesión, cierre de sesión y modificación del perfil del usuario. Además, los objetos perdidos son descargados de la base de datos y se muestran en la aplicación correctamente.

Finalmente, merece la pena comentar uno de los componentes que se desarrollaron de forma secundaria y no planeada, pero que mejoran la experiencia de usuario y la usabilidad de la aplicación en gran medida. En concreto el componente de mensajes de error y éxito, que se muestra en la parte superior de la aplicación con una animación de entrada y salida, e informa al usuario de si la acción que acaba de realizar ha tenido éxito o ha fallado. Un elemento primordial para la usabilidad de la aplicación. Un ejemplo de su uso se puede observar en la Figura 7.15 y 7.16.

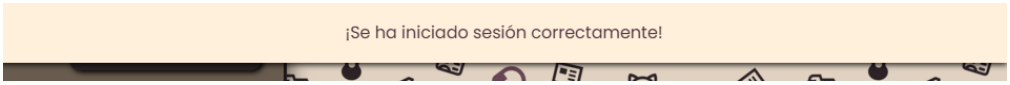

Ilustración 7.15: Componente de mensajes: **Éxito**.

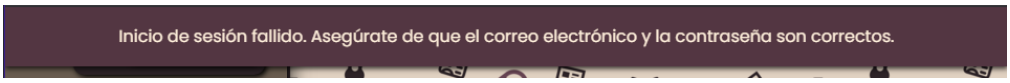

Ilustración 7.16: Componente de mensajes: **Error**.

7.2.2. Iteración

Se ha seguido una metodología iterativa/incremental, por lo que en este incremento también se ha iterado sobre el diseño y la implementación anterior de la aplicación, refactorizando código, modificando estilos y mejorando la usabilidad de la aplicación. En este caso, los aspectos nuevos más relevantes que aún no se han comentado son los siguientes:

- Creación de un nuevo fondo para la aplicación con ayuda de `Canva`, visible en la Figura 7.17.

- Modificación de la barra de navegación. Se muestran más o menos opciones dependiendo de si el usuario ha iniciado sesión o no y se ha añadido funcionalidad a los botones referentes a lo desarrollado en este incremento.

- Modificación de los detalles del objeto. Ahora el botón de *Perfil de usuario* visible en la Figura 7.7 es funcional y muestra los detalles del usuario asociado a ese objeto.

- Modificación de estilo en todos los componentes.

- Refactorización menor del código.

Ilustración 7.17: Fondo final de la aplicación.

7.3. Diseño e implementación de vistas del usuario

7.3.1. Incremento

El objetivo principal de este incremento fue el de dejar finalizadas las vistas principales del usuario registrado a excepción de la sección de notificaciones. Por lo que en este lapso de tiempo se realiza principalmente lo siguiente:

- Crear proceso para publicar en la base de datos un objeto perdido o encontrado.
- Crear proceso para que un usuario pueda notificar a otro que ha encontrado un objeto perdido.
- Crear sistema de filtros para la búsqueda de objetos perdidos.
- Crear la tabla de notificaciones de usuario en la base de datos.

De forma secundaria y no planeada se realiza lo siguiente:

- Crear componente que muestra al usuario sus objetos publicado.
- Implementar mapa de `OpenStreetMap` con la librería `Leaflet` en vez de hacer uso de `Google Maps`.
- Alojar rama de producción en los servidores de `Vercel`.

Al crear la tabla de notificaciones en la base de datos y dejarla preparada para su futuro uso, se comienza desarrollar el proceso de publicación de objeto perdido o encontrado. En las siguientes Figuras 7.18 7.19 7.20 7.23 se muestra fragmentos relevantes de lo que se ha desarrollado.

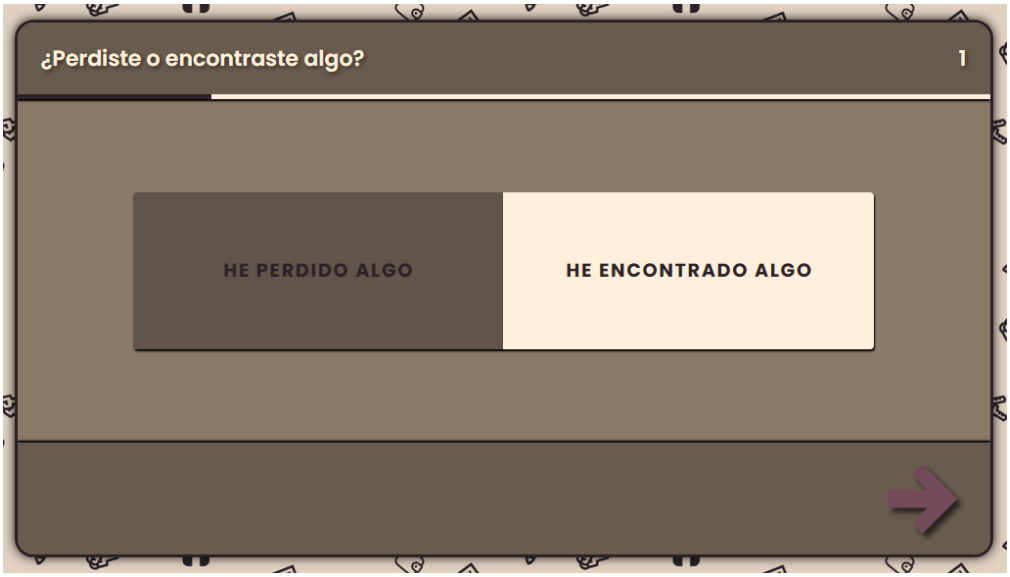

Ilustración 7.18: Proceso de publicación de objeto perdido o encontrado: **Inicio**.

Al iniciar el proceso, el usuario puede elegir entre publicar un objeto que ha perdido o uno que ha encontrado. Los objetos perdidos se encontrarán en la página principal de objetos extravidos, mientras que los objetos encontrados estarán ocultos y solo serán visibles por los administradores de la aplicación.

Se toma esta decisión puesto que mostrar objetos encontrados al público podría llegar a generar una avalancha de personas que aseguran ser los propietarios de lo publicado, lo que podría generar confusión, malentendidos y romper la dinámica de comunidad que la aplicación pretende construir.

No obstante, este rol de administrador no ha sido implementado aún y ha quedado delegado como trabajo futuro, puesto que se ha querido priorizar el desarrollo de las notificaciones entre usuarios, considerada la funcionalidad primordial para alcanzar un mínimo producto viable que sea atractivo para el usuario final.

Ilustración 7.19: Proceso de publicación de objeto perdido o encontrado: **¿Qué?**.

En el siguiente paso, el usuario deberá rellenar un formulario de datos detallados sobre el objeto que ha perdido o encontrado. La estructura del formulario cambia dependiendo de los opciones elegidas por el usuario. Por ejemplo, en caso de que como categoría se elija *Animal*, el campo de *Marca* se eliminará y se añadirá un campo de *Raza* y *Sexo*. Todo esto hecho de manera dinámica y sutil, con el objetivo de mejorar la experiencia de usuario y evitar que este rellene campos innecesarios.

En cuanto a los archivos que el usuario puede subir, se ha limitado a un máximo de 5 por publicación y un tamaño máximo de 5MB por archivo. Es posible subir imágenes de cualquier tipo, incluidos *.gifs*, pero no se permite adjuntar elementos de cualquier otro tipo.

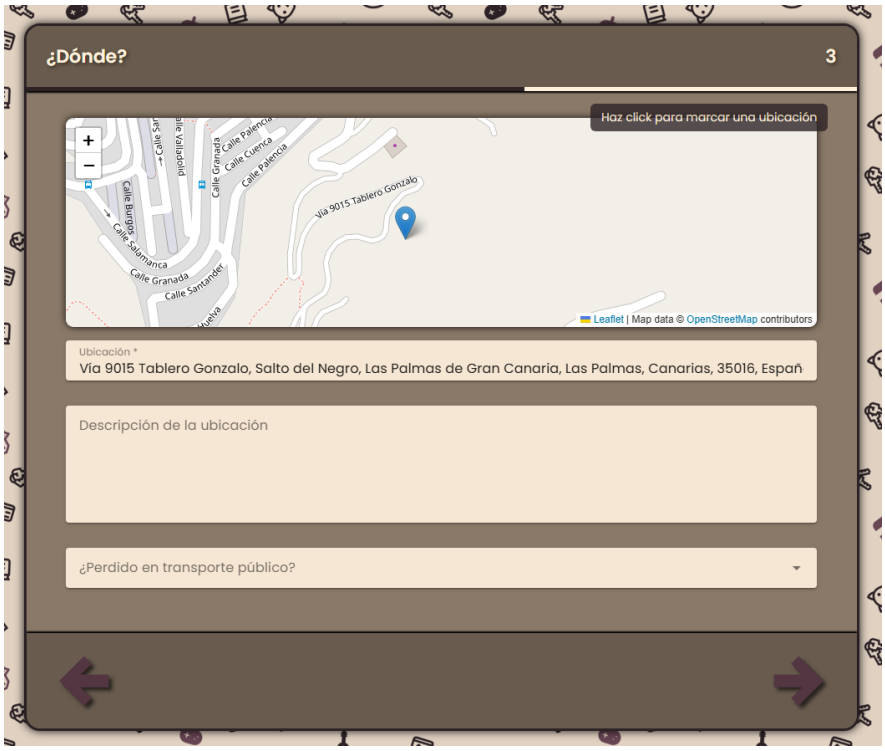

Ilustración 7.20: Proceso de publicación de objeto perdido o encontrado: **¿Dónde?**.

A lo largo de este paso, el usuario deberá especificar la última ubicación aproximada conocida del objeto que ha perdido o el lugar donde se ha encontrado así como una pequeña descripción, si el quisiera, del lugar. También se le ofrece la posibilidad de especificar si se ha perdido/encontrado en un transporte público(taxi, autobús, tranvía, tren, etc), lo que mostraría otro campo adicional opcional para identificar el número de serie del transporte.

Cuando el usuario alcanza el tercer paso, el buscador que usa le notifica de que la página está intentando acceder a su ubicación. En caso de que aceptara, el mapa visible en la Figura 7.20 se centraría en su ubicación actual, rellenando en el proceso el campo *Ubicación* al hacer uso de tecnología de geolocalización inversa. El usuario es libre de usar el mapa a su antojo, en caso de que su posición actual no fuera el lugar donde hubiera perdido o encontrado el objeto en cuestión, y todo se actualizaría automáticamente.

Se ha hecho uso de la librería `Leaflet`, que usa tecnología de `OpenStreetMaps` para facilitar mapas interactuables de prácticamente el mundo entero, y la API de `Nominatim`, que ofrece capacidades de geolocalización inversa. Todo esto como alternativa a `Google Maps`. Se toma esta decisión puesto que las tecnologías descritas, apoyadas en `OpenStreetMaps`, son de código abierto, de gran calidad, con licencias libres y pocas restricciones de uso para proyectos de esta envergadura. Así mismo, se ha creado un nuevo *store* para gestionar las distintas características que permiten que el mapa funcione y mantener así un código limpio y escalable.

En la Figura 7.21 se puede observar un fragmento del *store* creado para gestionar el mapa. En este caso, se observa la función que crea y muestra el propio mapa en el elemento que se le pase. Lo construye usando las coordenadas geográficas especificadas, le asigna el zoom predeterminado, aplica la atribución a `OpenStreetMaps` y inicializa el evento de clic en el mapa.

```
async function initInteractuableMap(containerId: string) {
    resetMap();

    const mapContainer = document.getElementById(containerId);
    if (!mapContainer) return;

    if (!map.value) {
        map.value = L.map(mapContainer, {
            doubleClickZoom: false,
        }).setView([latLng.value[0], latLng.value[1]], 18);

        L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
            attribution: 'Map data © <a href="https://openstreetmap.org">OpenStreetMap</a> contributors',
        }).addTo(map.value as L.Map);

        if (userMarker.value) {
            userMarker.value.setLatLng([latLng.value[0], latLng.value[1]]).addTo(map.value as L.Map);
        }

        map.value.on('click', onMapClick);
    }
}
```

Ilustración 7.21: Función para crear el mapa.

En la Figura 7.22 se observa otro fragmento, este se encarga de localizar la posición del usuario y llamar a la API de `Nominatim` para rellenar el campo de ubicación, haciendo uso de geolocalización inversa.

```
async function locateUser(): Promise<void> {
    return new Promise((resolve, reject) => {
        if (!map.value) return;

        map.value.locate({
            setView: true,
            maxZoom: 18,
            timeout: 10000,
        });

        map.value.once('locationfound', async (e) => {
            try {
                if (!userMarker.value) {
                    userMarker.value = L.marker([e.latlng.lat, e.latlng.lng], { icon: markerIcon }).addTo(map.value as L.Map);
                } else {
                    userMarker.value.setLatLng([e.latlng.lat, e.latlng.lng]);
                }

                (map.value as L.Map).setView([e.latlng.lat, e.latlng.lng], 18);

                latLng.value = [e.latlng.lat, e.latlng.lng];
                location.value = await reverseGeocode(e.latlng.lat, e.latlng.lng);
```

Ilustración 7.22: Función para localizar al usuario.

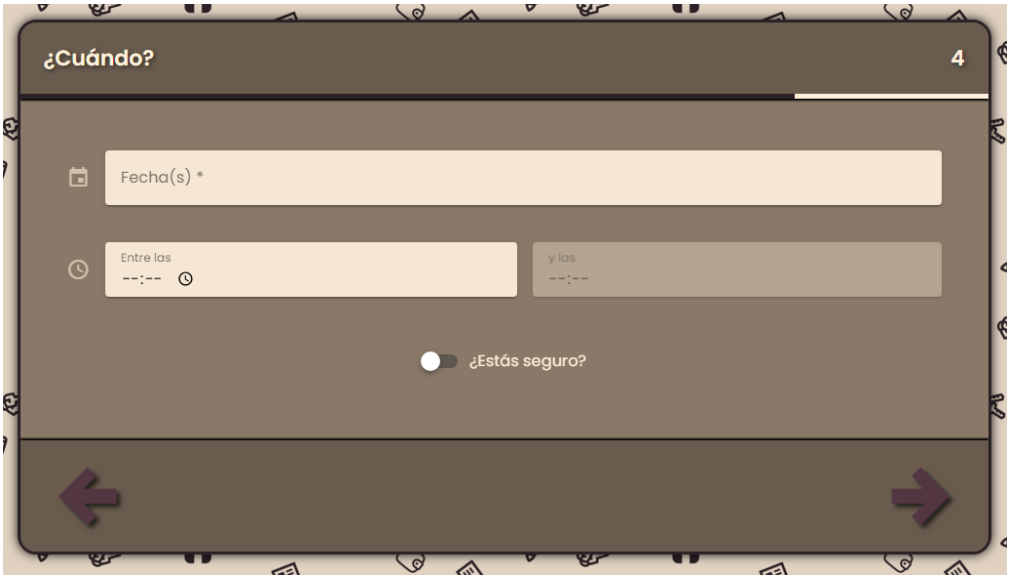

Ilustración 7.23: Proceso de publicación de objeto perdido o encontrado: **¿Cuándo?**.

En el paso número 4, el usuario deberá especificar cuándo ha perdido o encontrado el objeto. Puede seleccionar varias fechas en caso de que no recordara con exactitud el día en el que perdió o encontró el objeto. Además, se le da la opción de elegir un rango de horas, en caso de que no lo sepa de forma exacta, y también se ofrece la posibilidad de no aportar hora ninguna. Por último, el usuario puede especificar si está seguro, o no, de lo que ha rellenado en este paso.

Todo esto se desarrolla teniendo en cuenta que muchas veces los usuarios no recuerdan con claridad el rango temporal en el que han perdido o encontrado un objeto. Se da múltiples opciones al usuario para cubrir este tipo de situaciones, mejorar su experiencia y dar la mayor información posible a los futuros administradores y a las otras personas que navegan por la aplicación.

Después de este formulario, existe un último paso en el que el usuario puede especificar si quiere ofrecer una recompensa por la devolución de su objeto o, en caso de que fuera un objeto encontrado, especificar si se ha entregado en alguna oficina o departamento de objetos perdidos o está aún en su posesión.

Al terminar el proceso, el objeto será registrado en la base de datos y tanto el usuario que ha realizado el procedimiento como el resto de interesados podrán interactuar con la publicación.

Como siguiente punto a desarrollar, se implementa el proceso que permite a un usuario notificar a otro que ha encontrado lo que el ha perdido. Este proceso es similar al de la publicación de un objeto encontrado, pero con algunas diferencias. A continuación se resume el procedimiento:

1. Especificar lugar donde se ha encontrado haciendo uso del mapa de `Leaflet`.

2. Especificar fecha y hora de cuándo se ha encontrado.

3. Especificar si se ha entregado el objeto en alguna oficina o departamento de objetos perdidos o si el usuario que notifica aún lo tiene en su poder.

4. Añadir imágenes del objeto encontrado y un mensaje opcional.

5. Especificar método de contacto en caso de que el usuario que notifica no estuviera registrado o tuviera su cuenta en modo privado.

El primer paso descrito puede ser observado en la Figura 7.24. Cabe destacar que se ha implementado otro *store* en el que se interactúa con la nueva tabla creada para almacenar las notificaciones en la base de datos, siguiendo los mismos principios de programación que en los anteriores *stores*.

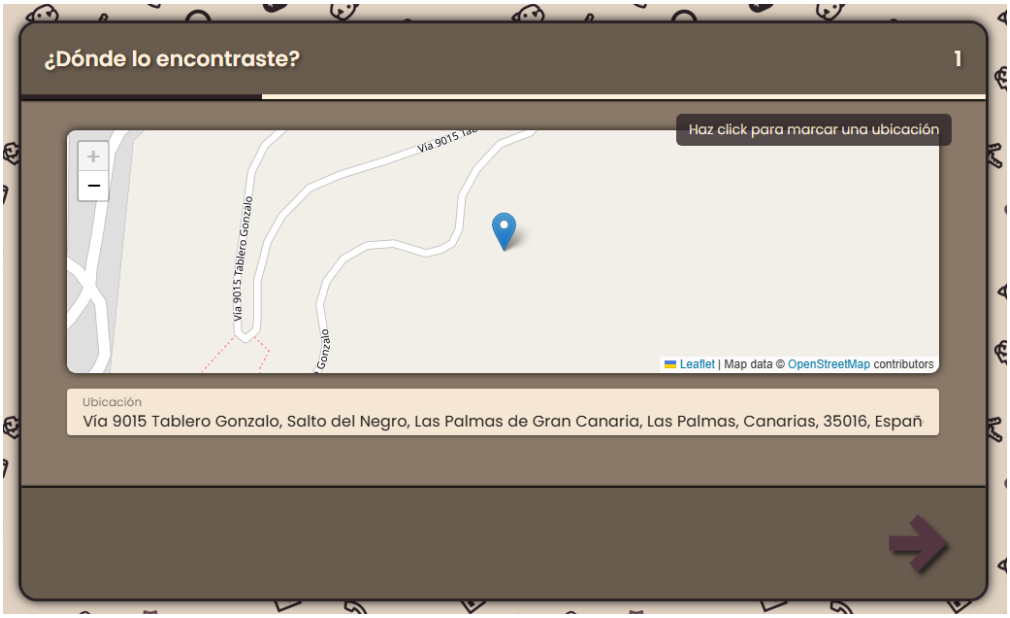

Ilustración 7.24: Proceso de notificación de objeto encontrado: **Inicio**.

A continuación, y como parte final de este incremento, se implementa el sistema de filtros en la página principal de objetos perdidos, visible en la Figura 7.25. Ahora existe un botón con un icono de lupa en la parte superior derecha de la página que permite al usuario abrir un menú con distintas opciones de filtrado. Este botón se puede ver en la Figura 7.26.

Ilustración 7.25: Sistema de filtros en la página principal de objetos perdidos.

Todos los filtros mostrados en la figura anterior son funcionales y permiten al usuario filtrar los objetos disponible en la lista inferior como el prefiera. La vista se actualiza en tiempo real a medida que el usuario va seleccionado opciones de filtrado. El componente de filtros es dinámico, de manera que, por ejemplo, si el usuario selecciona categoría entonces podrá seleccionar subcategoría. Esto con el objetivo de mejorar la experiencia del usuario y evitar que el menú de filtros ocupe demasiado espacio en la pantalla cuando no sea necesario.

Los filtros pueden cerrarse con el botón de la parte superior derecha, visible en la Figura 7.25, y al hacerlo las opciones de filtrado se mantienen.

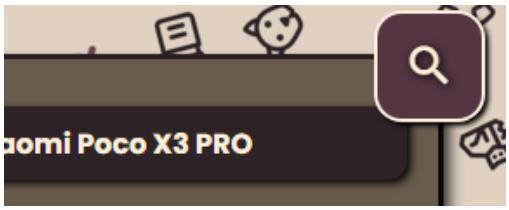

Ilustración 7.26: Botón para abrir filtros en la página principal de objetos perdidos.

En caso de que existan filtros aplicados, el botón visible en la Figura 7.26 cambiará al visible en la Figura 7.27.

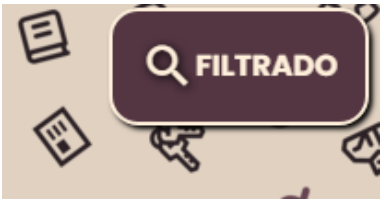

Ilustración 7.27: Botón cuando hay filtros activados.

7.3.2. Iteración

A lo largo de este incremento se ha realizado una iteración sobre lo desarrollado en el anterior. En este caso, los aspectos nuevos más relevantes que aún no se han comentado son los siguientes:

- Creado un nuevo *store* para gestionar la carga de la página. Ahora el componente de la superposición de carga solo aparece cuando hay que cargar datos y no como antes, que era al navegar a ciertos componentes.

- Modificación de la tarjeta de objeto perdido. Ahora muestra menos información, solo la esencial, con el objetivo de mejorar la experiencia de usuario y la usabilidad de la aplicación. Cambio visible en la Figura 7.28.

- Modificación de la barra de navegación. Se han añadido nuevos botones y se han modificado algunos de los existentes.

- Modificación de la página principal de objetos perdidos. Al modificar la tarjeta de objeto perdido, se decide transformar la vista de la página principal para que se puedan ver más tarjetas si la resolución del dispositivo lo permite. Cambio visible en la Figura 7.30.

- Modificación del registro de usuario. El estilo del proceso se ha actualizado para que estuviera más acorde con el resto de procedimientos de la aplicación. Cambio visible en la Figura 7.29. Añadida la capacidad de hacer el perfil privado.

- Modificación de la página de detalles del objeto. Ahora, si el objeto es de la propiedad del usuario que tiene la sesión iniciada, podrá ser borrado con un botón disponible en la parte inferior de la página. Añadido un icono al lado del campo de ubicación que permite al usuario abrir el mapa de `Leaflet` y ver la ubicación de manera precisa. Cambio visible en la Figura 7.31.

- Ahora la información de perfil de un usuario que tiene su cuenta privada no es visible para el resto de usuarios.

- Posibilidad de cerrar sesión desde la barra de navegación.

- Modificaciones menores de estilo y funcionalidad en el resto los componentes.

Ilustración 7.28: Tarjeta de objeto perdido. Segunda versión.

Ilustración 7.29: Registro de usuario. Segunda versión.

Ilustración 7.30: Página principal de objetos perdidos. Segunda versión.

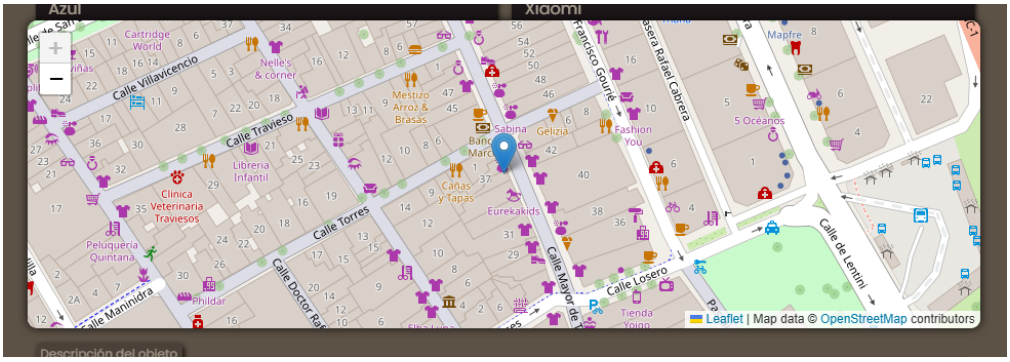

Ilustración 7.31: Página de detalles de objeto. Mapa de ubicación.

7.4. Diseño e implementación de sistema de notificación entre usuarios

7.4.1. Incremento

Para este cuarto incremento se ha desarrollado las siguientes funcionalidades:

- Componente de notificaciones entrantes.
- Componente de notificación.
- Interactuar con las notificaciones.

Al comenzar este incremento, se desarrolla la nueva sección de notificaciones, visible en la figura 7.32.

Ilustración 7.32: Sección de notificaciones

Este componente permite que el usuario vea una lista de notificaciones entrantes, ordenadas de más recientes a más antiguas, relacionadas con los objetos que ha publicado. Estas tarjetas de notificaciones portan poca información para no invadir el espacio disponible en la pantalla, sin embargo, si el usuario así lo desea, puede ver los detalles de la notificación al hacer clic en el botón pertinente. En la Figura 7.33 se puede observar la información detallada de una notificación de ejemplo. Como dato secundario, hacer clic en el botón *Objeto asociado*, redirige al usuario a los detalles del objeto perdido en cuestión.

Ilustración 7.33: Detalles de la notificación

En esta parte del desarrollo se ha decidido quitar la posibilidad de enviar imágenes en las notificaciones. Existen unas cuantas razones por las que se ha tomado esta decisión:

1. Incluso usuarios no registrados son capaces de enviar una notificación de objeto encontrado a otro usuario. Este es el único lugar de la aplicación que permite a estos interactuar con la base datos. Por lo que se considera apropiado que no se pueda enviar imágenes para evitar problemas de seguridad y spam.

2. Se prevé que haya más notificaciones que publicaciones de objetos perdidos, por lo que si cada notificación llevara una imagen asociada, el número de imágenes que se tendrían que almacenar en la base de datos sería muy elevado.

3. Se considera poco práctico o no primordial, ya que las notificaciones vienen con un número y/o un email de contacto, por lo que el dueño del objeto puede contactar con el usuario para pedirle una imagen del objeto, en caso de que estuviera en sus manos aún.

De igual manera la opción no se descarta del todo ya que, como se mencionará en el apartado de trabajo futuro, se piensa implementar un sistema *CAPTCHA* y confirmación de email y teléfono para evitar el spam, la suplantación de identidad y problemas de seguridad derivados de ello.

Continuando con el componente, el usuario puede eliminar la notificación con el botón *Este objeto no es mio* en el caso de que haya identificado con la información vista o tras hablar con el usuario que se ha comunicado con él, que el objeto que ha encontrado, no es el suyo. En este caso, la notificación se eliminará de la base de datos. Al hacer clic en el botón, se muestra una pestaña de confirmación, visible en la Figura 7.34.

Por otro lado, el usuario puede confirmar que el objeto que se le ha notificado es efectivamente suyo y ha sido recuperado, al hacer click en el botón *He recuperado este objeto*. En este otro caso, tanto las notificaciones asociadas a ese objeto perdido, como el propio objeto extraviado, se elimina de la base de datos. Al hacer clic en el botón, se muestra una pestaña de confirmación, visible en la Figura 7.35.

Estas características han sido desarrolladas teniendo como objetivo mejorar la usabilidad de este componente y la experiencia de usuario.

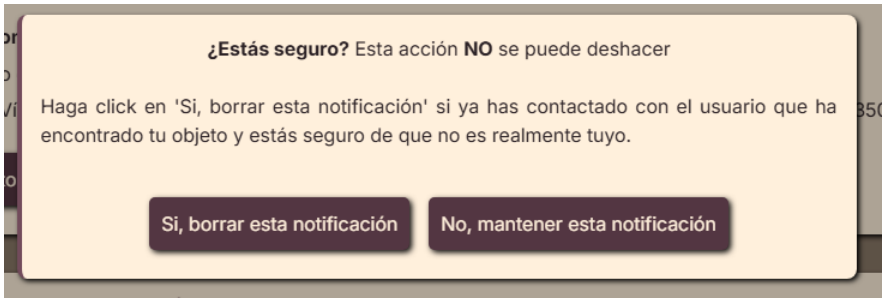

Ilustración 7.34: Objeto no es del usuario

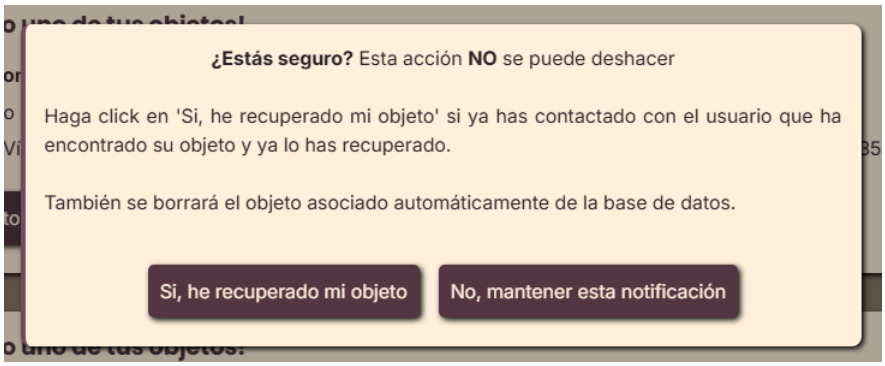

Ilustración 7.35: Objeto es el del usuario

Como colofón del apartado visual de esta sección, en la Figura 7.36 se observa que también es posible ver de manera detallada el lugar en el que el emisor de la notificación dice haber encontrado el objeto, tras hacer clic en el icono junto a la ubicación.

Ilustración 7.36: Mapa de notificación

Cabe mencionar que se ha actualizado el *store* de notificaciones creado en el anterior incremento, con el fin de que la sección de notificaciones esté suscrita a una tabla *real-time* de la base de datos. De esta manera, el componente se actualiza automáticamente cada vez que se añade o elimina una notificación. Esto permite que la aplicación sepa en todo momento si el usuario ha recibido una notificación nueva o no, lo que mejora la experiencia de usuario y la escalabilidad de esta parte del portal web. Se logra en gran parte gracias a la función disponible en la Figura 7.37.

```
function listenToRealtime(): void {
    const channel = supabase
        .channel('public:item_notifications')
            .on('postgres_changes', {
                event: 'INSERT',
                schema: 'public',
                table: 'item_notifications',
            }, (payload) => {
                const newNotification = mapToItemNotification(payload.new)
                if (newNotification.receiverID === user.value?.id) {
                    userNotifications.value.unshift(newNotification)
                }
            })
            .on('postgres_changes', {
                event: 'DELETE',
                schema: 'public',
                table: 'item_notifications',
            }, (payload) => {
                userNotifications.value = userNotifications.value.filter(n => n.id !== payload.old.
                id)
        })

        channel.subscribe()
}
```

Ilustración 7.37: Función para usar *real-time* de `Supabase`.

7.4.2. Iteración

A lo largo de esta incremento se ha iterado sobre las anteriores partes del proyecto. En este caso, los aspectos nuevos más relevantes que aún no se han comentado son los siguientes:

- Modificación de filtros de la página principal de objetos perdidos.
- Modificación de detalles de los objetos. Ahora el usuario no puede notificarse a sí mismo ni guardar su propio objeto.
- Modificación de la tarjeta de objeto perdido. Si el usuario no asigna recompensa a su objeto, se muestra la fecha y la hora en su lugar.
- Modificación de la barra de navegación. Añadida funcionalidad al botón de notificaciones y cambiado el estilo del botón de cerrar sesión.
- Otras modificaciones de estilo y funcionalidad menores en la aplicación.

Cabe mencionar con mayor detenimiento las modificaciones realizadas a los filtros de la página principal de objetos perdidos. Los cambios son visibles en la Figura 7.38 y consisten en lo siguiente:

- Ahora es posible filtrar objetos por proximidad al usuario o a una ubicación concreta. No sustituye al anterior filtro de ubicación, que se mantiene como otra opción. Cambio visible en la Figura 7.39.
- Ahora es posible filtrar objetos por recompensa mínima, máxima y exacta.
- Ahora es posible filtrar por fecha máxima de pérdida. No sustituye el anterior filtro de fecha de publicación, que se mantiene como otra opción.

Ilustración 7.38: Filtros finales de la página principal de objetos perdidos

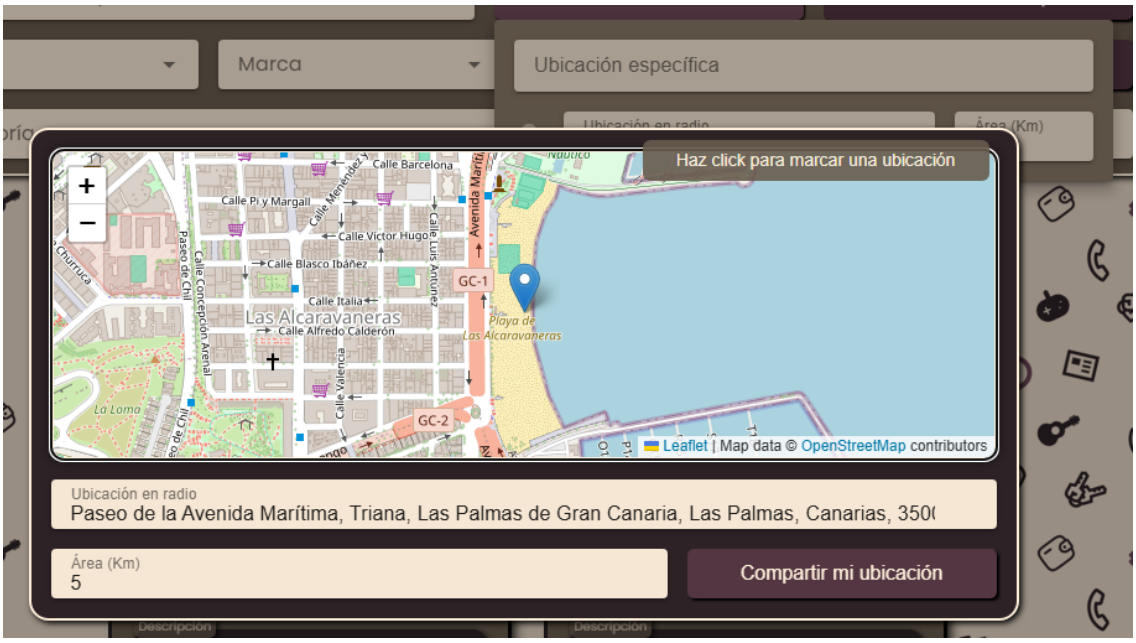

Ilustración 7.39: Filtro de ubicación por proximidad

Mejora fundamental para el sistema de filtros que ahora permite que los usuarios puedan buscar objetos que hayan sido extraviados cerca de su ubicación, o comprobar, en el caso de que hayan encontrado algo, si alguien ha publicado un objeto perdido cerca de su posición, entre otras usos.

Estos filtros son completamente funcionales y han sido probados con detenimiento en diferentes buscadores y dispositivos.

Para lograr esta funcionalidad se ha utilizado la Fórmula del semiverseno o Fórmula de Harversine [30], que permite calcular la distancia entre dos puntos en la superficie de una esfera, en este caso, la Tierra. Esta fórmula es especialmente útil para este caso y aporta resultados de gran precisión.

En la Figura 7.40 se puede observar el método construido.

```
You, last week | 1 author (You)
// Haversine formula to calculate the distance between two points on the Earth
export function isInRadius(lat1: number, lon1: number, lat2: number, lon2: number, radiusKm: number): boolean {
    const R = 6371;
    const dLat = (lat2 - lat1) * Math.PI / 180;
    const dLon = (lon2 - lon1) * Math.PI / 180;
    const a =
        Math.sin(dLat / 2) ** 2 +
        Math.cos(lat1 * Math.PI / 180) * Math.cos(lat2 * Math.PI / 180) *
        Math.sin(dLon / 2) ** 2;
    const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
    const d = R * c;
    return d <= radiusKm;
}       You, last week via  PR #4 • feature: notification component finished …
```

Ilustración 7.40: Método de la fórmula de Harversine

Mientras que la fórmula es la siguiente:

Fórmula de Haversine

$$a = \sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\Delta\lambda}{2}\right)$$

$$c = 2 \cdot \arctan 2\left(\sqrt{a}, \sqrt{1-a}\right)$$

$$d = R \cdot c$$

Donde:

- d es la distancia entre los dos puntos.
- R es el radio de la Tierra (aproximadamente 6371 km).
- ϕ_1 y ϕ_2 son las latitudes de los dos puntos en radianes.
- $\Delta\phi$ es la diferencia entre las latitudes de los dos puntos en radianes.
- $\Delta\lambda$ es la diferencia entre las longitudes de los dos puntos en radianes.

Al finalizar el cálculo en la función 7.40 se devuelve el resultado de una comparación con el radio en el que el usuario quiere delimitar las coincidencias, con un valor predeterminado de 5 kilómetros.

7.5. Pruebas realizadas

A lo largo de todos los incrementos del proyecto, se han realizado diversas pruebas para asegurar el correcto funcionamiento de las características implementadas.

Se han hecho uso de los siguientes navegadores:

- Google Chrome
- Brave
- DuckDuckGo
- Microsoft Edge

En los siguientes dispositivos:

- Ordenador con Windows 11 con pantalla de 27 pulgadas.
- Ordenador con Windows 11 con pantalla de 21 pulgadas.
- Ordenador portátil con Windows 10 con pantalla de 15 pulgadas.
- Tablet android con pantalla de 10 pulgadas.
- Teléfono móvil android con pantalla de 6 pulgadas.

La aplicación ha sido diseñada y probada con un rango horizontal de resoluciones de pantalla de entre *375* píxeles a *2560* píxeles y un tamaño vertical de mínimo *530* píxeles. Esto asegura que la aplicación se visualice correctamente en la mayoría de los dispositivos modernos, incluyendo teléfonos móviles, tablets y ordenadores de diferente tipo.

Al finalizar el desarrollo de la aplicación, se ha buscado la ayuda de diferentes usuarios para realizar pruebas de usabilidad. Se ha solicitado la colaboración de usuarios con diferentes niveles de experiencia:

- Usuarios con experiencia en el uso de aplicaciones web.
- Usuarios sin experiencia en el uso de aplicaciones web.
- Usuario con experiencia en la gestión de objetos perdidos en el ámbito público.

Las pruebas consistieron en completar diferentes tareas dentro de la aplicación con la meta de usar todas las funcionalidades disponibles. Tras la realización de los tests, se recogieron las opiniones de los usuarios involucrados, y aunque fue bien recibido por los usuarios con experiencia, los no experimentados necesitaron ayuda o más tiempo para completar las tareas. Por otro lado, el usuario con experiencia en la gestión de objetos perdidos, un funcionario jubilado de la Policía Local, pudo hacer uso de la aplicación y tuvo una opinión positiva sobre la misma, aunque destacó el hecho de que la aplicación aún no posee el rol de administrador y todo lo asociado al mismo.

Tras las pruebas, se llega a la conclusión que como trabajo futuro se deberá priorizar la implementación del rol de administrador y la creación de un manual de usuario o tutorial dentro de la propia web para guiar a las personas con menos experiencia en el uso del portal.

Capítulo 8

Resultados, conclusiones y trabajo futuro

En este último capítulo, se comentará el resultado final del trabajo realizado, las conclusiones obtenidas y el trabajo futuro planeado.

8.1. Resultados

Se ha respetado y seguido una metodología iterativa/incremental, dando como resultado un portal web que cubre los casos de uso descritos, a excepción de lo relacionado con el rol de administrador, y ofrece nuevas funcionalidades no previstas.

Un usuario no registrado puede realizar las siguientes acciones:

- Ver página de inicio.
- Ver página de *Quiénes somos*
- Ver lista de objetos perdidos publicados en el portal web.
- Usar sistema de filtros avanzados. Filtrar objetos por cercanía a una ubicación especificada en el radio deseado.
- Ver información de perfil de otros usuarios.
- Notificar a otro usuario que ha encontrado su objeto perdido.

Cuando un usuario se registra también puede realizar las siguientes acciones:

- Iniciar de sesión.
- Ver y modificar datos personales.
- Ver lista de objetos perdidos de su propiedad.
- Publicar un objeto perdido/encontrado.
- Borrar publicación propia de objeto perdido/encontrado.
- Ver y responder a notificaciones de coincidencias.

A continuación, se ilustra en la Figura 8.1 la versión más reciente de las tablas de la base de datos de `Supabase`.

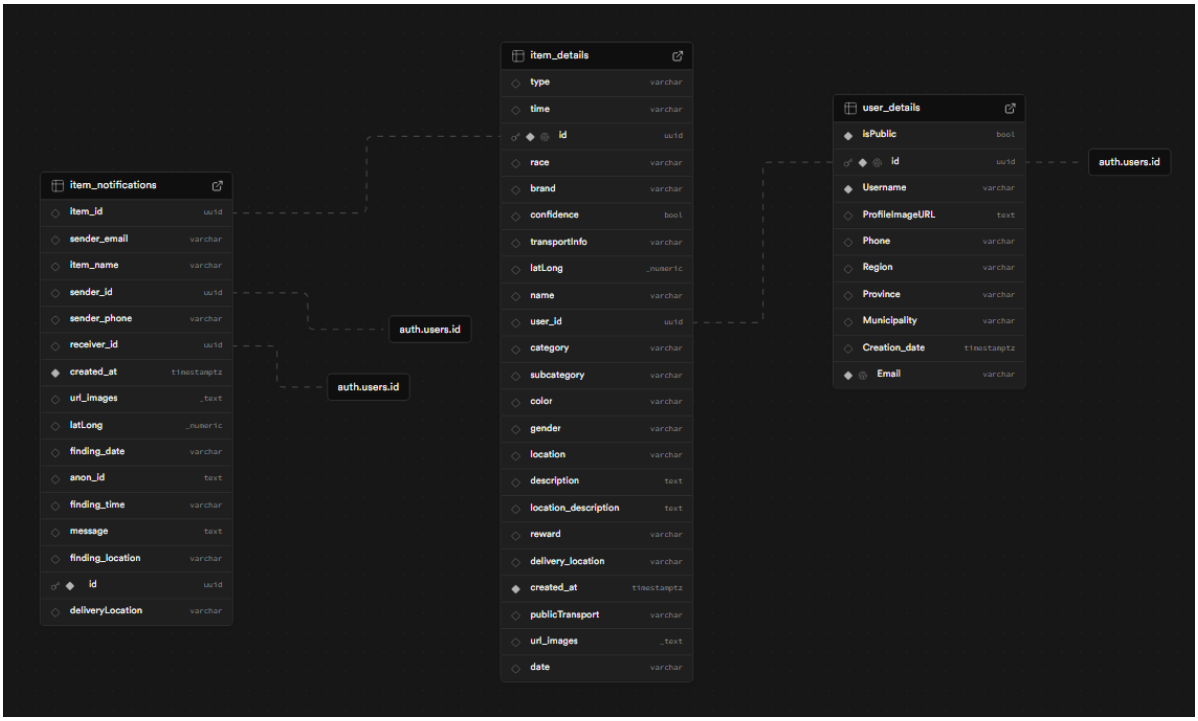

Ilustración 8.1: Tablas de la base de datos de Supabase.

Se han hecho uso de los servicios de `Vercel` para poner en producción este proyecto. Se ha optado por este servicio por su capacidad de desplegar aplicaciones web de forma gratuita y por su integración con `GitHub`, lo que permite una gestión eficiente de las ramas de desarrollo y producción.

A fecha de redacción de esta memoria es posible acceder al portal web bajo la url https://lost-hub.vercel.app/. Se recomienda su visita si se desea ver en detalle los componentes, comprobar la funcionalidad de la herramienta, observar las características menores no mencionadas, así como observar las variaciones de diseño *responsive* que se producen al utilizar distintas resoluciones.

Como colofón de este capítulo, cabe mencionar que se ha alcanzado un resultado que, aunque no es el inicialmente previsto, ha sido satisfactorio en cumplir las expectativas puestas sobre el proyecto. El portal web es funcional y se considera una herramienta útil para la búsqueda y gestión de objetos perdidos.

8.2. Conclusiones

Cuándo se realizó el análisis del estado del arte, se determinó que la mayoría de alternativas existentes se limitan a gestionar de forma interna las publicaciones de objetos enviadas, pero no incitan a los usuarios a cooperar para que dichos objetos vuelvan a las manos de sus propietarios, puesto que no permiten interacción con los objetos extraviados publicados. Otras herramientas si ofrecen mayor interacción, pero se limitan a mostrar publicaciones de mascotas perdidas. Por tanto, cuando se comenzó el diseño de la funcionalidad de la aplicación web tras el previo análisis del estado del arte, se fijó como meta una herramienta avanzada de gestión de objetos perdidos que permitiera mayor interacción entre usuarios y los propios objetos extraviados, con el objetivo de crear una comunidad de personas que activamente colaboren e inciten a otros a hacerlo también.

A lo largo del desarrollo de esta aplicación web se ha tenido en cuenta mejorar y/o mantener la escalabilidad de la misma. Esto se ha logrado al encapsular la lógica de negocio dentro de los *stores* de *Pinia* y con la organización de carpetas basada en el sistema de vista, contenedor y componente descrita con anterioridad. Igualmente, la implementación de una base de datos de `Supabase` permite el transporte de la misma a otros servicios en la nube o a servidores propios. Así mismo, se implementa todos los componentes visuales además del diseño completamente *responsive* haciendo uso de librerías de estilo de distinto tipo, con el objetivo final de crear una herramienta que aporte gran usabilidad, accesibilidad y una buena experiencia de usuario. Finalmente, se ha utilizado el lenguaje inglés de forma interna, con la meta de que el código fuente esté al alcance de mayor cantidad de personas, lo que también contribuye a la escalabilidad del proyecto.

Gracias en parte a todo esto, la aplicación web es completamente modular y permite que se pueda cambiar la base de datos utilizada con poco trabajo. Por lo que si se diese el caso de obtener oportunidades para cooperar con empresas externas que ya poseen grandes cantidades de registros de objetos perdidos, y que a su vez necesiten una herramienta de gestión, es posible adaptar la aplicación a sus necesidades con una carga laboral baja, y por tanto, a bajo costo. Incluso se podría cooperar con más de una si se diese el caso.

Al finalizar el desarrollo del portal web, se ha conseguido una herramienta que, aunque no se alinea con todos los casos de uso inicialmente previstos, si que cumple con las expectativas definidas en el análisis previo a su realización. La herramienta web permite a los usuarios publicar objetos perdidos y encontrados, notificar coincidencias y gestionar las propias. Además, ofrece un sistema de filtros avanzados que permite al usuario, entre otras funcionalidades, filtrar objetos perdidos por proximidad.

8.3. Trabajo futuro

Como trabajo futuro se prioriza la implementación de lo dejado incompleto en el desarrollo de la aplicación web y, después de ello, la adición de nuevas características que amplíen las capacidades de la herramienta. Se especifican a continuación:

1. Implementar el rol de administrador.
2. Ofrecer más posibilidades a la hora de publicar un objeto.
3. Mejorar la seguridad de la web.
4. Terminar sistema de recompensas.
5. Crear aplicación móvil asociada.
6. Implementar tecnología de automatización basada en IA.
7. Añadir anuncios de terceros y ampliar servicios.
8. Ampliar público objetivo.
9. Otras modificaciones y adiciones menores.

8.3.1. Implementar el rol de administrador

El rol de administrador es el siguiente gran paso en el desarrollo de la aplicación web. Este rol tendrá como objetivo gestionar a sus usuarios y a los objetos disponibles en la plataforma. Para la correcta implementación de este rol se deberá desarrollar lo necesario para permitir la gestión del portal web.

Lo primero a realizar sería un componente donde se especificará las reglas de conducta a seguir para el uso de la aplicación. Luego, se facilitará herramientas para eliminar objetos perdidos y encontrados de la aplicación que no cumplan con las normas de uso, así como vetar a usuarios que inflijan los códigos de conducta establecida. Igualmente, se creará un nuevo componente basado en los anteriores que proporcione los medios para navegar por una lista tanto de objetos perdidos como encontrados. Esto con un sistema de filtros más avanzado exclusivo para este rol que permita filtrar por usuario, tipo de objeto, fecha de publicación, etc. Después, se implementará un sistema de coincidencias con el fin de que el administrador pueda relacionar un objeto perdido con uno encontrado, enviando una notificación a ambos usuarios para que puedan ponerse en contacto.

El sistema de notificaciones será ampliado, con la meta de que los administradores puedan enviar alertas de distinto tipo a los usuarios de la plataforma.

8.3.2. Ofrecer más posibilidades a la hora de publicar un objeto

La carencia principal de la aplicación en cuanto a la publicación de objetos es la incapacidad de dejar constancia de manera precisa de los objetos que están dentro de contenedores, como bolsos, carteras, mochilas, etc. Para ello, se implementará un sistema de subobjetos, que permita a los usuarios añadir subobjetos dentro de los objetos. De esta manera, se podrá especificar que un bolso contiene unas llaves y un teléfono móvil, por ejemplo. Esto se hace con la meta de que, en el caso de que se pierda un bolso, por ejemplo, y alguien encontrara las llaves de dentro de ese bolso, hubiera conexión entre ambos elementos y se pudiera notificar al usuario que perdió el objeto en cuestión.

Se deberá de crear y asociar una nueva tabla en la base de datos para almacenar esta información. Además, los filtros de búsqueda se ampliarán para tener en cuenta esta nueva funcionalidad, permitiendo buscar objetos por sus subobjetos, por ejemplo.

Por otro lado, se añadirá la posibilidad de modificar las publicaciones ya subidas en la base de datos. Por si hubieran erratas o actualizaciones que hacer sobre el estado del objeto.

8.3.3. Mejorar la seguridad de la web

La seguridad de la aplicación web es un aspecto fundamental que debe ser mejorado. Para ello, y en primer lugar, se implementará la API reCAPTCHA de Google [31]. Esto se hará en los formularios de registro, inicio de sesión, publicación de objetos y notificación de objeto encontrado. Además, en los mismos componentes, se implementará un sistema de validación de email y número de teléfono, exigiendo al usuario un código de verificación cada vez que introduzca o actualice su información de contacto. Todo esto con la meta de evitar el spam, los bots, el uso de cuentas falsas y la suplantación de identidad.

De forma secundaria, se añadirá un sistema de autentificación de dos factores (2FA) para aumentar la seguridad de las cuentas de usuario.

8.3.4. Terminar sistema de recompensas

Aunque la idea básica de sistema de recompensas ya está implementada, aún queda por desarrollar los componentes relacionados con la pasarela de pago, así como la infraestructura necesaria para gestionar las transacciones y el sistema de recompensas en sí. Se propone que, para evitar malentendidos o el mal uso de esta característica de la aplicación, el usuario deba enviar, bajo contrato y siguiendo las pautas legales necesarias, el dinero que ofrece como recompensa a una cuenta segura gestionada por los administradores. El dinero es retenido, y solo cuando esa recompensa sea reclamada por el usuario que encontró el objeto, o cuando el objeto es eliminado de la base de datos por otras razones, que se realiza la transacción requerida.

Adicionalmente, y especificado en las futuras reglas de uso, un porcentaje todavía no planteado del dinero de la recompensa será destinado a la plataforma. Para cubrir los datos de mantenimiento de los servicios y la infraestructura de la aplicación web.

8.3.5. Crear aplicación móvil asociada

Aunque la web está diseñada para adaptarse a dispositivos móviles, no posee las comodidades de una aplicación nativa. Por lo que se propone el desarrollo de una aplicación móvil asociada a la web, que permita a los usuarios acceder a las funcionalidades de la plataforma desde sus dispositivos móviles de forma más cómoda y rápida. Para ello, se deberá hacer un previo análisis de las tecnologías disponibles, y usar la más adecuada.

8.3.6. Implementar tecnología de automatización basada en IA

Con el rol de administrador implementado, y la capacidad de gestionar coincidencias entre objetos perdidos y encontrados de manera manual, se propone buscar una alternativa basada en inteligencia artificial para automatizar el proceso y, si llegara el caso, poder prescindir de la intervención por parte de los administradores en este aspecto.

8.3.7. Añadir anuncios de terceros y ampliar servicios

En el caso de que la aplicación web se consolide, y tras un análisis de la situación, se propone ampliar lo relacionado con producción y la base de datos, pagando por los servicios premium de `Supabase` y `vercel`, o buscando otras alternativas que ofrezcan mejores o más adecuadas características.

Para ayudar a costear estos servicios, ahora de pago, se añadiría un sistema de anuncios de terceros, así como la creación del rol usuario premium. Este usuario, junto con los administradores, tendrán la posibilidad de ver la web sin anuncios. No se descarta añadir otras nuevas funcionalidades exclusivas para el nuevo rol.

8.3.8. Ampliar público objetivo

Con el fin de ampliar el público objetivo, se propone traducir la aplicación web a otros idiomas, comenzando por el inglés. Además, se deberá adaptar a los países en cuestión, desde la implementación de las localizaciones nuevas hasta las normativas y legislaciones pertinentes.

8.3.9. Otras modificaciones y adiciones menores

Con la meta final de mejorar de forma continua la usabilidad y la experiencia de usuario, se realizarán otras modificaciones y adiciones menores, relacionadas en gran medida con el estilo de la aplicación o funcionalidades no primordiales. Algunos ejemplos de estas modificaciones son:

- Implementar el sistema de *favoritos*/publicaciones guardadas comentado en anteriores capítulos.
- Ejecutar sonido cuando el usuario recibe notificación.
- Mostrar el número de notificaciones sin leer en la barra de navegación.
- Diferenciar entre notificaciones leídas y no leídas.
- Añadir un sistema de reportes de usuario.
- Ampliar la documentación del proyecto.
- Crear tutoriales o manual de usuario.
- Investigar la posibilidad de añadir entrega a domicilio.

Bibliografía

[1] La Razón, "Lo que Madrid esconde en su Oficina de objetos perdidos: desde cunas hasta décimos de lotería." https://www.larazon.es/madrid/que-madrid-esconde-oficina-objetos-perdidos-cunas-decimos-loteria_20250517682844e457a5aa2cde264b4a.html. Accedido: 20-may-2025.

[2] FindMyLost, "FindMyLost - Digital Lost & Found." https://www.linkedin.com/company/findmylost/. Accedido: 25-mar-2025.

[3] FindMyLost, "FindMyLost – Lost & Found Digital." https://www.findmylost.it/es. Accedido: 25-mar-2025.

[4] Foundspot, "Foundspot – Perfil de empresa." https://www.linkedin.com/company/foundspot/. Accedido: 25-mar-2025.

[5] Foundspot, "Foundspot – Encuentra objetos perdidos." https://www.foundspot.com/es/. Accedido: 25-mar-2025.

[6] Missing Pets, "Missing Pets – Encuentra tu mascota perdida (Google Play)." https://play.google.com/store/apps/details?id=com.pet.missingpets&hl=es&pli=1. Accedido: 25-mar-2025.

[7] Missing Pets, "Missing Pets – Encuentra tu mascota perdida (App Store)." https://apps.apple.com/us/app/missing-pets-find-lost-pet/id1475782956. Accedido: 25-mar-2025.

[8] Objetos Perdidos 2.0, "Objetos Perdidos 2.0 – Información." https://www.facebook.com/objetosperdidos2.0/info. Accedido: 25-mar-2025.

[9] Policía Local de Las Palmas de Gran Canaria, "Objetos Perdidos." http://policialpa.es/es/servicios/objetos-perdidos/. Accedido: 25-mar-2025.

[10] Vue.js, "Vue.js - The Progressive JavaScript Framework." https://vuejs.org/. Accedido: 25-mar-2025.

[11] OpenStreetMap contributors, "OpenStreetMap." https://www.openstreetmap.org/. Accedido: 16-may-2025.

[12] Leaflet, "Leaflet – An open-source JavaScript library for mobile-friendly interactive maps." https://leafletjs.com/. Accedido: 16-may-2025.

[13] Nominatim, "Nominatim – Search engine for OpenStreetMap data." https://nominatim.org/. Accedido: 16-may-2025.

[14] Pinia, "Pinia – The Vue Store that you will enjoy using." https://pinia.vuejs.org/. Accedido: 25-mar-2025.

[15] Supabase, "Supabase – The Open Source Firebase Alternative." https://supabase.com/. Accedido: 25-mar-2025.

[16] Figma, "Figma – The collaborative interface design tool." https://www.figma.com/. Accedido: 25-mar-2025.

[17] GitHub, "GitHub – Where the world builds software." https://github.com/. Accedido: 25-mar-2025.

[18] Microsoft, "Visual Studio Code - Code Editing. Redefined.." https://code.visualstudio.com/. Accedido: 25-mar-2025.

[19] npm, "npm – a JavaScript package manager." https://www.npmjs.com/. Accedido: 20-may-2025.

[20] Coolors, "Coolors – The super fast color palettes generator!." https://coolors.co/. Accedido: 26-mar-2025.

[21] Canva, "Canva – Herramienta de diseño gráfico en línea." https://www.canva.com/. Accedido: 20-may-2025.

[22] Google Fonts, "Inter – Google Fonts." https://fonts.google.com/specimen/Inter?query=inter. Accedido: 28-mar-2025.

[23] Google Fonts, "Poppins – Google Fonts." https://fonts.google.com/specimen/Poppins?query=poppins. Accedido: 28-mar-2025.

[24] Vercel, "Vercel – develop. preview. ship.." https://vercel.com/. Accedido: 19-may-2025.

[25] Vee-Validate, "Vee-Validate – The best validation library for Vue.js." https://vee-validate.logaretm.com/. Accedido: 25-mar-2025.

[26] Vuetify, "Vuetify – Material Design Framework for Vue.js." https://vuetifyjs.com/. Accedido: 25-mar-2025.

[27] Draw.io, "draw.io – Online diagramming tool." https://www.draw.io/. Accedido: 26-mar-2025.

[28] Responsively App, "Responsively – A modified browser for responsive web development." https://responsively.app/. Accedido: 16-may-2025.

[29] Open Font License, "How to Use OFL Fonts." https://openfontlicense.org/how-to-use-ofl-fonts/. Accedido: 28-mar-2025.

[30] Wikipedia, "Fórmula del semiverseno – Wikipedia, la enciclopedia libre." https://es.wikipedia.org/wiki/F%C3%B3rmula_del_semiverseno. Accedido: 20-may-2025.

[31] Google, "reCAPTCHA – Protección contra bots y abuso." https://www.google.com/recaptcha/about/. Accedido: 23-may-2025.

[32] Google LLC, "Google Maps – Servicio de mapas en línea." https://maps.google.com/. Accedido: 20-may-2025.