Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN ENXEÑARÍA DO SOFTWARE

Desarrollo de una aplicación web para la gestión conjunta del patrimonio histórico.

Estudante: Marcos Francisco Varela Marcos

Dirección: José Losada Pérez

A Coruña, febrero de 2026.

A mis abuelos.

Agradecimientos

A mi tutor José Losada, a mis profesores, compañeros y especialmente a mi familia, sin quienes no habría podido llegar hasta aquí.

Resumen

A lo largo del tiempo, se han realizado constantemente hallazgos de objetos con gran valor arqueológico y cultural de forma intencionada o accidental. En algunos de ellos, los objetos se han perdido para siempre y muchas veces aparecen en registros o patrullas rutinarias, algunos incluso decorando la ventana de su descubridor. Por su parte, el fenómeno de la detección de metales no es nuevo. Pese a que estos aparatos, en forma más o menos primitiva, han existido desde las grandes guerras del siglo pasado, ha sido en las últimas décadas cuando se han empezado a popularizar de un modo recreativo. Dada la riqueza arqueológica del sur de Europa, en países como Italia, Francia y especialmente España esto ha provocado conflictos sociales y legales entre los profesionales de la arqueología y los ciudadanos que descubren piezas, y más especialmente los usuarios de los sistemas de detección de metales o las empresas de construcción, cuyas máquinas encuentran objetos de gran valor histórico durante movimientos de tierra.

Todo ello ha llevado a las autoridades a implantar normas que restringen, aunque en muchos casos de forma descentralizada, ambigua y poco rastreable, el uso de detectores de metales, las zonas y forma de actuación en la ejecución de obras y los procedimientos a seguir cuando se produce un hallazgo accidental. A nivel nacional, dichas actividades están reguladas mediante la Ley 16/1985, de 25 de junio, del Patrimonio Histórico Español, y por las diferentes leyes del Patrimonio Histórico y Cultural de las distintas Comunidades Autónomas.

Este trabajo de fin de grado tiene como objetivo el desarrollo de una aplicación web de uso institucional, DIGREPORT, que permita a los tres agentes implicados: autoridades, profesionales de la arqueología y ciudadanos -ya sean detectoaficionados o personas que han hecho un hallazgo accidental- trabajar juntos para la protección y la correcta gestión del patrimonio histórico. Este sistema ofrece funcionalidades tales como el registro de un hallazgo porparte de un particular, la revisión y validación del mismo realizado por un profesional, todo ello supervisado por las autoridades competentes que dispondrán, a su vez, de un amplio repertorio de información que les permita clasificar, rastrear y controlar los hallazgos realizados para su posterior procesamiento como bienes integrantes del patrimonio cultural.

Se utilizará una metodología iterativa e incremental basada en sprints, cada uno de los cuales incorporará nuevas funcionalidades respecto al anterior. Por cada uno de los mismos se realizarán las cuatro fases siguientes: análisis de requisitos,diseño, implementación y pruebas.

Abstract

Over time, archaeological valuable objects have been constantly discovered, either intentionally or accidentally. In some cases, these objects have been lost forever, and in many others they use to be discovered during routine inspections or patrols, and some even end up decorating the discoverer's window. The phenomenon of metal detecting is not new. Although such devices, in more or less primitive forms, have existed since the major wars of the last century, it is in recent decades that they have become popular as a leisure activity. Due to the archaeological richness of southern Europe, in countries such as Italy, France, and especially Spain, this has led to social and legal conflicts between archaeologists and citizens who discover artifacts, particularly users of metal detection systems or construction companies whose machinery uncovers objects of great historical value during earth movements.

All of this has led authorities to implement regulations that restrict, although in many cases in a decentralized, ambiguous, and poorly traceable way, the use of metal detectors, the areas where they may be used, the procedures to be followed during construction works, and the actions to be taken when an accidental discovery happens. At the national level, these activities are regulated by Law 16/1985 of June 25 on Spanish Historical Heritage, as well as by the various historical and cultural heritage laws of the different regions.

The objective of this project is the development of an institutional web application, DIGREPORT, which enables the three involved actors —authorities, archaeology professionals, and citizens (whether metal detecting enthusiasts or individuals who have made an accidental discovery)— to work together in the protection and good management of historical heritage. This system provides functionalities such as the registration of a discovery by an individual, its review and validation by a professional, all under the supervision of the competent authorities, who will also have access to a wide range of information allowing them to classify, track, and monitor discoveries for their subsequent processing as assets belonging to cultural heritage.

An iterative and incremental methodology based on sprints will be used, each of which will incorporate new functionalities with respect to the previous one. For each sprint, the following four phases will be carried out: requirements analysis, design, implementation, and testing.

Palabras clave:

- Patrimonio histórico
- Aplicación web
- Vue.js
- Spring Boot
- Inteligencia Artificial
- Detector de metales
- Hallazgo

- Mapa interactivo

Keywords:

- Cultural heritage
- Web application
- Vue.js
- Spring Boot
- Artificial Intelligence
- Metal detector
- Finding
- Interactive map

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1
Introducción

1.1 Motivación y problema

La situación en la que se encuentra el patrimonio histórico español es excepcionalmente grave. Se entiende como Patrimonio Histórico[1], según la Ley 16/1985, de 25 de junio, del Patrimonio Histórico Español, los inmuebles y objetos muebles de interés artístico, histórico, paleontológico, arqueológico, etnográfico, científico o técnico, documental y bibliográfico. Esto incluye los yacimientos arqueológicos y los espacios naturales cuyas peculiaridades les otorguen un valor en los aspectos anteriormente mencionados. De igual forma, se incluye también el Patrimonio Cultural Inmaterial, como los discursos famosos grabados, de acuerdo a lo que se estableczca en la legislación especial.

Así pues, es habitual, de forma subconsciente, darle la consideración de bien patrimonial solo a la Torre de Hércules, la Mezquita de Córdoba o el Anfiteatro de Mérida. Sin embargo, y a pesar de que la ley las protege de igual manera, restos menores como las pocas piedras sumergidas en musgo que pueden quedar de un castro sin excavar o los restos de cañones de bronce de antiguas baterías costeras, sufren año tras año un deterioro importante a causa de obras, fenómenos naturales, expolio profesional o simplemente, abandono.[1]

Este proyecto, basándose en el concepto del sistema PAS británico y su éxito[2], nace para facilitar la recopilación y estudio de todos esos objetos históricos testigos de lo cotidiano, con la colaboración de todos los agentes sociales, y con el fin último que le atribuye su autor a la tecnología: mejorar la vida de las personas, por encima de servir como un instrumento productivo y comercial.

[1] https://www.boe.es/buscar/act.php?id=BOE-A-1985-12534
[2] https://www.bbc.com/news/articles/cm2yerrqel7o

1.1.1 Los detectores de metales: entre la amenaza y la colaboración

Uno de los temas más polémicos y debatidos, especialmente en los últimos años, es el del uso de detectores de metales. Estos sistemas existen desde las grandes guerras del siglo XX y uno de sus usos principales fue la localización de minas y explosivos. Sin embargo, desde finales del pasado siglo su uso recreativo se ha popularizado, viendo, en los últimos 5-10 años, un aumento considerable de su uso gracias a su difusión como afición en redes sociales. Existe un amplio debate entre los profesionales del sector con respecto al uso de estos dispositivos, mientras que las voces más conservadoras abogan por su total prohibición fuera del ámbito académico y profesional, otras más díscolas hablan de regulación justa y colaboración entre agentes sociales, reconociendo así sus beneficios. Lo cierto es que el país de Europa con regulación más laxa al respecto, el Reino Unido, es también uno de los que mejor gestiona su patrimonio histórico[2] -basta una visita a cualquier pueblo de las islas británicas- y así lo atestigua, entre otras cosas, el éxito del sistema PAS. Además, es una terapia en auge para personas con problemas de salud mental, destacando el colectivo de los veteranos de conflictos bélicos[3].

Frente a la tendencia legislativa actual, que tiende a la prohibición todas aquellas actividades que supongan un riesgo aunque realmente no causen daño, este proyecto pretende abrir la puerta a la resolución de un problema que es vital para que podamos, con colaboración y con participación desde la sociedad civil, mejorar la conservación de nuestro patrimonio histórico.

Capítulo 2

Estado del arte

En la actualidad la tecnología juega un papel fundamental en la protección del Patrimonio Histórico. Desde herramientas de prospección de suelos, bases de datos de bienes inventariados o el uso de drones y óptica avanzada para vigilar espacios protegidos y evitar saqueos.

En el caso de ese trabajo, se pretende crear una solución que permita desbloquear situaciones legislativas difíciles de gestionar, a la vez que abre la puerta a que la sociedad civil participe de forma activa en la protección de su historia local, pues no deja de ser el ciudadano de a pie el que convive con el patrimonio día a día.

Existen algunos proyectos con similares objetivos pese a que muy pocos lo hacen de forma automatizada y completamente digital.

2.1 Alternativas nacionales

Dédalo

El software Dédalo[1] es una solución que actúa como un gestor digital de Patrimonio Histórico y está destinado especialmente para archivos, instituciones y museos, no teniendo por ende como objetivo documentar ”sobre el terreno”, sino digitalizar lo ya documentado. Además, siendo una herramienta on-premise (es decir, corriendo en un servidor que existe en las instalaciones del propio usuario) su complejidad técnica aumenta debido a instalaciones y además la convierte en impracticable para el ciudadano.

Pese a ello, su amplia funcionalidad así como su API para integrarlo en otros servicios lo convierten en un software importantísimo en este campo.

[1] https://dedalo.dev/?lang=es

Tekeum

Es un Software As Service (SaaS) desarrollado por el grupo Tuinbit, en Ciudad Real, que se focaliza fundamentalmente en archivo de piezas museísticas y gestión interna de colecciones, así como de la experiencia de usuario dentro de los museos, por ejemplo creando audioguías o gamificación[2]. Permite almacenar registros patrimoniales, pero su objetivo primordial es la gestión interna museística e institucional.

CA-Sistema de Gestión de Colección

Es una herramienta open-source desarrollada por IBAISCANBIT, en Gasteiz, que sigue la línea de las dos anteriores. Es también una solución on-premise[3].

2.2 Alternativas internacionales: PAS

El proyecto Portable Antiquities Scheme (PAS) es, con total seguridad, el que más similitudes comparte con ese proyecto. Gestionado por el Museo Británico, su principal cometido es documentar y hacer partícipe al ciudadano del cuidado diario de su historia[4]. Pese a esto, no es totalmente digital, pues para reportar los hallazgos se debe acudir a una de las 80 oficinas que el la organización tiene repartidas a lo largo de Inglaterra y Gales[5].

Con una efectividad ampliamente demostrada [4, 5], cuenta con una guía explicativa para reportar los hallazgos, un manual del buen uso de los detectores de metales y una base de datos completa para que los ciudadanos interesados puedan documentar todos los hallazgos realizados, contribuyendo de esta forma a difundir la historia del país y a mantener los objetos que la componen.

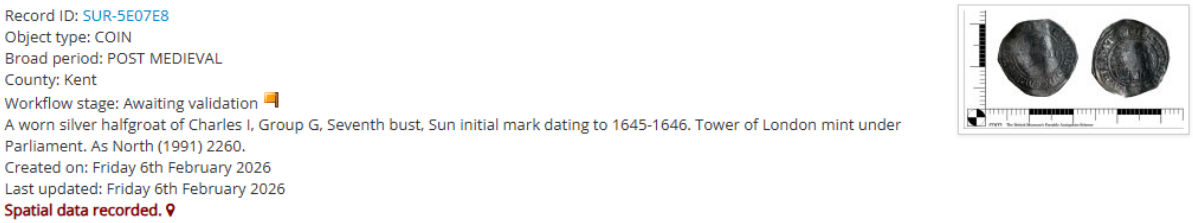

Figura 2.1: Ejemplo de hallazgo consultado en la plataforma del PAS

[2] https://tekeum.com/software-de-gestion-de-patrimonio-cultural
[3] https://ibaiscanbit.com/biblioteca/ca-sistema-de-gestion-de-coleccion-principales-ventajas/
[4] https://finds.org.uk/getinvolved
[5] https://finds.org.uk/contacts

Capítulo 3

Tecnologías y estándares

Este capítulo expone y justifica las decisiones que se han tomado con respecto a las tecnologías utilizadas para construir este sistema, por qué son beneficiosas y cómo se han utilizado para aprovechar todo su potencial.

3.1 Lenguajes de programación

Java

Figura 3.1: Logotipo de Java

Se ha seleccionado Java[6] como lenguaje principal del proyecto por su madurez y estabilidad en entornos empresariales, características fundamentales para un sistema destinado a gestionar datos sensibles y a operar en un entorno oficial. Su sistema de tipado estático fuerte permite detectar errores en tiempo de compilación, reduciendo significativamente los problemas en producción. Además, cuenta con un ecosistema extenso de librerías y marcos de desarrollo que facilitan la implementación de funcionalidades clave, tales como la persistencia de datos, seguridad o comunicación entre plataformas sin necesidad de desarrollar soluciones desde cero. Asimismo garantiza un rendimiento adecuado para gestionar potencialmente miles de registros y usuarios concurrentes. La portabilidad multiplataforma que ofrece simplifica el despliegue en diferentes infraestructuras de la administración pública, mientras que

sus versiones LTS aseguran soporte y actualizaciones de seguridad a largo plazo, aspecto fundamental para la continuidad operativa. Finalmente, la experiencia previa con el lenguaje adquirida durante la formación académica del autor le ha permitido aprovechar eficientemente todas estas ventajas técnicas.

HTML y CSS

Figura 3.2: Logotipos de HTML, CSS, JavaScript

TypeScript

Figura 3.3: Logotipo de TypeScript

Aunque HTML[1] y CSS[2] no se consideran, generalmente, lenguajes de programación puros, junto con JavaScript (y TypeScript)[3] conforman una de las herramientas más potentes y por ende más utilizadas del mundo digital para la parte de interfaz de usuario del desarrollo web. Esto garantiza que su alto nivel de estandarización y su universalidad proporcionen el potencial necesario para desarrollar aplicaciones web de alto nivel garantizando una buena experiencia de usuario, fluida y accesible.

El primero nos garantiza un diseño altamente personalizable de la estructura semántica del frontend, mientras que CSS nos permite, con sus estilos, personalizar visualmente dicha estructura gracias a su amplísimo ecosistema y sus inagotables posibilidades.

[1] https://es.wikipedia.org/wiki/HTML
[2] https://es.wikipedia.org/wiki/CSS
[3] https://es.wikipedia.org/wiki/JavaScript

JavaScript, por su parte, se encarga de la operatividad e interactividad del sistema y la comunicación con el frontend. La amplísima adopción de estas tecnologías en todos los navegadores modernos permite la utilización de este sistema en cualquier dispositivo sin necesidad de instalaciones adicionales. TypeScript[4] no es más que un *superset* (ampliación) de JavaScript que incluye tipado estático, más recomendable para aplicaciones que requieran robustez, documentación y mantenibilidad[7].

3.2 Herramientas y marcos de desarrollo

Spring Boot

Figura 3.4: Logotipo de Spring Boot

Spring Boot[5] es, de facto, el framework empresarial estándar para Java. Su enorme comunidad y su cantidad de documentación conforman un ecosistema ideal y casi infinito de posibilidades. Con respecto a su ”hermano mayor” Spring Framework, ahorra muchísimo tiempo en el desarrollo al ser modular y autoconfigurable, y entre sus características más destacadas y que más han pesado a la hora de ser elegido como marco de trabajo principal, están la inyección de dependencias nativa, muy importante a la hora de implementar ciertas arquitecturas; soporte nativo para API REST, lo que nos permitirá la integración con servicios externos, y un servidor embebido sin necesidad de configurar uno externo. Por estas razones, se convierte en el ecosistema perfecto para el desarrollo de un software de estas características. Además, su módulo Security, en combinación con JWT, ha permitido implementar múltiples características de autenticación y seguridad. La gestión de dependencias se realiza mediante la herramienta Apache Maven[6], que ha servido para simplificar la configuración inicial del proyecto, la compilación y la instalación de dependencias y librerías.

Hibernate JPA

[4] https://www.typescriptlang.org
[5] https://spring.io/projects/spring-boot
[6] https://maven.apache.org

Figura 3.5: Logotipo de Hibernate

El ORM Hibernate[7] es una implementación del estándar JPA (Java Persistence API) que permite, entre otras cosas, trabajar con objetos en vez de con SQL manual, lo cual es útil para optimizar el modelo de datos; abstraer los elementos del modelo de la base de datos (por lo que permitiría trabajar con cualquier sistema); o realizar validaciones a nivel de entidad mediante anotaciones.

Postgre SQL

Figura 3.6: Logotipo de PostgreSQL

En cuanto al motor de base de datos, se ha elegido PostgreSQL[8], un sistema gestor de base de datos relacionales de código abierto y cuya robustez, fiabilidad y resiliencia en entornos empresariales ha sido ampliamente demostrada, así como su adopción por parte de proyectos del sector público[8].

Vue.js

[7] https://hibernate.org
[8] https://www.postgresql.org/about/

Figura 3.7: Logotipo de Vue.js

Vue[9] es un framework progresivo inicialmente planteado para JavaScript que también implementa compatibilidad con TypeScript. Sus características principales son la facilidad para proporcionar un equilibrio entre flexibilidad y estructura, la suavidad de su curva de aprendizaje con respecto a otros competidores como React y la sencillez a la hora de integrarlo con numerosas librerías que han permitido desarrollar las funcionalidades del proyecto.

Figura 3.8: Logotipo de Vite

Vite[10] es la herramienta de construcción utilizada para el desarrollo frontend, proporcionando un servidor con Hot Module Replacement, que permite al instante ver cambios en la interfaz de usuario cuando estos se producen en el código lo que agiliza enormemente el desarrollo.

JUnit5

Figura 3.9: Logotipo de JUnit 5

[9] https://vuejs.org
[10] https://vite.dev

JUnit 5 es la herramienta de testing utilizada para implementar la automatización de pruebas unitarias por la alta integración que tiene con Spring Boot y su ecosistema moderno de diseño de tests mediante anotaciones (@Test, @BeforeEach, @ParameterizedTest) o aserciones expresivas.

Otros

A mayores de las ya mencionadas anteriormente, se han utilizado otras herramientas más específicas que también convendría citar dado el impacto que han tenido en el desarrollo del proyecto:

Pinia:[11] Proporciona un sistema de gestión de estado centralizado para Vue.js que almacena datos compartidos entre componentes, tales como la sesión del usuario autenticado o los hallazgos cargados en un determinado momento.

Axios:[12] Cliente HTTP para comunicaciones con el backend REST de Spring Boot desde el frontend Vue.js.

npm:[13] Gestor de paquetes que controla las dependencias del proyecto frontend mediante el archivo package.json.

Mockito:[14] Herramienta para JUnit que facilita la construcción (mock) de objetos para realizar los tests, permitiendo aislar el código que se va a probar de sus dependencias.

3.3 Entorno y herramientas complementarias

Control de versiones

GitHub:[15] Plataforma de control de versiones por excelencia que nos permite mantener un historial de cambios, recuperación rápida ante errores y trabajo simultáneo mediante ramas de trabajo.

Postman: Pruebas de la API

Postman[16] es un software que permite documentar y probar la comunicación con los endpoints[17] de la API del backend de forma interactiva y personalizada para validar el funcionamiento de los controladores.

[11] https://pinia.vuejs.org
[12] https://axios-http.com/es/docs/intro
[13] https://www.npmjs.com
[14] https://site.mockito.org
[15] https://github.com
[16] https://www.postman.com
[17] Un endpoint de API es una ubicación digital donde una interfaz de programación de aplicaciones (API) recibe llamadas de API, también conocidas como solicitudes de API, para recursos en su servidor. (IBM)

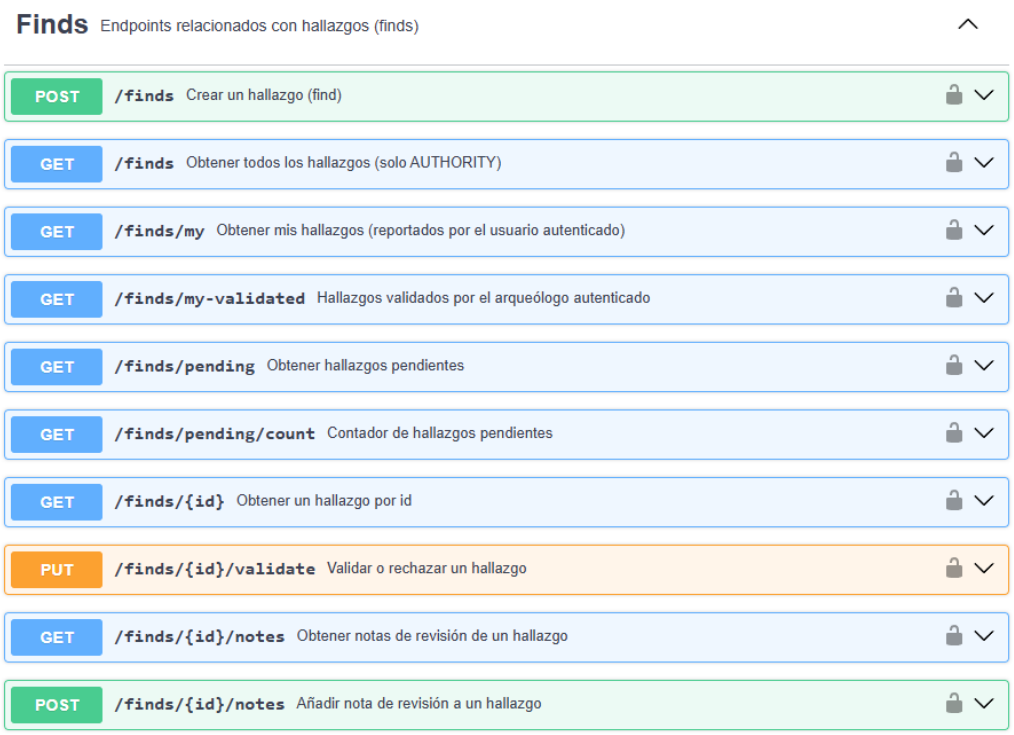

Figura 3.10: Algunos endpoints HTTP del sistema

Entorno Back-End: IntelliJ IDEA Ultimate

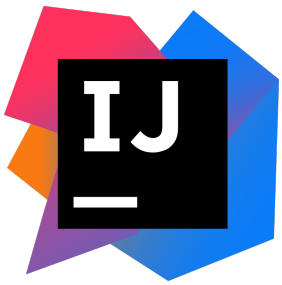

Figura 3.11: Logotipo de IntelliJ IDEA

El IDE escogido para el desarrollo del backend o lado cliente ha sido JetBrains IntelliJ IDEA[18] en su versión Ultimate con licencia educativa, dada su potencia y su amplio abanico de características, como el autocompletado inteligente y las potentes integraciones con Spring Boot, PostgreSQL y soporte nativo para Maven.

[18] https://www.jetbrains.com/idea/

Entorno Front-End: Visual Studio Code

Figura 3.12: Logotipo de Visual Studio Code

Por otra parte, se ha escogido el IDE Microsoft Visual Studio Code[19] para el desarrollo del front-end o lado servidor por su ligereza, su alta personalización o la función Live Server, que consiste en lanzar un servidor de desarrollo local que facilita el desarrollo y el seguimiento de los cambios realizados en tiempo real.

3.4 Estándares

JWT: JSON Web Token

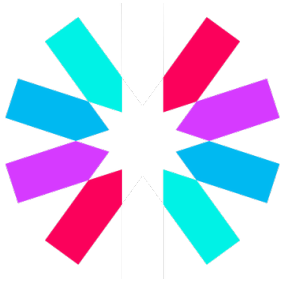

Figura 3.13: Logotipo de JWT

JWT es un estándar abierto definido en la RFC 7519[20] utilizado para intercambiar información segura entre dos partes, que consiste en enviar un JavaScript Object Notation (JSON) codificado que contiene una serie de reclamos y una firma. Es ampliamente utilizado en sistemas de autenticación de arquitecturas basadas en microservicios y una de sus ventajas es que es *stateless* (no tiene estado).

[19] https://code.visualstudio.com
[20] https://datatracker.ietf.org/doc/html/rfc7519

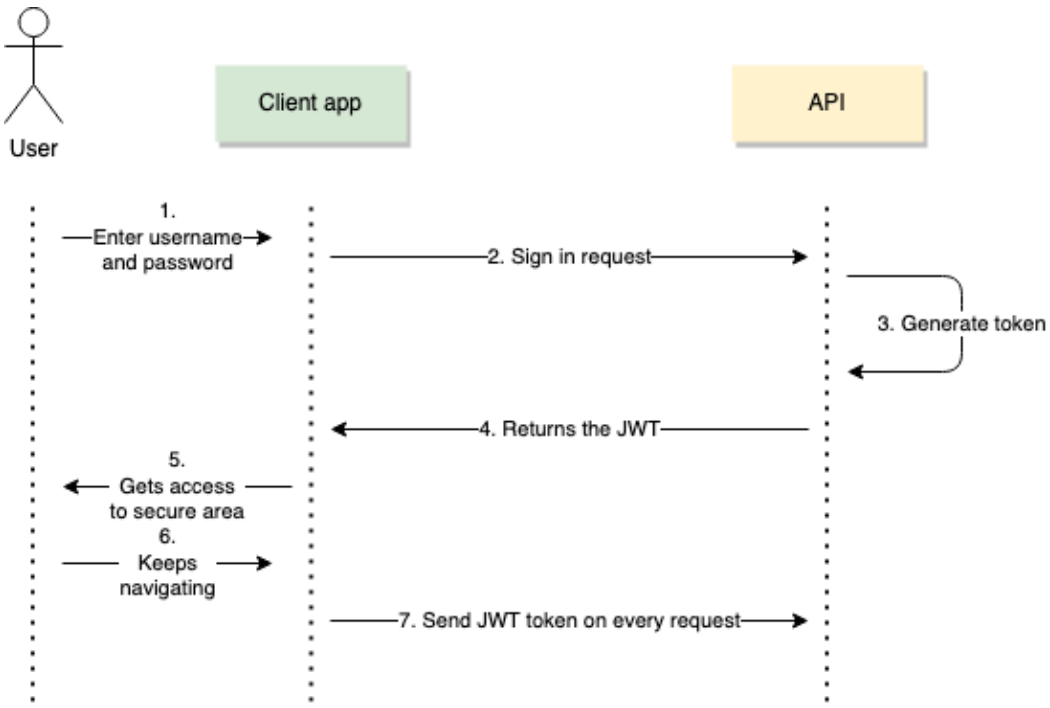

Figura 3.14: Diagrama de secuencia de JWT.

3.5 Arquitectura hexagonal

Este proyecto se construye utilizando el patrón denominado de arquitectura hexagonal o de puertos y adaptadores. Propuesta por Alistair Cockburn en 2005[9], permite construir sistemas con un nivel de acoplamiento mínimo que permite probar los diferentes componentes de manera independiente sin depender de la interfaz de usuario ni de los almacenes de datos.

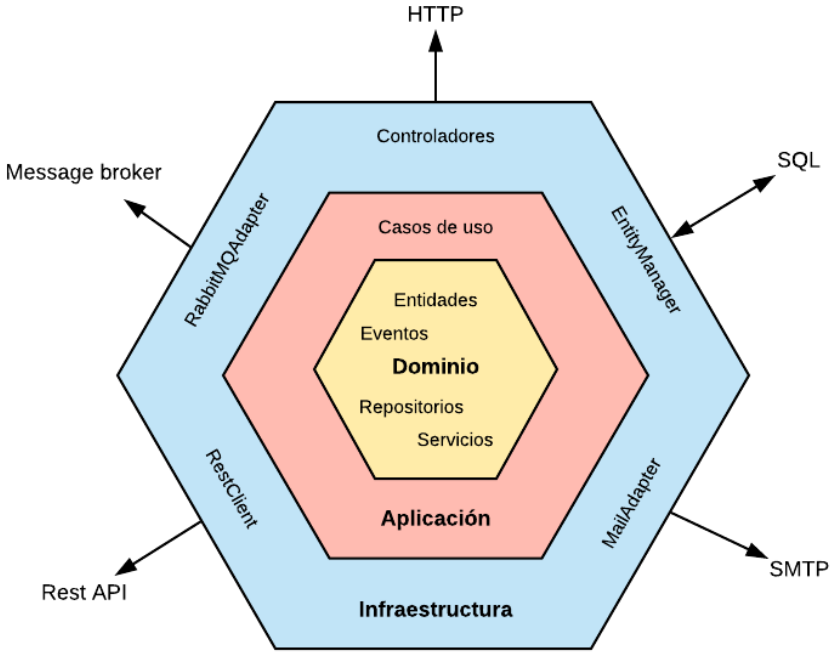

Figura 3.15: Ejemplo de arquitectura hexagonal (https://medium.com)

En el caso de un sistema pensado para operar en un entorno oficial es indispensable disponer de la máxima flexibilidad a la hora de mantener, actualizar o ampliar sus características y funcionalidades. Dicho patrón permite realizar estas intervenciones con gran facilidad dado que es independiente de la tecnología y el cambio en la misma será totalmente ajeno a la lógica de negocio o empresarial de la aplicación, puesto que esta se comunicará con los componentes externos a través de interfaces (contratos) denominados *puertos*, utilizando los *adaptadores* para traducir la información con dichos componentes. Esto será de gran utilidad si se pretende, por ejemplo, integrar en sistemas de software de distintas administraciones que estén construidos de manera diferente y con tecnologías diversas.[10]

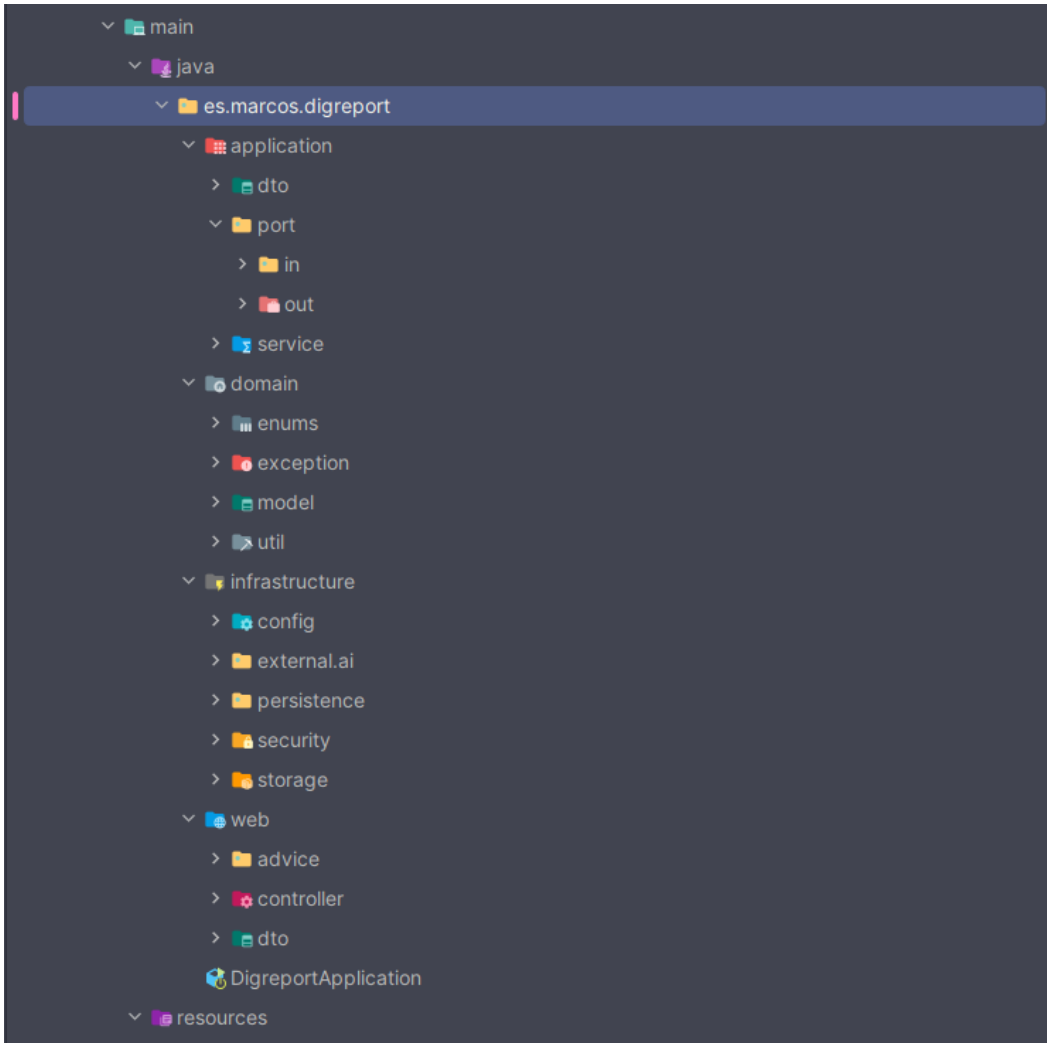

Figura 3.16: Estructura general que ilustra la arquitectura hexagonal escogida.

Capítulo 4

Introducción al desarrollo

Este proyecto tiene como objetivo la creación de una plataforma en línea que facilite la gestión del Patrimonio Histórico de manera colaborativa, no solo con funciones de inventariado y consulta sino estableciendo procesos que permitan a los ciudadanos colaborar en todas estas tareas bajo la supervisión de profesionales autorizados y organismos de la Administración Pública.

Para alcanzar estos objetivos se han definido una serie de metodologías basadas en lo que hoy se conoce como desarrollo ágil o *agile*, basándose en criterios tales como el tamaño del equipo o la tolerancia a posibles cambios en los requisitos (por ejemplo, se ha atendido a cambios recientes en las leyes de Patrimonio de las Comunidades Autónomas aunque, finalmente, no hayan tenido impacto en el desarrollo).

4.1 Scrum

Scrum es la denominación de la más conocida de las metodologías ágiles, y sus pilares fundamentales son la flexibilidad, la adaptabilidad y la transparencia.

Dentro de esta metodología se definen tres roles fundamentales que participarán a lo largo de todo el ciclo de vida del software: *Product Owner*, *Scrum Master* y *Development Team* o Equipo de Desarrollo.

Product Owner

Es el encargado de velar por las necesidades de los clientes o *stakeholders*, definiendo para ello las características que ha de tener el producto final y organizando los *sprints*.

Scrum Master

Es el experto en Scrum del equipo y su función es supervisar el cumplimiento y seguimiento de los principios y prácticas de la metodología.

Development Team

El equipo de desarrollo será quien trabaje en el proyecto propiamente dicho, autoorganizándose y colaborando con los otros roles, para construir los distintos incrementos del mismo.

4.1.1 Los sprints

En Scrum, un sprint es como se define al periodo de trabajo y generalmente tiene una duración predefinida, que suele variar entre una semana y cuatro semanas, siendo dos lo más común. Se constituye de las fases de planificación, implementación, revisión y retrospectiva. A lo largo de las mismas, se implementarán los nuevos requisitos correspondientes previamente especificados por el Product Owner, conociéndose este conjunto como *incremento*. Cada una de estas versiones deberá poder ser entregada al cliente.

4.1.2 Artefactos clave

Historias de usuario

Es el artefacto atómico de Scrum y hace referencia a cualquier elemento que componga el proyecto, pudiendo consistir en una nueva funcionalidad, el arreglo de un error o mejoras en el rendimiento.

Pila de producto

La pila de producto o *product backlog* es el conjunto de todas las historias de usuario del proyecto priorizadas según los criterios acordados con el equipo.

Pila de sprint

La pila de sprint o *sprint backlog* es el subconjunto de la pila de producto correspondiente a un sprint determinado.

Incremento

Es el resultado final de las nuevas implementaciones realizadas en cada sprint.

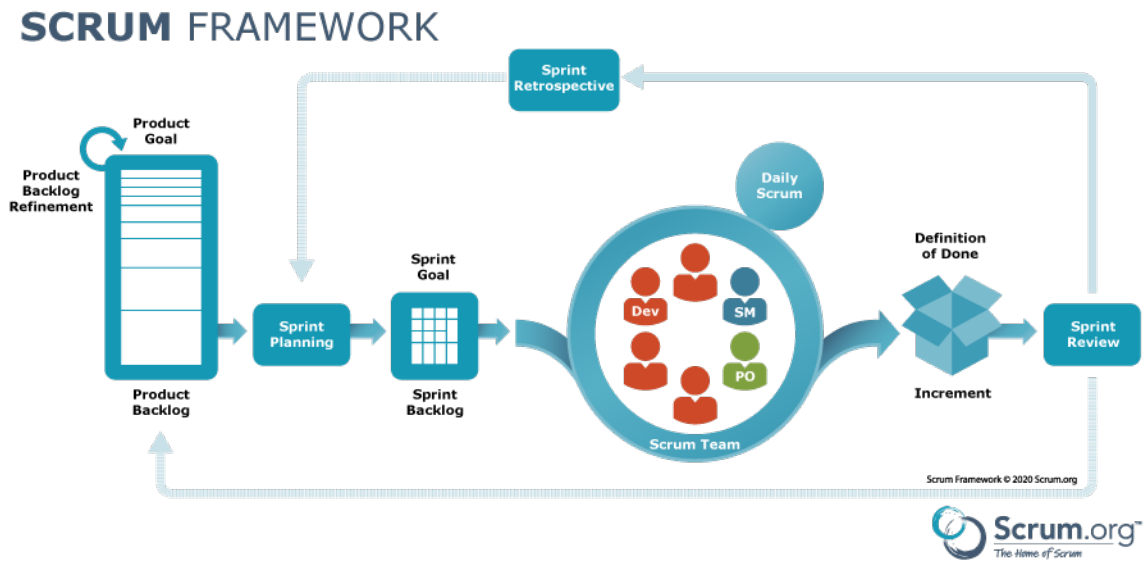

Figura 4.1: Flujo de un proyecto Scrum (scrum.org)

En este proyecto tanto el rol de Development Team como el de Product Owner corresponderá al alumno, mientras que el de Scrum Master será desempeñado por el tutor del trabajo.

Capítulo 5

Viabilidad

Con el fin de garantizar la viabilidad del proyecto y facilitar la toma de decisiones, se ha realizado un análisis exhaustivo de su viabilidad tanto técnica como económica.

5.1 Viabilidad técnica

La infraestructura de desarrollo y despliegue se compone de un entorno local, el ordenador de sobremesa del alumno, buscando diseñarlo de la forma más parecida posible a un entorno real de producción. El servidor de aplicaciones ejecuta tanto el back-end Spring Boot como sirve los recursos estáticos del frontend compilado. La base de datos se aloja en el mismo servidor. Esta configuración ha probado ser suficiente para validar todos los requisitos funcionales establecidos.

El sistema deberá incorporar funcionalidades de análisis de imágenes mediante integración con inteligencia artificial, en concreto para generar descripciones automáticamente, reduciendo la carga cognitiva del usuario a la vez que facilita la identificación de objetos. Esta funcionalidad muestra como este sistema y otros de sus mismas características pueden incorporar tecnologías punteras y emergentes. El almacenamiento de imágenes se gestiona mediante archivos locales con referencias en la base de datos. Este enfoque es idóneo y suficiente para las características de un proyecto académico y permite escalar el proyecto a soluciones *cloud* ante una eventual puesta en producción.

5.2 Viabilidad económica

5.2.1 Personal

El equipo de desarrollo estará compuesto por dos personas que ocuparán los distintos roles del desarrollo y a efectos de estimar su viabilidad económica, se han realizado las siguientes estimaciones:

Salario medio programador junior en España: ∼11€/h [1]

Salario medio Scrum Master en España: ∼22€/h [2]

Carga fiscal del salario medio en España, incluyendo costes para la empresa: ∼52% [11]

Rol	Horas trabajadas	Coste total
Developer/Product Owner	420h	4.620€
Scrum Master	54h	1.188€
Subtotal	**474h**	**5.808€**
Carga fiscal (52%)	-	3.020€
TOTAL	**474h**	**8.828€**

Cuadro 5.1: Desglose de costes estimados de personal

En contraste con los datos reales:

[1] https://www.glassdoor.es/Sueldos/programador-junior-sueldo-SRCH_KO0,18.htm
[2] https://www.glassdoor.es/Sueldos/scrum-master-sueldo-SRCH_KO0,12.htm

Rol	Horas trabajadas	Coste total
Developer/Product Owner	450h	4.950€
Scrum Master	57h	1.254€
Subtotal	**507h**	**6.204€**
Carga fiscal (52%)	-	3.226€
TOTAL	**507h**	**9.430€**

Cuadro 5.2: Desglose de costes reales de personal

El coste total real de personal ascendió a 9.430€, lo que se traduce en una desviación del 6,8% respecto a la planificación inicial de 8.828€. Esta desviación se debe a las 30 horas extras invertidas por el equipo de desarrollo durante el sprint 5, lo que indica una planificación ajustada pero mayormente precisa. El sobrecoste fue aceptable y ascendió a 602€.

A nivel técnico, esta desviación se detalla más adelante durante la especificación de sprints.

5.2.2 Material

Concepto	Coste
Ordenador personal (desarrollo)	1.600€
Monitor y periféricos	400€
Licencias de software	0€
TOTAL	**2.000€**

Cuadro 5.3: Desglose de costes materiales

Todo el software utilizado en el desarrollo del proyecto corresponde a herramientas open source (Vue.js, Spring Boot, PostgreSQL, Visual Studio Code) o licencias educativas previamente disponibles (IntelliJ IDEA Ultimate), por lo que no se han generado costes adicionales en este aspecto.

5.3 Planificación de los sprints

5.3.1 Sprint 1 – Puesta en marcha

Duración: 2 semanas (60 horas)

Este sprint inicial se centró en configurar el entorno de desarrollo y definir la arquitectura del sistema. Las principales tareas incluyeron:

- Configuración del repositorio Git y estructura de carpetas
- Instalación y configuración de Spring Boot
- Configuración de Vue.js 3 con Vite y rutas
- Configuración de PostgreSQL

Al finalizar el sprint, se disponía de un esqueleto funcional del proyecto.

5.3.2 Sprint 2 – Gestión de usuarios y roles

Duración: 2 semanas (60 horas)

El segundo sprint abordará la implementación del sistema de autenticación y autorización, fundamental para el resto de funcionalidades. Se desarrolló un sistema completo de gestión de usuarios con control de acceso basado en roles, inicialmente utilizando Spring Security e implementando JSON Web Token (JWT) en las fases finales del sprint. Las tareas principales fueron:

- Implementación de registro de usuarios con validación de datos
- Definición de roles de usuario: Arqueólogo (posteriormente renombrado a Profesional), Usuario y Autoridad (rol administrador)
- Middleware de autorización en backend
- Sistema de autenticación mediante JWT
- Gestión de perfiles de usuario y actualización de información personal

Como resultado, se logrará un sistema robusto para autenticar usuarios antes de consumir los recursos, que servirá de base para el resto de funcionalidades.

5.3.3 Sprint 3 – Gestión de hallazgos y validación

Duración: 2 semanas (60 horas)

Este sprint implementará la gestión básica de hallazgos. Las funcionalidades implementadas fueron las siguientes:

- Formulario de registro de hallazgos con validaciones en back-end y frontend.
- Listado de hallazgos.
- Visualización detallada de hallazgos individuales.
- Sistema de validación de hallazgos por parte de profesionales.

Al concluir este sprint, los usuarios podrán registrar, consultar y validar hallazgos arqueológicos de manera elemental pero funcional.

5.3.4 Sprint 4 – Panel de administración

Duración: 2 semanas (60 horas)

Este sprint se enfocará en desarrollar herramientas de administración del sistema, permitiendo supervisar la actividad general:

- Panel con estadísticas generales del sistema.
- Panel con la lista de todos los hallazgos reportados en el sistema.
- Gestión de usuarios: activación, desactivación y cambio de roles.

Al finalizar, los administradores dispondrán de herramientas completas para supervisar y gestionar todos los aspectos del sistema de manera centralizada.

5.3.5 Sprint 5 – Gestión de zonas protegidas

Duración: 3 semanas (90 horas)

Este sprint, que implementará la gestión integral de zonas protegidas, se divide en las siguientes tareas:

- Diseño del modelo de datos para zonas protegidas.
- Actualización de la base de datos para modelar estos nuevos elementos.
- Integración con Leaflet para disponer de un mapa interactivo.

- Implementación de dibujo de polígonos y agregación de marcadores en el mapa.
- Sistema CRUD para gestión de zonas protegidas por el rol Autoridad.
- Visualización de zonas protegidas sobre el mapa base
- Gestión de permisos específicos para creación y modificación de zonas

El resultado será un sistema completo de gestión territorial que permita a los administradores delimitar áreas de especial protección para garantizar las buenas prácticas en materia patrimonial.

5.3.6 Sprint 6 – Gestión de la reputación

Duración: 2 semanas (60 horas)

El sexto sprint implementará un sistema de reputación para fomentar la participación activa y premiar contribuciones de calidad. Además, se implementará también la carga de imágenes de hallazgos y su procesamiento por Inteligencia Artificial para generar una descripción detallada del objeto. Se desarrollarán las siguientes funcionalidades:

- Sistema de puntos por acciones (registro de hallazgos con validaciones)
- Ranking público de usuarios más activos
- Insignias por posiciones
- Integración con Google Gemini API
- Diseño de un *system prompt* optimizado para garantizar la máxima precisión en las descripciones
- Actualización de la interfaz de usuario para permitir generar descripciones por IA.
- Actualización de la base de datos para modelar la carga de imágenes.

Como resultado, se obtendrá una característica que premie la participación continuada y ética de los usuarios.

5.3.7 Sprint 7 – Memoria

Duración: 6 semanas (60 horas)

El sprint final se dedica a la elaboración de la memoria del proyecto. Teniendo menor intensidad horaria semanal (10 horas), su duración es de seis semanas para garantizar una revisión exhaustiva y cuidadosa. Las actividades realizadas serán:

- Redacción de capítulos
- Documentación técnica de arquitectura, buenas prácticas y decisiones de diseño
- Análisis de viabilidad técnica y económica
- Documentación de resultados y conclusiones
- Revisiones y correcciones
- Preparación de la presentación del proyecto

A continuación se presenta un diagrama cronológico del proyecto en reflejando la duración real de los sprints:

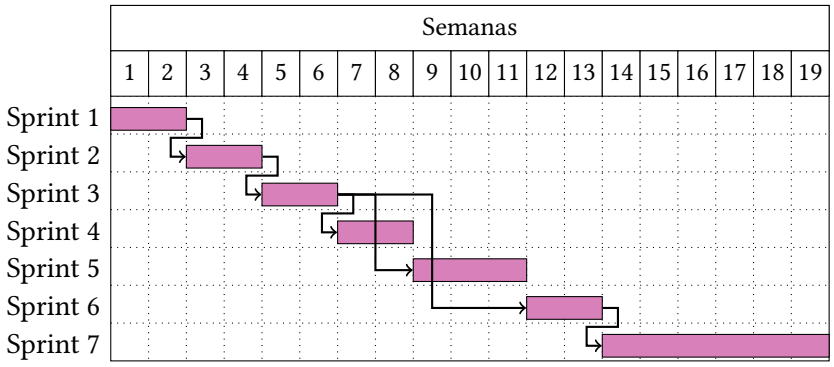

Figura 5.1: Cronología del proyecto DigReport

Capítulo 6

Análisis y requisitos

En este capítulo se abordará el análisis de requisitos que deberá satisfacer el producto final de acuerdo a los objetivos definidos en la planificación.

6.1 Actores

Tras un periodo de análisis de los distintos escenarios, se han identificado cuatro actores principales:

- Usuario no registrado
- Usuario registrado
- Profesional
- Autoridad

En este caso, la jerarquía de roles y permisos es horizontal. Es decir, el rol de Autoridad no garantiza el acceso a funcionalidades que sí tendrán disponibles los Profesionales, como es el caso de la validación de hallazgos. Durante una eventual puesta en producción, sería necesario definir un protocolo que determine a qué tipo de perfil dentro de la administración le compete administrar el sistema completo y adaptar el programa al respecto. Su arquitectura hexagonal y modular permitiría realizar esta actualización de forma sencilla y ágil.

6.2 Requisitos funcionales

Conjunto compuesto por todas las características que ofrecerá el sistema.

6.3 Requisitos no funcionales

No relativos a la funcionalidad concreta, si no a aquellos definen en qué condiciones es capaz el sistema de realizar dichas características.

- Escalabilidad: Ante una eventual puesta en producción y un aumento del número de usuarios utilizándola, será necesario que la aplicación pueda escalar correctamente sin necesidad de replicar la infraestructura.
- Robustez: Al tratarse de una aplicación destinada al uso público, es prioritario que el sistema tenga una tasa baja de fallos
- Disponibilidad: En la línea del punto anterior, una aplicación al servicio del ciudadano debe ofrecer el servicio de forma ininterrumpida.
- Usabilidad: El software, especialmente el público, debe ser sencillo de usar y ofrecer una interfaz amigable independientemente de la destreza del usuario, ya que su público objetivo es amplio y variado.
- Rendimiento: Debe realizar las operaciones con rapidez y agilidad, permitiendo operar con normalidad al usuario también en momentos de alto flujo de trabajo.

6.4 Historias de Usuario

Concepto	Descripción
Título	Registro en el sistema
Descripción	Como usuario no registrado, quiero registrarme en la aplicación para poder acceder a sus funcionalidades
Actor	Usuario no registrado
Acción	Como usuario no registrado, quiero proporcionar mis credenciales para poder registrarme en la aplicación
Objetivo	Poder acceder a las funcionalidades correspondientes a mi rol en el sistema

Cuadro 6.1: HU-01: Registrarse

Concepto	Descripción
Título	Inicio de sesión
Descripción	Como usuario registrado, quiero iniciar sesión en la aplicación para acceder a mis funcionalidades personalizadas
Actor	Usuario registrado
Acción	Como usuario registrado, quiero introducir mi correo electrónico y contraseña para autenticarme en el sistema
Objetivo	Acceder a mi cuenta personal y utilizar las funcionalidades según mi rol asignado

Cuadro 6.2: HU-02: Iniciar sesión

Concepto	Descripción
Título	Cierre de sesión
Descripción	Como usuario autenticado, quiero cerrar sesión en la aplicación para proteger mi cuenta y datos personales
Actor	Usuario autenticado
Acción	Como usuario autenticado, quiero poder cerrar mi sesión activa de forma segura
Objetivo	Finalizar mi sesión de trabajo y garantizar que nadie más pueda acceder a ella

Cuadro 6.3: HU-03: Cerrar sesión

Concepto	Descripción
Título	Edición de información de perfil de usuario
Descripción	Como usuario autenticado, quiero editar la información de mi perfil
Actor	Usuario autenticado
Acción	Como usuario autenticado, quiero modificar datos de mi perfil
Objetivo	Mantener mi información personal actualizada y personalizar mi perfil en la plataforma

Cuadro 6.4: HU-04: Editar información del perfil

Concepto	Descripción
Título	Registro de un nuevo hallazgo
Descripción	Como usuario autenticado, quiero registrar un hallazgo para contribuir al patrimonio cultural documentado
Actor	Usuario autenticado
Acción	Como usuario autenticado, quiero introducir información detallada del hallazgo (ubicación, descripción, fotografías, fecha) para crear un nuevo registro
Objetivo	Documentar hallazgos arqueológicos encontrados y ponerlos a disposición de profesionales para su validación

Cuadro 6.5: HU-05: Registrar hallazgo

Concepto	Descripción
Título	Generación automática de descripción mediante IA
Descripción	Como usuario autenticado, quiero que la IA genere automáticamente una descripción del hallazgo basándose en las imágenes proporcionadas
Actor	Usuario autenticado
Acción	Como usuario autenticado, quiero subir fotografías del hallazgo y obtener una descripción preliminar generada por inteligencia artificial
Objetivo	Facilitar el proceso de documentación del hallazgo mediante asistencia automatizada, ahorrando tiempo en la descripción inicial

Cuadro 6.6: HU-06: Generar descripción por IA de un hallazgo

Concepto	Descripción
Título	Validación o rechazo de hallazgos reportados
Descripción	Como profesional, quiero validar o rechazar hallazgos registrados para garantizar la calidad y veracidad de la información en la plataforma
Actor	Profesional
Acción	Como profesional, quiero revisar la información de un hallazgo pendiente y aprobar o rechazar su publicación con comentarios justificativos y asignándole una prioridad
Objetivo	Asegurar que solo hallazgos verificados y de interés histórico o arqueológico queden registrados en la plataforma

Cuadro 6.7: HU-07: Validar o rechazar hallazgo

Concepto	Descripción
Título	Consulta de hallazgos reportados por el usuario
Descripción	Como usuario autenticado, quiero consultar todos los hallazgos que he reportado para hacer seguimiento de su estado
Actor	Usuario autenticado
Acción	Como usuario autenticado, quiero acceder a un listado de todos mis hallazgos registrados con su estado actual (pendiente, validado, rechazado)
Objetivo	Realizar seguimiento de mis contribuciones a la plataforma y conocer el estado de validación de cada hallazgo

Cuadro 6.8: HU-08: Consultar mis hallazgos reportados

Concepto	Descripción
Título	Consulta de hallazgos validados por el profesional
Descripción	Como profesional, quiero consultar todos los hallazgos que he validado para revisar mi historial de validaciones
Actor	Profesional
Acción	Como profesional, quiero acceder a un listado de todos los hallazgos que he aprobado con los detalles de cada validación
Objetivo	Mantener un registro de mi trabajo de validación y revisar decisiones anteriores si es necesario

Cuadro 6.9: HU-09: Consultar mis hallazgos validados

Concepto	Descripción
Título	Consulta de validaciones pendientes de revisar
Descripción	Como profesional, quiero consultar todos los hallazgos pendientes de validación
Actor	Profesional
Acción	Como profesional, quiero acceder a un listado de hallazgos pendientes de validación ordenados por fecha de registro
Objetivo	Organizar mi trabajo y priorizar la revisión de hallazgos según criterios establecidos

Cuadro 6.10: HU-10: Consultar mis validaciones pendientes

Concepto	Descripción
Título	Consulta del panel de estadísticas generales
Descripción	Como autoridad, quiero consultar estadísticas globales de la plataforma para analizar el uso y actividad del sistema
Actor	Autoridad
Acción	Como autoridad, quiero acceder a un panel con métricas como número de hallazgos registrados, validados, usuarios activos y distribución geográfica
Objetivo	Obtener información estadística para tomar decisiones sobre la gestión del patrimonio arqueológico

Cuadro 6.11: HU-11: Consultar panel de estadísticas

Concepto	Descripción
Título	Añadir comentarios a hallazgos
Descripción	Como usuario autoridad o profesional, quiero añadir comentarios a los hallazgos para aportar información adicional o iniciar discusiones
Actor	Usuario autenticado, Profesional
Acción	Como usuario autoridad o profesional, quiero escribir y publicar comentarios en la ficha de un hallazgo validado
Objetivo	Fomentar la colaboración y el intercambio de conocimientos entre los actores sobre hallazgos arqueológicos

Cuadro 6.12: HU-12: Añadir comentario a un hallazgo

Concepto	Descripción
Título	Creación de nueva área protegida
Descripción	Como autoridad, quiero crear una nueva área protegida en el mapa para delimitar zonas de especial interés arqueológico o natural
Actor	Autoridad
Acción	Como autoridad, quiero definir los límites geográficos de un área protegida, asignarle un nombre, descripción y nivel de protección
Objetivo	Establecer zonas protegidas en el sistema para controlar el acceso y la actividad arqueológica en áreas sensibles

Cuadro 6.13: HU-13: Crear área protegida

Concepto	Descripción
Título	Edición de área protegida existente
Descripción	Como autoridad, quiero editar las propiedades de un área protegida para actualizar su información
Actor	Autoridad
Acción	Como autoridad, quiero modificar un área existente
Objetivo	Mantener actualizada la información de las áreas protegidas según cambios normativos o nuevos descubrimientos

Cuadro 6.14: HU-14: Editar área protegida

Concepto	Descripción
Título	Eliminación de área protegida
Descripción	Como autoridad, quiero eliminar un área protegida del sistema cuando ya no sea necesaria su delimitación
Actor	Autoridad
Acción	Como autoridad, quiero borrar de forma permanente un área protegida del mapa y la base de datos
Objetivo	Mantener actualizado el sistema eliminando áreas que han perdido su estatus de protección o fueron creadas por error

Cuadro 6.15: HU-15: Eliminar área protegida

Concepto	Descripción
Título	Creación de nuevo monumento
Descripción	Como autoridad, quiero registrar un nuevo monumento en el sistema para documentar el patrimonio histórico
Actor	Autoridad
Acción	Como autoridad, quiero introducir información del monumento
Objetivo	Catalogar monumentos históricos y hacerlos visibles en el mapa para su consulta y protección

Cuadro 6.16: HU-16: Crear monumento

Concepto	Descripción
Título	Edición de monumento existente
Descripción	Como autoridad, quiero editar la información de un monumento para mantener sus datos actualizados
Actor	Autoridad
Acción	Como autoridad, quiero modificar un monumento existente
Objetivo	Actualizar la información de monumentos

Cuadro 6.17: HU-17: Editar monumento

Concepto	Descripción
Título	Eliminación de monumento
Descripción	Como autoridad, quiero eliminar un monumento del sistema cuando sea necesario
Actor	Autoridad
Acción	Como autoridad, quiero borrar de forma permanente un monumento del catálogo y del mapa
Objetivo	Mantener el catálogo actualizado eliminando registros duplicados, erróneos o monumentos que han sido destruidos

Cuadro 6.18: HU-18: Eliminar monumento

Concepto	Descripción
Título	Consulta del ranking
Descripción	Como cualquier usuario de la plataforma, quiero disponer de información sobre la participación en la plataforma
Actor	Todos los roles
Acción	Como usuario de la plataforma, quiero disponer de un panel donde pueda ver los usuarios que más han contribuido exitosamente al registro de hallazgos
Objetivo	Fomentar la participación y el buen hacer entre los usuarios de la plataforma, recompensándolos con reconocimiento y honra pública.

Cuadro 6.19: HU-19: Consulta del ranking

Capítulo 7

Decisiones de diseño y arquitectura

La fase de diseño es el núcleo del desarrollo, y por ende se ha realizado un análisis exhaustivo de la arquitectura más conveniente, así como evaluar la posibilidad de aplicar los diversos patrones y buenas prácticas del desarrollo de software.

7.1 Arquitectura general: cliente-servidor

En esta arquitectura identificamos claramente dos componentes o subsistemas, el **servidor**, encargado de gestionar la persistencia, realiza las operaciones de negocio, controlar la seguridad y el acceso a los recursos y definir una API para comunicarse con la intefaz de usuario; y el subsistema **cliente**, que proporciona la citada interfaz y ordena y presenta visualmente los datos obtenidos en las diferentes peticiones a la API del servidor. Se ilustra su estructura en el siguiente diagrama:

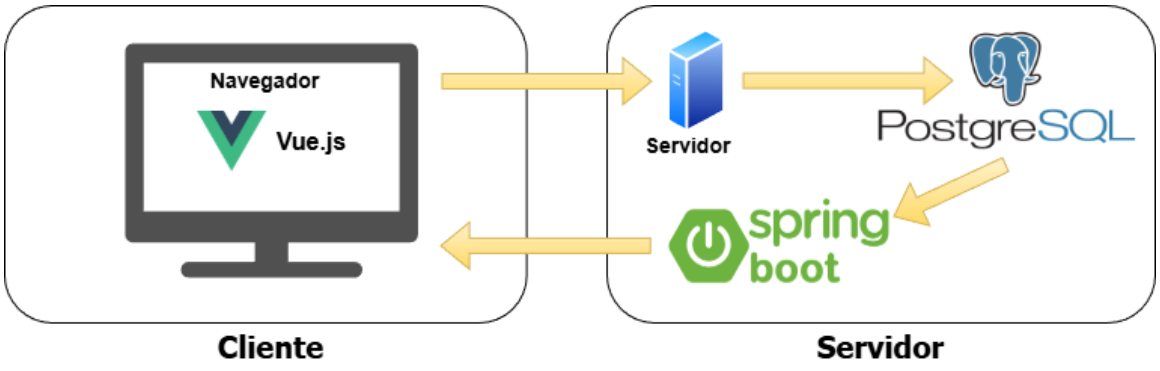

Figura 7.1: Esquema de la arquitectura general del proyecto

7.2 Modelo de datos

A continuación se muestra el diagrama de entidades persistentes y otros elementos de especial interés como los tipos enumerados siguiendo el estándar UML. Cada una de las entidades dispone de una serie de atributos y de un id incremental y autogenerado por el gestor de base de datos.

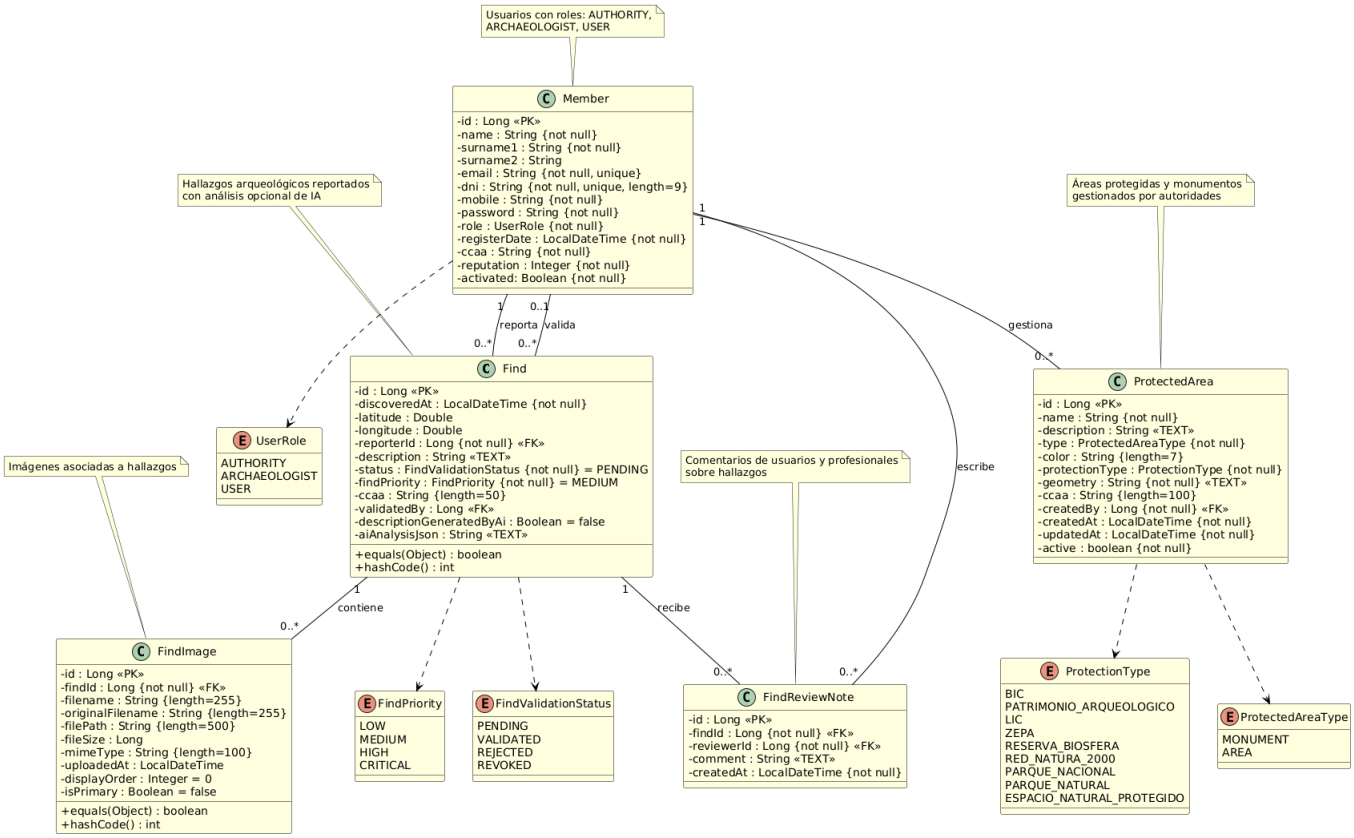

Figura 7.2: Diagrama UML de persistencia de DigReport

7.3 Arquitectura: back-end

La arquitectura elegida ha sido la hexagonal o de puertos y adaptadores. Esta decisión se basa, especialmente, en la necesidad de mantener la lógica de negocio completamente independiente de los detalles de implementación y las tecnologías empleadas. La arquitectura hexagonal permite que el dominio de la aplicación permanezca aislado de las capas externas. En este proyecto esto es crítico si queremos que sea un proyecto que puedan adoptar distintas administraciones públicas u organismos, cada uno con sus propias tecnologías, infraestructuras y capacidades.

En DIGREPORT, esta arquitectura se materializa en cuatro secciones diferenciadas: el dominio, la aplicación, la infraestructura y la web. El *dominio* contiene las entidades Plain Old Java Object (POJO) y la lógica de negocio pura, sin dependencias externas. La capa de *aplicación* define los puertos de entrada y salida mediante interfaces, implementando el contrato que deben cumplir las distintas características. Por su parte, la capa de *infraestructura* implementa los adaptadores que conectan el sistema con el mundo exterior, tales como repositorios y entidades JPA y servicios externos como la API de Gemini. Por último, la parte *web* contiene todas las implementaciones referentes a los endpoints y la comunicación con la interfaz de usuario, así como el manejo de excepciones a nivel de protocolo HTTP.

Esta separación proporciona múltiples ventajas en el proyecto. En primer lugar, permite realizar pruebas unitarias del dominio sin necesidad de levantar infraestructura compleja. En segundo lugar, facilita la incorporación de nuevas tecnologías, pudiéndose sustituir, por ejemplo, la base de datos relacional por una orientada a documentos sin modificar la lógica de negocio[1]. Además, la arquitectura hexagonal favorece la aplicación de principios SOLID[2], especialmente el de inversión de dependencias, al hacer que las capas externas dependan de abstracciones definidas en el núcleo, el de responsabilidad única y el de separación de interfaces, puesto que cada funcionalidad tendrá su propio contrato e implementación, lo que conlleva a construir un sistema con bajo acoplamiento favoreciendo el mantenimiento en el futuro y la escalabilidad.

7.3.1 Componente de aplicación: Patrones y principios

Se han aplicado patrones de diseño adecuados así como los principios de diseño para las buenas prácticas de desarrollo. No obstante, también se ha evaluado el impacto que tendría su no utilización para poder valorar si el costo de aplicarlos era mayor que la ganancia obtenida. Los marcos utilizados más destacados son los siguientes:

SRP: Single Responsibility Principle

Con toda seguridad el más visible y representativo de los principios. Casi inherente a la arquitectura hexagonal, en él cada componente tiene una única responsabilidad. Así, los controladores no tienen lógica de negocio y sólo gestionan peticiones HTTP, los servicios implementan casos de uso y los DTO son los encargados de transmitir datos entre capas.

[1] https://www.tactech.cl/blog/teach/una-pincelada-sobre-la-arquitectura-hexagonal
[2] https://openwebinars.net/blog/ventajas-de-la-arquitectura-hexagonal/

```java
@RestController  marcosvarela5
@RequestMapping("/api/profile")
public class MemberSelfController {

    private final MemberSelfService memberSelfService;  2 usages

    public MemberSelfController(MemberSelfService memberSelfService) { this.memberSelfService = memberSelfService; }

    @PutMapping  marcosvarela5
    public ResponseEntity<MemberDto> updateProfile(
            Authentication authentication,
            @Valid @RequestBody UpdateProfileRequest request
    ) {
        String currentEmail = authentication.getName();
        MemberDto updatedProfile = memberSelfService.updateProfile(currentEmail, request);
        return ResponseEntity.ok(updatedProfile);
    }
}
```

Figura 7.3: Ejemplo de aplicación del Principio de Responsabilidad Única

DIP: Dependency Inversion Principle

El principio de inversión de dependencias se basa en que todas las implementaciones de alto nivel deben depender de abstracciones y no de implementaciones concretas. Por ejemplo, *FindController* depende de *FindService* (el contrato o interfaz) y no de *FindServiceImpl* (la implementación). Además, Spring gestiona automáticamente la inyección de dependencias mediante constructores. Ambas cosas se observan en el siguiente ejemplo:

```java
17  @RestController  marcosvarela5
18  @RequestMapping("/api/finds")
19  public class FindController {
20
21      private final FindService findService;  12 usages
22
23      public FindController(FindService findService) { this.findService = findService; }
26
27      @PostMapping(consumes = MediaType.MULTIPART_FORM_DATA_VALUE)  marcosvarela5
28      @PreAuthorize("hasRole('USER')")
29      public ResponseEntity<FindDto> createFind(
30              Authentication authentication,
31              @RequestPart("data") @Valid CreateFindRequest request,
32              @RequestPart(value = "images", required = false) List<MultipartFile> images
33      ) {
34          String email = authentication.getName();
35
36          try {
37              FindDto created = findService.createFind(email, request, images);
38              return ResponseEntity.status(HttpStatus.CREATED).body(created);
39          } catch (IllegalArgumentException e) {
40              return ResponseEntity.badRequest().body(null);
41          }
42      }
43
```

Figura 7.4: Ejemplo de aplicación del Principio de Inversión de Dependencias

ISP: Interface Segregation Principle

El principio de segregación de interfaces señala que las implementaciones no deben depender de interfaces que no utilizan. Es mejor tener interfaces específicas para cada servicio que una general o dividirlas por secciones. Así, en el caso de este sistema, cada servicio tiene su propia interfaz definida. Este principio va también muy ligado a la arquitectura hexagonal.

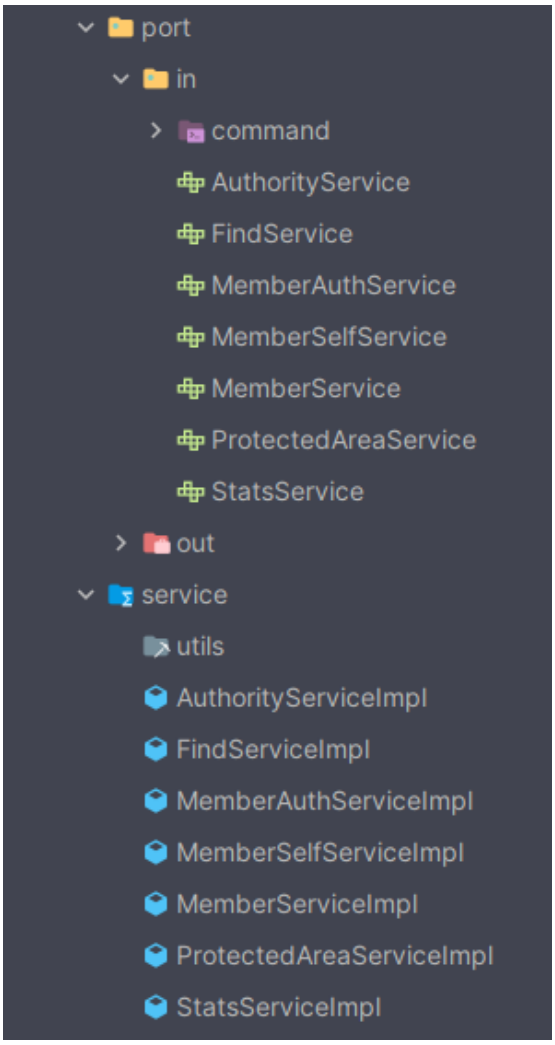

Figura 7.5: Ejemplo de aplicación del Principio de Segregación de Interfaces

A modo de demostración, se detalla a continuación la especificación de la interfaz principal del proyecto, que ilustra como se ha realizado la separación de responsabilidades.

```java
public interface FindService {  9 usages  1 implementation  marcosvarela5 *

    FindDto createFind(String reporterEmail, CreateFindRequest request, List<MultipartFile> images);  19 usages  1 implementation  marcosvarela5

    List<FindDto> getMyFinds(String reporterEmail);  2 usages  1 implementation  marcosvarela5

    Optional<FindDto> getFindById(Long id, String userEmail);  4 usages  1 implementation  marcosvarela5

    List<FindDto> getPendingFinds();  1 implementation  marcosvarela5

    FindDto validateFind(Long findId, String archaeologistEmail, ValidateFindRequest request);  5 usages  1 implementation  marcosvarela5

    List<FindDto> getAllFinds();  2 usages  1 implementation  marcosvarela5

    List<ReviewNoteDto> getReviewNotes(Long findId);  2 usages  1 implementation  marcosvarela5

    ReviewNoteDto addReviewNote(Long findId, String archaeologistEmail, AddReviewNoteRequest request);  4 usages  1 implementation  marcosvarela5

    int getPendingCount();  2 usages  1 implementation  marcosvarela5

    List<FindDto> getMyValidatedFinds(String archaeologistEmail);  2 usages  1 implementation  marcosvarela5

    String analyzeImagesWithAi(List<MultipartFile> multipartFiles);  3 usages  1 implementation  marcosvarela5

    List<FindImageDto> getFindImages(Long findId);  1 usage  1 implementation  new *

}
```

Figura 7.6: Contrato definido por la interfaz del servicio de hallazgos, FindService.

De igual modo, esta técnica se aplica también en los repositorios JPA, que manejan el acceso a datos. En el siguiente ejemplo se adjunta el puerto de salida del repositorio de Find, siguiendo además el patrón de arquitectura hexagonal, al ser independiente el contrato de la implementación y haciéndolo por tanto desacoplándolo de la tecnología utilizada:

```java
public interface FindRepositoryPort {  marcosvarela5

    Find save(Find find);  1 implementation  marcosvarela5

    Optional<Find> findById(Long id);  1 implementation  marcosvarela5

    List<Find> findAll();  1 implementation  marcosvarela5

    void deleteById(Long id);  1 implementation  marcosvarela5

    List<Find> findByReporterId(Long reporterId);  11 usages  1 implementation  marcosvarela5

    List<Find> findByStatus(FindValidationStatus status);  4 usages  1 implementation  marcosvarela5

    Boolean existsByReporterId(Long reporterId);  no usages  1 implementation  marcosvarela5

    List<Find> findByValidatedBy(Long validatedBy);  2 usages  1 implementation  marcosvarela5

    Long countByStatus(FindValidationStatus status);  28 usages  1 implementation  marcosvarela5

    Long count();  1 implementation  marcosvarela5

}
```

Figura 7.7: Contrato definido por la interfaz del puerto de hallazgos, FindService.

7.3.2 Componente Web: Controladores y endpoints

Este componente es el encargado de proporcionar el acceso a los servicios enrutando las peticiones HTTP del cliente, conformando así la API RESTful del sistema.

Cada una de estas peticiones llevará asociado uno de los siguientes tipos:

- GET para obtener los recursos
- POST para crear recursos
- PUT para actualizar recursos
- DELETE para eliminar recursos

Dichas peticiones estarán configuradas de manera que sean accesibles solo por los roles esperados utilizando la anotación *@PreAuthorize* proporcionada por Spring Boot. Del mismo modo, JWT aplicará una capa extra de seguridad mediante autenticación con token, tal y como se detalla en el capítulo 3.

A continuación se detallan todos los endpoints que componen el sistema, siguiendo el estándar OpenAPI[3]:

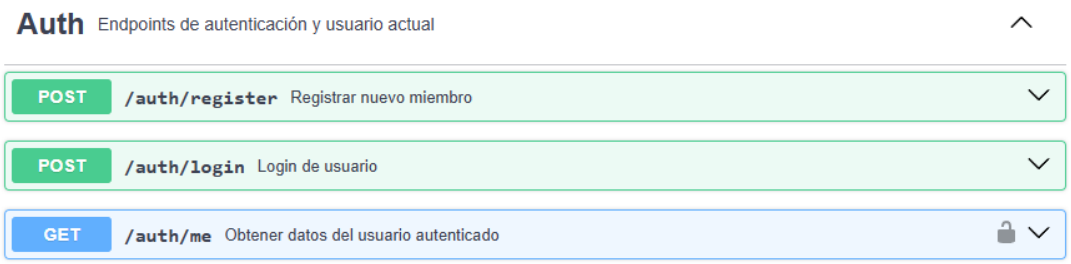

Figura 7.8: Endpoints relativos al servicio de autenticación

[3] https://www.openapis.org

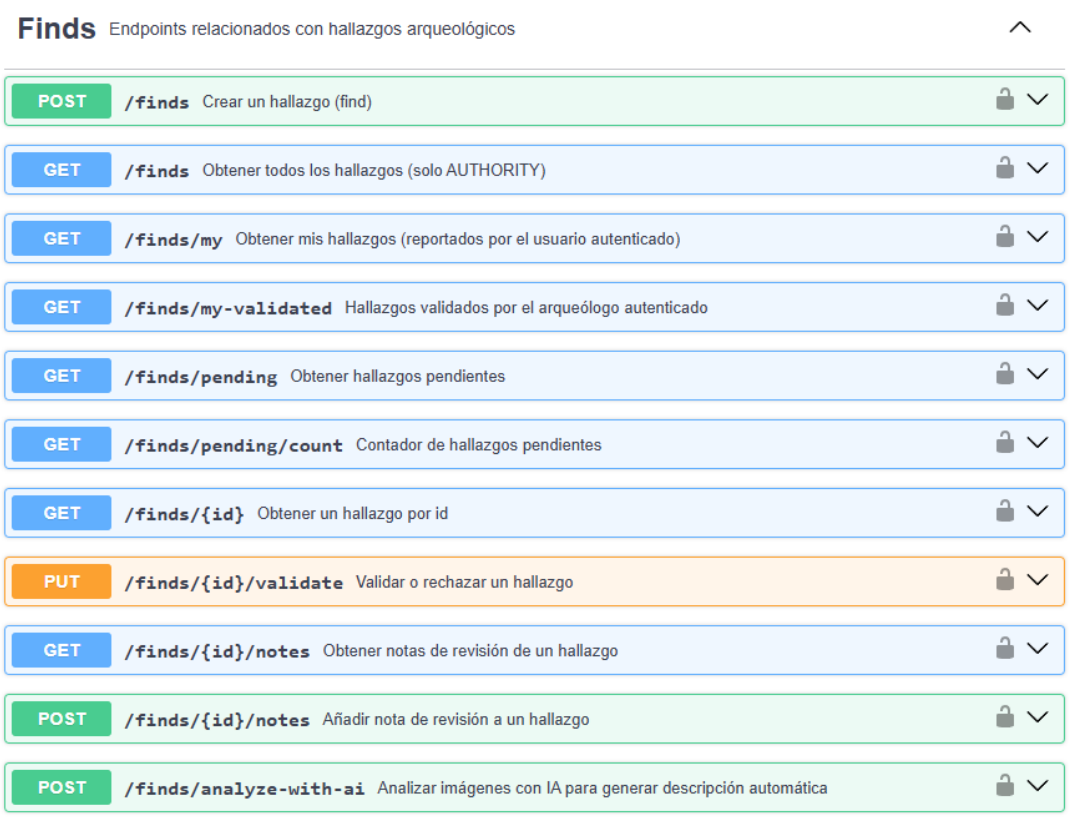

Figura 7.9: Endpoints relativos al servicio de hallazgos

Figura 7.10: Endpoints relativos al servicio de usuarios

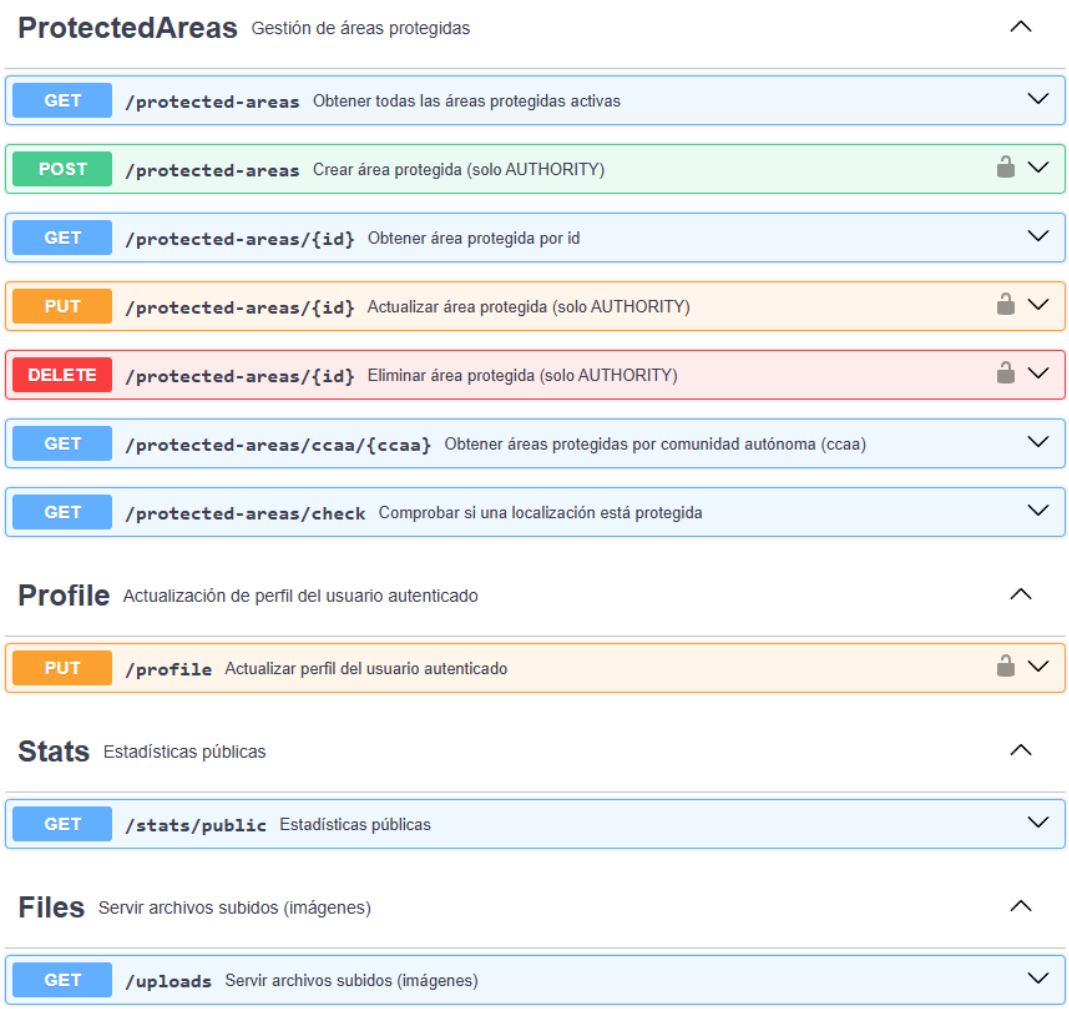

Figura 7.11: Endpoints relativos al servicio de áreas y monumentos, edición de perfil, estadísticas públicas y subida de imágenes

Manejo global de excepciones

El manejo de excepciones se realiza de forma centralizada, de acuerdo con el Principio de Responsabilidad Única, y se ha implementado utilizando la anotación *@ControllerAdvice* de Spring Boot. Todas las respuestas de error siguen una estructura JSON que incluye *timestamp* (marca temporal), código HTTP, descripción y mensaje del error. Esto tiene un impacto muy positivo en la experiencia de usuario y facillita la depuración y el manejo de errores en el lado front-end. Cabe remarcar la importancia de incluir un mensaje genérico que no permita a un potencial atacante que, por

ejemplo, esté tratando de iniciar sesión con credenciales ajenas, inferir qué dato está siendo incorrecto.

Asimismo, se implementa un método que gestionará las excepciones no contempladas en el resto, como se ve en el siguiente ejemplo:

```java
@ControllerAdvice  marcosvarela5
public class GlobalExceptionHandler extends ResponseEntityExceptionHandler {

    @ExceptionHandler(DuplicatedEmailException.class)  marcosvarela5
    public ResponseEntity<Object> handleDuplicatedEmail(DuplicatedEmailException ex) {
        return buildErrorResponse(HttpStatus.CONFLICT, ex.getMessage());
    }

    @ExceptionHandler(DuplicatedDniException.class)  marcosvarela5
    public ResponseEntity<Object> handleDuplicatedDni(DuplicatedDniException ex) {
        return buildErrorResponse(HttpStatus.CONFLICT, ex.getMessage());
    }

    @ExceptionHandler(ValidationException.class)  marcosvarela5
    public ResponseEntity<Object> handleValidation(ValidationException ex) {
        return buildErrorResponse(HttpStatus.BAD_REQUEST, ex.getMessage());
    }

    @ExceptionHandler(Exception.class) // no manejadas  marcosvarela5
    public ResponseEntity<Object> handleGeneric(Exception ex) {
        return buildErrorResponse(HttpStatus.INTERNAL_SERVER_ERROR,  message: "Unexpected error: " + ex.getMessage());
    }

    @ExceptionHandler({ BadCredentialsException.class, UsernameNotFoundException.class, AuthenticationException.class })
    public ResponseEntity<Map<String,String>> handleAuth(Exception ex) {
        return ResponseEntity.status(HttpStatus.UNAUTHORIZED)
                .body(Map.of( k1: "error",  v1: "Email or password incorrect"));
    }
```

Figura 7.12: Fragmento de implementación del manejo centralizado de excepciones

7.3.3 Componente de infraestructura

En este componente se encuentra toda la configuración relativa a la seguridad, como la protección de distintas rutas o la implementación de la generación del token y el filtro de JWT. En segundo lugar, contiene también la implementación de la persistencia: entidades JPA y mapeadores, así como la configuración de subida de imágenes.

```java
@Configuration  marcosvarela5
@EnableMethodSecurity
public class SecurityConfig {

    @Bean  marcosvarela5
    public SecurityFilterChain filterChain(HttpSecurity http, JwtAuthFilter jwtAuthFilter) throws Exception {
        http
                .cors( CorsConfigurer<HttpSecurity> cors -> cors.configurationSource(corsConfigurationSource()))
                .csrf(AbstractHttpConfigurer::disable)
                .sessionManagement( SessionManagementConfigurer<HttpSecurity> sm -> sm.sessionCreationPolicy(SessionCreationPolicy.STATELESS))
                .authorizeHttpRequests( AuthorizationManagerRequestMat... auth -> auth

                        .requestMatchers(HttpMethod.POST, "/api/auth/register").anonymous()
                        .requestMatchers("/api/auth/login").permitAll()
                        .requestMatchers("/api/auth/refresh").permitAll()
                        .requestMatchers("/api/protected-areas/**").permitAll()
                        .requestMatchers("/api/authorities/**").hasRole(UserRole.AUTHORITY.name())
                        .requestMatchers("api/members/ranking/public/**").permitAll()
                        .requestMatchers("/uploads/**").permitAll()

                        .anyRequest().authenticated()
                )
                .httpBasic(AbstractHttpConfigurer::disable)
                .formLogin(AbstractHttpConfigurer::disable)
                .exceptionHandling( ExceptionHandlingConfigurer<HttpSecurity> ex -> ex.authenticationEntryPoint(( HttpServletRequest req,  HttpServletRespon
                .addFilterBefore(jwtAuthFilter, UsernamePasswordAuthenticationFilter.class);

        return http.build();
    }
```

Figura 7.13: Fragmento de implementación SecurityConfig, para la protección de endpoints

Por último contiene también las implementaciones de los adaptadores de los distintos servicios de IA. Al definir un puerto de salida genérico, se podrá añadir muchos motores distintos sin preocuparnos por la implementación.

```java
public interface AiAnalysisPort {  7 usages  2 implementations  marcosvarela5

    /**
     * Analiza imágenes de un hallazgo arqueológico usando IA
     *
     * @param images Lista de imágenes a analizar
     * @return Resultado del análisis en formato JSON
     * @throws AiAnalysisException si hay error en el análisis
     */
    String analyzeImages(List<MultipartFile> images) throws AiAnalysisException;  1 usage  2 implementations  marcosvarela5

}
```

Figura 7.14: Puerto genérico para implementación de generación por IA.

A continuación se adjunta una visión general del componente:

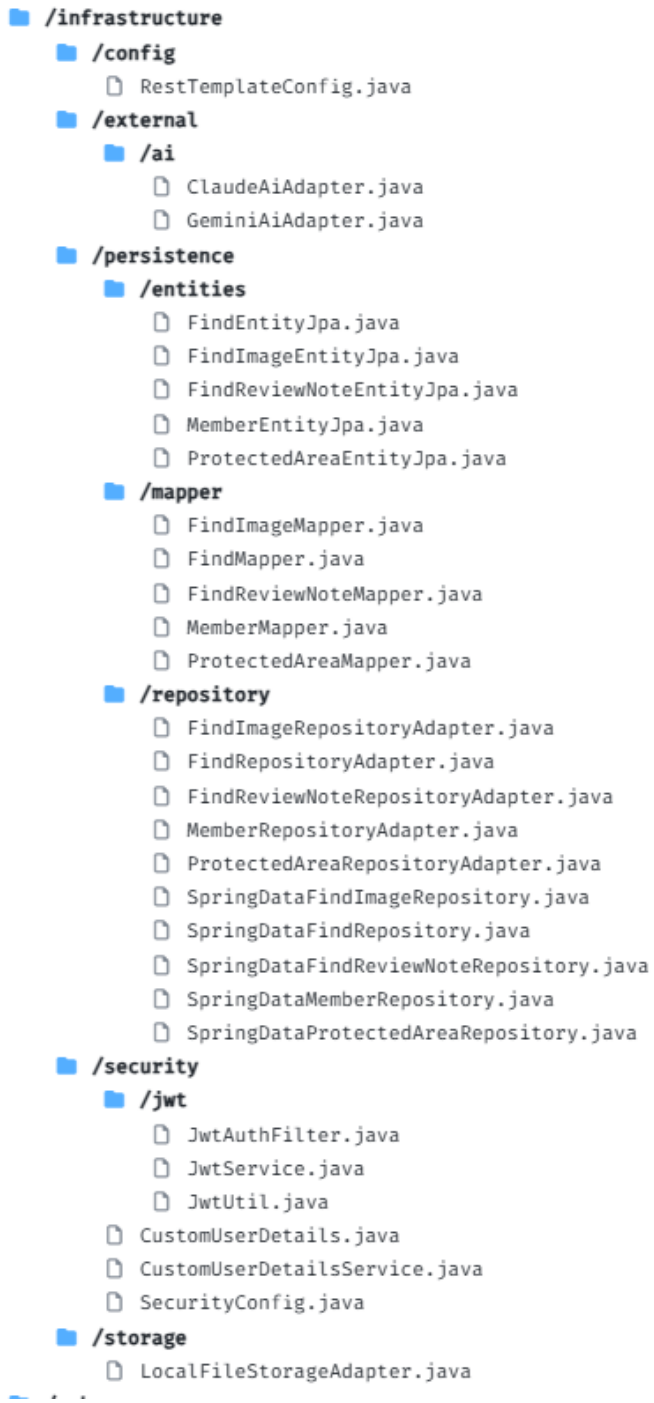

Figura 7.15: Estructura del componente de infraestructura

7.4 Arquitectura: interfaz de usuario

Para la realización de la parte front-end, se ha utilizado el sistema modular por componentes que ofrece Vue, asociando cada uno de ellos a su propia vista, lo que permite reutilizar componentes, cumplir con la separación de responsabilidades y disponer de la configuración bien diferenciada de la propia implementación.

En cuanto a la gestión del estado, se ha definido una *store* utilizando Pinia, que aporta un diseño modular que permite utilizar solo los métodos concretos necesarios para cada componente en vez del store completo, lo que disminuye notablemente la complejidad de la implementación. Un store es un contenedor centralizado de datos, que guarda y gestiona datos compartidos entre distintos componentes de Vue. Esto facilita la sincronización y elimina la duplicidad de datos y código.

Por último, para comunicación con el lado servidor o back-end se hace uso de la librería Axios, que simplifica sintáctica y organizativamente la gestión de llamadas HTTP contra la API. En este caso, tan solo definimos apiClient y después bastará con llamarlo utilizando *apiClient.get()* con la url del recurso.

```
import axios from 'axios'
import type { AxiosError, AxiosResponse } from 'axios'

// Configuración base de axios
export const apiClient : AxiosInstance = axios.create({ Show usages  marcosvarela5
    baseURL: 'http://localhost:8080',
    timeout: 60000,
    headers: {
        'Content-Type': 'application/json',
    },
})

// Interceptor JWT
apiClient.interceptors.request.use(  marcosvarela5
    onFulfilled: (config : InternalAxiosRequestConfig<any... ) => {
        const token : string | null  = localStorage.getItem( key: 'digreport-token')
        if (token) {
            config.headers.Authorization = `Bearer ${token}`
        }
        return config
    },
    onRejected: (error) => {
        return Promise.reject(error)
    }
)
```

Figura 7.16: Código referente a la implementación del cliente Axios

7.5 Uso de Inteligencia Artificial

Aunque forma parte de una de las funcionalidades del proyecto a modo de herramienta complementaria, y este trabajo no está basado en la Inteligencia Artificial ni la tiene como eje principal, el autor quiere dedicar un breve capítulo al uso de esta tecnología, dada su especial relevancia en estos últimos años. Incluyéndola en este proyecto ha pretendido ofrecer un ejemplo académico de cómo la IA puede impactar en proyectos de desarrollo software, y, más allá del uso generalista (y no siempre ético), intentar mejorar, a través de ella, la vida de su comunidad. En este caso, se ha decidido utilizar esta tecnología para generar descripciones precisas de un hallazgo reportado, aportando información clave acerca del mismo.

7.5.1 Configuración inicial: Google Gemini

El motor escogido ha sido Google Gemini, dado que ofrece una versión sin coste cuya potencia y límite de peticiones a la API resultaba más que suficiente para un proyecto a escala académica.

Ha sido necesario utilizar Google AI Studio para definir una API key, que nos permitirá comunicarnos con la API, y configurar de esta manera el fichero de propiedades del proyecto.

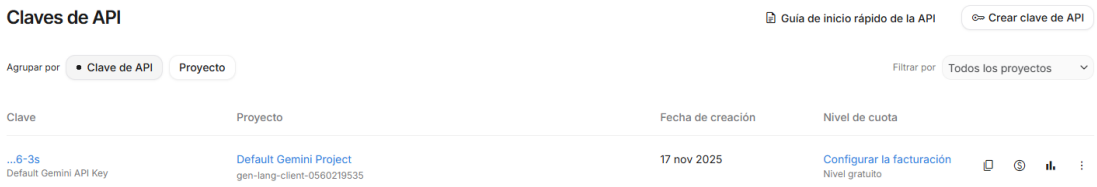

Figura 7.17: Configuración de la clave de Google Gemini

Una vez realizada esta configuración, será necesario implementar la lógica de peticiones y procesamiento de respuestas. Esta parte es idéntica a cualquier otro servicio ordinario y además, su modularidad permite añadir, si se dispone de planes y suscripciones a tal efecto, otros motores de IA de reconocido prestigio y potencia como es el caso de Claude[4] de Anthropic o GPT[5] de OpenAI.

[4] https://claude.ai/
[5] https://chatgpt.com

7.5.2 Definición del system prompt

No obstante, algo a destacar y sí exclusivo de la parte del servicio que afecta a la IA sería el *system prompt*. El *system prompt* es un texto plano que la IA consumirá al realizarse una petición y que le sirve a modo de guía para que "sepa" cómo tiene que inferir, cómo debería expresarse o requisitos que debe tener en cuenta. Es, en definitiva, como cualquier otra entrada de texto que recibiría de un usuario en cualquier servicio de chat, pero a nivel interno y que por tanto el motor siempre tendrá en cuenta. Estos textos pueden ser largos, de cientos de miles de palabras, aunque en el caso de este proyecto, siendo su alcance académico, se ha elaborado uno más genérico para que sirva de prueba, a la vez que lo suficientemente completo para que dichas pruebas sean ilustrativas del funcionamiento a escala real.

```
private String getArchaeologicalPrompt() {  1 usage   marcosvarela5
    return """
    Eres un arqueólogo experto en numismática y patrimonio histórico español.
    Analiza las imágenes proporcionadas de una moneda española aplicando las siguientes reglas heurísticas.

    REGLAS HEURÍSTICAS GENERALES:
    - Forma: circular → moneda; circular deteriorado → moneda antigua; irregular → fragmento.
    - Material: cobre/bronce (pátina verde), plata (gris plateado), oro/latón (dorado), hierro/plomo (oscuro)
    - Si el objeto identificado es una moneda → no usar "bronce puro", emplear cobre, vellón o latón.
    - Si el objeto identificado es otro hallazgo (herramienta, arma, medalla, botón, sello, vasija, estatua)
    - Bordes: perfectos → modernos (s.XIX+); irregulares → antiguos (pre-industrial).
    - Valor: moderno → texto arqueado con cifras arábigas; antiguo → números romanos o símbolos.
    - Año de acuñación: moderno → parte inferior; antiguo → posición variable.

    REGLAS POR ÉPOCA:

    1. SIGLO XV-XVI (Reyes Católicos, Austrias tempranos):
       - Escudos con yugo y flechas, símbolos heráldicos.
       - Leyendas en latín o castellano antiguo.
       - Monedas de vellón, cobre y plata.

    2. SIGLO XVII-XVIII (Austrias tardíos y Borbones iniciales):
       - Escudos complejos con cuarteles múltiples.
       - Monedas de plata con busto real.
       - Valor indicado en números romanos o abreviaturas.
```

Figura 7.18: Fragmento del prompt del proyecto

Con todo esto, se realizaron numerosas pruebas de llamadas y peticiones, y a medida que se detectaban errores, se procedía a refinar iterativamente el prompt, de forma que se fuese haciendo más preciso. El resultado final alcanzado es plenamente funcional y satisfactorio, aunque en el caso de hallazgos más deteriorados o menos

comunes es todavía sensible a imprecisiones. A continuación se muestra un ejemplo de una imagen procesada para analizar:

Figura 7.19: Imagen insertada para procesar. En ella, una moneda española de 1870.

La respuesta devuelta por la IA, en formato JSON, fue la siguiente:

```
{
    "tipo_probable": "moneda",
    "material_estimado": {
        "moneda": "cobre",
        "otros_objetos": "otro"
    },
    "periodo_estimado": "siglo XIX",
    "confianza": 0.95,
    "caracteristicas_clave": [
        "Forma circular y bordes definidos con dentículos, indicando acuñación industrial",
        "Figura de león rampante sosteniendo un escudo cuartelado de España",
        "Leyenda superior 'CIEN PIEZAS EN KILOG.'",
        "Denominación inferior 'DIEZ CENTIMOS' en texto arqueado",
        "Pátina de color rojizo-marrón característica del
```

```
    cobre o sus aleaciones"
15      ],
16      "descripcion": "Moneda española de cobre con un león
    rampante sujetando el escudo de España. Las leyendas
    'CIEN PIEZAS EN KILOG.' y 'DIEZ CENTIMOS' son distintivas
    de las emisiones del Gobierno Provisional o inicios de la
    Primera República en el último tercio del siglo XIX.",
17      "advertencias": "La imagen solo muestra una cara de la
    moneda, por lo que el año de acuñación no es visible. La
    composición exacta podría ser una aleación de cobre como
    bronce o latón, aunque visualmente se asimila al cobre de
    la época.",
18      "detalles_moneda": {
19          "resellada": false,
20          "anio_acunacion": "indeterminado",
21          "valor": "texto arqueado",
22          "fecha_legible": false
23      },
24      "forma_borde": "circular_perfecto"
25  }
```

Listing 7.1: Ejemplo de respuesta JSON del análisis de IA con Google Gemini

A través de la interfaz de usuario, la información se mostraría de una manera redactada, teniendo la opción también de consultar un informe estructurado que aporta más información. En la siguiente captura de pantalla, correspondiente a un intento posterior al del JSON insertado anteriormente, es interesante reparar en la imprecisión que comete, como se señalaba anteriormente en este mismo capítulo: Tratándose de céntimos de peseta, la IA los califica como céntimos de escudo. En esta época, recién implantada la peseta, era común que circularan otras muchas monedas y durante años coexistieron dos sistemas monetarios diferentes, lo que podría explicar estos errores.

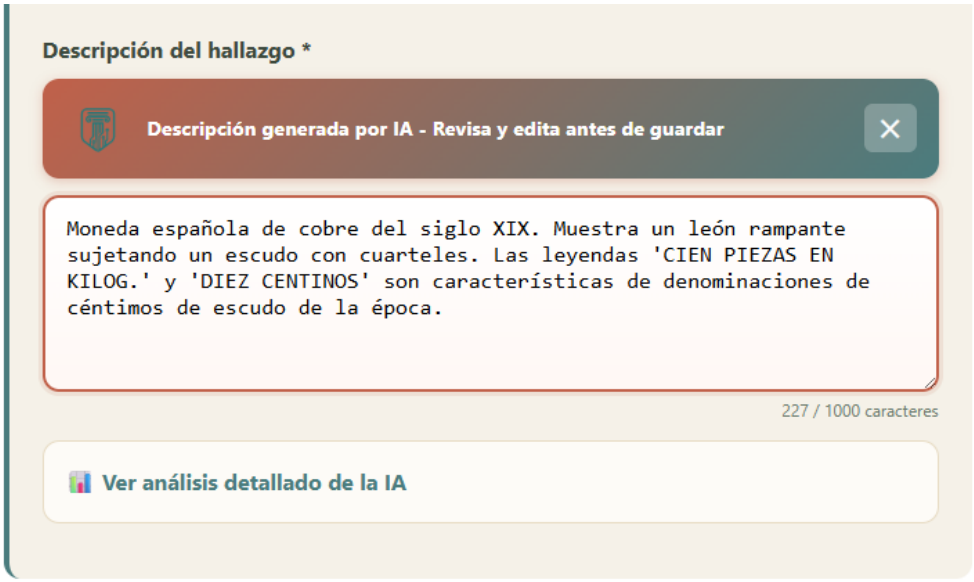

Figura 7.20: Descripción generada por IA en la interfaz de usuario

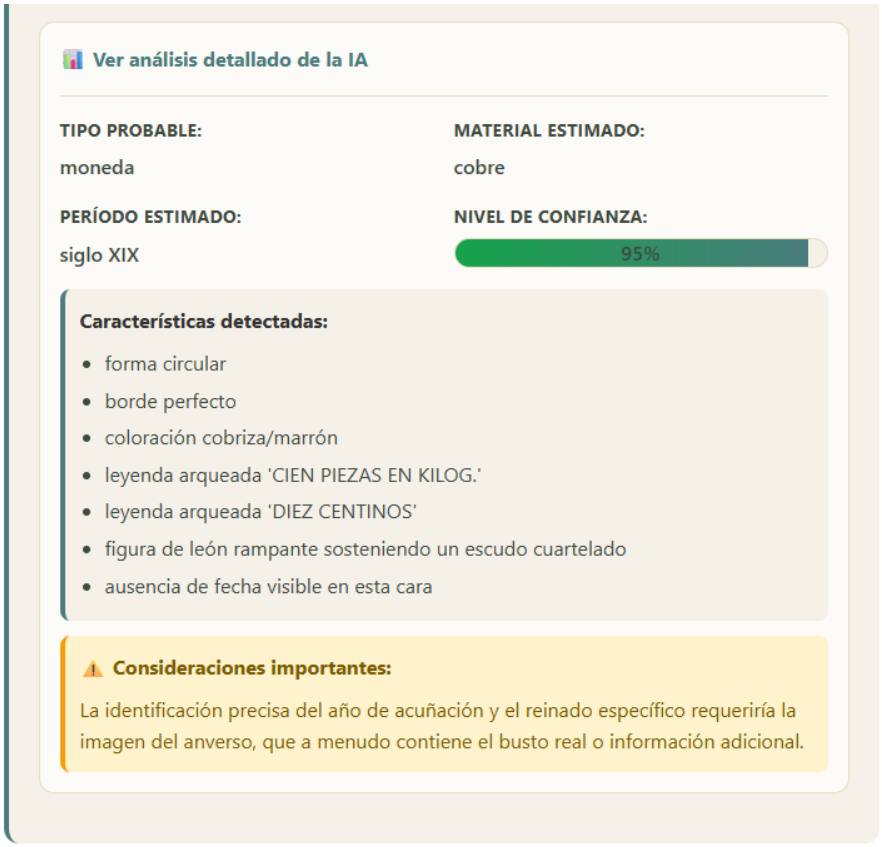

Figura 7.21: Informe detallado generado por IA en la interfaz de usuario.

Capítulo 8

Pruebas automatizadas

Las pruebas forman el principal método de control de la calidad durante el desarrollo del proyecto, lo que las convierte en una parte fundamental y crítica del mismo. Tienen los objetivos de verificar y validar que el software cumple los requisitos establecidos, pero también de identificar los defectos. Así pues, se dice que *una prueba exitosa es una prueba que falla.*

Utilizando las herramientas descritas en el tercer capítulo, se han realizado pruebas tanto de los servicios, que verifiquen el correcto funcionamiento de la lógica de negocio, como de los controladores, para probar que la comunicación con la interfaz de usuario se realiza de la manera esperada.

8.1 Pruebas unitarias

La implementación de pruebas unitarias ha sido ágil y estructurada gracias a Mockito, cuya principal ventaja es capaz de maquetar (*mock*) entidades y otros elementos, con el fin de tener que evitar tener que insertar registros en base de datos o simular manualmente escenarios de prueba.

De esta forma, se han realizado pruebas unitarias que prueban cada uno de los métodos de cada uno de los servicios, alcanzando resultados satisfactorios, como se ve en la siguiente imagen cuya información genera el propio IDE.

A continuación se muestra en la figura la cobertura alcanzada, desglosada por clase, método, línea y rama condicional.

Element	Class, %	Method, %	Line, %	Branch, %
es.marcos.digreport.application.service	100% (9/9)	100% (53/53)	99% (410/411)	95% (114/120)
AuthorityServiceImpl	100% (1/1)	100% (7/7)	100% (48/48)	100% (12/12)
FindServiceImpl	100% (2/2)	100% (19/19)	99% (159/160)	86% (39/45)
MemberAuthServiceImpl	100% (1/1)	100% (4/4)	100% (56/56)	100% (18/18)
MemberSelfServiceImpl	100% (1/1)	100% (3/3)	100% (34/34)	100% (28/28)
MemberServiceImpl	100% (1/1)	100% (7/7)	100% (35/35)	100% (6/6)
ProtectedAreaServiceImpl	100% (2/2)	100% (11/11)	100% (66/66)	100% (9/9)
StatsServiceImpl	100% (1/1)	100% (2/2)	100% (12/12)	100% (2/2)

Figura 8.1: Cobertura alcanzada en las pruebas unitarias

8.2 Pruebas de integración

Para realizar las pruebas de integración se ha seguido la lógica de las pruebas unitarias pero se ha prescindido de Mockito. Se ha definido un fichero de propiedades para crear un perfil de testing, ya que mientras que para las pruebas unitarias no es necesario levantar Spring Boot al maquetar los datos, sí lo será para las pruebas de integración. Se ha utilizado, en esta configuración, una base de datos H2, un tipo de base de datos que se ejecuta en memoria y que por lo tanto es más rápida, además de mantener limpia la base de datos principal del sistema.

```
spring.application.name=digreport-test

# ==============================
# H2 en memoria para tests
# ==============================
spring.datasource.url=jdbc:h2:mem:testdb;MODE=PostgreSQL;DB_CLOSE_DELAY=-1
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=

# ==============================
# JPA / Hibernate
# ==============================
spring.jpa.hibernate.ddl-auto=create-drop
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

# Google Gemini API (no se usa en tests, pero necesita estar definida)
google.gemini.api.key=test-key

# Configuración de archivos para tests (directorio temporal)
file.upload.dir=target/test-uploads
file.base.url=http://localhost:8080/test-uploads

# Configuración de archivos
spring.servlet.multipart.enabled=true
spring.servlet.multipart.max-file-size=10MB
spring.servlet.multipart.max-request-size=50MB
```

Figura 8.2: Fichero de propiedades del perfil de testing con la creación de la base de datos H2

El objetivo de las pruebas de integración es verificar el funcionamiento entre componentes y capas, por lo que es preciso que distintos elementos, como servicios, repositorios, bases de datos o puertos interactúen entre sí sin maquetar nada. Los resultados alcanzados fueron, igual que en las anteriores, también altamente satisfactorios. A continuación se adjunta una imagen de la configuración inicial de un test de integración para el servicio de hallazgos y los resultados de las distintas pruebas, observando la configuración inicial así como los diferentes tests definidos, correspondientes a diferentes escenarios que se podrán dar dentro del caso de uso de registrar hallazo.

```
@SpringBootTest  new *
@ActiveProfiles("test")
@Transactional
@DisplayName("FindService - Test de Integración: Crear Hallazgo")
class FindServiceIntegrationTest {

    @Autowired
    private FindService findService;

    @Autowired
    private FindRepositoryPort findRepository;

    @Autowired
    private MemberRepositoryPort memberRepository;

    private Member testCitizen;  8 usages
    private CreateFindRequest validRequest;  6 usages

```

Figura 8.3: Configuración inicial de un test de integración

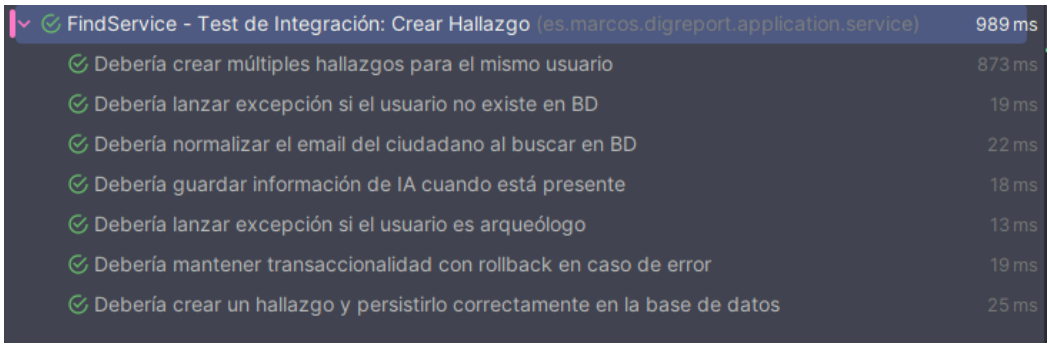

Figura 8.4: Ejemplo de pruebas de integración exitosas

8.3 Pruebas de la API

Para llevar a cabo las pruebas de la API se ha utilizado la herramienta Postman, que ha permitido simular peticiones HTTP reales tal y como se harían por un usuario real, y comprobar sus respuestas. Su entorno dinámico e intuitivo fueron clave a la hora de convertirla en una herramienta imprescindible para las pruebas.

Se construyó una colección de pruebas asociadas al proyecto, para poder tener la estructura completa y poder realizar simulaciones variadas de una forma más ágil y

eficiente.

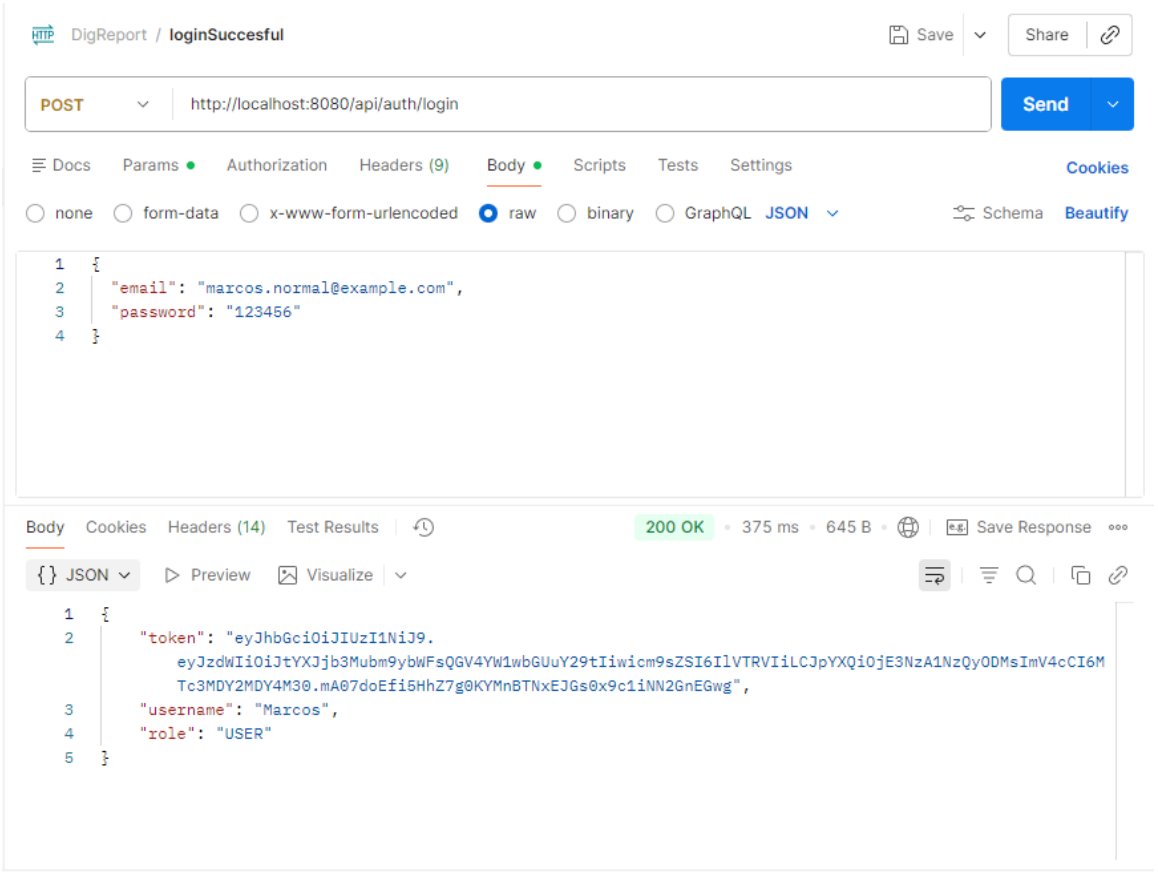

Figura 8.5: Prueba de un inicio de sesión exitoso utilizando Postman. En la parte superior se puede ver los datos enviados, y en la inferior, los recibidos.

Capítulo 9

Conclusiones

En este trabajo se desarrolló una plataforma web para la gestión conjunta del patrimonio histórico con participación de toda la sociedad civil.

Conclusiones generales

- El proyecto se ha completado satisfactoriamente cumpliendo los requisitos planteados en la fase de análisis. La plataforma permite a ciudadanos participar directamente registrar hallazgos con geolocalización, facilita a profesionales la validación de estos descubrimientos y proporciona a las autoridades capacidad de supervisión, así como herramientas analíticas para la toma de decisiones sobre protección patrimonial.
- El proyecto permitió al alumno poner en práctica los conocimientos adquiridos durante el grado, como es el caso de los patrones de diseño, las arquitecturas de software, la toma de decisiones o los principios de código limpio. De igual forma, permitió al alumno adquirir conocimientos nuevos en campos desconocidos como por ejemplo la integración de APIs externas de IA o el uso de Mockito para el testing.
- El alumno utilizó por primera vez la arquitectura hexagonal, cuya aplicación fue de gran complejidad en los compases iniciales del desarrollo, acabando por resultar mucho más cómoda para las fases posteriores. El alumno lo considera uno de los aprendizajes fundamentales adquiridos durante la realización de este trabajo.
- La utilización por primera vez del framework Vue.js permitió al alumno descubrir y aprender un *framework* con un enorme potencial, versátil y ligero. El autor lo considera una de sus referencias principales en el desarrollo front-end.

- El proyecto ha sido un reto personal para el alumno, pero su desarrollo también ha representado una actividad de gran interés y disfrute para él, provocando así una comprensión mucho más amplia de lo que es el desarrollo software al tener que realizar una implementación desde cero.
- El autor tiene intención de seguir desarrollando y perfeccionando este proyecto.

9.1 Futuras líneas de trabajo

Funcionalidades avanzadas

- Sistema de notificaciones para informar al usuario de la actividad relacionada con él en la plataforma.
- Exportar archivos, reportes e informes a distintos formatos
- Gamificación: Línea temporal interactiva con todos los hallazgos, superposición de capas en el mapa, mapas de calor…

Rendimiento y escalabilidad

- Servicio de almacenamiento en la nube para las imágenes
- Optimización de consultas complejas a la base de datos

Inteligencia Artificial

- Reentrenamiento del modelo de IA para conseguir un modelo de visión específico adaptado a patrimonio histórico español (*fine-tuning*)
- Traducción automática según la cultura del navegador

Dominio

- Integración con el sector público: Registro y sesión mediante el sistema Cl@ve.
- Integración de datos existentes en bases de datos de las instituciones públicas de Cultura y Patrimonio
- Estandarización de datos, formatos y procesos a los requerimientos de los organismos públicos

Experiencia de usuario

- Aplicación móvil nativa
- Alternancia entre interfaz clara y oscura
- Internacionalización: Traducción manual a todos los idiomas regionales del Estado.

Apéndices

Apéndice A

Manual de usuario

Como apéndice a este trabajo, se adjunta un manual de usuario consistente en imágenes de ejemplo de las principales pantallas del sistema junto con una breve explicación de las mismas.

A.1 Autenticación

Inicio de sesión

Inicio de sesión básico, consistente en un formulario con campos para correo electrónico y contraseña.

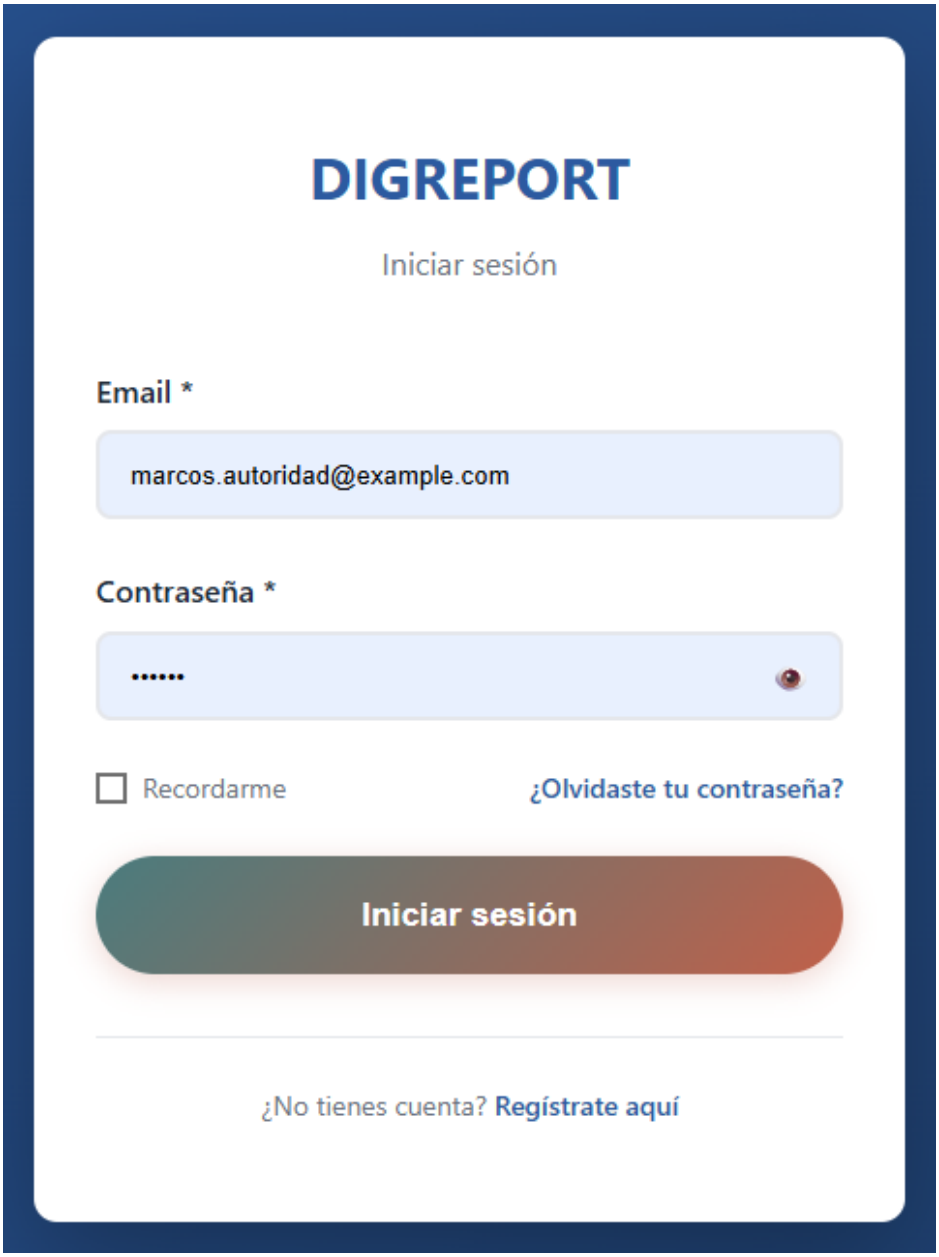

Figura A.1: Pantalla de inicio de sesión.

Registro de nuevos usuarios

Formulario de alta en el sistema, consistente en un formulario divido en dos secciones, una de información personal y otra de información de cuenta.

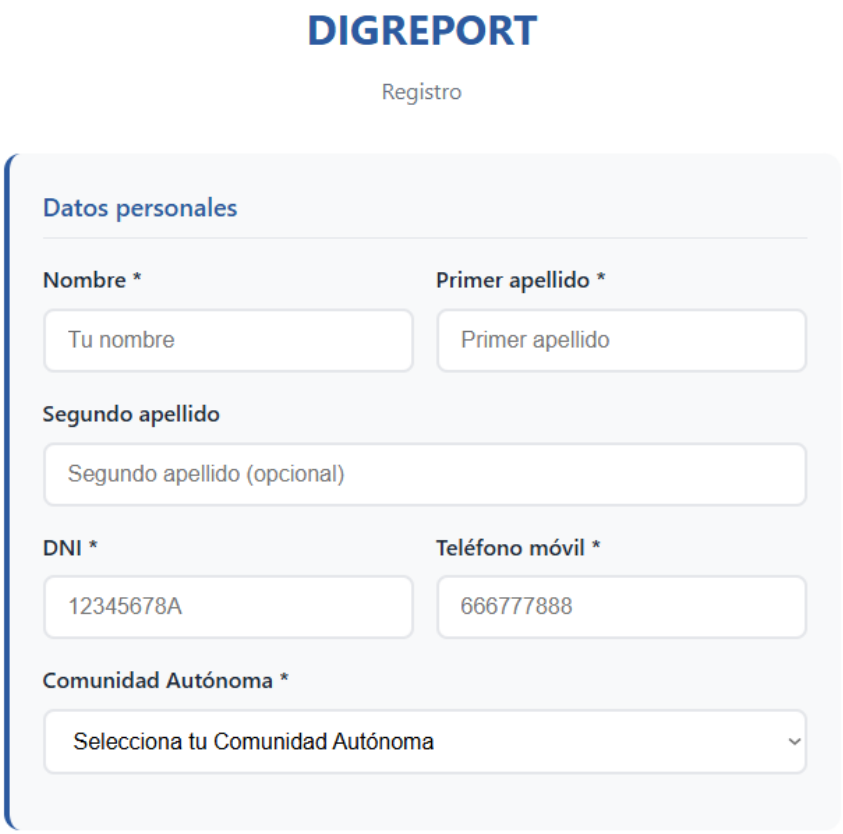

Figura A.2: Pantalla de registro, información personal.

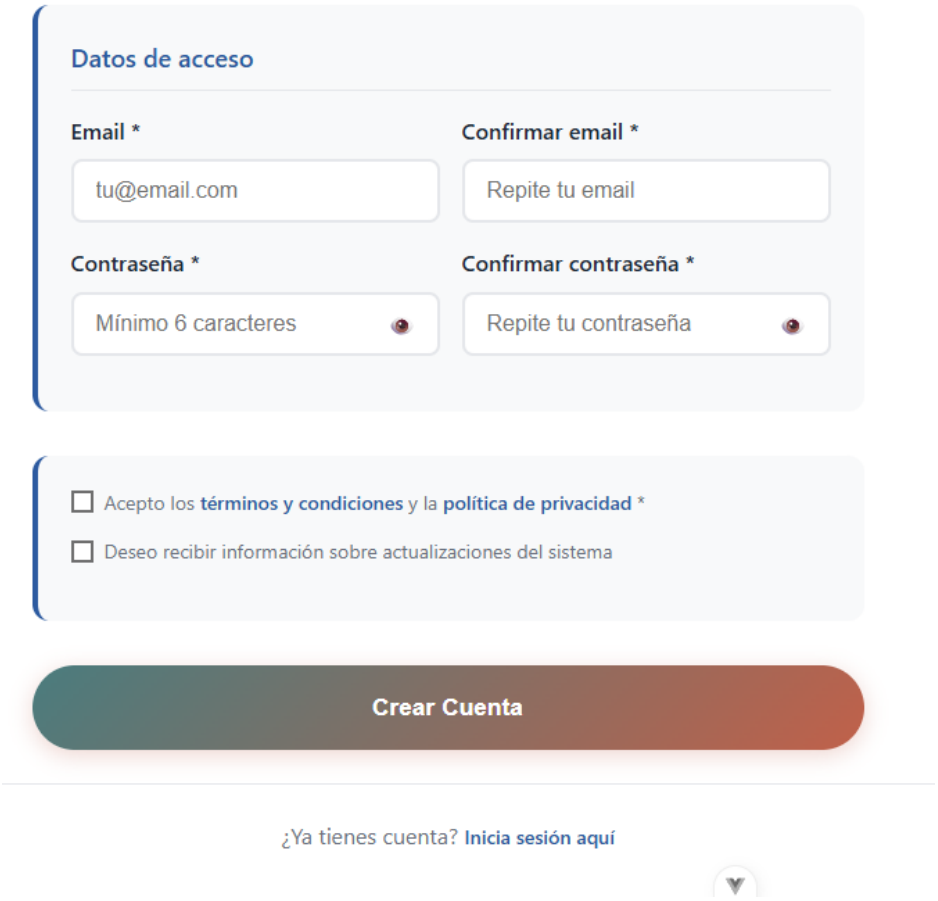

Figura A.3: Pantalla de registro, información de cuenta.

A.2 Pantalla principal genérica

Consiste en la pantalla principal de la aplicación, a donde será redirigido el usuario una vez se registra o inicia sesión. Es igual para todos los usuarios, salvo para las autoridades, que a mayores contarán con un menú de acceso a las zonas protegidas.

Figura A.4: Pantalla principal I, con el ranking de usuarios en la parte izquierda.

Figura A.5: Pantalla principal II - Actores participantes.

Figura A.6: Pantalla principal III - Breve explicación del flujo.

Figura A.7: Pantalla principal IV - Características ofrecidas.

A.2.1 Menú de perfil

Este menú desplegable alberga las opciones de perfil para cada usuario. Las opciones disponibles en él serán diferentes según el rol ostentado, marcándolo con un estilo visual claro y comunicativo.

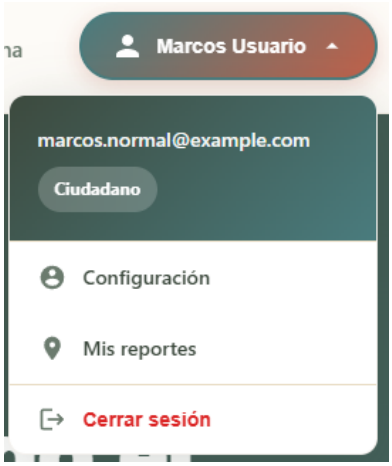

Figura A.8: Menú de perfil para el rol usuario.

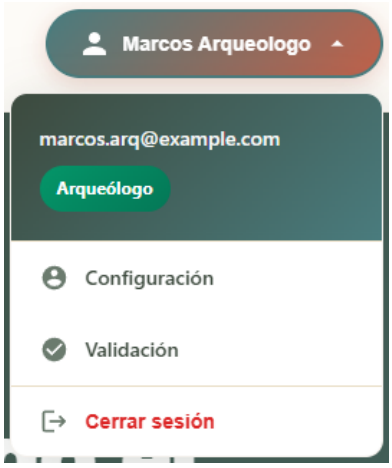

Figura A.9: Menú de perfil para el rol profesional.

Figura A.10: Menú de perfil para el rol autoridad.

A.2.2 Pie de página

El footer o pie de página contiene referencias a las diferentes secciones de la página principal, así como al marco legal vigente en el dominio de la aplicación. La redirección se hace directamente a la ley de patrimonio histórico español publicada en la página web del BOE.

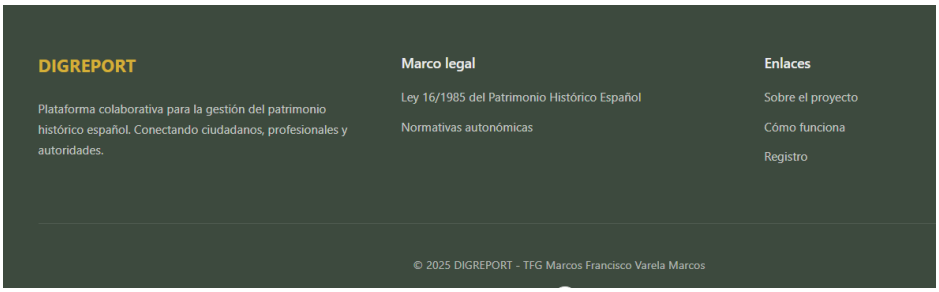

Figura A.11: Pie de página.

A.3 Edición de perfil

Consiste de un formulario con los campos editables del perfil y un botón para confirmar y guardar los mismos.

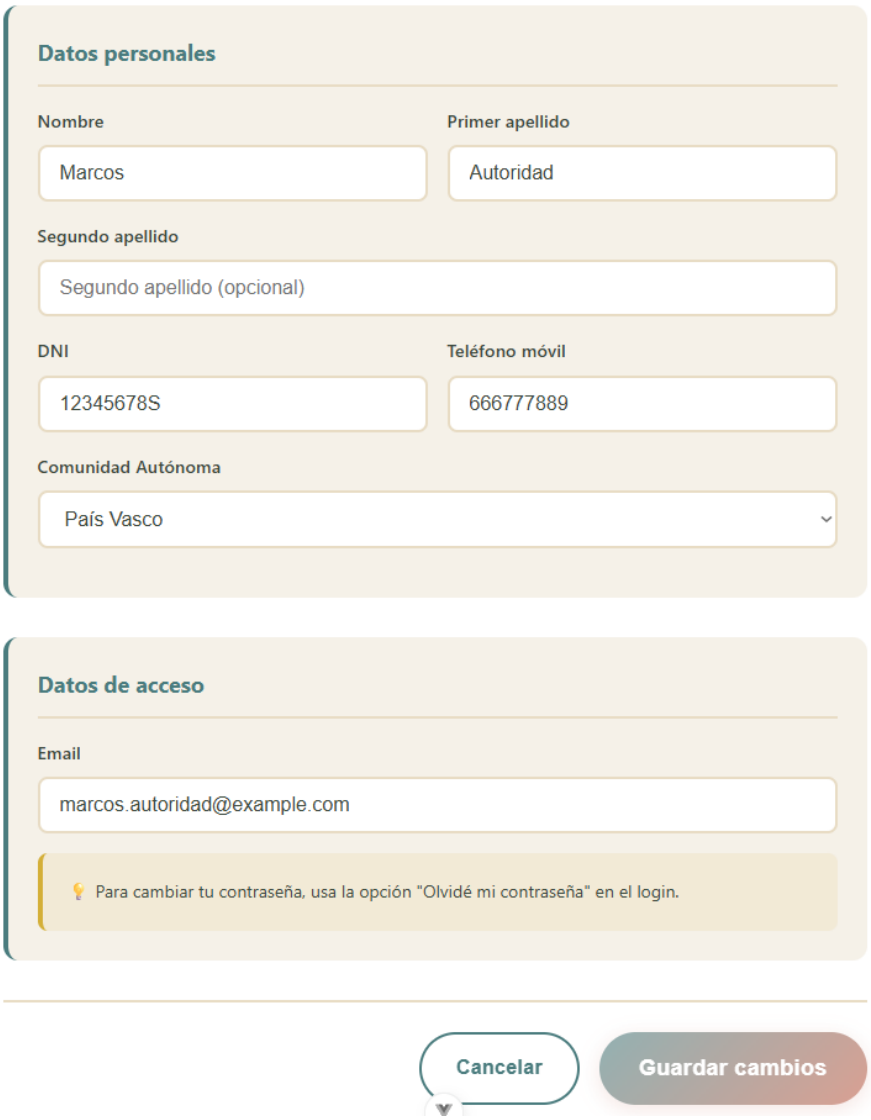

Figura A.12: Formulario de edición de perfil.

A.4 Manejo de hallazgos

A.4.1 Reporte de hallazgo

Consiste en un formulario en el cual se pueden insertar imágenes, bien para arrojar información extra o para ser procesadas por IA y generar una descripción. Además se debe introducir información geográfica lo más precisa posible, pudiendo utilizar la ubicación actual del equipo en el que se utilice la aplicación.

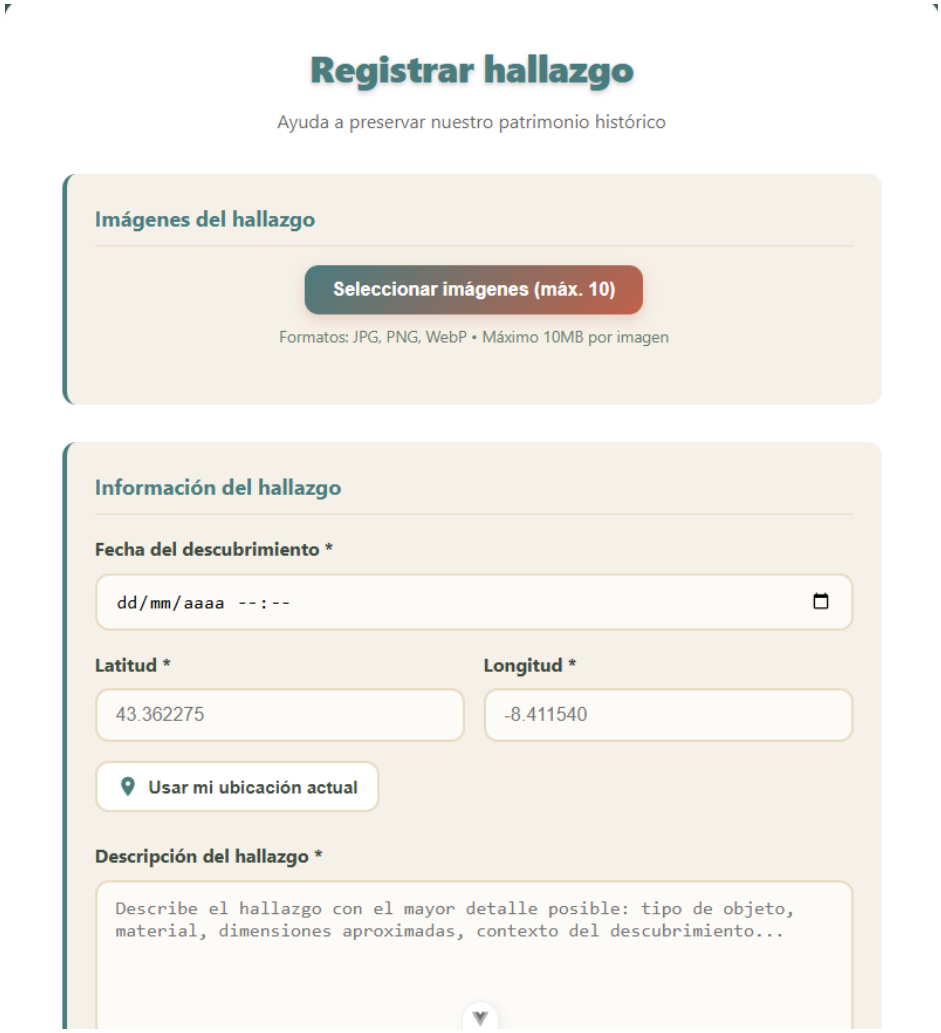

Figura A.13: Primera sección del formulario de reporte de hallazgo

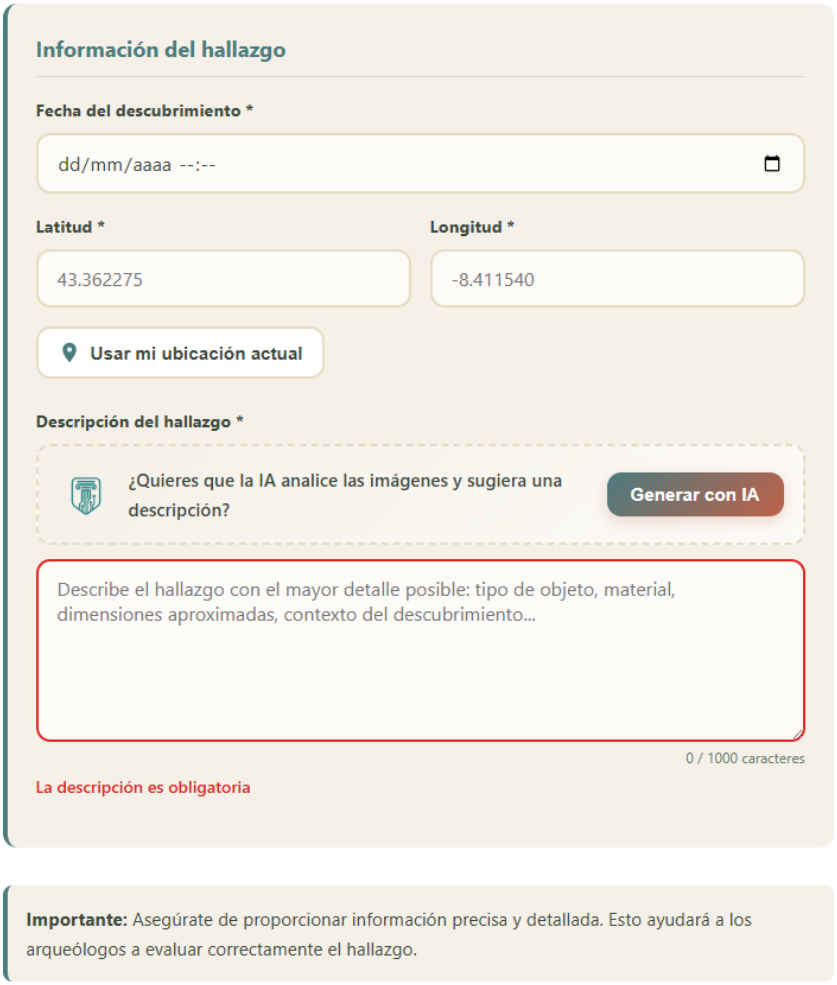

Figura A.14: Segunda sección del formulario de reporte de hallazgo

Además, al generar la descripción con IA también tendremos acceso a un informe detallado y esquematizado:

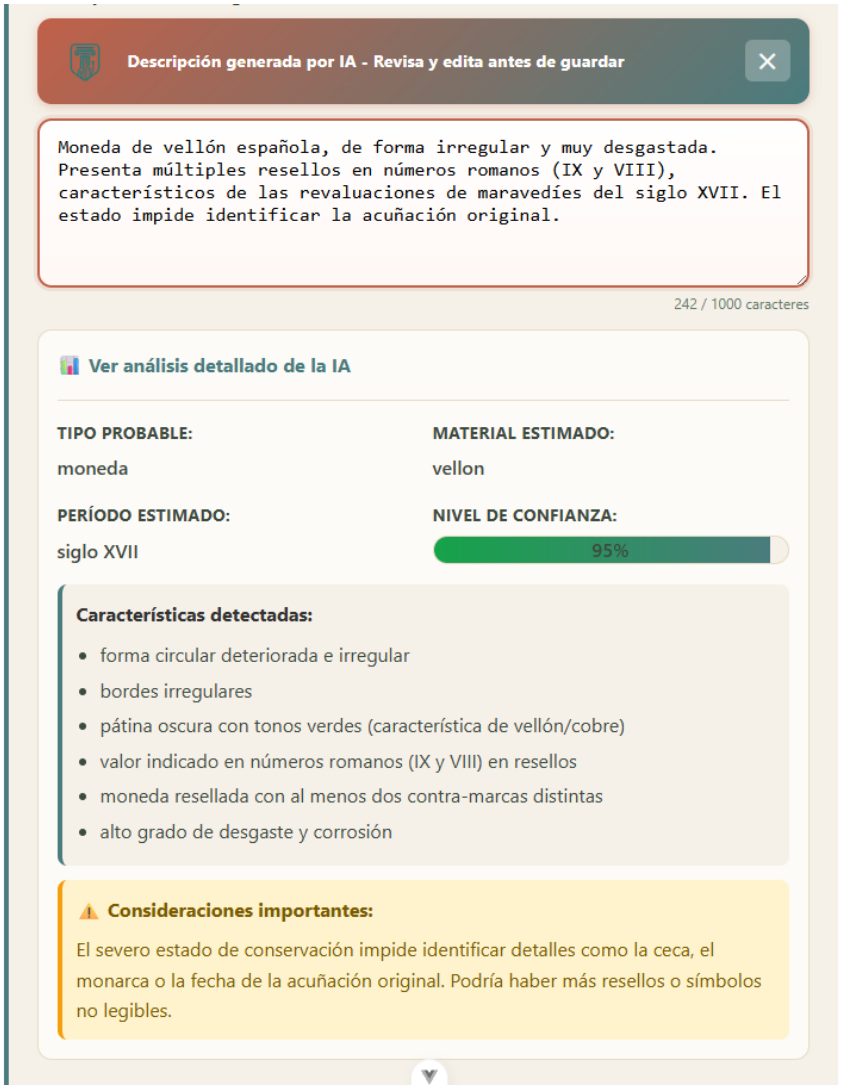

Figura A.15: Descripción y análisis generados con IA

A.4.2 Resumen de mis reportes

Esta sección, compartida por usuarios ciudadanos y profesionales, permite ver los hallazgos de forma resumida.

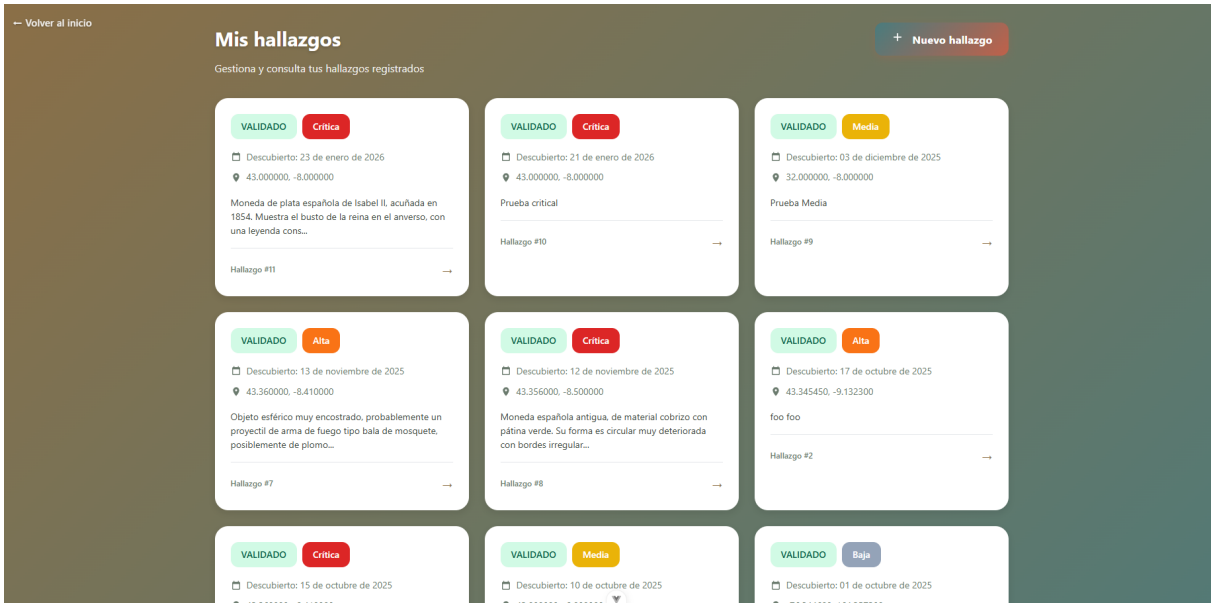

Figura A.16: Resumen de mis hallazgos reportados (rol ciudadano) o validados (rol profesional)

En el caso de los usuarios con rol profesional, podrán ver en similar sección los reportes pendientes de validación o que ellos mismos ya han validado, pudiendo filtrar entre ambas opciones:

Figura A.17: Resumen de mis hallazgos validados

A.4.3 Detalles de hallazgo

En esta pantalla se muestran con todo detalle cada uno de los hallazgos. En el caso del rol profesional, se dispondrá de un botón que permitirá validar los hallazgos cuando esta operación no se hubiera realizado todavía. Contará con toda la información introducida durante el reporte, así como la opción de abrir en Google Maps la ubicación del hallazgo y una sección de comentarios, donde tanto cualquier usuario con el rol autoridad, como el profesional que esté validando el hallazgo podrán añadir comentarios.

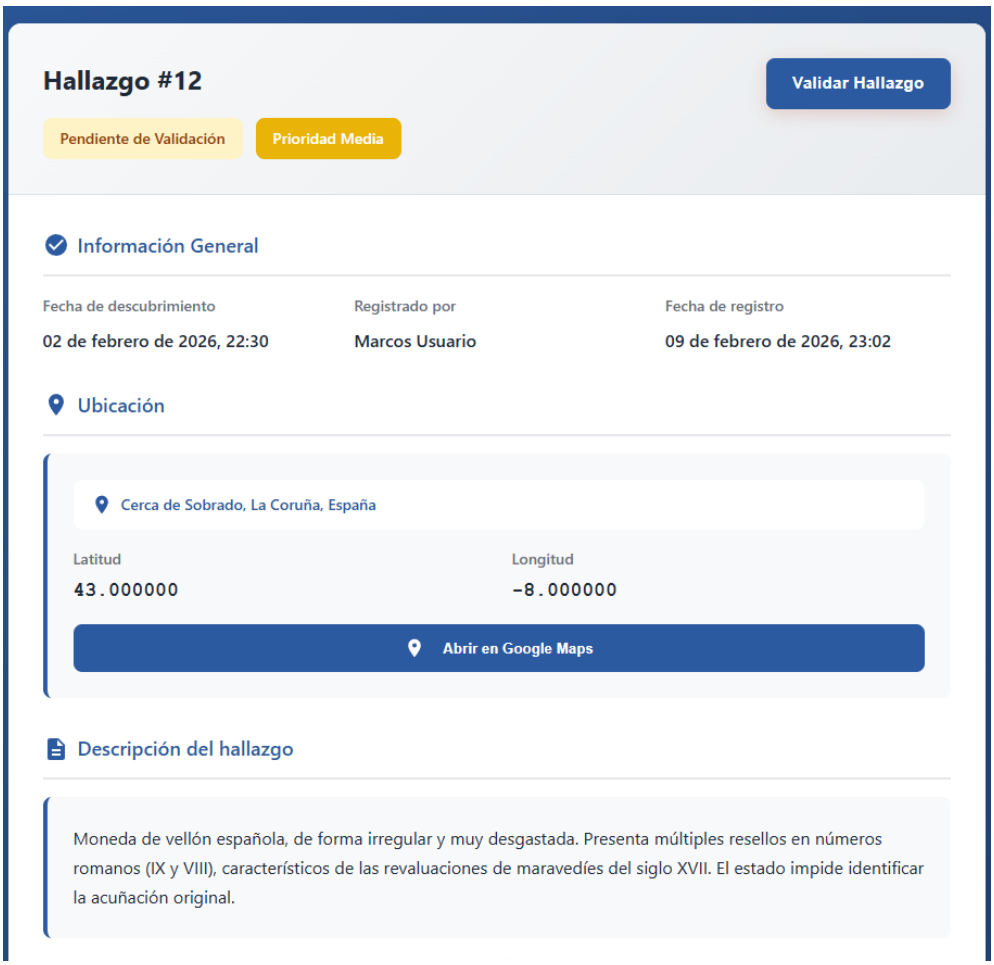

Figura A.18: Detalles de hallazgo I

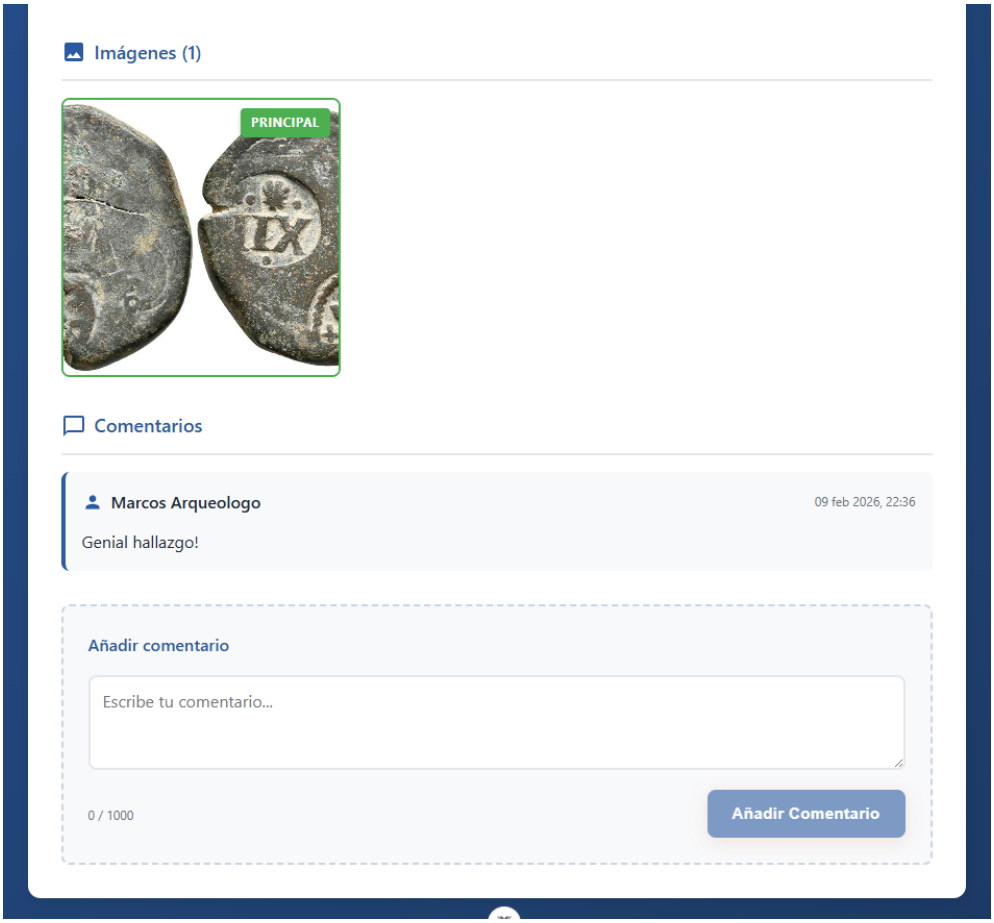

Figura A.19: Detalles de hallazgo II

Figura A.20: Visor en pantalla completa

Al validar un hallazgo aparecerá un formulario que permitirá aceptarlo o rechazarlo, y requerirá asignarle la prioridad al mismo así como añadir un comentario justificando esa decisión.

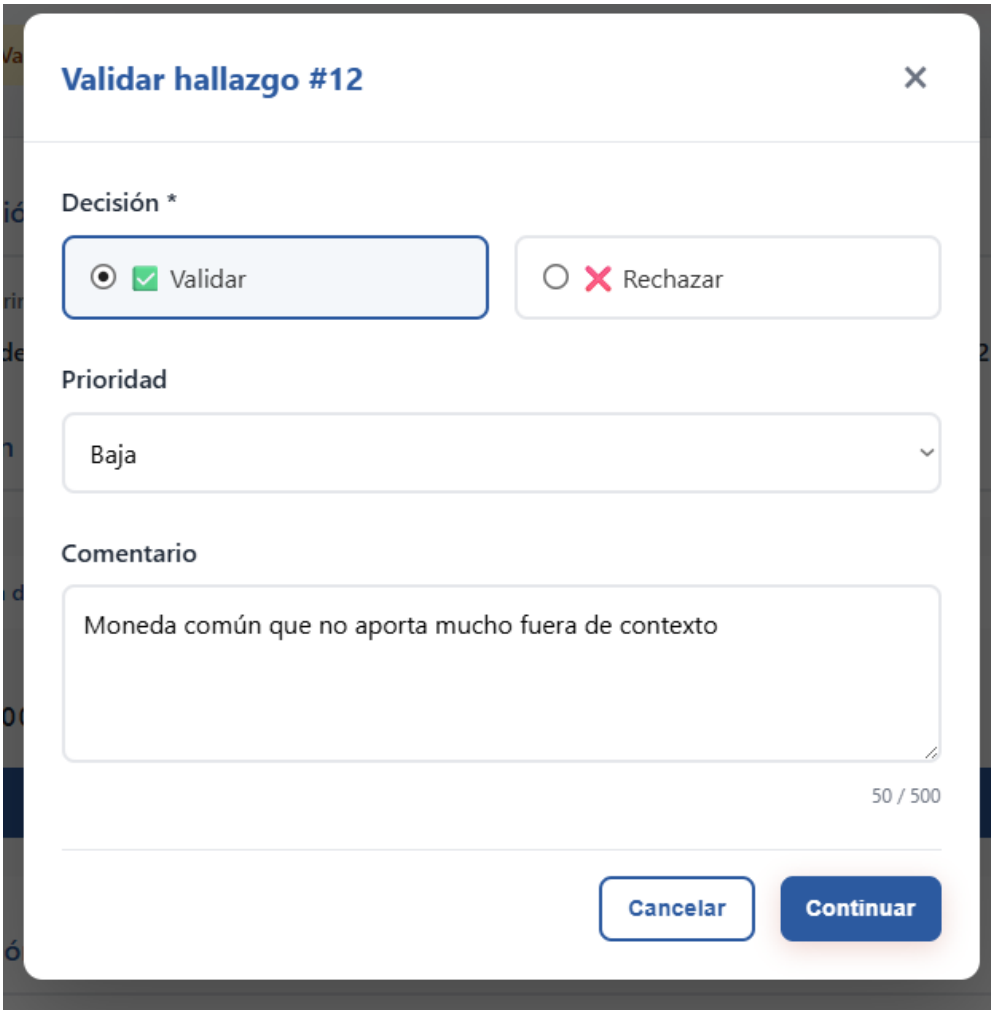

Figura A.21: Información de validación

A.5 Panel de análisis (solo autoridades)

Esta sección, disponible solo para autoridades, permitirá consultar información de especial relevancia que facilite la toma de decisiones. Dispone de tres secciones:

A.5.1 Estadísticas generales

Información de especial relevancia, resumen de volumen de reportes, actividad de usuarios e información geográfica de interés.

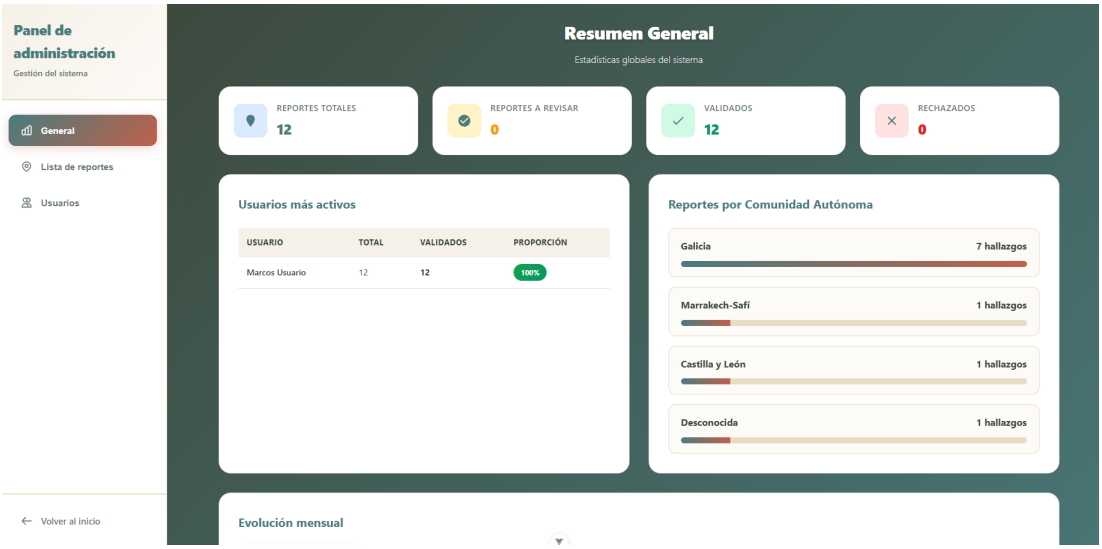

Figura A.22: Panel I - Estadísticas, primera parte

Evolución mensual de los reportes para identificar patrones de actividad.

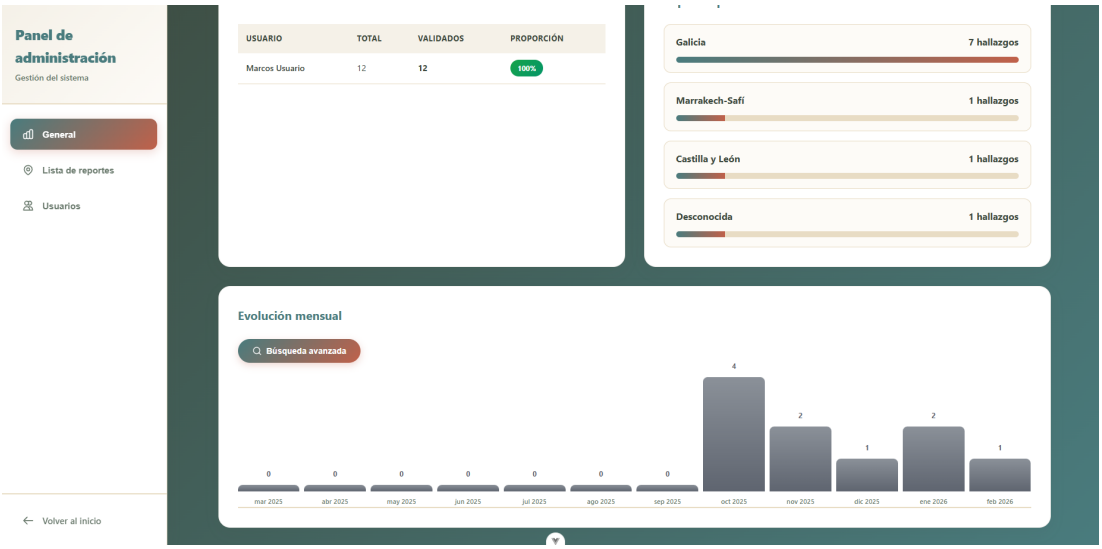

Figura A.23: Panel I - Estadísticas, segunda parte

A.5.2 Lista de miembros

Figura A.24: Panel II - Lista de usuarios dados de alta

A.5.3 Lista de hallazgos

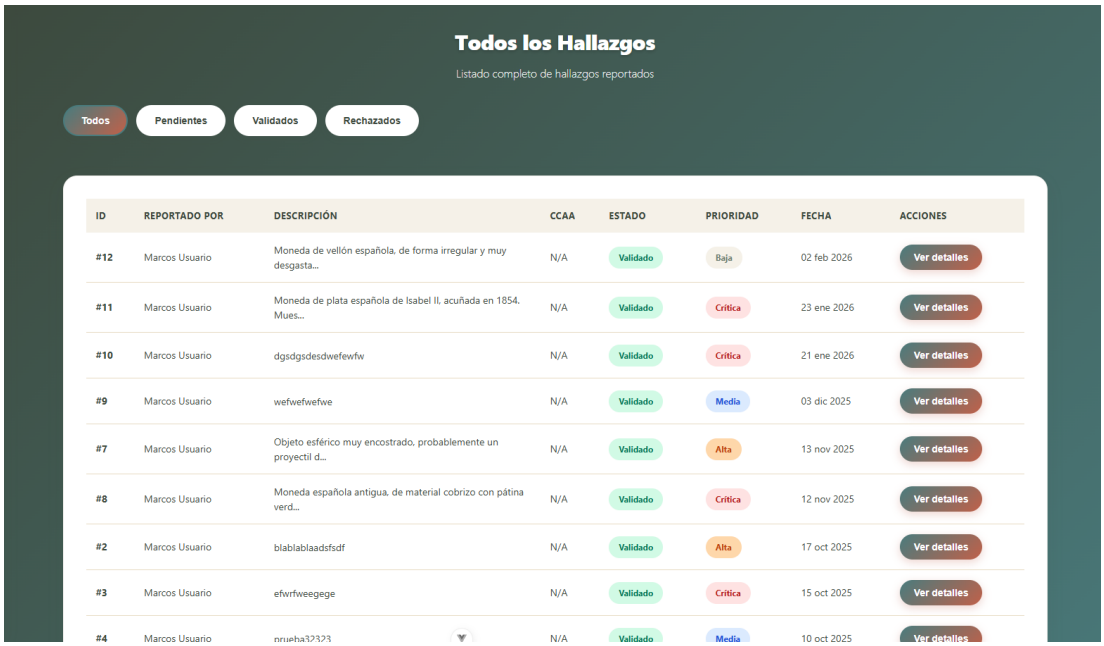

Figura A.25: Panel III - Lista de hallazgos reportados

A.6 Mapa de protección

En él, las autoridades pueden añadir o eliminar zonas y monumentos protegidos con el fin de disponer de una referencia visualmente organizada que ayude a identificar potenciales expolios o zonas sensibles con una alta actividad de hallazgos, lo que podría, por ejemplo, dar indicio de la necesidad de regular actividades de remoción de suelos en dicha área.

Los monumentos se crearán con un marcador de coordenadas únicas, mientras que para las zonas la creación se realizará dibujando un polígono de tantos lados como necesite el usuario. La información se introducirá tras marcar el elemento geográficamente.

A.6.1 Vista general

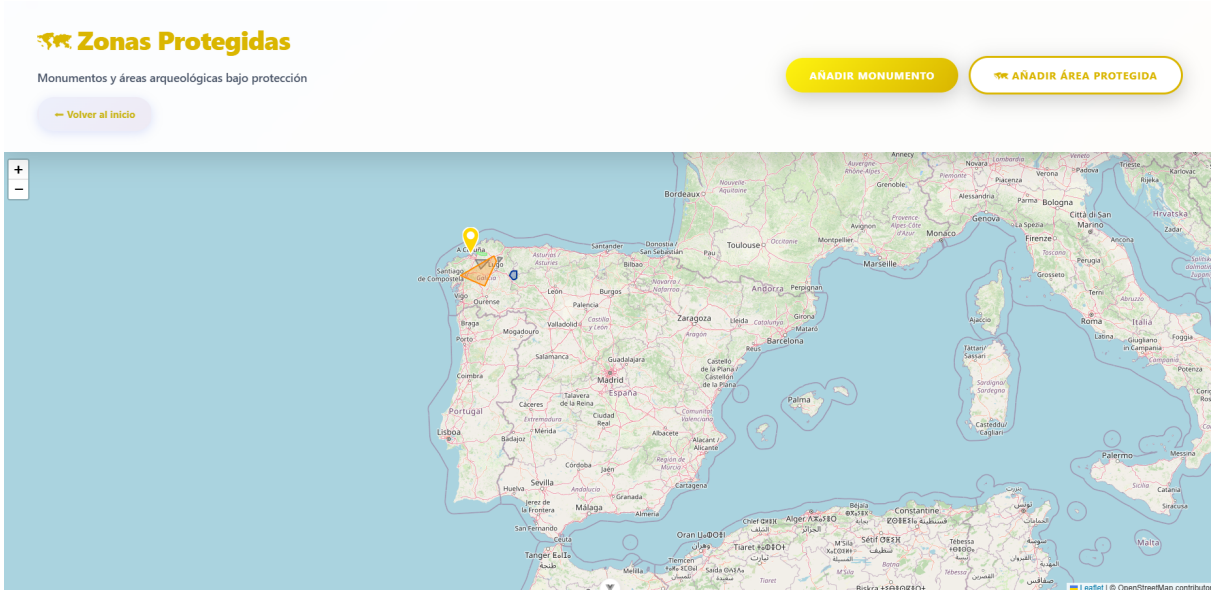

Figura A.26: Mapa I - Vista genérica

A.6.2 Información de áreas

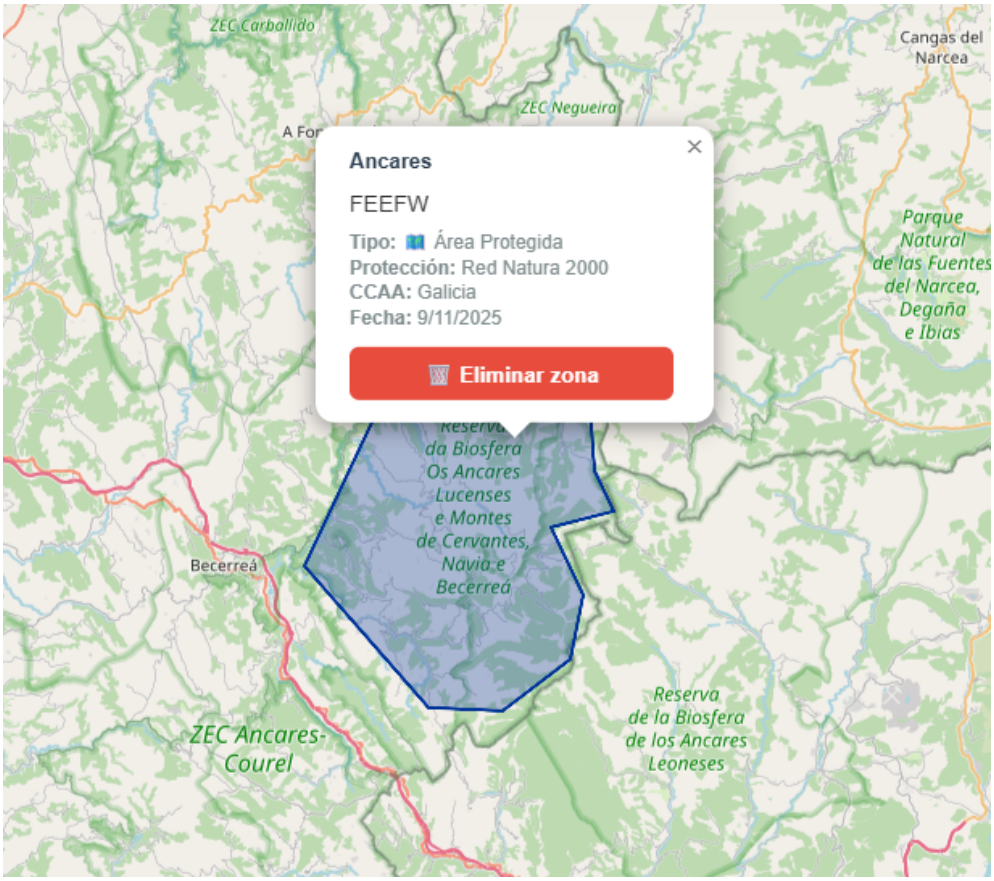

Figura A.27: Mapa II - Detalles de una zona

A.6.3 Creación de áreas

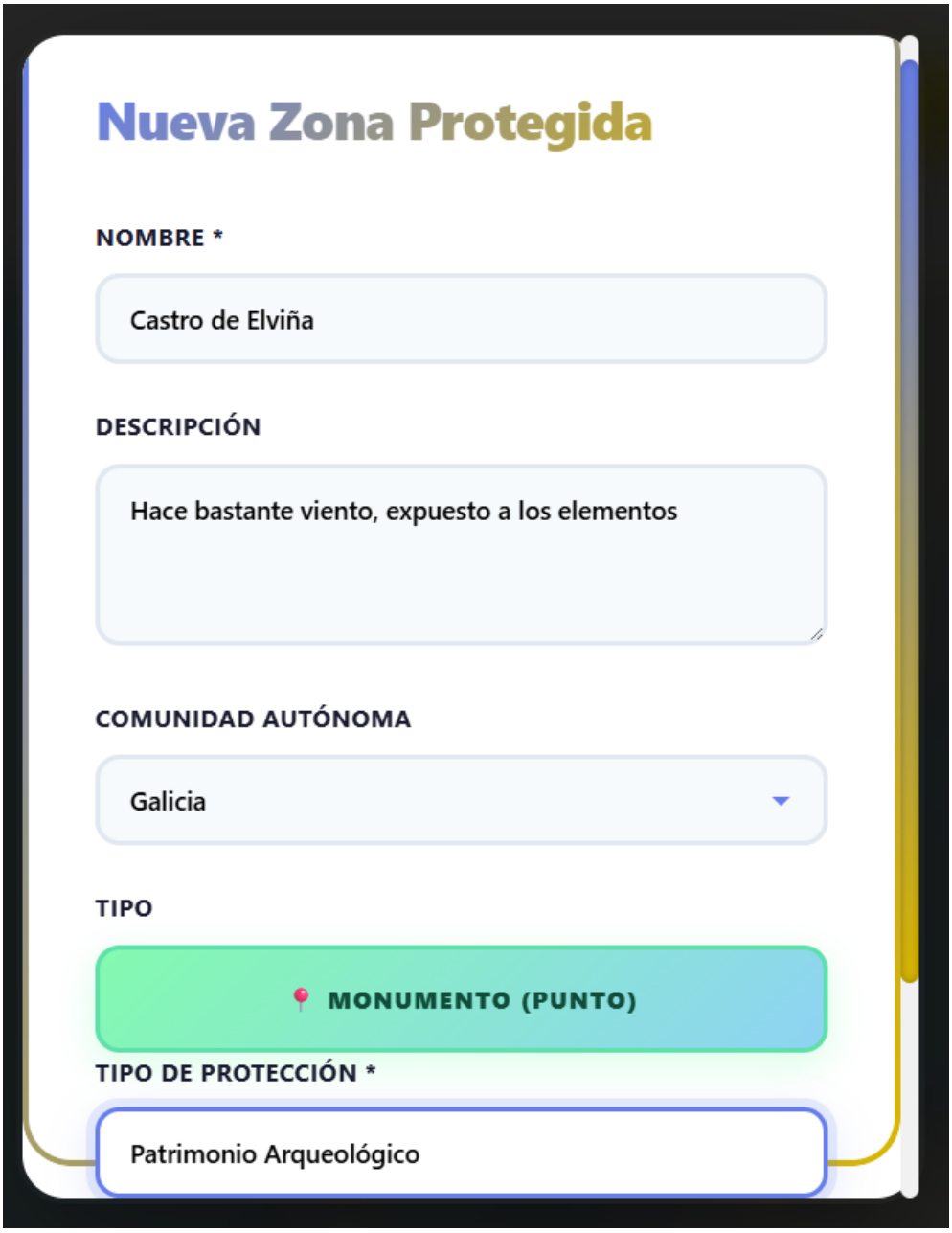

Figura A.28: Mapa III - Formulario de creación de una zona

Lista de acrónimos

API Application Programming Interface. 3, 7, 10, 24, 35, 37, 41, 48

DTO Data Transfer Object. 37

IDE Integrated Development Environment. 11, 12, 53

JPA Java Persistence API. 8, 37, 40, 44

JSON JavaScript Object Notation. 12, 51

JWT JSON Web Token. 7, 22, 41, 44

ORM Object Relational Mapper. 8

PAS Portable Antiquities Scheme. 1, 2, 4

POJO Plain Old Java Object. 37

REST Representational State Transfer. 7, 10, 41

SaaS Software As Service. 4

SQL Structured Query Language. 8

Bibliografía

[1] L. Rey. (2014, ene) As pallozas, un icono de galicia en estado de abandono. consultado el 12 de febrero de 2026. [En línea]. Disponible en: https://www.lavozdegalicia.es/noticia/lugo/2014/01/10/pallozas-icono-galicia-estado-abandono/0003_201401L10C9991.htm

[2] Historic England. (2025, mar) The value of high street heritage action zones. consultado el 12 de febrero de 2026. [En línea]. Disponible en: https://historicengland.org.uk/advice/heritage-action-zones/regenerating-historic-high-streets/evaluation-report/

[3] TheVeteransAssociation, "Veteran metal detecting uk: A hobby with purpose," 2025, consultado el 12 de febrero de 2026. [En línea]. Disponible en: https://www.theveteransassociation.org/blog/veteran-metal-detecting-uk-a-hobby-with-purpose/

[4] First Online. (2025) El museo británico registra 56.000 hallazgos arqueológicos en un año. consultado el 12 de febrero de 2026. [En línea]. Disponible en: https://www.firstonline.info/es/rapporto-annuale-del-british-museum-pas-aumento-dei-reperti-trovati-in-costante-crescita/

[5] Reuters. (2009) El museo británico registra 56.000 hallazgos arqueológicos en un año. consultado el 12 de febrero de 2026. [En línea]. Disponible en: https://cadenaser.com/ser/2009/09/24/internacional/1253753122_850215.html

[6] Wikipedia. (2025) Entrada de java en wikipedia. consultado el 12 de febrero de 2026. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Java_(lenguaje_de_programación)

[7] Inesdi Business School. (2024) Typescript vs javascript: ¿qué diferencias hay

entre ambos? consultado el 12 de febrero de 2026. [En línea]. Disponible en: https://www.inesdi.com/blog/typescript-vs-javascript/

[8] Oficina de Coordinación FLOSS, Xunta de Galicia, "Informe de seguimento del plan de acción en materia de software libre año 2011," Consellería de Medio Ambiente, Territorio e Infraestructuras - Xunta de Galicia, Tech. Rep., 2011. [En línea]. Disponible en: https://www.mancomun.gal/wp-content/uploads/2016/11/BalanceFLOSS2011-es.pdf

[9] Wikipedia. (2025) Entrada de arquitectura hexagonal (software). consultado el 12 de febrero de 2026. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Arquitectura_hexagonal_(software)

[10] Amazon Web Services. (2025) Patrón de arquitectura hexagonal. consultado el 12 de febrero de 2026. [En línea]. Disponible en: https://docs.aws.amazon.com/es_es/prescriptive-guidance/latest/cloud-design-patterns/hexagonal-architecture.html

[11] Asociación Española de Asesores Fiscales. (2025) La carga fiscal para el salario medio supone el 52,4 por ciento del sueldo completo. consultado el 12 de febrero de 2026. [En línea]. Disponible en: https://www.aedaf.es/es/documentos/descarga/72029/la-carga-fiscal-para-el-salario-medio-supone-el-52-4-del-sueldo-completo