Desarrollo del Portal e Interfaz para el Proyecto TeresIA

E1-Infraestructura: Despliegue y Uso de Ontoportal

4 de agosto de 2025

[Resumen Ejecutivo 3](#_Toc204074958)

[Objetivos y Alcance del Hito 3](#_Toc204074959)

[Descripción de la Infraestructura Base 3](#_Toc204074960)

[Proceso de Despliegue 3](#_Toc204074961)

[Validación Inicial 4](#_Toc204074962)

[Plan para Próximos Hitos 4](#_Toc204074963)

[Conclusiones 4](#_Toc204074964)

# 

# Resumen Ejecutivo

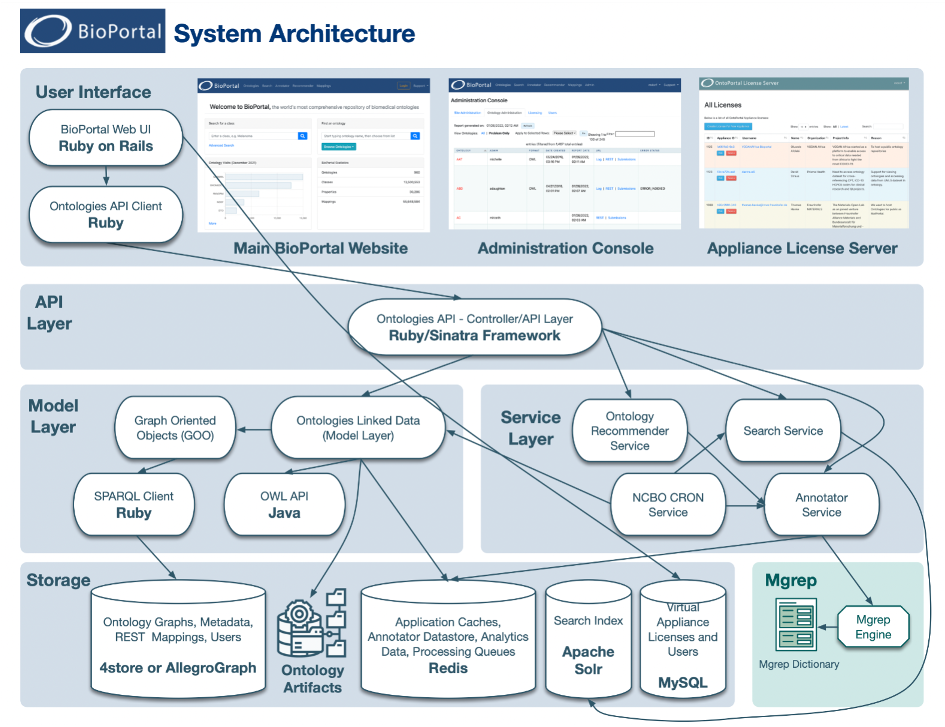
Este documento describe el despliegue inicial de la infraestructura del proyecto TeresIA, centrada en la instalación y configuración de OntoPortal como núcleo funcional. El objetivo de este hito es establecer la base tecnológica necesaria para el metabuscador, el catálogo de terminologías y los futuros servicios de inteligencia artificial. En esta fase se han preparado los entornos de desarrollo y producción, así como la integración inicial con una versión existente de OntoPortal.

# Objetivos y Alcance del Hito

Este hito tiene como objetivo principal el despliegue de la infraestructura base necesaria para el portal TeresIA. Incluye:

* Configuración de la arquitectura técnica básica (entorno de producción y desarrollo).
* Puesta en marcha de OntoPortal adaptado a los requisitos de TeresIA.
* Despliegue de los entornos en servidores del CSIC con pipelines de CI/CD.

## Descripción de la Infraestructura Base



OntoPortal se ha configurado como el núcleo funcional del sistema, proporcionando capacidades de metabúsqueda, gestión y publicación de ontologías. La infraestructura de OntoPortal inicialmente incluye:

* Una interfaz web en Ruby on Rails, que se comunica con la API a través de un cliente API en Ruby.
* La API, que gestiona los endpoints y realiza las acciones solicitadas por peticiones HTTP (creación de usuarios, obtención de ontologías, etc.).
* La capa de modelo (Linked Data), que utiliza la API para realizar operaciones en las bases de datos.

Inicialmente, los servicios con los que cuenta OntoPortal son los siguientes:

* **Redis**: Base de datos en memoria para caché.
* **Solr**: Motor de búsqueda para indexación.
* **Mgrep**: Servicio de anotación y búsqueda.
* Bases de datos RDF. Se pueden utilizar dos opciones:
  + **4store**: Base de datos de grafos RDF.
  + **AllegroGraph**: Base de datos de grafos RDF.

Sin embargo, en el alcance de este proyecto se prevén cambios respecto a los servicios usados, de forma que se implementarán e integrarán los siguientes servicios en fases siguientes:

* Data Lake híbrido con AllegroGraph/Neo4J.
* Motor de búsqueda en Elasticsearch.

# Proceso de Despliegue

## Inicialización del OntoPortal

Como primer paso de este desarrollo, se ha comenzado por tratar de levantar una instancia del OntoPortal en un entorno local. Para ello, se ha hecho un fork de los repositorios principales de OntoPortal en la organización de Github de TeresIA:

* [ontoportal\_ui](https://github.com/proyectoTeresIA/ontoportal_ui): Una aplicación en Ruby on Rails que implementa las vistas de la interfaz de usuario.
* [ontoportal\_api](https://github.com/proyectoTeresIA/ontoportal_api): API REST con los endpoints necesarios para acceder a las funcionalidades del OntoPortal.

Cabe destacar que estos repositorios tienen como dependencias otros repositorios de OntoPortal en forma de gemas de Ruby, pero para esta primera inicialización se ha dejado que utilicen los repositorios originales sin hacer ningún *fork* o modificación.

### Inicialización de la API

Tras clonar el repositorio de la API al que hemos hecho *fork*, debemos seguir los pasos descritos en la [documentación oficial](https://ontoportal.github.io/documentation/docs/developers-docs/development-environment/backend_environment/" \l "run-the-code-locally) para instalar y configurar correctamente el repositorio. Sin embargo, para poder levantarlo correctamente en el entorno local, se han añadido cambios al *docker-compose.yml* de forma que los puertos de Redis y 4store quedaran expuestos, además de cambiar y mejorar la gestión de variables de entorno, necesarias para el funcionamiento de la aplicación y de crear y configurar un docker-compose específico para el entorno de producción.

En el entorno de desarrollo tenemos dos opciones: un perfil de 4store y otro de AllegroGraph. Podemos levantar cada uno según el perfil que especifiquemos en el comando de docker compose:

$ docker compose –profile [4store | agraph] up -d

Por otro lado, podemos también levantar el entorno de producción, especificando el fichero docker-compose adecuado:

$ docker compose –f docker-compose.production.yml up -d

En ambos casos hay que especificar las variables de entorno necesarias en *.env* y *.env.production* respectivamente.

Una vez levantado podemos acceder a la API a través de nuestro navegador en localhost:9393. Sin embargo, para la mayoría de peticiones HTTP de la API será necesario incluir una API key, la cual se obtiene creando un usuario. Dado que el endpoint para la creación de usuarios también está protegido por una API Key, se ha creado un script *create\_admin\_user.rb* que crea un usuario con el rol “Administrador” programáticamente y muestra la API Key que podremos emplear en las peticiones HTTP.

### Inicialización de la UI

De nuevo, para la inicialización de esta aplicación seguimos los pasos descritos en la [documentación oficial](https://ontoportal.github.io/documentation/docs/developers-docs/development-environment/frontend_environment/). En este caso también se ha tenido que cambiar el *docker-compose.yml* para establecer correctamente las redes y los puertos de forma que funcionase en este entorno local, se ha mejorado el manejo de variables de entorno y se han hecho modificaciones para la solución de algunos errores del código original.

También es importante añadir la API Key obtenida en el paso anterior al fichero .env para que la UI pueda hacer peticiones a la API correctamente. Con esta configuración, podremos ejecutar el siguiente comando para levantar el entorno de desarrollo y acceder a la interfaz de usuario en localhost:3000:

$ docker compose up -d

De nuevo, para entrar en el entorno de producción debemos especificar el fichero correcto:

$ docker compose –f docker-compose.production.yml up -d

Aunque en este caso, cabe mencionar que el entorno de producción está configurado para forzar SSL, de modo que no podremos acceder mediante localhost simplemente y hace falta una instancia Nginx con HTTPS.

### Repositorio unificado

Se ha dejado preparado [un repositorio](https://github.com/proyectoTeresIA/portal) que incluye como submódulos la UI y la API con un *docker-compose.yml* que comprende todos los servicios de la aplicación, de forma que, siguiendo las instrucciones en el *README.md* se pueda desplegar toda la aplicación de forma más rápida y sencilla.

En este fichero de docker compose, se han dejado definidos también los servicios de Elasticsearch y Neo4J, con la idea de integrarlo en OntoPortal en próximas fases del desarrollo. Sin embargo, se han definido con el perfil “not\_integrated”, para que no se despliegue innecesariamente con el resto de servicios.

# Validación Inicial

Se realizaron pruebas básicas de funcionamiento tanto en el entorno de desarrollo como en el de producción:

* Verificación de acceso básico al portal y a todas sus pestañas.
* Validación de endpoints REST básicos.
* Comprobación del correcto registro de usuarios.

Incidencias abiertas: ninguna crítica en esta fase.

# Plan para Próximos Hitos

En el Hito 2 se desplegará la primera versión funcional del catálogo de terminologías y se integrará Neo4J y Elasticsearch con OntoPortal. También se habilitará la ingesta de terminologías en formatos RDF.

# Conclusiones

La infraestructura base de TeresIA ha quedado operativa tras este primer hito. El despliegue de OntoPortal y la configuración de los entornos de desarrollo proporcionan una base sólida para el desarrollo incremental del proyecto. No se han identificado riesgos críticos, aunque se recomienda monitorizar la aplicación bajo diferentes cargas en producción.