UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA
CAMPUS D'ALCOI

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Politécnica Superior de Alcoy

Aplicación multiplataforma para geolocalización de especies invasoras

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Rodriguez Santiago, Adrian

Tutor/a: Pérez Llorens, Rubén

CURSO ACADÉMICO: 2021/2022

Resumen

Se trata de una aplicación que permitirá a los usuarios fotografiar, comentar y geolocalizar especies invasoras, u otras incidencias ecológicas, para transmitir esta información a un servidor que será consultable por los administradores para poder tomar las medidas oportunas.

La aplicación constará entonces de una App Multiplataforma para la recopilación de la información, de un servidor con una BBDD para almacenarla, y de un *front-end* web para poder consultar esta información almacenada.

El proyecto se desarrollará basándose en una propuesta de la Colla Ecologista La Carrasca.

Palabras clave

aplicación; multiplataforma; *web app*; xamarin; base de datos; especies invasoras; geolocalización; ecología

Resum

Es tracta d'una aplicació que permetrà als usuaris fotografiar, comentar i geolocalitzar espècies invasores o altres incidències ecològiques per transmetre aquesta informació a un servidor que serà consultable pels administradors per poder prendre les mesures oportunes.

L'aplicació constarà llavors d'una App Multiplataforma per a la recopilació de la informació, d'un servidor amb una BBDD per emmagatzemar-la i d'un *front-end* web per poder consultar aquesta informació emmagatzemada.

El projecte es desenvoluparà basant-se en una proposta de la Colla Ecologista La Carrasca.

Paraules clau

aplicació; multiplataforma; *web app*; xamarin; base de dades; espècies invasores; geolocalització; ecologia

Abstract

This is an application that will allow users to photograph, comment and geolocate invasive species, or other ecological incidents, to send this information to a server that will be consultable by administrators to respond appropriately.

The application will then consist of a Multiplatform App to collect the information, a server with a database to store it, and a web front-end to consult this stored information.

The project will be developed based on a proposal from the “Colla Ecologista La Carrasca”.

Keywords

application; multiplatform; web app; xamarin; database; invasive species; geolocation; ecology

Dedicatoria

A mi madre, que ha estado apoyándome en mis estudios todos estos años.

A mi padre, que sé que le hubiera hecho ilusión verme terminar la carrera.

A mi hermano, por ayudarme y darme buenos consejos.

Agradecimientos

A mi tutor, Rubén Pérez, por su ayuda y su paciencia conmigo.

A mis amigos/as y compañeros/as de la carrera, por su ayuda y consejos.

Índice de contenido

Índice de figuras

Índice de ilustraciones

1 Introducción

Existe en el mundo un gran problema a nivel económico, medioambiental e incluso sanitario, que son las especies invasoras.

Estas son especies que se han introducido en un ecosistema de forma natural, accidental o intencionada. Se adaptan a un medio que no es el suyo y consiguen prosperar, reproducirse satisfactoriamente y crecer en número de individuos, desplazando a las especies nativas del medio donde se han introducido.

Las especies invasoras, que pueden ser animales, plantas e incluso hongos y microorganismos, son una de las grandes preocupaciones a nivel global actualmente. "Estas especies son la segunda causa de pérdida de biodiversidad en el mundo, según el Programa de las Naciones Unidas para el Desarrollo (PNUD)" [1].

Hay múltiples razones por las que se pueden introducir especies en un medio que no es el suyo, por ejemplo: el mercado de especies exóticas, el transporte de mercancías, la liberación de especies, e incluso el cultivo en la agricultura.

El primer paso para solucionar este problema es el reconocimiento de especies, rastrear su ubicación, y que un grupo especializado elimine o, si no es posible, reduzca al máximo posible el número de individuos de la especie invasora.

Es por tanto importante que el reconocimiento de especies y las posibles acciones de erradicación, si proceden, sean realizadas por profesionales.

Existe un caso muy cercano, ya que se encuentra bastante presente en Alcoy: la especie invasora ailanto (*Ailanthus altissima*), también conocida como "caña de la pudor". Este árbol se reproduce rápidamente y puede desplazar las plantas nativas gracias a sustancias bioquímicas que afectan a la germinación de otras plantas causando problemas en el ecosistema.

[1] Anónimo (2022): ¿Cómo impactan las especies exóticas invasoras sobre la biodiversidad?

Aparte las flores tienen mal olor, al ser polinizadas por las abejas, provocan que la miel adquiera un sabor desagradable, causando daños económicos[2].

Hay otros casos de plantas u hongos invasores, que provocan daños a la flora nativa y afectando por ende a la fauna autóctona. Son amenazas más difíciles de erradicar ya que se pueden reproducir de forma invisible y se desconoce su expansión en el medio. En el caso de los hongos provocan además daños masivos a la agricultura causando grandes pérdidas económicas.

El objetivo final será la obtención de una herramienta que permita notificar de inmediato el avistamiento de una especie invasora, compartiendo la ubicación de dónde se ha visto, a la asociación Colla Ecologista La Carrasca, que se encargara de analizar la especie y si procede revisar la zona y tomar las medidas oportunas.

1.1 Motivación

Hay una gran dificultad para controlar las especies invasoras a nivel nacional; no hay recursos suficientes para solucionar este problema. Aparte, la gestión del reconocimiento y erradicación de especies invasoras son competencia de las comunidades autónomas lo que dificulta la solución del problema, ya que no todas las comunidades pueden priorizar este problema y estas especies se expanden con facilidad.

A raíz de este problema, no son pocas las organizaciones que deciden realizar acciones al respecto para intentar mejorar la situación, informando e intentado evitar la expansión de estas especies y sus consecuencias. Esta aplicación surge por iniciativa de la Colla Ecologista La Carrasca, dada la necesidad de controlar los posibles avistamientos de especies invasoras en el ecosistema, con el añadido de otras alertas medioambientales.

1.2 Motivación personal

Mi motivación por elaborar este TFG ha sido tener la oportunidad de aportar mis conocimientos adquiridos en el Grado de Ingeniería Informática con el fin de crear una aplicación útil que ayude a proteger el ecosistema de Alcoy, colaborando con una magnifica organización cuyos miembros

[2] Ministerio de Agricultura, Alimentación y Medioambiente (2022): Ailanthus altissima – Catalogo Español de Especies Exóticas Invasoras

siempre están dispuestos a poner sus conocimientos al servicio de la protección del medioambiente local.

A su vez, la realización de este proyecto me permite ampliar mis conocimientos sobre aplicaciones que trabajan con la localización GPS y con archivos multimedia como son las imágenes.

1.3 Objetivos del proyecto

El objetivo principal de este proyecto es realizar, por un lado, una aplicación móvil para que los usuarios puedan informar de avistamientos de especies invasoras. Estos avistamientos son: la ubicación de la especie, una evidencia (si es posible), y la posibilidad de enviar información adicional.

Por otro lado, la creación de una web para que usuarios autorizados y administradores puedan confirmar y gestionar los avistamientos. Podrán listar todos los avistamientos con su ubicación, evidencia e información, confirmar o descartar el avistamiento de la especie invasora.

La aplicación móvil cumplirá los siguientes requisitos:

- Que sea una aplicación móvil multiplataforma, siendo las principales plataformas Android, iOS y Windows.
- Tener acceso a la geolocalización del dispositivo y poder compartir su ubicación.
- Compartir imágenes, ya sea desde la galería o realizando una fotografía en ese momento.
- No requiere que los usuarios se tengan que registrar para utilizar la herramienta.
- Poder consultar las evidencias enviadas.
- El usuario tiene la posibilidad de enviar información adicional.

La web cumplirá los siguientes requisitos:

- Permitir a los usuarios acceder a todos los avistamientos enviados por los usuarios.
- Se pueden marcar y visualizar los registros con los estados “revisado” y “pendiente”.
- Los usuarios tendrán que registrarse para poder tener acceso a los datos.
- Podrán eliminar los registros que vean oportunos.

- Podrán editar algunos campos como el título y observaciones de los registros recibidos.
- Poder visualizar de forma sencilla en un mapa la geolocalización recibida.

1.4 Estructura de la memoria

El apartado 2, “Estado del arte”, trata de una comparativa con otras herramientas de la misma índole.

En el apartado 3, “Análisis del problema”, se enumeran los requisitos necesarios para la realización de esta aplicación

En el apartado 4, “Diseño de la solución”, se explica cómo se ha llevado a cabo la realización de la aplicación.

En el apartado 5, “Desarrollo de la solución propuesta”, se enumeran las dificultades y soluciones que ha habido durante la realización del proyecto.

En el apartado 6, “Tour por la Aplicación”, se expondrán, a través de imágenes, el resultado final de la aplicación.

En el apartado 7, “Conclusiones”, se mencionan las conclusiones que se han obtenido en la realización de este proyecto.

En el apartado 8, “Trabajos futuros”, se enumeran futuras mejoras de la aplicación que no han podido desarrollarse para este proyecto.

En el apartado 9, “Referencias”, se listan las diferentes fuentes bibliográficas de donde se ha obtenido información para la realización de este proyecto.

2 Estado del arte

Uno de los principales problemas a la hora de resolver la gestión de las especies invasoras es detectar y confirmar la ubicación de los individuos. Hace un tiempo se podían informar de avistamientos a las autoridades o a organizaciones, pero no era posible confirmar que fuese una especie invasora y era más difícil delimitar la ubicación de un avistamiento, por lo que se perdía mucho tiempo y recursos en investigar un avistamiento.

Hoy en día, gracias a los dispositivos móviles, podemos conocer la ubicación de forma concreta y es posible obtener evidencias gráficas. Sin embargo, no todo el mundo tiene conocimientos para analizar e identificar una especie invasora. Tampoco el tiempo ni la experiencia para poder rastrear el origen o expansión de uno o varios avistamientos, por lo que se requiere poder notificar dichos avistamientos con la ubicación más o menos exacta y, si es posible, adjuntando una fotografía, para que personas con los conocimientos requeridos puedan identificar e investigar estas posibles evidencias, realizando un seguimiento y rastreando las ubicaciones para encontrar el origen y/o expansión de las especies invasoras confirmadas. Con esta información, se puede notificar a las autoridades para que gestionen su control o erradicación.

2.1 Crítica al estado del arte

Actualmente hay algunas aplicaciones disponibles en el mercado móvil, que realizan tareas similares a las que se pretenden en este proyecto. También existen muchas páginas webs con información detallada sobre especies invasoras, tanto a nivel nacional como algunas zonas específicas.

Hay que hacer mención especialmente a “GVA Invasoras” [3], una gran base de datos informativa en la que se listan las especies invasoras, con sus fotos y descripción.

También se puede enviar ubicaciones, y se pueden visualizar en un mapa las especies invasoras confirmadas con su localización.

Sin embargo, a pesar de ser una aplicación muy completa, no permite el envío de fotografías para facilitar la identificación de las especies.

Otro problema es que hay que introducir la especie que se ha avistado, y no siempre el usuario puede reconocer la especie.

[3] GVA Invasoras

Y, por último, tiene algunos problemas como que no se especifican las ubicaciones concretas, ni cómo confirman la especie invasora, ni hay forma de informar de una posible nueva especie. También se desconoce si se realiza un seguimiento de la evolución de la situación o si se realizan acciones para solucionar este problema. Parece más bien algo informativo.

2.2 Propuesta

Lo que se pretende con este trabajo es facilitar al máximo las tareas de detección de especies invasoras, gestionando y controlando mejor la situación actual al basarse en una pequeña zona.

Los voluntarios especializados catalogan las especies y realizan incursiones para confirmar los avistamientos. Es un complemento perfecto para realizar actividades educativas e informativas ecológicas, en las que se rastrea el origen de las posibles especies invasoras y/o su captura controlada. Ayudando de esta forma a las autoridades en la erradicación de la especie invasora en el medio local.

No se añade información sobre las especies, ya que existen muchos recursos actualizados y fiables, que se complementan perfectamente con el presente trabajo. También se pueden usar las ubicaciones confirmadas para contribuir en la mejora de la base de datos existente creada por la GVA (Generalitat Valenciana).

3 Análisis del problema

El objetivo principal es realizar un proyecto con los siguientes puntos diferenciados:

- Aplicación móvil: Tiene que ser capaz de enviar la ubicación y captar y enviar fotografías, también debe de poder consultar las evidencias enviadas anteriormente. Es importante que esta aplicación se pueda utilizar en la mayor parte posible de los dispositivos móviles del mercado. No se registrará el acceso, se podrá identificar el dispositivo mediante un identificador único.
- Página web: Gestión de los registros que se han enviado. Es muy importante poder visualizar las ubicaciones de forma cómoda en un mapa y poder revisar las evidencias gráficas enviadas por los usuarios. Se van a poder catalogar los registros y se va a poder modificar registros para realizar anotaciones en las evidencias enviadas. Acceso solo mediante registro.
- Base de datos: Almacenar las evidencias permitiendo crear, modificar y eliminar registros de forma segura.
- Servidor: Encargado de guardar o recuperar los registros de la base de datos.

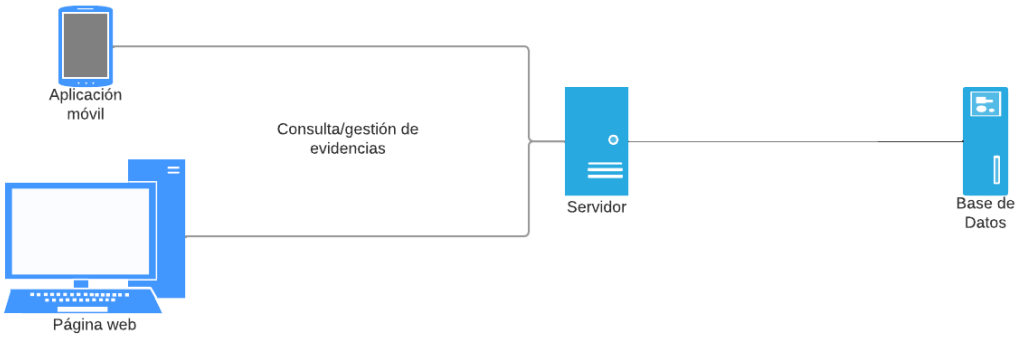

Figura 1: Diagrama de red del proyecto

3.1 Análisis de requisitos

La aplicación constará de una App Multiplataforma para la recopilación de la información, de un servidor con una BBDD para almacenarla, y de un *front-end* web para poder consultar esta información almacenada.

Se va a utilizar un lenguaje de programación que permita su uso en múltiples plataformas, siendo un lenguaje ampliamente utilizado y actual.

La aplicación multiplataforma tendrá las siguientes funcionalidades:

- Menú simple y completo, que facilitara el acceso a todos los puntos del programa.
- No requerirá identificarse ni registro alguno por parte del usuario, evitando que sea tedioso para el mismo.
- Tendrá que permitir acceso a su ubicación, ya que su principal labor es enviar las coordenadas de la localización.
- Podrán compartir imágenes desde su galería o realizar una fotografía en ese momento.
- Permitirá que el usuario pueda revisar las incidencias enviadas, pero no le permitirá editarlas.

El *front-end* web tendrá las siguientes funcionalidades:

- Requerirá identificarse por parte de los empleados de la Colla Ecologista La Carrasca, ya que desde aquí se tendrá acceso a toda la información almacenada.
- Los usuarios podrán gestionar esta información, procediendo a eliminarla de la base de datos, o marcarla como pendiente de revisar o completada.
- Los usuarios podrán modificar el texto de las incidencias recibidas si lo ven oportuno, pero no podrán modificar las imágenes ni la ubicación de estas.
- Podrán gestionar su propio perfil de usuario.
- La ubicación de las incidencias se podrá visualizar tanto en un mapa de Google Maps como ver sus coordenadas.

3.2 Identificación y análisis de soluciones posibles

A lo largo de este proyecto se ha comprobado que algunas de las decisiones estaban más o menos claras.

3.2.1 Parte móvil

Hay dos formas de implementar un proyecto para dispositivos móviles con acceso a cámara y GPS. Las opciones eran:

Web App

Se trata de aplicaciones móviles que se abren usando el navegador web. Cumple con los requisitos básicos ya que, con el uso de una *web app*, se puede acceder a la ubicación o a la cámara del móvil.

Algunas de sus ventajas son que, al ser usado desde el navegador, se puede usar en cualquier dispositivo independientemente de la plataforma, no requiere que los usuarios actualicen la *web app*, y no usa tantos recursos como una aplicación nativa.

Sus desventajas son que solo se puede usar teniendo acceso a internet. Hay que dedicarle tiempo en mejorar la página web, ya que de nada sirve si no adaptas tu página para que se pueda reproducir con calidad en cualquier dispositivo.

Se descarta esta opción porque a pesar de que con el uso de HTML5, cumpliría con los requisitos, esta nueva funcionalidad es reciente y podría dar problemas ahora o en el futuro a la hora de cumplir con los requisitos [4].

App Nativa

Tenemos la opción también de crear una aplicación, ya que también cumple con los requisitos básicos.

Los puntos positivos de este tipo de aplicación serian que se puede usar sin internet, aunque para el envío de la evidencia sí que va a requerir acceso a Internet. Otro punto es que puede acceder mejor a las funcionalidades de GPS, cámara y/o galería de forma fiable.

Entre sus puntos negativos se encuentran que hay que instalar la aplicación en el dispositivo, y que hay que actualizar los instalables con las actualizaciones. Otro problema es que, aunque hay soluciones multiplataforma, es posible que haya problemas en la implementación en todos los dispositivos y hay que realizar pruebas en la mayor cantidad de dispositivos posibles.

Lenguajes nativos

El uso de lenguajes nativos como Java para Android u Objective C para iPhone con sus entornos de trabajo correspondientes, se pueden usar para realizar aplicaciones muy potentes y especializadas.

El problema es que se debería de crear una aplicación para cada entorno, lo cual es muy costoso en cuanto a recursos humanos y temporales.

[4] Anónimo (2021): Acceder a la cámara o el carrete del móvil desde *web app* con HTML5

Uno de los requisitos de la aplicación es que debía de ser sencilla y fácil de usar, por lo que carece de sentido el invertir recursos en esta solución.

Plataformas híbridas

Existen en el mercado múltiples plataformas con distintos lenguajes de programación, que permiten con el mismo código, crear aplicaciones para cualquier dispositivo.

Xamarin

El lenguaje que utiliza es C#. Tiene un acceso a todas las funcionalidades de todos los dispositivos móviles, permitiendo crear aplicaciones nativas para smartphones o tablets Android, iOS o Windows.

Funciona con el IDE Visual Studio, y ofrece muchos emuladores para poder probar el funcionamiento en múltiples dispositivos emulados.

Flutter

El lenguaje que utiliza es Dart. Permite crear aplicaciones en varias plataformas, tanto móviles Android, iOS, web y de escritorio.

En cuanto a desventajas, podría decirse que es relativamente nuevo, por lo que tiene pocas bibliotecas de terceros. Las aplicaciones generadas tienden a ser de gran tamaño, si el espacio de almacenamiento del usuario es limitado, el rendimiento de la aplicación puede verse afectado. Y el lenguaje Dart no es muy popular [5].

Otras Alternativas

Existen muchas alternativas, las más conocidas son aquellas que usan JavaScript. Por mencionar la que más nos interesa, sería Cordova de parte de Apache, ya que es la que mejor trabaja con las funcionalidades nativas del dispositivo como la cámara o el GPS.

No tiene un entorno de desarrollo por defecto, y las aplicaciones creadas con esta plataforma suelen ser rápidas y ligeras.

Sin embargo, el acceso a la funcionalidad nativa del dispositivo no es tan óptima como Xamarin y no se puede considerar una aplicación nativa.

[5] Anónimo (2021): Qué es Flutter? Ventajas y cómo funciona

3.2.2 Almacenamiento

Una opción para almacenar información de las evidencias sería una base de datos relacional. Ya que esta solución nos va a permitir almacenar los datos de los avistamientos, lo datos de los usuarios administradores, las categorías de las especies, y permite crear muchas más relaciones, por lo que es una solución escalable.

La problemática de esta aplicación en cuanto al almacenamiento se encuentra en el guardado de las fotografías. En esta casuística tendríamos una opción más.

NAS

Con el uso de un servidor de almacenamiento en la nube se podrían almacenar imágenes de forma rápida y sencilla sin requerir ninguna conversión. Usando la ubicación del archivo, se hace la relación de la evidencia con la imagen adjunta. Se utiliza este tipo de servidor en los casos en los que haya múltiples imágenes por cada registro y haya una cantidad masiva de imágenes.

La desventaja es que requeriría un servidor adicional funcionando y es un elemento más que habría que gestionar y mantener.

Gestor de Base de Datos relacional

Con el uso de los tipos de campos especiales como el BLOB, se permiten almacenar objetos binarios muy grandes. Son utilizados para almacenar XML o imágenes. Por lo que para usar esta opción en el caso de las imágenes hay que tener en cuenta que habría que codificar el formato del campo binario para poder almacenar el objeto, y decodificarlo para obtenerlo.

PostgreSQL

PostgreSQL es un gestor de base de datos relacional muy conocido y uno de los que más trayectoria tiene en el mercado. Es de código libre y multiplataforma, se puede usar en una gran cantidad de entornos.

No es el más ligero ni rápido, pero está optimizado para trabajar con grandes grupos de datos.

Oracle

Una de las soluciones más completas, es el gestor de base de datos de Oracle. Tiene un alto grado de escalabilidad, y mantiene un buen rendimiento a medida que crecen los datos almacenados.

El principal problema es que no es de software libre, por lo que a pesar de que tiene opción gratuita, esta ofrece poco soporte.

MySQL

Es otro gestor de base de datos muy conocido. Es propiedad de Oracle, aunque está enfocado a ser un software libre. Tiene mucha documentación, y aunque no es óptimo como la de Oracle ya que es más lento y limitado en cuanto a capacidad de almacenamiento, es una de las soluciones más populares.

3.2.3 Interfaz de usuario

Vamos a necesitar una interfaz gráfica, para que los voluntarios de la asociación puedan gestionar los registros recibidos.

Hay varias formas de crear interfaces, pero tiene que ser ligera y sencilla, usarse en la mayor cantidad de dispositivos posibles, por lo que la solución que mejor se adapta a estos requisitos es un cliente web, ya que se adapta a los requisitos perfectamente, al ser accesible desde un navegador web, se puede entrar desde prácticamente cualquier tipo de dispositivo. También es importante que tenga un diseño responsivo.

Actualmente, para el desarrollo *front-end* se suelen utilizar *frameworks* basados en JavaScript. JavaScript es uno de los lenguajes más conocidos y utilizados en el mundo.

Angular

Es uno de los más populares actualmente, tiene múltiples funcionalidades que mejoran y aceleran el desarrollo con esta tecnología. Se considera consistente, es sencillo crear módulos, servicios y componentes.

La principal desventaja que afecta al presente proyecto es que los usuarios lo suelen considerar uno de los *frameworks* de *front-end* más complejos y que requieren de más tiempo para aprender a utilizar.

React

React es otro *framework* basado en JavaScript, es bastante sencillo de utilizar y tiene un buen rendimiento respecto a los cambios de estado. Sus principales problemas es que este *framework* se actualiza y crece constantemente, y los nuevos cambios no están documentados, por lo que puede ser algo complicado para los desarrolladores ponerse al día con las últimas actualizaciones.

Vue.js

Es otro de los *frameworks* ampliamente utilizado, que destaca por su flexibilidad. También es el más sencillo e intuitivo de utilizar, con un buen rendimiento y tiene las mejores herramientas actualmente. El exceso de flexibilidad en Vue puede provocar que algunas aplicaciones se vuelvan complejas provocando errores y complicaciones.

3.2.4 Servidor

Vamos a necesitar un servidor que se comunique con los clientes web y móvil, realice el registro y se comunique con la base de datos.

Python

Lenguaje de programación gratuito, de código libre y fácil de utilizar. Dispone de múltiples bibliotecas para por ejemplo acceso a la Base de Datos, por lo que algunas de las características ya se encuentran total o parcialmente implementadas y no requieren ser desarrolladas.

Su principal inconveniente es la dependencia con bibliotecas externas y, similar a Java, la lentitud y el alto consumo de recursos.

Java

Se trata de un lenguaje fácil de usar, escalable y bastante seguro. Sin embargo, no es la solución más eficiente para servidores y tiene un alto costo en hardware. Debido a que la aplicación se tiene que mantener simple, esta opción está directamente descartada.

PHP

Lenguaje de código abierto ampliamente utilizado. Es fácil de entender y dispone de un entorno de programación simple. Es también muy seguro por sus características de seguridad integradas.

3.3 Solución propuesta

3.3.1 Xamarin.Forms

En este proyecto se requiere una parte móvil para que los usuarios puedan enviar fotos junto con la localización actual. Por lo que es importante que se pueda usar en múltiples plataformas.

La limitación en este tipo de aplicaciones móviles es que, para desarrollar en cada tipo de plataforma móvil, se requiere un lenguaje de programación distinto, por ejemplo, Java para Android o Objective-C para iOS. Por esta razón he decidido usar Xamarin, que se trata de una plataforma en la que con el mismo código y lenguaje en C# se puede crear una aplicación móvil para todas las plataformas.

Otra de sus ventajas es que es una plataforma ampliamente usada, que ya lleva años de trayectoria, por lo que no va a tener problemas de pérdida de soporte. En un futuro próximo, va a dar el salto a .NET MAUI, ampliando de esta forma sus plataformas en las que se podrá desarrollar aplicaciones incluyendo a MacOS.

3.3.2 MySQL

Se trata de un gestor de base de datos con múltiples ventajas en el mercado, es gratuita y ampliamente conocida y utilizada, con mucha documentación y sencilla de utilizar.

Al tener varias capas de seguridad, se pueden crear múltiples usuarios con diversos permisos y se puede securizar ampliamente la aplicación según las necesidades actuales y futuras.

Es uno de los gestores más rápidos y con mejor rendimiento, no requiere muchos recursos hardware.

El único inconveniente seria a la hora de trabajar con datos muy masivos, a pesar de tener un buen rendimiento hay otros gestores que trabajan mejor en esas circunstancias, sin embargo, en esta aplicación no se va a trabajar con conjuntos enormes de datos de por ejemplo millones de datos, por lo que la solución propuesta se adapta a las necesidades actuales y futuras de la aplicación.

3.3.3 Vue.js

Se requiere de un cliente web para los trabajadores de la Colla puedan gestionar las imágenes recibidas por los usuarios.

El principal lenguaje para el desarrollo de clientes web es JavaScript, desde hace casi una década se están usando *frameworks* basados en JavaScript que mejoran el diseño y facilitan el desarrollo.

Los principales *frameworks* son React, Angular y Vue.js. Estas tres opciones son muy similares, pero Vue.js es la más sencilla de aprender a utilizar, es el mejor documentado y el que mayor soporte de la comunidad de desarrolladores tiene.

3.3.4 PHP

Para la parte servidor que conectara con la base de datos se ha elegido el lenguaje PHP. Es un lenguaje de código abierto que se utiliza para generar páginas webs dinámicas, por lo que podemos vincularlo a una base de datos para que el contenido sea cambiante.

3.4 Plan de Trabajo

El proyecto se ha dividido en varias etapas, con un total de 13 semanas y media de trabajo, desde el 24/03/2022 al 28/06/2022, y aproximadamente 5 horas de lunes a viernes, dando un total de 335 horas.

A continuación, se enumeran las etapas realizadas:

- Toma y análisis de requisitos.
- Inicio de la web con Vue.js: Creación del proyecto web con el *framework* Vue.js y realización de pruebas con datos locales.
- Diseño y creación de la base de datos: Creación de una base de datos usando MySQL Workbench, creando las tablas oportunas.
- Inicio del servidor con PHP: Creación de un servidor que conecte con la base de datos y realice las peticiones ya sea insertando datos como haciendo consultas.
- Pruebas desde la web: Interactuar con la base de datos a través del servidor mediante peticiones GET y POST.
- Pruebas con imágenes: Insertar imágenes en la base de datos.
- API de Google Maps: Insertar mapa al proyecto Vue.js usando los datos de latitud y longitud.
- Arreglar diseño de la web: Insertar reglas al CSS para mejorar la visualización de la web.
- Xamarin: Inicio de la aplicación multiplataforma en Xamarin.
- Obtener ubicación: Pruebas para obtener ubicación del dispositivo.
- Obtener foto: Pruebas para realizar fotos desde el dispositivo.

- Enviar datos desde Xamarin: Pruebas para enviar la información al servidor PHP para insertar datos en la base de datos.
- Modificación de la base de datos: Añadiendo nuevos campos a la tabla y dando mayor capacidad a algunos campos para contener más cantidad de datos.
- Mapa en Xamarin: Insertar mapa a la aplicación multiplataforma.
- Mejorar diseño: Mejorar diseño de la aplicación multiplataforma y web.
- Últimos problemas: Resolver problemas pendientes en ambas aplicaciones.
- Pruebas finales: Realizar pruebas desde ambas aplicaciones para comprobar su correcto funcionamiento.

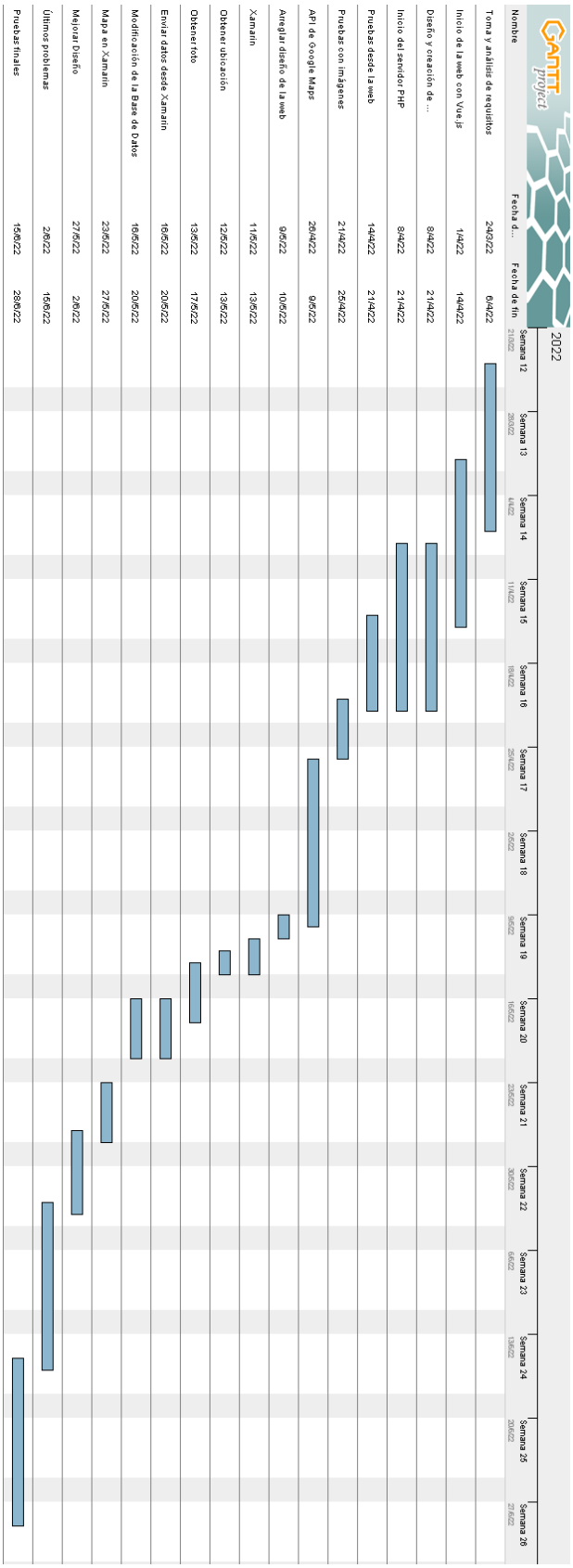

Figura 2: Diagrama de Gantt del plan de trabajo

4 Diseño de la solución

A continuación, voy a describir como he diseñado la aplicación.

4.1 Arquitectura del Sistema

Se va a trabajar con una arquitectura Cliente/Servidor, siendo el cliente tanto la aplicación móvil como el navegador web, y el servidor un servidor web. La idea es que las aplicaciones clientes hagan peticiones al servidor, y este se encargue de conectar a la base de datos realizando las peticiones con los datos recibidos.

El servidor usará el lenguaje PHP, realiza la conexión a la base de datos mediante la clase MySQLi (MySQL Improved), que ya desde PHP 5 permite acceder a bases de datos MySQL. Mediante peticiones POST, GET, PUT y DELETE se encargará de añadir, listar, actualizar o eliminar los datos correspondientes.

El cliente web usara el *framework* de Vue.js, para realizar las peticiones al servidor, usaremos Axios, un cliente HTTP.

El cliente móvil, usara Xamarin, y para realizar las peticiones al servidor, usara la clase HttpClient. Los datos irán serializados en JSON usando la clase Newtonsoft.Json.

La base de datos está en MySQL y contiene tres tablas (users, registros y categoría)

La tabla users contará con los usuarios de la Colla Ecologista la Carrasca que se encargarán de gestionar la aplicación web, revisando los datos recibidos, eliminando o modificando los que vean convenientes.

La tabla registros contendrá los datos enviados desde la aplicación multiplataforma por parte de los usuarios que utilizan la herramienta, estos usuarios no necesitan registrarse, pero se enviará un UID generado de sus dispositivos.

La tabla categoría registrará diferentes categorías para clasificar los registros recibidos, esto está pensando para implantarse en un futuro.

4.2 Diseño Detallado

4.2.1 Patrón Model-View-ViewModel (MVVM)

La aplicación multiplataforma está diseñada siguiente el patrón Model-View-ViewModel, este ayuda a separar la lógica de negocio de su interfaz de usuario.

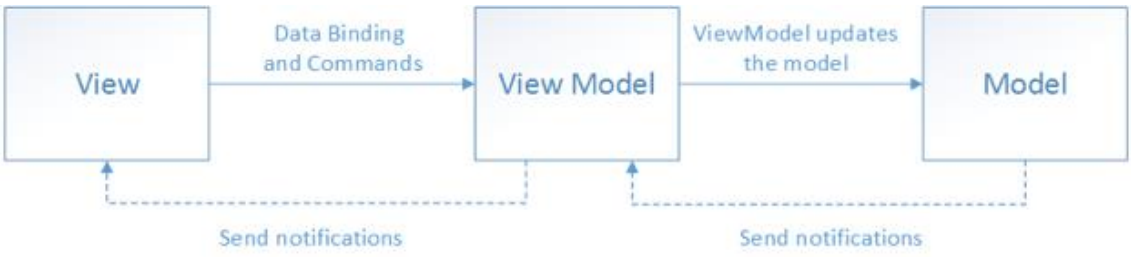

Figura 3: Diagrama Patrón MVVM - Relaciones entre componentes

Fuente: Documentación de Microsoft

El Modelo contiene las clases. A la hora de iniciar una aplicación es el primer componente tomado en cuenta, ya que aquí definimos los atributos que usaremos en la App.

El ViewModel contiene toda la lógica de presentación. Se encarga de realizar la comunicación entre los modelos y las vistas.

El View (Vista) es la interfaz de usuario, define como la información y las funcionalidades se mostrarán gráficamente.

4.3 Tecnología Utilizada

4.3.1 Entorno de desarrollo integrado (IDE)

Para poder programar en el lenguaje de programación escogido, se necesita un entorno que sea amigable y facilite el desarrollo del proyecto.

Para la programación en Xamarin usaré Microsoft Visual Studio.

Microsoft Visual Studio

Es un IDE desarrollado por Microsoft, se pueden desarrollar aplicaciones tanto para Windows UWP, como Xamarin, como múltiples lenguajes, tales como C++, C#, Visual Basic .NET, F#, Java, Python, Ruby y PHP.

Se puede usar Visual Studio para MAC, pero no se creará el Proyecto para UWP. Y para generar la app para iOS se necesita un Mac obligatoriamente.

Para la programación en Vue.js y en PHP usaré Visual Studio Code.

Visual Studio Code

Editor de código fuente desarrollado por Microsoft. Es gratuito y de código abierto. Facilita a los usuarios la escritura del código.

Cuenta con una gran compatibilidad con los lenguajes de programación que se usan a diario, como por ejemplo JavaScript, CSS, HTML, C, C++, Java, C#, PHP, Python, Visual Basic, Perl, etc.

Tiene soporte multilingüe incorporado, es decir, detecta fácilmente si hay algún fallo o referencia entre lenguajes.

4.3.2 Soporte de la base de datos

Para la base de datos en MySQL se utilizará MySQL Workbench, una herramienta de software libre con multitud de funcionalidades para administrar la base de datos.

Es multiplataforma y es una de las soluciones más conocidas y populares para trabajar con MySQL con un amplio recorrido. Esta herramienta nos va a permitir diseñar y modelar la base de datos, tiene un editor SQL, y administrar la base de datos de forma cómoda con su completa interfaz gráfica.

4.3.3 Geolocalización

La creación de un mapa, y trabajar con las coordenadas recibidas del GPS del dispositivo móvil es uno de los requisitos principales de este proyecto.

Hay algunas herramientas con las cuales se pueden visualizar mapas, sin embargo, para la web con Vue se ha utilizado Google Maps ya que está mejor documentado. Todas las opciones son más o menos similares, pero Google Maps ofrece mejor experiencia al usuario, y es muy importante la correcta visualización del mapa sobretodo en el navegador.

Para implementar el mapa hay que generar una API Key de Google Maps, dentro de un contenedor se genera el mapa y se visualizará en el componente padre. Para evitar errores, se crea el mapa cuando se haya cargado los datos.

Para Xamarin la cosa cambia, se utiliza una herramienta distinta por cada entorno. El proceso es similar, necesitamos permisos especiales en cada plataforma y se genera la API Key según si es Android o Windows.

Tanto para los dispositivos Android como iOS se utiliza Google Maps, y por último UWP se usaría Bing Maps.

5 Desarrollo de la solución propuesta

5.1 Desarrollo

5.1.1 UID y reconocimiento de dispositivos

La asociación solicitó que el acceso a la aplicación fuese anónimo, pero necesitamos alguna manera de reconocer los dispositivos para que los usuarios puedan listar los registros enviados. Esta información también es muy útil, porque permite realizar un seguimiento de los usuarios de forma anónima y para saber cuántos usuarios han instalado la aplicación en sus dispositivos móviles.

Esto se puede realizar con el uso de un UID, o identificador único. Se trata de un identificador autogenerado, se crearía en el dispositivo del usuario, y se guarda en local. En caso de que no exista un identificar, se genera uno nuevo.

```
0 referencias
protected override void OnStart()
{
    if (App.Current.Properties.ContainsKey("uid"))
    {
        uid = (string)App.Current.Properties["uid"];
    }
    else
    {
        uid=Guid.NewGuid().ToString();
        Properties["uid"] = uid;
    }
}
```

Ilustración 1: Código de generación del UID

5.1.2 Envío de datos

A continuación, se explica cómo se ha realizado la implementación del envío de datos entre aplicaciones. Los clientes móviles y web envían peticiones al servidor, y este realiza la acción solicitada y devuelve la información. Para implementar esto, se ha creado una API Rest, mediante la cual los clientes se pueden comunicar con el servidor. Existen varias acciones para las peticiones: GET para solicitar información, POST para insertar datos, PUT para modificar registros, DELETE para borrado de datos.

Para realizar este proceso, es importante que los datos que se envían a través de la red se encuentren serializados, por ellos se van a codificar los datos a formato JSON. Vamos a listar cómo se ha implementado esto en

cada aplicación, usando ejemplos gráficos para el caso de obtener la lista de registros, ya que se usa en todos las situaciones.

Estas llamadas al servidor han de ser asíncronas, ya que hay una latencia entre el envío de la petición y la respuesta.

Móvil

En el caso del cliente móvil hay varias formas de consumir un API, si no hay ningún proceso en cola, va a conectarse mediante HttpClient. El cliente va a enviar una petición al servidor, en este caso con su UID, para que el servidor busque los registros creados por este usuario, la petición será GET ya que estamos obteniendo registros para listarlos en el móvil.

```
2 referencias
private async Task GetLista()
{
    if (IsBusy) return;

    Exception error = null;
    try
    {
        IsBusy = true;

        using (var client = new HttpClient())
        {
            var jsontxt = await client.GetStringAsync(RequestUri + "?uid="+ App.Current.Properties["uid"]);
            var items = JsonConvert.DeserializeObject<List<Registro>>(jsontxt);

            ListaPersonal.Clear();
            foreach (var item in items)
            {
                ListaPersonal.Add(item);
            }
        }
```

Ilustración 2: Código de ejemplo de cliente Rest

La otra acción disponible para el móvil es POST ya que se permite insertar nuevos registros. No se utiliza DELETE, ya que el borrado no se puede realizar desde la aplicación móvil, esto solo lo puede realizar un usuario autorizado. Lo mismo pasa con PUT, ya que no se permite modificar los datos enviados.

Web

Para el cliente de Vue, tenemos varias formas de consumir el API, se va a utilizar axios que se trata de un cliente HTTP basado en promesas. Le enviamos la ruta y los parámetros, y con el uso de las promesas, podemos hacer la llamada, obtener la respuesta y capturar los posibles errores de forma cómoda.

```
function listRegistros(token, query, cb) {

    axios.get(url_registros, { params: query })
        .then((resp) => {
            cb(null, resp.data);
        })
        .catch((err) => {
            if (err.response) {
                cb(new Error(err.response.status));
            } else if (err.request) {
                cb(new Error('No response received'));
            } else {
                cb(err);
            }
        });
}
```

Ilustración 3: Código de ejemplo de cliente Rest

En el caso de Vue vamos a utilizar también los métodos PUT para modificar datos, este caso es muy útil para poder gestionar los registros marcando si están pendientes o no. Se usa también la acción DELETE, para poder eliminar los registros que hayan sido descartados. De esta forma se tiene un control básico en la gestión de registros.

Servidor

Tendremos los métodos GET, PUT, POST y DELETE publicados en el servidor. En caso de recibir una petición GET, el servidor se comunica con la base de datos ejecutando una sentencia SQL de tipo Select sobre la tabla de registros para obtener todos los datos de la tabla y poder enviárselos al cliente. Además, en caso de que la petición tenga algún parámetro, se guardara en la variable Where para filtrar los resultados de la sentencia.

```php
} elseif (!$_SERVER['PATH_INFO'] and $_SERVER['REQUEST_METHOD'] == 'GET') {
    // listRegistros
    error_log('listar registro');

    $sql = "SELECT * FROM registros";
    $where = '';

    foreach ($_GET as $key => $val) {
        error_log($key.' --- '.$val);
        if ($where) $where = $where . " AND $key='$val'";
        else $where = "$key='$val'";
    }

    if ($where) $sql = $sql . ' WHERE ' . $where;
    error_log($sql);

    $regs = array();
    $res = $db->query($sql);


    if ($res) {
        while ($row = $res->fetch_assoc()) {

            $reg = array(
                'id' => $row['id'],
                'name' => $row['name'],
                'img' => ($row['img']),
                'lat' => floatval($row['lat']),
                'lng' => floatval($row['lng']),
                'obs' => $row['obs'],
                'fecha' => $row['fecha'],
                'uid' => $row['uid'],
                'categ_id' => $row['categ_id'],
                'pending' => boolval($row['pending'])
            );

            array_push($regs, $reg);
        }

        error_log('registro listado');
        header('Content-Type: application/json');
        echo json_encode($regs);
    } else {
```

Ilustración 4: Código de ejemplo del servidor Rest

Con los resultados de la sentencia, el servidor creará una estructura para almacenar todos los campos de los registros, posteriormente lo codificará con JSON indicando en la cabecera el tipo de contenido que se envía.

Tendremos todas las acciones declaradas en el servidor, por lo que los clientes pueden consumir la acción que requieran, el servidor crea la sentencia SQL que corresponda añadiendo filtros si procede, y ejecuta la sentencia contra la base de datos. Con el resultado de la ejecución de la sentencia de base de datos, el servidor devolverá datos o resultados de la petición.

5.2 Funcionalidades móviles nativas

Los principales requisitos de la aplicación es obtener la ubicación actual y poder realizar fotografías. Para esto Xamarin va a tener que acceder a las funcionalidades nativas de receptor GPS y cámara que hay en la mayoría de los dispositivos móviles.

5.2.1 Localización

Vamos a necesitar usar la funcionalidad móvil del GPS para obtener las coordenadas actuales, para esto en Xamarin vamos a usar el plugin Geolocator. En cada plataforma habrá que dar en el manifiesto los permisos para el uso de la localización.

```
1 referencia
private async void Location()
{
    try
    {
        var locator = CrossGeolocator.Current;
        locator.DesiredAccuracy = 20;
        pos = await locator.GetPositionAsync(TimeSpan.FromSeconds(20), null, true);
        if (pos != null)
        {
            btnEnviar.IsEnabled = true;
        }
        else
        {
            btnEnviar.IsEnabled = false;
        }
    }
```

Ilustración 5: Código de la localización

Creamos en primer lugar una instancia de Geolocator, y posteriormente vamos a obtener la ubicación. El plugin se utiliza para poder utilizar la funcionalidad nativa del dispositivo móvil de GPS.

El GPS obtiene la posición del dispositivo conectándose a varios satélites, debe tener conexión con varios satélites y cuantos más satélites más precisa será la posición. Usando las redes y frecuencias con las que operan los satélites, el receptor GPS recibe la señal de los satélites conociendo su situación en el cielo y el software de posicionamiento calculará la posición

tanto horizontal como vertical. La posición no será exacta, siempre suele haber un pequeño descuadre, aunque se ha mejorado mucho esta funcionalidad con el paso de los años.

Solo cuando se haya seleccionado una ubicación, se podrá clicar en el botón de "Enviar". También se va a permitir obtener una ubicación y realizar el envío posteriormente.

5.2.2 Cámara

Vamos a usar MediaPicker, que nos va a permitir utilizar tanto la cámara para hacer una fotografía como usar imágenes que tengamos en la galería. Hay que tener en cuenta que vamos a tener que configurar los permisos para cada plataforma para poder tener acceso a la cámara o a la galería del dispositivo.

```
0 referencias
async void BtnUsarCamara_Clicked(object sender, EventArgs e)
{
    if (MediaPicker.IsCaptureSupported)
    {
        var foto = await MediaPicker.CapturePhotoAsync(new MediaPickerOptions()
        {
            Title = "Toma la foto"
        });

        await ObtenerDetallesFoto(foto);
    }
}
0 referencias
async void BtnSeleccionar_Clicked(object sender, EventArgs e)
{
    var foto = await MediaPicker.PickPhotoAsync(new MediaPickerOptions()
    {
        Title = "Elige una imagen"
    });
    await ObtenerDetallesFoto(foto);
}
```

Ilustración 6: Código de la cámara

5.3 Problemas encontrados

5.3.1 Problema con el mapa en Vue.js

Al realizar la web con el *framework* Vue.js e implementar la API de Google Maps para mostrar el mapa enviándole los datos de latitud y longitud, surgía un problema, dado que se cargaba el mapa antes de recibir dichos datos, cargando el mapa con los valores que había por defecto a latitud y longitud, siendo estos 0 y 0. La solución a este problema fue el establecer un *if* para indicar que, si los valores no existen, no cargue el mapa.

```
<div class="mapa" v-if="Boolean(registro.lat) && Boolean(registro.lng)">
    <Mapa
    :lat="registro.lat"
    :lng="registro.lng">
    </Mapa>
</div>
```

Ilustración 7: Código del mapa

5.3.2 Problema con la imagen y JSON

En la realización de la aplicación móvil en Xamarin, se debe poder enviar la imagen en formato JSON para almacenarla en la base de datos, y también el poder recuperarla para mostrarla en la sección de enviados. Por ello, se convierte la imagen a base64 para poder serializarla en JSON.

El problema surge cuando se pretende recuperar la imagen para mostrarla al usuario en la sección de enviados. Se recupera la imagen en un campo de tipo *ImageSource*, dando error entonces a la hora de enviar registros, porque trata de serializar el campo de tipo ImageSource, y no es tipo texto, la solución de este error es establecer [JsonIgnore] para que no lo serialice ese campo.

```
[JsonIgnore]
0 referencias
public ImageSource Imagen
{
    get { return this.ImageFrom64(); }
}

1 referencia
public ImageSource ImageFrom64()
{
    byte[] byteArray = Convert.FromBase64String(this.Img);
    Stream stream = new MemoryStream(byteArray);
    return ImageSource.FromStream(() => stream);
}
```

Ilustración 8: Código del campo ImageSource

5.3.3 Problema de conexión al servidor desde Xamarin

A la hora de realizar la conexión de la aplicación para móvil en Xamarin con el servidor PHP, ocurre un error, debido a que la aplicación Android establece la conexión usando la dirección IP, mientras que la aplicación Windows establece conexión usando *localhost*. La solución de este problema se solventa realizando el siguiente *switch*.

```
switch (Device.RuntimePlatform)
{
    case Device.iOS:
        break;
    case Device.Android:
        RequestUri = new Uri("http://192.168.1.135/php/registros.php");
        break;
    case Device.UWP:
        RequestUri = new Uri("http://localhost/php/registros.php");
        break;
}
```

Ilustración 9: Código de URL del servidor

5.3.4 Problema con el envío de fotografías

A la hora de implementar la aplicación en el móvil, surgió el problema que, al realizar una fotografía y darle a "Enviar", está no se guardaba en la base de datos. El campo que guarda la imagen, que es de tipo BLOB, soporta hasta 4 GB de tamaño, por lo que el tamaño del campo no era el problema.

En los archivos de logs del servidor Apache se indica que el problema era que, cuando PHP intentaba enviar el archivo a la base de datos, está limitaba el tamaño, produciendo el error. Para solucionarlo había que modificar el parámetro *max_allowed_packet* aumentando la capacidad que podía recibir.

6 Tour por la Aplicación

A través de capturas de pantalla se va a explicar la aplicación en una versión demo.

Primero se procederá a enseñar la aplicación multiplataforma que usarán los usuarios para enviar información, y luego el *front-end* web que se utilizará para consultar esta información.

6.1 Android

A continuación, se verá la aplicación en funcionamiento en un móvil con Android. No es necesario registrarse para acceder, y se dispone de dos acciones, enviar una nueva incidencia o revisar las enviadas.

Ilustración 10: Diseño gráfico del menú en Android

Si se escoge la opción de “Nueva incidencia” se mostrará la siguiente pantalla donde se podrá introducir un nombre y observaciones, y nos da la opción de seleccionar una imagen de la galería o realizar una fotografía con la cámara.

Ilustración 11: Diseño gráfico del formulario en Android

Aquí se puede apreciar una fotografia recién realizada, y también se puede seleccionar una imagen desde los archivos del móvil.

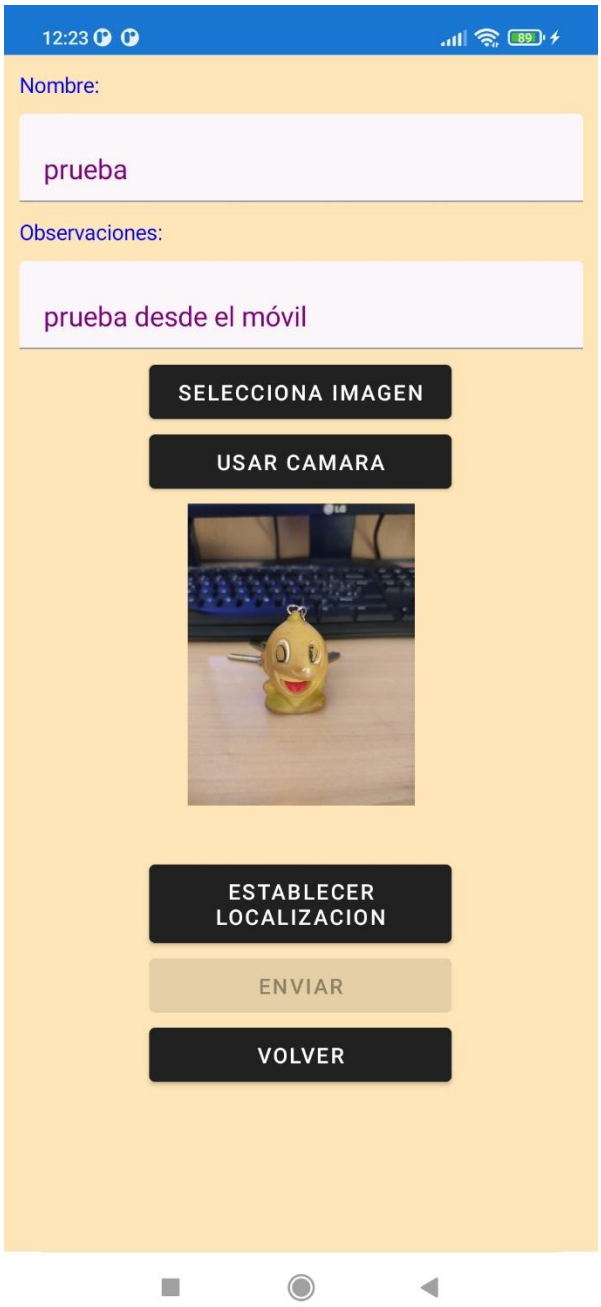

Ilustración 12: Diseño gráfico del formulario con fotografía en Android

Para poder enviar se debe establecer la localización en la que se encuentra el usuario. Una vez hecho se podrá enviar y volver al menú principal.

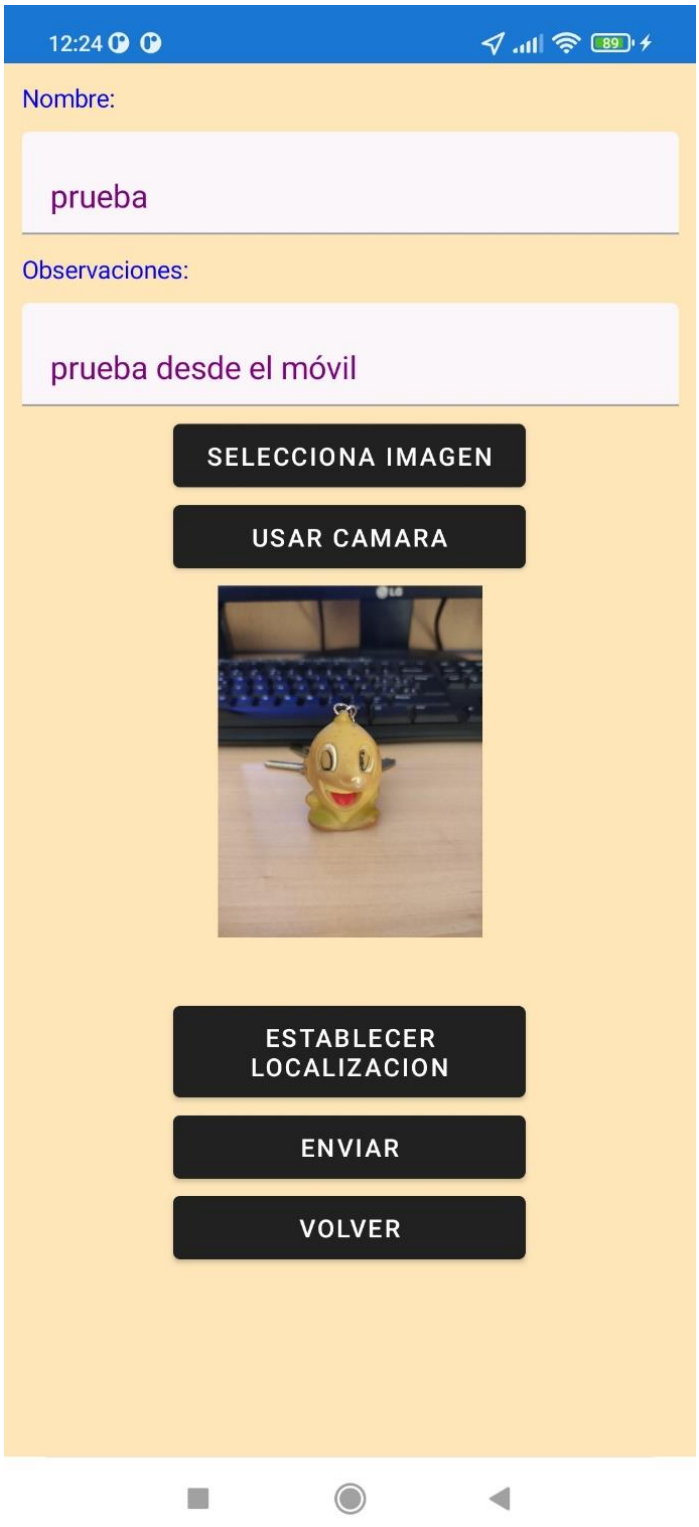

Ilustración 13: Diseño gráfico del formulario con localización en Android

Desde el menu principal se puede escoger ver los enviados para visualizar una lista de las propuestas enviadas, esto es posible porque cuando se envía una incidencia se incluye el UID del dispositivo, y luego se puede realizar una consulta a la base de datos sobre las incidencias enviadas con el UID recibido.

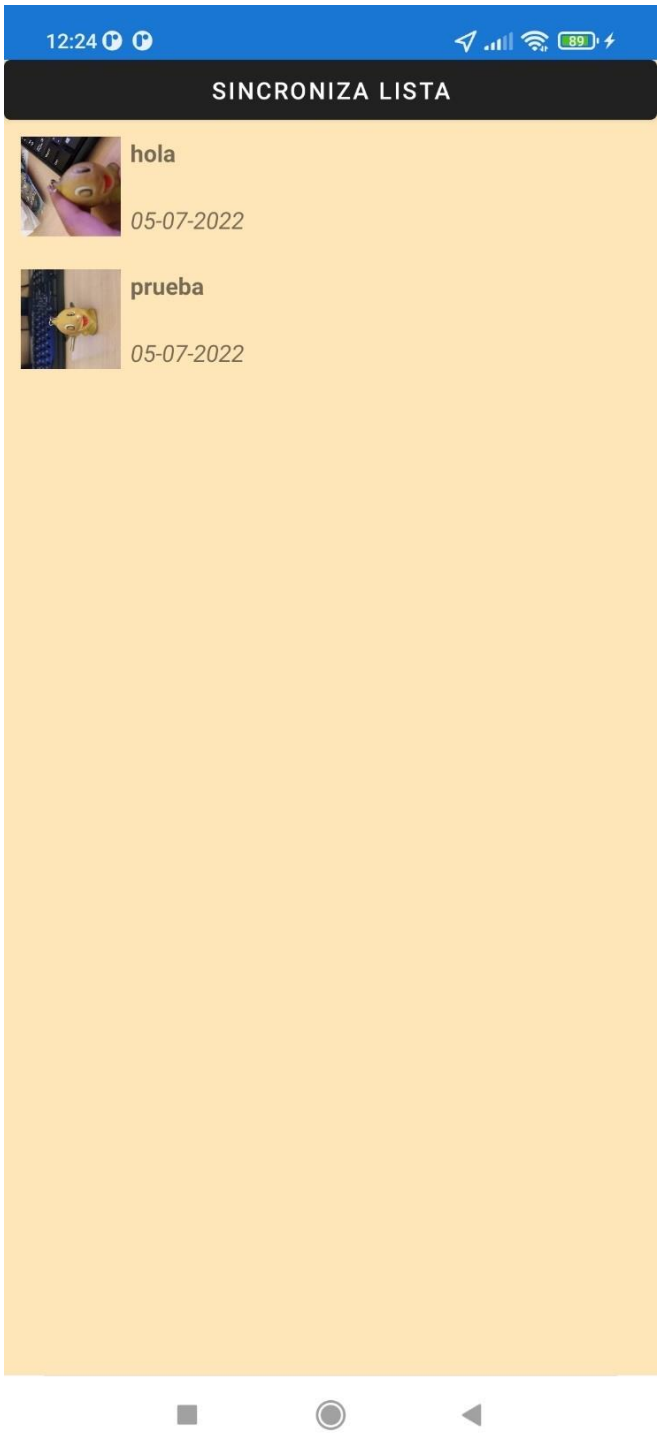

Ilustración 14: Diseño gráfico de la lista en Android

Al pulsar sobre alguna de las incidencias enviadas, se visualizará los datos además del mapa de Google Maps con la ubicación enviada.

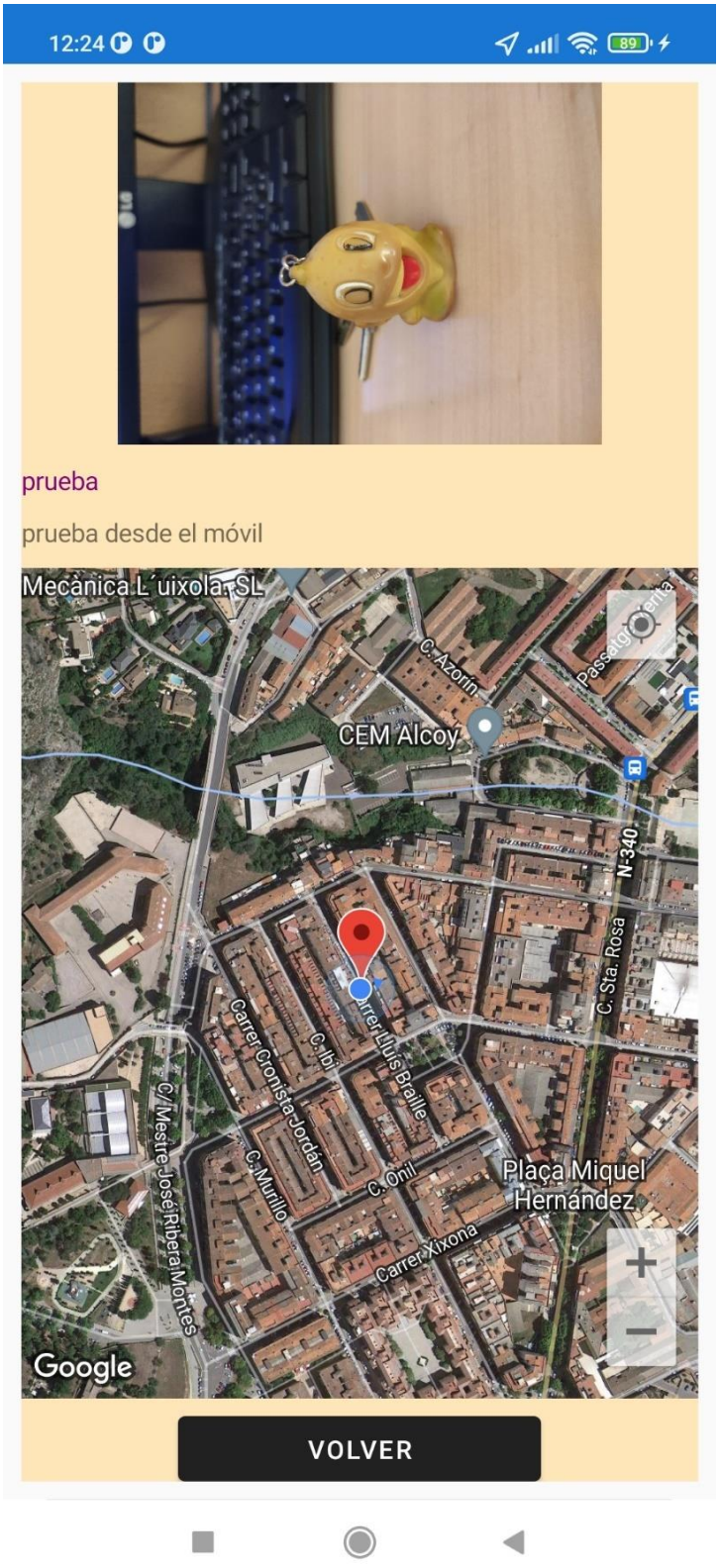

Ilustración 15: Diseño gráfico de la página detalle en Android

6.2 Windows

Si se accede a la aplicación desde la plataforma Windows se puede observar una variación del diseño, pero por lo demás es igual.

Ilustración 16: Diseño gráfico del menú en Windows

En la aplicación Windows se puede acceder a la cámara de la webcam para realizar fotografías.

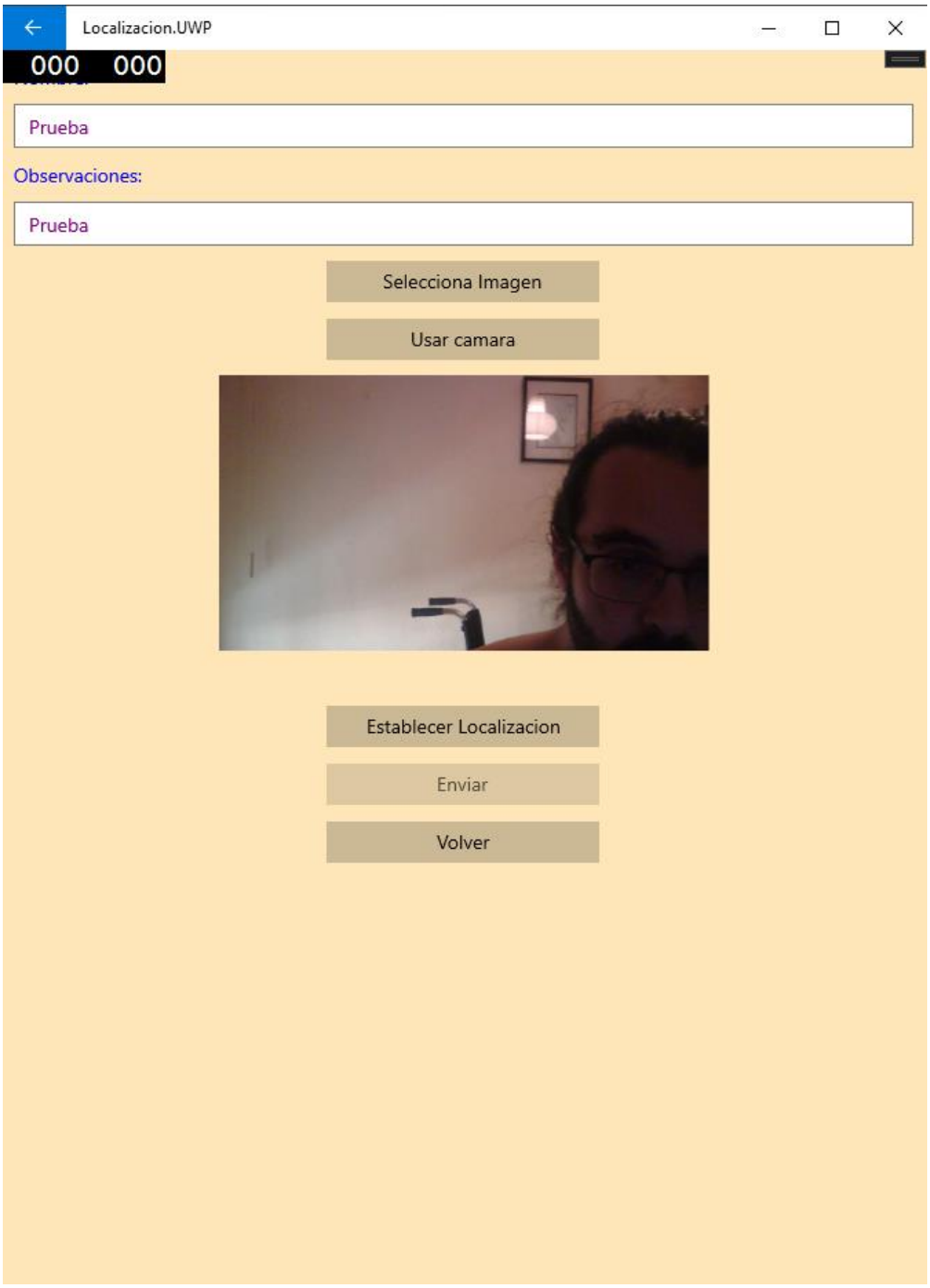

Ilustración 17: Diseño gráfico del formulario con fotografía en Windows

Y se puede seleccionar una imagen desde la galería del ordenador.

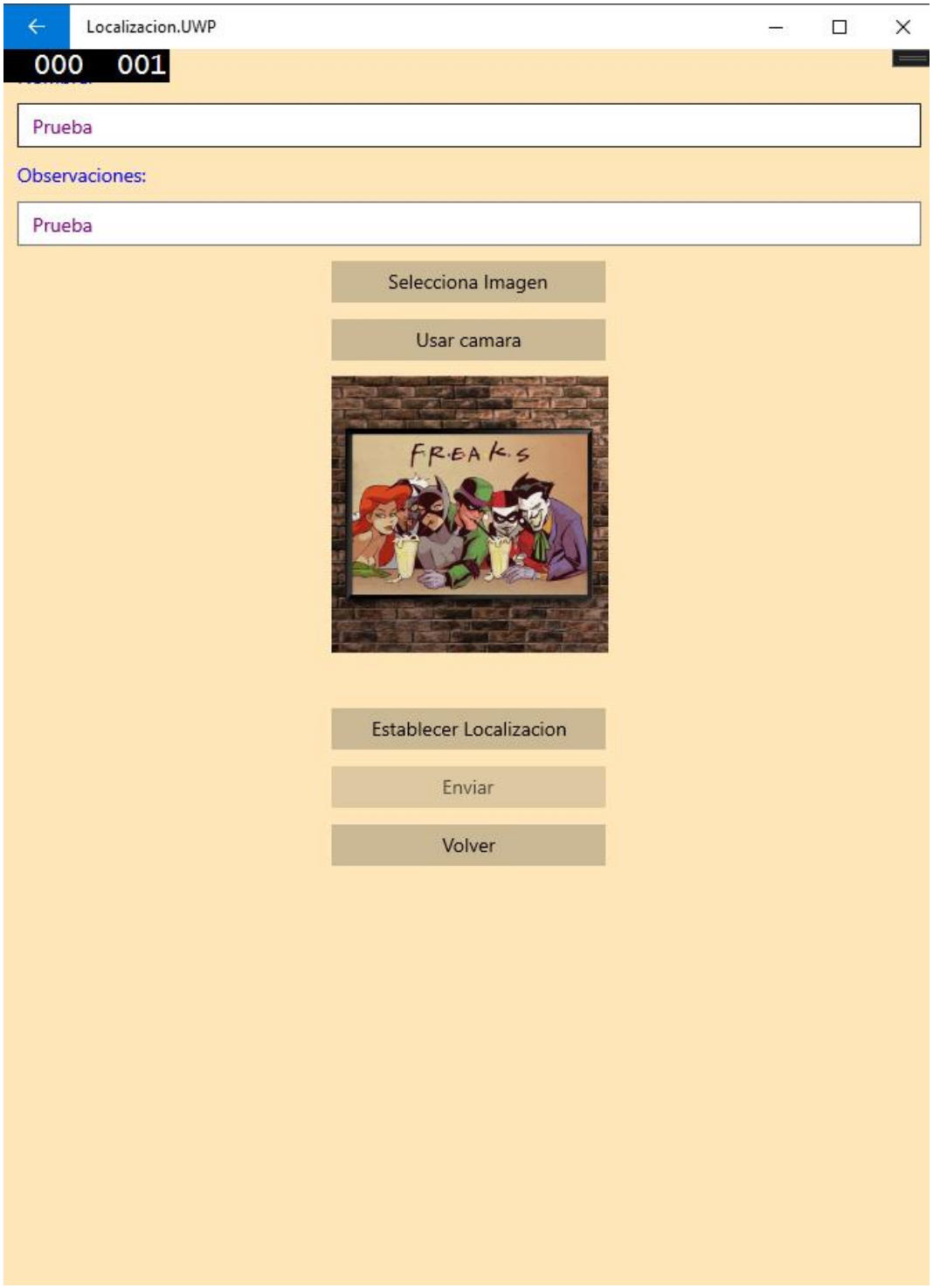

Ilustración 18: Diseño gráfico del formulario con imagen en Windows

Una vez establecida la localización y enviada la incidencia, se volverá al menú principal, y desde aquí se podrá consultar las incidencias enviadas.

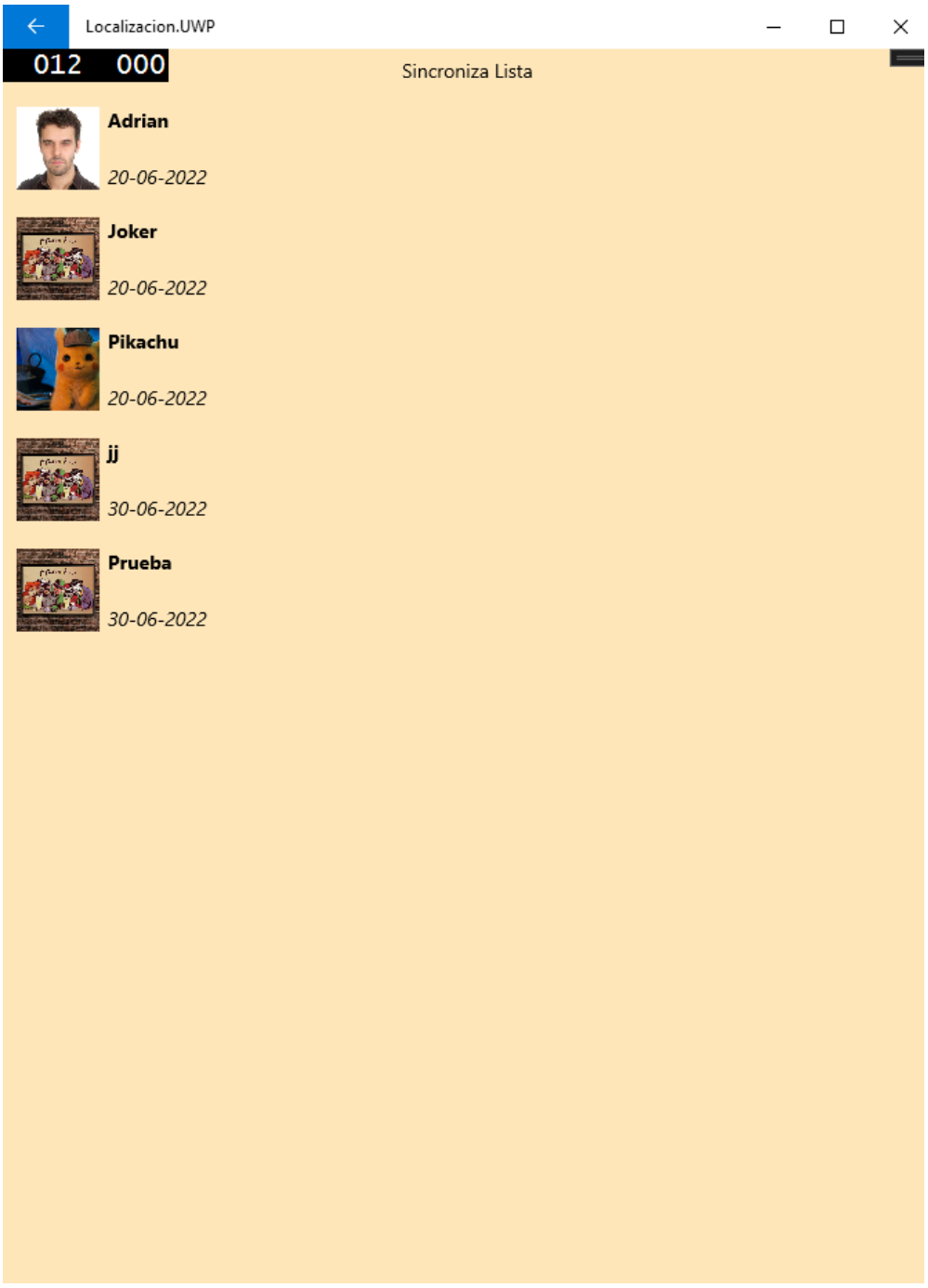

Ilustración 19: Diseño gráfico de la lista en Windows

Y si se accede a ver una incidencia se podrán ver los datos y la localización en el mapa. La aplicación del mapa es diferente a la de Android, ya que la de Android se basa en Google Maps y este en Bing Maps.

Ilustración 20: Diseño gráfico de la página detalle en Windows

6.3 iOS

Lamentablemente, para mostrar la aplicación en la plataforma de iOS es necesario disponer de algún dispositivo de Apple, y no se dispone de ninguno.

6.4 *Front-end* web

La Colla Ecologista La Carrasca tendrá acceso desde esta web a todas las incidencias recibidas para poder gestionarlas.

Para acceder tendrán que registrarse creando un usuario, para luego poder identificarse y acceder.

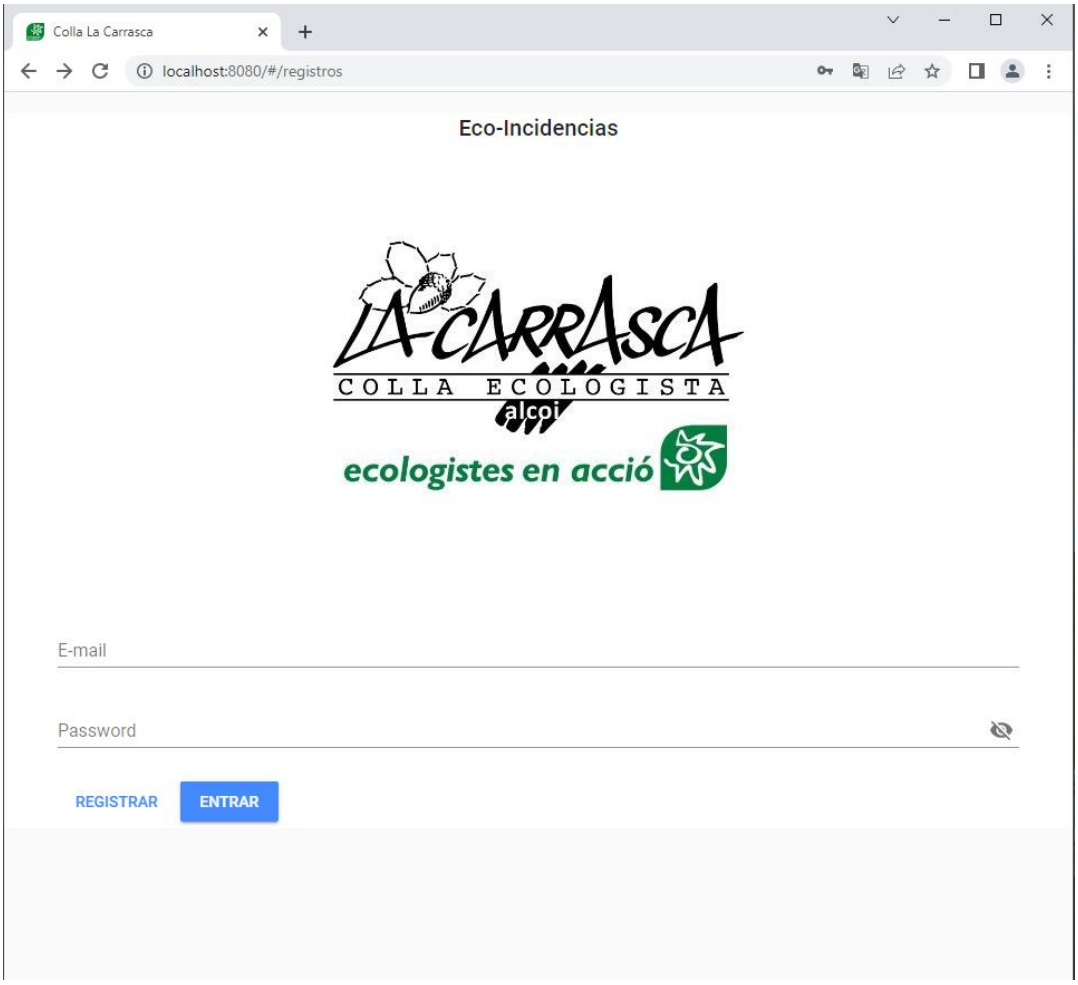

Ilustración 21: Diseño gráfico del menú en web

Para registrarse solo tienen que poner unos pocos datos y si quieren seleccionar una imagen.

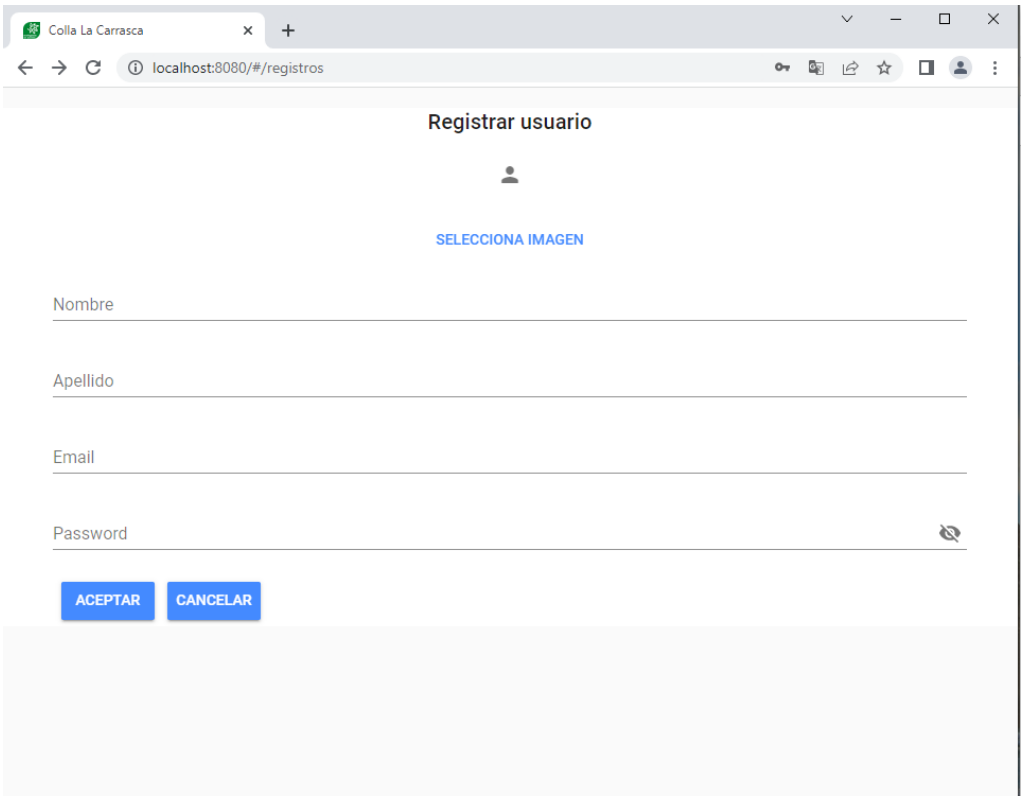

Ilustración 22: Diseño gráfico de formulario en web

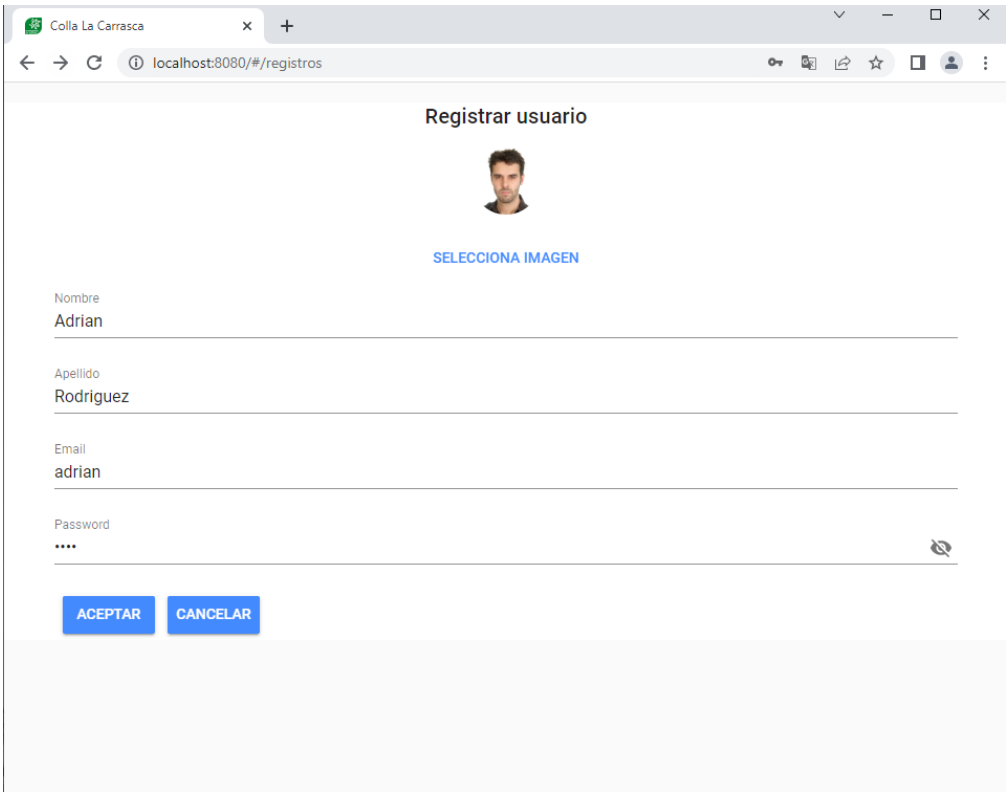

Ilustración 23: Diseño gráfico del formulario rellenado en web

Una vez dentro podrán visualizar todas las incidencias recibidas pudiendo filtrar por las que están pendientes de revisar por las que ya lo han sido.

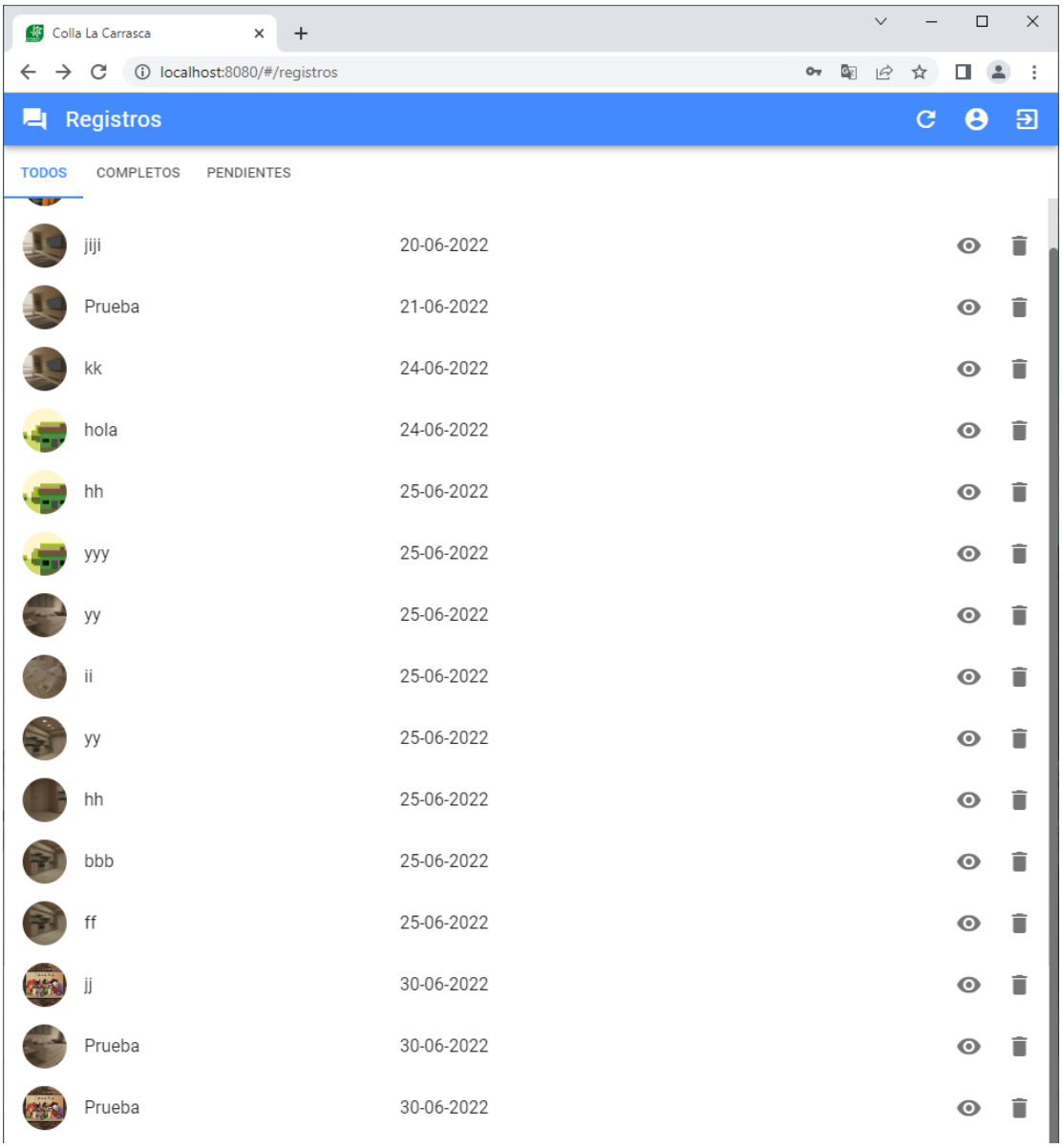

Ilustración 24: Diseño gráfico de la lista en web

Desde este listado pueden eliminar de la base de datos las que vean oportunas, o visualizar sus datos.

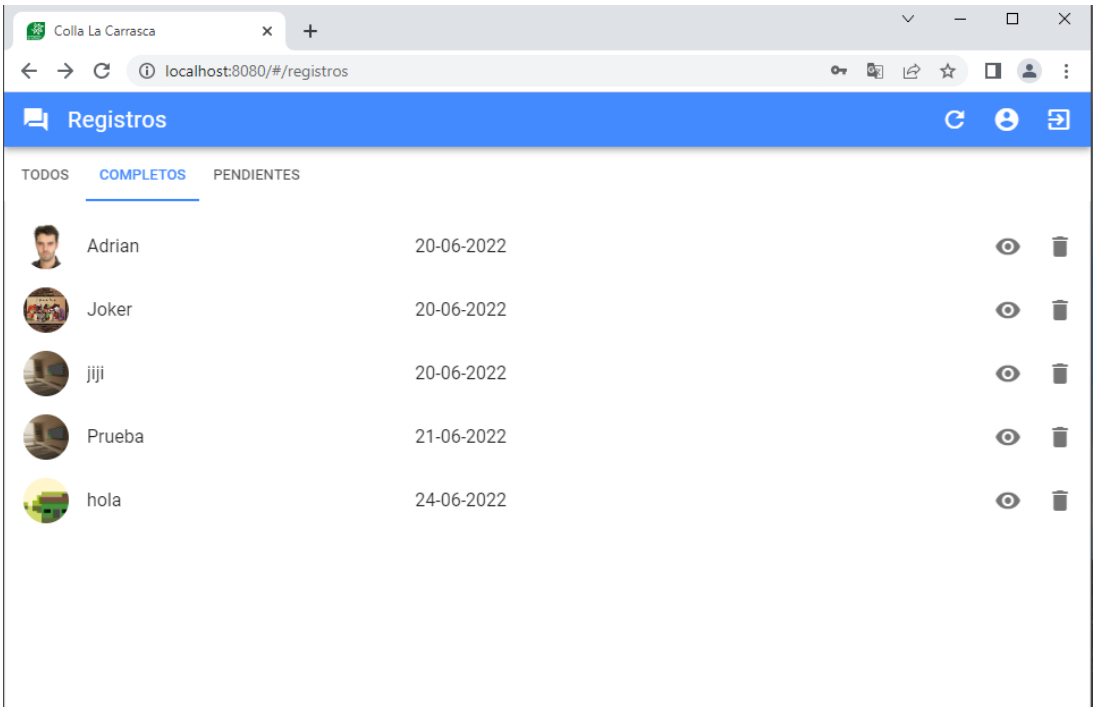

Ilustración 25: Diseño gráfico de la lista filtrada en web

Pueden acceder a sus datos de usuario para poder modificarlos si lo desean.

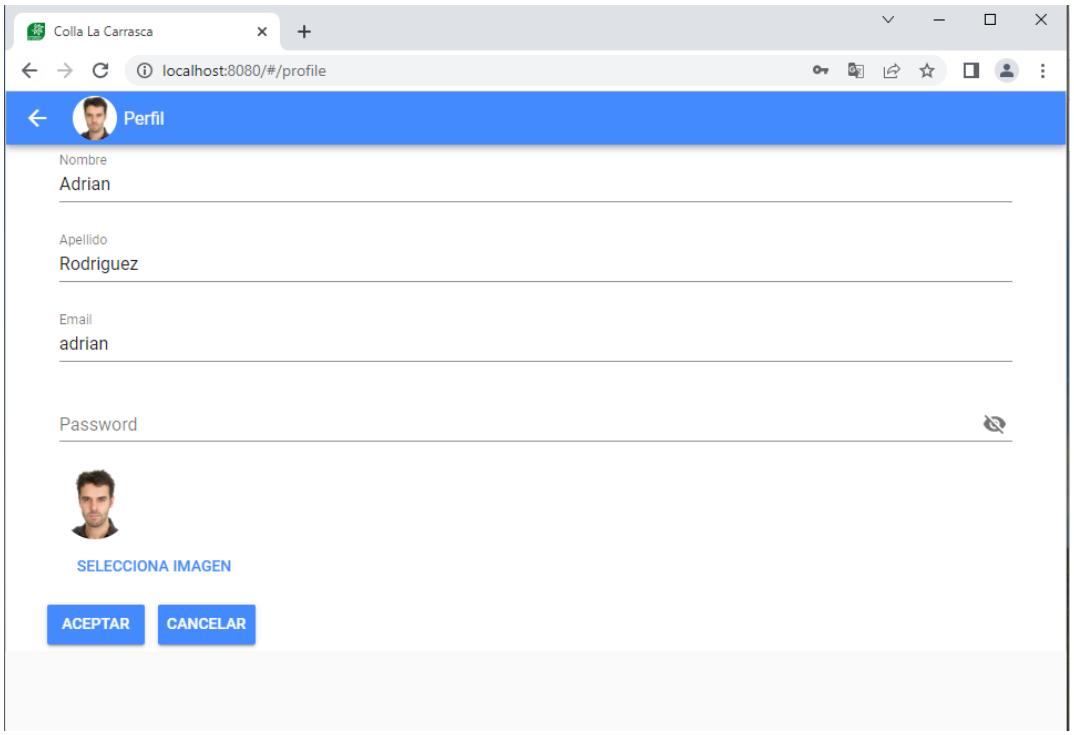

Ilustración 26: Diseño gráfico de formulario de perfil en web

Si acceden a los datos de una incidencia podrán visualizar los datos recibidos a su vez que un mapa con la ubicación además de las coordenadas. También pueden ver el UID del dispositivo que envió la incidencia.

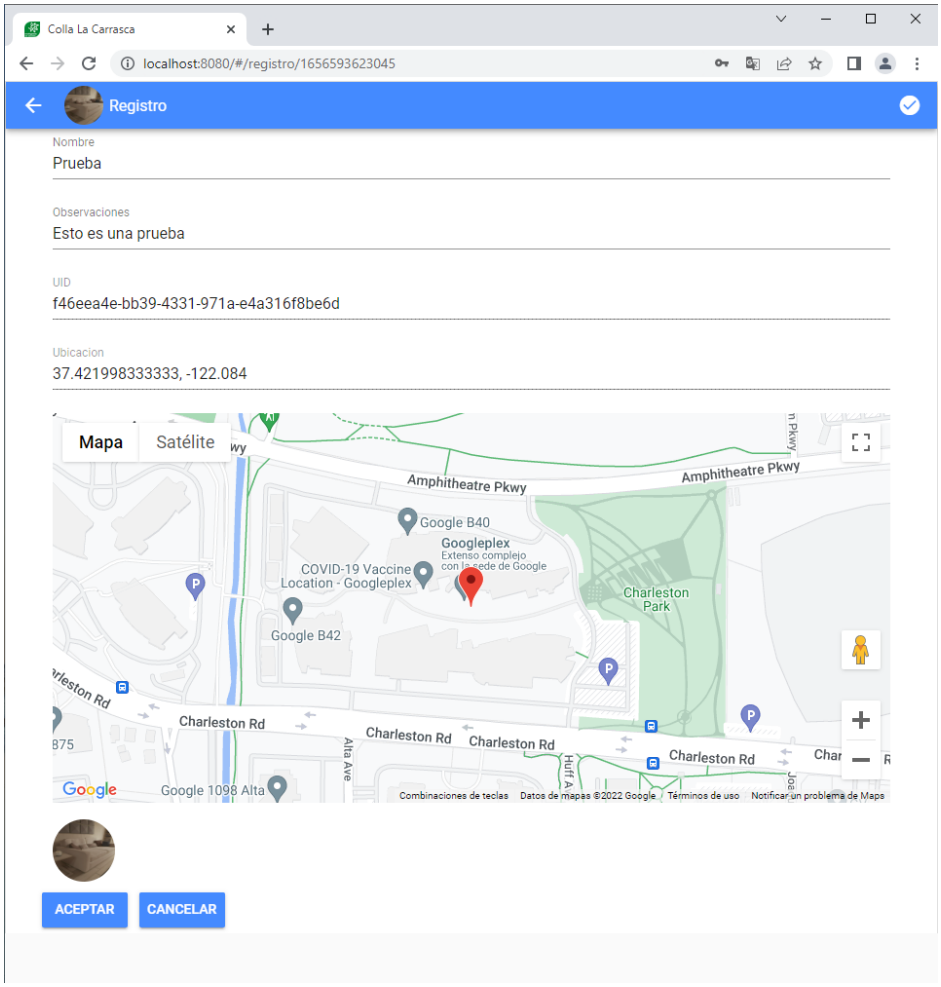

Ilustración 27: Diseño gráfico de formulario de registro en web

Pulsando sobre la imagen podrán aumentar su tamaño para visualizarla mejor.

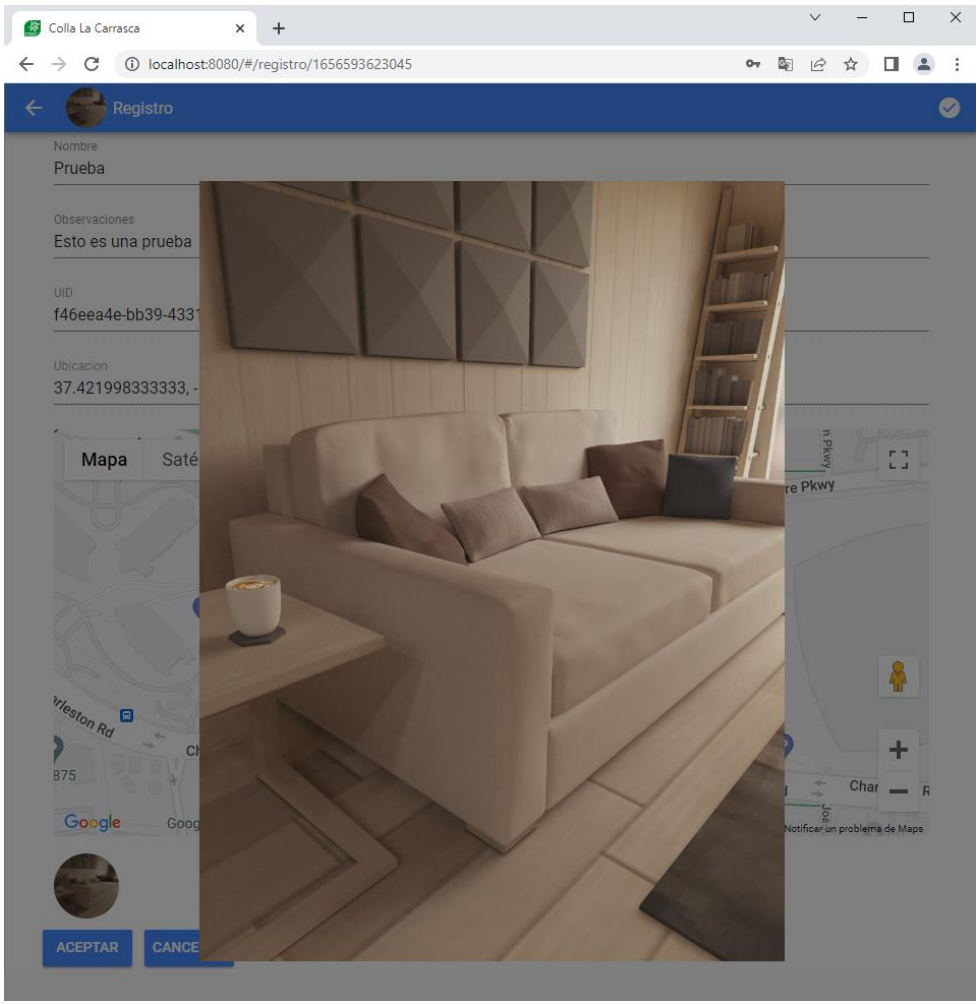

Ilustración 28: Diseño gráfico de fotografía en web

7 Conclusiones

En primer lugar, cabe mencionar que se han alcanzado los objetivos marcados con un desarrollo satisfactorio.

Gracias a la realización de dicho TFG, se han podido ampliar los conocimientos acerca de Xamarin y de Vue.js.

Respecto Xamarin, se ha aprendido a habilitar la localización del GPS para poder compartirla, y sobre como mostrar un mapa con dicha localización. Que según la plataforma debes establecer ciertas características para cada una.

A pesar de las dificultades, se pudieron solventar gracias a la amplia documentación y la comunidad de desarrolladores que se encuentra en internet, además del material que se ha adquirido en el Grado de Ingeniería Informática.

8 Trabajos futuros

En dicho proyecto se le puede complementar la realización de más trabajos con el fin de mejorar su funcionamiento y así poder prestar un mejor servicio.

Crear un apartado de Wikipedia sobre especies invasoras de la zona, en el cual los usuarios puedan consultar para adquirir algunos conocimientos.

Y tener el poder de clasificar por categorías las incidencias enviadas.

9 Referencias

9.1 Referencias Bibliográficas

- Pérez, R. (2020): Apuntes de la asignatura Soluciones informáticas para dispositivos móviles.
- Pérez, R. (2021): Apuntes de la asignatura Especialista universitario en computación móvil y ubicua.
- Esparza, J. (2020): Apuntes de la asignatura Desarrollo web.

9.2 Referencias Web

- Anónimo(2022): ¿Cómo impactan las especies exóticas invasoras sobre la biodiversidad?
- Ministerio de Agricultura, Alimentación y Medioambiente (2022): Ailanthus altissima – Catálogo Español de Especies Exóticas Invasoras
- Ministerio para la transición ecológica y el reto demográfico (2022): Catálogo Español de Especies Exóticas Invasoras
- GVA Invasoras
- Anónimo (2021): Acceder a la cámara o el carrete del móvil desde *web app* con HTML5
- Anónimo (2021): Qué es Flutter? Ventajas y cómo funciona
- Vue.js El *Framework* JavaScript Progresivo
- Stack Overflow
- All Questions de Microsoft
- Vue Material
- Documentación técnica de Microsoft
- PHP Tutorial
- Documentación de PHP
- Trafaniuc, V. (2020): Guía rápida: ¿cómo obtener la API key de Google Maps?
- Anónimo (2019): Cómo crear y usar los props en Vue JS
- Culoccioni, S. (2016): Almacenar archivos en campos BLOB con PHP y MySQL
- Documentación json.NET
- Documentación MySQL
- Wikipedia