UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Our Wildlife: una aplicación para aprender de la naturaleza

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: García Santes, Daniel

Tutor/a: Canós Cerdá, José Hilario

CURSO ACADÉMICO: 2021/2022

Resumen

Our Wildlife es una aplicación web para descubrir fauna y flora, con la que podremos aprender más sobre el lugar en el que vivimos y todo lo que nos rodea. La aplicación web proporciona al usuario la posibilidad de ir completando su propia lista de fauna y flora durante sus excursiones por la naturaleza, así como ver qué seres vivos le faltan para seguir incrementando el número de avistamientos. Está pensada como un registro de todos los seres vivos que, tras una excursión y/o paseo, el usuario puede dar de alta subiendo una foto para identificar la especie. Además, una vez dada de alta se pueden leer características de esa especie para saber algo más acerca de ella.

Palabras clave: Aplicación web; Fauna; Flora; Animales; Seres vivos; Avistamientos.

Abstract

Our Wildlife is a web application to discover wildlife, with which we can learn more about the place where we live and everything that surrounds us. The web application provides the user with the possibility of completing their own list of wildlife, as well as seeing which species are missing to continue increasing the number of sightings. It is designed as a record of all species that, after an excursion and/or walk, the user can register by uploading a photo to identify the species. In addition, once registered, you can read the characteristics of that species to learn more about it.

Keywords: Web Application; Wildlife; Species; Sightings.

Índice de contenidos

Índice de imágenes

Índice de tablas

1 Introducción y motivación

Este trabajo deriva de mi afición por los animales en general, y por aquellos que viven en libertad en la naturaleza, en particular. Llegado el momento de escoger un tema para el Trabajo Fin de Grado (TFG), pensé en combinar esa afición -casi diría que pasión- con la tecnología informática y desarrollar una aplicación con la que poder aprender al tiempo que se disfruta de la naturaleza.

Cuando se sale a pasear al campo se observan muchos animales y plantas de los que no se sabe qué especies son, ni nada acerca de ellos. Con frecuencia, queda la intriga de qué animal o planta se ha visto, intentándolo buscar en *Internet* como buenamente se puede, viendo la fauna y flora característica de la zona y navegando entre muchas imágenes, lo cual no es garantía de éxito. Es por esta razón que se pensó en una herramienta que solucionara este problema, añadiéndole persistencia a esas observaciones para que el usuario pudiera conservarlas a modo de historial, y haciendo uso de servicios de identificación para facilitar la búsqueda.

1.1 Objetivos

Como objetivo principal, del trabajo se planteó la creación de una aplicación web que permita registrar observaciones realizadas en la naturaleza, identificando de forma automática la especie observada, a fin de que los usuarios que utilicen la aplicación amplíen sus conocimientos sobre la naturaleza que les rodea.

Como objetivos secundarios, se podrían nombrar:

- Poder sugerir especies a otras observaciones ayudando a verificar la especie observada.
- Mostrar mapas para interactuar con las observaciones.
- Poder hacer rutas y ver qué especies hay cerca.
- Identificar especies sin necesidad de ser visibles para el resto de los usuarios de la aplicación.
- Ofrecer un buscador de especies que facilite encontrar una en concreto.

Además de todas las mencionadas, cabe destacar que este trabajo tratará alguno de los objetivos de desarrollo sostenibles que ahora mismo están siendo promovidos por las Naciones Unidas.

1.2 Metodología

Después de haber definido los objetivos que se van a afrontar, tenemos que escoger cómo va a ser el proceso software de implementación.

Se tenía claro que se iba a adoptar una metodología ágil [1], pero había que investigar cuál era la que más se adaptaba a la forma en la que se tenía pensado trabajar.

Durante los estudios de grado se realizaron un par de proyectos de desarrollo donde se nos enseñaba a aplicar metodologías ágiles, más en concreto *Scrum* [2]. Esta metodología define un proceso iterativo estructurado en incrementos o s*prints*. Un *sprint* es un intervalo de tiempo, normalmente entre 2 y 4 semanas, donde se desarrollarán funcionalidades previamente acordadas. Al finalizar uno, se realiza una entrega y validación de lo desarrollado con una persona especializada en el dominio de la aplicación.

Además de este tipo de metodología ágil se investigó acerca de la *Kanban* [2], en la cual se utilizan tarjetas para representar las tareas a realizar y se dispone de un tablero donde estas tarjetas pasan entre distintos estados, todos ellos personalizados al gusto de cada usuario. Esto, permite un flujo de trabajo continuo y un control sobre todo lo que se está llevando a cabo.

Una vez vistas y analizadas estas dos opciones, se observó que la metodología *Kanban* se adaptaba mejor a la idea de flujo de trabajo que se tenía contemplado realizar. Esto es debido a que se basa en un flujo de trabajo continuo que, actualizando el estado de las tareas en el correspondiente tablero, permite un avance más natural.

Para adoptar esta metodología se ha utilizado *Trello* [3], un software gratuito que nos proporciona todas las herramientas para realizar todo este trabajo muy cómodamente.

1.3 Estructura del documento

El documento está formado por diez capítulos contando la Introducción, apartado donde se está ahora mismo. Los restantes nueve capítulos se estructuran de la siguiente manera.

El Capítulo 2 tratará cómo se encuentra el mercado actualmente respecto a aplicaciones similares a la que se va a desarrollar. El Capítulo 3 detallará las técnicas utilizadas para realizar un análisis y especificación de requisitos que darán dimensión al proyecto a realizar. El Capítulo 4 tratará la arquitectura que seguirá la aplicación, así como un diseño detallado de las entidades con las que se trabajaran dentro de ésta.

El Capítulo 5 se dedicará exclusivamente para hablar de las tecnologías que se van a utilizar. Y el sexto detallará cómo ha sido el proceso de implementación, dividiéndolo en dos partes, una para el *backend* y la otra para el *frontend*.

El Capítulo 7 abordará las pruebas que validarán el correcto funcionamiento de la aplicación. En el octavo y noveno capítulos, se explicarán las conclusiones a las que se han llegado sobre el trabajo, y también los futuros trabajos que se podrán llevar a cabo sobre este tema.

Finalmente, como décimo y último apartado se dispondrá de la bibliografía consultada a lo largo de la redacción de este documento. También se disponen de anexos con información más detallada y un glosario para ampliar la información de ciertos términos.

2 Estado del arte

Con el fin de comprender mejor el dominio del problema, antes de optar por una solución potencialmente incompleta, se ha realizado un estudio de aplicaciones con una finalidad similar a la aplicación que se va a desarrollar para conocer qué funcionalidades se están ofreciendo actualmente en el mercado. La aplicación que se busca es cualquier plataforma donde gente de todo el mundo registre sus observaciones de la naturaleza y las comparta para aprender sobre ella. Estas observaciones son públicas, y en ciertas aplicaciones se pueden opinar sobre éstas, y determinar la especie a la que pertenece la observación, de modo que, se pueda tener más certeza de la especie observada. Estas funcionalidades de registrar la observación y, en algunos casos, hacer sugerencias sobre ésta, es sobre lo que pivotan las aplicaciones de esta temática, añadiendo funcionalidades complementarias para hacer más fácil la navegación y el flujo de información dentro de la misma.

Fruto de la investigación realizada, se han encontrado tres competidores interesantes de analizar. Primero se van a describir uno a uno para luego mostrar una tabla comparativa que resuma las diferencias entre todos ellos.

iSpot [4] es una aplicación web lanzada al mercado en 2009 como parte de una colaboración con *Open Air Laboratories*, iniciativa para involucrar a la ciudadanía británica en temas relativos a la naturaleza. Es por ello que la mayoría de los usuarios y observaciones registradas están hechas en el Reino Unido, aunque la aplicación no es restrictiva geográficamente hablando, ya que se pueden publicar observaciones en cualquier parte del mundo. Además de esto, se puede interactuar con otras observaciones sugiriendo otra especie o aceptando la sugerida por el usuario. En la Figura 1 se puede observar la interfaz de esta aplicación, disponiendo de un menú sencillo pese a que no se puede cambiar de idioma.

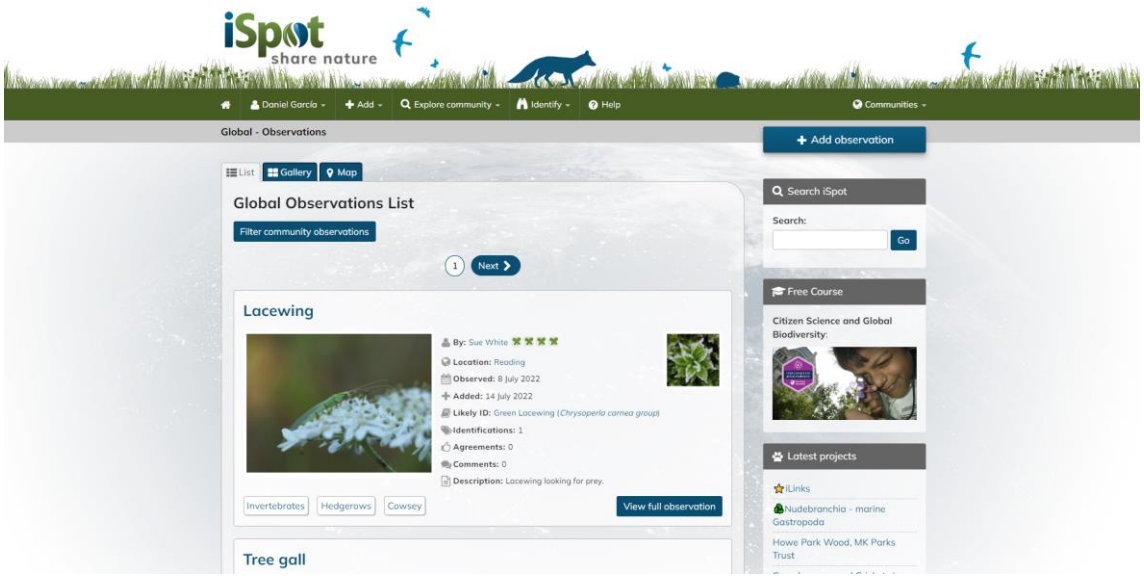

Figura 1: iSpot

Observation [5] es una aplicación, que además de disponer de versión web, cuenta con varias aplicaciones móviles [6] en el mercado para permitir el registro de las observaciones directamente cuando se están observando. Pero centrándonos en la página web, esta permite

cargar las observaciones sugiriéndonos posibles especies. En cambio, ningún usuario puede ayudarte a identificar la especie de la observación, pues no se pueden sugerir especies ni comentar una observación ajena. Además, como se puede ver en la Figura 2, en su vista principal donde se listan las observaciones, no hay ningún mapa que ayude a ver dónde se han realizado.

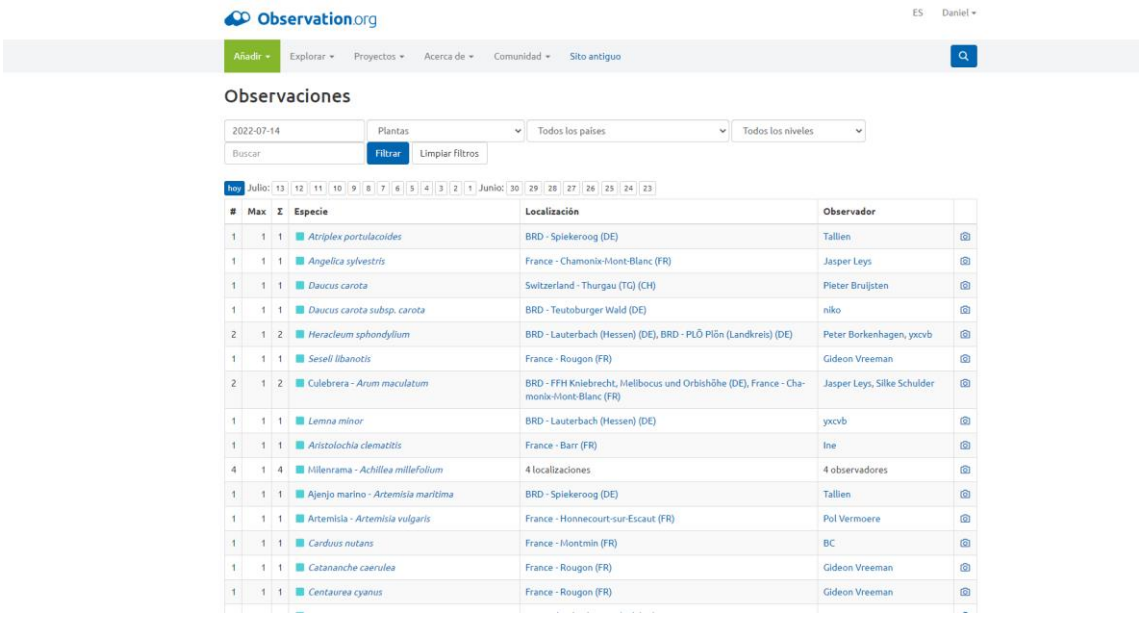

Figura 2: Observation

Por último, *iNaturalist* [7] es una aplicación web, que también tiene su respectiva aplicación móvil, que nació como proyecto final de Máster de un grupo de alumnos de la *University of California at Berkeley's School of Information*. Después, continuó desarrollándose hasta alcanzar colaboraciones con entidades como la *California Academy of Sciences* o *National Geographic Society*, creciendo como software de código abierto.

Esta aplicación posee una red de aplicaciones hijas para organizaciones especializadas, creando la sensación de una plataforma nueva pese a ser una especialización de la anterior. Más en concreto, tiene colaboraciones en España (Natusfera[1]), Argentina (ArgentiNat[2]) y muchas más que tienen su propio entorno, pero con la misma finalidad que *iNaturalist*, aunque para un dominio más concreto. La Figura 3 muestra el flujo de trabajo de la aplicación, que consta de tres pasos: ver una especie y realizarle una foto, publicarla en la aplicación para compartirla con todos los usuarios de la aplicación y poder hablar sobre las publicaciones.

[1] https://natusfera.gbif.es/
[2] https://www.argentinat.org/

Figura 3: Flujo iNaturalist, Fuente: https://www.inaturalist.org/home

Como se observa en la Figura 4 posee un menú sencillo y una primera vista que incita a seguir navegando por la página.

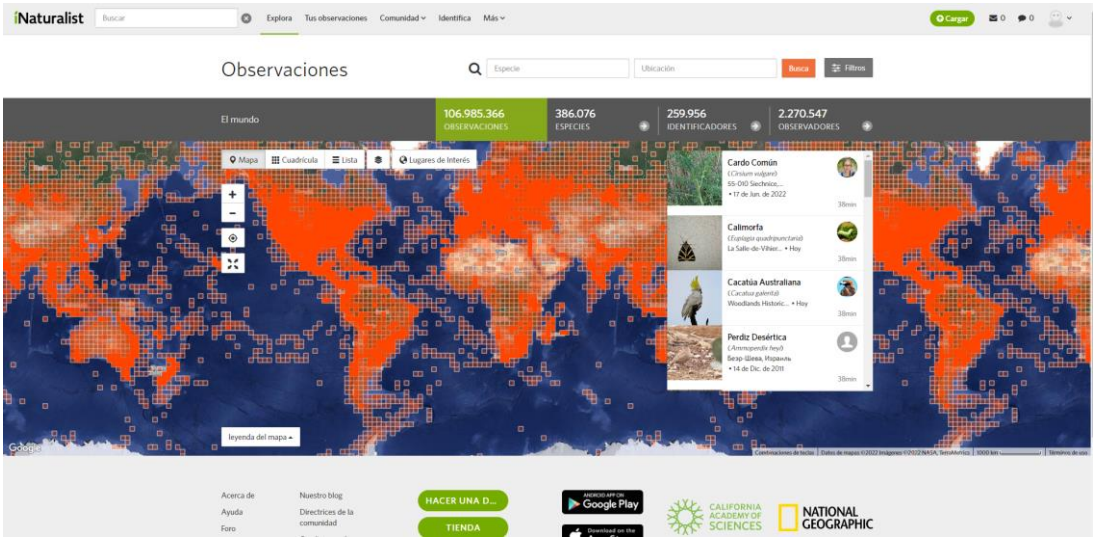

Figura 4: iNaturalist

Lo que más puede llamar la atención de la Figura 4, son los cuadraditos que se ven en el mapa, estos representan las zonas con más observaciones, y, a medida que se amplía el mapa se distinguen las observaciones con marcadores, habilitando una leyenda para refinar de qué tipo es la observación, véase la Figura 5.

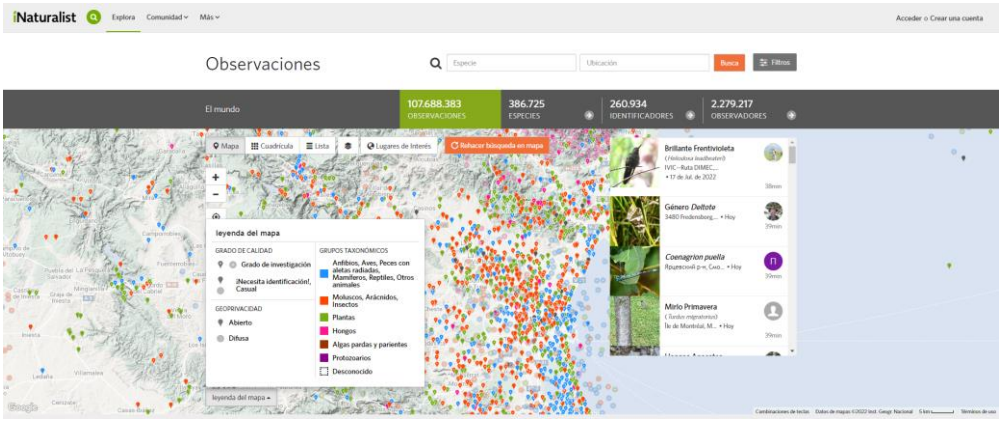

Figura 5: Observaciones iNaturalist

Cabe destacar que, de estas tres aplicaciones competidores analizadas, a la que más nos queremos asemejar es a *iNaturalist*, ya que sus interfaces son muy agradables, además de ser la más intuitiva de todas, aspecto que consideramos fundamental para que los usuarios se hagan habituales tras su primer uso.

La Tabla 1 muestra una comparativa de las aplicaciones presentadas con respecto a los objetivos expresados en el capítulo 1. Como se puede apreciar, existen algunas funcionalidades no presentes en ninguna de las aplicaciones, hueco que intentaremos llenar con nuestro trabajo.

Tabla 1: Comparativa competidores

Funcionalidades	Descripción	iSpot	Observation	iNaturalist
Registrar Observación	A partir de una imagen, permite publicarla asociándole una especie.	✓	✓	✓
Sugerir Especie	Permite opinar sobre qué especie es la de una observación.	✓	✗	✓
Ver Observaciones	Listado de observaciones pudiéndose visualizar en un mapa.	✓	✓	✓
Buscador Especies	Se puede buscar una especie por su nombre común o científico.	✓	✓	✓
Algoritmo para identificar especies de foto	Bien propio o de terceros disponer de un algoritmo que sugiera las especies sobre una imagen.	✗	✓	✓
Observación Privada	A partir de una imagen se puede identificar qué especie se ha observado quedando únicamente visible para el usuario que la ha registrado.	✗	✗	✗
Buscar Ruta	Mapa donde se puede trazar una ruta viendo qué especies se han observado cerca de ésta.	✗	✗	✗

3 Análisis y Especificación de Requisitos

En este apartado se va a describir con detalle el problema a resolver. Cabe señalar que, como se menciona en apartados anteriores, la metodología que se adopta es ágil, por lo que estos análisis varían a lo largo del desarrollo del producto, de forma que se van moldeando en lo que finalmente se convierte como la solución adoptada.

La aplicación pretende ser un punto de conexión de los usuarios con la naturaleza, de modo que se aprenda de todos los seres vivos que nos rodean, y para ello hay que conocer qué especies estamos observando. Por todo esto, nuestra aplicación va a permitir dos modalidades principales de identificación a través de una imagen. La primera de ellas será la forma 'clásica' de identificación que hay en otras aplicaciones, consistente en cargar la imagen en un servidor, para que el sistema sugiera especies al usuario que tienen posibilidades de ser la especie observada, eligiendo el usuario la que crea que es correcta. Adicionalmente, el sistema, a través de metadatos asociados a la imagen, extraerá las coordenadas de donde fue tomada la foto, así como la fecha. Una vez sacados todos estos datos, el usuario podrá editarlos a su gusto. Con esta publicación lo que se consigue es que todos los usuarios del sistema puedan ver esa observación, con lo que podrán sugerir si están de acuerdo con la especie que pone en la observación o creen que se trata de otra especie.

La segunda forma de identificar consta de un flujo parecido, donde el usuario también sube la imagen, aunque en este caso al dar de alta la identificación, el sistema asociará directamente a esta, varias especies candidatas, cada una de ellas con un porcentaje de acierto determinado, sin dejar elegir al usuario cuál cree que es la especie observada. De nuevo, la fecha y lugar de la foto se cogerá a través de metadatos. Lo que más la diferencia respecto de la modalidad anterior es que esta será privada y únicamente la podrá ver el usuario que la ha subido. A esta observación privada la llamaremos a partir de ahora identificación.

Además de poder registrar una imagen en cualquiera de las dos formas citadas, el usuario podrá trazar una ruta para sus caminatas, en la que se mostrarán qué observaciones se han publicado cerca de la zona que va a visitar.

Con las dos modalidades de identificación de una especie lo que se consigue es poder enseñar al usuario un historial de sus observaciones e identificaciones, mostrando todos sus datos asociados y un filtrado para refinar una búsqueda concreta.

Para aportar más información al usuario, se podrán consultar los datos de todas las especies registradas, así como donde han sido observadas. Y de forma parecida, se podrán consultar los usuarios del sistema, viendo su perfil y observaciones publicadas.

Como pantalla principal se observará un mapa con todas las observaciones publicadas, ofreciendo un filtrado que permita encontrar una observación determinada.

3.1 Modelo de dominio

La Figura 6 muestra el modelo de dominio en forma de diagrama de clases UML.

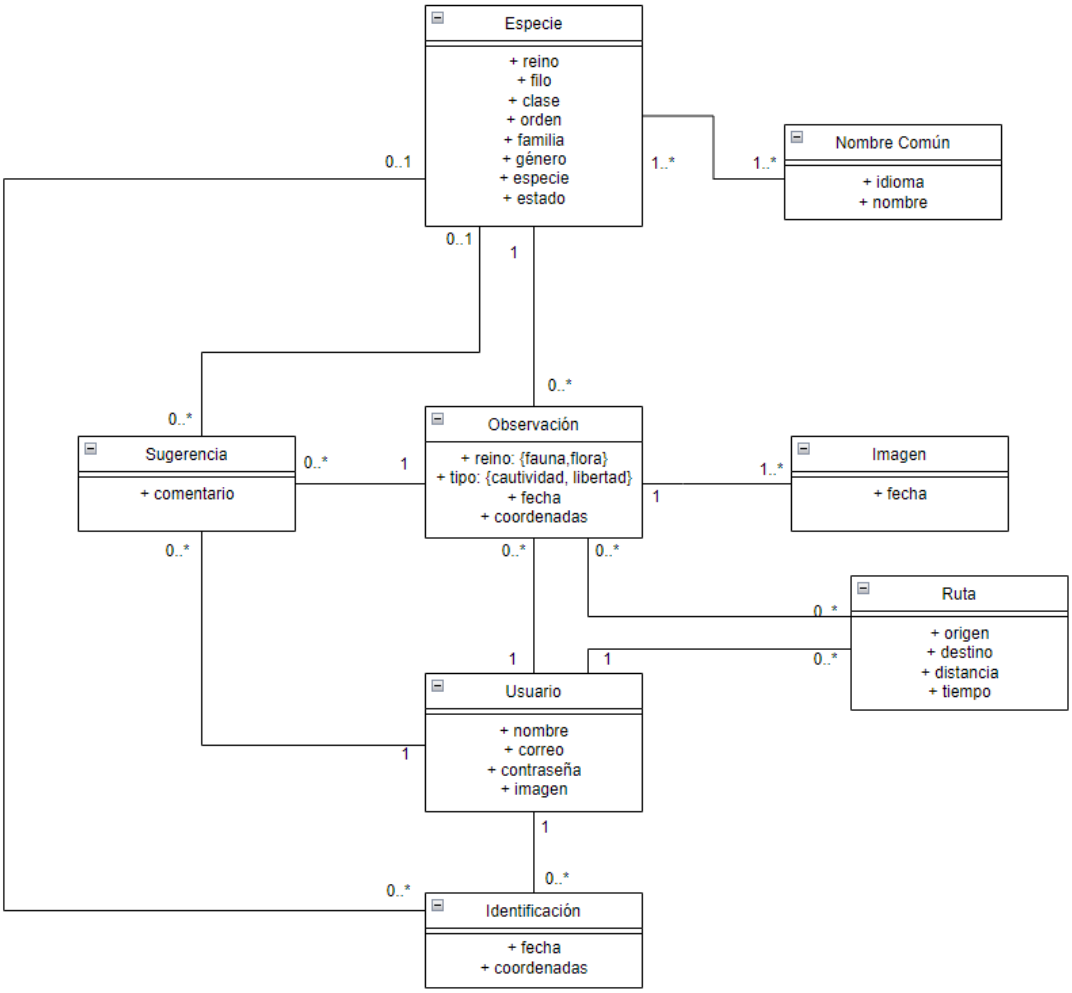

Figura 6: Modelo de dominio

En este diagrama se observa que el usuario dispone de dos formas de identificar una especie, la identificación, que solo tiene fecha, coordenadas y una relación con la especie, y la observación, donde se especifica el reino de la especie observada, si se ha visto en libertad o cautividad, fecha y coordenadas. Además, se relaciona con las rutas ya que en estas se podrán ver las observaciones de la aplicación y con las sugerencias, puesto que, al ser observaciones públicas, otros usuarios pueden ayudar a identificar la especie observada.

Sobre las especies cabe señalar que posee un estado de conservación y que todos sus demás atributos permiten clasificarlo de forma taxonómica. Adicionalmente, cada especie puede tener varios nombres comunes con los que se le conoce en diferentes lugares del mundo.

3.2 Modelo de Casos de Uso

En este apartado se van a presentar los diferentes diagramas que constituyen el Modelo de Casos de Uso, con el fin de conocer el contexto del sistema, así como todas las funcionalidades de las que se compone la aplicación.

3.2.1 Diagrama de Contexto

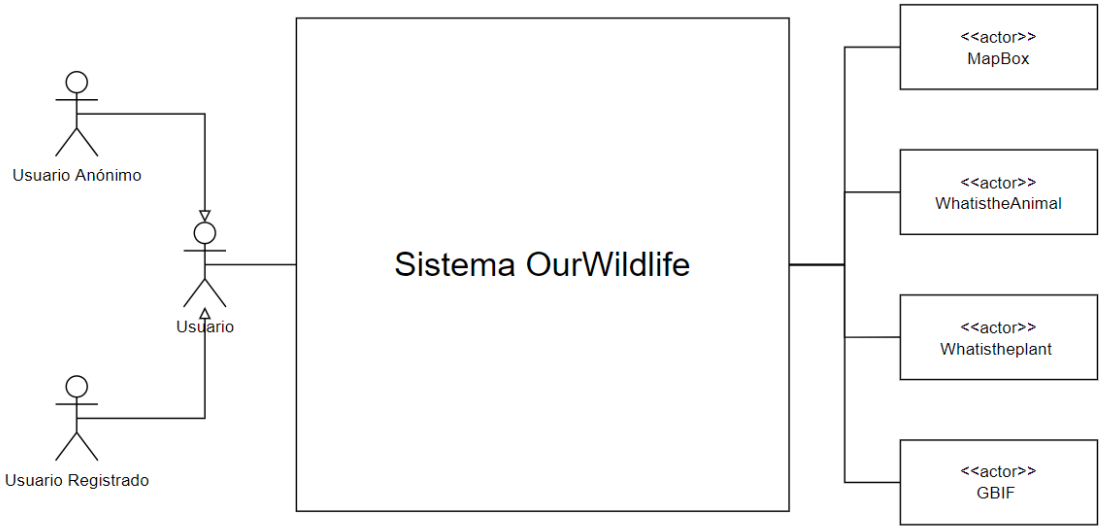

Figura 7: Diagrama de contexto

Como se puede apreciar en la Figura 7, existen tres tipos de usuarios humanos que pueden interactuar con el sistema. Por un lado, el Usuario Anónimo tendrá acceso a funcionalidades para las que no es necesario iniciar sesión, estando el resto solo disponibles para los Usuarios Registrados. Asimismo, se ha representado la herencia puesto que estos dos actores comparten ciertas funcionalidades que se agrupan en el actor Usuario, como detallaremos en el modelo de casos de uso (ver sección 3.2.2).

Además de estos tres tipos de usuario, se pueden apreciar cuatro actores externos al sistema de los que se consumirán sus servicios para desarrollar ciertas funcionalidades. Estos cuatro actores externos son:

- ***Mapbox***: permitirá mostrar mapas a lo largo de la aplicación y un buscador geográfico que facilite la búsqueda de rutas ya mencionada.
- ***WhaistheAnimal/Whatistheplant***: permitirán el reconocimiento de especies animales y vegetales, respectivamente, a través de una imagen usando inteligencia artificial.
- ***GBIF***: es una interfaz de programación de aplicaciones (API) que proporcionará todos los datos necesarios de una especie, concretamente su clasificación taxonómica.

3.2.2 Diagrama de Casos de Uso

La Figura 8 muestra el diagrama de Casos de Uso del sistema. Nótese que el actor externo *Mapbox* aparece tres veces de forma deliberada para mejorar la legibilidad del modelo.

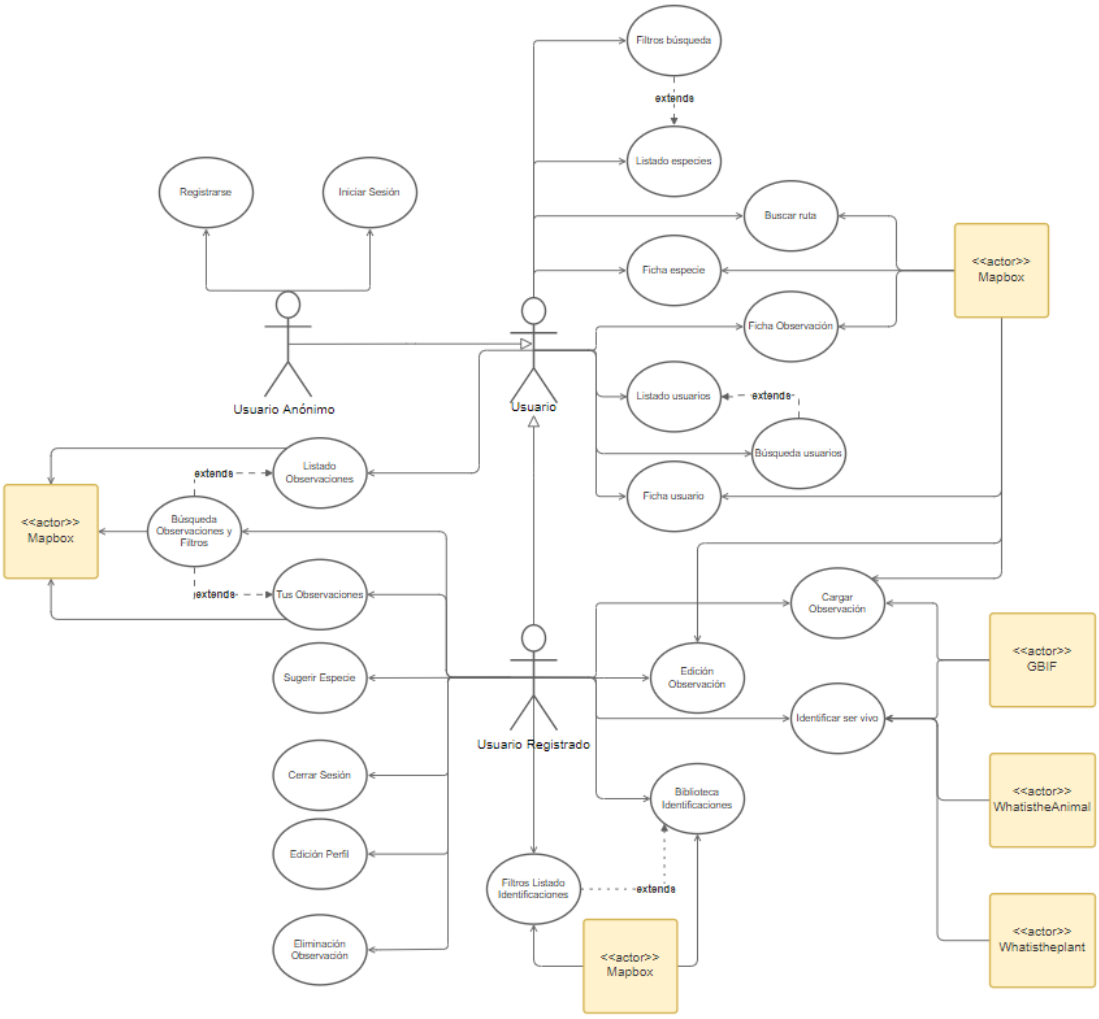

Figura 8: Diagrama Casos de uso

La Tabla 2 muestra una breve descripción de los casos de uso. La especificación completa de cada uno de ellos se encuentra en el ANEXO B: Especificación Casos de uso. Recordamos que aquellos casos en que aparece el Usuario como actor, implícitamente se incluye a sus especializaciones Usuario Anónimo y Usuario Registrado.

Tabla 2: Resumen Casos de Uso

Caso de Uso	Descripción	Actores
CU-1. Registrarse	El usuario podrá registrarse para obtener una cuenta en la aplicación web.	Usuario Anónimo
CU-2. Iniciar Sesión	El usuario podrá acceder a la	Usuario Anónimo

	aplicación a través de una cuenta.	
CU-3. Cerrar Sesión	El usuario podrá cerrar la sesión de su cuenta.	Usuario Registrado
CU-4. Identificar Ser Vivo	Se podrán subir imágenes para identificar a qué especie pertenece sin compartirlas con el resto de los usuarios de la aplicación.	Usuario Registrado, WhatistheAnimal/Whatistheplant y GBIF
CU-5. Biblioteca Identificaciones	Los usuarios registrados tendrán disponibles todas las identificaciones realizadas.	Usuario Registrado y Mapbox
CU-6. Filtros Listado Identificaciones	Se dispondrá de una opción para filtrar identificaciones según diversos criterios.	Usuario Registrado y Mapbox
CU-7. Ficha Especie	El usuario podrá ver todos los datos de una especie en su correspondiente ficha.	Usuario y Mapbox
CU-8. Listado Especie	Listado que mostrará todas las especies almacenadas en la aplicación, además de dar la posibilidad de acceder a sus datos.	Usuario
CU-9. Filtros Búsqueda	Para poder obtener un listado de especies más concreto se dispondrá de un filtro para refinar la búsqueda.	Usuario
CU-10. Buscar Ruta	El usuario podrá buscar en un mapa y ver las observaciones realizadas más cercanas.	Usuario y Mapbox
CU-11. Cargar Observación	El usuario podrá registrar una observación para compartirla con los usuarios de la aplicación.	Usuario Registrado, WhatistheAnimal/Whatistheplant,GBIF y Mapbox
CU-12. Edición Observación	El usuario podrá modificar las observaciones que haya realizado.	Usuario Registrado y Mapbox

CU-13. Eliminación Observación	El usuario podrá eliminar una observación en cualquier momento.	Usuario Registrado
CU-14. Listado Observaciones	El usuario podrá ver en forma de lista todas las observaciones registradas en la aplicación, así como su ubicación en un mapa.	Usuario y Mapbox
CU-15. Ficha Observación	El usuario podrá consultar todos los datos asociados a una observación.	Usuario y Mapbox
CU-16. Búsqueda Observaciones y Filtros	El usuario podrá buscar observaciones utilizando filtros o bien, el buscador para encontrar los de una especie en concreto.	Usuario y Mapbox
CU-17. Tus Observaciones	El usuario podrá ver en forma de lista todas las observaciones registradas por él en la aplicación, así como su ubicación en un mapa.	Usuario Registrado y Mapbox
CU-18. Sugerir Especie	El usuario podrá comentar y sugerir especies a cualquier observación publicada.	Usuario Registrado
CU-19. Listado Usuarios	Se podrán ver todas las cuentas registradas en la aplicación.	Usuario
CU-20. Ficha Usuario	Se podrá ver información de los usuarios de la aplicación.	Usuario y Mapbox
CU-21. Búsqueda Usuarios	El usuario podrá buscar por nombre de usuario una cuenta registrada.	Usuario
CU-22. Edición Perfil	El usuario podrá cambiar los datos de su perfil.	Usuario Registrado

4 Diseño de la solución

En este apartado se van a presentar los rasgos más importantes respecto al diseño de la solución, viendo la arquitectura utilizada, así como el diseño detallado de la aplicación.

4.1 Arquitectura

La arquitectura que se va a presentar es de muy alto nivel, y refleja la típica estructura de las aplicaciones Web contemporáneas.

En concreto, tratándose de una aplicación web orientada a servicios, se va a emplear el estilo REST [8] que permitirá una comunicación sin estado en la que se podrá separar muy cómodamente las diferentes capas funcionales de la misma.

Se trata de una arquitectura de tres capas donde el *frontend* implementa la capa de presentación y el *backend* implementa las capas de negocio y persistencia. Para la capa de negocio se requiere de servicios externos y la persistencia la proporciona la base de datos.

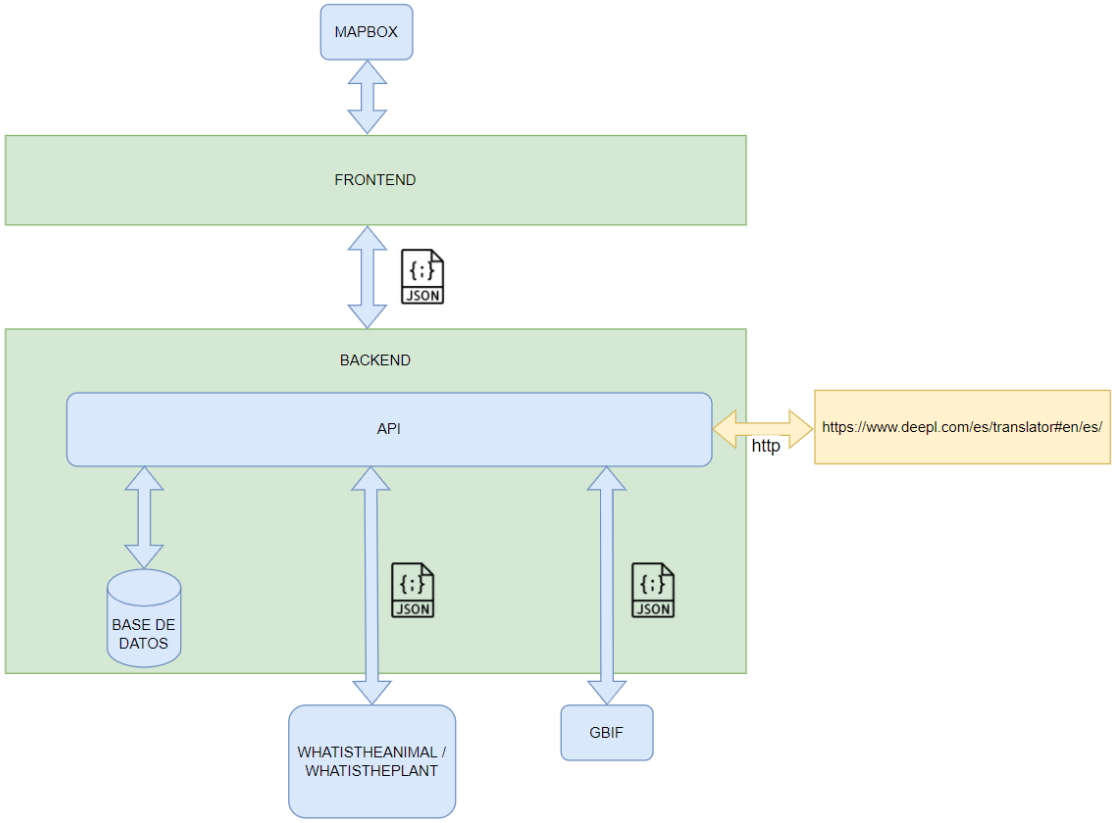

Figura 9: Arquitectura Alto Nivel

Como se puede observar en la Figura 9, la aplicación se dividirá en un *frontend* y un *backend* separados. Una API REST gestiona la comunicación entre ambos.

Dentro del *backend* hay conexiones con otros servicios externos (*Whatistheanimal*, *Whatistheplant* y *GBIF*) para obtener datos más concretos, además de hacer un *scrapper* en el caso de que estos servicios no proporcionen el nombre común en castellano de una especie.

4.2 Diseño Detallado

El diagrama de clases que se observa en la Figura 10 es el que se ha seguido a la hora de implementar la estructura en la base de datos.

En esencia, es un refinamiento del modelo de dominio (Figura 6) que representa las entidades y las relaciones entre estas.

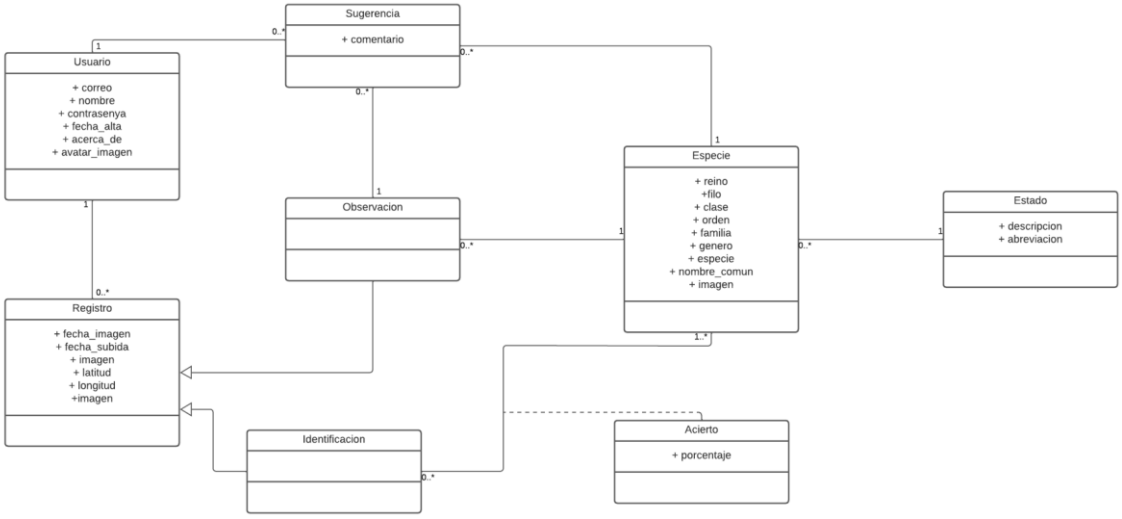

Figura 10: Diagrama de clases

En la Figura 10 se puede observar cómo el usuario puede sugerir especies a una observación o realizar un registro, diferenciando entre observación e identificación según la finalidad de este. Adicionalmente, como se ha explicado en el análisis y especificación de requisitos, una modalidad de registrar observaciones era hacerlas privadas al resto de usuario de la aplicación, llamadas identificaciones, y, en ellas es un servicio externo el que sugiere la especie con el porcentaje más alto que concuerda con la imagen aportada.

4.3 Diseño de la interfaz

Una técnica muy utilizada para validar los casos de uso especificados es el diseño de prototipos que, además, de forma visual, ayudarán al desarrollador a saber cómo programar la parte *frontend* de la aplicación, entendiendo todo el flujo de esta y viendo un resultado previo al desarrollo.

Se han diseñado todas las interfaces que componen la aplicación, pero en este apartado se van a exponer solo algunas de ellas, estando todas ellas recogidas en el ANEXO C: Diseños de las interfaces.

Como la aplicación gira entorno a las observaciones se van a mostrar los diseños de las interfaces relacionadas con la carga de observaciones y de identificaciones. Estos dos modos de subir una imagen tienen una ventana de carga similar diferenciándose solo cuando el sistema ya ha procesado la imagen. A esta ventana de carga se accederá a través de unos botones ubicados en la barra de navegación que por su nombre diferencian que acción hace cada uno de los dos.

En la Figura 11 se observa la ventana de carga a través de la cual se subirá la imagen deseada y se procederá, bien a mostrar el formulario de una observación si se ha accedido a través de su correspondiente botón, bien a cargar la identificación si se ha elegido esa opción desde la barra de navegación.

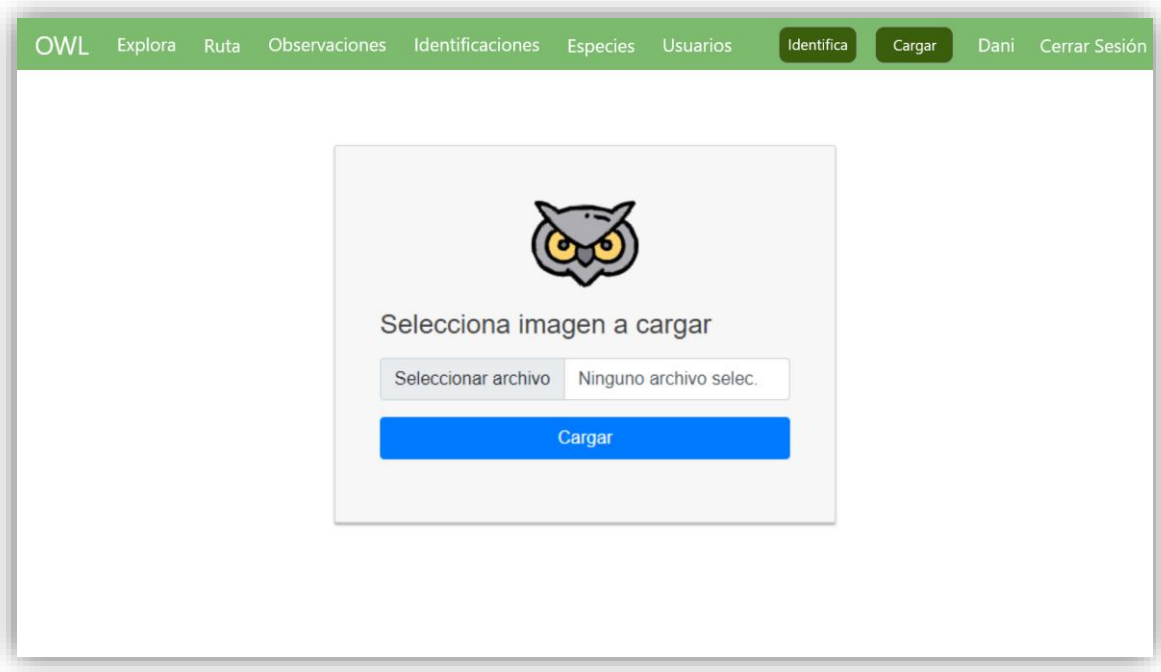

Figura 11: Diseño pantalla de carga

Una vez procesada la imagen cada uno de ellos redirigirá a una de estas dos pantallas según sea el caso. Si se trata de una observación mostrará el formulario de la Figura 12, en cambio si es una identificación la dará de alta mostrando su correspondiente ficha, como muestra la Figura 13.

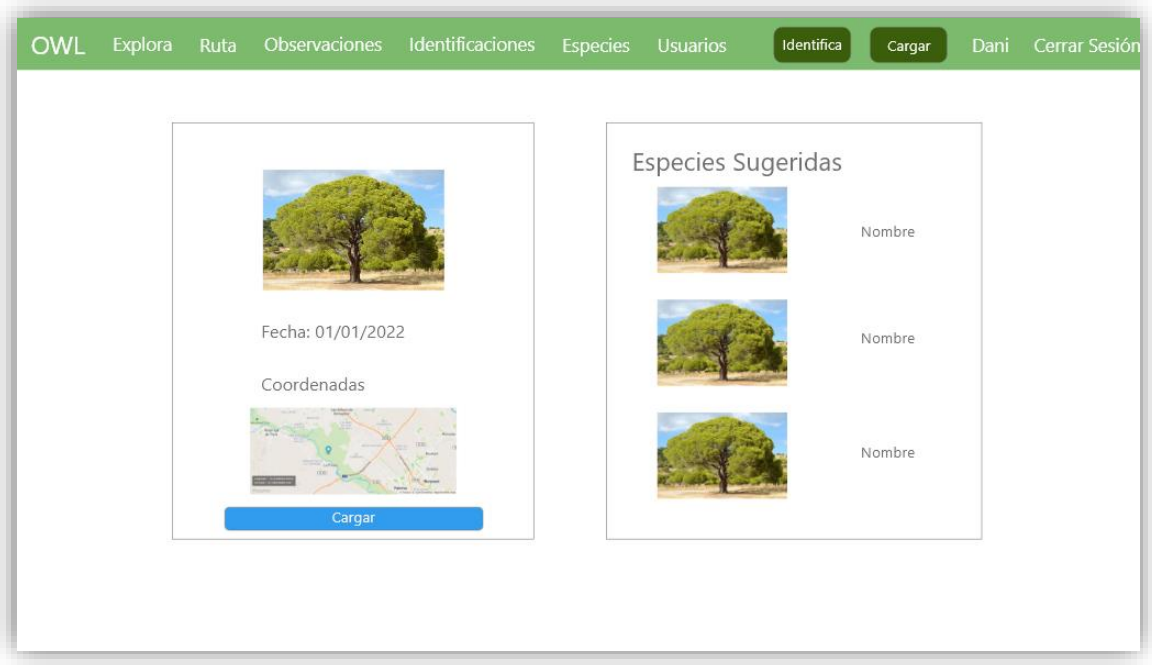

Figura 12: Formulario de una observación

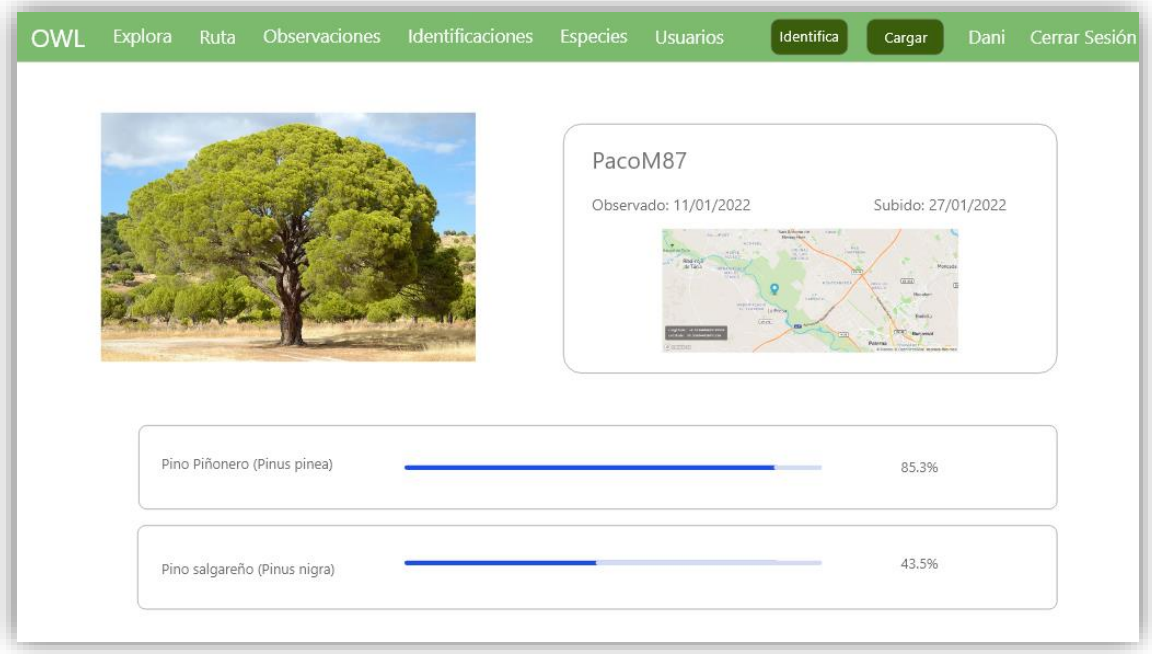

Figura 13: Ficha Identificación

5 Selección de la Tecnología

Sobre la tecnología utilizada se va a hacer una diferenciación entre las dos partes implementadas, en las que se usan diferentes herramientas de trabajo.

5.1 Backend

En esta parte se ha optado por usar tecnologías ya conocidas, lo que nos facilitará la creación del esqueleto de la aplicación.

5.1.1 Tecnologías

Java: se ha elegido este lenguaje ya que es el que más hemos aprendido durante los estudios de grado. Además, es uno de los lenguajes de programación más utilizados y con una muy buena documentación. Para emplear este lenguaje se utilizó la herramienta **Spring Boot** [9]**,** que facilita y acelera la creación de proyectos que vayan a utilizar el *framework Spring* [10]. Incluye un servidor embebido, del mismo modo que facilita la incorporación de nuevas librerías al proyecto.

Adicionalmente, en diversas asignaturas se hicieron desarrollos con este *framework* y resultó ser bastante sencillo de utilizar. Como ventajas de la elección de esta herramienta se podrían destacar:

- La rapidez con la que se puede desarrollar una API REST.
- Tiene una gran comunidad de usuarios y, por lo tanto, es muy fácil encontrar soluciones y soporte a posibles errores o dudas durante la implementación.
- *Spring* permite la creación de la base de datos a partir de la creación de las entidades con anotaciones de este *framework*; por lo tanto, ahorra esa tarea de crear explícitamente en SQL la base de datos.

SQL: se ha elegido este lenguaje de consulta para bases de datos relacionales por ser el que se ha estudiado durante los estudios de grado. Como sistema de gestión de base de datos se ha utilizado **MySQL** [11]**,** sistema relacional que se basa en el lenguaje de consulta SQL y nos permite almacenar toda la información necesaria en una base de datos local.

5.1.2 Bibliotecas Spring

Como se ha comentado, *Spring* ofrece la posibilidad de incorporar múltiples bibliotecas a nuestro proyecto con distintas finalidades. Las que se han utilizado son:

- **Spring Data JPA** [12]**:** es un módulo que nos permite generar un *CRUD* de las entidades de manera muy rápida y fácil.
- **Spring Starter Web** [13]**:** nos facilita la creación de aplicaciones web usando API REST o Spring MVC (Modelo-Vista-Controlador).
- **Spring Starter Security** [14]**:** módulo que permite dotar de seguridad los métodos expuestos por la API.

- **Lombok** [15]**:** biblioteca de java que permite ahorrar la escritura de código y mantenerlo lo más limpio posible mediante anotaciones en las clases.
- **Gson** [16]**:** biblioteca de código abierto que permite la serialización y deserialización de objetos Java, para representarlos en la notación *JSON*.
- **Selenium** [17]**:** nos permite hacer *screen scraping* para extraer información de las páginas web que se consideren oportunas. Se ha utilizado para un caso muy particular que se comentará en el apartado Desarrollo de la aplicación.
- **Cloudinary** [18]**:** es un servidor web que permite el almacenamiento de imágenes y así no almacenarlas directamente en nuestra base de datos.
- **JWT** [19]**:** es un estándar que nos permite dotar de seguridad, a través de tokens, el acceso a ciertos apartados de nuestra aplicación.
- **Metadata-extractor** [20]**:** nos va a permitir acceder a los metadatos de imágenes y vídeos.

5.1.3 Herramientas

IntelliJ IDEA [21]**:** es un entorno de desarrollo que nos proporciona todo lo necesario para desarrollar de manera cómoda y su modo de depuración de código**DepuraciónDepuraciónDepuración** es especialmente agradable.

XAMPP [22]**:** nos ha permitido alojar la base de datos en la que guardamos la información, así como proporcionarnos mediante *phpMyAdmin* una fácil gestión de los datos almacenados.

Postman [23]**:** es una aplicación que nos permite hacer llamadas a *endpoints* y, por lo tanto, ir comprobando el desarrollo de la API REST sin esperarse a desarrollar el *frontend*.

5.2 Frontend

En esta parte es donde se han aprendido tecnologías nuevas y, por tanto, la que ha costado más de desarrollar.

5.2.1 Tecnologías

Vue JS [24]**:** es un *framework* de *Javascript* que combina las tecnologías básicas del *frontend*: {HTML, CSS y Javascript}, para ofrecer una convención de trabajo más rápida y sencilla.

Es una tecnología de la que no se tenían conocimientos y, por ello, se han seguido diversos cursos para aprenderlo. Se optó por este *framework*, y no por otros, porque su curva de aprendizaje es mucho más rápida, siendo uno de los más utilizados actualmente.

Axios [25]**:** es un cliente HTTP que nos permite hacer peticiones y obtener respuestas de estas para facilitar esa comunicación *frontend-backend*.

5.2.2 Librerías utilizadas

Para realizar el desarrollo *frontend* se han utilizado las siguientes librerías:

- **Vuex** [26]**:** es una librería que se encarga de gestionar los estados para aplicaciones Vue JS.
- **Vue-Router** [27]**:** es un sistema de Vue JS que permite crear y gestionar las *URLs* que atacará la aplicación.
- **Bootstrap** [28]**:** es una biblioteca que permite un sencillo y rápido diseño de elementos web.
- **Mapbox** [29]**:** es un proveedor de mapas en línea que además ofrece funcionalidades como un buscador de lugares y trazado de rutas.

5.2.3 Herramientas

Visual Studio Code [30] es un entorno de desarrollo con una interfaz muy amigable que nos proporciona todo lo necesario para nuestro desarrollo *frontend*.

Adobe XD [31]: es un programa que nos ha servido para realizar los bocetos de las pantallas que se iban a desarrollar en la aplicación.

5.3 Control de Versiones

Para el control de versiones, y poder manejar los cambios que se producen durante el desarrollo de ambos proyectos, *frontend* y *backend*, se ha utilizado **Github** [32], tanto en su versión web, para crear el repositorio y monitorizar el estado de los proyectos, como su versión en escritorio, vinculada a los proyectos locales para subir los cambios de código realizados y mantener un buen flujo de trabajo.

6 Desarrollo de la aplicación

Este apartado se va a dividir en dos partes, la primera dedicada a la implementación del *backend* y la segunda, del *frontend*. En ambos subapartados se detallarán cómo se ha pasado de la especificación de requisitos a su implementación, documentando todos los problemas y dificultades que han ido surgiendo durante el desarrollo de la aplicación.

6.1 Backend

El primer paso que se dio a la hora de implementar el *backend* fue crear la base de datos. Para ello, se recurrió a las anotaciones que proporciona *Spring*, que permiten generar la estructura de la base de datos simplemente implementando las clases básicas y anotándolas como es debido.

Sobre esta parte de la implementación solo surgió una dificultad: seleccionar qué estrategia abordar a la hora de implementar la herencia en Java, más en concreto entre las entidades que se observan en la Figura 14.

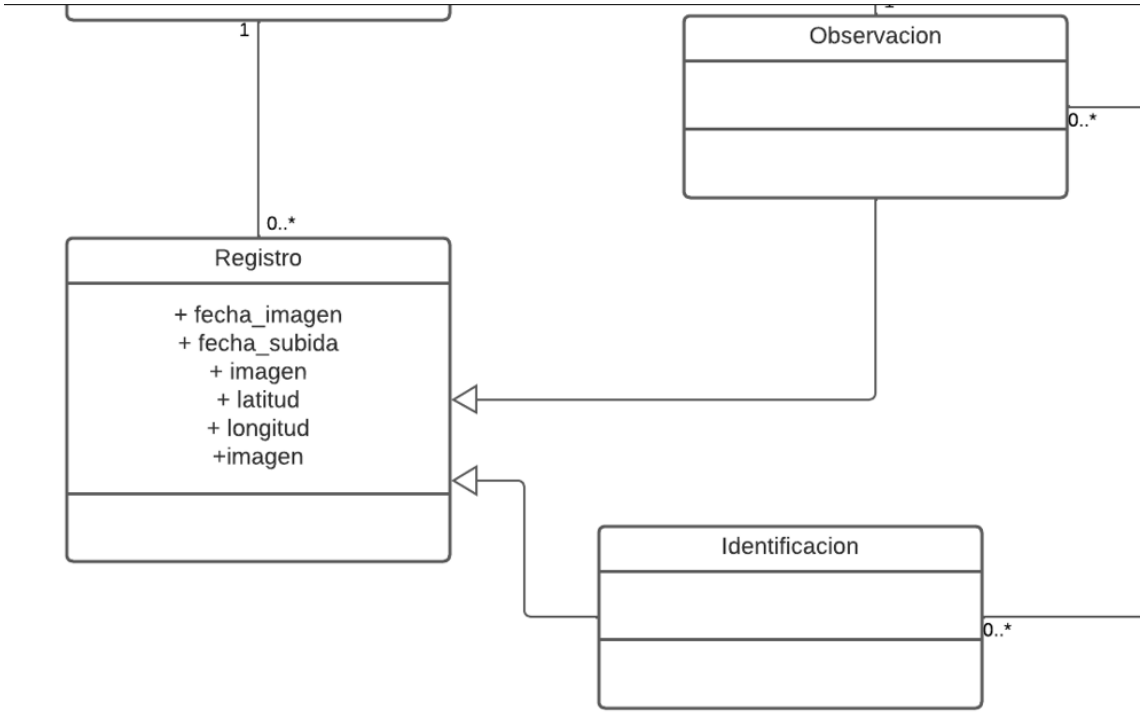

Figura 14: Herencia relevante

Investigando sobre cómo implementar la herencia con *JPA*, módulo de *Spring* que nos facilita la gestión de las entidades, se encontraron las siguientes alternativas [33]:

- **Mapped Superclass:** esta estrategia lo que hace es ignorar la jerarquía y, por tanto, la relación entre la clase padre (*Registro*) y el usuario no se podría llevar a cabo.

- **Single Table:** esta estrategia crea una tabla para todos los hijos, diferenciándolos con un atributo (columna en la base de datos) que indicará de qué hijo se trata. Es una estrategia que ofrece un gran rendimiento y no requiere de operaciones muy complejas. Aunque un inconveniente es que puede llegar a ser muy grande en cuanto a volumen de datos almacenados.
- **Table per class:** en esta estrategia se ignora que las clases hijas mantengan una relación de herencia por lo que hay que realizar una operación de unión entre todas las tablas para saber el tipo del hijo.
- **Joined Table:** en esta estrategia crea una tabla para cada entidad, lo que hace que al guardar uno de los hijos haya que almacenarlo también en la tabla del padre.

Con todas estas alternativas posibles, se eligió adoptar la estrategia **Single Table**, por su fácil implementación y rendimiento.

Una vez hechas todas las entidades que conformarán la base de datos, se procedió a implementar los servicios y accesos a los datos almacenados de cada una de ellas.

En las clases que gestionan el acceso a la base de datos, lo que se hizo es extender de *CrudRepository*, interfaz que nos genera los métodos básicos para crear, eliminar, encontrar y guardar la entidad sobre su tabla en la base de datos. Además, estas clases son las que establecen la comunicación con la parte de la persistencia, base de datos.

Por otro lado, los servicios se dividieron en dos partes, una con las interfaces que declaran los métodos que van a implementar y la otra con las implementaciones de las mismas. Con esto, conseguimos tener una estructura ordenada, poder realizar implementaciones diferentes y tener una mayor seguridad, pues exponiendo la interfaz no se está mostrando el código de cómo se implementan las funcionalidades que se declaran en ésta.

Además, en los servicios se hizo uso de las clases antes mencionadas para establecer ese puente de comunicación entre la base de datos y la lógica que se implementa en esta parte.

La implementación de la API se ha dividido en distintas clases que expondrán las funcionalidades para ser consumidas en la parte *frontend*. Como se puede ver en la Figura 15, la API está formada por diversas clases, cada una de ellas exponiendo diferentes funcionalidades relacionadas con el nombre que estas clases tienen.

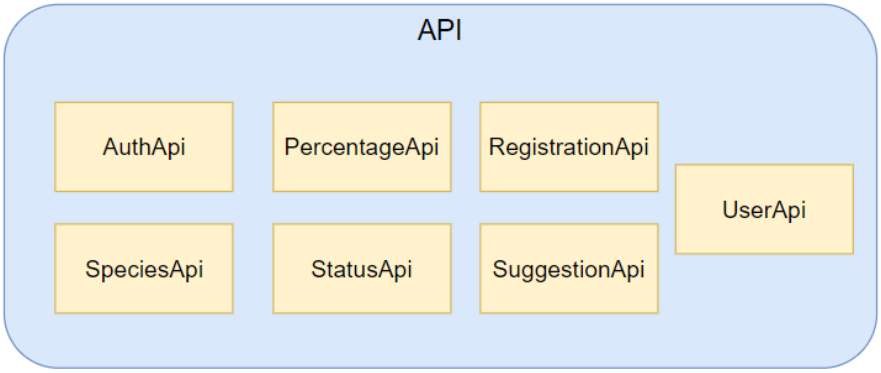

Figura 15: Estructura de la API

En la Tabla 3 se observan las clases que se han creado, detallando las *urls* a las que deberá atacar la interfaz para obtener los datos y una pequeña descripción de las funcionalidades que albergan estas clases.

Tabla 3: Clases que conforman la API

Clase	Descripción	Url
AuthApi	Permite realizar el inicio de sesión y el registro en la aplicación	/api/auth
PercentageApi	Permite obtener los porcentajes de una identificación, así como cuál es el de mayor acierto.	/percentage
RegistrationApi	Gestiona todo acerca de las identificaciones y observaciones, es decir, su cargado, edición, eliminación …	/registration
SpeciesApi	Permite obtener todas las especies disponibles en la aplicación, así como su registro y filtrado	/species
StatusApi	Expone las funcionalidades de obtener y filtrar los estados de conservación de las especies.	/status
SuggestionApi	Permite publicar y borrar sugerencias asociadas a observaciones, al igual que obtenerlas.	/suggestion
UserApi	Permite obtener todos los usuarios de la aplicación, realizar una búsqueda o actualizar el perfil de una cuenta.	/user

Durante la implementación de estas clases es donde se utilizaron los servicios de terceros mencionados en el diagrama de contexto. Respecto a ellos, cabe señalar que surgió un pequeño problema relacionado con las fuentes *Whatistheanimal* y *Whatistheplant*, de donde se iba a identificar la especie a través de una imagen.

Como se menciona en apartados anteriores, para el reconocimiento de una especie se necesita echar mano de programas de terceros, por tanto, al analizar el problema se buscó un software que se adaptara a las necesidades y se encontraron los sitios web *Whatistheanimal* y *Whatistheplant*, que identifican plantas y animales. El inconveniente vino a la hora de la programación cuando, de repente, tras verificar su funcionamiento meses atrás, las páginas siempre se quedaban colgadas. Tras un contacto por correo que no contestaron, se decidió buscar una solución y, la que pensamos que era la más adecuada era, utilizar una API llamada *PlantNet* que solo reconoce plantas. Este cambio de fuentes afectó tanto al diagrama de contexto como a los casos de uso, donde se cambia la API con la que interactuar.

Además de esta fuente, en el *backend* se utilizaron otros dos servicios externos:

- **GBIF:** nos permite obtener el estado de conservación de una especie y también, su nombre común en castellano.
- **Cloudinary:** durante la especificación de requisitos no se tuvo en mente utilizar este servicio, pero llegada la hora de la implementación se usó debido a su gran utilidad. Este servicio permite almacenar en la nube las imágenes, ahorrándonos espacio en la base de datos.

Añadido al problema de las fuentes comentado, en las clases que conforman la API cabe destacar el flujo que se sigue para dar de alta una especie, pues no todas las especies disponen de todos sus datos en *GBIF*, sino que el nombre común por el que se reconoce a las especies, a veces no se encuentra en castellano. Por ello, se utiliza la librería *Selenium* haciendo una consulta a un traductor en línea del que se extrae la traducción al instante.

Por último, para dotar de seguridad a la aplicación se ha utilizado el estándar *JWT*, además de las anotaciones que ofrece la dependencia de *Spring Security*.

Con *JWT* se consigue que el acceso a la aplicación esté protegido y se realice a través de *tokens*, que se asocian al usuario en el momento del inicio de sesión y tienen un plazo de caducidad, a partir del cual dejan de ser válidos. Además, estos *tokens* se generan de forma aleatoria y cada vez que se cierra sesión, se eliminan para dejar de ofrecer acceso a la web.

Esto, combinado con las anotaciones de *Spring Security*, que permiten determinar qué tipo de usuarios pueden realizar ciertas operaciones, permitirá disponer de una aplicación dotada de seguridad y distintos tipos de usuarios, registrados y anónimos.

Todo este desarrollo *backend* fue siendo verificado a la par que implementado utilizando *Postman*, herramienta que permite consumir nuestra API para ver si están funcionando bien, sin tener que esperar a desarrollar la interfaz. Además de poder ver el formato que seguirán nuestros documentos *JSON* a la hora de manejarlos en el *frontend*.

6.1.1 Especificación de la API

Una buena documentación es igual o más importante que la propia API, ya que esta documentación va a hacer que sea más fácil de mantener a lo largo del tiempo, definiendo los servicios disponibles, cómo se utilizan y cómo responden.

La herramienta que se ha utilizado es *Swagger* [34], apoyándose en *Swagger UI* para la interfaz que mostrará la especificación de la API y en la dependencia *Spring-Fox*, que permite generar la documentación en formato *Swagger*.

Swagger permite documentar una API de forma muy rápida y sencilla, pues solo es necesario añadir unas dependencias al proyecto y configurar una clase de arranque para que todo funcione correctamente [35]. Ahora bien, al disponer de autenticación para la utilización de ciertos servicios, esa seguridad también hay que abordarla en la documentación. Por ello, la configuración ha sido algo más complicada, pues se ha tenido que poner a disposición un inicio de sesión que habilite aquellos servicios que requieran tener la sesión iniciada, todo esto consultado a través de documentación muy detallada [36].

Como resultado se obtiene una interfaz íntegramente dedicada a la especificación de la API con un entorno de prueba, véase la Figura 16.

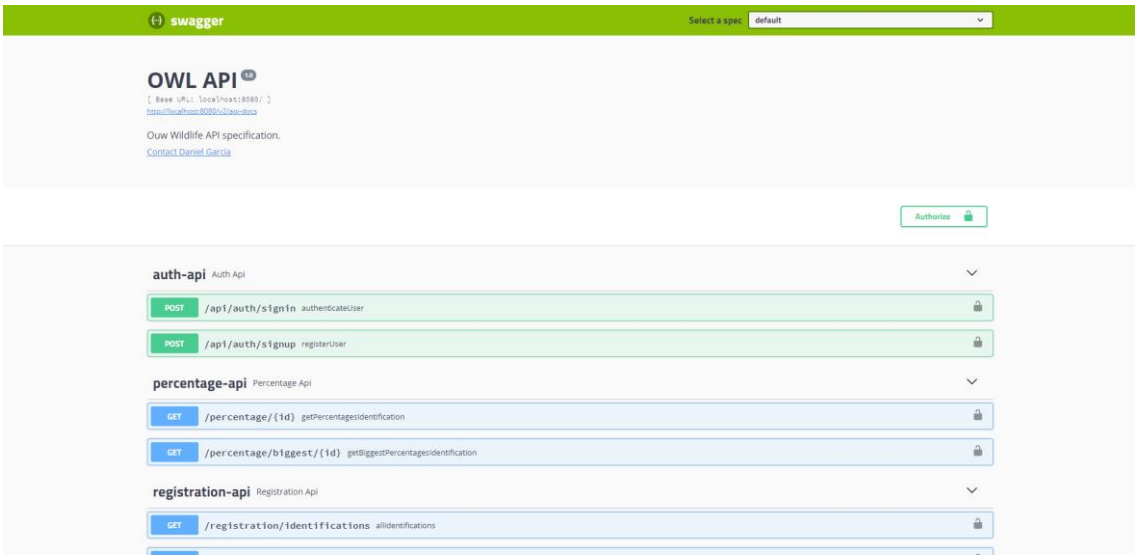

Figura 16: Documentación API en Swagger

6.2 Frontend

Para la programación de esta parte se utilizó el *framework* Vue JS combinado con *Axios* para realizar las llamadas a la API que se ha desarrollado en el *backend*.

La estructura que se ha seguido es la siguiente:

- **Componentes**: encapsulan el comportamiento y la apariencia de una parte de la interfaz, siendo reutilizable en distintas pantallas.
- **Servicios**: implementan métodos donde se realizan las llamadas al *backend* mediante *Axios*.
- **Main js**: archivo donde se inicializa la aplicación indicando los servicios a utilizar y librerías de estilos como *Bootstrap* y *FontAwesomeIcons*.

- **Router js**: archivo donde se definen todos los componentes a utilizar en la web y se les asocia una ruta que es la que se mostrará en el navegador.

Al desarrollar tanto *backend* como *frontend* en el mismo ordenador, a este último se le ha puesto en la configuración de arranque un puerto diferente para así, tener ambos proyectos abiertos sin causar conflictos.

Dentro de los componentes mencionados en la estructura del proyecto, se sigue la siguiente estructura:

- En primer lugar, el HTML con los componentes que conformarán la vista.
- En segundo lugar, el comportamiento que tendrá esa vista, definido en *Javascript*.
- Por último, la hoja de estilo CSS que aportará estilos a los componentes HTML definidos en un apartado superior.

La programación de esta parte de la aplicación ha sido la más costosa debido al desconocimiento del lenguaje previo al trabajo. Por ello, se visualizó un tutorial práctico [37] con el fin de coger soltura y poder lanzarse a programar todas las vistas.

Durante el desarrollo de los componentes las mayores dificultades que se encontraron fueron dos, cada una de ellas relacionada con una librería distinta.

La primera de ellas se encontró al utilizar *Mapbox* para mostrar mapas dentro de la web, pues para combinar las observaciones con el mapa se iba a seguir un tutorial [38] de la propia librería, adaptándolo a nuestras necesidades. Seguir este tutorial y adaptarlo a un lenguaje que no se manejaba muy bien fue difícil, pero una vez conseguido se obtuvo un buen nivel de conocimiento, haciendo más fácil y rápido el desarrollo de las siguientes vistas.

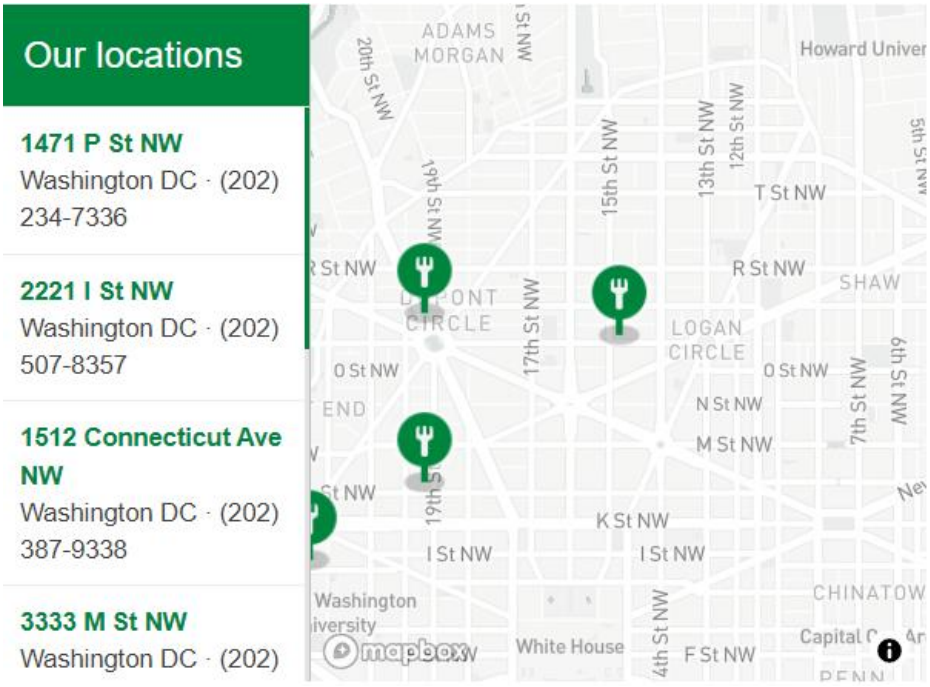

Figura 17: Resultado del tutorial Mapbox

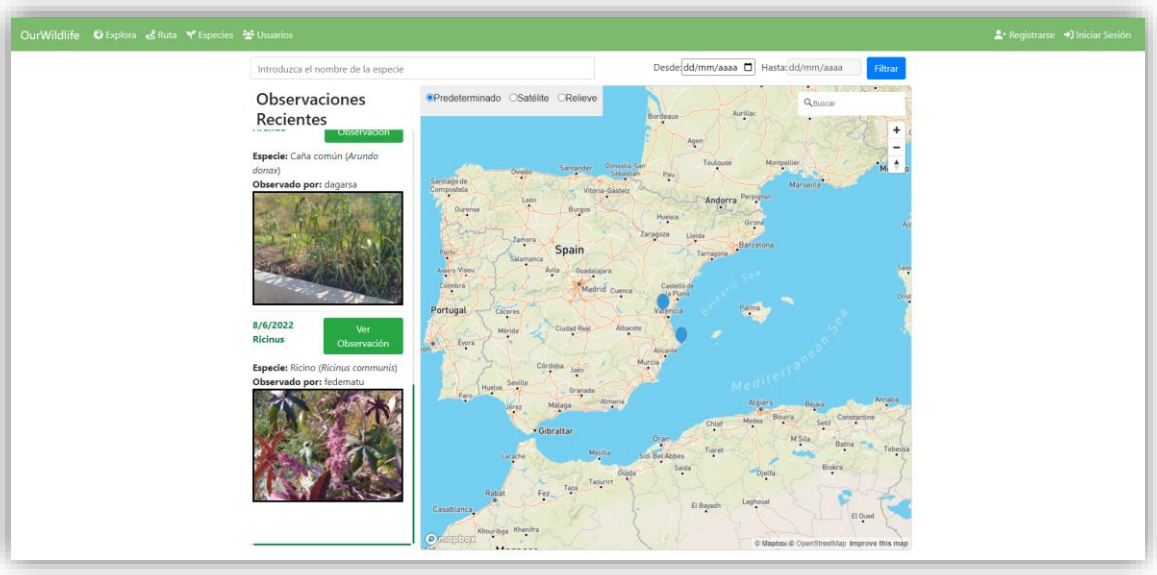

Figura 18: Resultado interfaz tras aplicar tutorial Mapbox

La segunda dificultad encontrada fue durante la utilización de *Axios*, pues según el tipo de petición que se formule los parámetros de la llamada cambian. Como *Javascript* no ofrece ayuda mientras se programa, este error costó de encontrar por desconocimiento de la herramienta. Pero una vez leída la documentación [39] con más detenimiento y tras consultar algunos foros, se entendió la causa y cuál era la solución; concretamente el causante de los problemas era el orden de los parámetros. A continuación, se detallan las declaraciones de cada una de las llamadas que se han utilizado.

- **GET *axios.get(url[, config])*:** indica que el *token* del usuario registrado se debe pasar como segundo argumento de la función.
- **POST *axios.post(url[, data[, config]])*:** indica que el *token* del usuario registrado se debe pasar como tercer argumento de la función y como segundo argumento los datos, por ejemplo, la imagen que se debe dar de alta en distintos formularios.
- **DELETE *axios.delete(url[, config])*:** sigue la misma estructura que la petición ***get***, pasando como segundo argumento la autorización que indica que la sesión se encuentra iniciada.
- **PUT *axios.put(url[, data[, config]]):*** sigue la misma estructura que la petición ***post***, pasando como segundo parámetro los datos y como tercero la cabecera que indicará si se ha iniciado sesión.

Con estas dos dificultades solucionadas, no hubo más problema a la hora de implementar las interfaces, simplemente quedaba dedicar tiempo a que el diseño cumpliera todas las funcionalidades especificadas en el análisis del problema.

7 Pruebas

En este apartado se van a explicar las técnicas utilizadas para validar el software desarrollado, implementando pruebas unitarias que validen cada módulo de la aplicación y pruebas de validación para verificar si se han cumplido los casos de uso especificados.

7.1 Pruebas Unitarias

Las pruebas unitarias se realizan para validar el correcto funcionamiento de pequeñas unidades de código que realizan una funcionalidad muy concreta. Para su desarrollo se han utilizado dos herramientas:

- ***JUnit*** [40]: permite automatizar estas pruebas para validar rápidamente si el comportamiento de la funcionalidad es correcto o no.
- ***Mockito*** [41]: permite simular el comportamiento de métodos que en el entorno de *test* no son accesibles. Un ejemplo sería cuando se consultan los datos de una tabla de la base de datos, como estos datos no son accesibles desde el entorno de *test* y la consulta siempre se realiza, habría que falsear esos datos para establecer un resultado a la llamada, y precisamente es lo que se consigue con esta herramienta.

Estas pruebas son de mucha utilidad cuando se vayan a realizar cambios en estas funcionalidades, pues para verificar que sigue funcionando como antes solo hay que lanzar la prueba correspondiente y esperar el resultado, no siendo necesario comprobar en la interfaz esa misma funcionalidad. Con esto se consigue un ahorro de tiempo enorme, al mismo tiempo que tener un control de si todas las funcionalidades básicas están funcionando correctamente.

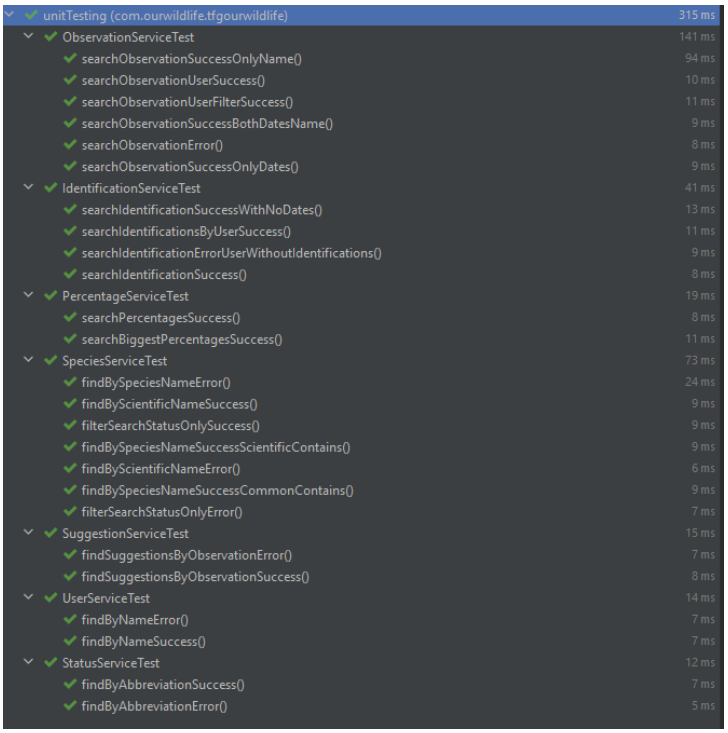

Figura 19: Resultado ejecución Tests Unitarios

En la Figura 19 se pueden observar todas las pruebas lanzadas, obteniendo un resultado positivo, evaluando los casos en que existe un flujo normal de la funcionalidad y si es necesario comprobando los flujos alternativos o los que conducen a errores. También se puede ver que se han probado los servicios que cada entidad dispone en el *backend.*

7.2 Pruebas de validación

Estas pruebas van a verificar que los casos de uso definidos durante el análisis y especificación de requisitos se cumplen. Como se observa en la Tabla 4 sí se ha conseguido cumplir con todos los casos de uso especificados.

Tabla 4: Validación casos de uso

Caso de Uso	Postcondición	¿Se cumple?
CU-1. Registrarse	Se crea un usuario con los datos rellenados en el formulario	Sí
CU-2. Iniciar Sesión	Se accede a la página web con la cuenta correspondiente.	Sí
CU-3. Cerrar Sesión	La sesión que estaba iniciada se cerrará.	Sí
CU-4. Identificar Ser Vivo	Se registra una identificación privada al usuario que la sube.	Sí
CU-5. Biblioteca Identificaciones	Se muestran todas identificaciones realizadas en su cuenta.	Sí
CU-6. Filtros Listado Identificaciones	Se muestran solo las identificaciones que coinciden con el criterio a filtrar.	Sí
CU-7. Ficha Especie	Se muestran los datos de la especie.	Sí
CU-8. Listado Especie	Se muestran todas las especies almacenadas.	Sí
CU-9. Filtros Búsqueda	Se muestran solo las especies que coinciden con el criterio a filtrar.	Sí

CU-10. Buscar Ruta	Se mostrarán las observaciones junto la ruta marcada.	Sí
CU-11. Cargar Observación	Se registra una observación con los datos rellenados en su formulario.	Sí
CU-12. Edición Observación	Se actualizan los campos de una observación.	Sí
CU-13. Eliminación Observación	Se elimina la observación.	Sí
CU-14. Listado Observaciones	Se muestran todas las observaciones registradas.	Sí
CU-15. Ficha Observación	Se muestran los datos de una observación.	Sí
CU-16. Búsqueda Observaciones y Filtros	Se muestran las observaciones que coincidan con el criterio de búsqueda.	Sí
CU-17. Tus Observaciones	Se muestran todas las observaciones registradas por el usuario en cuestión.	Sí
CU-18. Sugerir Especie	Se guarda la sugerencia asociada a la observación correspondiente.	Sí
CU-19. Listado Usuarios	Se muestran todos los usuarios registrados en la aplicación.	Sí
CU-20. Ficha Usuario	Se muestra la información del usuario.	Sí
CU-21. Búsqueda Usuarios	Se muestran los usuarios que coinciden con el nombre introducido en el buscador.	Sí
CU-22. Edición Perfil	Se guardan los cambios del usuario.	Sí

8 Conclusiones

El principal objetivo del trabajo era poder desarrollar una página web que permitiera ampliar conocimientos sobre la naturaleza que nos rodea. Dicho objetivo se considera cumplido, pues realizando las pruebas que validaban las funcionalidades se han conocido un montón de especies locales que antes se desconocían. Además de este conocimiento adquirido, se han aprendido tecnologías nuevas, así como el proceso de cómo se da vida a una aplicación desde cero. Todas estas aptitudes conseguidas se consideran muy valiosas ya que se ve el trabajo y el conocimiento necesario para abordar el desarrollo de un producto software, aunque sea a una escala muy reducida.

En la Tabla 5 se puede ver más claramente que se ha cumplido la diferenciación que se pretendía conseguir respecto a las aplicaciones competidoras, añadiendo funcionalidades que ninguna de las otras aplicaciones ofrece actualmente.

Tabla 5: Comparación competidores con Our Wildlife

Funcionalidades	Descripción	iSpot	Observation	iNaturalist	Our Wildlife
Registrar Observación	A partir de una imagen y se puede publicar de qué especie se trata.	✓	✓	✓	✓
Sugerir Especie	Permite opinar sobre qué especie es la de una observación.	✓	✗	✓	✓
Ver Observaciones	Listado de observaciones pudiéndose visualizar en un mapa.	✓	✓	✓	✓
Buscador Especies	Se puede buscar una especie por su nombre común o científico.	✓	✓	✓	✓
Algoritmo para identificar especies de foto	Bien propio o de terceros disponer de un algoritmo que sugiera las especies sobre una imagen.	✗	✓	✓	✓
Observación Privada	A partir de una imagen se puede identificar qué especie se ha observado	✗	✗	✗	✓

	quedando únicamente visible para el usuario que la ha registrado.				
Buscar Ruta	Mapa donde se puede trazar una ruta viendo qué especies se han observado cerca de ésta.	✗	✗	✗	✓

8.1 Relación del trabajo desarrollado con los estudios cursados

Muchas asignaturas han servido para tener el conocimiento base necesario para realizar este trabajo, pero en especial las que más han aportado han sido las asignaturas pertenecientes a la rama de Ingeniería del Software, concretamente:

- La asignatura *AER* (Análisis y Especificación de Requisitos) para saber qué técnicas utilizar a la hora de plasmar los requisitos del problema, visible en el apartado Análisis y Especificación de Requisitos.
- Las asignaturas *PSW* (Proceso de Software) y su continuación *PIN* (Proyecto de Ingeniería de Software) por permitir el aprendizaje de tecnologías que luego se han aplicado en este mismo trabajo, como es el caso de la tecnología escogida para desarrollar el *backend*. Además, en dichas asignaturas aprendí cómo emplear metodologías ágiles para el desarrollo del software.
- La asignatura *IEI* (Integración e Interoperabilidad) por introducirnos las APIs así como herramientas de extracción de datos como *Selenium*.
- La asignatura *MES* (Mantenimiento y Evolución de Software) por darle la importancia que se merece a la documentación y mantenibilidad del código, aplicado en el trabajo con la herramienta *Swagger*.

9 Trabajos futuros

El desarrollo de este trabajo se ha realizado muy a gusto por dos razones principales, las tecnologías utilizadas y el dominio del problema. Debido a esto se tiene pensado seguir trabajando en mejorar la aplicación por el hecho de aprender más sobre las tecnologías pudiendo llegar a ofrecer una plataforma bastante completa.

Como mejoras a poder implementar en un futuro cercano se podrían destacar:

- Implementar un apartado de estadísticas con las especies más observadas o difíciles de encontrar.
- Mejorar el apartado del usuario, ofreciéndole más datos sobre sus observaciones e implementando una funcionalidad para cambiar o recuperar la contraseña.
- Implementar un sistema de notificaciones para avisar al usuario cuando le sugieran especies a una observación publicada.
- Implementar una aplicación móvil, aprovechando la API ya creada.
- Investigar más herramientas para poder implementar también el reconocimiento de especies animales.
- Poder crear proyectos públicos de un dominio en concreto y participar en él, por ejemplo, un proyecto donde se junten varios usuarios que quieran saber todas las especies que se pueden observar en la Comunidad Valenciana. Todas las observaciones tendrán que localizarse en la zona determinada, así como las especies observadas verificar que sí pueden ser observadas en esa zona.
- Como el punto anterior impulsa la creación de vínculos entre distintos usuarios, se podría implementar un sistema de mensajería básico o derivar en otro para la facilitar la comunicación entre ellos.
- Mejorar la interfaz del usuario haciéndola más atractiva.
- Mejorar la calidad del código.

Como se puede ver el proyecto puede continuar evolucionando hacia un producto más complejo y completo.

10 Bibliografía

[1] Red Hat, 13 Junio 2022. [En línea]. Available: https://www.redhat.com/es/devops/what-is-agile-methodology. [Último acceso: 13 Junio 2022].

[2] Atlassian, 13 Junio 2022. [En línea]. Available: https://www.atlassian.com/es/agile/kanban/kanban-vs-scrum. [Último acceso: 13 Junio 2022].

[3] Trello, 13 Junio 2022. [En línea]. Available: https://trello.com/es. [Último acceso: 13 Junio 2022].

[4] iSpot, 14 Junio 2022. [En línea]. Available: https://www.ispotnature.org. [Último acceso: 14 Junio 2022].

[5] Observation, 14 Junio 2022. [En línea]. Available: https://observation.org/. [Último acceso: 14 Junio 2022].

[6] Aplicaciones móviles Observation, 14 Junio 2022. [En línea]. Available: https://observation.org/apps/. [Último acceso: 14 Junio 2022].

[7] iNaturalist, 14 Junio 2022. [En línea]. Available: https://www.inaturalist.org/home. [Último acceso: 14 Junio 2022].

[8] Desarrollo Web; Características REST, 17 Septiembre 2021. [En línea]. Available: https://desarrolloweb.com/articulos/que-es-rest-caracteristicas-sistemas.html. [Último acceso: 18 Junio 2022].

[9] Spring Boot, 22 Junio 2022. [En línea]. Available: https://spring.io/projects/spring-boot. [Último acceso: 22 Junio 2022].

[10] Baeldung, Build a REST API with Spring.

[11] MySQL, 22 Junio 2022. [En línea]. Available: https://www.mysql.com/. [Último acceso: 22 Junio 2022].

[12] Spring Data JPA, 22 Junio 2022. [En línea]. Available: https://spring.io/projects/spring-data-jpa. [Último acceso: 22 Junio 2022].

[13] Spring Boot Starter Web, 22 Junio 2022. [En línea]. Available: https://www.javatpoint.com/spring-boot-starter-web. [Último acceso: 22 Junio 2022].

[14] Spring Starter Security, 22 Junio 2022. [En línea]. Available: https://spring.io/guides/gs/securing-web/. [Último acceso: 22 Junio 2022].

[15] Project Lombok, 22 Junio 2022. [En línea]. Available: https://projectlombok.org/. [Último acceso: 22 Junio 2022].

[16] Adictos al trabajo; Gson, 17 Septiembre 2012. [En línea]. Available: https://www.adictosaltrabajo.com/2012/09/17/gson-java-json/. [Último acceso: 22 Junio 2022].

[17] Selenium, 22 Junio 2022. [En línea]. Available: https://www.selenium.dev/. [Último acceso: 22 Junio 2022].

[18] Cloudinary, 22 Junio 2022. [En línea]. Available: https://cloudinary.com/. [Último acceso: 22 Junio 2022].

[19] JSON Web Tokens, 22 Junio 2022. [En línea]. Available: https://jwt.io/. [Último acceso: 22 Junio 2022].

[20] Drewnoakes; Extractor Metadatos, 22 Junio 2022. [En línea]. Available: https://github.com/drewnoakes/metadata-extractor/. [Último acceso: 22 Junio 2022].

[21] Jet Brains; IntelliJ IDEA, 22 Junio 2022. [En línea]. Available: https://www.jetbrains.com/es-es/idea/. [Último acceso: 22 Junio 2022].

[22] XAMPP, 22 Junio 2022. [En línea]. Available: https://www.apachefriends.org/es/index.html. [Último acceso: 22 Junio 2022].

[23] Postman, 22 Junio 2022. [En línea]. Available: https://www.postman.com/. [Último acceso: 22 Junio 2022].

[24] Vue js, 23 Junio 2022. [En línea]. Available: https://vuejs.org/. [Último acceso: 23 Junio 2022].

[25] Axios, 23 Junio 2022. [En línea]. Available: https://axios-http.com/es/docs/intro. [Último acceso: 23 Junio 2022].

[26] Vuex, 23 Junio 2022. [En línea]. Available: https://vuex.vuejs.org/. [Último acceso: 23 Junio 2022].

[27] Vue Router, 23 Junio 2022. [En línea]. Available: https://router.vuejs.org/. [Último acceso: 23 Junio 2022].

[28] Introducción a Bootstrap, 23 Junio 2022. [En línea]. Available: https://getbootstrap.com/docs/5.0/getting-started/introduction/. [Último acceso: 23 Junio 2022].

[29] Mapbox, 23 Junio 2022. [En línea]. Available: https://www.mapbox.com/. [Último acceso: 23 Junio 2022].

[30] Visual Studio Code, 23 Junio 2022. [En línea]. Available: https://code.visualstudio.com/. [Último acceso: 23 Junio 2022].

[31] Adode XD, 23 Junio 2022. [En línea]. Available: https://www.adobe.com/es/products/xd.html. [Último acceso: 23 Junio 2022].

[32] Github, 23 Junio 2022. [En línea]. Available: https://github.com/. [Último acceso: 23 Junio 2022].

[33] Danielme; Herencia en JPA, 22 Octubre 2021. [En línea]. Available: https://danielme.com/2021/10/22/curso-jakarta-ee-jpa-con-hibernate-relaciones-de-herencia/. [Último acceso: 25 Junio 2022].

[34] Swagger, 5 Julio 2022. [En línea]. Available: https://swagger.io/. [Último acceso: 5 Julio 2022].

[35] Oscar Blancarte Blog; API REST Spring Boot y Swagger, 28 Agosto 2020. [En línea]. Available: https://www.oscarblancarteblog.com/2020/08/28/documentar-un-api-rest-con-swagger-y-spring-boot/. [Último acceso: 5 Julio 2022].

[36] Baeldung Swagger, 11 Mayo 2022. [En línea]. Available: https://www.baeldung.com/spring-boot-swagger-jwt. [Último acceso: 5 Julio 2022].

[37] Curso Vue desde Cero, 31 Mayo 2021. [En línea]. Available: https://www.youtube.com/watch?v=Wd9dOIlTWCc&t=5807s. [Último acceso: 30 Junio 2022].

[38] Mapbox Tutorial, 30 Junio 2022. [En línea]. Available:

https://docs.mapbox.com/help/tutorials/building-a-store-locator/. [Último acceso: 30 Junio 2022].

[39] Documentación Axios, 30 Junio 2022. [En línea]. Available: https://axios-http.com/docs/api_intro. [Último acceso: 30 Junio 2022].

[40] JUnit, 2 Julio 2022. [En línea]. Available: https://junit.org/junit5/. [Último acceso: 2 Julio 2022].

[41] Mockito, 2 Julio 2022. [En línea]. Available: https://site.mockito.org/. [Último acceso: 2 Julio 2022].

[42] Naciones Unidas, 9 Julio 2022. [En línea]. Available: https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/. [Último acceso: 9 Julio 2022].

[43] Mayo Clinic, 27 Febrero 2019. [En línea]. Available: https://www.mayoclinic.org/es-es/healthy-lifestyle/fitness/in-depth/walking/art-20046261. [Último acceso: 9 Julio 2022].

[44] Clínica Universidad de Navarra, 9 Julio 2022. [En línea]. Available: https://www.cun.es/chequeos-salud/vida-sana/deporte/caminar-salud#:~:text=Caminar%20previene%20el%20inicio%20de,las%20fracturas%20en%20las%20ca%C3%ADdas.. [Último acceso: 9 Julio 2022].

[45] D. W. B. y. V. Reynolds, Caminar para la salud, INDE Publicaciones, 2006.

Glosario

API (Application Programming Interface): es un conjunto protocolos y definiciones que permiten el consumo de las funcionalidades implementadas.

Backend: es la capa de acceso a datos donde también se implementa la lógica de una aplicación, todo esto realizado de manera que no es visible para el usuario.

Clasificación taxonómica: permite clasificar de forma ordenada y siguiendo una jerarquía todas las especies existentes. Para ello se basa en diferentes categorías anidadas que se van refinando hasta llegar al nombre de la especie en concreto.

CRUD (Create Read Update Delete): conjunto de operaciones básicas, concretamente, creación, lectura, actualización y eliminación de un objeto o entidad.

Depuración: acción que permite interrumpir la ejecución de una funcionalidad para ver el estado de las variables, permitiendo un análisis minucioso de la cada instrucción ejecutada. Favorece la búsqueda y solución de errores existentes en el código.

Endpoints: son las *urls* que exponen las funcionalidades implementadas para la API.

Framework: es un marco de trabajo que ofrece unos estándares con el objetivo de hacer un desarrollo software de forma más rápida y sencilla.

Frontend: es la parte visible de una aplicación que permite al usuario interactuar con esta.

JSON (Javascript Object Notation): formato que sirve para estructurar los objetos y ser más manejables durante el intercambio de datos.

Metadatos: son datos sobre otros datos, como puede ser un archivo o una imagen. Pretende mantener una información estructurada sobre los documentos a los que pertenece. Un ejemplo de metadatos son el dispositivo, fecha y coordenadas de una imagen.

Metodología ágil: procedimiento de desarrollo software iterativo e incremental que permite la evolución de los requisitos y soluciones a lo largo del tiempo.

REST (Representational State Transfer): es una interfaz que permite conectar sistemas basados en el protocolo HTTP.

Scrapper: programa o funcionalidad mediante la que se consiguen obtener datos de sitios web.

Screen scraping: técnica de programación que permite obtener datos extrayéndolos de la interfaz o vista en los que están contenidos.

Servidor embebido: permite desplegar la aplicación en el entorno local.

Token: cadena de texto formada por tres partes: cabecera que indica el algoritmo y tipo de token, carga útil donde aparecen los datos del usuario y la firma que permite validar este *token*. Todas ellas codificadas.

UML (Unified Modeling Language): es un lenguaje de modelado de datos que permite representar a través de diagramas la documentación necesaria para un desarrollo software.

ANEXO A: Objetivos de desarrollo sostenible

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No Procede
ODS 1. **Fin de la pobreza.**				**X**
ODS 2. **Hambre cero.**				**X**
ODS 3. **Salud y bienestar.**		**X**		
ODS 4. **Educación de calidad.**		**X**		
ODS 5. **Igualdad de género.**				**X**
ODS 6. **Agua limpia y saneamiento.**				**X**
ODS 7. **Energía asequible y no contaminante.**				**X**
ODS 8. **Trabajo decente y crecimiento económico.**				**X**
ODS 9. **Industria, innovación e infraestructuras.**				**X**
ODS 10. **Reducción de las desigualdades**.				**X**
ODS 11. **Ciudades y comunidades sostenibles**.				**X**
ODS 12. **Producción y consumo responsables.**				**X**
ODS 13. **Acción por el clima.**				**X**
ODS 14. **Vida submarina.**				**X**
ODS 15. **Vida de ecosistemas terrestres.**			**X**	
ODS 16. **Paz, justicia e instituciones sólidas.**				**X**
ODS 17. **Alianzas para lograr objetivos.**				**X**

Los Objetivos de Desarrollo Sostenible (ODS) [42] son unas metas que se definieron en el año 2015 y pretenden construir un mundo más sostenible para el futuro. Todas estas metas tienen previstas ser revisadas en la Agenda 2030, reunión que acogerá a todos los líderes mundiales para ver la evolución que se ha logrado y revisar si los objetivos definidos se han cumplido y en qué medida.

El trabajo expuesto a lo largo de esta memoria está relacionado con tres de los diecisiete objetivos que se definieron en aquella reunión ya comentada, concretamente con el ODS 3 (Salud y Bienestar), ODS 4 (Educación de calidad) y ODS 15 (Vida de ecosistemas terrestres).

En primer lugar, la relación con el ODS 3, se explica desde dos partes muy importantes de la aplicación. La primera de ellas es propia del flujo que sigue la aplicación, es decir, esta se basa en la subida de imágenes para el reconocimiento de especies y ofrecer con ellas distintas funcionalidades. Bien, estas imágenes hay que conseguirlas de alguna forma, y no es otra que, andando por el campo, o bien realizando rutas si se quieren recoger más imágenes y de especies menos comunes.

La segunda está relacionada con la funcionalidad que provee el CU-10. Buscar Ruta pues este caso de uso está pensado para aquellas personas que planean sus rutas de senderismo, y que de esta forma pueden ver las observaciones de qué especies se han realizado cerca de por donde ellos van a pasar andando.

Gracias a estos dos motivos podemos decir que la aplicación promueve directa e indirectamente un ejercicio tan básico como importante, caminar. Pues son muchos los estudios que nombran los beneficios de realizar este ejercicio tan sencillo [43] [44] [45], entre ellos:

- Mejora el flujo sanguíneo en las piernas, así como el estado cardiovascular.
- Ayuda a reducir el nivel de estrés.
- Ayuda a fortalecer los huesos.

En segundo lugar, se ha relacionado el trabajo con el ODS 4 debido a que, durante el desarrollo del mismo, se han incrementado los conocimientos sobre especies vegetales presentes en nuestra naturaleza. Este tipo de conocimiento no está muy presente en las asignaturas de enseñanza obligatoria y se está perdiendo, pues mucha gente no sabe identificar la mayoría de las especies que se pueden observar en el campo, siendo muy importante conocer la naturaleza que nos rodea. Además, para profundizar en ese aprendizaje, se pueden consultar los datos de las especies que hay registradas y acceder a enlaces de interés como puede ser su página de la *Wikipedia*, donde informarse más al detalle sobre sus características.

Por último, también se ha relacionado con el ODS 15 en un nivel bajo, ya que en todas las especies que se muestran en la aplicación se puede observar su estado de conservación, es decir, en qué situación de vulnerabilidad se encuentra la especie, informando así al usuario de qué especies hay respetar más debido a su estado en el ecosistema.

ANEXO B: Especificación Casos de uso

Tabla 6: CU-1. Registrarse

Nombre:	CU-1. Registrarse
Actores:	Usuario Anónimo
Descripción:	El usuario podrá registrarse para obtener una cuenta en la aplicación web.
Precondición:	
Flujo:	1. El usuario rellena los datos obligatorios y pulsará para crear la cuenta. 2. El sistema valida los datos, se guarda la cuenta y muestra la pantalla de inicio de sesión.
Postcondición:	Se crea una cuenta con todos los datos rellenados en el formulario correspondiente.
Alternativas/Excepciones:	2.1 Se deberán rellenar todos los campos obligatorios para poder darse de alta.

Tabla 7: CU-2. Iniciar Sesión

Nombre:	CU-2. Iniciar Sesión
Actores:	Usuario Anónimo
Descripción:	El usuario podrá acceder a la aplicación a través de una cuenta.
Precondición:	Tener creada una cuenta con la que poder acceder.
Flujo:	1. El usuario rellena los campos de correo y contraseña y pulsar para acceder. 2. El sistema valida los datos y si son correctos accede a la web con la cuenta correspondiente.
Postcondición:	Acceder a la página web con la cuenta correspondiente.
Alternativas/Excepciones:	2.1 Si se introduce un correo o contraseña incorrectos, la aplicación avisa al usuario.

Tabla 8: CU-3. Cerrar Sesión

Nombre:	CU-3. Cerrar Sesión
Actores:	Usuario Registrado
Descripción:	El usuario podrá cerrar la sesión de su cuenta.
Precondición:	Haber iniciado sesión previamente.
Flujo:	1. El usuario en la barra de navegación pulsará para cerrar sesión. 2. El sistema cierra sesión y muestra el inicio de sesión.
Postcondición:	La sesión que estaba iniciada se cerrará.
Alternativas/Excepciones:	

Tabla 9: CU-4. Identificar Ser Vivo

Nombre:	CU-4. Identificar Ser Vivo
Actores:	Usuario Registrado, WhatistheAnimal/Whatistheplant y GBIF
Descripción:	Se podrán subir imágenes para identificar a qué especie pertenece sin compartirlas con el resto de los usuarios de la aplicación.

Precondición:	Tener una sesión iniciada.
Flujo:	1. Acceder al apartado del menú que corresponda. 2. Rellenar el formulario para enviar los datos. 3. El sistema por IA identifica cuál es la especie más probable. 4. El sistema guarda la identificación.
Postcondición:	Se guarda la identificación.
Alternativas/Excepciones:	

Tabla 10: CU-5. Biblioteca Identificaciones

Nombre:	**CU-5. Biblioteca Identificaciones**
Actores:	Usuario Registrado y Mapbox
Descripción:	Los usuarios registrados tendrán disponibles todas las identificaciones realizadas, con los datos que éstas tienen.
Precondición:	Haber iniciado sesión y realizado alguna identificación.
Flujo:	1. En la barra de navegación se accederá al apartado correspondiente. 2. El sistema mostrará una vista con todas las identificaciones que correspondan y pulsando sobre ellas se podrá navegar a su ficha.
Postcondición:	Se muestran todas identificaciones realizadas en su cuenta.
Alternativas/Excepciones:	

Tabla 11: CU-6. Filtros Listado Identificaciones

Nombre:	**CU-6. Filtros Listado Identificaciones (EXTENDS CU-5)**
Actores:	Usuario Registrado y Mapbox
Descripción:	Para que sea más fácil la búsqueda de ciertas identificaciones, se dispondrá de una opción para filtrar según diversos criterios.
Precondición:	Tener identificaciones en la biblioteca de identificaciones.
Flujo:	1. Añadir algún filtro por el que buscar. 2. El sistema filtrará las identificaciones por el filtro añadido.
Postcondición:	Se muestran solo las identificaciones que coinciden con el criterio a filtrar.
Alternativas/Excepciones:	2.1 Si no hay ninguna identificación que coincida con el filtro aplicado, el sistema lo notificará, además de no mostrar ninguna identificación.

Tabla 12: CU-7. Ficha Especie

Nombre:	**CU-7. Ficha especie**
Actores:	Usuario y Mapbox
Descripción:	El usuario podrá ver todos los datos de una especie en su correspondiente ficha, así como las observaciones que se han realizado de ella.
Precondición:	
Flujo:	1. Buscar en el listado de especies y pulsar sobre la que más cuadra al usuario. 2. El sistema mostrará todos los datos relacionados con la especie seleccionada.
Postcondición:	Se muestran los datos de la especie.

Alternativas/Excepciones:	

Tabla 13: CU-8. Listado Especie

Nombre:	**CU-8. Listado especies**
Actores:	Usuario
Descripción:	Listado que mostrará todas las especies almacenadas en la aplicación, además de dar la posibilidad de acceder a sus datos.
Precondición:	
Flujo:	1. Acceder al correspondiente apartado en la barra de navegación. 2. El sistema mostrará todas las especies de la aplicación.
Postcondición:	Se muestran todas las especies almacenadas en la base de datos.
Alternativas/Excepciones:	2.1 Si no hay especies, el sistema mostrará un mensaje para que el usuario lo sepa.

Tabla 14: CU-9. Filtros Búsqueda

Nombre:	**CU-9. Filtros búsqueda (EXTENDS CU-8)**
Actores:	Usuario
Descripción:	Para poder obtener un listado de especies más concreto se dispondrá de un filtro para refinar la búsqueda.
Precondición:	Que haya varias especies registradas de características distintas.
Flujo:	1. Acceder al listado de especies, **CU-8. Listado especies**. 2. Pulsar sobre el botón 'Filtros' y añadir algún filtrado por el que buscar, también se puede buscar por el nombre de la especie. 3. El sistema mostrará las especies que se adecuen a los filtros aplicados.
Postcondición:	Se muestran solo las especies que coinciden con el criterio a filtrar.
Alternativas/Excepciones:	3.1 Si no hay ninguna especie que coincida con el filtro aplicado, el sistema lo notificará, además de no mostrar ninguna identificación

Tabla 15: CU-10. Buscar Ruta

Nombre:	**CU-10. Buscar ruta**
Actores:	Usuario y Mapbox
Descripción:	El usuario podrá buscar en un mapa para hacer una ruta entre dos sitios. Al mostrar en el mapa la ruta, verá las observaciones realizadas más cercanas y, por lo tanto, ver qué especies puede estar pendiente de observar en su recorrido.
Precondición:	
Flujo:	1. Elegirá, o través del propio mapa o a través de un buscador, los dos puntos entre los que va a realizar la ruta (origen y destino). 2. En el mapa aparecerán como popups las observaciones de la zona.
Postcondición:	Se mostrarán las observaciones junto la ruta marcada.
Alternativas/Excepciones:	

Tabla 16: CU-11. Cargar Observación

Nombre:	**CU-11. Cargar Observación**
Actores:	Usuario Registrado, WhatistheAnimal/Whatistheplant, GBIF y Mapbox
Descripción:	El usuario podrá registrar una observación para compartirla con los usuarios de la aplicación.
Precondición:	Haber iniciado sesión
Flujo:	1. El usuario accede a la pantalla de la carga, a través de un botón en la barra de navegación. 2. Se rellenan todos los datos obligatorios para poder registrar la observación, habrá algunos como fecha y coordenadas que se sacarán por metadatos. 3. El sistema da de alta la observación con todos los datos de ésta.
Postcondición:	Observación registrada.
Alternativas/Excepciones:	2.1 Campos obligatorios no rellenados.

Tabla 17: CU-12. Edición Observación

Nombre:	**CU-12. Edición Observación**
Actores:	Usuario Registrado y Mapbox
Descripción:	El usuario podrá modificar las observaciones que haya realizado.
Precondición:	Haber registrado alguna observación.
Flujo:	1. Bien acceder a la ficha de la observación a modificar, o, desde tus observaciones pulsar el botón correspondiente asociado a la observación que se quiera. 2. Modificar los datos que se deseen de la observación. 3. Guardar cambios.
Postcondición:	Observación actualizada.
Alternativas/Excepciones:	3.1 El sistema valida que ningún campo obligatorio se quede vacío.

Tabla 18: CU-13. Eliminación Observación

Nombre:	**CU-13. Eliminación Observación**
Actores:	Usuario Registrado
Descripción:	El usuario podrá eliminar una observación en cualquier momento.
Precondición:	Haber registrado alguna observación.
Flujo:	1. Acceder a 'Tus Observaciones' (**CU-19. Tus Observaciones**) para ver todas tus observaciones. 2. Pulsar el botón eliminar sobre la observación que se quiera borrar. 3. Aceptar el modal de confirmación. 4. El sistema eliminará la observación de la base de datos.
Postcondición:	Observación eliminada
Alternativas/Excepciones:	

Tabla 19: CU-14. Listado Observaciones

Nombre:	**CU-14. Listado Observaciones**
Actores:	Usuario y Mapbox
Descripción:	En la ventana principal de la aplicación, el usuario podrá ver en

	forma de lista todas las observaciones registradas en la aplicación, así como su ubicación en un mapa.
Precondición:	
Flujo:	1. Acceder a la ventana principal (Explora) de la aplicación. 2. El sistema mostrará todas las observaciones registradas.
Postcondición:	El sistema mostrará todas las observaciones registradas.
Alternativas/Excepciones:	2.1 Si no hay ninguna observación registrada, el sistema mostrará un mensaje.

Tabla 20: CU-15. Ficha Observación

Nombre:	**CU-15. Ficha Observación**
Actores:	Usuario y Mapbox
Descripción:	El usuario podrá consultar todos los datos asociados a una observación.
Precondición:	Que haya observaciones registradas.
Flujo:	1. Acceder a la vista principal o a 'Tus Observaciones', si ha iniciado sesión, para ver todas las observaciones. 2. Pulsar sobre la que se quiera ver más datos. 3. El sistema mostrará los datos de la observación seleccionada.
Postcondición:	Se muestran los datos de la observación.
Alternativas/Excepciones:	

Tabla 21: CU-16. Búsqueda Observaciones y Filtros

Nombre:	**CU-16. Búsqueda Observaciones y Filtros (EXTENDS CU-14 y CU-17)**
Actores:	Usuario y Mapbox
Descripción:	El usuario podrá buscar observaciones utilizando filtros o bien, el buscador para encontrar los de una especie en concreto.
Precondición:	Que haya observaciones para buscar.
Flujo:	1. Acceder a la página principal o a 'Tus Observaciones', si ha iniciado sesión. 2. Introducir la especie a buscar o pulsar el botón 'Filtros' y establecer unos filtros por los que buscar. 3. El sistema actualizará el mapa y listado para mostrar las observaciones que coinciden con el criterio aplicado.
Postcondición:	Se muestran las observaciones que coincidan con el criterio de búsqueda.
Alternativas/Excepciones:	3.1 Si no hay ninguna coincidencia, se mostrará un mensaje para informar al usuario.

Tabla 22: CU-17. Tus Observaciones

Nombre:	**CU-17. Tus Observaciones**
Actores:	Usuario Registrado y Mapbox
Descripción:	En el apartado de tus observaciones, el usuario podrá ver en forma de lista todas las observaciones registradas en su cuenta en la aplicación, así como su ubicación en un mapa.
Precondición:	Haber registrado alguna observación.
Flujo:	1. Acceder desde la barra de navegación a 'Tus

	Observaciones'. 2. El sistema mostrará todas las observaciones registradas por este usuario.
Postcondición:	El sistema mostrará todas las observaciones registradas por el usuario en cuestión.
Alternativas/Excepciones:	2.1 Si no hay ninguna observación registrada por el usuario, el sistema mostrará un mensaje.

Tabla 23: CU-18. Sugerir Especie

Nombre:	**CU-18. Sugerir Especie**
Actores:	Usuario Registrado
Descripción:	El usuario que registra una observación decide en base a los resultados que da la IA de qué especie se trata, en cambio, puede haber distintas opiniones, por tanto, los usuarios de la aplicación pueden dar la suya al respecto.
Precondición:	Haber iniciado sesión.
Flujo:	1. Acceder al listado de observaciones (**CU-14. Listado Observaciones**) y pulsar sobre una observación. 2. Buscar la especie que se quiere sugerir y si se desea, poner un comentario. 3. El sistema guardará la sugerencia y la asociará tanto al usuario que la pone como a la observación en la que se ha hecho.
Postcondición:	Se guarda la sugerencia asociada a la observación correspondiente.
Alternativas/Excepciones:	2.1 Se debe seleccionar una especie siempre, sino no deja enviar la sugerencia.

Tabla 24: CU-19. Listado Usuarios

Nombre:	**CU-19. Listado Usuarios**
Actores:	Usuario
Descripción:	El usuario podrá ver todos los usuarios de la aplicación.
Precondición:	
Flujo:	1. Acceder al apartado correspondiente de la barra de navegación (Usuarios). 2. El sistema mostrará todos los usuarios de la web.
Postcondición:	Se muestran todos los usuarios de la web.
Alternativas/Excepciones:	2.1 Si no hubiera ninguno, el sistema lo mostraría en un mensaje.

Tabla 25: CU-20. Ficha Usuario

Nombre:	**CU-20. Ficha Usuario**
Actores:	Usuario y Mapbox
Descripción:	El usuario podrá ver información de los usuarios de la aplicación.
Precondición:	
Flujo:	1. Acceder al listado de usuarios (**CU-19. Listado Usuarios**). 2. Pulsar sobre un usuario. 3. El sistema mostrará la información asociada al usuario pulsado.

Postcondición:	Se muestra la información de dicho usuario.
Alternativas/Excepciones:	

Tabla 26: CU-21. Búsqueda Usuarios

Nombre:	**CU-21. Búsqueda Usuarios**
Actores:	Usuario
Descripción:	El usuario podrá buscar por nombre de usuario o correo un usuario en específico.
Precondición:	
Flujo:	1. Acceder al listado de usuarios (**CU-19. Listado Usuarios**). 2. Introducir el nombre del usuario a buscar. 3. El sistema devuelve los usuarios que coincidan con la búsqueda.
Postcondición:	Se muestran los usuarios que coincidan con la búsqueda.
Alternativas/Excepciones:	3.1 Si no hay ninguna coincidencia, el sistema lo mostrará en un mensaje.

Tabla 27: CU-22. Edición Perfil

Nombre:	**CU-22. Edición Perfil**
Actores:	Usuario Registrado
Descripción:	El usuario podrá cambiar los datos de su perfil.
Precondición:	Tener una cuenta en la aplicación.
Flujo:	1. Acceder al perfil propio desde la barra de navegación. 2. Pulsar en editar perfil. 3. Cambiar los datos que se deseen. 4. El sistema guardará los cambios asociados a la cuenta.
Postcondición:	Los cambios se guardan en la base de datos asociados a la cuenta en la que se han realizado.
Alternativas/Excepciones:	4.1 El sistema valida que se hayan rellenado todos los campos obligatorios.

ANEXO C: Diseños de las interfaces

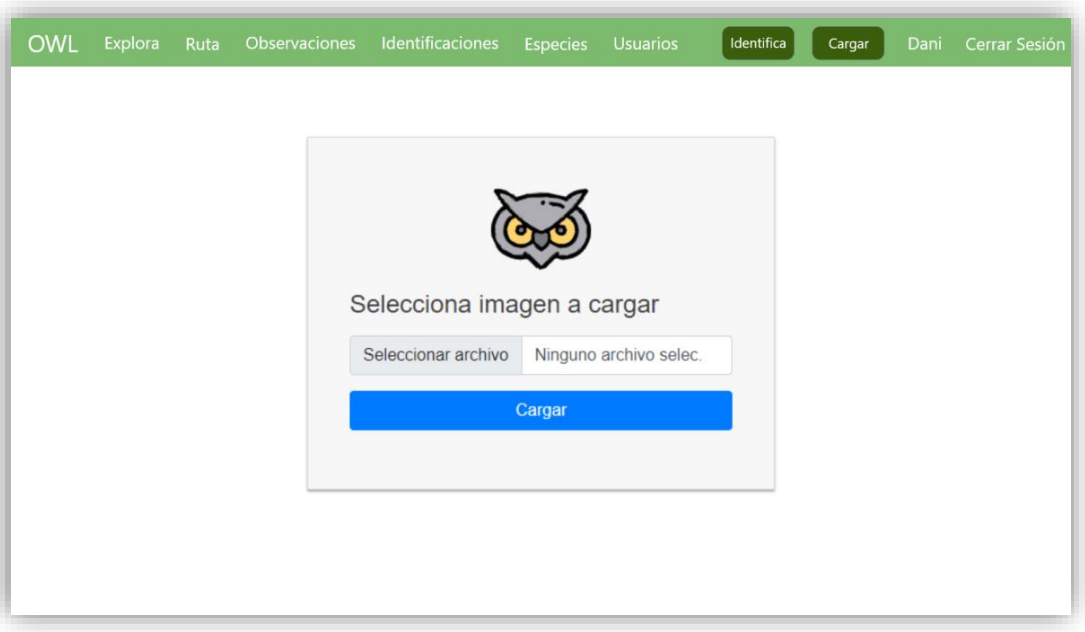

Figura 20: Diseño Pantalla de carga

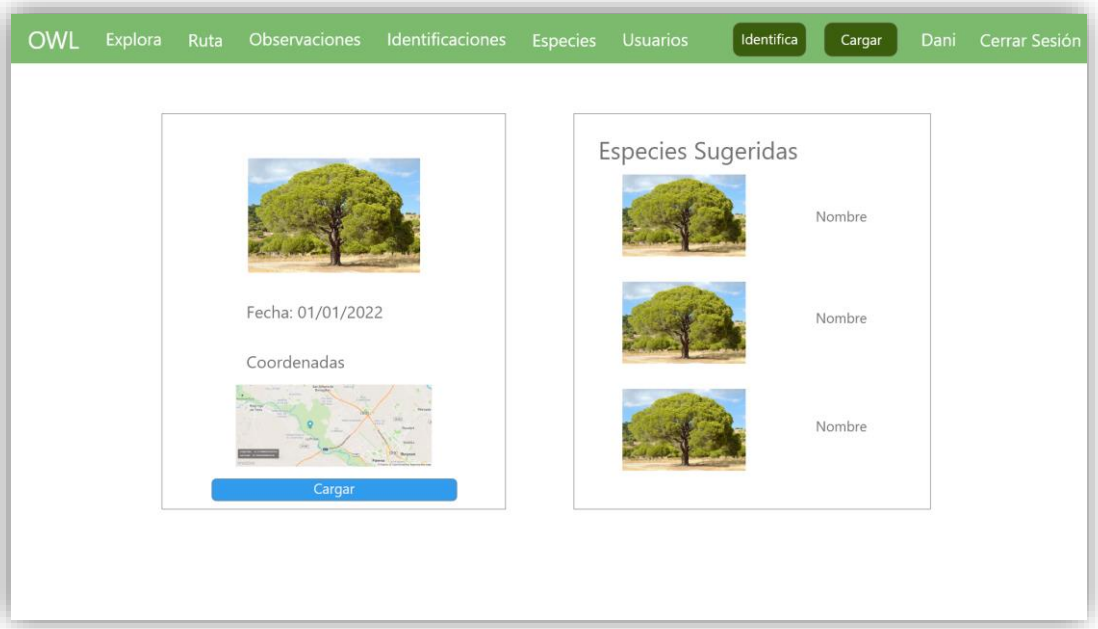

Figura 21: Diseño Pantalla Carga/Edición Observación

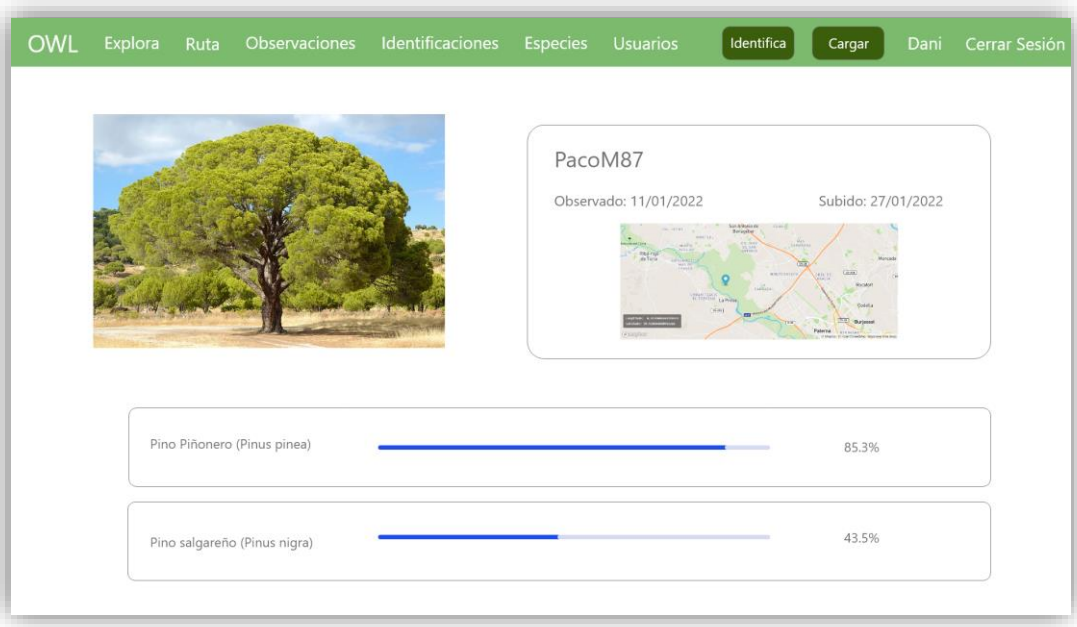

Figura 22: Diseño Ficha Identificación

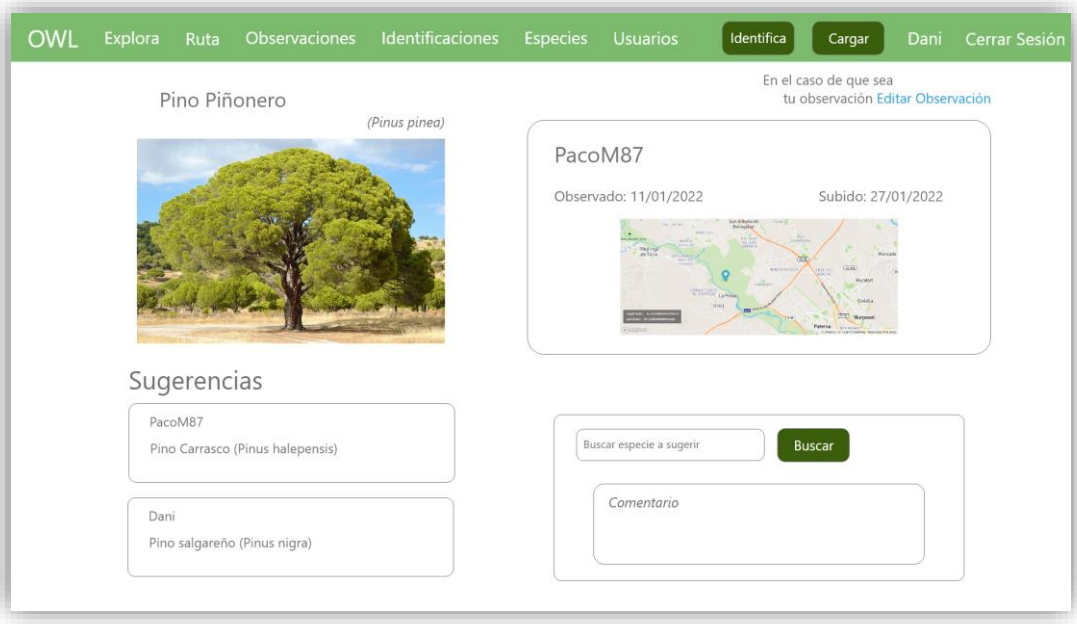

Figura 23: Diseño Ficha Observación

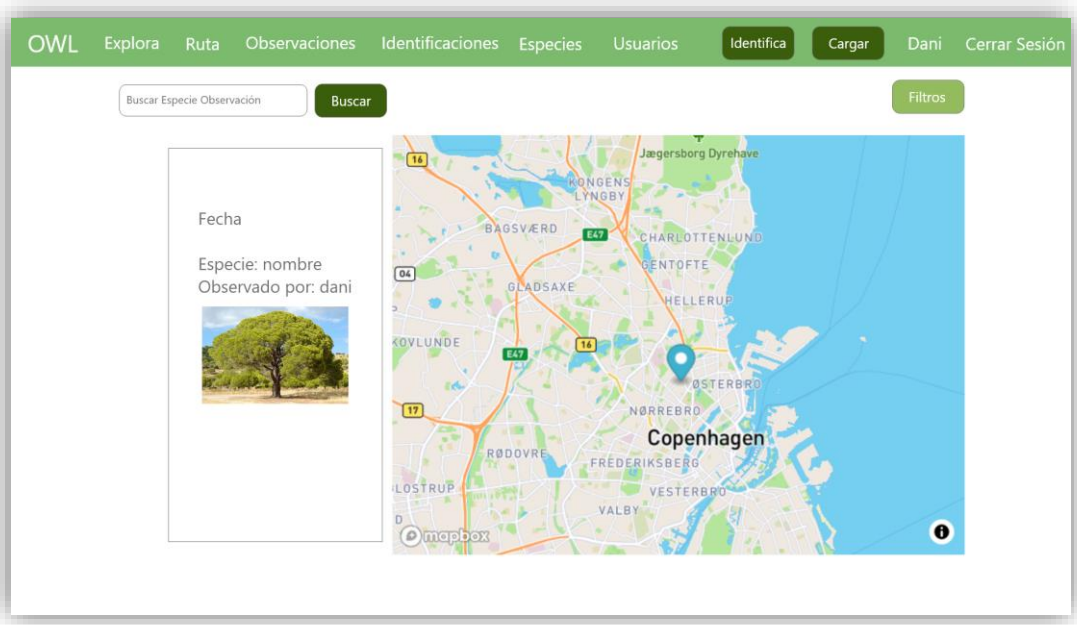

Figura 24: Diseño Explora Todas las Observaciones y Tus Observaciones

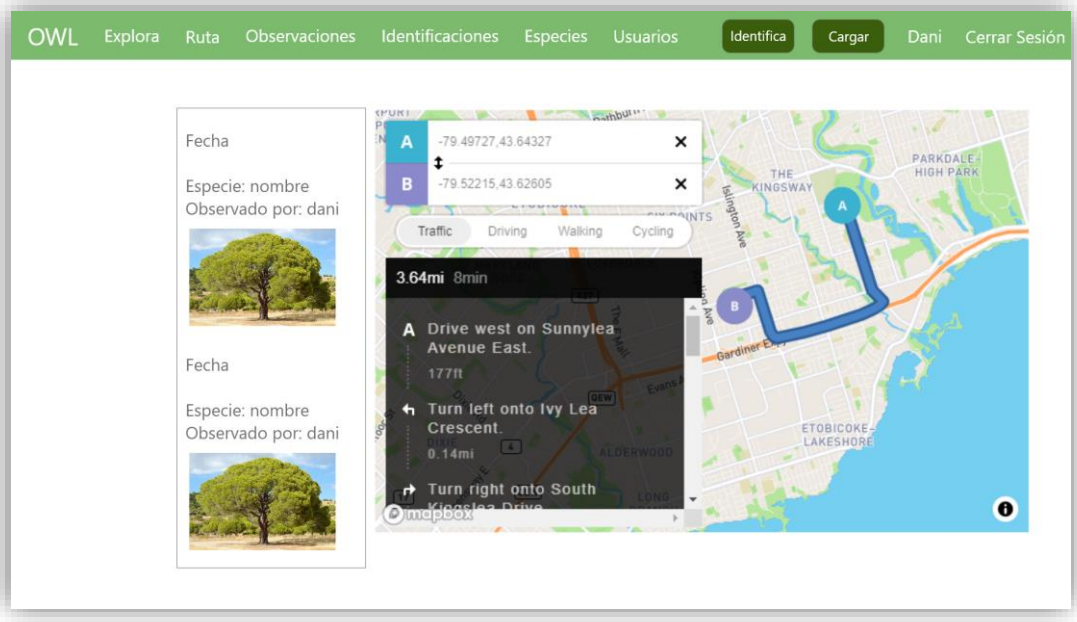

Figura 25: Diseño Buscar Ruta

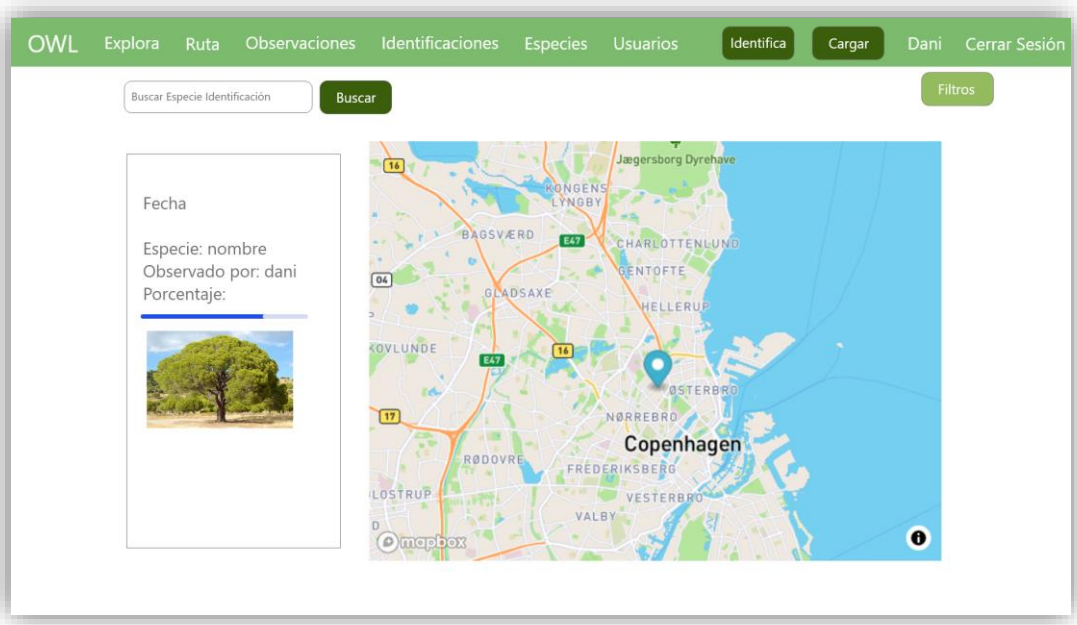

Figura 26: Diseño Tus Identificaciones

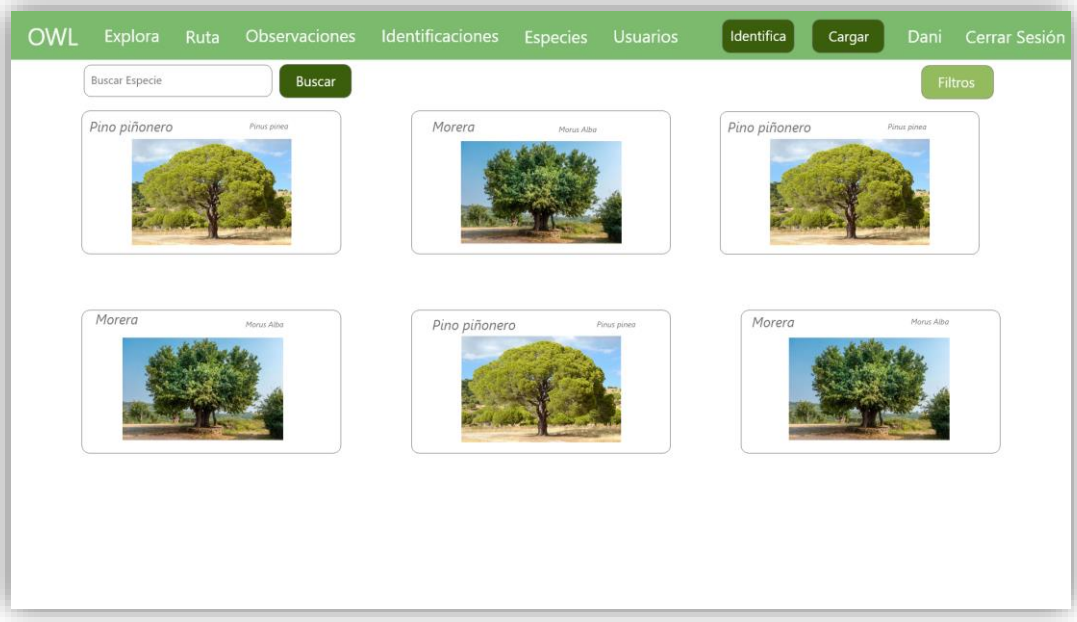

Figura 27: Diseño Listado Especies

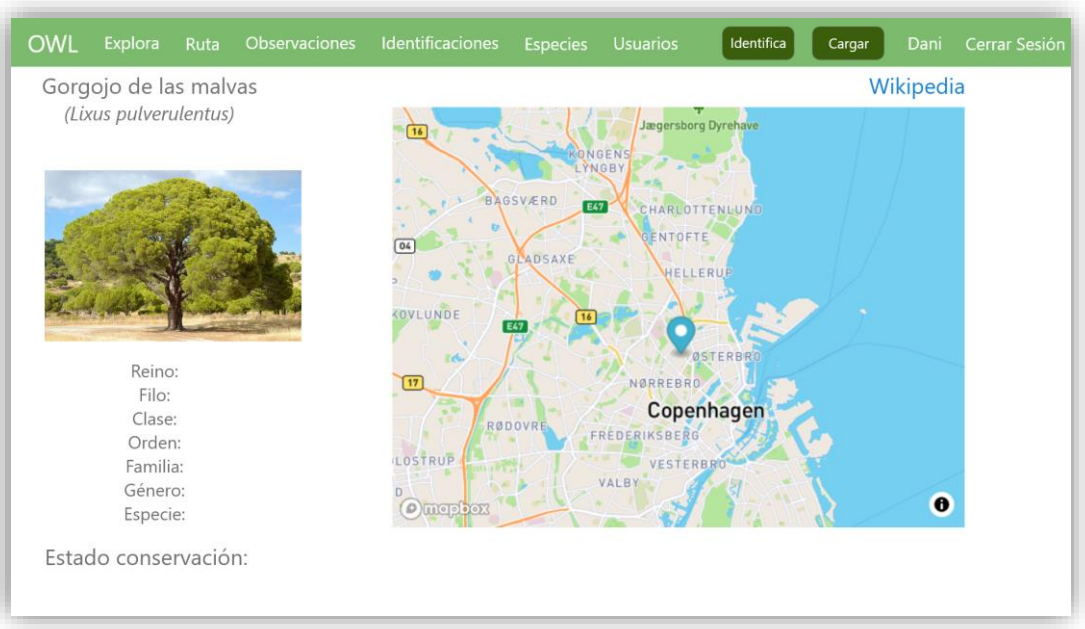

Figura 28: Diseño Ficha Especie

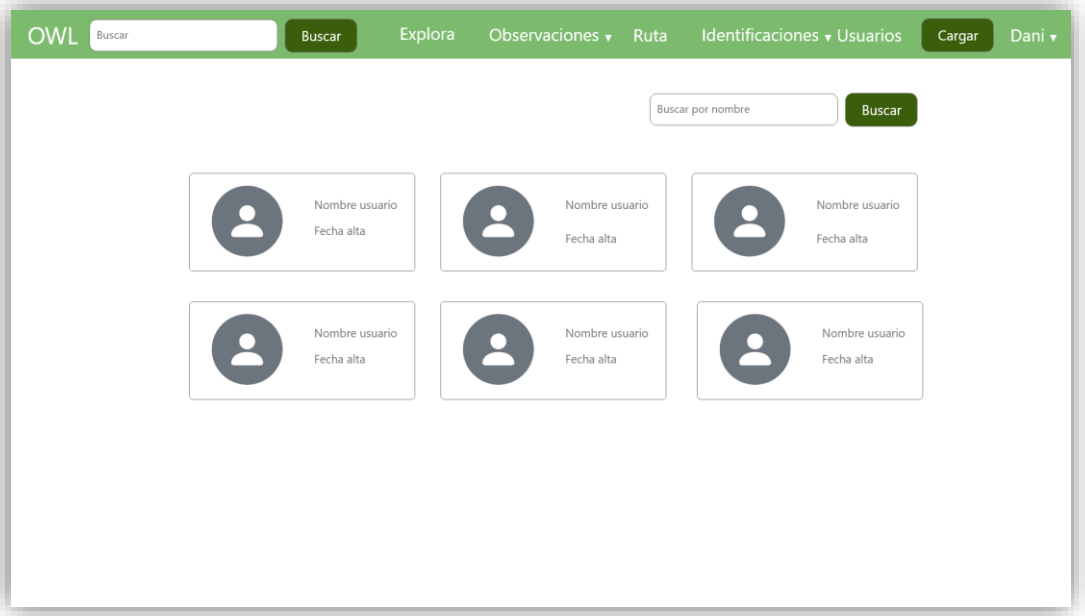

Figura 29: Diseño Listado Usuarios

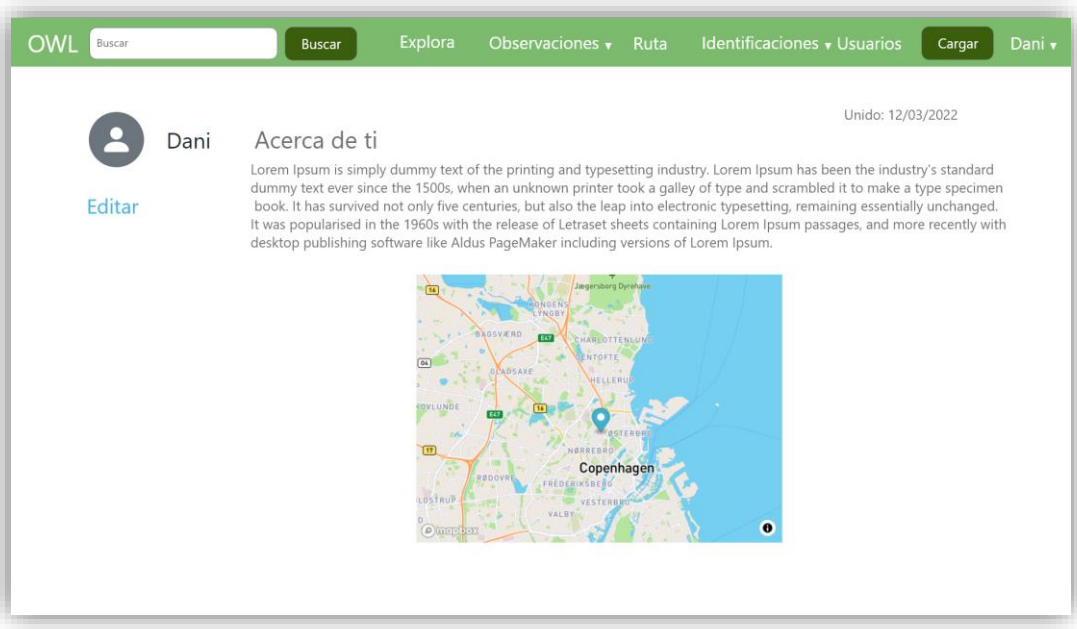

Figura 30: Diseño Perfil Usuario

ANEXO D: Resultado obtenido

Figura 31: Ventana Principal

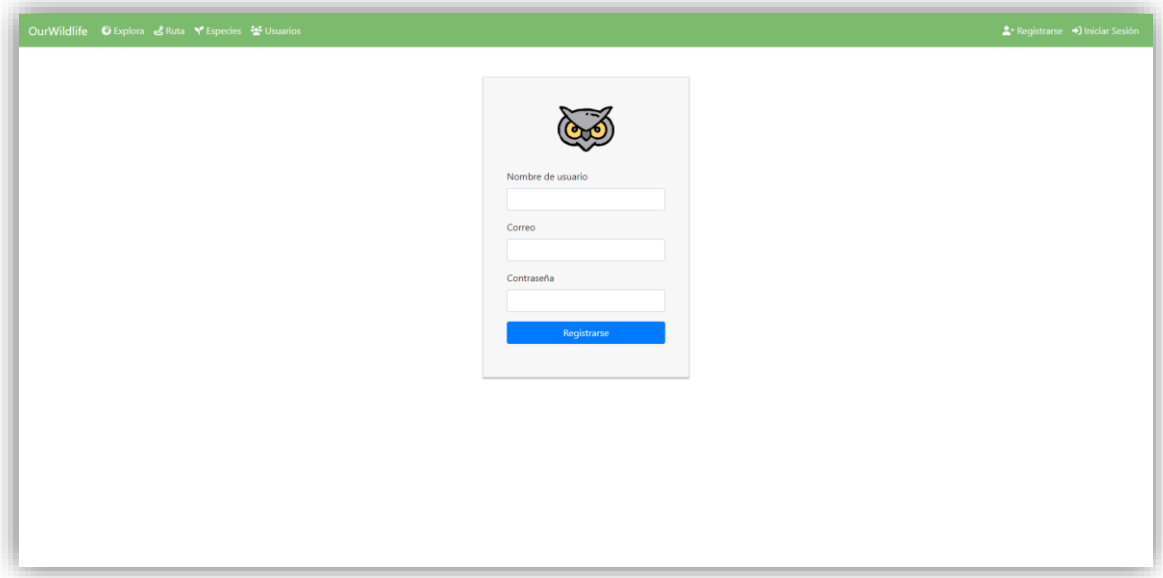

Figura 32: Registro

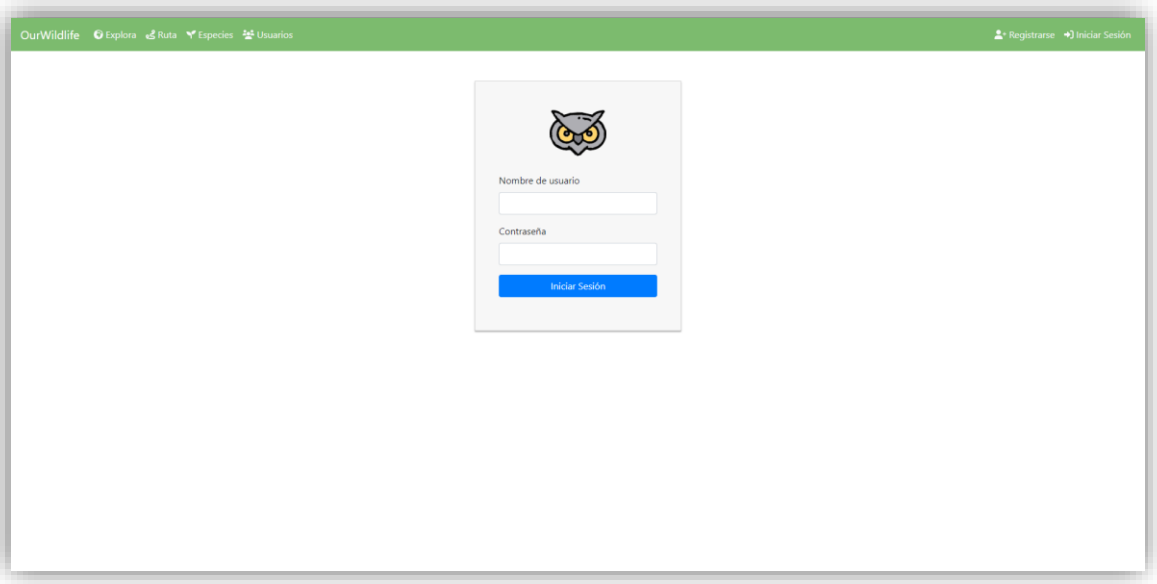

Figura 33: Inicio Sesión

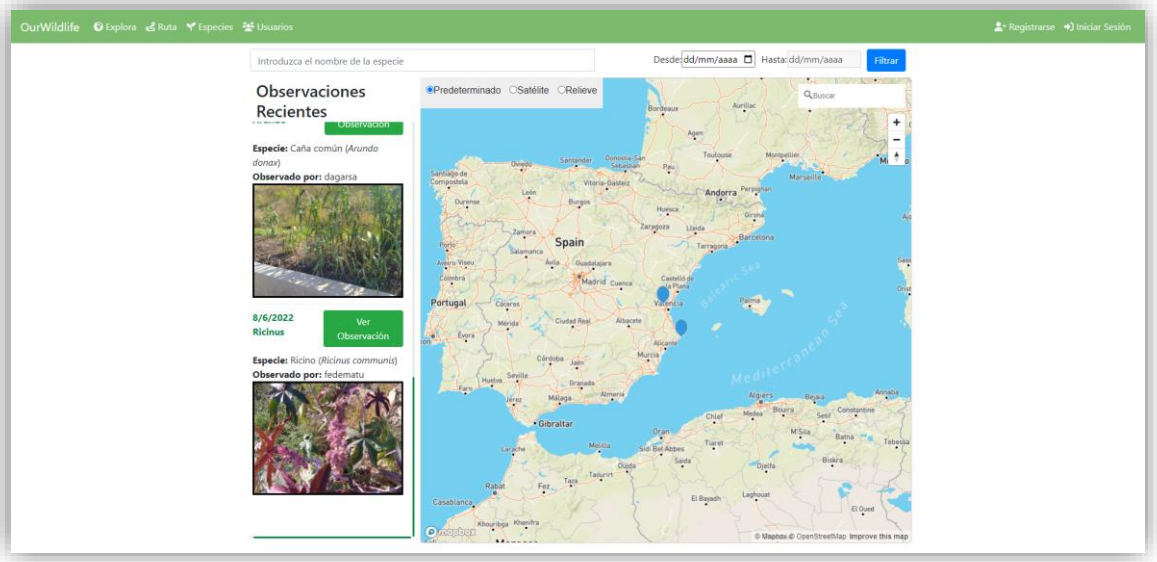

Figura 34: Explora con todas las Observaciones

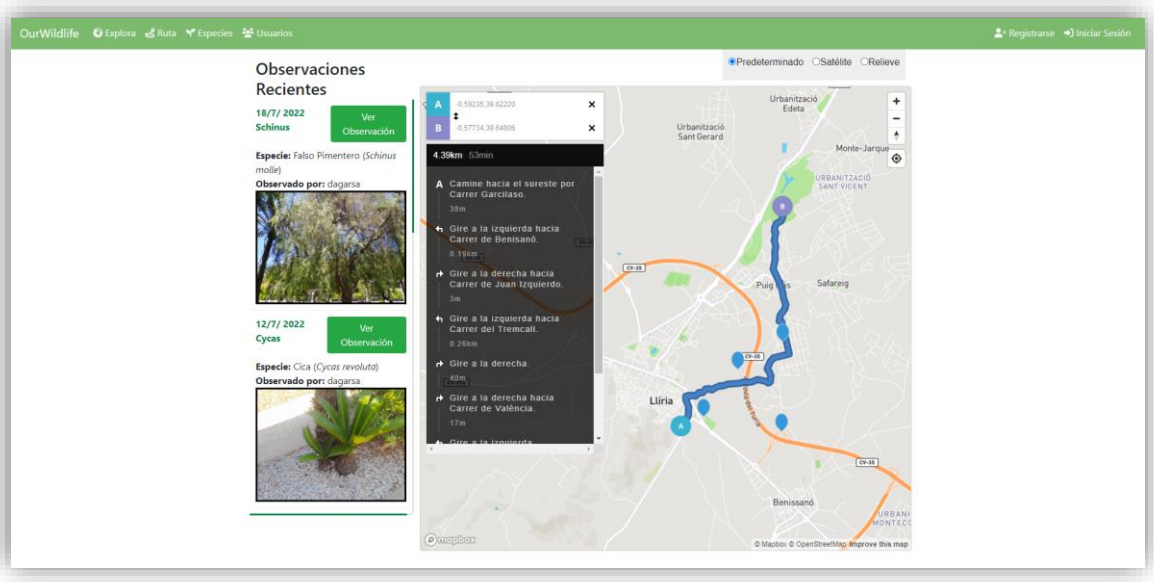

Figura 35: Buscador Rutas

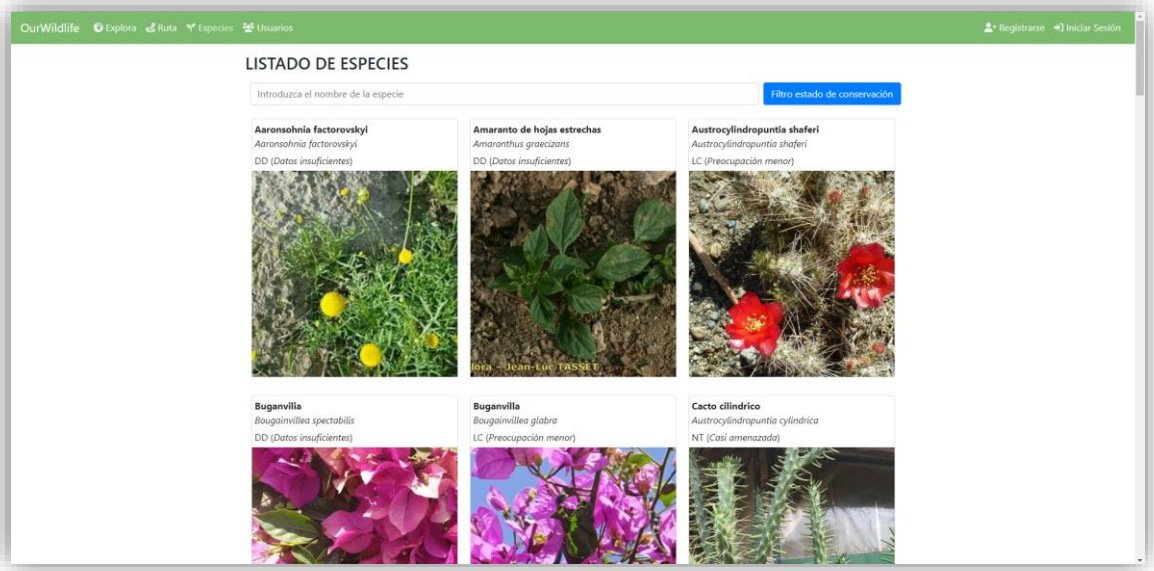

Figura 36: Listado Especies

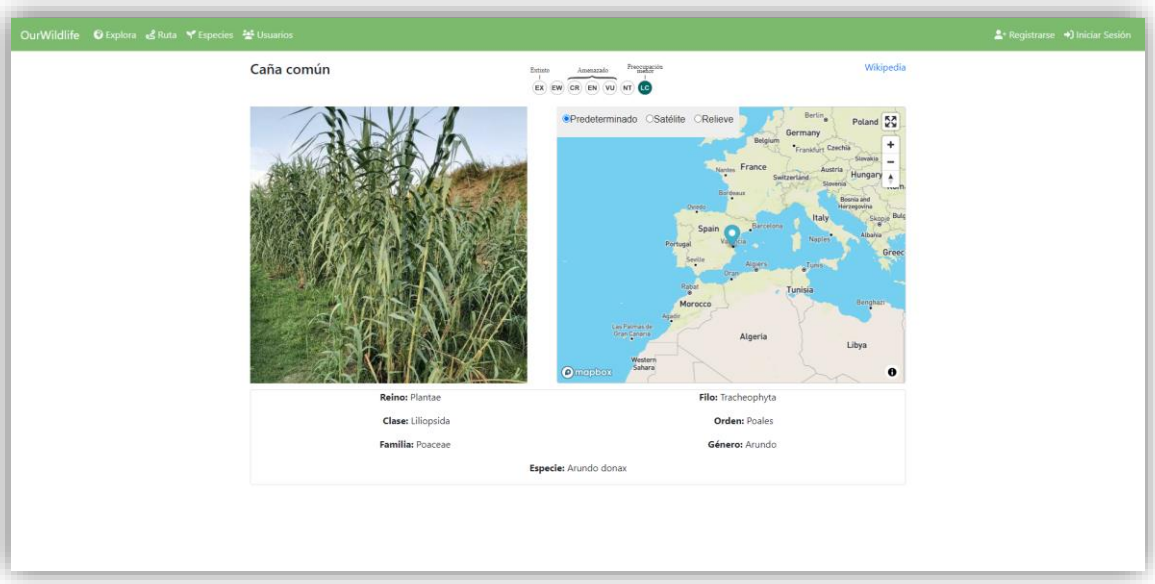

Figura 37: Ficha Especie

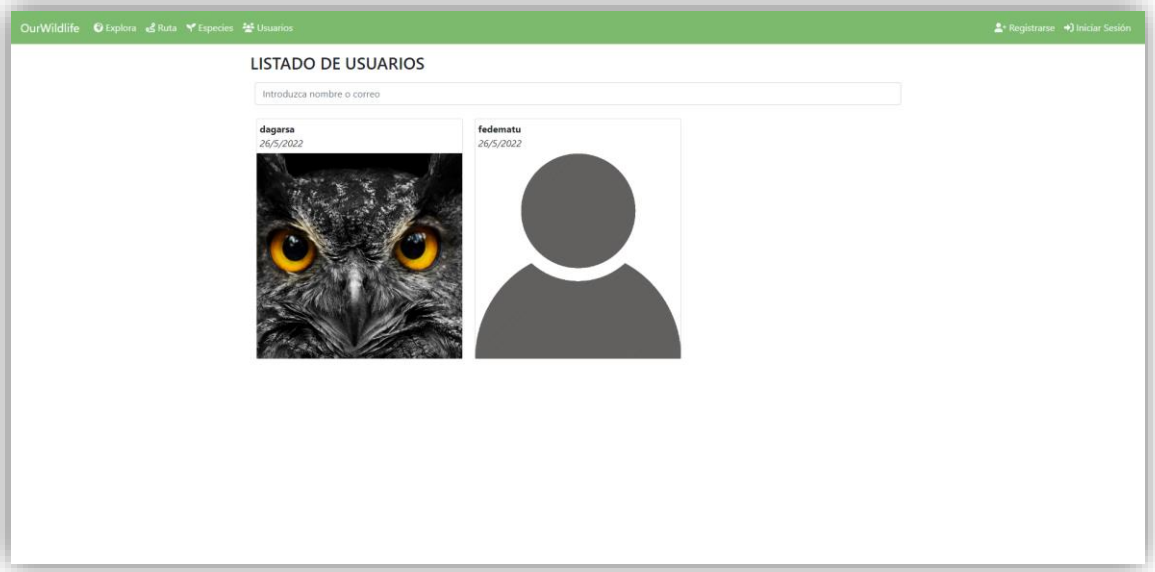

Figura 38: Listado Usuarios

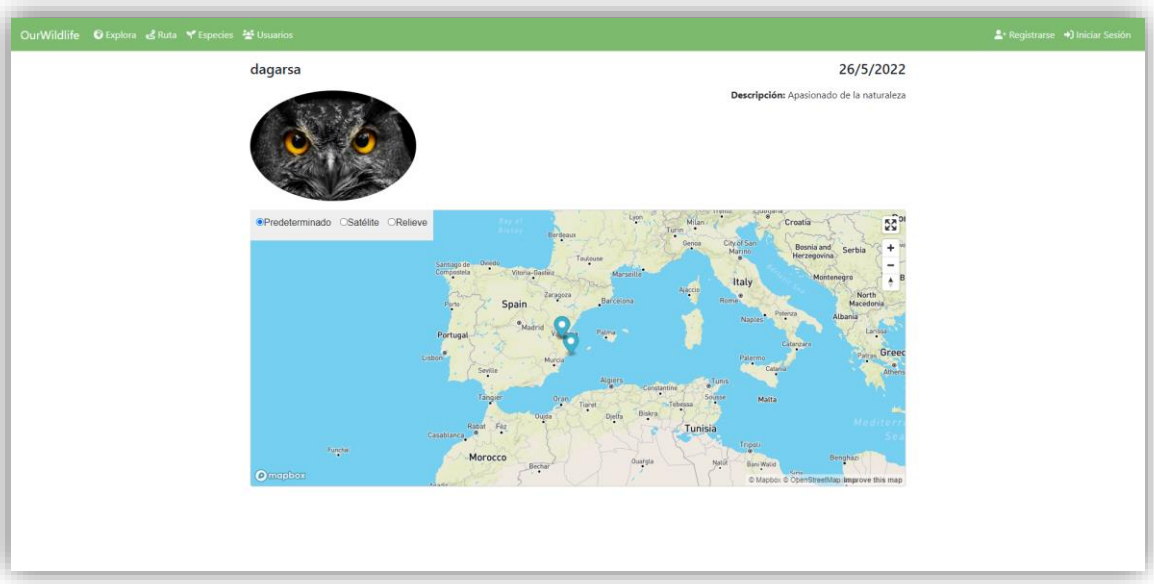

Figura 39: Ficha Usuario

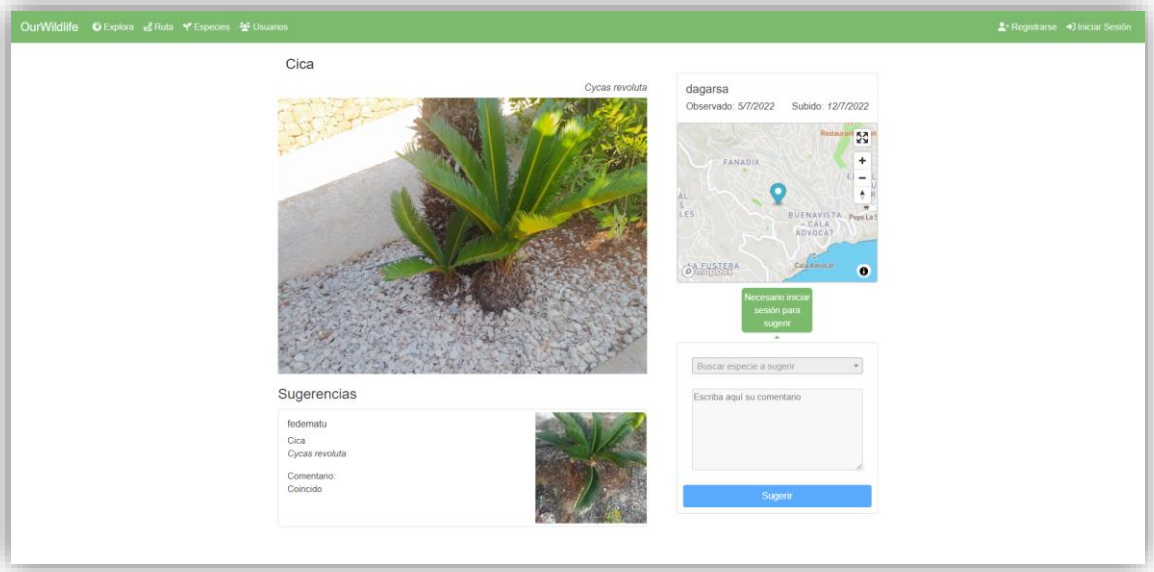

Figura 40: Ficha Observación con Usuario Anónimo

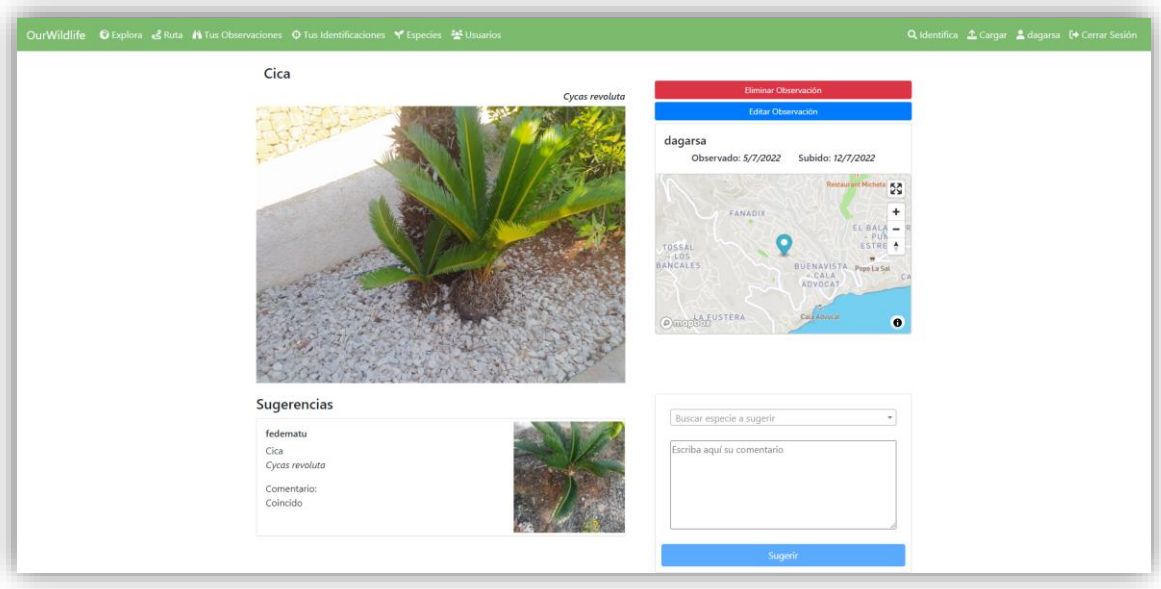

Figura 41: Ficha Observación con Usuario Registrado

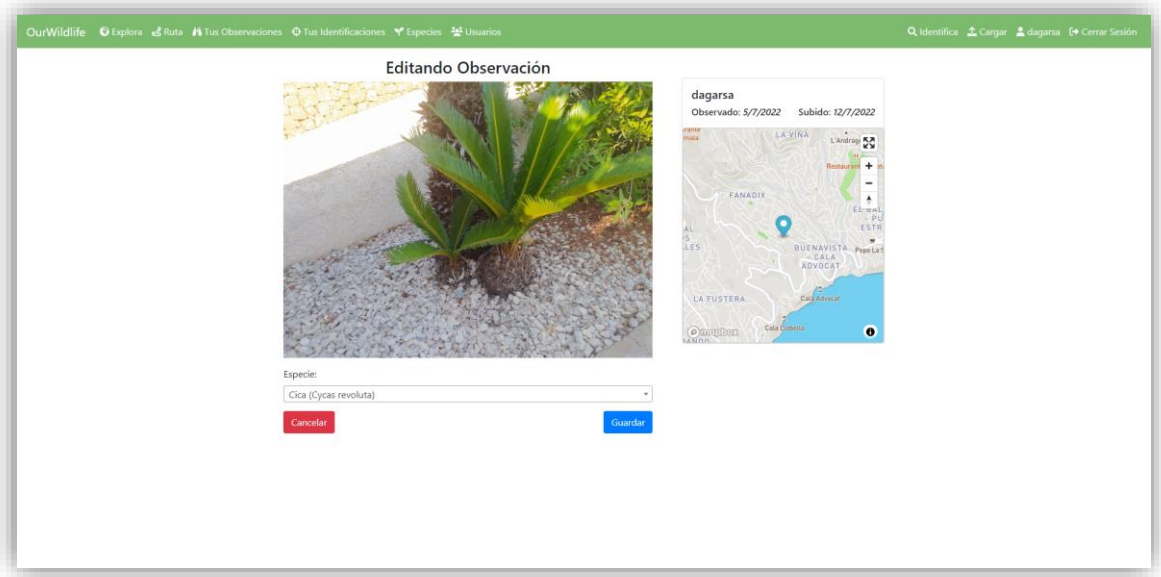

Figura 42: Edición Observación

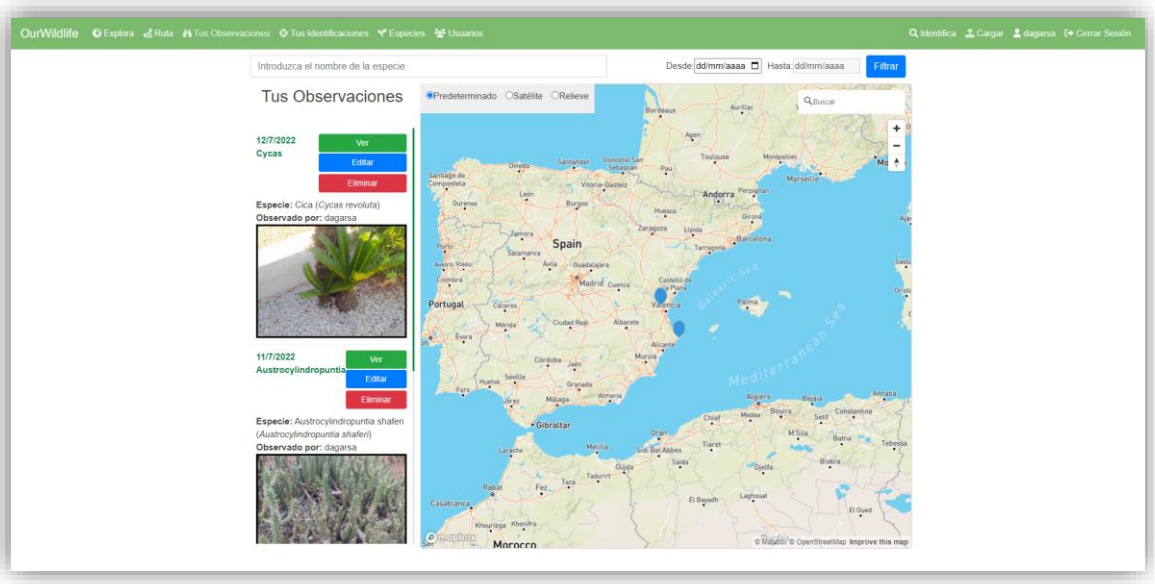

Figura 43: Tus Observaciones

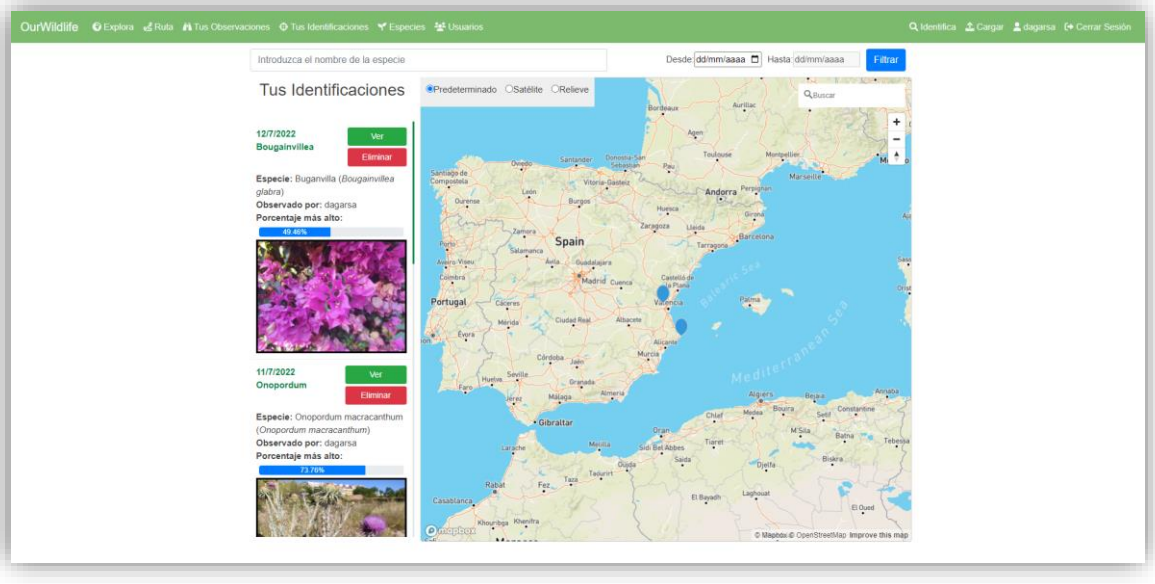

Figura 44: Tus Identificaciones

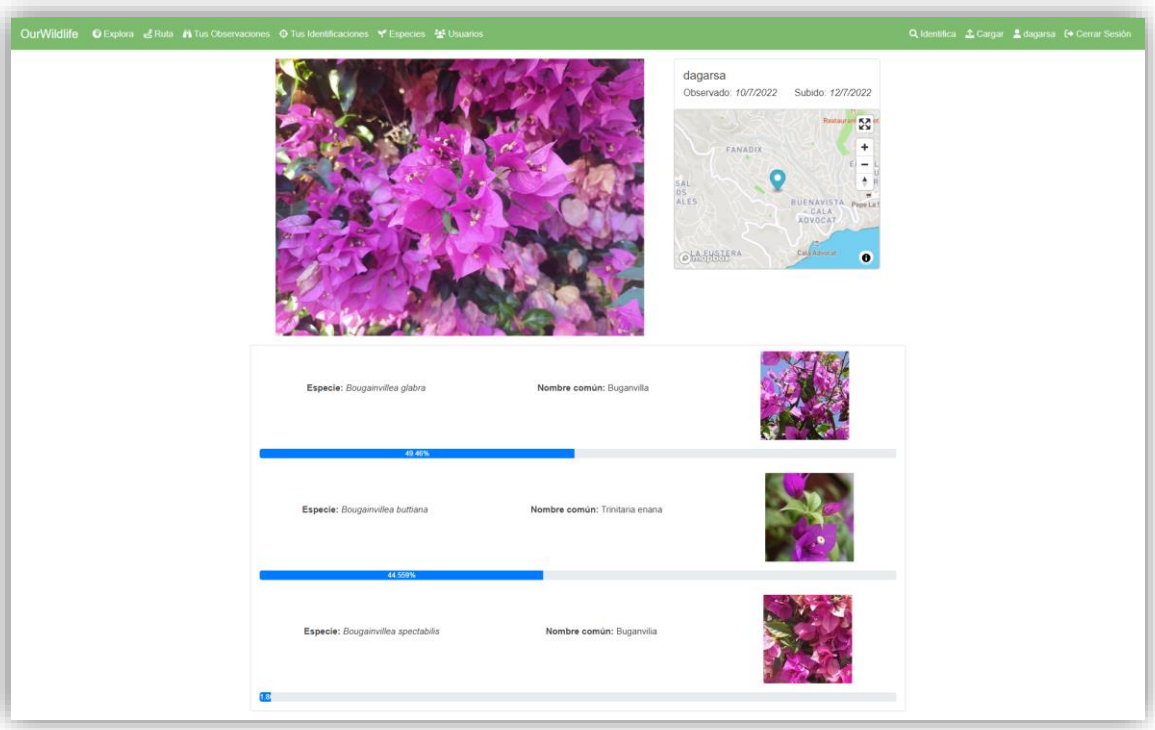

Figura 45: Ficha Identificación

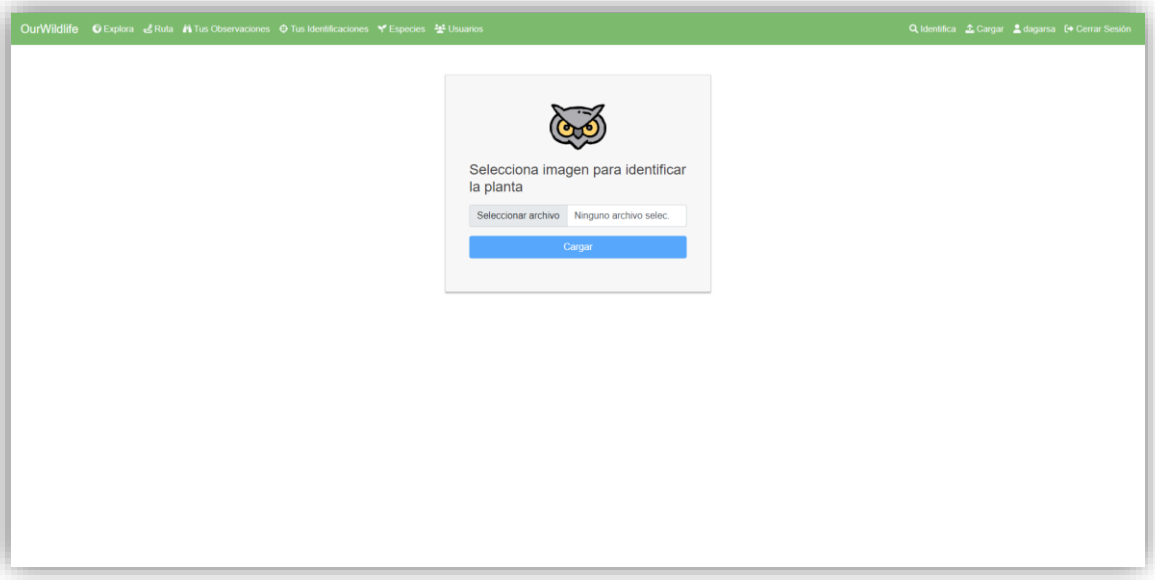

Figura 46: Cargar Identificación

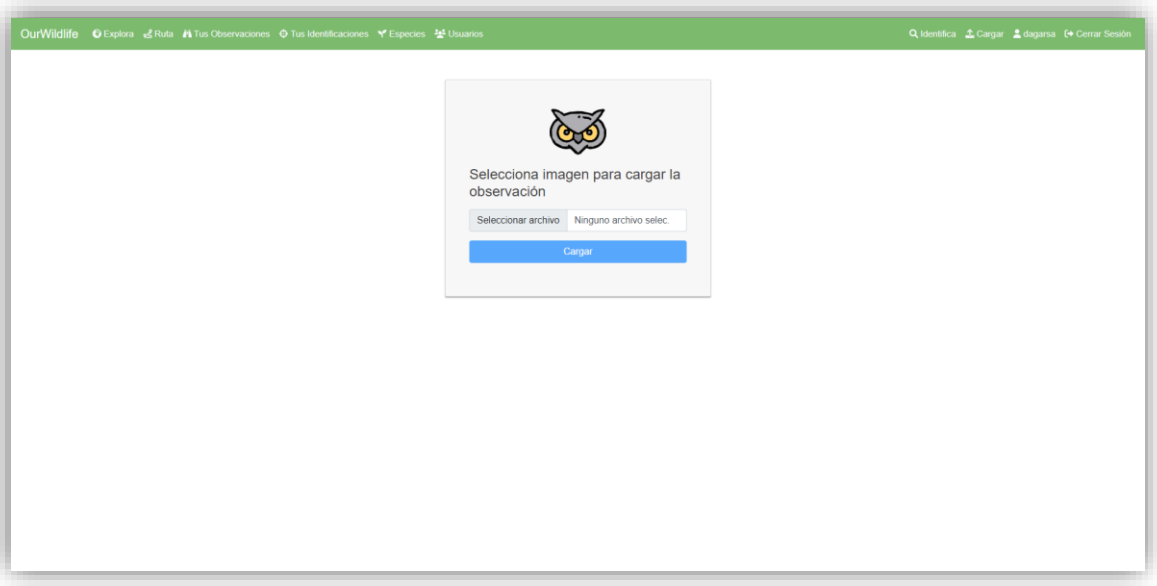

Figura 47: Cargar Observación

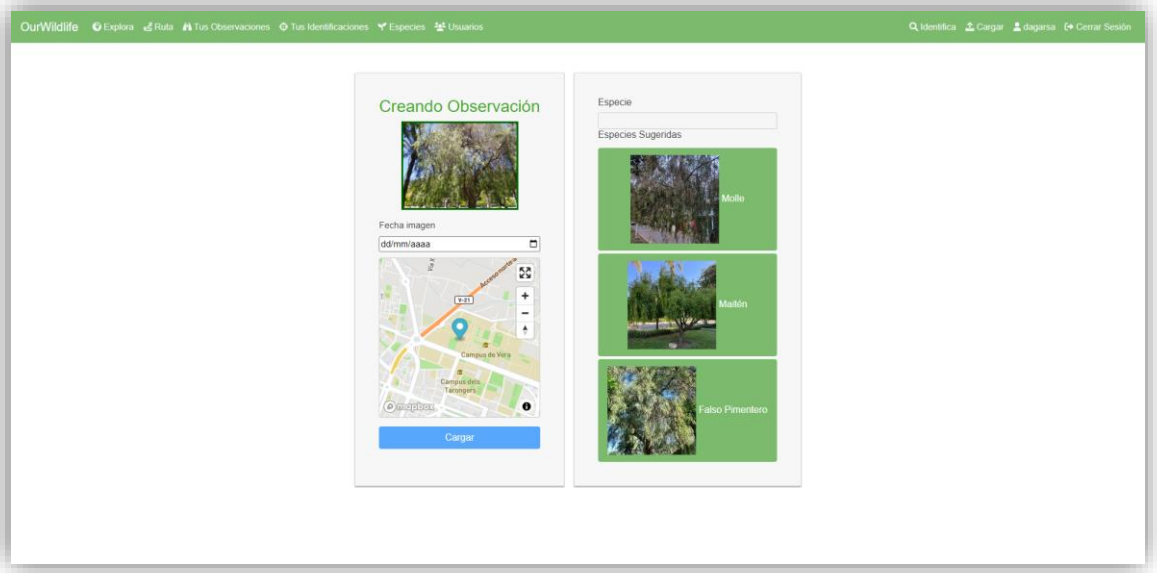

Figura 48: Formulario Cargar Observación

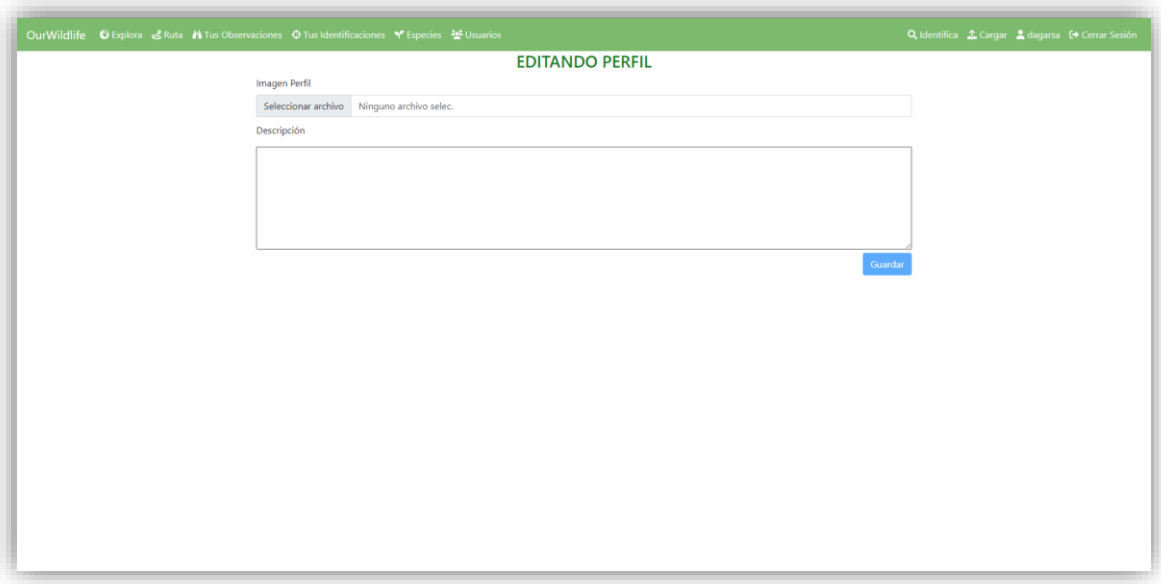

Figura 49: Edición Perfil Usuario