UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

GRADO EN INGENIERÍA DE SOFTWARE

Portal de ontología. Un repositorio para consumo de datos estructurados y algoritmos relacionados con la extracción y procesamiento de datos

Ontology portal. A repository intended for structured data consumption and data mining related algorithms

Realizado por
Nicolás Zhilie Zhao

Tutorizado por
María del Mar Roldán García
Cristóbal Barba González

Departamento
Lenguajes y Ciencias de la Computación

MÁLAGA, MAYO DE 2023

UNIVERSIDAD
DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA DEL SOFTWARE

Portal de ontología. Un repositorio para consumo de datos estructurados y algoritmos relacionados con la extracción y procesamiento de datos

Ontology portal. A repository intended for structured data consumption and data mining related algorithms

Realizado por:

Nicolás Zhilie Zhao

Tutorizado por:

María del Mar Roldán García

Cristóbal Barba González

Departamento:

Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA

MÁLAGA, MAYO DE 2023

Resumen

La web semántica es un área de investigación y desarrollo muy relevante en la actualidad. El aumento exponencial del volumen de datos y la necesidad de manejar información de manera eficiente han llevado a las tecnologías semánticas a ganar más importancia. En particular, la web semántica se utiliza cada vez más para mejorar la calidad y precisión de la información disponible en línea, lo que resulta en una mejor experiencia del usuario y una mayor eficiencia en la gestión de datos.

En consecuencia, han surgido muchos portales web que utilizan la web semántica para organizar y clasificar la información en línea. Estas aplicaciones proporcionan un punto centralizado para acceder a los datos de ontologías, lo que facilita su uso y ayuda a los usuarios a encontrar la información que necesitan de manera más rápida y eficiente.

En este contexto, el presente Trabajo de Fin de Grado se enfoca en el desarrollo de un portal web de ontologías para el proyecto nacional "Aether". Este proyecto es una iniciativa que reúne a cinco grupos de investigación de universidades españolas en el campo de análisis de datos y Big Data. El objetivo principal del proyecto es encontrar soluciones tecnológicas para problemas reales de interés para la industria y la sociedad, como la agricultura, la medicina , el análisis de datos científicos, las ciudades inteligentes, la Industria 4.0 y el análisis de riesgos. En este sentido, el portal web de ontologías será una herramienta crucial para mejorar la eficiencia y calidad de la información disponible en línea.

El trabajo no solo describirá el proceso de desarrollo de la aplicación, desde la metodología hasta la implementación de la solución, sino que también se enfocará en el rendimiento, la accesibilidad y el SEO, para asegurar la mejor experiencia posible para los usuarios.

Palabras clave: Ontologías, Web Semántica, Ciudades Inteligentes, Accesibilidad, SEO, Análisis de Datos, Big Data.

Abstract

The semantic web is a very relevant area of research and development in the present. The exponential increase in the volume of data and the need to handle information efficiently have led to semantic technologies to gain more importance. In particular, the semantic web is used increasingly to improve the quality and accuracy of the information available online, resulting in a better user experience and greater efficiency in data management.

Consequently, many web portals using the semantic web have emerged. to organize and classify information online. These apps provide a centralized point for accessing ontology data, making it easier to use and helps users find the information they need. they need faster and more efficiently.

In this context, the present Final Degree Project focuses on the development of a web portal of ontologies for the national project "Aether". This project is an initiative that brings together five research groups from spanish universities in the field of data analysis and Big Data. The main objective of the project is to find technological solutions for real problems of interest for industry and society, such as agriculture, medicine, scientific data analysis, smart cities, Industry 4.0 and risk analysis. In this sense, the web portal of ontologies will be a crucial tool to improve the efficiency and quality of information available online.

The work will not only describe the development process of the application, from the methodology until the implementation of the solution, but also will focus on performance, accessibility and SEO, to ensure the best possible experience for users.

Keywords: Ontologies, Semantic Web, Smart Cities, Accessibility, SEO, Data Analysis, Big Data.

Índice

1. Introducción

1.1. Contexto y problema

El término Web Semántica es definido por el consorcio W3C[1] como la web de datos enlazados. W3C busca, mediante los datos y metadatos enlazados, definir la Web 3.0 que permita mejorar internet ya que amplía la interoperabilidad entre sistemas informáticos. Todo ello gracias el uso de metadatos para la descripción de los datos que se pueden encontrar en la Web [1]. Las diferentes tecnologías de Web Semántica permiten a las personas crear datos y metadatos almacenados en la Web, crear vocabularios, y escribir reglas para manejar datos. Los datos enlazados se definen mediante tecnologías como OWL [2], RDF [3], SPARQL [4] y SKOS [5].

A rasgo de eso han surgido diversos portales Web para facilitar el acceso de estos datos enriquecidos desde aplicaciones software. Entre las herramientas más destacadas se encuentran BioPortal [6], AgroPortal [7], EcoPortal [8] y MedPortal [9].

1.2. Motivación

El grupo de investigación Khaos Research[2] de la Universidad de Málaga dispone de un gran número de ontologías creadas en los diferentes proyectos en los que ha estado involucrado [10, 11, 12, 13]. Estas ontologías están entre ellas relacionadas ya que sus datos están entrelazados. El objetivo de este proyecto es generar una aplicación Web que permita aprovechar estos datos enlazados en las ontologías para presentárselas a los usuarios finales de una manera fácil e intuitiva, tal como se hace también en BioPortal [14]. Con la idea de que los usuarios puedan buscar los datos y metadatos ya disponibles en las diferentes ontologías para poder hacer uso de ellos. Además, se busca facilitar la búsqueda de conceptos y de ontologías para facilitar la reutilización de ellas, ya que la reutilización de las ontologías y sus conceptos es uno de los pasos de la metodología para el desarrollo de ontologías [15]. Para ello se va a utilizar las ontologías desarrolladas por el grupo de investigación Khaos Research.

A raíz de esta necesidad, en el presente Trabajo de Fin de Grado diseñamos e implementamos una aplicación Web, cuyo motivación principal es facilitar el acceso y vi-

[1]Página oficial de W3C: `https://www.w3.org/`

[2]Página oficial de Khaos Research: `https://khaos.uma.es/`

sualización rápida de las ontologías del proyecto Aether, así como punto de entrada para documentación más detallada de los datos estructurados.

1.3. Objetivos

1. Diseño y desarrollo de una aplicación Web que muestre todo los datos relevantes de cada ontología, como en BioPortal [14].
2. Facilitar la búsqueda y reutilización de ontologías y sus conceptos.
3. Aplicación de integración y distribución continuo para la automatización de la ejecución de pruebas software, análisis estático, compilación, construcción y despliegue de la aplicación [16].
4. Descripción detallado del proceso de implementación y de la metodología empleada [17].
5. Demostración de la efectividad y correcto funcionamiento de algoritmos empleados, a partir de metodologías de pruebas software.

Para satisfacer estos objetivos se presentarán los siguientes resultados:

- Documento de requisitos [18] donde se especifica todos los detalles relevantes sobre la funcionalidad del portal, así como otros requisitos relacionados con la infraestructura empleada.
- Aplicación Web desarrollada con las tecnologías Web más relevantes en la actualidad [19].
- Script para la transformación de los datos semánticos a estructuras de datos más compatible con las técnicas de desarrollo Web más actualizadas.
- Manual de usuario sobre el uso de esta aplicación.

1.4. Estructura del documento

Este Trabajo de Fin de Grado se compone de cinco capítulos que recogen toda la información relevante sobre el desarrollo del proyecto. A continuación, se presenta una breve descripción de cada uno de ellos.

En el Capítulo 2, se realiza una revisión exhaustiva de los conceptos y tecnologías clave en los que se basa el proyecto, con especial énfasis en la Web Semántica y el desarrollo Web. Además, se profundiza en las tecnologías y herramientas software utilizadas en el desarrollo del proyecto, y se presenta el estado del arte en los campos de la Web Semántica

y el desarrollo Web, haciendo referencia a las ontologías relevantes y a las aplicaciones desarrolladas en el dominio.

El Capítulo 3 describe detalladamente la metodología software y el diseño del sistema, incluyendo los requisitos funcionales y no funcionales, así como los casos de uso necesarios para asegurar el cumplimiento de dichos requisitos.

En el Capítulo 4 se detalla la implementación del sistema, destacando los algoritmos de filtro y las estructuras de datos seleccionados para lograr el mejor rendimiento posible en el lado del cliente. Además, se especifica la metodología de pruebas utilizada y se presentan los marcos de trabajo empleados.

En el Capítulo 5, se presenta un tour de la aplicación desarrollada, relacionando cada componente de la interfaz con su correspondiente funcionamiento.

En el Capítulo 6, se discuten las ventajas e inconvenientes del diseño del sistema implementado en este Trabajo de Fin de Grado, así como las alternativas de las infraestructuras hardware y software disponibles, analizando los diferentes aspectos relevantes y las posibles mejoras a considerar en futuros desarrollos.

En resumen, este Trabajo de Fin de Grado aborda el diseño y desarrollo de un portal web de ontologías para el proyecto nacional “Aether”, con el objetivo de mejorar la calidad y eficiencia de la información disponible en línea.

2. Estado del Arte

2.1. Evolución Web

En el artículo científico «Web evolution and Web science» [20], se examina la progresión de la World Wide Web o WWW, como una red de redes, desde sus comienzos en los primeros años de los 90 hasta lo que se conoce hoy como Web 4.0 [21].

La WWW comenzó primero como la Web de Lectura o Web 1.0 [22], donde el usuario solamente podía consumir contenido almacenado en formato estándar de Hyper Text Markup Language o HTML. Su posterior evolución permitió el paso a la Web Social o Web 2.0, centrada en la creación y compartición de contenidos. Sin embargo, para proporcionar software capaz de procesar contenido Web, razonar sobre él y producir inferencias automáticamente, se necesitaba una extensión de la Web 2.0. Esta extensión es conocida como la Web Semántica o Web 3.0. Finalmente, la evolución de la WWW ha dado lugar a la Web Ubicua o Web 4.0, cuyas principales características son el provisionamiento a tiempo real de servicios software, reconocido como cloud computing [23] y la interacción con cualquier tipo de objetos [24].

En la siguiente sección nos centraremos únicamente en la Web Semántica, centrándonos en su origen y evolución.

2.2. La Web Semántica

La Web Semántica surge de la necesidad de dotar a la Web de un significado para que las máquinas puedan comprender y procesar su contenido de manera más efectiva. La idea detrás de la Web Semántica es añadir metadatos y ontologías a la Web para permitir la integración y el intercambio de información entre aplicaciones y sistemas de una manera más automatizada y precisa.

En la Web Semántica, los recursos de la Web son descritos mediante ontologías, que son modelos formales que especifican los conceptos y las relaciones entre ellos. Estos modelos son utilizados por los agentes inteligentes para razonar sobre los datos y extraer información útil.

La Web Semántica ha evolucionado en la forma en que se modelan los datos. En la primera etapa, la Web Semántica se centró en la representación de los datos en forma de grafos RDF (Resource Description Framework) [3], que se utilizan para representar

información semántica. Posteriormente, la ontología OWL (Web Ontology Language) [2] se desarrolló para permitir la definición de ontologías en la Web Semántica. OWL es una familia de lenguajes de ontologías que se utilizan para representar conocimiento formal y permitir la inferencia automática en la Web Semántica. Ambas representaciones forman parte del "Tech stack"[3], como podemos observar en la Figura 1.

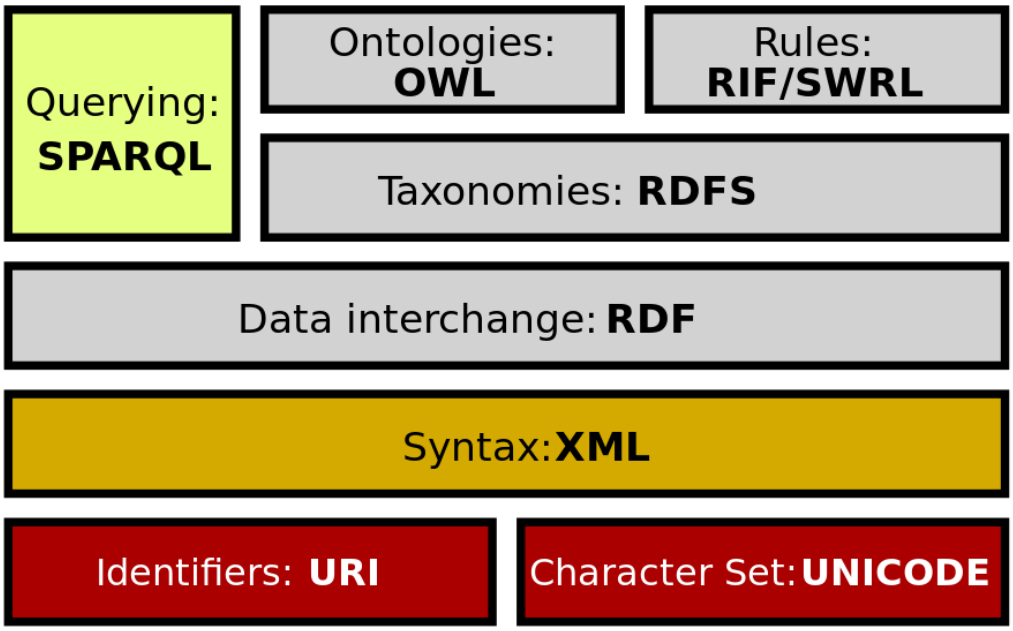

Figura 1: Conjunto de tecnologías de la Web Semántica.

Resource Description Framework (RDF)

RDF es un estándar del W3C para representar la información en forma de grafos dirigidos etiquetados. En RDF, los recursos se representan como nodos y las relaciones entre ellos se representan como vértices etiquetados.

RDF proporciona un marco para describir información y relaciones entre recursos en la web de una manera estructurada y procesable por máquina. Es compatible con varios formatos de serialización, como RDF/XML, Turtle, N-Triples y JSON-LD, lo que facilita la integración de datos de diferentes fuentes y su procesamiento por aplicaciones web semánticas.

En el Fragmento 1 se muestra un ejemplo.

[3]Tech stack: Conjunto de herramientas, tecnologías y lenguajes de programación utilizados para construir y mantener un software o sistema de información.

```
<http://example.com/juan> <http://example.com/come> <http://example.com/pizza> .
```

Fragmento de código 1: Ejemplo de código RDF

En el Fragmento 1, se afirma que Juan (identificado por la URI[4] "http://example.com/juan") come (identificada por la URI "http://example.com/come") pizza (identificada por la URI "http://example.com/pizza").

La sintaxis de una declaración RDF consta de tres partes:

- Sujeto: es una URI que identifica el recurso que se está describiendo. En el ejemplo anterior, el sujeto es "http://example.org/juan".
- Predicado: es una URI que identifica la propiedad o relación que se está describiendo. En el ejemplo anterior, el predicado es "http://example.org/come".
- Objeto: es un valor o recurso que se está relacionando con el sujeto mediante el predicado. En el ejemplo anterior, el objeto es "http://example.org/pizza".

Una sentencia RDF se termina con un punto ("."). Es importante destacar que la URI utilizada en la declaración RDF no tiene que ser necesariamente una URI de un recurso en la web, sino que puede ser una URI local que se utilice para identificar un recurso específico.

Ontology Web Language (OWL)

OWL (Web Ontology Language) es un lenguaje de ontologías que permite la descripción formal de conceptos, clases, propiedades y relaciones entre ellos. Se utiliza para definir y representar el conocimiento en la Web Semántica. En el Fragmento 2, se presenta un ejemplo de su sintaxis:

[4]URI: Uniform Resource Identifier, es una cadena de texto que permite la identificación de recursos en Internet

```xml
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:ex="http://www.example.com/ontologies/example.owl#"
         xmlns:owl="http://www.w3.org/2002/07/owl#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:vann="http://purl.org/vocab/vann/">

   <owl:Ontology rdf:about="http://www.example.com/ontologies/example.owl" />

   <owl:Class rdf:about="http://www.example.com/ontologies/example.owl#Persona" />

   <owl:ObjectProperty rdf:about="http://www.example.com/ontologies/example.owl#conoce">
      <rdfs:domain rdf:resource="http://www.example.com/ontologies/example.owl#Persona" />
      <rdfs:range rdf:resource="http://www.example.com/ontologies/example.owl#Persona" />
   </owl:ObjectProperty>

   <owl:NamedIndividual rdf:about="http://www.example.com/ontologies/example.owl#Juan">
      <rdf:type rdf:resource="http://www.example.com/ontologies/example.owl#Persona" />
      <ex:conoce rdf:resource="http://www.example.com/ontologies/example.owl#Maria" />
   </owl:NamedIndividual>

   <owl:NamedIndividual rdf:about="http://www.example.com/ontologies/example.owl#Maria">
      <rdf:type rdf:resource="http://www.example.com/ontologies/example.owl#Persona" />
   </owl:NamedIndividual>
</rdf:RDF>
```

Fragmento de código 2: Ejemplo de código OWL

Este Fragmento 2 define una ontología simple con una clase “Persona” y una propiedad “conoce” que se establece entre dos individuos de la clase “Persona”. También se definen dos individuos, “Juan” y “Maria”, ambos son de la clase “Persona” y “Juan” tiene una relación “conoce” con “Maria”.

Los elementos básicos de OWL se muestran en la Tabla 1.

Elemento	Descripción
Clases	Definido con owl:Class
Propiedades	Definido owl:ObjectProperty o owl:DatatypeProperty
Individuos	Definido owl:NamedIndividual
Restricciones	Definido owl:Restiction
Axiomas	Definido con owl:Axiom

Tabla 1: Elementos básicos de OWL

Evolución Web Semántica

En el artículo «A review of the semantic web field» [25], sugiere una perspectiva subjetiva de la evolución de la Web Semántica. El documento distingue 3 fases, la primera era motivada por las "ontologías", la segunda conducida por "datos enlazados" y en la tercera, que es la fase actual, "grafos de conocimientos".

Ontologías. Predominó en la década de 2000 y su origen tiene lugar en el artículo científico de «Toward principles for the design of ontologies used for knowledge sharing?» [26]. El objetivo era proporcionar el vocabulario y la representación estándar para garantizar el alineamiento y coherencia de los datos.

Datos enlazados. Surgió en el año 2006 y se convertió en el objeto principal de investigación hasta los comienzos de la década de 2010. El objetivo es la construcción de grafos RDF de tamaño descomunales, enlazando los grafos RDF a partir de sus Uniform Resource Indentifiers (URIs) que aparecen repetidos en múltiples grafos. Uno de los datasets enlazados más reconocidos es *DBPedia* [27], cubriendo un conjunto de 6 millones de entidades y 9.5 mil millones de tripletas RDF, todos ellos extraídos y procesados desde *Wikipedia* [28].

Grafos de conocimiento. El concepto de grafo de conocimiento, en el campos de Web semántica fue introducido por primera vez en el artículo *Introducing the Knowledge Graph: things, not strings* [29]. Es la base del motor de búsqueda de Google que permite la presentación de información contextualizada y respuestas más precisas a las consultas de los usuarios. El grafo de conocimiento de Google es una gran base de datos interconectada de entidades, conceptos y relaciones entre ellos, que permite a los usuarios encontrar información relevante en función del contexto de sus búsquedas. En el grafo de

conocimiento, las entidades se representan como nodos y las relaciones entre ellas como arcos, permitiendo una representación más rica y detallada de la información que en los sistemas de búsqueda tradicionales. El éxito del grafo de conocimiento de Google ha llevado a otros actores del sector, como Facebook y Microsoft, a desarrollar sus propios grafos de conocimiento con el objetivo de mejorar la calidad y la relevancia de las respuestas de sus motores de búsqueda.

El término "grafo de conocimiento" se ha adoptado ampliamente en el campo de la inteligencia artificial y se ha utilizado para describir una variedad de sistemas de representación y razonamiento del conocimiento a gran escala.

Numerosos estudios recientes busca la automatización de la construcción de estos grafos de conocimientos desde fuentes de datos de múltiples lenguajes naturales. Para ello el empleo números algoritmos de extracción y procesado son requeridos, en el artículo «Research of Chinese intangible cultural heritage knowledge graph construction and attribute value extraction with graph attention network» [30], propone un marco de trabajo general para construir un grafo de conocimiento a partir del patrimonio cultural inmaterial chino, apoyándose en las bases teóricas de otras publicaciones como «Bidirectional LSTM-CRF Models for Sequence Tagging» [31], «An Intelligent Question Answering System of the Liao Dynasty Based on Knowledge Graph» [32] y «Preliminary Study on the Knowledge Graph Construction of Chinese Ancient History and Culture» [33].

Además de todo lo mencionado anteriormente, la Web semántica es aplicada en muchos campos de investigación como en el data mining [34], en la medicina [35], en la inteligencia artificial [36], en la agricultura [37], en la ganadería [38], en la pesca [39], etc.

2.3. Portales de ontología

Existen numerosos portales de ontologías que ofrecen acceso a una gran cantidad de ontologías y datos de conocimiento en una variedad de disciplinas, desde la biomedicina hasta las ciencias sociales y la informática. Estos portales proporcionan herramientas y servicios para la búsqueda, navegación, visualización y descarga de ontologías y datos relacionados, lo que facilita el desarrollo y la implementación de sistemas inteligentes y aplicaciones basadas en la web semántica.

En la siguiente lista, se presentan algunos de los portales de ontologías más relevantes, junto con una breve descripción de cada uno.

- **BioPortal** [6]: Un portal de ontologías y datos de conocimiento biomédicos que ofrece acceso a más de 600 ontologías, incluidas las ontologías del Consorcio de Ontología Biomédica (BCO) y otras ontologías relevantes para la investigación biomédica.
- **EcoPortal** [8]: Una plataforma para el intercambio y acceso de recursos semánticos de ecología, que cumple con los principios de Findable, Accessible, Interoperable y Reusable (FAIR) para datos y recursos. El portal ofrece un conjunto de herramientas que facilitan la creación, el descubrimiento y la reutilización de ontologías, vocabularios y otros recursos semánticos en el dominio de la ecología.
- **MedPortal** [9]: Un repositorio y plataforma de ontología biomédica enfocado en la medicina de precisión. Está diseñado para integrar diversos tipos de datos relacionados con la medicina de precisión, incluidos datos genéticos, genómicos, clínicos y ambientales, y proporcionar una interfaz estandarizada y fácil de usar para que investigadores y médicos accedan y analicen los datos.
- **AgroPortal** [7]: Un portal de ontologías y datos de conocimiento agrícola que ofrece acceso a más de 30 ontologías y vocabularios relacionados con la agricultura, la ganadería y la alimentación. Este portal es desarrollado por la Iniciativa de Datos Abiertos Agrícolas (AgroDataCube) y ofrece herramientas y servicios para la búsqueda, navegación y visualización de ontologías, así como también para la integración y el enriquecimiento de datos agrícolas.

Comparaciones

Tenemos referencias a otros trabajos sobre metodologías de cómo comparar portales de ontología [40], sin embargo, el campo de desarrollo web ha avanzado de forma veloz desde la última publicación (2014) que se realiza una comparación. Por ello, a continuación se presenta otra forma de medida más cercana al estado del arte del desarrollo web.

Para comparar los distintos sitios web se ha usado "Google Lighthouse"[5], una herramienta de código abierto desarrollada por Google que se utiliza para evaluar la calidad de los sitios web en función de un conjunto de métricas definidas. Estas métricas incluyen la velocidad de carga, la accesibilidad, las mejores prácticas de desarrollo web y el rendimiento en motores de búsqueda.

[5]Documentación oficial de Google Lighthouse: `https://developer.chrome.com/docs/lighthouse/overview/`

La Tabla 2 muestra los resultados de "Google Lighthouse" para los sitios web de mencionados en el apartado anterior, sobre el endpoint[6] "/search" porque es donde reside la funcionalidad que más demanda de rendimiento, la búsqueda de los elementos.

Sitio web	**Rendimiento**	**Accesibilidad**	**Buenas prácticas**	**SEO**[7]
BioPortal	91	75	83	?
MedPortal	88	87	83	64
EcoPortal	85	75	83	64
AgroPortal	92	75	92	?
AetherPortal	100	100	100	100

Tabla 2: Resultados del informe de Google Lighthouse.

Las respectivas imágenes de los resultados procedentes de Google Lighthouse se encuentran en el apéndice A.

2.4. Proyecto Aether

El proyecto "Aether" es una iniciativa en España que involucra a cinco grupos de investigación de diferentes universidades, enfocados en análisis de datos y Big Data. Su objetivo principal es encontrar soluciones tecnológicas para problemas reales en diferentes áreas, como la agricultura, la medicina, el análisis científico, las ciudades inteligentes, la industria y el análisis de riesgos. Cada universidad se enfoca en un área específica y el proyecto utiliza un marco de trabajo para proporcionar una estructura base. El objetivo es desarrollar técnicas de inteligencia artificial seguras, escalables y autoconfigurables que puedan acercarse a los usuarios finales.

Fuente: *El proyecto nacional 'Aether', punto de encuentro de cinco de los principales grupos de I+D en análisis de datos y Big Data* [41].

[6]Endpoint: Dirección de una API (Interfaz de Programación de Aplicaciones) a la que se puede acceder para interactuar con un servicio o aplicación web.

2.5. Tecnologías utilizadas

Python[8]

Python es un lenguaje de programación de alto nivel y de código abierto que se utiliza en una amplia variedad de aplicaciones, desde la creación de scripts y la automatización de tareas hasta el desarrollo de aplicaciones web y móviles, aprendizaje automático e inteligencia artificial.

A continuación se muestra las librerías más relevantes para el desarrollo del trabajo:

- rdflib: Librería que proporciona una interfaz genérica para interactuar con RDF. Entre las funcionalidades relevantes para el presente proyecto, destacan la navegación, serialización y deserialización de los grafos RDF. `https://rdflib.readthedocs.io/en/stable/`
- pytest: Librería para pruebas software del código fuente. `https://docs.pytest.org/en/7.2.x/`

Typescript

TypeScript[9] es un lenguaje de programación de código abierto desarrollado y mantenido por Microsoft que se basa en la sintaxis de JavaScript, pero agrega un sistema de tipos estáticos y otras características orientadas a objetos y de programación funcional. Está diseñado para facilitar el desarrollo de aplicaciones más grandes y complejas mediante la detección temprana de errores y la mejora de la capacidad de mantenimiento del código a medida que los proyectos crecen en tamaño y complejidad.

Playwright

Playwright[10] es una herramienta de automatización de pruebas de extremo a extremo de código abierto que se utiliza para probar aplicaciones web en diferentes navegadores y sistemas operativos.

[8]Python: `https://www.python.org/`
[9]Typescript: `https://www.typescriptlang.org/`
[10]Playwright: `https://playwright.dev/`

Vite

Vite[11] es un sistema de compilación y construcción de aplicaciones web para proyectos basados en JavaScript y TypeScript. Fue desarrollado por Evan You, creador de Vue.js, y está diseñado para mejorar la experiencia de desarrollo de aplicaciones web modernas.

Svelte

Svelte[12] es un framework de JavaScript que se enfoca en la creación de interfaces de usuario dinámicas y reactivas con un rendimiento mejorado. A diferencia de otros frameworks de JavaScript como React o Vue, Svelte no utiliza una librería de tiempo de ejecución en el navegador para manejar la actualización del DOM, sino que compila el código del framework en código JavaScript optimizado para el rendimiento.

Github Actions

GitHub Actions[13] es una herramienta de automatización de flujos de trabajo integrada en GitHub que permite a los desarrolladores automatizar y personalizar sus procesos de trabajo. Con GitHub Actions, los desarrolladores pueden crear flujos de trabajo que se activan automáticamente en respuesta a eventos específicos en sus repositorios de GitHub, como la creación de una nueva solicitud de extracción o la recepción de una nueva confirmación de código.

Los flujos de trabajo de GitHub Actions se pueden personalizar para realizar una amplia variedad de tareas, como la ejecución de pruebas automatizadas y la construcción de paquetes de software. GitHub Actions proporciona una amplia variedad de acciones preconstruidas que los desarrolladores pueden utilizar para automatizar sus flujos de trabajo, así como la capacidad de crear y compartir acciones personalizadas con otros usuarios.

Github Pages

GitHub Pages[14] es un servicio de alojamiento web gratuito proporcionado por GitHub que permite a los usuarios alojar sus sitios web estáticos directamente desde sus reposito-

[11] Vite: `https://vitejs.dev`
[12] Svelte: `https://svelte.dev/`
[13] Github Actions: `https://docs.github.com/en/actions`
[14] Github Pages: `https://pages.github.com/`

rios de GitHub. Los sitios web alojados en GitHub Pages pueden ser públicos o privados y se pueden personalizar para adaptarse a las necesidades del usuario.

Docker

Docker[15] es una plataforma de software que revoluciona la forma en que los desarrolladores crean, prueban y despliegan aplicaciones. Al aprovechar la tecnología de contenedores de software, Docker proporciona un entorno aislado y ligero que encapsula la aplicación y sus dependencias, lo que garantiza que se ejecute de manera coherente en cualquier entorno. A diferencia de las máquinas virtuales tradicionales, los contenedores de Docker son una forma más eficiente y rápida de virtualización, ya que no virtualizan el hardware sino el kernel del sistema operativo. En resumen, Docker permite a los desarrolladores crear y desplegar aplicaciones de forma rápida y fácil, y asegura que funcionen de manera consistente en cualquier entorno.

[15]Docker: `https://www.docker.com/`

3. Metodología

3.1. Descripción de las metodologías

La metodología es un conjunto de principios, técnicas y herramientas que se utilizan para llevar a cabo un proyecto de manera sistemática y efectiva. En cualquier área de trabajo, la metodología es importante para garantizar el éxito del proyecto, ya que ayuda a definir los procesos y procedimientos que se deben seguir para lograr los objetivos establecidos. En este contexto, existen diversas metodologías que se utilizan en distintos campos, cada una con sus propias ventajas y desventajas. En las siguientes secciones, se describirán las metodologías utilizadas en el desarrollo de proyectos, incluyendo Scrum, documento de requisitos, diseño de sistemas y las implementaciones de un proyecto.

3.2. Scrum

El método Scrum es un marco de trabajo ágil que se utiliza para el desarrollo de proyectos en equipo, y se puede aplicar para un trabajo de fin de grado. A continuación, se profundiza la organización llevada a cabo durante el desarrollo del presente trabajo:

- **Establecer el equipo**: En este caso, el equipo está compuesto por el alumno como desarrollador, la tutora como product owner y el cotutor como scrum master.
- **Definir el backlog**: La tutora y el cotutor trabajan juntos para definir el backlog, documento que incluirá todas las tareas necesarias para completar el trabajo de fin de grado, desde la elaboración del marco teórico hasta la presentación final del trabajo. El backlog debe estar ordenado por prioridad, de manera que las tareas más importantes se realicen primero.
- **Realizar los sprints**: El equipo debe planificar las sprints de dos a cuatro semanas de duración cada una, en este caso dos, en las que se concentrará en las tareas más importantes del backlog. El cotutor supervisará el progreso del equipo durante los sprints, asegurándose de que se cumplan las fechas límite y de que el alumno no tenga ningún obstáculo en el camino.
- **Revisar y mejorar**: Al final de cada sprint, el equipo revisa el trabajo realizado y hace mejoras en el proceso. En este momento, el tutor proporciona retroalimentación y sugerencias para mejorar el trabajo del alumno. Además, el cotutor hace ajustes en el backlog y en las sprints posteriores, para garantizar que el proyecto se esté llevando

a cabo de la manera más eficiente y efectiva posible.

- **Entrega del proyecto**: Después de completar todos los sprints, el equipo debe hacer una revisión final del trabajo y hacer los ajustes finales antes de presentar el trabajo de fin de grado.

En la Figura 2 se muestra un diagrama para aclarar el procedimiento descrito previamente.

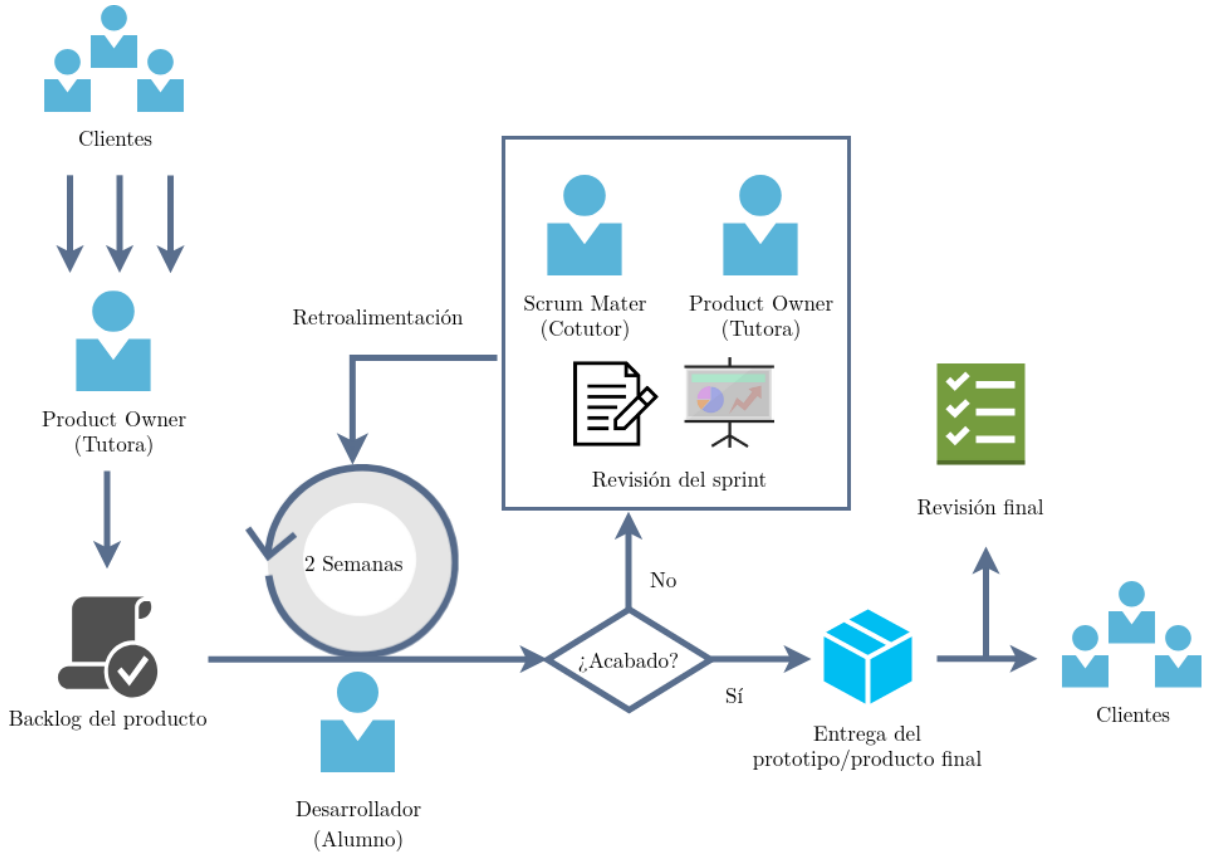

Figura 2: Diagrama del marco de trabajo Scrum

3.3. Descripción de requisitos

Los requisitos funcionales y no funcionales son fundamentales para el desarrollo de cualquier proyecto de software, ya que permiten establecer las características y funcionalidades que debe cumplir el sistema para satisfacer las necesidades del usuario. Los requisitos funcionales describen las funcionalidades específicas que el sistema debe tener para cumplir con los objetivos del proyecto, mientras que los requisitos no funcionales establecen las características que el sistema debe tener en términos de rendimiento, seguridad, usabilidad y otros aspectos relacionados con la calidad del software.

En el caso del desarrollo de un sitio web que sirva de portal para visualizar y consumir ontologías de múltiples sitios externos, los requisitos funcionales se enfocan en las funcionalidades que permiten al usuario visualizar y trabajar con ontologías, mientras que los requisitos no funcionales se enfocan en aspectos relacionados con la calidad del software, como la escalabilidad, mantenibilidad, seguridad y accesibilidad, entre otros.

Por lo tanto, es importante que los requisitos funcionales y no funcionales se definan claramente desde el inicio del proyecto, ya que esto permitirá que el equipo de desarrollo tenga una guía clara para la construcción del sistema, y que los usuarios puedan tener expectativas claras sobre las funcionalidades y características que tendrá el sitio web. Además, es importante que los requisitos se mantengan actualizados durante todo el ciclo de vida del proyecto, ya que esto permitirá ajustar el enfoque del desarrollo de acuerdo a las necesidades del usuario y a los cambios en el entorno del proyecto.

En la Tabla 3, se especifica los requisitos funcionales y en la Tabla 4, se muestra los requisitos no funcionales.

RF1	La aplicación debe permitir la visualización y búsqueda de ontologías provenientes de sitios externos.
RF2	La aplicación debe permitir la integración de ontologías de distintos formatos estándar.
RF3	La aplicación debe permitir la visualización de la estructura de las ontologías.
RF4	La aplicación debe permitir la visualización de los datos de cada ontología.
RF5	La aplicación debe mostrar una documentación detallada de cómo subir una ontología.
RF6	La aplicación debe poder realizar el filtro y búsqueda de ontologías.
RF7	La aplicación debe poder realizar el filtro y búsqueda de clases, individuos y propiedades.
RF8	La aplicación debe poder redirigir el usuario a la documentación oficial de cada individuo, propiedad, clase u ontología.

Tabla 3: Tabla de requisitos funcionales.

RNF1	La aplicación web debe ser fácil de usar y fácil de navegar.
RNF2	La aplicación web debe tener una buena velocidad de carga.
RNF3	La aplicación web debe ser compatible con distintos navegadores web.
RNF4	La aplicación web debe estar disponible las 24 horas del día.
RNF5	La aplicación web debe ser escalable para soportar grandes cantidades de ontologías, alrededor de 100.
RNF6	La aplicación web debe ser fácil de mantener y actualizar.
RNF7	La aplicación web debe cumplir con los estándares de accesibilidad web para personas con discapacidades.
RNF8	La aplicación web debe tener una buena calidad del código fuente y seguir las mejores prácticas de desarrollo.
RNF9	La aplicación web debe tener una buena gestión de errores y excepciones para minimizar los problemas técnicos.
RNF10	La aplicación web debe ser compatible con distintas versiones navegadores.

Tabla 4: Tabla de requisitos no funcionales.

3.4. Casos de uso

Los casos de uso muestran cómo un producto, servicio o tecnología puede ser utilizado en situaciones reales. Los casos de uso pueden proporcionar una perspectiva detallada de cómo un producto o servicio funciona en un escenario específico, y también pueden ilustrar los beneficios y desafíos de su implementación.

En esta sección se muestran detalles de los casos de uso de los principales requisitos funcionales de la aplicación.

Caso de uso	Visualización y búsqueda de ontologías externas
Descripción	El usuario puede ver y buscar ontologías de sitios externos en la aplicación.
Actores	Usuario
Precondiciones	El usuario tiene acceso a Internet y ha abierto la aplicación.
Postcondiciones	El usuario puede ver y buscar ontologías de sitios externos en la aplicación.
Flujo principal	1. El usuario navega hasta la sección "Ontologies". 2. El usuario ingresa palabras clave de búsqueda para las ontologías deseadas. 3. La aplicación muestra una lista de ontologías que coinciden con las palabras clave de búsqueda de sitios externos. 4. El usuario selecciona una ontología de la lista para ver.
Flujos alternativos	A. No se encuentran ontologías que coincidan con las palabras clave de búsqueda. *a*) La aplicación muestra un mensaje informando al usuario que no se encontraron resultados. *b*) El usuario puede intentar con una palabra clave de búsqueda diferente.

Tabla 5: Visualización y búsqueda de ontologías

Caso de uso	Integración de ontologías de diferentes formatos estándar
Descripción	El usuario puede integrar ontologías de diferentes formatos estándar en la aplicación.
Actores	Usuario
Precondiciones	La aplicación está abierta y el usuario tiene acceso a las ontologías que desea integrar en un formato estándar compatible con la aplicación.
Postcondiciones	El usuario puede utilizar las ontologías integradas en la aplicación.
Flujo principal	1. El usuario navega hasta el repositorio Github oficial de la aplicación `https://github.com/ProyectoAether/Aether-Portal`. 2. El usuario añade la URI de la ontología en el fichero "ontologies.txt". 3. El usuario guardar los cambios del fichero "ontologies.txt". 4. El usuario crea una pull request y añade comentarios necesarios. 5. El usuario espera la respuesta de los administradores y mantenedores de la aplicación. 6. Los administradores aprueban los cambios, actualiza la aplicación y finalmente la despliega.

Flujos alternativos	A. El usuario introduce una URI mal formada. *a*) La aplicación muestra un mensaje informando al usuario de que la URI es incorrecta. *b*) El usuario puede seleccionar un formato de archivo diferente para la ontología.
Excepciones	E. La aplicación experimenta problemas técnicos al procesar la ontología integrada. *a*) La aplicación muestra un mensaje informando al usuario del problema. *b*) El usuario puede intentar integrar la ontología nuevamente más tarde.

Tabla 6: Integración de ontologías de distintos formatos estándar

Caso de uso	Visualización de la estructura de las ontologías
Descripción	La aplicación debe permitir al usuario visualizar la jerarquía de clases de cada ontologías.
Actores	Usuario
Precondiciones	La ontología ha sido seleccionada y cargada correctamente en la aplicación.
Postcondiciones	El usuario puede visualizar la estructura de la ontología en la aplicación.
Flujo principal	1. El usuario selecciona una ontología para visualizar su estructura. 2. La aplicación muestra la estructura de la ontología, incluyendo los conceptos y relaciones que la componen.
Flujos alternativos	A. La ontología seleccionada no se carga correctamente en la aplicación. *a*) La aplicación muestra un mensaje de error informando al usuario del problema. *b*) El usuario puede intentar cargar la ontología nuevamente. B. La ontología seleccionada no tiene estructura definida. *a*) La aplicación muestra un mensaje informando al usuario de que no hay estructura disponible para esta ontología.

Tabla 7: Visualización de la estructura de las ontologías

Caso de uso	Visualización de los datos de cada ontología.
Descripción	El usuario puede visualizar datos relevantes de cada ontología como nombre de las clases, propiedades y los individuos. Así como, el título de la ontología y sus autores
Actores	Usuario
Precondiciones	La aplicación está abierta, funciona correctamente y tiene ontologías integradas.
Postcondiciones	El usuario puede visualizar los datos de las ontologías integradas en la aplicación.
Flujo principal	1. El usuario navega hasta la pestaña "Ontologies". 2. El usuario realiza una búsqueda, tecleando en la caja de texto. 3. El usuario navega al enlace de una ontología, en la sección de resultados de búsqueda. 4. El usuario visualiza los datos de la ontolgía.
Flujos alternativos	A. El usuario accede los datos desde la pestaña "Search". *a*) El usuario navega hasta la pestaña "Search". *b*) El usuario realiza una búsqueda, tecleando en la caja de texto. *c*) El usuario navega al enlace de una ontología, en la sección de resultados de búsqueda. *d*) El usuario visualiza los datos de la ontolgía.

Tabla 8: Visualización de los datos de cada ontología

Caso de uso	Presentación de una documentación detallada de cómo subir una ontología.
Descripción	El usuario puede conseguir más información sobre cómo integrar una ontología personalizada en la propia aplicación.
Actores	Usuario
Precondiciones	La aplicación está abierta y funciona correctamente.
Postcondiciones	El usuario puede visualizar la documentación.
Flujo principal	1. El usuario navega hasta la pestaña “Submit”. 2. El usuario puede leer documentación sobre cómo integrar ontologías a la aplicación.

Tabla 9: Documentación detallada de subida de ontología.

Caso de uso	Filtro y búsqueda de las ontologías integradas.
Descripción	El usuario puede obtener el nombre, descripción de la ontología a partir de una búsqueda.
Actores	Usuario
Precondiciones	La aplicación está abierta, funciona correctamente y tiene ontologías integradas.
Postcondiciones	El usuario puede obtener las ontologías resultantes de la búsqueda o el filtro.
Flujo principal	1. El usuario navega hasta la pestaña "Ontologies". 2. El usuario introduce texto en la barra de búsqueda de la aplicación. 3. El usuario obtiene las ontologías resultantes de la búsqueda en la sección de resultados de búsqueda.

Tabla 10: Filtro y búsqueda de las ontologías integradas.

Caso de uso	Filtro y búsqueda de clases, individuos y propiedades de las ontologías integradas.
Descripción	El usuario puede conseguir el nombre de clases, individuos y propiedades, y la ontología a la que pertenecen.
Actores	Usuario
Precondiciones	La aplicación está abierta, funciona correctamente y tiene ontologías integradas.
Postcondiciones	El usuario puede obtener las clases, individuos y propiedades resultantes de la búsqueda o el filtro.
Flujo principal	1. El usuario navega hasta la pestaña "Search". 2. El usuario introduce texto en la barra de búsqueda de la aplicación. 3. El usuario obtiene las clases, individuos y propiedades resultantes de la búsqueda en la sección de resultados de búsqueda.

Tabla 11: Filtro y búsqueda de clases, propiedades e individuos.

Caso de uso	Redirección a la documentación oficial.
Descripción	El usuario puede acceder a la documentación oficial de cada individuo, propiedad, clase y ontología.
Actores	Usuario
Precondiciones	La aplicación está abierta, funciona correctamente y tiene ontologías integradas.
Postcondiciones	El usuario puede acceder a la documentación oficial de las clases, individuos y propiedades.
Flujo principal	1. El usuario navega hasta la pestaña "Search". 2. El usuario introduce texto en la barra de búsqueda de la aplicación. 3. El usuario obtiene las clases, individuos y propiedades resultantes de la búsqueda en la sección de resultados de búsqueda. 4. El usuario puede acceder a la documentación oficial haciendo click izquierdo sobre el icono de documentación, a la derecha del enlace.

Tabla 12: Redirección a la documentación oficial.

3.5. Diseño de sistema

El diseño de sistemas es un proceso que se enfoca en la creación de un modelo del sistema que se va a desarrollar. En el diseño de sistemas se definen los componentes del sistema, sus funciones y relaciones, así como también se determina cómo se van a implementar los requisitos funcionales y no funcionales del sistema.

En la Figuras 3 y 4 se muestra el flujo de trabajo, diseñado para la creación y compilación del módulo del navegador, y el despligue en Github pages.

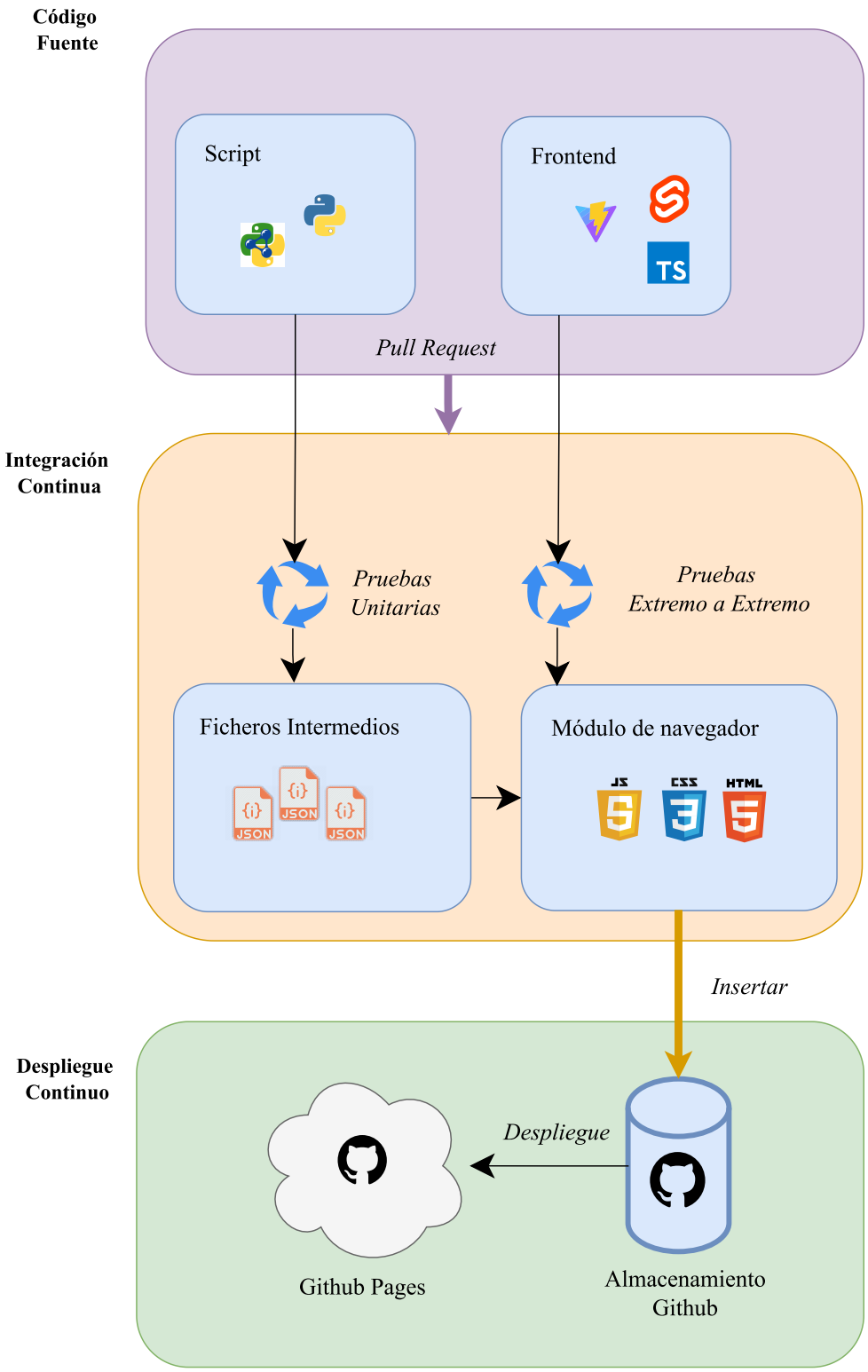

Figura 3: Diseño del flujo de trabajo. El flujo consta de principalmente 3 partes: Subida de código fuente, integración continua y despliegue continuo. Esto permite automatizar el proceso de pruebas, generación de ficheros intermedios, compilación a módulo de navegador y despliegue en la nube.

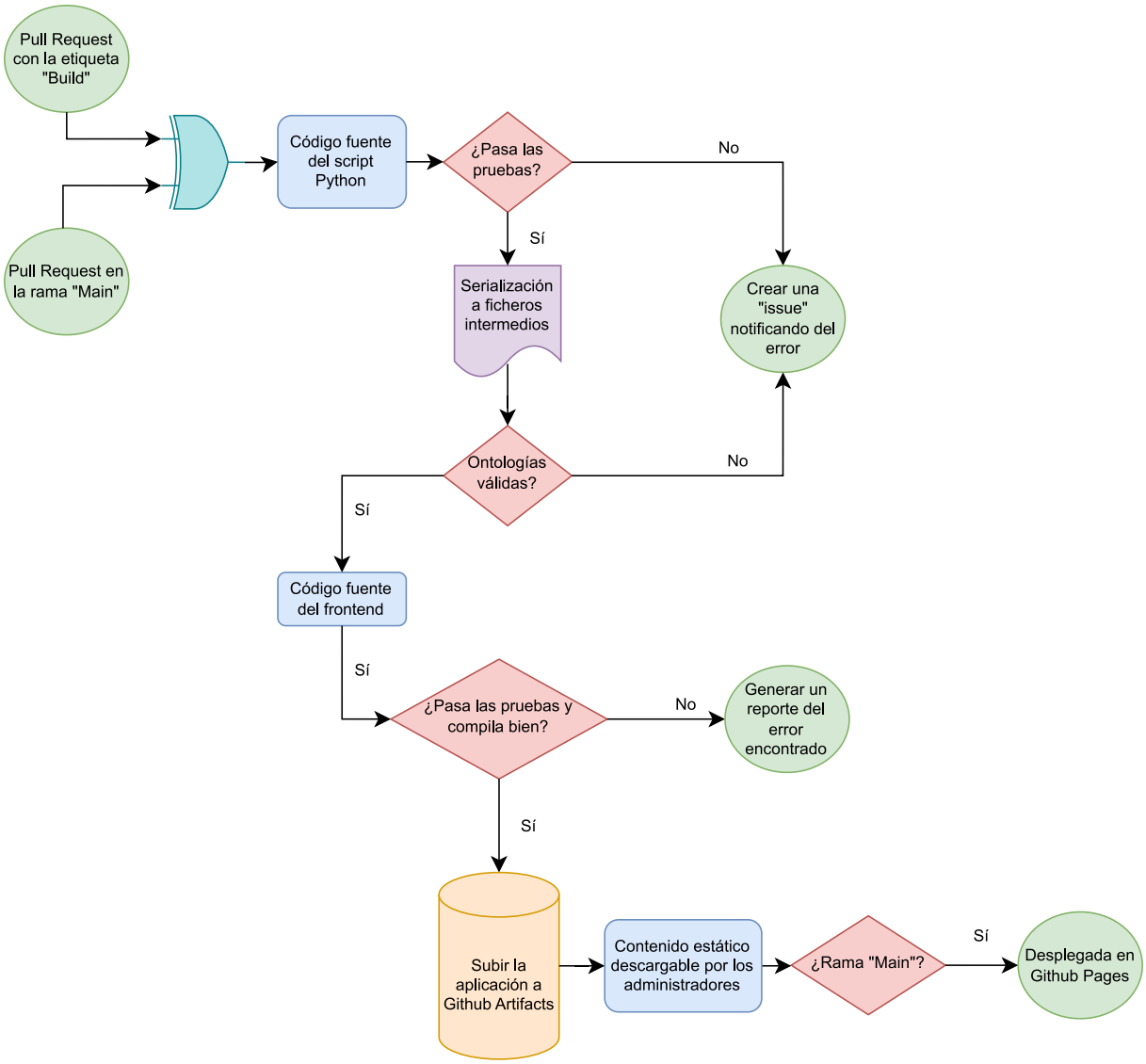

Figura 4: Diagrama de flujo. Los círculos corresponden a puntos de inicio y fin del flujo; rectángulos a pasos en el flujo; diamantes como punto de decisión; cilindros como almacenamiento no volátil de datos; y rectángulos con una línea curvilínea como pasos con producción de ficheros estáticos.

En primer lugar, es importante destacar que existe una clara separación entre la lógica de procesamiento y serialización de datos, y la lógica de muestreo y representación en el navegador, tal como se muestra en las Figuras 3 y 4. El proyecto consta de un script Python que utiliza la librería RDFLib para deserializar y serializar los datos de las ontologías, y de código en Typescript y Svelte para crear el frontend[16] de la aplicación web.

Para que la aplicación consuma los datos de ontologías, es necesario, en primer lugar,

[16]Frontend: Capa de presentación de una aplicación software.

ejecutar el script Python para comprobar la validez de las ontologías y generar archivos estáticos intermedios en formato JSON[17]. En segundo lugar, se requiere código fuente responsable del frontend de la aplicación para la compilación y creación de un módulo compuesto exclusivamente de HTML, Javascript y CSS[18] para su posterior presentación en el navegador web. Debido a que el navegador no admite nativamente Typescript y Svelte, es necesario utilizar una herramienta de compilación y empaquetado llamada Vite. Esta herramienta genera una versión minimizada de los módulos de Typescript y Svelte que se compilan en un módulo comprimido de HTML, Javascript y CSS.

Para agilizar el proceso previo, es recomendable automatizar el proceso. Para ello, se utiliza Github Actions para simplificar la compilación del código fuente y el despliegue automático del módulo estático a Github Pages. Además, se restringe la ejecución del flujo de trabajo de Github Actions solo en caso de una "pull request"[19], ofreciendo varias opciones de ejecución. Por ejemplo, si se proporciona una etiqueta de "build" en los comentarios de la pull request, se ejecutará solo el proceso de compilación del módulo del navegador. Sin embargo, si no se proporciona la etiqueta anterior y la pull request tiene lugar en la rama "main", el proceso también desplegará la aplicación en Github Pages.

Las principales razones de hacer de Github Pages en lugar de otros sitios de hosting, se debe a varias razones:

- **Servicio de hosting gratuito**: A diferencia de otros servicios como Clouflare[20], Netlify[21] o AWS Amplify[22]
- **Integración con Git**: Github Pages está integrado directamente con Git, lo que

[17]JSON: "Javascript Object Notation" es un formato de intercambio de datos ligero y fácil de leer y escribir. Se basa en una estructura de pares clave-valor que permite representar datos estructurados de manera sencilla y eficiente.

[18]CSS: "Cascading Style Sheets", es un lenguaje de diseño utilizado para describir cómo se presenta un documento HTML (o XML) en la pantalla, papel u otros medios de salida.

[19]Pull request: Es una solicitud que un desarrollador hace en un repositorio de código fuente para fusionar sus cambios en el proyecto principal. Cuando se trabaja en un proyecto de equipo, los desarrolladores suelen trabajar en ramas separadas del repositorio para evitar conflictos con otros desarrolladores que puedan estar trabajando en el mismo código. Cuando el desarrollador termina su trabajo y desea incorporar sus cambios en la rama principal del proyecto, envía una solicitud de extracción para que los revisores revisen y aprueben los cambios.

[20]Cloudflare: `https://www.cloudflare.com/`

[21]Netlify: `https://www.netlify.com/`

[22]AWS Amplify Hosting: `https://aws.amazon.com/amplify/hosting/`

facilita la publicación de sitios web a través de repositorios Github. Esto permite una gestión eficiente del versionado y la colaboración en el desarrollo de un sitio web.

- **Personalización del dominio**: Github Pages permite personalizar el dominio de un sitio web alojado, lo que significa que se puede usar un nombre de dominio personalizado en lugar de utilizar el dominio predeterminado de Github.
- **Seguridad**: Github Pages utiliza HTTPS de manera predeterminada, lo que garantiza que los datos que se intercambian entre el servidor y el cliente estén encriptados y sean seguros.
- **Escalabilidad**: Github Pages puede escalar automáticamente para manejar grandes volúmenes de tráfico. Además, como es un servicio en la nube, los sitios web alojados en Github Pages pueden ser accesibles desde cualquier lugar del mundo con conexión a Internet.

4. Implementación

4.1. Detalles de la implementación

En esta sección, se presenta el proceso de implementación utilizado para desarrollar la aplicación web basada en ontologías descrita en este trabajo. Se explicará a profundidad los desafíos y soluciones encontrados durante el proceso de implementación, prestando especial atención a las estructuras de datos y algoritmos empleados. En primer lugar, se ilustra una estrategia ingenua para filtrar los datos de las ontologías; así como una posterior optimización para filtrar mayores números de ontologías que se podrá añadir a la aplicación en un futuro; se comentará algunas desventajas de las tecnologías empleadas para el desarrollo del frontend; y finalmente, una descripción general de cómo funciona la estructura de datos elegida y su mayor ventaja con respecto la implementación con la librería estándar de Javascript.

Nociones básicas

Como se ha visto previamente en la sección 2.2, los datos de las ontologías no son más que tripletas de "Sujeto - Predicado - Objeto". Conocemos también que la fortaleza de las ontologías reside en la estandarización de la sintaxis de estos datos. De esta forma, es muy simple la extracción de estos datos teniendo en cuenta que:

1. Los tipos de cada sujeto se indica de la siguiente forma:

   ```
   <Sujeto> - rdf:type - <Tipo>
   ```

2. El conjunto de tipos que un sujeto puede tomar, asumiendo que vamos a tratar con datos de ontologías que emplea el vocabulario OWL, son las descritas previamente en la Tabla 1.
3. La jerarquía de clases de relación "Padre-Hijo" se podría construir de forma relativamente fácil, como se puede observar en el Fragmento 3:

```
<?xml version="1.0"?>
<rdf:RDF xmlns="https://www.example.com/ontologies/example.owl#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:base="https://www.example.com/ontologies/example.owl#">
<owl:Ontology rdf:about="https://www.example.com/ontologies/example.owl">
    </owl:Ontology>
    <owl:Class rdf:about="http://www.example.com/ontologies/example.owl#Bar"/>
    <owl:Class rdf:about="http://www.example.com/ontologies/example.owl#Baz"/>
    <owl:Class rdf:about="http://www.example.com/ontologies/example.owl#Foo"/>
    <owl:Class rdf:about="http://www.example.com/ontologies/example.owl#Qux"/>
    <owl:Class rdf:about="http://www.example.com/ontologies/example.owl#Baz">
        <rdfs:subClassOf rdf:resource="http://www.example.com/ontologies/example.owl#Foo"/>
    </owl:Class>
    <owl:Class rdf:about="http://www.example.com/ontologies/example.owl#Bar">
        <rdfs:subClassOf rdf:resource="http://www.example.com/ontologies/example.owl#Foo"/>
    </owl:Class>
    <owl:Class rdf:about="http://www.example.com/ontologies/example.owl#Bar">
        <rdfs:subClassOf rdf:resource="http://www.example.com/ontologies/example.owl#Qux"/>
    </owl:Class>
</rdf:RDF>
```

Fragmento de código 3: Ejemplo de jerarquía de clases

Aunque a primera vista el fragmento anterior puede parecer verboso, realmente sería equivalente, al menos en nuestro caso, al siguiente Fragmento 4:

```
Baz - rdf:type - owl:Class
Bar - rdf:type - owl:Class
Foo - rdf:type - owl:Class
Qux - rdf:type - owl:Class
Baz - owl:subClassOf - Foo
Bar - owl:subClassOf - Foo
Bar - owl:subClassOf - Qux
```

Fragmento de código 4: Ejemplo simplificado

Lo que es equivalente a la Figura 5.

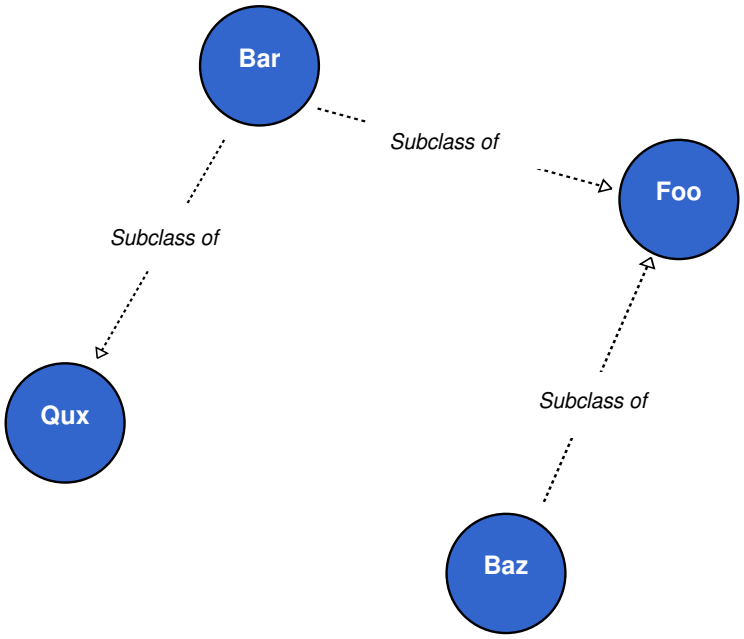

Figura 5: Ejemplo de jerarquía

Serialización a ficheros intermedios

Para entregar los datos de las ontologías al navegador, es necesario la selección de un formato de fichero. En este proyecto se ha hecho uso del formato JSON por el soporte nativo con el navegador, así como su amplio uso en la actualidad.

La generación de los ficheros intermedios es implementado en el script de Python, y sigue los siguientes pasos:

1. Deserialización de los datos de ontologías a una estructura de datos de grafos
2. Normalización de las distintas URIs.
3. Reconocimiento y extracción de las partes claves de los grafos, como el título y la URI.
4. Serialización a una colección de ficheros intermedios JSON.

Los ficheros que se generan son:

- **"index.json"**, cuyo clave tomará como valor el resultado de aplicar el algoritmo SHA256[23] a la URI de una ontología, y su valor será los otros datos relevantes como el valor de la URI, el título de la ontología, etc. De esta forma nos permitiría acceder a los metadatos de cada ontología con un tiempo de complejidad de $O(N)$
- **"stats.json"**, recoge estadísticas relevantes como el número de propiedades, de clases,

[23]SHA256: Función criptográfica unidireccional de hash.

y de ontologías.

- **"searchable.json"**, es el conjunto de datos al que la barra de búsqueda hará el filtro.
- **"namespaces.json"**, tomando clave la URI de la ontología y de valor su forma compactada, permitiendo un acceso de $O(N)$
- **"<sha256(uri)>.json"**, donde "sha256(uri)" es básicamente el resultado de aplicar SHA256 sobre la URI de cada ontología. Este proceso de hash se utiliza principalmente para eliminar caracteres extraños de la URI de una ontología. En este fichero, se almacenará el conjunto de tripletas que le corresponde a cada ontología.

Desafíos y soluciones

Una vez organizado y empleado de forma correcta las estructuras de datos, su consumo es trivial. La mayor complejidad reside en el filtro sobre el fichero "searchable.json", dado que el tamaño del fichero aumenta de forma lineal con respecto al número de ontologías. Por ello, aunque para una cantidad alrededor de 20 ontologías el rendimiento de la aplicación se mantiene casi intacta, al realizar una prueba con 100 ontologías, el sitio web dejaría de funcionar por el exceso de copias innecesarias en memoria y el rendimiento pobre del algoritmo de filtro.

Para solucionar el problema anterior, es necesario la implementación de un algoritmo eficiente de búsqueda en el lado del navegador. En otras situaciones, esto no sería un problema dado que tendrías una API con el que interactuar a través del protocolo HTTP, y esta API será responsable de realizar la lectura, escritura, modificación y eliminación de los datos en la base de datos. En nuestro caso, por razones de los requisitos del sistema implementado, no tenemos una base de datos SQL o NoSQL por detrás, al que podríamos delegar la tarea de optimizaciones de consulta.

Tampoco nos serviría emplear Sparql [4] como motor/lenguaje de consulta de datos ontologías. Uno de los requisitos relevantes de la aplicación es el rendimiento y esta es la razón por la que la página es totalmente estática, es decir, no tiene una base de datos por detrás para proporcionar datos dinámicamente. Sparql está optimizado especialmente en el lado del backend[24]. En el lado del cliente, sin embargo, la implementación actual es la de crear en memoria el modelo del conjunto de datos para poder consultar a partir

[24]Backend: Capa de lógica de la aplicación encargada de proporcionar una interfaz con el almacenamiento no volátil (bases de datos) y garantizar seguridad.

de Sparql. Además las consultas complejas requieren de varios “JOINs”, es decir, juntar varios conjuntos de datos por lo que es costoso en rendimiento. Cabe destacar, también que Sparql se centra más en el retorno correcto de los datos de las ontologías que en el rendimiento. Un claro ejemplo de su pobre rendimiento es la falta de un sistema de “vista” como los de las bases de datos relacionales, estas “vistas” proporciona, como su nombre indica, una representación sobre el conjunto de datos y es capaz de guardar los resultados de consultas complejas para poder retornar resultados en menos tiempo.

Por ello, en nuestra situación, no nos queda otra opción que realizar la búsqueda en la memoria, y en la mayoría de los casos aprovechar al máximo la caché de la CPU. Por las razones anteriores es imprescindible encontrar una estructura de datos muy eficiente en términos de memoria y velocidad.

La estructura de datos elegido para solucionar el problema de rendimiento y memoria es “Apache Arrow”, centrándonos en aprovechar la eficiencia de las arquitecturas x86 [42]. Este formato nos permite:

1. Se basa en un diseño de columnas contiguas, que permite vectorización aprovechar operaciones SIMD (Single Instruction, Multiple Data), incluidos en los procesadores modernos.
2. Abstracción de coste cero.
3. Estandarización del formato de datos, para posibles futuras incorporaciones de bases de datos.
4. Lectura y escritura de formatos de ficheros como CSV y Apache Parquet.
5. Análisis y procesamiento de consultas en memoria.

En la siguiente Figura 6 se muestra como se dispone la memoria en Apache Arrow:

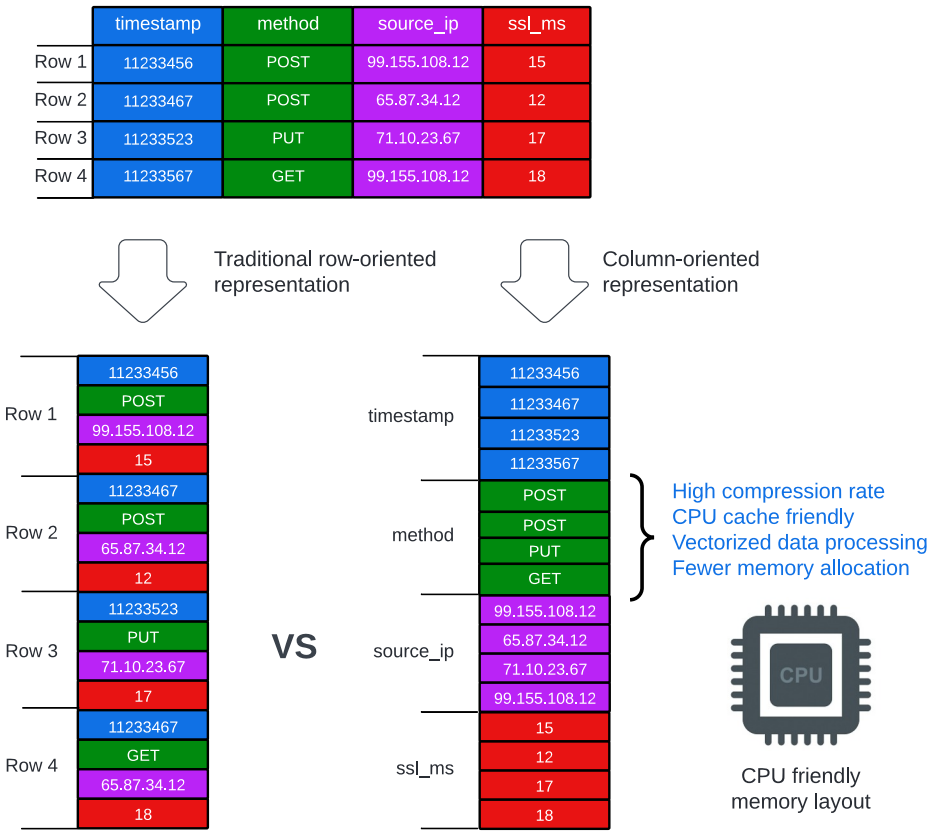

Figura 6: Disposición de memoria en Apache Arrow.

Lo que ocurre en la Figura 6 es el alineamiento de regiones contiguos de memoria de 64 bytes o 512 bits[25]. Esto permite cargar un bloque entero de 64 bytes de instrucciones en un registro SIMD[26], consiguiendo paralelismo a nivel de datos. Además, proporciona código más eficiente en cuanto a caché de CPU, puesto que al cargar bloques de datos de 64 bytes como línea de caché en la CPU, se asegura que cada unidad de procesamiento de la caché sea solamente un bloque de datos, en lugar de múltiple bloques que se extiende a múltiples líneas de caché.

4.2. Pruebas

La prueba software es una parte crucial del desarrollo y mantenimiento de aplicaciones, y asegura que funciona en situaciones diferentes. En el presente trabajo la mayor dificultad que nos encontramos es que disponemos de dos aplicaciones diferentes (script y frontend),

[25]64 bytes es el tamaño de registros SIMD más empleado en arquitecturas de x86.

[26]SIMD: "Single Instruction Multiple Data", instrucción máquina accesible a través del conjunto de instrucciones de la arquitectura (ISA), permitiendo realizar múltiples operaciones simultáneamente.

y además son en lenguajes de programación distintos.

Para solventar, el problema anterior necesitamos dos marcos de trabajo de pruebas distintos para cada parte de la aplicación. Utilizaremos Pytest para las pruebas unitarias [27] en el script, y Playwright para las pruebas de extremo a extremo [28] en el frontend.

Pytest es una popular biblioteca de pruebas para Python que proporciona una gran cantidad de funcionalidades para escribir y ejecutar pruebas. Por otro lado, Playwright es una herramienta que permite realizar pruebas end-to-end en aplicaciones web en diferentes navegadores y sistemas operativos. Con Playwright, se pueden automatizar tareas como completar formularios, hacer clic en botones y navegar por diferentes páginas. Además, Playwright es compatible con múltiples lenguajes de programación, incluyendo Typescript.

También es de gran importancia no integrar código con lógica de negocio incorrecto al repositorio Github. Podríamos añadir en la documentación dedicado al desarrollador que se asegure de que pase todas las pruebas antes de subir su código al repositorio. Sin embargo, este proceso es repetitivo y siempre hay riesgo de que el desarrollador se olvide de comprobar el código. Por ello, las pruebas también serán integradas al flujo de trabajo descrito previamente en la sección 3.5, permitiendo la automatización y asegurando el correcto funcionamiento de la integración continua.

Agregado a lo anterior, se utiliza Docker para poder realizar las pruebas en un entorno aislado. El fichero "Dockerfile" contiene todas las instrucciones para el procesado de las ontologías y el despliegue de la aplicación en el local. Esto nos permitirá comprobar que funciona el despliegue antes de subir el código al repositorio Github. Se emplea una imagen[29] de "Nginx"[30] para poder servir el contenido estático del paquete de Javascript, HTML y CSS.

[27]Pruebas unitarias: Un tipo de prueba de software que se enfoca en comprobar el correcto funcionamiento de las unidades de código individualmente, evaluando el comportamiento de cada función o método de forma aislada.

[28]Pruebas de extremo a extremo: Un tipo de pruebas de software que tienen como objetivo verificar el correcto funcionamiento de una aplicación en su conjunto, desde el inicio hasta el final del flujo de trabajo, simulando interacciones de un usuario real.

[29]Imagen Docker: Plantilla que se utiliza para crear contenedores de Docker. Son archivos que contienen todos los elementos necesarios para ejecutar una aplicación, incluyendo el sistema operativo, las bibliotecas, los archivos de configuración y el código fuente de la aplicación.

[30]Nginx: Un servidor web y proxy inverso de alto rendimiento y código abierto que se utiliza para manejar y enrutar solicitudes HTTP y HTTPS. En otras palabras, es un software que permite servir páginas web y aplicaciones en internet de manera rápida y eficiente.

5. Portal de Ontologías

En esta sección se presentan los elementos visuales del sitio web desarrollado, con el fin de proporcionar un tour de la página y explicar la funcionalidad que proporciona cada sección, botón, enlace, etc.

En primer lugar, se encuentra la página principal, tal como se muestra en la Figura 7. En esta sección se pueden visualizar estadísticas interesantes de la web, como el número de ontologías, número de clases y número de propiedades. Además, más abajo en la página, se encuentran enlaces a las ontologías más recientes, y al hacer clic sobre uno de estos enlaces, se redirige a la página con los detalles de cada ontología.

Figura 7: Página inicial.

A partir de esta primera página, se puede acceder a otros enlaces, ya sean internos o externos al sitio web en cuestión. En la cima de la página, se encuentran enlaces como "Search", "Ontologies" y "Submit", y al hacer clic izquierdo sobre ellos, se llevará a enlaces internos de la página web con funcionalidades distintas. Además, haciendo scroll hasta el final de cada página, se pueden encontrar los enlaces a la página oficial de Aether, la página del grupo de investigación Khaos Research, así como las redes sociales y contactos al grupo de investigación, tal como se muestra en la Figura 8.

Figura 8: Footer del portal.

Al entrar a la página de "Search", se muestran los 10 primeros resultados de la búsqueda. Esta búsqueda se realiza sobre las clases y propiedades de las ontologías, de tal forma que se pueden encontrar de forma fácil, rápida e intuitiva los elementos que el usuario desea descubrir. Al final de la vista, se presenta un elemento de paginación para evitar saturar la interfaz con posibles miles de resultados, permitiendo al usuario navegar al resultado que más le convenga. Además, se puede acceder tanto a la página interna de previsualización de los metadatos de la ontología como a la documentación externa de la ontología o elemento, tal como se puede apreciar en la Figura 9.

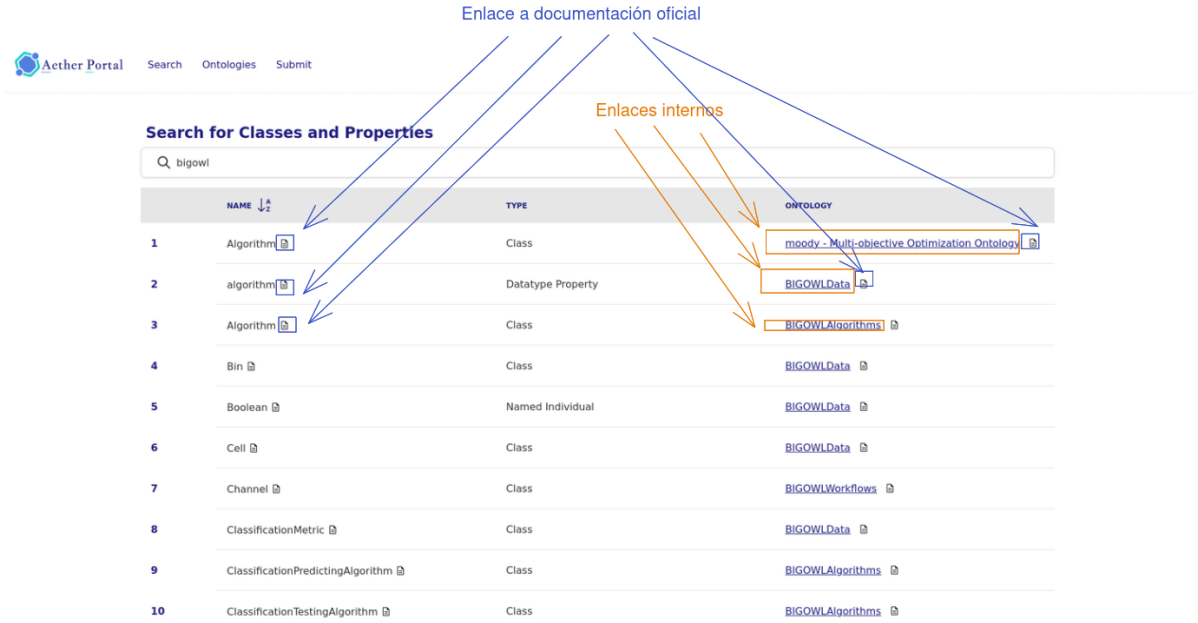

Figura 9: Vista de Search.

Al navegar por el enlace de "Ontologies", se presenta una interfaz parecida a la de "Search", mostrada en la Figura 10. Sin embargo, su funcionalidad es distinta y presenta leves diferencias con respecto a "Search". En primer lugar, la búsqueda solo se realiza sobre URI o títulos de cada ontología y, en segundo lugar, el resultado del filtro se presenta en forma de carta que muestra el título, la descripción y el logo de la ontología.

Estos resultados son enlaces que al hacer clic llevan al usuario a la página interna de previsualización de los metadatos de la ontología.

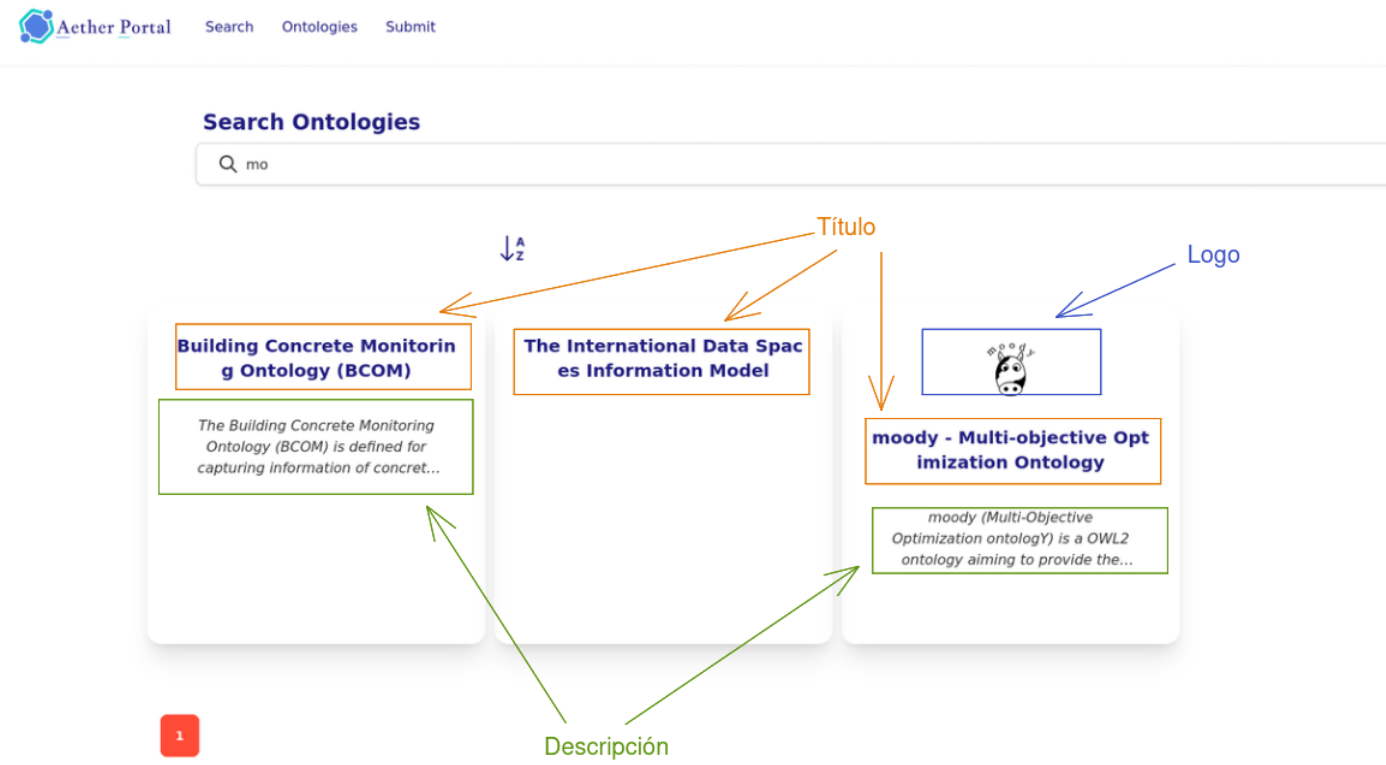

Figura 10: Vista de Ontologies.

Otro enlace de interés es “Submit”, en él se presentan las instrucciones a seguir para poder subir una ontología válida. En la Figura 11 muestra la evaluación de la validez de la ontología, y en la Figura 12 se proporciona paso a paso las instrucciones a seguir para integrar su propia ontología desarrollada.

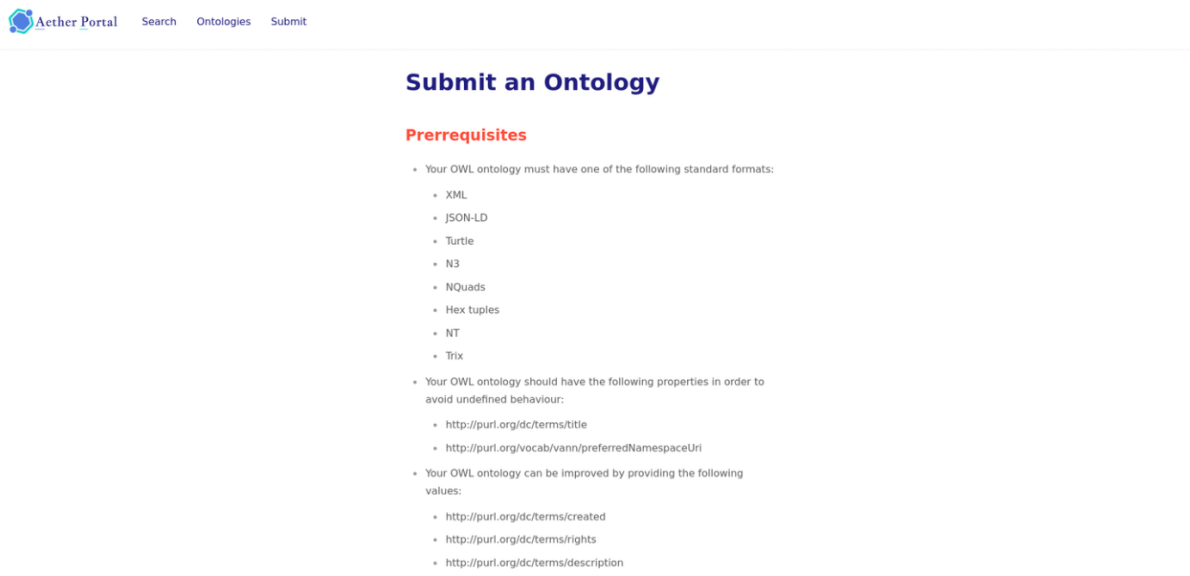

Figura 11: Prerrequisitos de subida de ontología.

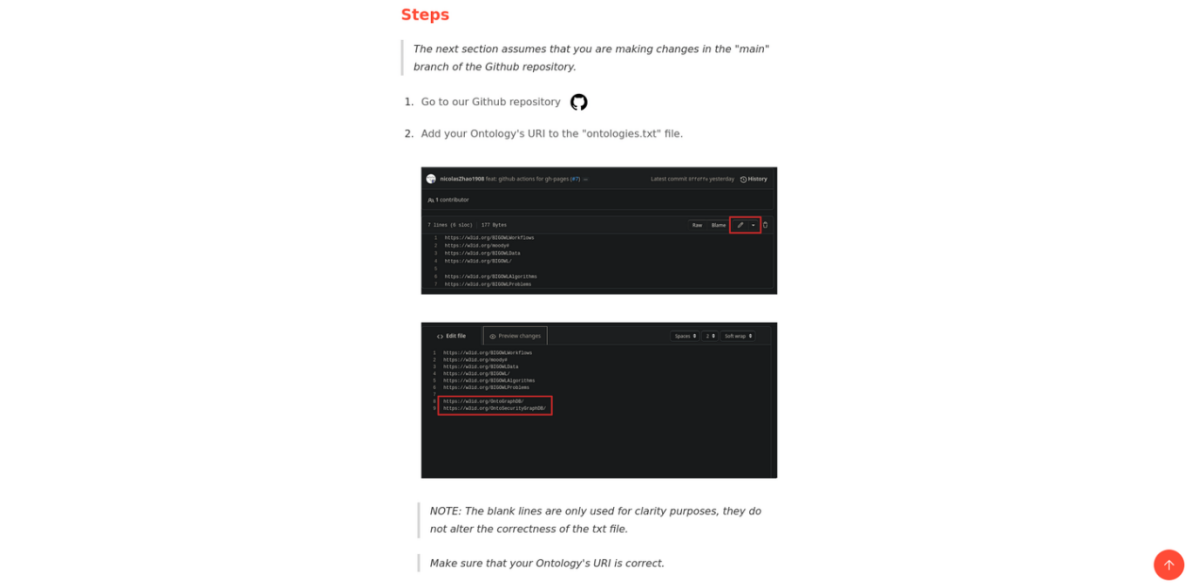

Figura 12: Instrucciones de subida de ontología.

El enlace más interesante a destacar es la de previsualización de los metadatos de la ontología, mostrada en la Figura 13. Al principio de la página, se expone en una tabla metadatos relevantes como el título, la URI y las importaciones a otras ontologías de la ontología elegida. Sin embargo, hay otras funcionalidades más destacadas al final de la página como se presenta en la Figura 14. Se puede visualizar tanto todos los elementos que integran la ontología, la jerarquía de la clases y las referencias a otras ontologías o "mappings" de la ontología seleccionada.

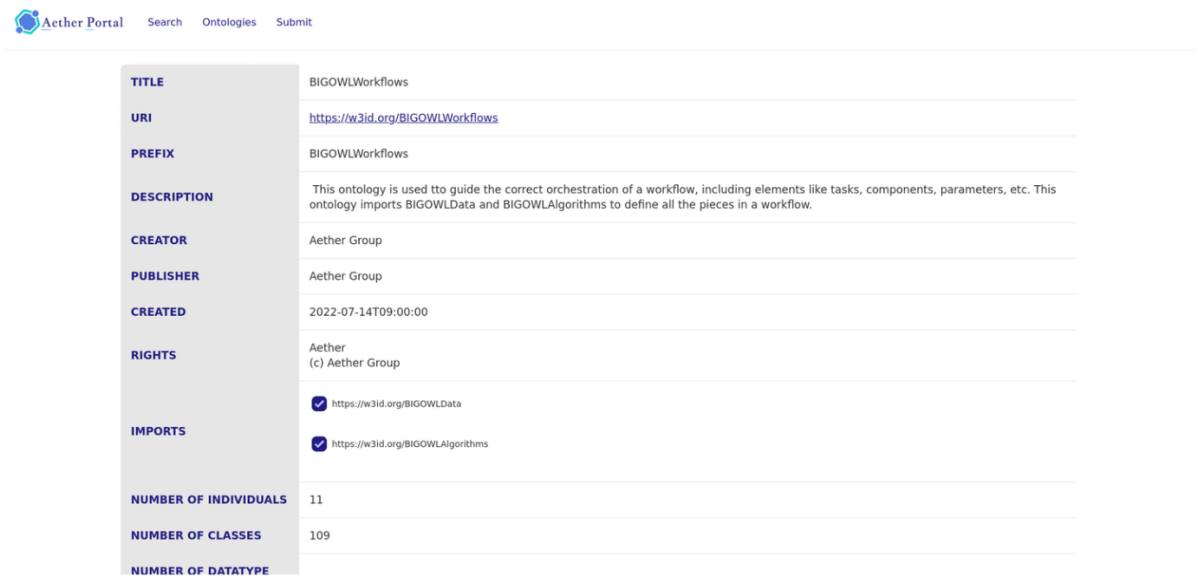

Figura 13: Tabla de metadatos de una ontología.

En la Figura 14, se muestra todos los elementos que componen la ontología seleccionada. Exponiendo 10 filas de la tabla y proporcionando un elemento de paginación para navegación el usuario. Además presenta enlaces para redirigir al usuario a la documentación oficial de cada elemento. En la Figura 15, se presenta la jerarquía de clases de la ontología como lista desplegables. Y finalmente en la Figura 16, se exhibe el número de referencias a ontologías externas.

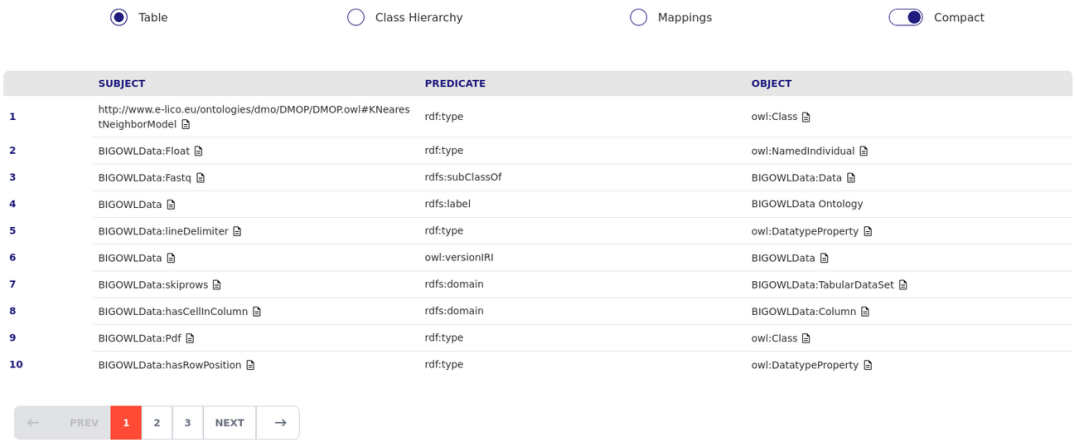

Figura 14: Componentes de una ontología.

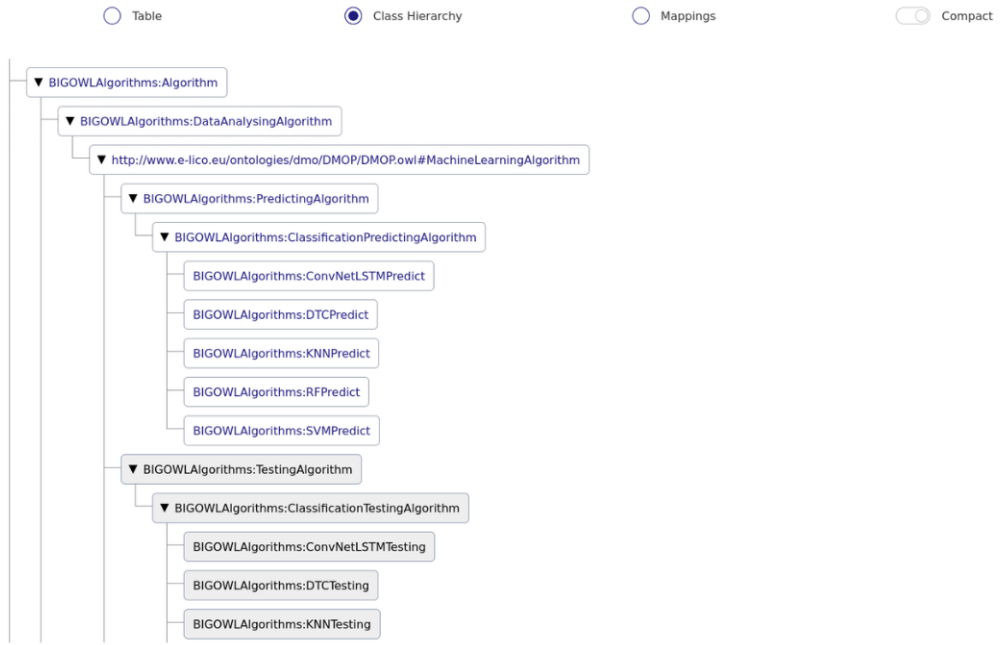

Figura 15: Jerarquía de clases de una ontología.

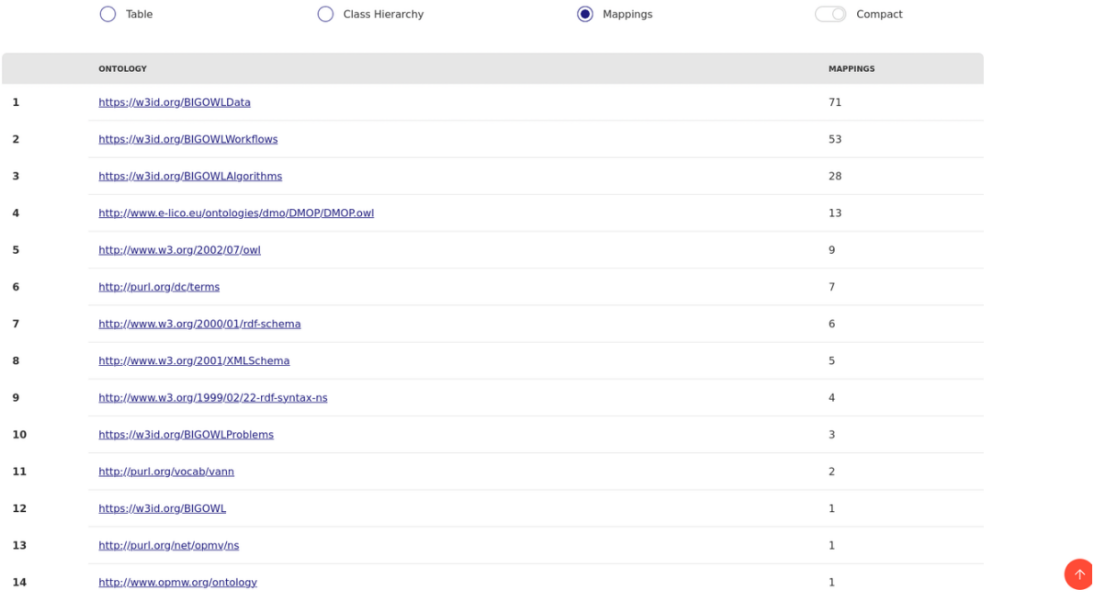

Figura 16: Mappings de una ontología.

Todos los fichero intermedios explicados en la Sección 4.1 se puede hallar bajo el endpoint "https://portal.aether.es/public/<nombre del fichero>", donde "<nombre de fichero>" corresponde al título que se le ha asignado a cada fichero. De esta forma, tanto el usuario y el administrador puede ver el conjunto de namespaces[31], el conjunto

[31] `https://portal.aether.es/public/namespaces.json`

de ontologías [32] y el conjunto de búsqueda que emplea la web [33], facilitando el acceso al conjunto de datos.

[32] https://portal.aether.es/public

[33] https://portal.aether.es/public/searchable.json

6. Conclusiones

A modo de conclusión, se ha logrado desarrollar con éxito un portal web de ontologías que cumple con los requisitos de rendimiento, accesibilidad y SEO. Durante la implementación, se superaron varios desafíos mediante la implementación de soluciones enfocadas en algoritmos y estructuras de datos relevantes actualmente, y se proporcionó información detallada sobre las tecnologías utilizadas y cómo se aseguró el correcto funcionamiento de la aplicación.

En cuanto a futuras mejoras, se podría considerar la implementación de nuevas funcionalidades para enriquecer aún más la experiencia del usuario, como la integración de herramientas de visualización de datos y la incorporación de funcionalidades de inteligencia artificial o machine learning para proporcionar una mejor clasificación y búsqueda de la información semántica. Además, se podría mejorar la velocidad de carga de la página, optimizando aún más los algoritmos y la estructura de datos utilizada.

Asimismo, se podría considerar la integración de una plataforma colaborativa que permita a los usuarios colaborar en la construcción y actualización de las ontologías, lo que podría enriquecer aún más la base de conocimientos disponible en la aplicación. También se podría incluir un sistema de recomendaciones basado en el perfil de usuario, un sistema de comentarios y valoraciones por parte de los usuarios, mejorar la accesibilidad de la aplicación y explorar la posibilidad de ofrecer la aplicación en múltiples idiomas.

En resumen, el portal web de ontologías desarrollado representa un éxito en el cumplimiento de los objetivos propuestos, pero aún queda un gran potencial para seguir mejorando y expandiendo su funcionalidad y utilidad.

Referencias

[1] Tim Berners-Lee, James Hendler y Ora Lassila. «The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities». En: *Scientific American* (mayo de 2001).

[2] Deborah L McGuinness, Frank Van Harmelen et al. «OWL web ontology language overview». En: *W3C recommendation* 10.10 (2004), página 2004.

[3] Dave Beckett y Brian McBride. «RDF/XML syntax specification (revised)». En: *W3C recommendation* 10.2.3 (2004).

[4] Steve Harris, Andy Seaborne y Eric Prud'hommeaux. *SPARQL 1.1 Query Language*. W3C Recommendation 21 March 2013. World Wide Web Consortium, 2013.

[5] Alistair Miles y Sean Bechhofer. «SKOS simple knowledge organization system reference». En: *W3C recommendation* (2009).

[6] Patricia L. Whetzel et al. «BioPortal: enhanced functionality via new Web services from the National Center for Biomedical Ontology to access and use ontologies in software applications». En: *Nucleic Acids Research* 39.suppl_2 (jun. de 2011), W541-W545. eprint: `https://academic.oup.com/nar/article-pdf/39/suppl\_2/W541/7631361/gkr469.pdf`.

[7] Clément Jonquet et al. «AgroPortal: A vocabulary and ontology repository for agronomy». En: *Computers and Electronics in Agriculture* 144 (2018), páginas 126-143.

[8] Xeni Kechagioglou et al. «EcoPortal: An Environment for FAIR Semantic Resources in the Ecological Domain». En: *Proceedings*. Volumen 1613. 2021, página 0073.

[9] J. Guo et al. «MedPortal: A biomedical ontology repository and platform focused on precision medicine». En: *Chinese Journal of Biomedical Engineering* 36 (oct. de 2017), páginas 557-564.

[10] Cristóbal Barba-González et al. «BIGOWL: Knowledge centered Big Data analytics». En: *Expert Systems with Applications* 115 (2019), páginas 543-556.

[11] Cristóbal Barba-González et al. «Injecting domain knowledge in multi-objective optimization problems: A semantic approach». En: *Computer Standards & Interfaces* 78 (2021), página 103546.

[12] Manuel Paneque, María del Mar Roldán-García y José García-Nieto. «e-LION: Data integration semantic model to enhance predictive analytics in e-Learning». En: *Expert Systems with Applications* 213 (2023), página 118892.

[13] María del Mar Roldán-García et al. «Ontology-driven approach for KPI metamodelling, selection and reasoning». En: *International Journal of Information Management* 58 (2021), página 102018.

[14] Natalya F. Noy et al. «BioPortal: ontologies and integrated data resources at the click of a mouse». En: *Nucleic Acids Research* 37.suppl_2 (mayo de 2009), W170-W173. eprint: `https://academic.oup.com/nar/article-pdf/37/suppl\_2/W170/3966326/gkp440.pdf`.

[15] Yousef Alfaifi. «Ontology Development Methodology: A Systematic Review and Case Study». En: *2022 2nd International Conference on Computing and Information Technology (ICCIT)*. 2022, páginas 446-450.

[16] SAIBS Arachchi e Indika Perera. «Continuous integration and continuous delivery pipeline automation for agile software project management». En: *2018 Moratuwa Engineering Research Conference (MERCon)*. IEEE. 2018, páginas 156-161.

[17] Ken Schwaber. «Scrum development process». En: *Business Object Design and Implementation: OOPSLA'95 Workshop Proceedings 16 October 1995, Austin, Texas*. Springer. 1997, páginas 117-134.

[18] John W Brackett. *Software requirements*. Informe técnico. CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST, 1990.

[19] M. Levlin, Annamari Soini y Dragos Truscan. «DOM benchmark comparison of the front-end JavaScript frameworks React, Angular, Vue, and Svelte». En: 2020.

[20] Wendy Hall y Thanassis Tiropanis. «Web evolution and Web science». En: *Computer Networks* 56.18 (2012), páginas 3859-3865.

[21] Fernando Almeida. «Concept and dimensions of web 4.0». En: *International journal of computers and technology* 16.7 (2017).

[22] Marshall Breeding. «Web 2.0? Let's get to Web 1.0 first.» En: *Computers in Libraries* 26.5 (2006), páginas 30-33.

[23] Ling Qian et al. «Cloud computing: An overview». En: *Cloud Computing: First International Conference, CloudCom 2009, Beijing, China, December 1-4, 2009. Proceedings 1*. Springer. 2009, páginas 626-631.

[24] Karen Rose, Scott Eldridge y Lyman Chapin. «The internet of things: An overview». En: *The internet society (ISOC)* 80 (2015), páginas 1-50.

[25] Pascal Hitzler. «A review of the semantic web field». En: *Communications of the ACM* 64.2 (2021), páginas 76-83.

[26] Thomas R Gruber. «Toward principles for the design of ontologies used for knowledge sharing?» En: *International journal of human-computer studies* 43.5-6 (1995), páginas 907-928.

[27] Soren Auer et al. «Dbpedia: A nucleus for a web of open data». En: *The Semantic Web: 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007+ ASWC 2007, Busan, Korea, November 11-15, 2007. Proceedings*. Springer. 2007, páginas 722-735.

[28] Wikipedia. *Wikipedia*. PediaPress, 2004.

[29] Google Official Blog. *Introducing the Knowledge Graph: things, not strings*. 2012. URL: `https://blog.google/products/search/introducing-knowledge-graph-things-not/`.

[30] Tao Fan y Hao Wang. «Research of Chinese intangible cultural heritage knowledge graph construction and attribute value extraction with graph attention network». En: *Information Processing & Management* 59.1 (2022), página 102753.

[31] Zhiheng Huang, Wei Xu y Kai Yu. «Bidirectional LSTM-CRF Models for Sequence Tagging». 9 de ago. de 2015. arXiv: `1508.01991`.

[32] Shuang Liu et al. «An Intelligent Question Answering System of the Liao Dynasty Based on Knowledge Graph». En: *International Journal of Computational Intelligence Systems* 14.1 (1 de dic. de 2021).

[33] Shuang Liu et al. «Preliminary Study on the Knowledge Graph Construction of Chinese Ancient History and Culture». En: *Information (Switzerland)* 11.4 (1 de abr. de 2020).

[34] Petar Ristoski y Heiko Paulheim. «Semantic Web in Data Mining and Knowledge Discovery: A Comprehensive Survey». En: *Journal of Web Semantics* 36 (1 de ene. de 2016), páginas 1-22.

[35] Yuanzhe Yao et al. «An Ontology-Based Artificial Intelligence Model for Medicine Side-Effect Prediction: Taking Traditional Chinese Medicine as an Example». En: (2019).

[36] Bryar A. Hassan y Tarik A. Rashid. «Artificial Intelligence Algorithms for Natural Language Processing and the Semantic Web Ontology Learning». 31 de ago. de 2021. arXiv: 2108.13772.

[37] Ganesh Ramanathan et al. «Semantic Knowledge for Autonomous Smart Farming». En: *IFAC-PapersOnLine* 55.32 (2022), páginas 217-222.

[38] Heitor Magaldi Linhares et al. «FeedEfficiencyService: An Architecture for the Comparison of Data from Multiple Studies Related to Dairy Cattle Feed Efficiency Indices». En: *Information Processing in Agriculture* 9.3 (1 de sep. de 2022), páginas 378-396.

[39] Sai Sree Laya Chukkapalli et al. «Ontology Driven AI and Access Control Systems for Smart Fisheries». En: *SAT-CPS 2021 - Proceedings of the 2021 ACM Workshop on Secure and Trustworthy Cyber-Physical Systems* (28 de abr. de 2021), páginas 59-68.

[40] Julien Grosjean et al. «An Approach to Compare Bio-Ontologies Portals». En: (31 de ago. de 2014).

[41] Universidad de Málaga Grupo de Investigación Khaos Research. *El proyecto nacional 'Aether', punto de encuentro de cinco de los principales grupos de I+D en análisis de datos y Big Data.* Informe técnico. 2022.

[42] Daniel Kusswurm. *Modern X86 Assembly Language Programming.* Springer, 2014.

A. Informes Google Lighthouse

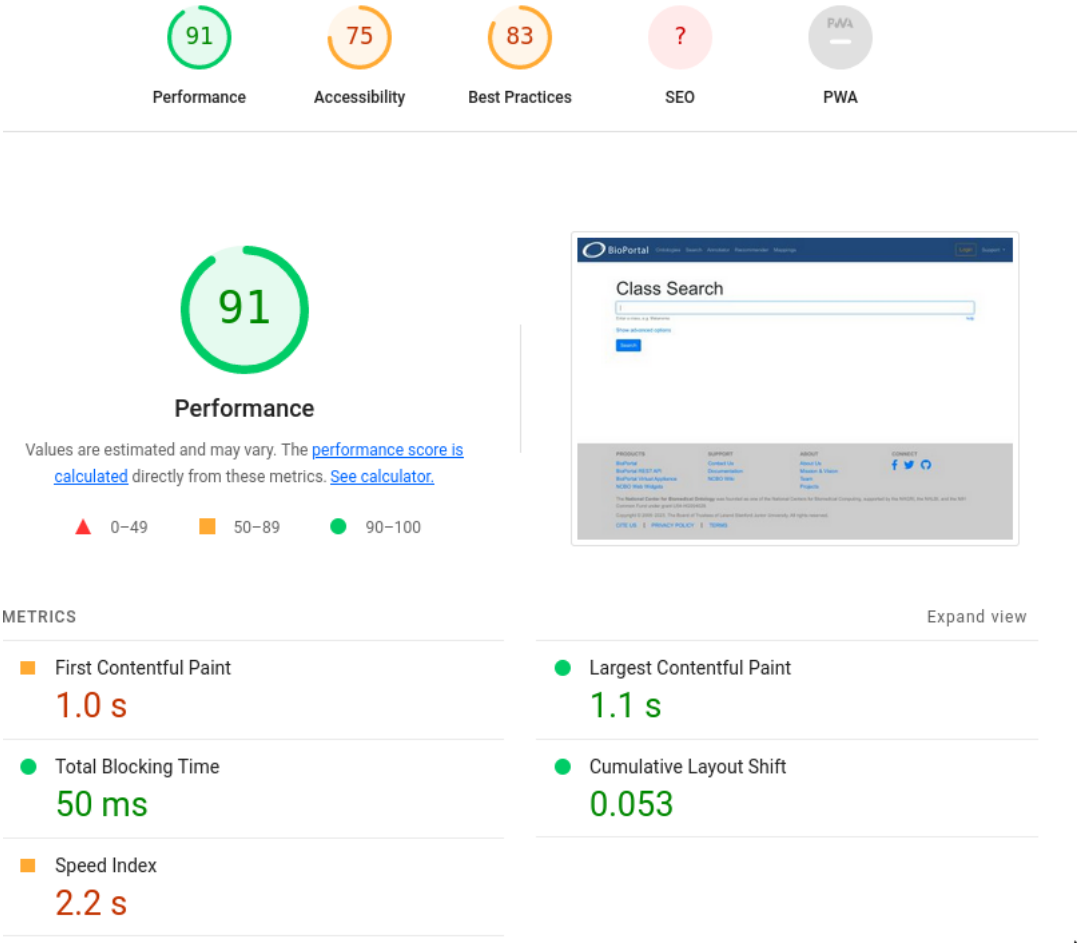

Figura 17: Informe de BioPortal. Fuente.

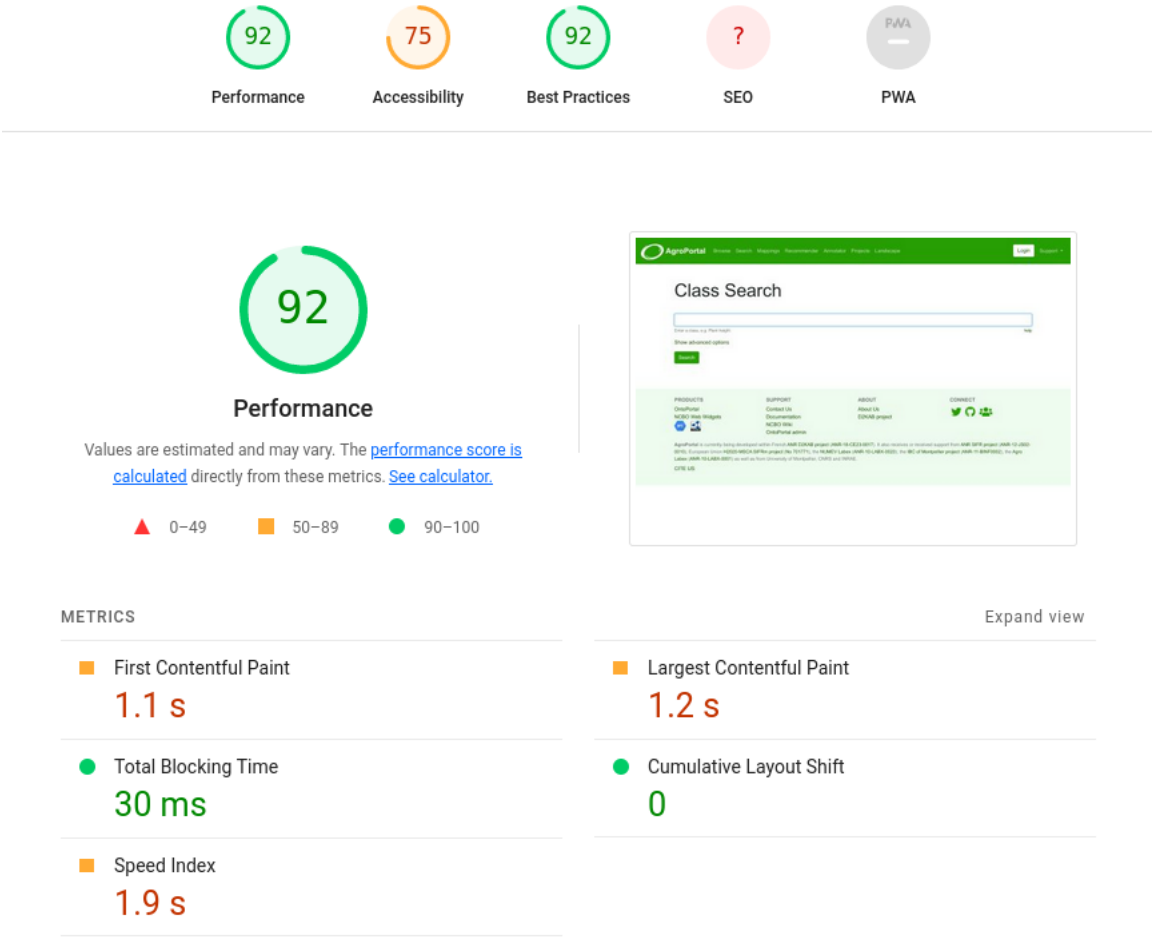

Figura 18: Informe de AgroPortal. Fuente.

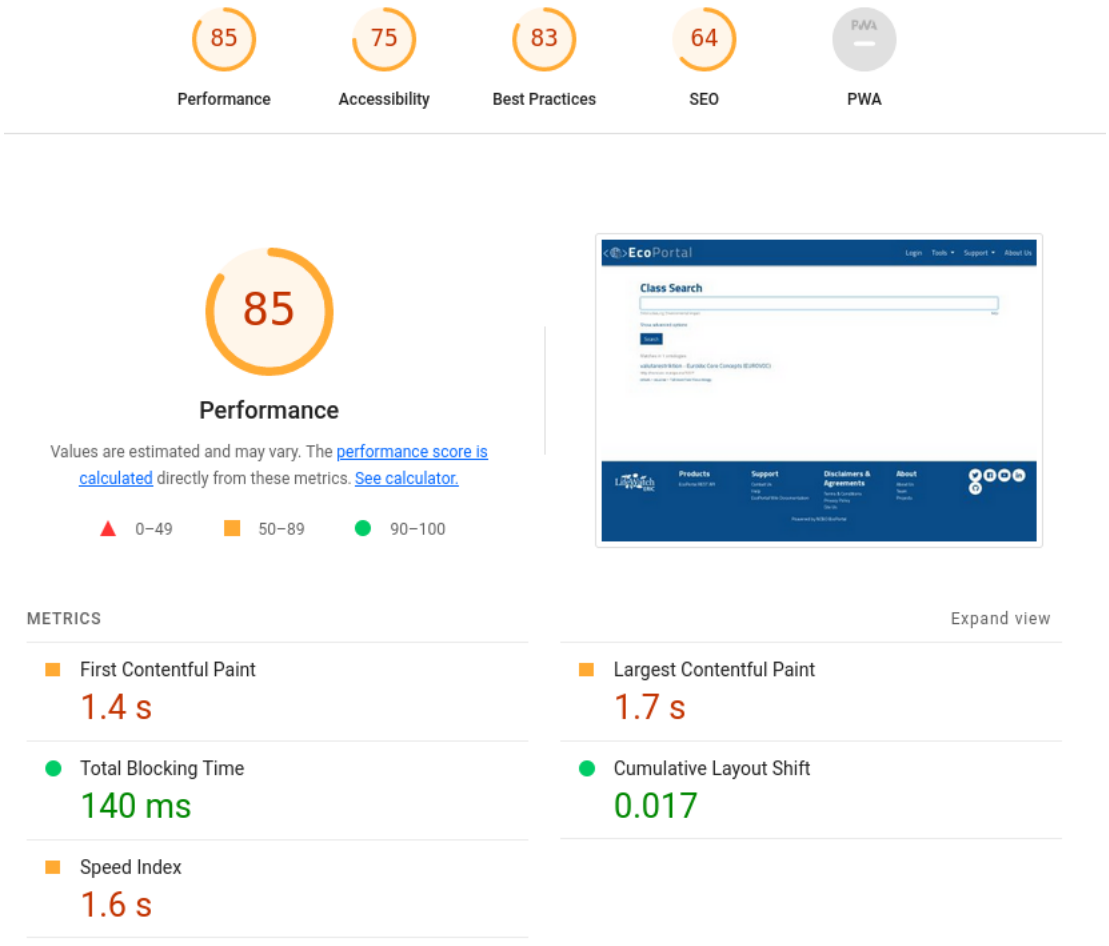

Figura 19: Informe de EcoPortal. Fuente.

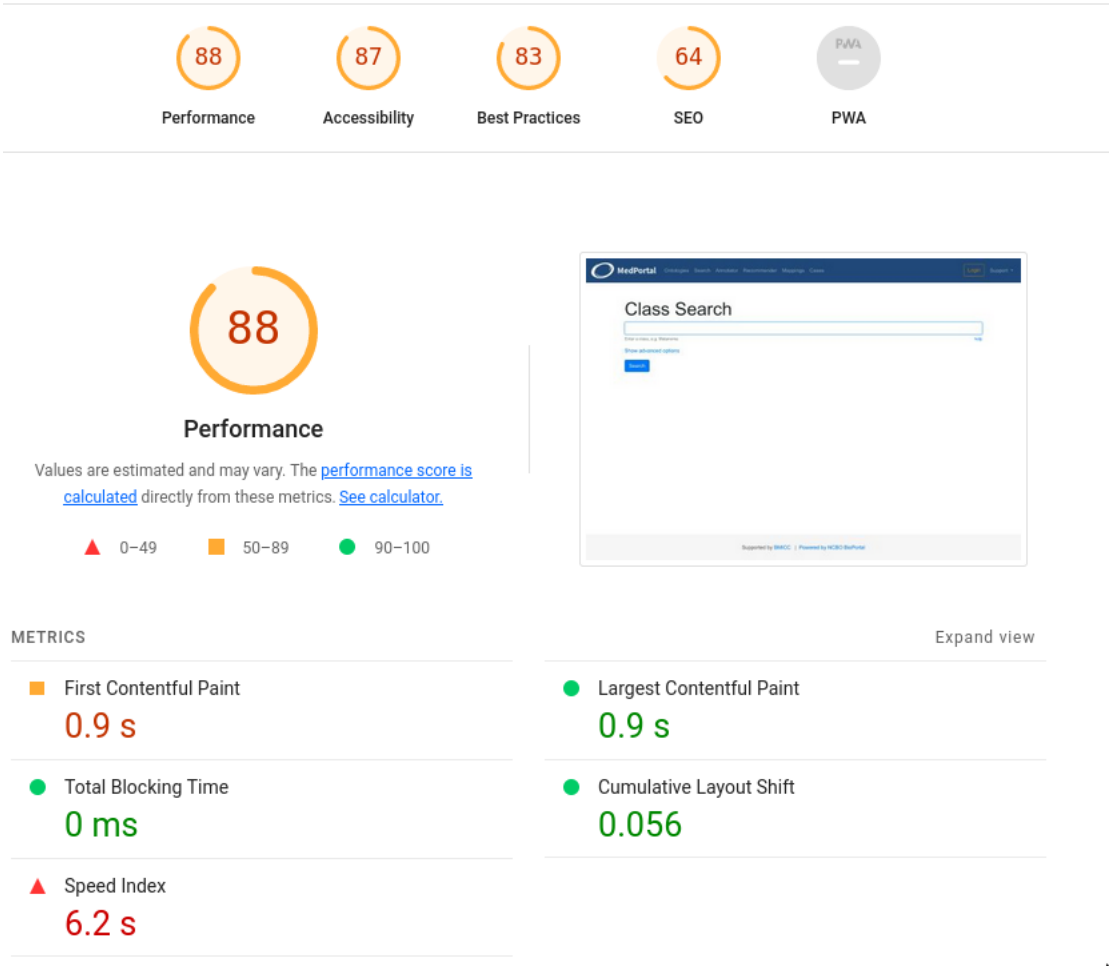

Figura 20: Informe de MedPortal. Fuente.

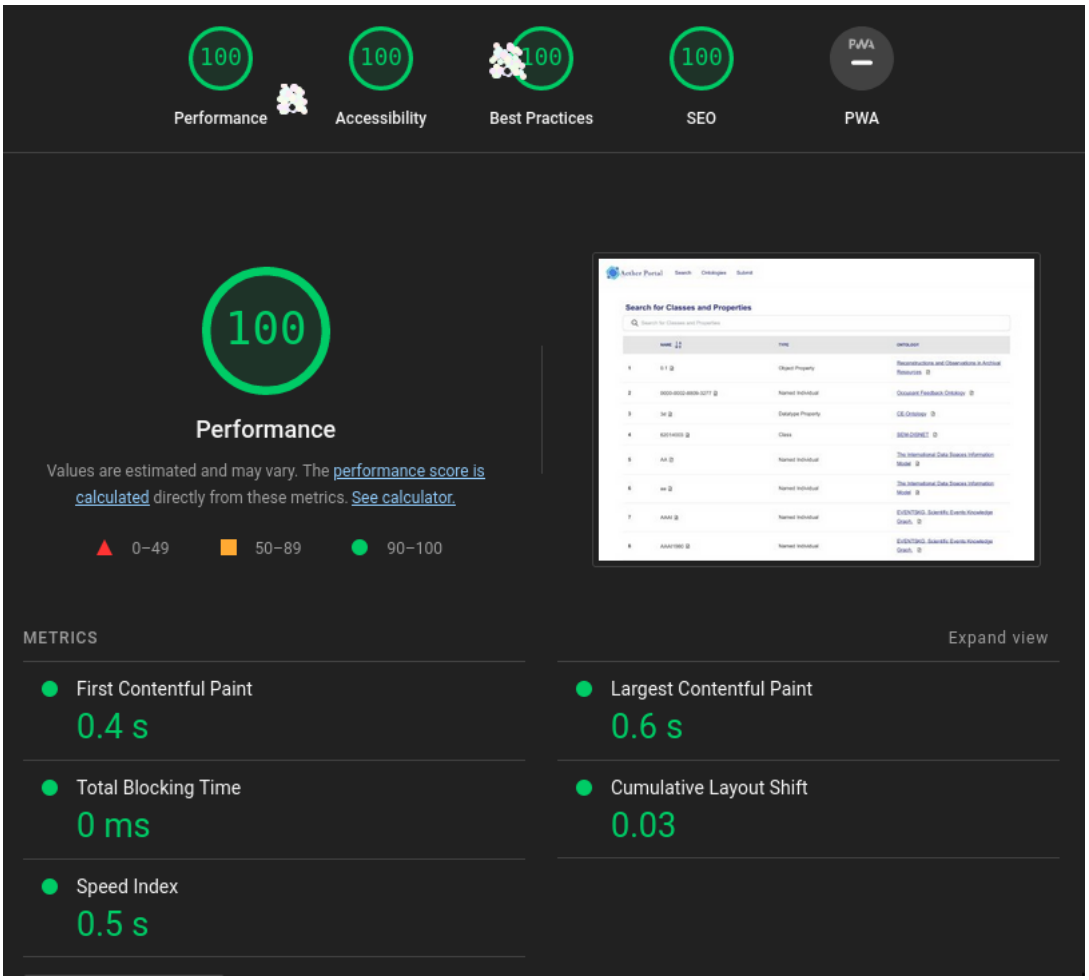

Figura 21: Informe de AetherPortal. Fuente.

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S. DE INGENIERÍA INFORMÁTICA

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga