



Trabajo Fin de Grado

Sistema de Crawler enfocado al descubrimiento de páginas web que tienen contenido similar al de una base de datos

Crawler system focused on the Discovery of web pages that have similar content to that of a database

Autor

Pablo Jordán Lucia

Director

Francisco Javier López-Pellicer

ESCUELA DE INGENIERÍA Y ARQUITECTURA

2022

Sistema de Crawler enfocado al descubrimiento de páginas web que tienen contenido similar al de una base de datos

RESUMEN

Debido a diversos motivos, como puede ser una limitación de la plataforma, la presencia de datos de baja calidad o una mala gestión, en la actualidad resulta complicado buscar información de calidad en portales web públicos. Generalmente, los buscadores de los portales (en ocasiones construidos sobre la tecnología de Google) no son capaces de categorizar las páginas correctamente. Además de esto, muchas veces tampoco presentan la posibilidad de filtrar la información por páginas que contengan formularios o diferentes tipos de ficheros. De hecho, incluso los resultados ofrecidos al usuario normalmente son bastante pobres y no concluyentes. El resultado de todo esto es que actualmente, mucha de la información que se ofrece en los portales públicos (y que terceros le dan valor), está oculta.

Este TFG busca ayudar a la resolución de este problema mediante el desarrollo de un sistema compuesto de una araña web o *crawler*, un conjunto de procesos encargados del tratamiento de la información y un buscador. El *crawler* (extendiendo su funcionalidad por defecto) recupera las páginas del portal, el sistema de tratamiento de la información clasifica y anota aquellas que tienen que ver con bases de datos o ficheros de datos, y el buscador permite buscar sobre los recursos que ha procesado el sistema anterior. De esta forma se consigue tener un portal donde los documentos que contienen “bases de datos” se pueden encontrar con más facilidad, es decir, se facilita el acceso y búsqueda de los datos, aspecto que juega un rol muy importante en la contemporánea sociedad de la información.

Este sistema se ha validado usando el sitio web oficial del Ministerio de Agricultura, Pesca y Alimentación facilitando a los usuarios poder localizar más de dos mil páginas clasificadas como entradas a bases de datos o tener un contenido similar entre las más de diez mil que contiene el portal.

Índice

1	INTRODUCCIÓN	5
1.1	CONTEXTO	5
1.2	MOTIVACIÓN	5
1.3	OBJETIVO DEL PROYECTO	6
1.4	ALCANCE DEL PROYECTO	6
1.5	ESTRUCTURA DEL DOCUMENTO	7
2	ANÁLISIS.....	8
2.1	EL PROBLEMA DEL MAPA.....	8
2.2	ANÁLISIS DEL SISTEMA A DESARROLLAR	9
2.2.1	<i>Sistema de extracción</i>	10
2.2.2	<i>Sistema de persistencia y procesamiento.....</i>	11
2.2.3	<i>Sistema de consulta</i>	12
2.3	REQUISITOS.....	13
2.4	CONCLUSIONES	14
3	DISEÑO.....	15
3.1	ARQUITECTURA DEL SISTEMA	15
3.2	SISTEMA DE EXTRACCIÓN	16
3.3	SISTEMA DE PERSISTENCIA Y PROCESAMIENTO	17
3.4	SISTEMA DE CONSULTA.....	18
3.5	CONCLUSIONES	19
4	IMPLEMENTACIÓN	20
4.1	SISTEMA DE EXTRACCIÓN	20
4.2	SISTEMA DE PERSISTENCIA Y PROCESAMIENTO	21
4.3	SISTEMA DE CONSULTA.....	23
5	PRUEBAS	25
5.1	RESULTADOS OBTENIDOS	25
5.2	PROCESO DE PRUEBAS.....	27
6	METODOLOGÍA Y PLANIFICACIÓN	29
6.1	METODOLOGÍA DE TRABAJO	29
6.2	PLANIFICACIÓN	30
6.3	DEDICACIÓN.....	31
6.4	HERRAMIENTAS DE GESTIÓN UTILIZADAS	32
7	CONCLUSIONES	34
7.1	RESULTADO DEL PROYECTO	34
7.2	LECCIONES APRENDIDAS	34
7.3	FUTURAS MEJORAS.....	34
7.4	VALORACIÓN PERSONAL DEL PROYECTO	35
8	BIBLIOGRAFÍA.....	36
ANEXO A ANÁLISIS DETALLADO	38	
A.1	EVIDENCIAS DEL PROBLEMA DEL MAPA	38
A.2	FUNCIONAMIENTO Y ESTRUCTURA DE APACHE NUTCH.....	42
A.3	SISTEMA DE PERSISTENCIA	44
A.4	VISUALIZACIÓN DE LA INFORMACIÓN.....	46
A.5	REQUISITOS DEL SISTEMA	47
ANEXO B DISEÑO DETALLADO	50	
B.1	ARQUITECTURA DEL SISTEMA	50
B.2	MAPA DE NAVEGACIÓN DE LA APLICACIÓN WEB	51
B.3	PROTOTIPO DE LA APLICACIÓN WEB (SISTEMA DE CONSULTA)	51
B.4	DIAGRAMA DE SECUENCIA	56
ANEXO C IMPLEMENTACIÓN DEL SISTEMA DE EXTRACCIÓN.....	57	
C.1	CONFIGURACIÓN DE NUTCH	57
C.2	PLUGINS DE NUTCH UTILIZADOS	57
C.2.1	<i>Otras propiedades de configuración de Nutch</i>	60
C.2.2	<i>Modificación de un Plugin (index-basic).....</i>	61

ANEXO D IMPLEMENTACIÓN DEL SISTEMA DE PERSISTENCIA Y PROCESAMIENTO.....	63
D.1 IMPLEMENTACIÓN DE CONSULTAS DSL.....	63
D.2 IMPLEMENTACIÓN DE LOS SCRIPTS DE POST PROCESO	64
ANEXO E IMPLEMENTACIÓN DEL SISTEMA DE CONSULTA	69
E.1 MODELO DE DATOS	69
E.2 API DE GESTIÓN DE USUARIOS.....	70
E.3 APLICACIÓN WEB	74
ANEXO F PRUEBAS.....	81
F.1 RESULTADOS OBTENIDOS	81
F.2 PROCESO DE PRUEBAS.....	87
ANEXO G MANUAL DE USUARIO	92

1 Introducción

Este apartado presenta el contexto del problema. A continuación, se justifica porqué es importante su desarrollo, explicando en las siguientes secciones el objetivo y alcance del proyecto. Finalmente, se expone la estructura del documento.

1.1 Contexto

Con el objetivo de poder buscar y filtrar información deseada, relevante y de calidad, es necesario que ésta sea correctamente recuperada de la web y posteriormente indexada. Un *crawler* o araña web [1] es una herramienta que busca datos en Internet en base a una configuración concreta, analiza el contenido y guarda la información obtenida en un índice con el fin de que ésta pueda ser recuperada más adelante por motores de búsqueda. En castellano se denominan arañas web o rastreadores web. El funcionamiento de estos sistemas comienza a partir de una semilla o lista de semillas conocidas. Cada una de estas semillas es una URL. Conforme se vaya analizando y recolectando información de cada una de estas páginas web, se irá almacenando también de forma paralela sus enlaces externos, convirtiéndose éstos en las próximas URL en rastrearse. Durante el proceso de rastreo, estos sistemas utilizan la propia estructura del documento, los enlaces entrantes y salientes y las meta etiquetas para poder categorizar el contenido.

Los *crawlers* se pueden subdividir en dos principales tipos: *crawlers* universales [2] y *crawlers* personalizados [3]. El *crawler* universal es el tipo de *crawler* más antiguo y se encarga de rastrear la web de forma constante, sistemática y automática. Ejemplos de este tipo son aquellos utilizados por las grandes empresas como Google, Bing, Yahoo!, etc. Sin embargo, el *crawler* personalizado es un sistema de rastreo centrado en un tópico u objetivo en concreto, es decir, su objetivo final no se basa en recuperar e indexar todos los documentos accesibles. El *crawler* enfocado [4], el tipo de *crawler* interesante para este proyecto, es un claro ejemplo de *crawler* personalizado ya que se encarga de buscar e indexar únicamente un subconjunto de documentos. A diferencia del *crawler* universal, el *crawler* enfocado es un sistema que acota y precisa el funcionamiento de un *crawler* de propósito general.

1.2 Motivación

La búsqueda sin éxito de información de calidad en portales web públicos es un tema de actualidad [5]. En ocasiones, cuando se busca de forma más exhaustiva un determinado dato, fichero o incluso página dentro de un portal web, se obtienen resultados de lo más variados, pocas veces precisos o a veces inexistentes. Quizás muchas veces el problema reside en que la información es difícil de buscar (posiblemente debido a la incorrecta implementación de las consultas de búsqueda) o se trata de información no disponible. Sea como fuere, todo esto convierte la búsqueda de información en una tarea compleja y costosa para el usuario.

Es por ello por lo que para buscar, filtrar y visualizar información precisa es necesario un sistema de información específicamente construido para esto, es decir, un sistema que pueda recuperar los contenidos y que permita la realización de las búsquedas y filtrados sobre estos mismos, mostrando resultados unificados e integrados.

Además, la web está creciendo de manera exponencial en esta última década por lo que cada vez existirán más portales web con una mayor cantidad de información que siempre

será necesario buscar. De hecho, la Comisión Europea declara que los datos son esenciales para el crecimiento económico, innovación, creación de nuevos puestos de trabajo y en general, para el progreso social [6]. Para poder cumplir todos los objetivos anteriores, los datos deberían ser más abiertos, accesibles y disponibles, características que hoy en día no todos los datos cumplen.

Precisamente esto último es la motivación principal del proyecto. Si se consiguen desarrollar sistemas de información con buscadores y filtros de datos más ambiciosos, la disponibilidad y accesibilidad de estos se vería drásticamente mejorada. Por este principal motivo se considera de interés el desarrollo de este proyecto.

1.3 Objetivo del proyecto

El objetivo principal del proyecto es el desarrollo de un *crawler* enfocado que se encargue de descubrir páginas web que tengan un contenido asimilable al de una base de datos. Se habla de *crawler* enfocado ya que se aleja de la idea de intentar recuperar e indexar todo el contenido posible, centrándose en aquellos documentos que puedan contener “bases de datos”. Se entiende como “bases de datos” los formularios web, los documentos estructurados (hojas de cálculo, formatos de intercambio) y algunos documentos no estructurados que podrían contener datos (documentos en PDF, documentos en WORD). Además, el usuario final debe poder visualizar y probar el resultado del proceso gracias a la existencia de un portal web “ad-hoc”.

Este sistema deberá ser configurable para poder personalizar el proceso de recuperación de la información y que pueda adecuarse a las necesidades concretas. En segundo lugar, será necesario un proceso de análisis cuyo cometido sea enriquecer la información recuperada en el proceso anterior. Este proceso será clave para poder buscar y encontrar, a posteriori, toda la información deseada.

Finalmente, debe tener un portal web que permita al usuario llevar a cabo acciones de búsqueda, filtrado y edición sobre el conjunto de datos recuperados. Esto último se sustenta además sobre un control de accesos basado en roles mediante el cual solamente los usuarios autorizados tendrán la posibilidad de editar los contenidos. Se hace de esta manera para mantener consistencia en los datos almacenados y prevenir así que cualquier usuario pueda realizar cambios no autorizados en los mismos. No obstante, las tres partes no funcionan individualmente, sino que trabajan de forma cohesionada para lograr una visión integral como sistema.

El proyecto se probará sobre la web del Ministerio de Agricultura, Pesca y Alimentación (MAPA)¹.

1.4 Alcance del proyecto

Desarrollar este proyecto implica la configuración de un *crawler* con el objetivo de obtener los documentos del Ministerio, la configuración de un motor de búsqueda que permita almacenar y recuperar esta información, el desarrollo de un conjunto de procesos

¹ <https://www.mapa.gob.es/es/>

que se encarguen de completar la información almacenada dando soporte a funcionalidades de eliminación de información no relevante y de incorporación de información útil en base a propiedades del mismo documento o de sus enlaces tanto entrantes como salientes.

Además, implica también el desarrollo (junto al proceso de pruebas correspondiente) de una aplicación web completa con facetas que permita a usuarios registrados (e invitados) buscar, filtrar y editar la información extraída y tratada previamente. Esta aplicación web deberá dar soporte a funcionalidades de control de accesos de usuarios, filtrado de documentos atendiendo a diferentes campos, búsqueda de información por texto y edición de la información.

Cabe destacar también que el proyecto tendrá que ser capaz de encontrar documentos con un contenido similar al de una base de datos. Sin embargo, el hecho de verificar si el contenido de un documento no estructurado (documento PDF o similar) realmente contiene datos o es simplemente un fichero con otro tipo de información, se escapa del alcance de este proyecto. De hecho, investigar y verificar si este tipo de documentos contienen datos, sería una posible línea de investigación para un futuro trabajo.

1.5 Estructura del documento

Tras esta introducción en la que se expone brevemente el contexto tecnológico del proyecto, la motivación, los objetivos y alcance de este, se detalla el proceso de análisis, comenzando con el problema que tiene el portal web del Ministerio de Agricultura, Pesca y Alimentación. Acto seguido se analiza, teniendo en cuenta los problemas previos, el sistema a desarrollar extrayendo en la última etapa los requisitos finales. En el apartado de Diseño se muestra la arquitectura a alto nivel del sistema y posteriormente se aportan detalles de diseño de cada uno de los tres subsistemas existentes. El siguiente apartado expone los principales aspectos de la implementación del sistema, dividiendo siempre los contenidos en los tres subsistemas presentes.

Finalmente, se incluyen en esta memoria los resultados obtenidos junto al proceso de pruebas seguido. Se incluye también una sección de metodología y planificación que explica la forma en la que se ha abordado el proyecto, las etapas que se han necesitado para ello y las herramientas de gestión usadas. Como punto final del proyecto, se exponen las conclusiones extraídas. Toda la información contenida en esta memoria está sustentada en una bibliografía y anexos.

2 Análisis

El objetivo de la primera sección del análisis es poner de manifiesto los problemas que se han identificado en el portal del Ministerio de Agricultura, Pesca y Alimentación (MAPA). Estos problemas son generalizables a otros portales públicos [5]. Acto seguido, se expone la solución propuesta para poder superar los retos anteriores, llevando a cabo el análisis del sistema a desarrollar. Como resultado del proceso de análisis, se presentan en la siguiente sección los requisitos que debe tener el sistema final. En la última sección se ofrece una conclusión que reúne las ideas principales de este apartado.

2.1 El problema del MAPA

Como se ha mencionado en el apartado de Introducción, algunos portales web que existen hoy en día no ofrecen facilidades a la hora de buscar información. En el caso del portal web del MAPA², cuando se requiere una búsqueda de contenido un poco más ambiciosa, se pueden apreciar ciertos problemas, que se recogen en la Tabla 1.

Tabla 1. Representación de los problemas del MAPA.

Problema	Descripción del problema
Filtros	El portal no cuenta con ningún filtro. Los filtros son esenciales para poder acceder de forma rápida al resultado más preciso posible.
Buscador que delega el trabajo en Google	El buscador delega todo el trabajo en Google y los criterios de Google no son especialmente adecuados para búsquedas detalladas.
Resultados de la búsqueda	La información que se presenta como resultado de una búsqueda tiene un contenido muy pobre.

El primer problema avistado es la clara falta de filtros. El propio buscador del MAPA es un buscador únicamente por texto, sin ofrecer al usuario ningún tipo de ayuda en la búsqueda. Al tratarse de una web pública que trabaja con datos, se debería ofrecer un servicio algo más amplio en cuanto a la búsqueda de datos se refiere ya que el buscador de un sitio web es un punto importante que el portal tiene para poder mejorar la interacción con el usuario, haciendo que éste pueda encontrar en el proceso de búsqueda la información que desea. En el ámbito de este problema habría muchos filtros interesantes que se podrían incorporar, por ejemplo, filtros por formato (para recuperar, por ejemplo, solo los PDF), por tipo de documento (con el objetivo de recuperar únicamente los formularios), por categoría (para recuperar solamente los documentos relacionados con la Ganadería), por fecha, etc.

Otro problema encontrado es el hecho de que exista un buscador optimizado por Google. Si bien es cierto que este tipo de buscadores tienen una serie de ventajas como la de generar una cierta confianza y credibilidad en la audiencia [7], también cuentan con la principal desventaja de imprecisión a la hora de realizar búsquedas concretas, aspecto que también se ve reflejado a la hora de hacer una búsqueda en el propio buscador de Google.

El último problema reside en la información devuelta como resultado de una búsqueda. Esta información es algo pobre y debería contener más datos acerca del documento en

² <https://www.mapa.gob.es/es/buscador/>

cuestión. Por ejemplo, se podría añadir la fecha de publicación del documento, los ficheros que contiene, la cantidad de cada uno, etc. De esta forma, con un simple vistazo, se pueden ir descartando resultados hasta encontrar el deseado.

Con mayor detalle, en la Figura 1 se hace visible el primer problema detectado.

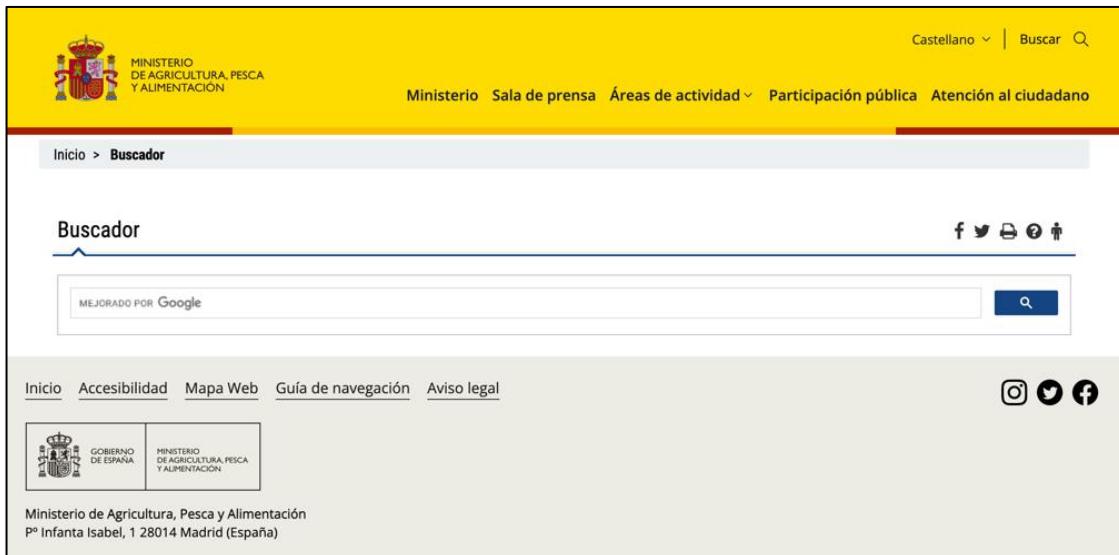


Figura 1. Buscador del portal web del MAPA. Fuente: <https://www.mapa.gob.es/es/ buscador/>.

Como se puede observar, el buscador del portal no contiene filtros, teniendo el usuario acceso únicamente a un buscador por texto. Como se ha comentado anteriormente, al tratarse de un portal web donde se trabaja con datos, y a colación de lo comentado también en la sección 1.2, éstos deberían ser fácilmente accesibles mediante filtros o facetas similares.

Un análisis más detallado con ejemplos concretos del resto de problemas puede consultarse en el Anexo A.1.

2.2 Análisis del sistema a desarrollar

Una vez expuestos los principales problemas del MAPA, esta sección tiene como objetivo mostrar la solución que mejor resuelve estos mismos. Si se desea construir una búsqueda más ambiciosa que la que actualmente posee el MAPA, se deben tener en posesión los documentos que forman parte del portal. Una vez se tienen esos documentos, éstos se deben poder buscar y procesar ya que, seguramente, la información que se extraiga sea insuficiente para realizar según qué consultas. Finalmente, el contenido debe estar disponible en un portal web de creación propia para que el usuario final pueda buscar y realizar los filtrados correspondientes.

En tal caso, el sistema final del proyecto requiere la extracción, persistencia, procesamiento y posterior búsqueda de información procedente del Ministerio de Agricultura, Pesca y Alimentación. Debido a la naturaleza modular del propio sistema, se puede dividir el mismo en tres grandes componentes o unidades autónomas que a posteriori deberán integrarse para trabajar conjuntamente y adquirir esa visión integral como sistema.

2.2.1 Sistema de extracción

Este sistema se encarga de extraer los documentos del MAPA, es decir, obtiene la “materia prima” por la que más adelante se consultará. Para llevar a cabo la extracción se necesita un software que pueda recuperar información de la web. Por este motivo, el sistema de extracción estará compuesto por un *crawler*, que además conforma el punto de entrada del sistema.

Un *crawler* tiene el siguiente funcionamiento:

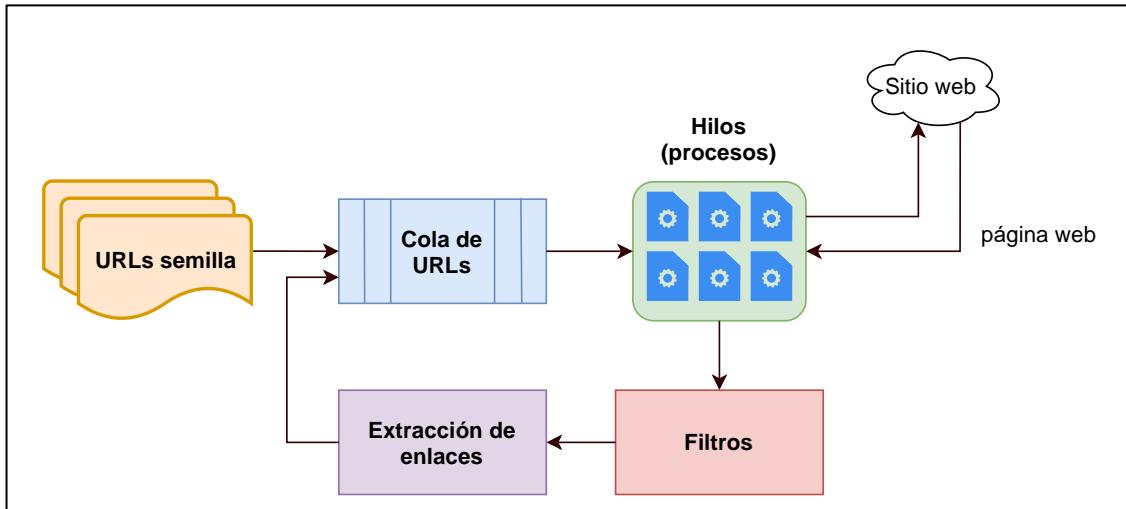


Figura 2. Funcionamiento de un crawler. Fuente: propia.

Desglosando las diferentes partes que lo constituyen, se observa en primer lugar el punto de entrada del sistema, las URL semilla, es decir, las URL a partir de las cuales se comenzará el proceso de extracción. Esas semillas se injectarán a una cola de URL. Existe un número fijo de hilos (procesos) que se encuentran continuamente leyendo la información de la cola de URL y haciendo la extracción de la información de la web. Cuando ya se tiene el documento, se aplican procesos de filtrado para ver si el documento se descarta (por ejemplo, porque está duplicado) o sigue el proceso. En la última etapa, los enlaces externos de ese documento se vuelven a encolar para que el grupo de procesos pueda continuar con el proceso de extracción [2].

Apache Nutch³, escrito en Java y Scrapy⁴, escrito en Python fueron los *crawlers* de código abierto que se analizaron para ser usados en esta etapa. Sopesando ventajas e inconvenientes de ambos, se considera más prometedor el *crawler* de Apache Nutch debido a su madurez y versatilidad. Otro motivo es que, al ser Java uno de los lenguajes de programación con los que más se ha trabajado durante los últimos años académicos, se espera que la curva de aprendizaje sea suave al principio.

³ <https://nutch.apache.org/>

⁴ <https://scrapy.org/>

Al tratarse de un *crawler*, a la hora de extraer información de la web, Nutch sigue el proceso mostrado previamente en la Figura 2 de esta misma sección. No obstante, se aporta un nivel extra de detalle acerca de su funcionamiento y estructura en el Anexo A.2.

Como ya se ha introducido brevemente en la sección 1.1, un *crawler* enfocado desempeña la misma función que un *crawler* universal, pero acotando el trabajo a un dominio o conjunto de dominios. Por ello, habrá que realizar una serie de modificaciones a su configuración por defecto para personalizar el proceso de crawl y orientarlo únicamente a la extracción y análisis de la información procedente del MAPA. Estos cambios incluyen la modificación de sus puntos de extensión y de sus ficheros de configuración.

2.2.2 Sistema de persistencia y procesamiento

En este punto ya se tiene extraído el contenido del portal web del MAPA, pero éste debe indexarse en algún soporte. Se necesita entonces un método de persistencia y posterior recuperación de la información. Por esto, este sistema debe contar con un subsistema de persistencia donde la información se pueda almacenar y consultar. Se plantea el uso de ElasticSearch⁵ ya que es de código abierto y puede caracterizarse como una base de datos documental [8]. El primer requerimiento es importante pero el segundo es crucial ya que, como la información final extraída por Nutch son documentos JSON, se necesita por fuerza una base de datos NoSQL (documental). Además, ElasticSearch cuenta con un mecanismo de consultas bastante potente que permite construir consultas con código que el propio sistema de almacenamiento ejecuta internamente (consultas de tipo “script”). Se valora también la opción de Apache Solr⁶ ya que cuenta con las dos características previas, pero se descarta debido a su desactualizada documentación y a la complejidad y verbosidad en sus consultas.

Se considera, por tanto, que ElasticSearch puede ser una solución muy prometedora puesto que, además, cuenta con una fuerte integración con Nutch y porque es parte del Elastic Stack⁷. Esto último es muy útil ya que, aparte del almacenamiento y recuperación de información, ofrece herramientas de visualización de los datos, aspecto que interesa mucho ahora y quizá también en fases posteriores del proyecto.

Entonces, la información procedente del MAPA se almacena en un índice de ElasticSearch para que sea rápidamente recuperable y accesible. Una vez la información se encuentra indexada, ElasticSearch ofrece un conjunto de API REST que se pueden consultar para operar sobre cualquier documento o conjunto de ellos. Se detalla este comportamiento en el Anexo A.3.

No obstante, es importante tener en cuenta que quizá la información descrita en los documentos que se han extraído no es suficiente para poder cubrir todas las búsquedas que se llevarán a cabo más adelante. Por ello, este sistema necesita también un subsistema de postproceso. El subsistema de persistencia simplemente alberga los documentos y responde a las peticiones de búsqueda, mientras que el de postproceso enriquece la información de los documentos que hay en el subsistema de persistencia.

⁵ <https://www.elastic.co/es/what-is/elasticsearch>

⁶ <https://solr.apache.org/>

⁷ <https://www.elastic.co/es/what-is/elk-stack>

El subsistema de postproceso estará compuesto por una serie de procesos que obtendrán los documentos almacenados en el índice del subsistema de persistencia consultando la API REST [9] de búsqueda que expone, modificarán la información que contienen (añadiendo propiedades relevantes y necesarias) si es necesario, y de nuevo, los volverán a almacenar. Para llevar a cabo esta tarea, se plantea la creación de una serie de procesos, *scripts*.

Las dos opciones que se barajan para desarrollar estos procesos son Python y Bash. Bash no es realmente un lenguaje de *scripting*, más bien se puede definir como un intérprete de comandos. Aparte de esto, es complejo de escribir y mucho menos potente que Python. Sin embargo, este último es un lenguaje de programación completo, capaz de crear desde *scripts* sencillos hasta programas complejos. Además, Python permite la importación de módulos de manipulación de objetos JSON o de gestión de conexiones HTTP que facilitan mucho la programación. Por todo ello, se considera más prometedora la opción de Python.

2.2.3 Sistema de consulta

A estas alturas, ya se tienen los documentos del MAPA extraídos, se han tratado para que sean relevantes y contengan información verdaderamente útil y se encuentran almacenados en un soporte que facilita su posterior búsqueda. El siguiente y último paso que se necesita efectuar es el desarrollo de un portal web que, mediante una interfaz gráfica, permita al usuario final realizar sus consultas.

Por tanto, el sistema de consulta estará formado por una aplicación web que tendrá una interfaz gráfica con facetas, con la que los usuarios podrán interactuar. Como también se tiene presente una gestión de usuarios mediante un inicio de sesión y un registro en la aplicación, se necesita por tanto un método de persistencia de los usuarios (una base de datos) y un sistema de *backend* (que exponga una API REST) que actúe de puente entre la aplicación web y la base de datos.

Aparte de las diferentes facetas de búsqueda que se deben incluir en este sistema, también sería útil la posibilidad de editar la información presente. Esta funcionalidad se considera interesante ya que de esta manera se puede corregir o enriquecer la información de los documentos. Del mismo modo, se considera interesante también una funcionalidad de tipo «me gusta» mediante la cual el usuario autorizado puede aumentar la relevancia de un documento si considera que lo es. Entonces, cuando se recupere la información, existirá un filtro que obtendrá solamente los documentos de la relevancia especificada. Para evitar problemas de inconsistencia de datos, estas dos funcionalidades deberán integrarse con un control de accesos basado en roles bien definidos.

En una fase temprana de desarrollo, en vez de construir directamente la aplicación web para visualizar los documentos, se puede usar Kibana como herramienta de visualización. Como se adelanta en la sección anterior, ElasticSearch pertenece al ecosistema de Elastic Stack, un conjunto de herramientas que trabajan cohesionadas, donde ElasticSearch actúa de motor de búsqueda y Kibana⁸ actúa como herramienta de visualización de la información contenida en el índice de ElasticSearch. Entonces, para visualizar y valorar

⁸ <https://www.elastic.co/es/what-is/kibana>

en una fase inicial del desarrollo si la información recuperada es la que se desea, se puede usar Kibana como herramienta dedicada a este fin. Se muestra en detalle, acompañado de un ejemplo, cómo se visualizaría la información con Kibana en el Anexo A.4.

No obstante, Kibana no cumple el objetivo de portal web con facetas y gestión de usuarios. Por eso, en una fase media de desarrollo, cuando se haya verificado que la información del índice es correcta, suficiente y accesible, se propone entonces la construcción de la aplicación web final. Se considera que el uso de la librería React⁹, escrita en JavaScript, resulta la opción más prometedora. La extensa documentación y tutoriales, el hecho de ser de código abierto, su actual popularidad, su escalabilidad y la reusabilidad de código que presenta mediante el concepto de Componente, son algunas de las muchas ventajas que cargan de peso la decisión y que desbanca el uso de otras tecnologías como Angular.

En cuanto a la base de datos, se opta por usar una base de datos clave-valor (base de datos documental), más concretamente MongoDB¹⁰. Se considera que ésta es la mejor opción debido a su flexibilidad en el esquema de datos, a la facilidad de uso y de acceso a los datos junto a la fuerte integración con Elasticsearch. Se descarta el uso de una base de datos relacional tradicional porque son muy cerradas en cuanto a modificaciones en el esquema de datos.

Finalmente, como punto de unión entre la aplicación web y la base de datos, se necesita construir un sistema de *backend*, que exponga una API REST. La API es una pieza fundamental en el desarrollo de este sistema. En el caso de que un usuario inicie sesión en la aplicación, es necesario conocer si en la base de datos está presente ese usuario para permitir o denegar su entrada. Para evitar comunicaciones directas con la base de datos, se añade una capa extra intermedia, la API. Ésta se encarga de recibir peticiones de la aplicación web (como por ejemplo un inicio de sesión), enviar peticiones a la base de datos, recibir su respuesta y finalmente, en base a lo recibido por parte de la base de datos, enviar la respuesta final a la aplicación web. Por su calidad, su rendimiento, facilidad de aprendizaje y paralelización de trabajos, se considera que Node.js¹¹ es la tecnología más prometedora para implementar el servidor de *backend*.

2.3 Requisitos

Como resultado del proceso de análisis anteriormente descrito, se han obtenido una serie de requisitos funcionales y no funcionales que el sistema debe cumplir. No obstante, como se han identificado tres subsistemas durante el análisis, se organizan los requisitos por subsistema. A continuación, se muestran aquellos que se consideran críticos para el éxito del sistema en su conjunto.

Para el sistema de extracción, se necesitará una configuración particular que permita la obtención de documentos del MAPA únicamente, extrayendo si es necesario más campos que los predeterminados ya que más tarde podrán ser objetivos de búsquedas. El sistema de persistencia y procesamiento deberá contar con un soporte (base de datos documental) donde se pueda almacenar y consultar la información junto a un conjunto de procesos que

⁹ <https://es.reactjs.org/>

¹⁰ <https://www.mongodb.com/es/what-is-mongodb>

¹¹ <https://nodejs.org/es/about/>

se encarguen de enriquecer y completar la información presente. Finalmente, el sistema de consulta deberá contar con una aplicación web con gestión de usuarios, control de accesos basado en roles para evitar inconsistencias en los datos que se editen y numerosas facetas que permitan encontrar la información de una forma sencilla y precisa.

En cuanto a los requisitos no funcionales, se considera que se debe realizar una correcta configuración del proceso de extracción para que funcione lo más rápido y respetuoso posible (vigilando la cantidad de peticiones al servidor) y el desarrollo de un portal web homogéneo, fácil de usar y que se pueda ejecutar desde diferentes navegadores web. Se considera también un requisito no funcional el uso de una base de datos documental para almacenar la información de los documentos.

Para consultar los requisitos funcionales y no funcionales al completo, ver Anexo A.5.

2.4 Conclusiones

El análisis nos muestra que el portal web del MAPA tiene una serie de problemas que convierten una tarea de búsqueda en una labor compleja y tediosa. Para tratar de solucionar estos problemas y construir un portal web ambicioso con facetas, se necesitan varios sistemas que trabajen de forma conjunta. En resumen, es necesario conseguir los documentos del portal del MAPA con un *crawler*, almacenarlos en una base de datos que cuenta además con un motor de búsqueda, procesarlos con unos programas específicos y finalmente, construir el buscador que ofrezca una interfaz web gráfica para llevar a cabo las consultas. De esta forma, el usuario final puede navegar e interactuar con el portal web para buscar datos a conveniencia. Además, se quiere enriquecer el último sistema añadiendo gestión de usuarios para poder controlar quién puede y quién no hacer determinadas acciones.

3 Diseño

En este apartado se presenta cómo se ha diseñado el sistema. En primer lugar, se revisa la arquitectura del sistema, la cual ofrece una visión general de este, mostrando todos sus componentes y cómo se comunican. Más adelante, se aporta una visión más detallada de cómo está diseñado cada módulo o componente por separado.

3.1 Arquitectura del sistema

Por su naturaleza, el sistema se tiene que descomponer en subsistemas. Esta necesidad hace que el sistema final siga una arquitectura multicapa, técnica de diseño que se basa en descomponer un problema complejo en diferentes subproblemas de menor complejidad de tal forma que se pueda aportar una solución robusta formada por la conjunción de estos subproblemas. Al seccionar un sistema en diferentes niveles de abstracción, o capas, se puede tratar cada capa de forma independiente [10]. La arquitectura del sistema es la que se muestra en la siguiente Figura (para un nivel de detalle superior, véase el Anexo B.1).

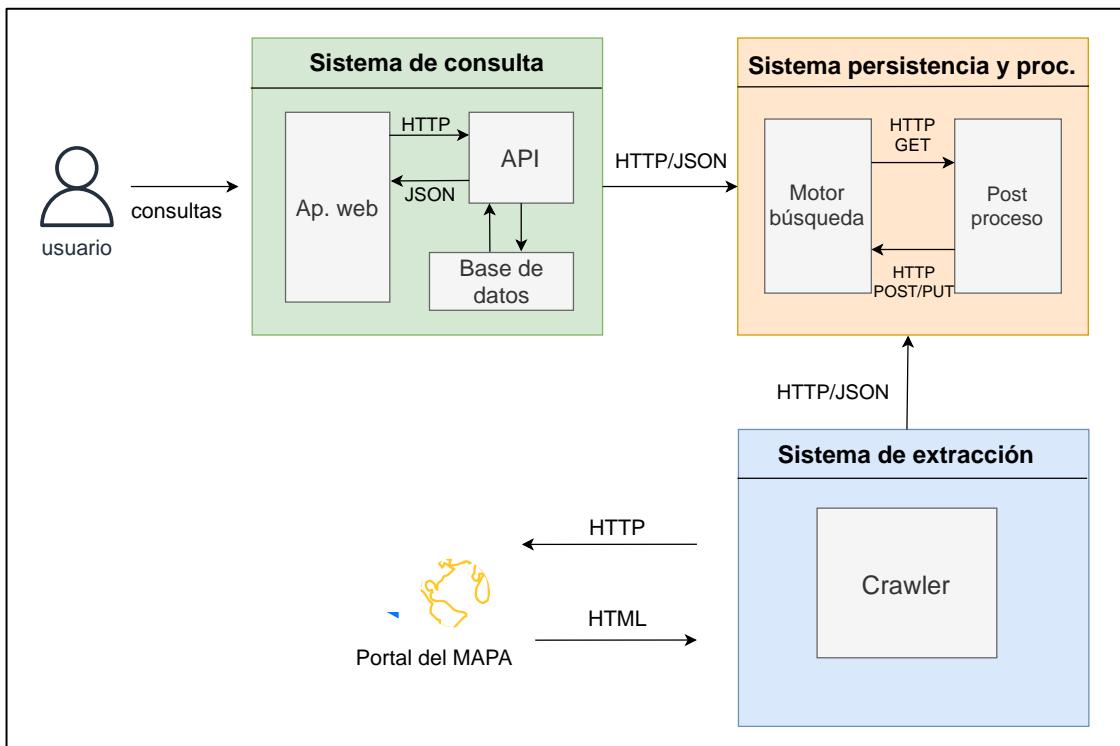


Figura 3. Arquitectura del sistema. Fuente: propia.

Como se puede apreciar en la imagen, se necesita un sistema de extracción con un *crawler*, un sistema de persistencia y procesamiento con un motor de búsqueda junto a un subsistema de postproceso y finalmente, un sistema de consulta formado por la aplicación web, la API REST y la base de datos. Estos tres sistemas son las capas o niveles de abstracción necesarios para que el sistema final funcione. En este ecosistema en particular, el diseño en capas es esencial para que en el caso de que, por ejemplo, se modifiquen las páginas web del portal del MAPA, no haya que modificar absolutamente todo sino únicamente hacer pequeñas modificaciones donde se considere oportuno. Con esto se trata de crear una arquitectura modular donde sus componentes no sean absolutamente dependientes entre sí pero que trabajen de forma conjunta.

Como es un sistema modular, las comunicaciones entre componentes son muy frecuentes y necesarias para que la información fluya correctamente. En primer lugar, el sistema de extracción obtiene los documentos HTML haciendo peticiones HTTP al portal del MAPA. Cuando se finaliza el trabajo de extracción, los documentos JSON que se obtienen, se indexan en el motor de búsqueda mediante peticiones HTTP al índice. El subsistema de postproceso también usa el protocolo HTTP para obtener los documentos y más tarde volverlos a almacenar de nuevo en el índice. Finalmente, el sistema de consulta también hace uso del protocolo HTTP para consultar el índice y mostrar los resultados de las consultas. Internamente, las peticiones que la aplicación web envía a la API REST son también usando este protocolo. Como se ha podido observar, el protocolo de comunicación usado para la transferencia de información es el protocolo HTTP.

A parte de la arquitectura del sistema, otro aspecto interesante que se necesita conocer es el modelo de información. Este concepto hace referencia a cómo fluye la información por nuestro sistema. La Figura 4 muestra este comportamiento.

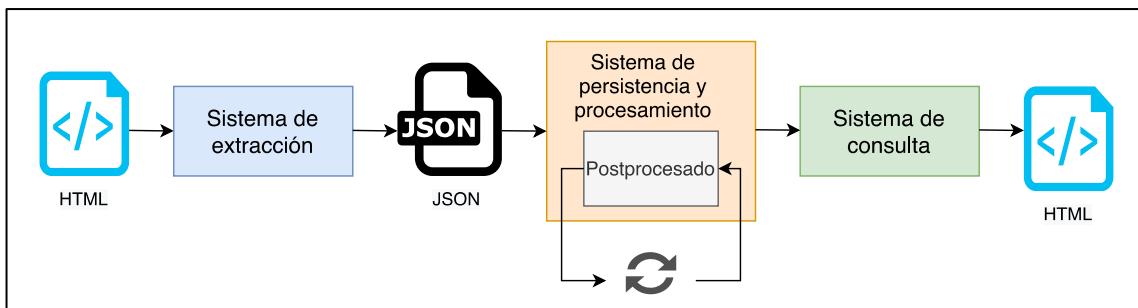


Figura 4. Modelo de información del sistema. Fuente: propia.

Se puede apreciar cómo el sistema de extracción parte de documentos HTML (las páginas web del MAPA) y convierte la información contenida en los HTML en documentos JSON. Éstos se indexan en el subsistema de persistencia (motor de búsqueda) y el subsistema de postproceso actualiza estos JSON añadiendo propiedades de interés que no estuviesen presentes en los documentos originales. Finalmente, el último sistema consulta estos documentos JSON y los convierte en resultados en formato HTML para que sean visualizados por pantalla por el usuario.

3.2 Sistema de extracción

El sistema de extracción necesita de un *crawler*, la herramienta que hay que configurar para obtener los documentos de la web mediante peticiones HTTP con el objetivo de que se puedan manipular y consultar más adelante. También se necesita una configuración para dirigir la extracción únicamente al dominio del MAPA (filtrando dominios no interesantes) y para extraer y examinar los metadatos de los documentos como pueden ser el tipo (que nos permite saber si se trata de un documento estructurado, no estructurado o de un formulario), el título (para tratar de extraer una posible categoría en base a él), la fecha de publicación, la URL (mismo objetivo que con el título), los enlaces entrantes (para saber qué documentos lo apuntan y poder enriquecerlos) y la extensión del documento (sabiendo la extensión del documento, se puede actualizar la información de los documentos que lo apuntan, añadiendo la información de que esos documentos contienen ese tipo de fichero). Finalmente, también debe existir una configuración para reunir toda esta información e indexar el resultado obtenido en el subsistema de persistencia (base de datos documental).

También se deberá diseñar el sistema para cumplir otro tipo de requisito más secundario que consiste en que el proceso de extracción sea lo más rápido posible sin dejar de ser respetuoso con el sitio web. Se deberá entonces configurar la cantidad de hilos que se usarán para realizar las peticiones, el tiempo de respuesta máximo que el crawler se demorará entre peticiones, la cantidad de hilos que tendrán acceso a la cola de extracción, etc. En definitiva, el diseño de esta configuración hará que el crawler obtenga los documentos del MAPA, los filtre, analice y, finalmente, los indexe en el subsistema de persistencia.

3.3 Sistema de persistencia y procesamiento

Una vez expuesto lo que se necesita configurar en el sistema de extracción compuesto únicamente por el *crawler*, el diseño del sistema de persistencia y procesamiento es algo más complejo. Por tanto, se aprovecha esta sección para ofrecer una imagen más detallada del diseño de este sistema y, sobre todo, de las comunicaciones entre ambos componentes de este.

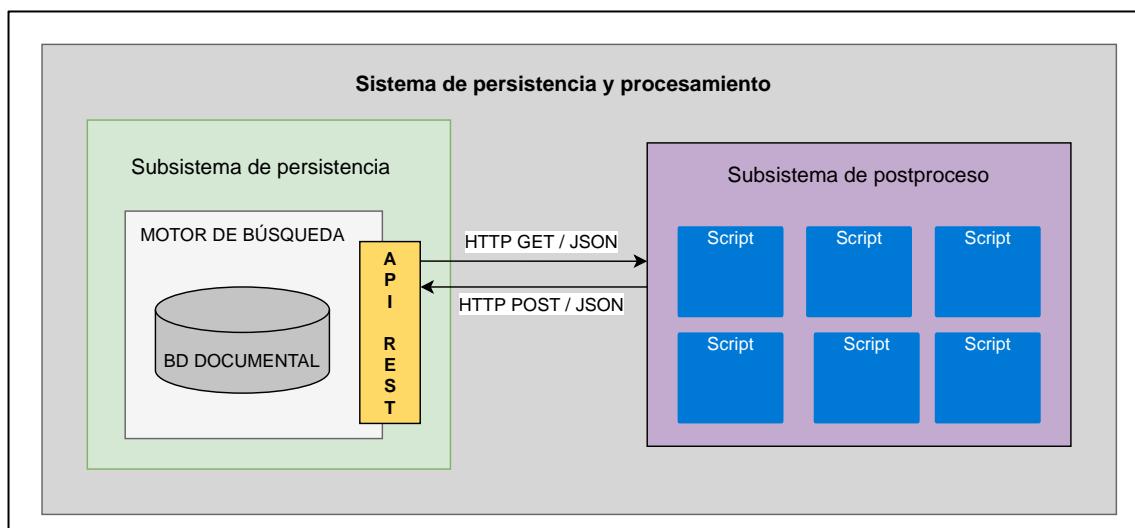


Figura 5. Diseño del sistema de persistencia y procesamiento. Fuente: propia.

Como ya se ha mencionado en apartados anteriores, los documentos JSON extraídos del MAPA y almacenados en el motor de búsqueda no siempre contendrán información suficiente para responder las consultas del sistema de consulta. Por eso, el subsistema de postproceso es clave. Los procesos o *scripts* que lo conforman atacan el API REST que ofrece el motor de búsqueda mediante peticiones HTTP. Una vez se tiene el documento, se valora si se necesita manipular. En caso afirmativo, el documento se modifica y se vuelve a indexar en el motor de búsqueda usando su API REST, al igual que en el caso de su obtención. Ejemplos de consultas necesarias pueden ser la obtención de todos los documentos PDF no visitados, la actualización de un documento específico añadiendo un campo determinado, la suma de una unidad en el número de ficheros de un documento si ya tenía más de uno, etc. Todas estas consultas son la base de los *scripts* de post proceso, que se encargan de preparar los campos de los documentos, actualizar su información, descubrir información oculta en la URL o título de un documento, cribar información, etc.

Por ejemplo, imaginar un documento recuperado del portal del MAPA e indexado en ElasticSearch que contiene el título, la URL, su propio contenido y la fecha de

publicación. Si más adelante, el buscador web que se construya permite hacer filtrados por categoría, esta consulta en específico será imposible de satisfacer ya que no hay ningún campo que haga referencia a la categoría del documento. Por tanto, el subsistema de postproceso deberá clasificar las páginas y asignarles una categoría. Más adelante, en el apartado de Implementación, se explica con detalle cómo se ha implementado esto.

3.4 Sistema de consulta

De la misma forma que en la sección anterior, esta se centra en ahondar en el diseño del sistema de consulta. En primer lugar, antes de comenzar con la fase de implementación, es necesario realizar un boceto de la aplicación web a construir. El boceto de la página principal del sistema de consulta se muestra en la Figura 6. No obstante, el diseño del resto de pantallas de la aplicación web junto con el correspondiente mapa de navegación que permite visualizar y entender cómo navegar por ella se incluyen en las secciones B.3 y B.2 del Anexo, respectivamente.

Focused crawler

<http://focused-crawler.com/>

Focused crawler endpoint

Username

Formatos

- pdf
- csv
- xml

Contiene

- PDF
- JSON

Tipo de página

...

Fecha de publicación

01/01/2022 - 01/09/2022

Categoría

- Agricultura
- Ganadería
- Pesca

Relevancia

0 5

Búsqueda

3 resultados encontrados

Datos 2020

src: ...

Relevancia: ★★★★☆

Fecha de publicación: 19/12/2021

Información de consumo de leche y derivados

src: ...

Relevancia: ★★★★★

Fecha de publicación: 26/12/2021

Solicitud de Ensayos

src: ...

Relevancia: ★★★★★

Fecha de publicación: 04/02/2022

PDF (4) XML (8) Editar Like

PDF (21) XLSX (11) PPTX (1) Editar Like

MSWORD (3) Editar Like

Figura 6. Boceto de la pantalla principal del sistema de consulta. Fuente: propia.

Tanto esta pantalla principal como el resto de las pantallas están diseñadas en base a los requisitos de la aplicación web. Se puede apreciar en la Figura 6 las diferentes facetas en la parte izquierda del buscador que permiten a los usuarios realizar los filtrados, la caja de búsqueda en la parte superior central que permite al usuario realizar consultas textuales y, debajo de esta, aparecen los resultados de la búsqueda con los documentos que satisfacen la consulta y que muestran información correspondiente al título, URL,

relevancia, fecha de publicación y ficheros que contiene. Como se comenta en la sección 2.2.3 correspondiente al análisis del sistema de consulta, este también cuenta con unos roles que se otorgan a los usuarios y que les permiten desempeñar según qué acciones. El diseño de estos roles se muestra en la Tabla 2.

Tabla 2. Roles existentes en la aplicación web.

Rol	Alcance
Administrador	Sólo existe un usuario con rol de administrador. Este rol permite realizar cualquier operación de búsqueda, edición y dar «me gusta». También, este rol permite modificar los roles de cualquier usuario presente en la base de datos para añadir o restringir el alcance de operaciones disponibles.
Editor de contenido	Los usuarios con este rol pueden hacer búsquedas, ediciones de los documentos y dar «me gusta» a los documentos. Sin embargo, no pueden modificar los roles del resto de los usuarios.
Revisor de contenido	Usuarios con este rol pueden realizar búsquedas y dar «me gusta» a las publicaciones, pero no pueden editar ni tampoco modificar el rol del resto de usuarios. Este es el rol por defecto que se otorga a los usuarios que se registran en el sistema.
Usuario	Este rol es el más restrictivo. Usuarios con este rol únicamente pueden hacer búsquedas. No se les permite editar, dar «me gusta» o modificar el rol del resto de usuarios.

Como se ha mencionado anteriormente, dependiendo del rol que tenga un usuario, este podrá acceder a unas funcionalidades u otras, pudiendo siempre hacer búsquedas en el sistema. Precisamente, tanto para gestionar las operaciones que conlleven un control de accesos como la edición, dar «me gusta» o la modificación de un rol de usuario (únicamente para el administrador) como también para gestionar otras operaciones como el inicio de sesión o registro, la API REST del sistema de consulta deberá exponer una serie de servicios que lleven a cabo todas estas funcionalidades. Esta API se considera un punto intermedio entre la aplicación web y la base de datos de usuarios.

Finalmente, para acabar de entender la dinámica de este sistema de forma global y cómo se comunican los componentes entre sí con un nivel superior de detalle, se incluye en la sección B.4 del Anexo un diagrama de secuencia que representa un posible caso de uso de la aplicación web final.

3.5 Conclusiones

Tal y como se ha podido comprobar, el sistema se ha diseñado siguiendo una arquitectura multicapa. De esta forma, se consigue separar el sistema en tres niveles de abstracción o capas de menor complejidad. Estas capas se comunican entre sí usando el protocolo HTTP y generalmente, intercambiando objetos JSON. La posibilidad de dividir el sistema en varias capas o módulos independientes se puede conseguir gracias a que existen unas API que permiten comunicar estas capas entre sí.

4 Implementación

La implementación sigue fielmente el diseño recogido en la sección anterior. Cada sección describe cómo se implementa un subsistema presente en el proyecto final (sistemas de extracción, persistencia y procesamiento y de consulta). El nivel de detalle presentado varía en función de la naturaleza del sistema.

4.1 Sistema de extracción

Hasta este punto, se conoce que el sistema de extracción está compuesto por un *crawler*. El *crawler* que se va a implementar es Apache Nutch. Como ya se ha mencionado, Nutch es un *crawler* de software libre escrito en Java sustentado por una estructura de ficheros que son parte fundamental de su funcionamiento. Por ese motivo Nutch es tan configurable, porque cuenta con una gran cantidad de ficheros y de *plugins* o puntos de extensión [11] que permiten modificar su comportamiento de forma sencilla. Este apartado tiene como objetivo aportar una pequeña descripción de la configuración escogida para el proceso de extracción. Se puede consultar la configuración al completo en el Anexo C.1.

En primer lugar, cabe mencionar que el punto central de la configuración de Nutch es el fichero “nutch-site.xml”. Este fichero contiene una serie de propiedades (etiquetas) que pueden añadirse o quitarse y que dictan el comportamiento del *crawler* en el proceso de extracción. Entre las principales etiquetas que se incluyen en este fichero se encuentran los *plugins* que se van a usar, el motor de búsqueda donde se indexará la información extraída, los metadatos que se quieren extraer y más información relevante al proceso de extracción.

Tras realizar un intenso estudio previo se exponen en la Tabla 3, de forma breve, las necesidades principales del proyecto y qué *plugins* se han configurado para solucionarlas. Para consultar en detalle los *plugins* utilizados junto a una detallada descripción de estos, ver Anexo C.2.

Tabla 3. Configuración de *plugins* utilizada.

Plugin usado	Necesidad que cubre
protocol-http	Se ha incluido este <i>plugin</i> ya que se necesita hacer peticiones HTTP al portal web del MAPA para obtener los documentos.
urlfilter	Se necesita este <i>plugin</i> para poder dirigir la extracción, es decir, para filtrar las siguientes URL del proceso de extracción.
parse	Es necesario también un mecanismo de análisis del contenido. Para ello se usa este <i>plugin</i> de análisis. No solamente se analiza el contenido HTML, sino también los metadatos (metatags) y ficheros (tika).
index	Una vez se tiene el contenido extraído y analizado, debe existir algún mecanismo de elección de los datos que se indexarán. Este <i>plugin</i> hace exactamente eso, escoge la información que se añadirá a los documentos indexados (información básica, meta etiquetas, links, etc.).
urlnormalizer	Para seguir inyectando URL en la cola de extracción, éstas se deben normalizar (estandarizar). Este <i>plugin</i> cubre esta necesidad.
indexer	Este <i>plugin</i> cubre la necesidad de indexación. Una vez la información está dispuesta y lista para indexar, este plugin se encarga de ello.

Gracias al *plugin* “parse-metatags”, se consiguen extraer las meta-etiquetas del documento del MAPA. Como se ha mencionado en la sección 3.2, hay muchos metadatos interesantes a extraer: el título y la URL (aparte del propio valor de identificación que suponen, son también un mecanismo para extraer una posible categoría del documento que se encuentre oculta en ellos), la fecha de publicación (para filtrados) o el tipo de documento. El tipo tiene tres posibles valores: “contenido”, “noticia” o “aplicación”. Un documento de tipo “contenido” puede apuntar a otros documentos, un documento de tipo “noticia” generalmente es texto plano y no apunta a otros documentos y finalmente, el documento de tipo “aplicación” se corresponde con una página web con formulario(s). Se puede concluir con que el metadato “tipo” de un documento es muy útil para decidir si una página web puede tener un contenido similar al de una base de datos.

Los *plugins* que se muestran en la Tabla 3 serían suficientes para cubrir las necesidades básicas de un *crawler*. No obstante, en este caso se necesitan más propiedades para adaptar el proceso de extracción a nuestro interés personal. Algunas de estas propiedades se recogen en la Tabla 4 (consultar Anexo C.2.1 para ver todas las propiedades usadas).

Tabla 4. Configuración de propiedades de Nutch utilizadas.

Propiedad	Uso
metatags.names	Esta propiedad supone la entrada del <i>plugin</i> “parse-metatags” y sirve para indicar las meta-etiquetas adicionales que se desean extraer.
index.parse.md	Gracias a esta propiedad, los metadatos adicionales extraídos y posteriormente analizados se preparan para indexar también.
elastic	Propiedad para indexar el contenido en ElasticSearch y no en otro motor de búsqueda. Se especificará aquí el nombre del índice, la IP del servidor de ElasticSearch, el puerto, el nombre del clúster, etc.
fetcher	Propiedad para controlar todo lo relacionado con el proceso de obtención de la información (cantidad de hilos a utilizar, retardo entre peticiones, hilos para acceder a la cola de extracción...).

Es necesario destacar la importancia de un *plugin*, el “index-basic”. Este *plugin* construye el documento JSON que se indexará más adelante con la información básica del documento original extraído (título, URL, contenido, enlaces, etc.). Sin embargo, sería muy útil contar también con el MIME type del documento para, a posteriori, hacer filtrados en base a esta propiedad. El MIME type o tipo MIME de un documento indica, de forma estandarizada (en IETF RFC 6838), la naturaleza y formato de un documento [\[12\]](#). Por ejemplo, algunos tipos MIME que existen son *image/gif*, *image/jpg*, *application/pdf* o *text/html*.

Este concepto, al no ser un metadato, no se puede incluir en el *plugin* de análisis de metadatos (parse-metatags) como sí se hace con el título y con la fecha de publicación. Por eso, para añadir esta propiedad a todos los documentos, se ha tenido que modificar el *plugin* index-basic. Se detalla cómo se ha realizado este proceso en el Anexo C.2.2.

4.2 Sistema de persistencia y procesamiento

Como se plantea en la sección 2.2.2 del análisis, la propia naturaleza del sistema de persistencia y procesamiento y los objetivos que persigue hacen que este deba dividir su funcionalidad en dos subsistemas: el subsistema de persistencia, que se ha implementado

en ElasticSearch y el subsistema de postproceso, implementado usando el lenguaje de programación Python.

Por un lado, el subsistema de persistencia se ha implementado usando ElasticSearch, una pieza de software configurable que encaja a la perfección con las necesidades específicas del problema y que, por tanto, se ha usado para poder cumplir con los objetivos del sistema. Este software cubre el objetivo de almacenamiento y recuperación de la información gracias a una serie de API REST que se pueden consultar utilizando el lenguaje específico de consultas DSL. Ver Anexo D.1 para conocer en detalle cómo se implementa una consulta DSL interpretable por ElasticSearch partiendo de una consulta en lenguaje natural.

Por otro lado, el subsistema de post proceso, implementado en Python, consta de varios *scripts* que recuperan la información del índice, la enriquecen y la vuelven a almacenar. Tras un estudio de la información existente en un documento extraído del MAPA (título, contenido, URL, tipo, etc.) y la información que se va a consultar más adelante en el buscador, se han obtenido un conjunto de propiedades que faltan y que, por tanto, deben añadirse a la información almacenada sobre los documentos. Se muestra en la Tabla 5 los *scripts* que se han desarrollado y el objetivo que cubre cada uno de ellos. Los nombres de los *scripts* son autodescriptivos.

Tabla 5. Scripts desarrollados y objetivos que cubren.

Script	Objetivo
postProcessing.py	Este <i>script</i> se encarga de añadir información acerca de los ficheros contenidos en el documento y la cantidad de cada uno. Es decir, si el documento tiene algún PDF, el <i>script</i> se encarga de añadir una propiedad que así lo indique junto a una propiedad que lleve la cantidad de éstos. Después de su ejecución, se pueden filtrar, por ejemplo, documentos que contengan algún PDF.
postProcessing_parent.py	Este <i>script</i> necesita del anterior para funcionar. Básicamente, en base a los ficheros que tiene el documento, construye una lista con ellos y la añade como atributo al documento. Si el documento tiene PDF y tiene XML, entonces se añade la lista [PDF, XML] al documento como atributo. De esta forma, se pueden hacer filtrados en conjunto. Se hace así ya que, para hacer filtrados de más de una propiedad, ElasticSearch debe tener todos los campos fuente en una lista.
addScoring.py	Añade una propiedad de <i>scoring</i> (si no estaba ya) al documento. Esta propiedad lleva un conteo de la cantidad de «me gusta» que posee el documento, es decir, de la relevancia de este.
quitarNoticias.py / quitarContenido.py	Estos <i>scripts</i> se encargan de limpiar la base de datos, es decir, su cometido es quitar documentos (noticias o contenido) que se consideren irrelevantes (que no contengan ningún documento, que no sean formularios, etc.).
keywords.py	Este <i>script</i> se encarga de clasificar los documentos extraídos del MAPA en categorías: agricultura, pesca, ganadería, alimentación, etc. En base a la URL y al título del documento se consigue realizar este proceso.

Además, los seis *scripts* implementados se han diseñado con capacidad para funcionar aun cuando se realicen nuevas extracciones. Esto significa que estos solamente modificarán la información fruto de las nuevas ejecuciones del proceso de extracción, es

decir, la información que ya se ha manipulado, no se vuelve a procesar. Esta metodología supone dos grandes ventajas. Por un lado, los *scripts* no tardan mucho tiempo en ejecutarse ya que no procesan toda la información de nuevo, solamente la más reciente. Por otro lado, la información que se edite desde el buscador, no se reescribirá al ejecutar estos programas y permanecerá intacta, como debe ser.

Conociendo esto, se procede a mostrar el funcionamiento e implementación a alto nivel del primer *script* en la Figura 7 (siendo muy similar para el resto de ellos).

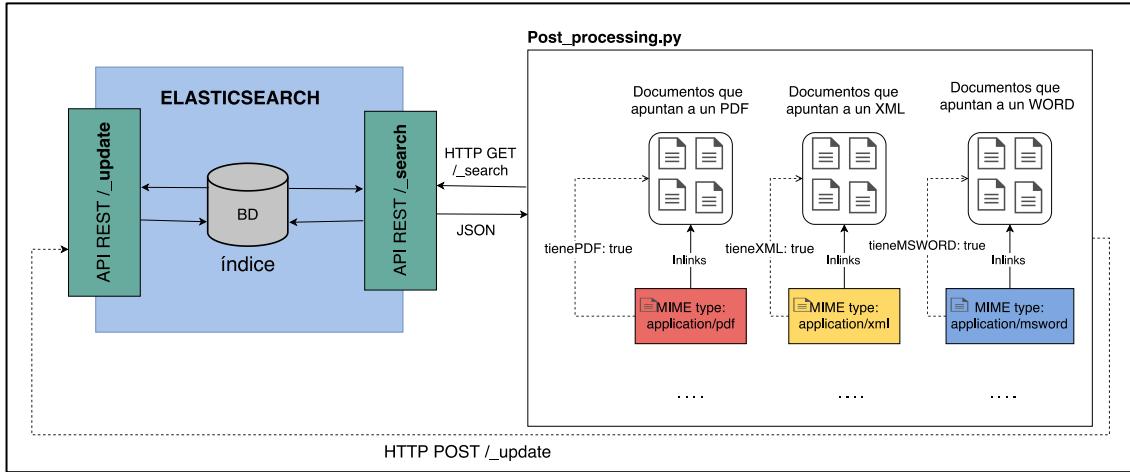


Figura 7. Implementación del script *postProcessing.py*. Fuente: propia.

Se puede apreciar la manera en la que se ha implementado el *script*. En primer lugar, se realiza una petición HTTP GET a la API de búsqueda de Nutch (`/_search`) con la consulta DSL correspondiente a la obtención de los tipos de ficheros existentes en el MAPA (PDF, XLSX, PPTX, MSWORD, XML, etc.). Al recibir el JSON respuesta, el *script* se encarga de navegar por los enlaces entrantes de cada uno de ellos, agregando al documento que los apunta la propiedad correspondiente al tipo MIME del documento apuntado junto a la cantidad de este tipo de fichero.

Finalmente, se realiza una petición de actualización (HTTP POST) a la API de actualización (`/_update`) de ElasticSearch enviando esta información en una consulta DSL. A modo de ejemplo, si los documentos X e Y son un PDF, el documento Z es un XML y todos son apuntados por el documento R, al final del proceso la base de datos guardará que el documento R tiene PDF y XML y que además tiene dos unidades del primer tipo y una del segundo tipo.

Para consultar más detalles de interés en la implementación de los demás *scripts*, consultar Anexo D.2.

4.3 Sistema de consulta

Este sistema se corresponde con la parte final del sistema completo. Al igual que el sistema anterior, este también tiene dos módulos: la aplicación web, implementada usando React y el *backend*, cuya base de datos se ha implementado en MongoDB y cuya API se ha implementado en Node.js.

La base de datos de usuarios es una base de datos NoSQL. Por ello, lo primero que se debe definir es el modelo de datos de un usuario (la única unidad almacenable que se

tiene). Se considera que la información que se debe almacenar respecto de un usuario es su correo electrónico, contraseña, fecha de registro, de su último inicio de sesión y su rol. Como se comenta en la sección 3.4, cada usuario cuenta con un rol que le permite ejecutar unas funcionalidades específicas dentro de la aplicación.

Una vez definido el modelo de datos (que se puede ver con más detalle en el Anexo E.1), se desarrollan los controladores que permiten realizar las operaciones requeridas, es decir, un controlador que implementa el inicio de sesión, el registro, la edición, el mostrar usuarios y roles, etc. Finalmente, se exponen al exterior una serie de rutas (servicios) que ejecutan los controladores anteriores. Al final de todo el proceso, se obtiene una API REST con una serie de servicios que realizan operaciones sobre la base de datos, abstrayendo esta complejidad al usuario final. Estos servicios de la API son los que se encargan de verificar que el usuario emisor de la petición tiene el rol adecuado para ejecutar la operación. Se puede consultar el Anexo E.2 para obtener más información del desarrollo de la API.

En lo referente al desarrollo de la aplicación web, se ha escogido el framework de Express¹² para su construcción. El proceso de implementación consiste en dos principales fases. En primer lugar, la implementación gradual de los diferentes requisitos de búsqueda y filtros que se han definido en la fase de análisis. Para ello, se ha usado la librería de ReactiveSearch¹³, que permite construir y diseñar componentes gráficos atractivos y personalizables. Una vez se tienen todos los filtros desarrollados, el siguiente paso es integrar la gestión de usuarios con la aplicación web desarrollada, desarrollando las vistas y conexiones del inicio de sesión, del registro, de la edición, del «me gusta» de un documento y del panel de administración (solo accesible por el administrador) donde puede realizar las modificaciones de roles de los usuarios del sistema. Se puede consultar información más detallada acerca de la implementación de la aplicación web en el Anexo E.3.

Durante todo el desarrollo del *frontend*, se ha seguido muy de cerca la documentación y guías oficiales de React [13] para tratar de construir el mejor sistema posible. Por eso, se ha utilizado con frecuencia el concepto de componente (agrupan funcionalidad), *hooks* (actualizan al instante el estado), *arrow-functions* (minimizan mucho la complejidad de la función), etc. También, a la hora de hacer peticiones HTTP (tanto a ElasticSearch como a la API), se ha seguido el estándar del protocolo HTTP/1.1 [14].

Se muestra el resultado final obtenido en el apartado de pruebas.

¹² <https://expressjs.com/es/>

¹³ <https://opensource.appbase.io/reactivesearch/>

5 Pruebas

En este apartado se exponen los resultados obtenidos y el proceso de pruebas seguido para depurar y probar los diferentes aspectos finales del sistema.

5.1 Resultados obtenidos

Una vez se tiene el sistema construido, es momento de evaluar el resultado final. Se aprovecha esta sección para mostrar principalmente el portal web desarrollado ya que es la parte gráfica del proyecto con la que el usuario interactúa y los documentos extraídos. Se ha intentado hacer un desarrollo lo más fiel posible al prototipo construido en la fase de diseño. Se muestra en la Figura 8 la pantalla principal del portal.

The screenshot displays the main interface of the 'Focused Crawler Endpoint' application. At the top, there's a blue header bar with the title 'Focused Crawler Endpoint' and a user icon labeled 'Invitado'. Below the header, on the left, is a sidebar with a 'Formatos' section listing file types and their counts: .pdf (6972), .doc (137), .xml (12), .html (5998), .docx (183), .xlsx (659), and .pptx (6). To the right of the sidebar, there's a search bar with placeholder text 'Búsqueda genérica' and a 'Clear All' button. Below the search bar, it says '5 de 25 resultados encontrados en 9 ms'. Underneath, there's a 'Ayudas Nacionales' section with a link to 'https://www.mapa.gob.es/es/alimentacion/temas/integracion-asociativa/apoyos-integracion/Ayudas_Nacionales.aspx', a relevance rating of 5 stars (0), and a publication date of '2022-07-08'. There are buttons for 'PDF(15)', 'MSWORD(2)', 'DOCX(5)', and 'XLSX(5)'. On the far right, there are 'Editar' and 'Like' buttons. Below this, there are two sections for 'BIENESTAR ANIMAL' with links to 'https://www.mapa.gob.es/es/ganaderia/temas/produccion-y-mercados-ganaderos/bienestanimal/default.aspx' and 'https://www.mapa.gob.es/es/ganaderia/temas/produccion-y-mercados-ganaderos/bienestanimal/'. Each section has a relevance rating of 5 stars (0), a publication date of '2022-06-03', and buttons for 'PDF(2)', 'MSWORD(1)', 'Editar', and 'Like'. At the bottom, there are sections for 'Tipo de página' (Content, News, Form) and 'Fecha de publicación' (with a date input field). The footer contains the text 'Higiene y trazabilidad'.

Figura 8. Pantalla principal de la aplicación web. Fuente: propia.

Como se puede observar, se ha intentado ser lo más fiel posible al boceto diseñado. Se pueden observar las distintas facetas de filtrado propuestas a la izquierda del portal, aunque la Figura no muestra todas las facetas. También, en la parte superior central, se puede observar la caja de búsqueda que ofrece la posibilidad de realizar una búsqueda en texto. Tanto los resultados del filtro como de la búsqueda por texto se listan en la parte derecha del portal. Los documentos aparecen con más información de la que había en un inicio en el MAPA. Se tiene la posibilidad de editar el contenido y de aumentar su relevancia con los botones situados en la parte inferior derecha del documento en específico.

Se puede consultar el resto del resultado del interfaz gráfico del portal web en el Anexo F.1.

A parte del resultado visible de la aplicación web, también es necesario mostrar el resultado del proceso de extracción, es decir, cuántos documentos se han recuperado, en cuánto tiempo y cuántos de ellos contienen posibles bases de datos. El sistema de extracción ha recuperado un total de 16.545 documentos (ficheros y páginas web). Se muestra en la Tabla 6 los documentos recuperados y el tiempo empleado para ello en la versión inicial del crawler y en la versión final del crawler aplicando las optimizaciones de tiempo.

Tabla 6. Documentos recuperados y tiempo empleado en la versión inicial y final del crawler.

Cantidad de documentos	Tiempo empleado con el crawler versión optimizada	Tiempo empleado con el crawler versión inicial
708	13 minutos	31 minutos
3742	29 minutos	53 minutos
9770	41 minutos	N/A
16545	56 minutos	N/A

Como muestra la Tabla anterior, las optimizaciones realizadas al crawler hacen que el proceso de extracción sea más ligero, comprendiendo que es un proceso intrínsecamente lento ya que tiene que extraer el contenido, analizarlo, normalizarlo e indexarlo.

De esos documentos, la Tabla 7 resume la cantidad de páginas web que tienen un contenido similar al de una base de datos.

Tabla 7. Páginas web con contenido similar al de una base de datos.

Páginas web que contienen	Tipo de fichero	Cantidad encontrada
Formularios		50
Documentos estructurados	Hojas de cálculo (XLSX)	131
	Documentos de intercambio (XML)	19
Documentos no estructurados	Documentos digitales (PDF)	1879
	Documentos de texto (DOCX, MSWORD)	190
	Documentos para presentación (PPTX)	4
Otros	Documentos para presentación no abiertos (PPTS)	1
Total		2.274

Tal y como refleja la anterior Tabla, se han encontrado 2.274 páginas web que tienen un contenido similar al de una base de datos. Como es de esperar, la gran mayoría de páginas del portal son páginas web que contienen PDF ya que es el soporte por excelencia para la lectura de información. Se puede concluir también que el portal web del MAPA prefiere almacenar sus datos en un soporte no estructurado, que un soporte estructurado como puede ser un fichero Excel.

5.2 Proceso de pruebas

Una vez se tiene la certeza de que los requisitos se satisfacen, se han escogido un conjunto de URL con las que se ha probado que los datos de éstas (ficheros que contienen, categoría, título, tipo de documento, etc.) coinciden con los datos que el sistema final ofrece.

Se han realizado diferentes pruebas manuales. Un tipo de prueba es aquella que se asegura de que los datos del documento del MAPA se reflejan correctamente en el portal web. Otro tipo de prueba consiste en la correcta identificación de los formularios, es decir, se elige un documento del MAPA que contenga un formulario y se busca en el portal para ver si se ha etiquetado como tal. Además, este tipo de prueba no se puede llevar a cabo en el portal web del MAPA ya que no existe manera de filtrar por formularios o similar. Otro tipo de prueba consiste en comprobar que un documento del portal web desarrollado pertenece a la categoría que el portal web del MAPA muestra para ese documento. Finalmente, otro tipo de prueba común es la de editar y «me gusta» del documento. Se analizan documentos con información incompleta o errónea y se editan para que sean correctos. También, si se considera relevante un documento, se prueba la funcionalidad de «me gusta» para que se pueda filtrar por relevancia en futuras búsquedas.

Se muestra a continuación un ejemplo del primer tipo de prueba que consiste en seleccionar una URL del MAPA y comprobar que los datos de ésta son los que aparecen en el portal web desarrollado. Visitando la URL¹⁴ del MAPA, se puede comprobar que la información referente a ésta es la que se muestra en la Tabla 8.

Tabla 8. Información de la URL.

Campo	Valor
Título	Proceso de control y certificación
Fecha de publicación	2022-08-19
Tipo de página	Contenido
Ficheros que contiene	11 PDF y 1 DOCX

Ahora, el objetivo de la prueba es ver si esa información es la que se refleja en el sistema. Si buscamos esa página en el portal web, observamos el resultado que se muestra en la Figura 9.

¹⁴https://www.mapa.gob.es/es/agricultura/temas/medios-de-produccion/semillas-y-plantas-de-vivero/produccion-comercializacion-certificacion-vivero/Control_certificacion.aspx

The screenshot shows a search interface titled "Focused Crawler Endpoint" with a user "admin". The search bar contains the query "Proceso de control y certificación". Below the search bar, it says "5 de 5889 resultados encontrados en 29 ms".

Result 1:

- Formatos:** .pdf (9178), .doc (142), .xml (12), .html (6241), .docx (184), .xlsx (774), .pptx (8).
- Páginas web que contienen...**: Buscar tipo de fichero(s)
- Documento:** 25032015 PLAN NACIONAL DE CONTROL Y CERTIFICACIÓN DE PLANTAS DE VIVERO
- Detalles:** src: https://www.mapa.gob.es/es/agricultura/temas/medios-de-produccion/seminas-y-plantas-de-vivero/produccion-comercializacion-certificacion-vivero/Control_certificacion.aspx, Relevancia: 5 stars (0), Fecha de publicación: 2022-08-19.
- Opciones:** PDF(11) (button), DOCX(1) (button), Editar (button), Like (button).

Result 2:

- Formatos:** PDF (1879), DOCX (137), XLSX (131), MSWORD (53), XML (19), PPTX (4), OTROS (1).
- Páginas web que contienen...**: Buscar tipo de fichero(s)
- Documento:** Certificación de fabricantes
- Detalles:** src: <https://www.mapa.gob.es/es/agricultura/temas/medios-de-produccion/productos-fertilizantes/certificacion-de-fabricantes/>, Relevancia: 5 stars (0), Fecha de publicación: N/A.
- Opciones:** PDF (button), Editar (button), Like (button).

Filtros y Opciones:

- Tipo de página:** Contenido (selected), Noticia, Formulario.
- Filtrar por fecha de publicación:** Calendario para seleccionar fechas.

Figura 9. Búsqueda del documento específico en el portal web desarrollado. Fuente: propia.

Se puede apreciar que la información es consistente. Si se busca el título del documento y se filtra por tipo “Contenido”, aparece esa página como resultado (el primero). También se aprecia que la fecha de publicación es válida. Finalmente, el portal web muestra también que ese documento contiene 11 PDF y 1 DOCX, justo la cantidad que hay en la página web del MAPA.

No obstante, como la plataforma está en constante desarrollo y se va añadiendo y modificando información continuamente, se puede usar la funcionalidad de editar (ese es su objetivo) para modificar cualquier información que se considere incompleta o errónea.

Se muestran en el Anexo F.2 ejemplos del resto de pruebas realizadas.

6 Metodología y planificación

En este apartado se aborda la metodología de trabajo usada a lo largo de todo del proyecto junto a la planificación que se ha seguido, las etapas existentes y en qué ha consistido cada una de ellas. También, se aprovecha la sección 6.3 para mostrar la dedicación al proyecto. Finalmente, en la última sección de este apartado, se mencionan las herramientas de planificación utilizadas.

6.1 Metodología de trabajo

Con el fin de cumplir los objetivos a tiempo, pero atravesando un proceso de aprendizaje paulatino, la gestión del proyecto se ha llevado a cabo usando una metodología basada en iteraciones incrementales. Esta metodología cuenta con cuatro principales fases: la fase de preparación, la fase de definición, la fase de prototipado iterativo y la fase de preparación final [15].

La fase más temprana es la fase de preparación. Esta fase comenzó el día 14 de febrero de 2022 con la primera reunión. En ésta, se trataron temas como el título y alcance del proyecto, posibles tecnologías a usar, algunos de los requisitos mínimos que debía cumplir y ciertos objetivos cortoplacistas, sin entrar en mucho detalle. Más adelante, cuando la realización del proyecto ya era un hecho, se llevó a cabo la reunión oficial o “kick-off”, dando comienzo a la fase de definición. En ella se plasmaron los requisitos del sistema de una forma más ambiciosa, se creó el repositorio de GitHub, se redactaron las ideas de ésta y la reunión previa en documentos compartidos y se comenzaron a planificar las primeras tareas.

Una vez la fase de preparación vence, la tercera fase de esta metodología es la fase de prototipado iterativo. Esta fase de prototipado iterativo comenzó a mediados del mes de marzo y fue la más larga ya que duró hasta el mes de julio. La idea que subyace bajo esta fase es la siguiente: se definen unas tareas a realizar en el período de una semana y media. Durante este tiempo, las tareas son trabajadas y en el caso de existir dudas, se pueden comunicar al tutor por correo electrónico o usando la herramienta “issues” de GitHub. Cuando concluye el tiempo, si se considera oportuno se convoca una reunión presencial para tratar los diferentes puntos y se aprovecha para planificar las siguientes tareas con el tutor. Esta dinámica se va repitiendo hasta que finalmente se tiene un resultado de proyecto adecuado. Aquí se comienza la última fase, la de preparación final. En esta etapa, que duró dos meses, se prueba el proyecto final y se comienza a redactar la documentación.

Considero que las reuniones fueron muy útiles ya que se conseguía enseñar en vivo al docente el avance del proyecto y, sobre todo, aprovechar su ayuda para tratar de solucionar errores surgidos previamente. Cabe mencionar también que esta metodología es muy flexible, es decir, permite modificar sin problema la periodicidad de las reuniones presenciales (ya que el proyecto debe compaginarse con el resto de las asignaturas del cuatrimestre y muchas veces una semana y media no era tiempo suficiente para cumplir las tareas pendientes).

En conclusión, se considera que este tipo de metodología ha sido muy útil debido a que permite disminuir el riesgo a la hora de encontrar un fallo ya que se realiza retroalimentación en cada iteración y, además, se consigue ver cómo va evolucionando

el proyecto al ir añadiendo pequeños avances, pudiendo volver siempre atrás en cualquier momento.

6.2 Planificación

Esta sección está enfocada en explicar cómo se han ido asignando las diferentes tareas en el tiempo, cuánto ha costado cada una de ellas y las herramientas de gestión que se han utilizado para tratar de hacer la mejor planificación posible. El proyecto consta de aproximadamente 5 etapas. Se parte de lo más básico y se van añadiendo funcionalidades hasta cumplir con los objetivos, como se ha comentado en el apartado anterior. Las etapas son las que se enumeran a continuación:

Análisis del problema a resolver: incluye investigar la problemática existente en los buscadores de portales web públicos, la declaración de los objetivos y alcance del proyecto y la definición de los requisitos mínimos que debe tener el sistema.

Diseño de una prueba de concepto: incluye definir la arquitectura a alto nivel del sistema, acabar de definir los requisitos de este y tratar de tener claro qué componentes hay y cómo se comunican.

Implementación del prototipo: incluye implementar un prototipo muy sencillo del sistema final, que ni siquiera ataca al portal web del Ministerio. Su implementación incluye extraer los documentos con Nutch (previamente estudiando la documentación para ser conocedores de cómo hacerlo), indexarlos a ElasticSearch y poder visualizarlos en Kibana. Esta etapa es importante ya que, completándola, se consigue tener un punto de partida de nuestro sistema final (con nuestros tres subsistemas en fase inicial).

Aplicación del caso real: la etapa más larga y dura sin duda alguna. Esta etapa incluye, entre otras muchas tareas: refinar el subsistema de extracción, desarrollar el postproceso de la información una vez está indexada y construir el cliente final con todos los filtros. Todo esto, sin olvidar el proceso de documentación correspondiente.

Validación de resultados y mejoras a realizar: incluye la realización de una serie de pruebas manuales para comprobar que la información que se ve reflejada en el cliente es la esperada, analizar futuras mejoras del sistema final y redacción de la memoria.

Además de todas estas tareas, en todas y cada una de las etapas se incluyen reuniones presenciales con el docente y especialmente en la última, se incluye la tarea de redacción de la presente memoria técnica. Se plasma en forma de Diagrama de Gantt las cinco etapas junto a su planificación en la Figura 9.

Se puede apreciar, viendo la anterior imagen, la aplicación práctica de la metodología basada en iteraciones incrementales. Incluso en épocas en las que se ha tenido alguna dificultad para compaginar todo, se puede concluir con que el trabajo y esfuerzo han sido constantes a lo largo de este período de tiempo.

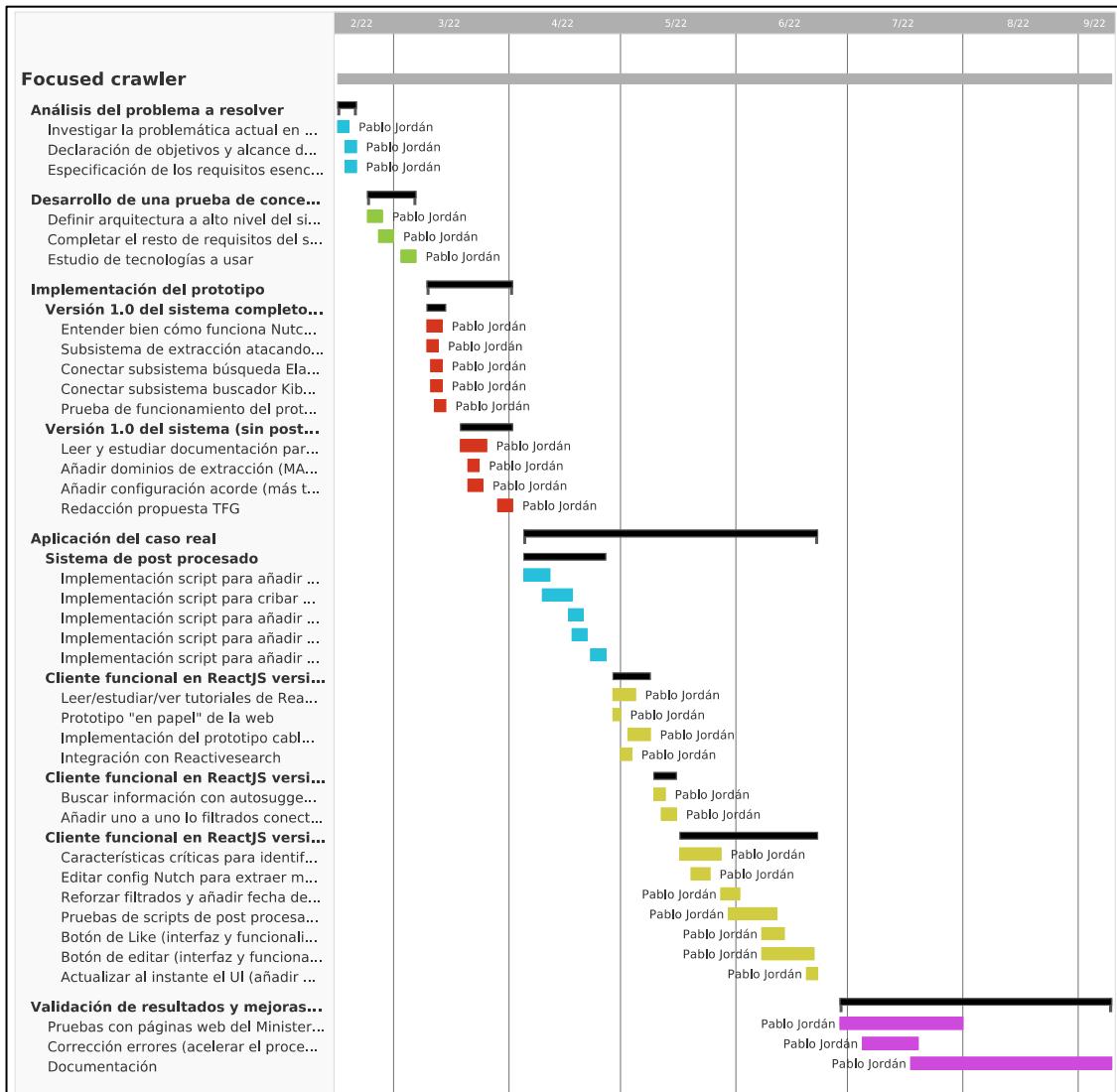


Figura 9. Diagrama de Gantt del proyecto. Fuente: propia

6.3 Dedicación

Esta sección está dedicada a mostrar de forma gráfica la dedicación parcial y total del proyecto. La Figura 10 expone el porcentaje de tiempo total invertido en cada una de las etapas o iteraciones.

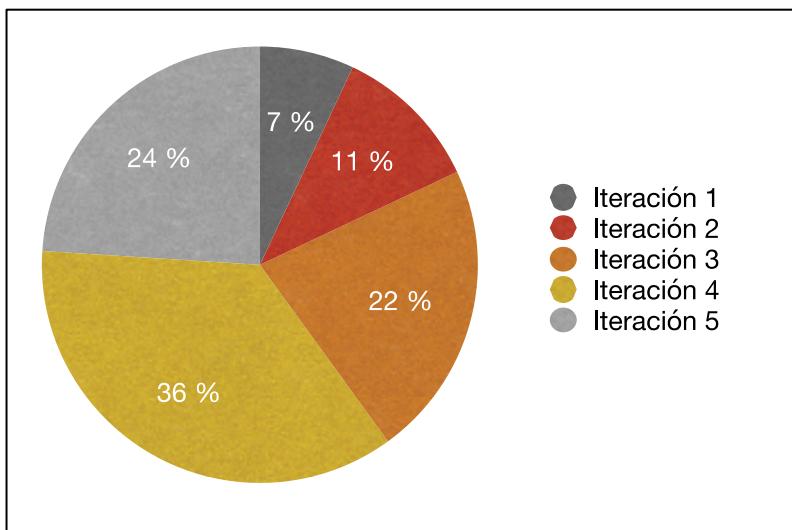


Figura 10. Porcentaje del tiempo total invertido en cada iteración. Fuente: propia.

Como se comenta en la sección 6.2, la iteración que ha llevado más tiempo es la cuarta, que completa el desarrollo de la aplicación real mientras que la iteración más “corta” es la primera ya que se corresponde con la fase inicial del proyecto.

Se muestra también la carga de trabajo a lo largo de los meses de realización del proyecto en la Figura 11, mostrando además el tiempo empleado en cada iteración.

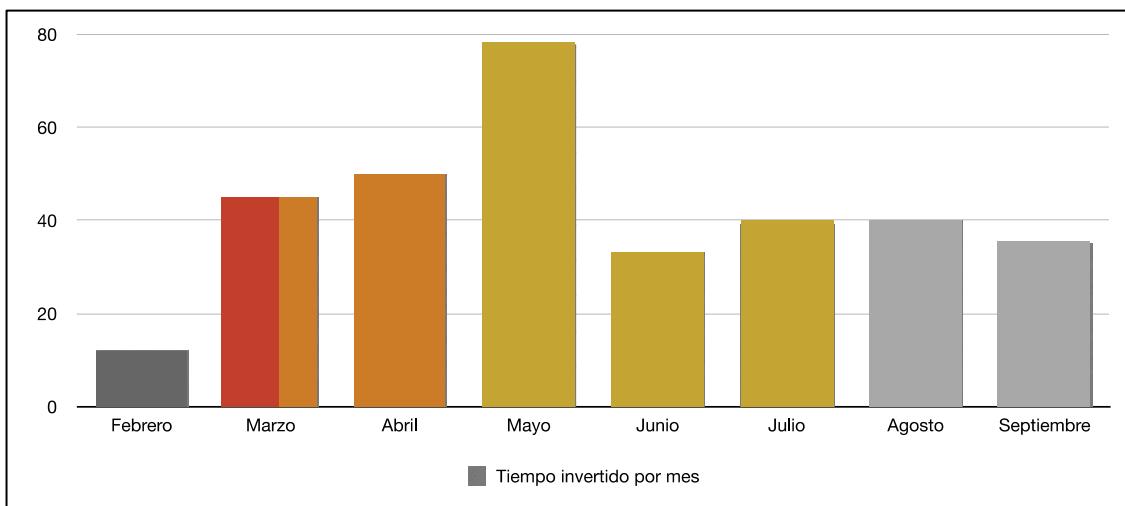


Figura 11. Tiempo invertido por mes (en horas). Fuente: propia.

6.4 Herramientas de gestión utilizadas

Con el objetivo de aplicar la metodología basada en iteraciones incrementales y de llevar un cierto control del estado del proyecto, se han utilizado una serie de herramientas de gestión de proyectos que han facilitado el trabajo a realizar. Con la intención de organizar las tareas a realizar y los períodos de tiempo necesarios para su desarrollo, se ha usado la herramienta TeamGantt¹⁵. Este software permite, de forma online, convertir los requisitos

¹⁵ <https://www.teamgantt.com/>

en tareas a realizar, asignar las tareas a diferentes personas (en este caso particular todas las tareas recaen sobre la misma persona), comprobar el estado de estas, modificar las fechas si existe algún tipo de contratiempo, etc.

Para organizar la documentación como la propuesta del proyecto o un borrador de la actual memoria técnica, se hace uso de Google Drive y se cuenta con una carpeta compartida con el docente. De forma incremental, los documentos se van llenando y se va recibiendo retroalimentación, de tal forma que se puede cambiar fácilmente el contenido en caso de existir algún error. Se ha escogido Google Drive por la facilidad que existe a la hora de trabajar y de compartir el contenido y también por el control de versiones que ofrece. Cada documento tiene un nombre descriptivo de su contenido. No obstante, una vez se tiene una memoria suficientemente completa, se ha usado Microsoft Word para redactar la memoria final ya que se considera una herramienta capaz de producir un resultado más profesional y acorde a este trabajo.

Para almacenar y mantener el código se utilizó la plataforma GitHub. Este entorno no solamente sirve como repositorio, sino que también se usa para tener un control de versiones. Dignas de mención son las denominadas *issues*, notas que se pueden compartir para señalar que hay un error en el código o una tarea pendiente y que han resultado ser bastante útiles ya que son una forma rápida y eficaz de mantener una conversación con el docente acerca de futuras tareas, del código o de la fecha de siguiente reunión, etc. Para la comunicación entre docente y alumno no solamente se utilizó GitHub, sino que el correo electrónico tuvo bastante peso también, alternando ambas según las necesidades específicas del momento.

7 Conclusiones

Este último apartado aporta una serie de conclusiones que se han obtenido del proyecto y se consideran importantes. La sección 7.1 resume el resultado del proyecto, mientras que la sección 7.2 resalta las lecciones aprendidas. La siguiente sección expone una serie de futuras mejoras que se consideran interesantes para mejorar la calidad del proyecto. Finalmente, la sección 7.4 analiza desde un punto de vista personal el proyecto, aportando una valoración más personal.

7.1 Resultado del proyecto

Se considera que los resultados del proyecto han sido satisfactorios. Se ha conseguido desarrollar el sistema que se planteó al inicio, cubriendo los requisitos propuestos. Se ha conseguido diseñar una configuración completa del crawler que permite obtener, analizar y filtrar los documentos del MAPA. Se ha conseguido desarrollar una aplicación web visual y sencilla con facetas donde el usuario puede navegar con facilidad, controlando y restringiendo el acceso a ciertos tipos de funcionalidades. La API desarrollada es consistente y utiliza correctamente los métodos HTTP. También, se ha conseguido usar diferentes tipos de tecnologías, ampliando los conocimientos que se tenían al inicio de este proyecto. Teniendo todo esto en cuenta, se considera que se han cumplido los objetivos propuestos.

7.2 Lecciones aprendidas

Durante el desarrollo del proyecto se han aprendido numerosas lecciones. Se pueden considerar como éxitos del proyecto el haber seguido una metodología de trabajo basada en iteraciones incrementales (con la posibilidad de volver al paso anterior en caso de fallo), la incorporación de una amplia fase de análisis que ha concluido en el correcto uso de las tecnologías escogidas (debido a la fuerte integración entre ellas) y, sobre todo, la meticulosa organización y planificación seguida.

De la misma forma, se han detectado también algunos errores de los que se ha aprendido. Por ejemplo, durante el desarrollo de la aplicación web, algunas de las funcionalidades tardaron más tiempo del esperado en completarse, lo que ocasionó un desplazamiento en el cronograma causando un error de planificación. Otra lección aprendida ha sido la organización del código, ya que al principio se escribía sin ningún tipo de estructura (un error), hasta que se consultó en detalle el manual de programación de React y se modularizó todo el código facilitando su lectura y reusabilidad.

7.3 Futuras mejoras

El proyecto tiene una serie de limitaciones que, en el caso de ser extendidas, provocarían una mejora al sistema siendo este más complejo y ambicioso. Una de las limitaciones es que el proceso de extracción se lleva a cabo localmente, en la máquina del autor. Si se tuviesen varias máquinas se podría aprovechar la potencia distribuida de Nutch (con Hadoop) y el proceso sería bastante más ligero ya que el trabajo se repartiría entre las distintas máquinas disponibles. Además, con esta mejora, el sistema global sería más potente y capaz de almacenar mucho más contenido del MAPA, lo que aumentaría la cantidad de información presente en el sistema. De esta forma, también se estaría más seguro de la información que no es relevante y que debe ser eliminada.

Asimismo, el sistema está enfocado en el portal web del Ministerio y, por tanto, todo el proceso está íntimamente ligado a los metadatos e información específica del mismo. Esto es una limitación ya que, si por cualquier motivo éstos cambiaseen o se quisiese replicar el proceso, pero para otro portal distinto, sería necesario revisar el sistema.

Otra mejora consistiría en conseguir que todo el proceso fuese automático. Por ejemplo, si ahora se eliminase todo el contenido de las bases de datos, sería muy útil tenerlo todo automatizado para que simplemente ejecutando un comando se realizase la extracción, el postproceso y se lanzase la aplicación web para usarse. Esto sería útil ya que el portal del MAPA está continuamente creando nuevo contenido y modificando el que ya existe, por lo que tener un proceso automático que se fuese ejecutando, por ejemplo, semanalmente, ayudaría a enriquecer mucho la información del portal.

También, como este tema es tan extenso, siempre se podría mejorar el sistema de consulta añadiendo más facetas, mejorar la funcionalidad de edición con la posibilidad de editar más campos, etc.

7.4 Valoración personal del proyecto

Si bien es cierto que el trabajo de fin de grado es un proyecto relativamente grande y ambicioso, éste es también una fuente brutal de aprendizaje. Realmente, es la primera toma de contacto con un proyecto “similar” a uno que podría realizarse en la vida laboral. El hecho de tratarse de un trabajo individual y contar únicamente con el apoyo del docente, asusta al principio. Sin embargo, en mi caso, la constancia y la capacidad de apoyo en el profesor ha sido clave para poder cumplir los objetivos.

En mi opinión, el proyecto tiene una serie de limitaciones, pero cumple con los objetivos principales y secundarios. Quizá el objetivo secundario más duro a priori, es el hecho de conseguir una buena organización y planificación. Sin embargo, el uso de herramientas adecuadas de gestión ha jugado un rol realmente importante en este tema. Otro aspecto que me causaba incertidumbre al principio era el uso de nuevas tecnologías nunca vistas. No obstante, excepto Nutch (que tiene una documentación algo antigua), las demás tecnologías están fantásticamente bien documentadas y eso ha ayudado mucho a la hora del desarrollo y aprendizaje de éstas.

En definitiva, teniendo todo en consideración, al tratarse de un proyecto muy alineado con la rama que he cursado (Sistemas de la información), la propia fase de diseño e implementación ha sido realmente una forma de reflejar todos los conocimientos adquiridos a la par de absorber otros muchos completamente novedosos. Por esto mismo, considero que se ha conseguido construir un producto lo más parecido posible al sistema que se tenía en mente.

8 Bibliografía

- [1]: *¿Qué es un web crawler? Cómo las arañas web optimizan Internet.* (2020, 21 octubre). IONOS Digital Guide. <https://www.ionos.es/digitalguide/online-marketing/marketing-para-motores-de-búsqueda/que-es-un-web-crawler/>
- [2]: Institute of Technology, Nirma University, Dvijesh Bhatt, D. D. B., Daiwat Amit, V. D. A., & Sharnil, P. S. (2015, abril). *Focused Web Crawler* (N.º 11). Krishisankriti Publications. https://www.krishisankriti.org/vol_image/21Jul201512071432%20%20%20%20%20%20D_AIWAT%20A%20%20VYAS%20%20%20%20%20%201-6.pdf
- [3]: Micarelli, A., Gasparetti, F., Sciarrone, F. & Gauch, S. (2007). *Personalized Search on the World Wide Web*. SpringerLink. Recuperado 14 de septiembre de 2022, de https://link.springer.com/chapter/10.1007/978-3-540-72079-9_6?error=cookies_not_supported&code=2de48fed-cf77-47d8-a490-849eff0f0752
- [4]: *Just a moment. . .* (s. f.-b). Recuperado 12 de septiembre de 2022, de <https://www.sciencedirect.com/science/article/abs/pii/S1389128699000523>
- [5]: Jiménez, S. (2021, 16 noviembre). *Los problemas de los buscadores internos de las webs públicas*. Sergio Jiménez. <https://sergiojimenez.net/los-problemas-de-los-buscadores-internos-de-las-webs-publicas/>
- [6]: *A European Strategy for data*. (2022, 5 julio). Shaping Europe's Digital Future. <https://digital-strategy.ec.europa.eu/en/policies/strategy-data>
- [7]: Pereda, T. (2020, 15 mayo). *5 ventajas de contar con un sitio web optimizado con una estrategia SEO*. Lemon Digital Marketing. <https://lemon.digital/estrategia-seo-ventajas/>
- [8]: Ángel, M. (2021, 28 enero). *ElasticSearch y Logstash*. El blog de MagMax. <https://magmax.org/blog/elasticsearch-y-logstash/>
- [9]: Berman, D. (2020, 29 octubre). *Elasticsearch API 101*. Logz.Io. <https://logz.io/blog/elasticsearch-api/>
- [10]: Arquitecturas Multicapa. (s.f.). <http://elvex.ugr.es/decsai/csharp/design/layers.xml>
- [11]: Nagel, S. N. (2020, 15 agosto). *AboutPlugins - NUTCH - Apache Software Foundation*. Nutch Confluence. <https://cwiki.apache.org/confluence/display/nutch/AboutPlugins>
- [12]: *Tipos MIME - HTTP / MDN*. (2022, 14 agosto). Mdn Web Docs. https://developer.mozilla.org/es/docs/Web/HTTP/Basics_of_HTTP/MIME_types
- [13]: Accesibilidad. (s.f.). React. <https://es.reactjs.org/docs/accessibility.html>
- [14]: *RFC 2616 - Hypertext Transfer Protocol -- HTTP/1.1*. (1999, 20 junio). Hypertext Transfer Protocol. <https://datatracker.ietf.org/doc/html/rfc2616#section-9.3>

- [15]: Martínez, S. M. (2014, 14 diciembre). Metodología iterativa o incremental en la gestión de proyectos. mundoerp. <https://www.mundoerp.com/blog/metodologia-iterativa-o-incremental-gestion-proyectos/>
- [16]: White, T. (2009). *Hadoop: The Definitive Guide*. Van Duuren Media.
- [17]: Mobomo, L. (2018, 3 mayo). *The Basics: Working with Nutch - Mobomo, LLC*. Medium. <https://medium.com/@mobomo/the-basics-working-with-nutch-e5a7d37af231#:~:text=Nutch's%20crawl%20cycle%20is%20divided,web%20and%20scrape%20each%20URL>
- [18]: Abueg, R. (2020, 22 septiembre). *Elasticsearch: What it is, How it works, and what it's used for*. Knowi. Recuperado 5 de septiembre de 2022, de <https://www.knowi.com/blog/what-is-elasticsearch>
- [19]: *org.apache.nutch.plugin (Endeca Web Crawler 11.1.0)*. (2014, 9 junio). Recuperado 5 de septiembre de 2022, de https://docs.oracle.com/cd/E55325_01/CAS.110/apidoc/web-crawler-javadoc/org/apache/nutch/plugin/package-summary.html
- [20]: colaboradores de Wikipedia. (2022, 16 junio). *JSON Web Token*. Wikipedia, la enciclopedia libre. Recuperado 7 de septiembre de 2022, de https://es.wikipedia.org/wiki/JSON_Web_Token
- [21]: Morgan, J. (2020, 19 mayo). *Cómo crear componentes personalizados en React*. DigitalOcean Community. Recuperado 6 de septiembre de 2022, de <https://www.digitalocean.com/community/tutorials/how-to-create-custom-components-in-react-es>
- [22]: Patricio, A. (2020, 28 octubre). *Componentes en React*. Recuperado 7 de septiembre de 2022, de <https://somospnt.com/blog/177-componentes-en-react>

Anexo A Análisis detallado

Este Anexo, al igual que todos los presentes en esta memoria, contiene información complementaria que debe estar presente ya que es un contenido relevante pero demasiado extenso para que se incluya en el cuerpo de la memoria.

Este Anexo en particular presenta evidencias del problema del MAPA, con ejemplos que apoyan este hecho. También detalla aspectos clave del análisis como el funcionamiento y estructura de ficheros de Nutch o el sistema de persistencia, aportando información de interés respecto a la naturaleza y lenguaje de consulta de este sistema. También se muestra un ejemplo de herramienta que permite visualizar la información una vez está almacenada. Finalmente, se cierra el análisis del sistema mostrando los requisitos finales al completo.

A.1 Evidencias del problema del MAPA

En esta sección se adjuntan capturas de pantalla del portal web del MAPA exponiendo ejemplos concretos de los problemas que tiene el buscador de este sitio. Como se muestra en la sección 2.1, la Figura 1 muestra el buscador de este portal web.

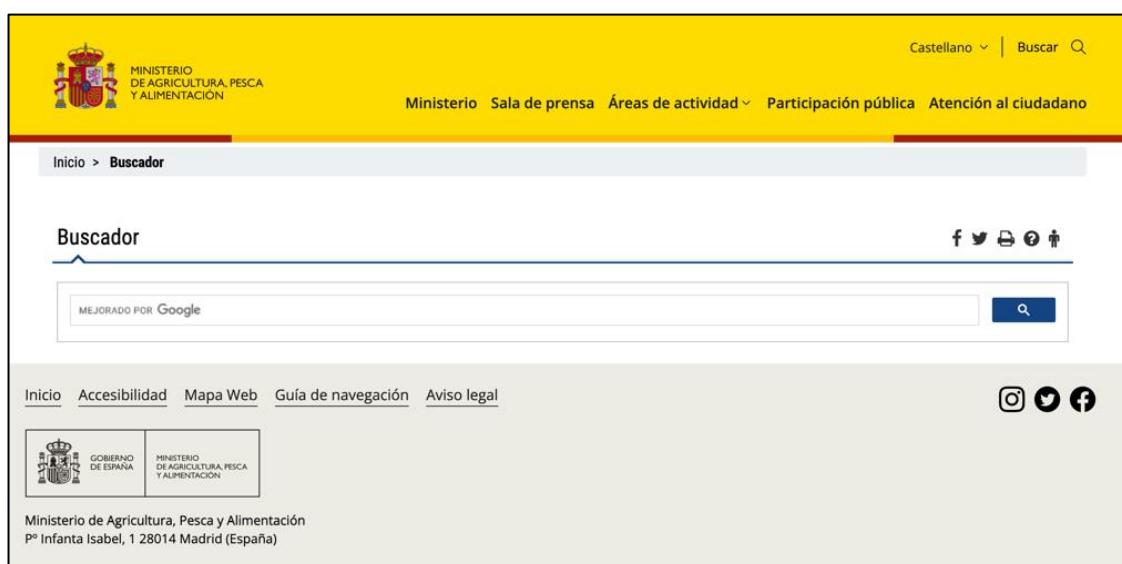


Figura 1. Buscador del portal web del MAPA. Fuente: <https://www.mapa.gob.es/es/buscador/>.

A parte de la mencionada carencia de filtros, que hace el proceso de búsqueda algo difícil e incómodo, otro problema detectado es el propio buscador optimizado por Google. A falta de filtros, si se hace una consulta precisa, como podría ser la obtención de documentos con formularios, el propio buscador es incapaz de mostrar esa información, presentando los resultados que se muestran en la Figura 2.

Buscador

formulario

Aproximadamente 1,870 resultados (0.21 segundos)

Ordenar por: Relevance ▾

Solicitudes y formularios
www.mapa.gob.es/registro-de-variedades/solicitudes-y-formularios

Solicitudes y **formulario**s · Solicitud de inscripción en el Registro de Variedades Comerciales · Solicitud de Autorización Provisional de Comercialización conforme ...

FORMULARIO DE INSPECCIÓN - Programa de Vigilancia Piloto de ...
www.mapa.gob.es/temas/4_formulariodeinspeccion_tcm30-111583

Formato de archivo: PDF/Adobe Acrobat
 17 jul 2012 ... **Formulario** de Inspección. 17 julio 2012. Índice. • Datos del Inspector. • Datos Generales del colmenar. • Investigación de las colmenas.

Productos pesqueros de terceros países. Importación y exportación
www.mapa.gob.es/productos-pesqueros-terceros-paises/control_acceso

Documentación necesaria para el acceso a puerto. **Formulario** de notificación previa. El capitán o representante del buque pesquero ...

Formulario 1 Gestcuotas
www.mapa.gob.es/pesca/temas/control-cuota/formulario-1-gestcuot...

Formato de archivo: PDF/Adobe Acrobat
Formulario 1. Datos necesarios para el alta de usuarios en WEB GESTCUOTAS. Remitir solicitudes a: bzn-asistencialic@mapa.es. Autorizo de manera voluntaria, ...

FORMULARIO CARTAMO
www.mapa.gob.es/agricultura/temas medios-de-produccion

Formato de archivo: Microsoft Word

Figura 2. Búsqueda de formularios en el MAPA. Fuente: <https://www.mapa.gob.es/es/buscador/>.

Los resultados no son los esperados. Se trata de documentos que contienen de alguna forma la palabra “formulario”, pero no son documentos que contengan formularios, es decir, posibles puntos de entrada a bases de datos.

Concretamente, se muestran en detalle en las siguientes Figuras los tres primeros resultados de la consulta mostrada en la Figura 2.

Castellano ▾ | Buscar 

MINISTERIO DE AGRICULTURA, PESCA Y ALIMENTACIÓN

Ministerio Sala de prensa Áreas de actividad ▾ Participación pública Atención al ciudadano

Inicio > Agricultura > Medios de producción > Oficina Española de Variedades Vegetales > Registro de variedades > **Solicitudes y formularios**

Oficina Española de Variedades Vegetales

- Registro de variedades
- Reg. de variedades comerciales
- Reg. de variedades protegidas
- Solicitudes y formularios**
- Composición de las Comisiones Nacionales
- Catálogos nacionales y comunitarios
- Registro de productores
- Otros catálogos de variedades
- Recursos fitogenéticos para la Agricultura y la Alimentación
- Producción, comercialización y certificación de plantas de vivero

Solicitudes y formularios

Solicitud de inscripción en el Registro de Variedades Comerciales

 [Solicitud de inscripción \(RVC\)-\(Actualizado marzo 2022\)](#)

Solicitud de Autorización Provisional de Comercialización conforme a la Decisión 2004/842/CE

 [Solicitud de APC de variedades de especies agrícolas](#)
 [Solicitud de renovación de APC de variedades de especies agrícolas](#)
 [Solicitud de APC de variedades de especies hortícolas](#)
 [Solicitud de renovación de APC de variedades de especies hortícolas](#)
 [Autorización provisional de comercialización \(APC\).Cantidad máximas de especies agrícolas para 2022, de acuerdo con la Decisión de la Comisión 2004/842/CE.](#)

Solicitud de inscripción en el Registro de Variedades Protegidas

 [Solicitud Título de Obtención Vegetal \(RVP\)-\(Actualizado marzo 2022\)](#)
 [Declaración relacionada con el cumplimiento de los requisitos de Novedad, Distinción, Homogeneidad y Estabilidad](#)

Novedades

 Análisis de la Realidad Productiva de Frutales de Hueso. 2020
Publicado el informe sobre las superficies de frutales de hueso en el año 2020
[+info](#)

Destacados

 Publicado avance de situación de mercado del sector oleícola 1,89 Mb
 Publicado balance de mercado de aceite de oliva, aceite de orujo de oliva y aceituna de mesa 681,28 Kb

Figura 3. Primer resultado de la búsqueda anterior. Fuente: <https://www.mapa.gob.es/>.

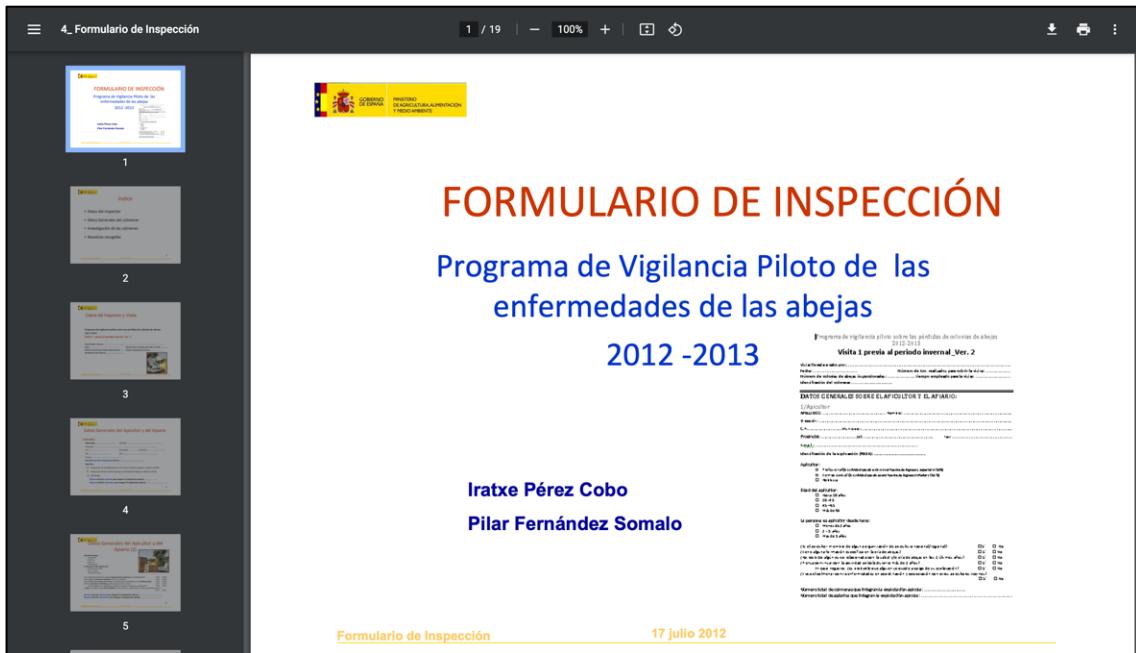


Figura 4. Segundo resultado de la búsqueda. Fuente: <https://www.mapa.gob.es/es/>.

Figura 5. Tercer resultado de la búsqueda. Fuente: <https://www.mapa.gob.es/es/>.

Efectivamente, como se puede comprobar, ninguna de las tres Figuras anteriores muestra documentos con formularios. A modo de ejemplo, la Figura 6 muestra un modelo de página web con un formulario. Este tipo de páginas web son las que deberían obtenerse al hacer la búsqueda que se muestra en la Figura 2.

Castellano | Buscar

Ministerio | Sala de prensa | Áreas de actividad | Participación pública | Atención al ciudadano

Inicio > Agricultura > Producciones agrícolas > Frutas y hortalizas

Temas <ul style="list-style-type: none"> Producciones agrícolas Cultivos herbáceos e industriales Frutas y hortalizas Patata Vitivinicultura Aceite de oliva y aceituna de mesa Flores y plantas ornamentales Producción integrada Producción ecológica Regulación de los mercados Pagos directos Medios de producción Biotecnología Sanidad vegetal 	<h3>Acceso aplicación</h3> <p>A V I S O</p> <p>Los datos de las organizaciones de productores y sus programas operativos correspondientes a 2016 deben enviarse obligatoriamente a través de la aplicación SOFYH, a la que accederán mediante el enlace:</p> <p>https://servicio.mapama.gob.es/sofyh</p> <p>Los datos de 2016 que se graben en ROPAS se considerarán no presentados y serán posteriormente borrados.</p> <p>Entrada aplicación</p> <p>Deberá introducir el usuario y contraseña para acceder a esta aplicación</p> <p>Acceso al sistema</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 15%;">Usuario</td> <td style="width: 85%;"><input type="text"/></td> </tr> <tr> <td>Contraseña</td> <td><input type="password"/></td> </tr> </table> <p style="text-align: right;">Enviar</p>	Usuario	<input type="text"/>	Contraseña	<input type="password"/>
Usuario	<input type="text"/>				
Contraseña	<input type="password"/>				

Figura 6. Ejemplo de formulario. Fuente: <https://www.mapa.gob.es/es/>.

Como se puede apreciar, esta página web sí que contiene un formulario y son precisamente este tipo de documentos los que deberían aparecer si se busca por un formulario.

Finalmente, la Figura 7 muestra la búsqueda “bases de datos MAPA” en el buscador de Google. El problema reside en que, en vez de mostrar las páginas con contenido similar al de una base de datos, muestran noticias con ese texto en ellas.

Castellano | Buscar

Ministerio | Sala de prensa | Áreas de actividad | Participación pública | Atención al ciudadano

Inicio > Agricultura > Medios de producción > Maquinaria agrícola > Base de datos

Maquinaria agrícola <ul style="list-style-type: none"> Ayudas Estación Mecánica Agrícola Demostraciones Estadísticas Ensayos de abonadoras Exenciones de uso de estructuras a tractores estrechos Estructuras para reformas de vehículos Inspecciones de equipos de aplicación de productos fitosanitarios Base de datos Tractores Estructuras Correlaciones Circulares 	<h3>Base de datos</h3> <p>Esta base de datos contiene todos los tractores agrícolas y sus estructuras de protección para su inscripción en los Registros Oficiales de Maquinaria Agrícola.</p> <p>En los tractores de homologación de tipo nacional, figura la potencia de inscripción de acuerdo en lo dispuesto en la Orden de 14 de febrero de 1964, por la que se establece el procedimiento de homologación de la potencia de los tractores agrícolas.</p> <p>Las estructuras de protección que figuran como homologación nacional, lo han sido mediante lo dispuesto en la Orden de 27 de julio de 1979, por la que se regula técnicamente el equipamiento de los tractores con bastidores y cabinas oficialmente homologados.</p> <p>Los tractores con homologación de tipo CE lo han sido mediante la aplicación de la Directiva 74/150/CEE, 2000/25/CE ó 2003/37/CE, que regulan la homologación de los tractores agrícolas y forestales.</p> <p>La consulta de la base de datos se puede hacer, tanto por las características de los tractores como de las estructuras y por las correlaciones entre ambas, de tal manera que aparecen todos los tractores que puedan llevar una determinada estructura y todas las estructuras de protección que puede montar cada modelo de tractor.</p>	<p>Novedades</p>  <p>Análisis de la Realidad Productiva de Frutales de Hueso. 2020</p> <p>Publicado el informe sobre las superficies de frutales de hueso en el año 2020</p> <p>+info</p> <p>Destacados</p> <p> Publicado avance de situación de mercado del sector oleícola 1,89 Mb</p> <p> Publicado balance de mercado de aceite de oliva, aceite de orujo de oliva y aceituna de mesa 681,28 Kb</p>
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Figura 7. Resultado de la búsqueda “base de datos MAPA” en Google. Fuente: <https://www.mapa.gob.es/es/>.

A.2 Funcionamiento y estructura de Apache Nutch

En este apartado del Anexo se muestra el funcionamiento en detalle de Nutch junto a la estructura de ficheros que posee. Una vez entendida la dinámica general de un *crawler*, se muestra a continuación el funcionamiento específico de Nutch.

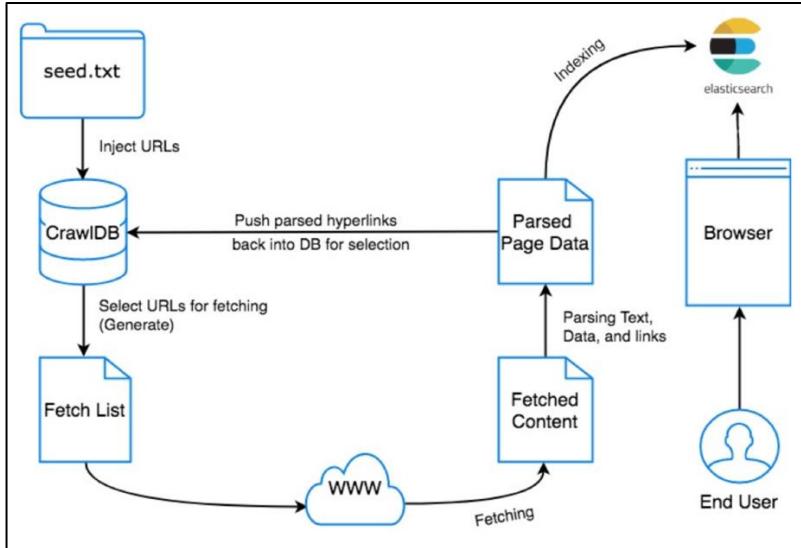


Figura 8. Funcionamiento de Nutch. Fuente: <https://github.com/satriawadhipurusa/nutch2-elasticsearch>.

A la vista de la información mostrada en la Figura previa, los diferentes pasos que sigue Nutch para la extracción de la información son los siguientes:

bin/nutch inject: el proceso de inyección lee las semillas del fichero “seed.txt” y las inyecta en la base de datos de URL, denominada “CrawlDB”. Esta base de datos interna de Nutch se encarga de almacenar el estado actual de cada URL, que se mapea en un par <url, DatosDeCrawl>. Esta base de datos interna se inicializa al principio con las URL que se inyectan del fichero de “seed.txt” pero se va actualizando con la información que ofrecen las páginas extraídas [16].

bin/nutch generate: el siguiente paso es seleccionar un subconjunto de URL de la base de datos de Nutch (CrawlDB) y prepararlas para el proceso de obtención. Este subconjunto de URL forma el denominado segmento. En otras palabras, se seleccionan las URL que van a continuar el flujo de trabajo y van a pasar a la fase de obtención de su información [17].

bin/nutch fetch: aquí es donde ocurre lo más relevante. En este paso, la información de las URL seleccionadas en el proceso anterior se almacena en segmentos (conjunto de URL). En estos segmentos se guarda la información de cada URL, es decir, su contenido [17].

bin/nutch parse: una vez se obtiene el contenido de las URL, éstas pasan por un proceso de análisis donde se lee la información verdaderamente útil, se descarta información no relevante y se obtienen los enlaces externos que en un futuro pasarán también por este proceso [17].

bin/nutch **updatedb**: los nuevos links externos que se han extraído como fruto del proceso anterior, se añaden a la base de datos “CrawlDB” para que, posteriormente, sean parte del proceso de crawl [\[17\]](#).

bin/nutch **index**: finalmente, el contenido que acabamos de extraer se indexa en algún motor de búsqueda (comúnmente ElasticSearch o Apache Solr) para que se pueda buscar posteriormente. Es decir, todo el contenido que se ha extraído se escribe en una base de datos documental (no relacional) para su futura búsqueda [\[17\]](#).

Nutch generalmente trabaja por bucles o iteraciones, esto es, todos los comandos anteriores se ejecutan secuencialmente una determinada cantidad de veces, las que se especifique. Como es normal, cuantas más iteraciones se realicen, más contenido se extraerá de la web. Una iteración completa supone la ejecución de los seis comandos anteriores, uno detrás de otro.

Para hacer esto de forma automática, Nutch ofrece un *script* (“bin/crawl”) que lleva a cabo precisamente esta tarea y que recibe como argumento el número de iteraciones deseadas junto al directorio que contiene el fichero “seed.txt” que sirve de punto de partida del proceso de crawl. Se muestra en la Figura 9 un ejemplo de este comando:

```
MacBook-Pro-de-Pablo:local pablojordan$ bin/crawl -i -s urls crawl 3
```

Figura 9. Ejemplo del comando bin/crawl. Fuente: propia.

El comando anterior realiza un proceso de extracción de tres iteraciones, ya que es la cantidad especificada al final del comando. Se usa la opción -i para forzar que Nutch indexe el contenido (no lo hace por defecto) en el soporte configurado y la opción -s para especificar cuál es el directorio que contiene el fichero “seed.txt”, en este caso, el directorio “urls”. Finalmente, se establece el nombre del directorio en el cual se almacenará toda la información de crawl (la base de datos interna “CrawlDB” y los segmentos), en este caso, el directorio “crawl”.

No obstante, para acabar de entender el funcionamiento de Nutch, es necesario conocer su estructura de ficheros:

```
runtime/local
  └── bin
  └── conf
  └── crawl
  └── lib
  └── logs
  └── plugins
  └── urls (u otro nombre)
    └── seed.txt
```

En el directorio “bin” se encuentran dos *scripts*: “bin/nutch” y “bin/crawl”, el cual se ha expuesto anteriormente. El primero permite ejecutar los comandos mostrados anteriormente (inject, generate, parse...) de forma individual mientras que el segundo engloba todo el proceso en un único comando. Simplemente, deberemos pasar como parámetros del comando el directorio que contiene el fichero de las URL semilla y el número de veces que deseamos ejecutar el proceso.

El directorio “conf” es muy importante. Aquí encontramos todos los ficheros de configuración de Nutch, ficheros con extensión “xml” y “txt” en su mayoría. En particular, el fichero de configuración central sobre el que Nutch basa su comportamiento es el “nutch-site.xml”. Este fichero tiene una serie de propiedades que, en caso de añadirse, cambian el comportamiento de Nutch. Se puede consultar el Anexo C.1 para visualizar la configuración completa de Nutch usada para el proceso de extracción.

El directorio “crawl” contiene la base de datos con la información referente a las URL con las que, a posteriori, se trabajará. Los directorios “lib” y “logs” contienen los JAR e información de registro, respectivamente. Por otro lado, el directorio “plugins” contiene toda funcionalidad que se puede añadir/quitar al proceso de extracción junto a la carpeta de pruebas. Existen *plugins* de indexación, de “scoring”, de “parsing”, etc. Más adelante se mostrarán los *plugins* utilizados y las modificaciones realizadas.

Finalmente, el directorio “urls” contiene un único fichero “seed.txt” con el contenido de las URL semilla. En particular, este es el lugar donde se colocan los dominios que conforman el punto de entrada del sistema y que permiten convertir a Nutch en un *crawler* enfocado.

A.3 Sistema de persistencia

ElasticSearch, el subsistema de persistencia, es el componente principal del sistema de persistencia y procesamiento. En esencia, ElasticSearch es un servidor que procesa peticiones JSON¹⁶ y devuelve como respuesta otro objeto JSON [18]. Con un mayor nivel de detalle, ElasticSearch es un motor de búsqueda de código abierto, construido sobre Apache Lucene y escrito en Java. Pese a ser un motor de búsqueda, ElasticSearch también cuenta con una base de datos NoSQL para almacenar los documentos en estructura JSON. Estos documentos pueden consultarse, actualizarse y eliminarse gracias a las múltiples API REST que presenta. Además, las respuestas que ofrece a las peticiones son muy rápidas debido a que no busca el texto directamente, sino que busca en el índice, construido expresamente para ello.

Un índice [18] es un conjunto de documentos que contienen características similares. En el ámbito del problema será necesario un único índice ya que los documentos que se extraen presentan características similares. Sin embargo, en un contexto distinto, como por ejemplo en una librería, quizás sea de interés tener varios índices: uno para libros, otro para revistas, otro para cómics, etc. Entonces, cuando se almacene información en ElasticSearch, se estará llevando a cabo una escritura en el índice. En particular, el último paso del proceso de extracción consiste en indexar (almacenar) los documentos recuperados en un índice.

Como ya se ha mencionado, una vez la información está indexada, ElasticSearch ofrece un conjunto de API REST para realizar consultas sobre esta información. Hay API para realizar cualquier consulta, es decir, existen API para realizar búsquedas, para incluir *scripts* en peticiones, para consultar el estado del servidor y del índice, etc. En este proyecto se usarán las API de búsqueda y de actualización.

¹⁶ <https://www.nextu.com/blog/que-es-json/>

Para consultar el API, ElasticSearch cuenta con su propio lenguaje de consultas basado en JSON denominado *Query DSL*. Se muestra en la Figura 10 un ejemplo de consulta DSL.

```
GET /_search
{
  "query": {
    "query_string": {
      "query": "(new york city) OR (big apple)",
      "default_field": "content"
    }
  }
}
```

Figura 10. Ejemplo de consulta DSL. Fuente: <https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-query-string-query.html>.

Como se puede apreciar en la imagen, la consulta está basada íntegramente en JSON. En este caso, se realiza una petición GET (petición para obtener documentos) al API de búsqueda `_search`. La consulta quiere obtener aquellos documentos que tengan en el campo específico “content” del documento, las palabras “new york city” o “big apple”. Como se puede observar, hay varias palabras clave que se usan, como es el ejemplo de “query” (campo que representa la consulta) o “default_field” (campo al que la consulta ataca), pero hay muchas más. La posibilidad de concatenar y agrupar diferentes palabras clave permite realizar consultas muy ambiciosas y complejas de una forma compacta.

De forma resumida y esquematizada, lo que está ocurriendo al hacer la petición es lo que se muestra en la Figura 11.

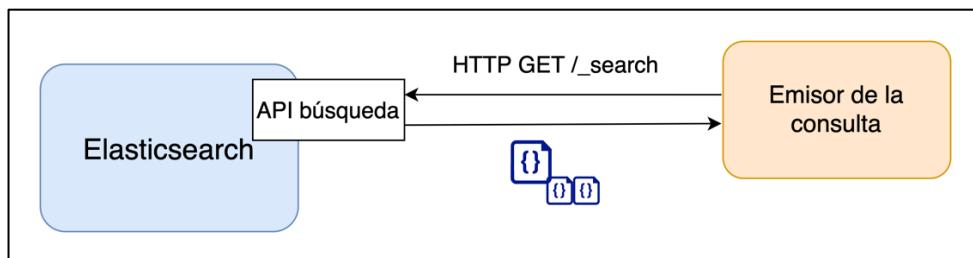


Figura 11. Operativa de una consulta DSL de Elasticsearch. Fuente: propia.

Se emite la consulta DSL a una de las API REST que ofrece ElasticSearch, en concreto a la API de búsqueda por medio de una petición HTTP GET. El motor de búsqueda procesa la petición y finalmente, devuelve los documentos JSON que satisfacen la búsqueda.

En definitiva, a diferencia de las bases de datos relacionales tradicionales que se consultan usando el lenguaje SQL, ElasticSearch utiliza el lenguaje específico DSL para realizar cualquier tipo de consulta a la base de datos documental (índice).

A.4 Visualización de la información

Kibana es una herramienta de visualización y exploración de la información contenida en el índice de ElasticSearch en tiempo real. Esta herramienta se usa para comprobar que la información recuperada es la correcta y esperada. La visualización de la información se realiza gracias a su interfaz gráfica, donde existen numerosas opciones que se pueden consultar (usuarios, datos, gráficas, etc.). Un ejemplo de la interfaz es la que se muestra en la Figura 12.

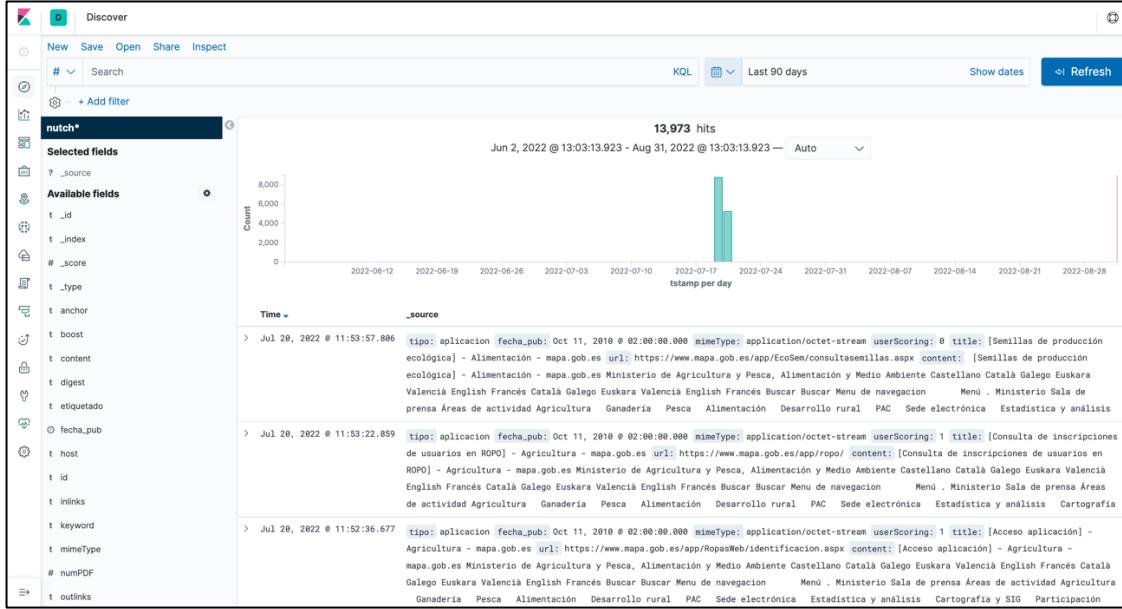


Figura 12. Interfaz gráfica de Kibana. Fuente propia.

Se puede observar como a la izquierda se encuentran todos los menús u opciones disponibles. La opción elegida es la de visualización de los datos, en la que Kibana muestra en la parte superior central la cantidad de documentos existentes en el índice junto a una gráfica que resalta el momento temporal en el que se realizó la inserción de esos documentos. A continuación, ofrece una lista de todos los documentos existentes, mostrando resumidamente algunos de sus atributos. La Figura 13 muestra un documento (el primero de la lista de la Figura 12) al completo.

t _id	https://www.mapa.gob.es/app/EcoSem/consultasemillas.aspx
t _index	nutch
# _score	-
t _type	_doc
t content	> [Semillas de producción ecológica] - Alimentación - mapa.gob.es Ministerio de Agricultura y Pesca, Alimentación y Medio Ambiente Castellano Català Gallego Frickars t digest
t etiquetado	true
⌚ fecha_pub	Oct 11, 2010 @ 02:00:00.000
t host	www.mapa.gob.es
t id	https://www.mapa.gob.es/app/EcoSem/consultasemillas.aspx
t keyword	alimentacion
t mimeType	application/octet-stream
☒ t outlinks	> https://www.mapa.gob.es/es/sistema/_bin/css/bootstrap-custom.css, https://www.mapa.gob.es/es/sistema/_bin/css/owl-carousel-custom.css, https://www.mapa.gob.es/es/sistema/_bin/css/font-awesome-custom.css, https://www.mapa.gob.es/es/sistema/_bin/css/nested-acordeon-custom.css, https://www.mapa.gob.es/es/sistema/_bin/css/prettyPhoto-custom.css, https://www.mapa.gob.es/es/sistema/_bin/style-magrama-rwd.css, https://www.mapa.gob.es/es/sistema/_bin/shims/mediaelementplayer.css, https://www.mapa.gob.es/es/sistema/_bin/css/header.css, https://www.mapa.gob.es/es/sistema/_bin/css/footer.css, https://www.mapa.gob.es/comunes/js_comunes/menu1.js, https://www.mapa.gob.es/es/comunes/js_comunes/ls_comunes.js, https://www.mapa.gob.es/sistema/_bin/js/jquery-3.3.1.js, https://www.mapa.gob.es/es/sistema/_bin/ie/owl_carousel_min_ie https://www.mapa.gob.es/eiectama/ie/owl_carousel_min_ie https://www.mapa.gob.es/eiectama/
t search	[Semillas de producción ecológica] - Alimentación - mapa.gob.es
t tipo	aplicacion
t title	[Semillas de producción ecológica] - Alimentación - mapa.gob.es
⌚ timestamp	Jul 20, 2022 @ 11:53:57.806
t url	https://www.mapa.gob.es/app/EcoSem/consultasemillas.aspx

Figura 13. Documento al detalle. Fuente propia.

Se puede observar como un documento consta de un identificador, su URL, la fecha de publicación, el servidor al que pertenece ese documento, los documentos a los que apunta, su contenido, etc. Gracias a esta herramienta, se puede comprobar fácilmente si los valores de los atributos de un documento extraído son los que deberían ser. En caso de ser así, se continuaría con la siguiente fase. En caso contrario, se debe volver al paso anterior e indexar de nuevo los documentos.

A.5 Requisitos del sistema

Como resultado del proceso de análisis, se definen los requisitos (al completo) que debe cumplir el sistema, perteneciendo cada uno de ellos a una zona de aplicación distinta. Los requisitos funcionales totales que se han identificado se recogen en la tabla mostrada a continuación.

Tabla 1. Requisitos funcionales del sistema.

Código	Zona de aplicación	Descripción del requisito
RF-1	Extracción	El sistema debe extraer únicamente documentos procedentes del dominio del Ministerio de Agricultura, Pesca y Alimentación.
RF-2	Extracción	El sistema debe almacenar como mínimo información del título, contenido, fecha de publicación, tipo MIME [12] y tipo para cada documento.
RF-3	Extracción	El sistema debe indexar los documentos JSON extraídos en el índice de ElasticSearch.

Código	Zona de aplicación	Descripción del requisito
RF-4	Persistencia y proc.	El sistema debe poder buscar los documentos mediante consultas DSL al índice.
RF-5	Persistencia y proc.	El sistema debe contar con un conjunto de procesos que manipulen los documentos para añadir atributos que sean de valor.
RF-6	Consulta	Existen dos tipos de usuario: administrador y usuario normal.
RF-7	Consulta	Todo usuario podrá buscar documentos introduciendo su consulta en una caja de búsqueda.
RF-8	Consulta	Todo usuario visualizará una lista con los resultados (paginados) que se ajusten más a la consulta/filtro.
RF-9	Consulta	El usuario podrá filtrar por fecha de publicación (un rango de fechas).
RF-10	Consulta	El usuario podrá filtrar por formato de documento. Ejemplo: si el usuario desea obtener únicamente los PDF, podrá hacerlo.
RF-11	Consulta	El usuario podrá filtrar por tipo de ficheros que contiene un documento. Ejemplo: si el usuario desea obtener los documentos que contienen XML y DOCX, podrá hacerlo.
RF-12	Consulta	El usuario podrá filtrar por tipo de documento. Ejemplo: si el usuario desea obtener documentos que sean un contenido o un formulario, podrá hacerlo.
RF-13	Consulta	El usuario podrá filtrar por relevancia del documento.
RF-14	Consulta	El usuario podrá filtrar por categoría del documento (agricultura, ganadería, pesca, etc.).
RF-15	Consulta	El usuario podrá iniciar sesión si tiene una cuenta, registrarse si aún no la tiene o entrar como invitado.
RF-16	Consulta	Si el usuario considera que un documento es muy relevante, es decir, toda la información expuesta es correcta y en base a sus criterios de búsqueda, podrá añadir un punto de relevancia al documento sirviendo esto de “garantía de calidad” frente al resto de documentos si su rol se lo permite.
RF-17	Consulta	Si el usuario considera que algún campo editable de un documento no es correcto, podrá editarlo a placer si su rol se lo permite.
RF-18	Consulta	El usuario administrador podrá acceder a un panel de administración (Dashboard) donde tendrá información de todos los usuarios del sistema junto a su rol, que podrá modificar.

De la misma forma, los requisitos no funcionales totales que se han identificado son los siguientes:

Tabla 2. Requisitos funcionales del sistema.

Código	Zona de aplicación	Descripción del requisito
RNF-1	Extracción	El sistema debe finalizar el proceso de extracción en un tiempo razonable, pero sin exceder los límites de peticiones ya que, en ese caso, la IP puede incluirse en una “blacklist”.

Código	Zona de aplicación	Descripción del requisito
RNF-2	Persistencia y proc.	El subsistema de persistencia debe estar compuesto de una base de datos documental para almacenar en JSON la información de los documentos del MAPA.
RNF-3	Consulta	El portal web deberá ser fácil de usar por parte del usuario para que no se haga complicado el proceso de búsqueda.
RNF-4	Consulta	El portal web deberá contar con un juego de colores homogéneo a lo largo de todas las pantallas. Además, cada tipo fichero se identificará con un color.
RNF-5	Consulta	El sitio web deberá poderse ejecutar desde los principales navegadores web.
RNF-6	Consulta	El sitio web deberá contar con un archivo de Log que recoja eventos importantes del sistema como las causas que han llevado al sistema a un estado de error.

Anexo B Diseño detallado

Este Anexo hace referencia a todo aquello relacionado con el proceso de diseño del sistema. Se muestra en primer lugar la arquitectura del sistema con más detalle, mostrando breves explicaciones de cuál es el comportamiento de cada componente del sistema. A continuación, se hace hincapié en el diseño del sistema de consulta mostrando cómo se navega por el sistema. Más adelante, se presentan los bocetos de las vistas de la aplicación web que permiten realizar esa navegación. La última sección presenta un diagrama de secuencia completo que ejemplifica un caso de uso habitual del sistema.

B.1 Arquitectura del sistema

Se puede observar en esta Figura la arquitectura del sistema con más detalle. Se muestran los tres sistemas y los diferentes componentes que tiene cada uno de ellos junto a las comunicaciones que siguen.

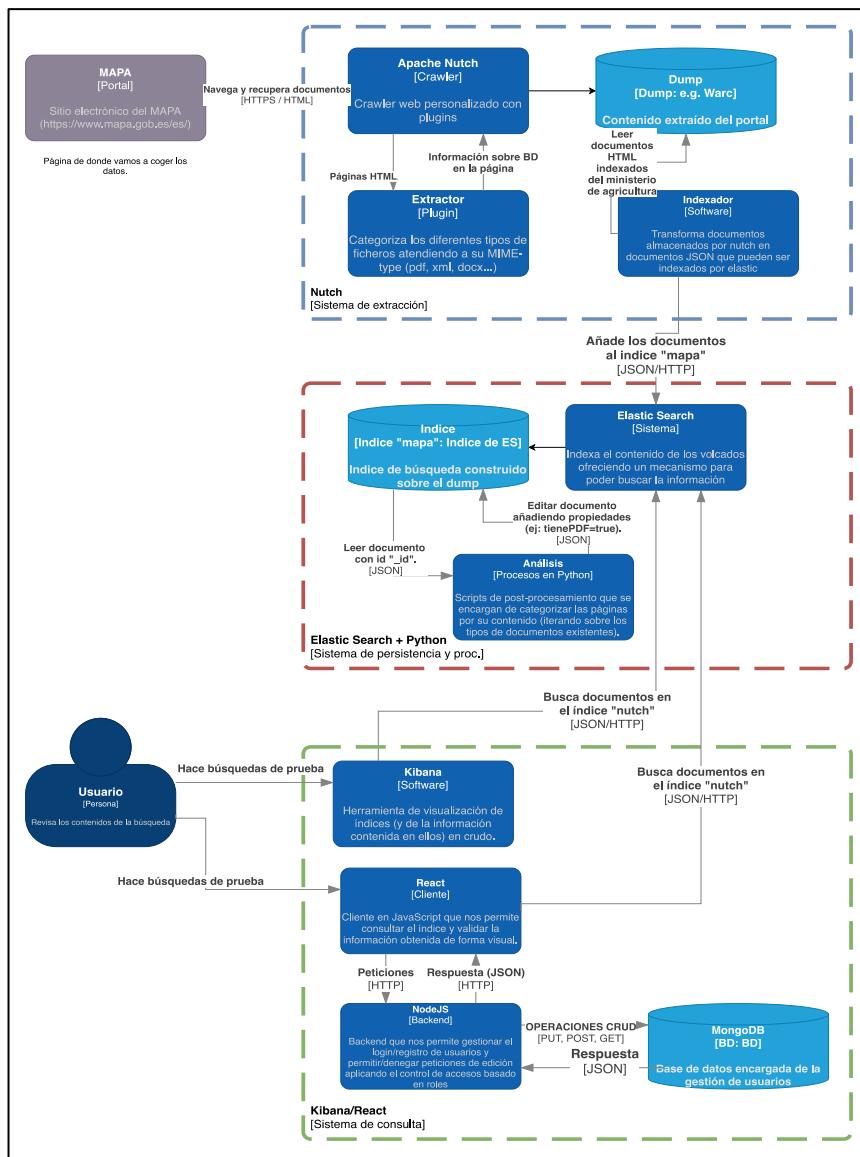


Figura 14. Arquitectura del sistema. Fuente: propia.

B.2 Mapa de navegación de la aplicación web

Se aprovecha esta sección para mostrar el mapa de navegación de la aplicación web. Todo usuario parte del inicio de sesión o registro. Una vez que un usuario convencional entra al sistema, puede navegar por la página principal del portal web. Cuando el usuario ya no necesite realizar más acciones, podrá cerrar sesión en cualquier momento, volviendo a la página de inicio de sesión. Ahora bien, si el inicio de sesión es por parte del administrador, entonces se abre el mismo portal web, pero con la posibilidad de acceso al panel de administración, donde este puede visualizar información privilegiada e incluso cambiar el rol de un usuario. De la misma forma que ocurre con un usuario convencional, el administrador podrá cerrar sesión en cualquier momento.

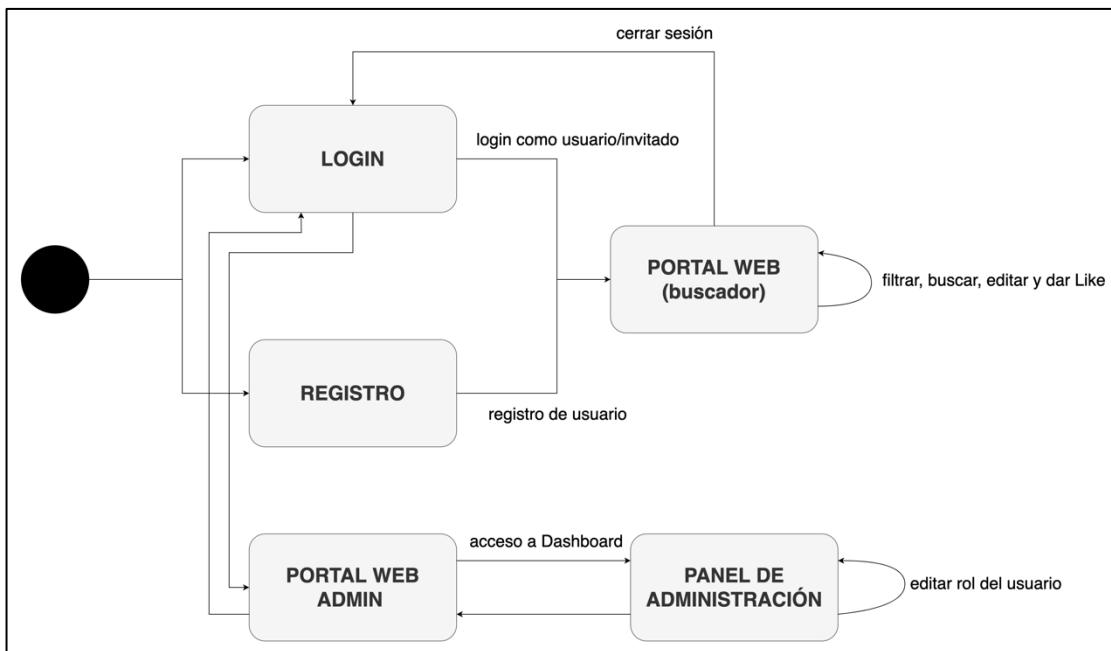


Figura 15. Mapa de navegación de la aplicación web. Fuente: propia.

B.3 Prototipo de la aplicación web (sistema de consulta)

Esta sección muestra el diseño del prototipo de la aplicación web. Se muestran, en primer lugar, las vistas del inicio de sesión y registro, que suponen el punto inicial del sistema. Como se trata sólo de un boceto, no se han puesto los mensajes de error que aparecerían si el usuario introdujese mal sus credenciales en el inicio de sesión o los mensajes de error que aparecerían si el usuario no cumpliese con los requisitos del registro.

Login - Focused crawler

http://focused-crawler.com/

Iniciar sesión

Correo electrónico

Contraseña

Entrar

— o —

Login como invitado

¿Todavía no tienes cuenta? [Regístrate](#)

Figura 16. Pantalla de inicio de sesión.

Registro - Focused crawler

http://focused-crawler.com/

Registrarse

Correo electrónico

Contraseña

Repita la contraseña

Registrarse

¿Ya tienes una cuenta? [Entrar](#)

Figura 17. Pantalla de registro.

A continuación, se muestra en la Figura 18 la pantalla principal con todas las facetas, tal y como se comenta en la sección 3.4. La Figura 19 muestra la vista de edición de un documento al hacer clic en el botón de Editar.

Focused crawler

<http://focused-crawler.com/>

Focused crawler endpoint

Username

Formatos

- pdf
- csv
- xml

Contiene

- PDF
- JSON

Tipo de página

- ...
- ...
- ...

Fecha de publicación

01/01/2022 – 01/09/2022

Categoría

- Agricultura
- Ganadería
- Pesca

Relevancia

0 5

Búsqueda

3 resultados encontrados

Datos 2020

src: ...
Relevancia: ★★☆☆☆
Fecha de publicación: 19/12/2021
[PDF \(4\)](#) [XML \(8\)](#) [Editar](#) [Like](#)

Información de consumo de leche y derivados

src: ...
Relevancia: ★★☆☆☆
Fecha de publicación: 26/12/2021
[PDF \(21\)](#) [XLSX \(11\)](#) [PPTX \(1\)](#) [Editar](#) [Like](#)

Solicitud de Ensayos

src: ...
Relevancia: ★★☆☆☆
Fecha de publicación: 04/02/2022
[MSWORD \(3\)](#) [Editar](#) [Like](#)

Figura 18. Pantalla principal del buscador.

Focused crawler

<http://focused-crawler.com/>

Focused crawler endpoint

Username

Formatos

- pdf
- csv
- xml

Contiene

- PDF
- JSON

Tipo de página

- ...
- ...

Fecha de publicación

01/01/2022

Categoría

- Agricultura
- Ganadería
- Pesca

Relevancia

0 5

Búsqueda

3 resultados encontrados

Editar documento

Título del documento:

Tipo de documento:

Ficheros que contiene:

[Cancelar](#) [Confirmar](#)

Solicitud de Ensayos

src: ...
Relevancia: ★★☆☆☆
Fecha de publicación: 04/02/2022
[MSWORD \(3\)](#) [Editar](#) [Like](#)

Figura 19. Pantalla de edición de un documento.

La Figura 20 muestra la vista de confirmación del aumento de relevancia de un documento del portal web (la misma pantalla para la confirmación de edición).

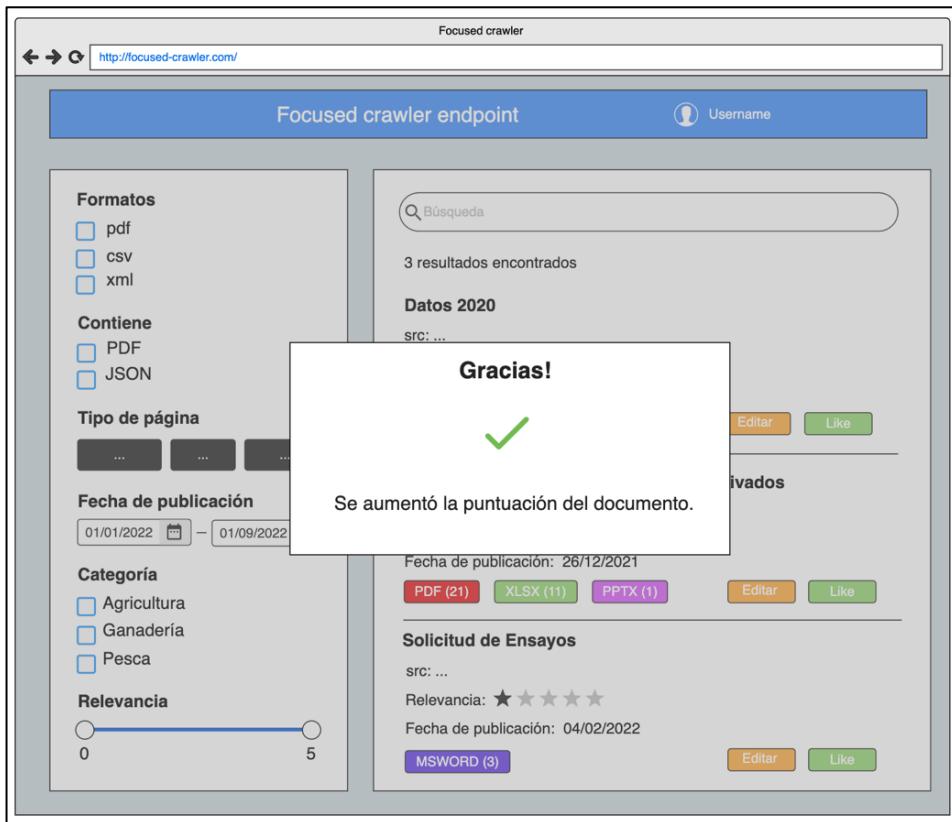


Figura 20. Pantalla de confirmación de la funcionalidad de «me gusta».

Esta Figura muestra el Dashboard del usuario administrador, donde este puede consultar la información de todos los usuarios disponibles en el sistema y cambiarles el rol.

Usuario	Fecha de registro	Rol	Editar rol
Ada Lovelace	2022-06-30	Editor	
Grace Hopper	2021-12-06	Lector	
Margaret Hamilton	2021-12-19	Lector	
Joan Clarke	2022-05-18	Lector	

Figura 21. Pantalla con el Dashboard del administrador.

La siguiente Figura muestra la vista de la modificación del rol de un usuario desde el panel de administración.

The screenshot shows a web-based administration interface for a 'Focused crawler' system. At the top, there's a header bar with a back/forward button, a refresh icon, and the URL 'http://focused-crawler.com/'. The main title is 'Focused crawler endpoint'. On the right, there's a 'Username' field with a user icon. Below the title, it says 'Home / Dashboard' and 'Cantidad de usuarios registrados: 5'. A table lists four users: Ada Lovelace, Grace Hopper, Margaret Hamilton, and Joan Clarke. Each row includes a 'Usuario' column, a 'Fecha de creación' column (with dates like 2021-12-06 and 2022-05-18), a 'Rol' column (with roles like 'Editor' and 'Lector'), and an 'Editar rol' column with edit icons. A modal window titled 'Nuevo rol para Ada Lovelace:' is open over the table, containing a dropdown menu labeled 'Seleccionar nuevo rol'.

Figura 22. Pantalla de modificación de rol del administrador.

Finalmente, la Figura 23 muestra la página de error que aparece cuando se trata de navegar a una ruta que no existe en el sistema.

The screenshot shows a 404 error page. The header bar has a back/forward button, a refresh icon, and the URL 'http://focused-crawler.com/notExistingPage'. The main content area features a large yellow triangle with a black outline and a white exclamation mark inside. Above the triangle, the text 'Error 404. Recurso no encontrado.' is displayed. Below the triangle, the text 'Hmmm... La página no se ha encontrado. [Volver a inicio.](#)' is shown, where 'Volver a inicio.' is a blue link.

Figura 23. Pantalla de error.

B.4 Diagrama de secuencia

Para culminar la fase de diseño, se presenta a continuación un posible caso de uso en forma de diagrama de secuencia. Este tipo de diagramas son útiles ya que se consiguen ver los componentes que son parte del proceso de comunicación y, sobre todo, el paso de mensajes entre ellos. El caso de uso que se quiere representar es el siguiente: “Un usuario inicia sesión en la aplicación y una vez dentro del portal, hace una consulta”. El diagrama correspondiente se muestra en la Figura 24.

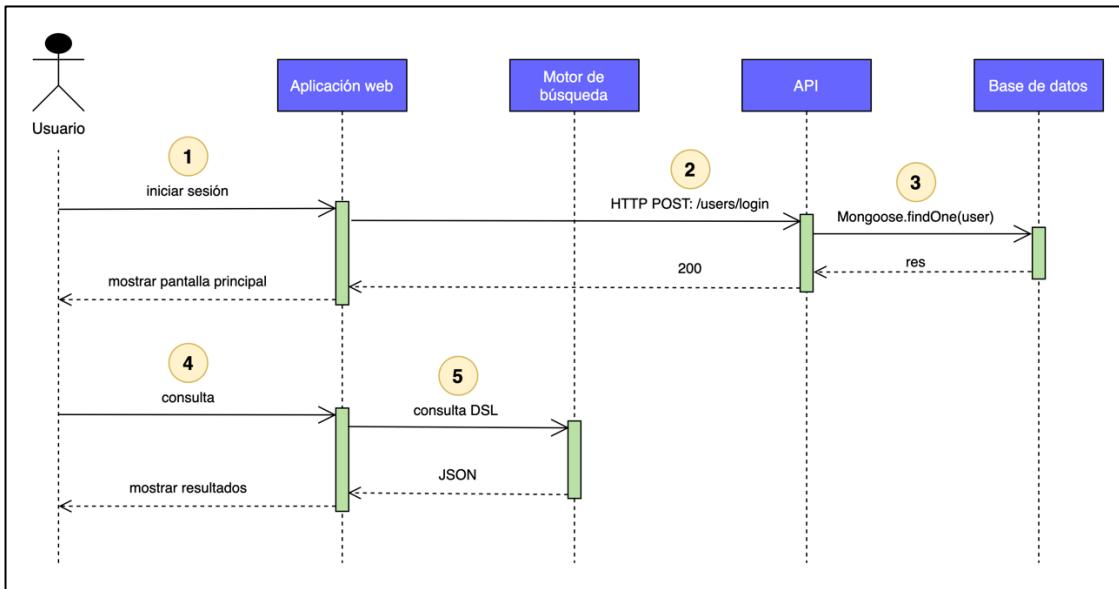


Figura 24. Diagrama de sesión correspondiente a un inicio de sesión y una búsqueda en el portal. Fuente: propia.

Se han asignado unos números a las diferentes acciones que se llevan a cabo para poder explicarlas a continuación. En primer lugar, marcado con el número uno, el usuario introduce sus credenciales en la pantalla de iniciar sesión. Una vez hecho esto, se pulsa el botón de Enviar. Esto se convierte en una petición HTTP POST a la API REST de nuestro sistema de consulta, más concretamente al servicio de “/users/login” (marcado con el número dos). Cuando la API recibe esa petición, marcado con el número tres, debe convertir esa orden en una consulta a la base de datos para ver si el usuario en cuestión está almacenado en la base de datos de usuarios. En este caso de uso, el usuario está registrado y, por lo tanto, la API devuelve un código de estado 200 (éxito) en su respuesta. La aplicación web entonces, permite el inicio de sesión al usuario y muestra la pantalla principal del portal.

Acto seguido, marcado con el número cuatro, el usuario hace una consulta desde el portal web (por ejemplo, un filtrado). Esa consulta que el usuario realiza debe transformarse en una consulta usando el lenguaje DSL, ya que es el lenguaje propio del motor de búsqueda (marcado con el número cinco). Cuando se recibe la consulta, el motor de búsqueda prepara la respuesta y la envía de vuelta a la aplicación web. Finalmente, la aplicación web convierte esa respuesta (en JSON), en una respuesta HTML que el navegador interpreta y el usuario puede visualizar.

Anexo C Implementación del sistema de extracción

Este Anexo hace referencia a todo aquello relacionado con la implementación del sistema de extracción, que se basa en la adecuada configuración del crawler Apache Nutch. Se expone en este Anexo la configuración de Nutch en detalle (tanto *plugins* como otras propiedades de interés).

C.1 Configuración de Nutch

Se muestra en esta sección la configuración de Nutch utilizada, al completo. La Tabla 3 recoge esta información, que luego se serializa en un fichero de configuración de Nutch.

Tabla 3. Configuración detallada de Nutch.

Propiedad	Valor
plugins.include	protocol-http, urlfilter-regex, parse-html, parse-tika, parse-metatags, index-basic, index-metadata, index-links, urlnormalizer-regex, urlnormalizer-basic, indexer-elastic
metatags.names	tipo, fecha_pub
index.parse.md	tipo, fecha_pub
db.ignore.external.links	true
db.ignore.internal.links	true
linkdb.ignore.internal.links	false
elastic.host	localhost
elastic.port	9300
elastic.cluster	elasticsearch
elastic.index	nutch
fetcher.server.delay	3
fetcher.threads.fetch	32
fetcher.threads.per.queue	2

Tras diferentes pruebas con diferentes configuraciones, se ha llegado a la conclusión que todas estas propiedades son las que se necesitan y son, por tanto, las que se han incluido en la configuración final. Como se menciona anteriormente, estas propiedades se vuelcan en el fichero “nutch-site.xml” de Nutch para que se hagan efectivas en el proceso de extracción. Las siguientes secciones de este apartado explican con más detalle las propiedades usadas.

C.2 Plugins de Nutch utilizados

En esta sección se explican todos los *plugins* de Nutch utilizados con cierto nivel de detalle. Un *plugin* de Nutch es un contenedor que tiene un comportamiento personalizado y que, por tanto, sirve como punto de extensión a la funcionalidad principal de extracción que presenta el *crawler* [19]. Además, éstos se pueden añadir y quitar bajo demanda, aportando una gran flexibilidad en su comportamiento. Los *plugins* que se desean usar se añaden a la propiedad “plugin.includes” del fichero “nutch-site.xml” (como se puede ver

en la Tabla 3 del Anexo C.1). En el contexto del problema, los *plugins* que se han usado son los siguientes:

protocol-http: este *plugin* es un cliente HTTP que sirve para obtener los contenidos de la web. Se hace cargo de peticiones HTTPS y de gestionar cookies de sesión. Existen otras opciones para realizar peticiones HTTP como son los *plugins* *protocol-httpclient* o *protocol-okhttp*. Tras realizar varias pruebas comparando su rendimiento, se concluyó que las tres soluciones eran muy similares en cuanto a desempeño. Se escogió el *plugin* protocol-http ya que era la que mejores resultados medios ofrecía.

urlfilter-regex: este *plugin* es muy importante. Su objetivo es filtrar las URL que se van extrayendo durante el proceso de extracción en base a la expresión regular, o expresiones, que se reflejan en el fichero “*regex-urlfilter.txt*”. Aquellas URL que se consideran irrelevantes en el proceso de extracción se pueden colocar en este fichero (en forma de expresión regular también, pero precedidas por un signo menos) para que sean ignoradas, como se aprecia en la Figura 25.

```
# skip file: ftp: and mailto: urls
-^(?:file|ftp|mailto):

# skip URLs longer than 2048 characters, see also db.max.outlink.length
#-^.{2049,}

# skip image and other suffixes we can't yet parse
# for a more extensive coverage use the urlfilter-suffix plugin
-(?i)\.(?:gif|jpg|png|ico|css|sit|eps|wmf|zip|ppt|mpg|xls|gz|rpm|tgz|mov|exe|jpeg|bmp|js)$

# skip URLs containing certain characters as probable queries, etc.
-[?*!@=]

# skip URLs with slash-delimited segment that repeats 3+ times, to break loops
-.*(/[^/]+)/[^/]+\1/[^\/]+\1/
```

Figura 25. Fichero *urlfilter-regex.txt* por defecto. Fuente: propia.

Se puede apreciar cómo, de arriba abajo, se ignora el protocolo ftp, file y mailto, se saltan aquellas URL con más de 2048 caracteres, se saltan imágenes y otros tipos de ficheros no interesantes y se ignoran aquellas URL con caracteres especiales (de consulta y barras delimitadoras “/”).

Como se puede intuir, este *plugin* es muy útil para el contexto de nuestro problema ya que un *crawler* enfocado no tiene como finalidad extraer toda la web, sino una parte concreta de la misma. Para extraer solamente la información referente al dominio del MAPA, simplemente se añade una expresión regular (precedida por el signo más) acorde para dirigir la extracción e ir eliminando las URL que se salgan de este dominio. Se observa en la línea 65 de la siguiente Figura la expresión regular deseada para enfocar el proceso de extracción al dominio del MAPA únicamente.

```
63  # accept only same site domain
64  +^(http|https)://([a-z0-9-]+\.)*mapa\.gov\.es/app/
65  +^(http|https)://([a-z0-9-]+\.)*mapa\.gov\.es/es/
```

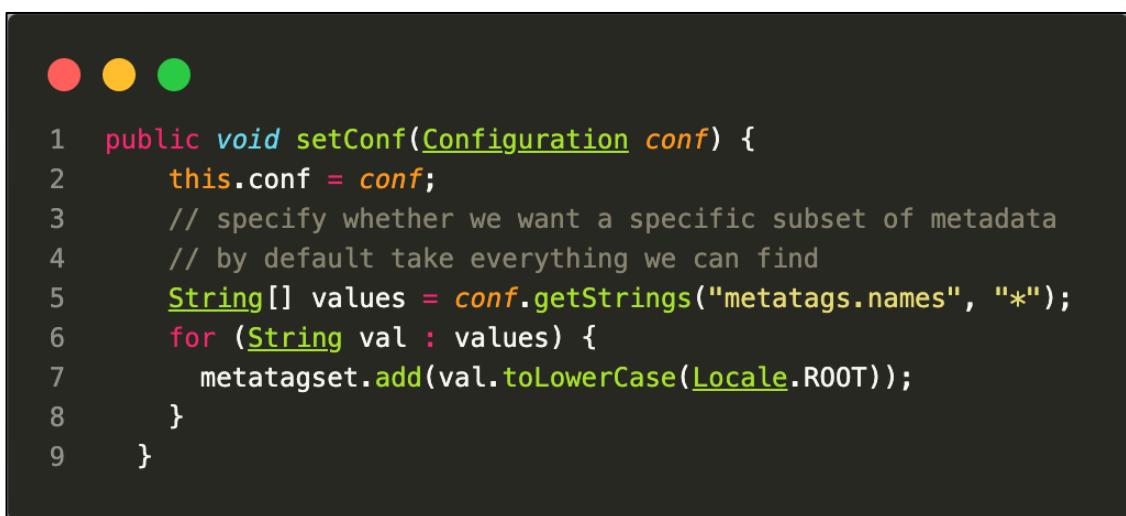
Figura 26. Expresión regular para dirigir la extracción al dominio del MAPA. Fuente: propia.

Se puede comprobar como en la línea 64 aparece otra expresión regular distinta. Esta expresión regular hace referencia a los formularios. Tras el minucioso proceso de análisis del MAPA, se pudo concluir con que el dominio de los documentos que contenían formularios seguía esa expresión regular. Como la estructura del dominio de los documentos con formularios es diferente al del resto de documentos del MAPA, se tuvo que agregar esa expresión regular para extraer ambos tipos de contenidos.

parse-(html | tika | metatags): la propia definición de este *plugin* es una expresión regular. Esto significa que se usará el *plugin* parse-html, parse-tika y parse-metatags. Todos ellos tienen en común que analizan información. Son muy necesarios y se han incluido ya que obtienen información del HTML del documento (que luego se analizará) y, sobre todo, de los metadatos escogidos (que sirven, por ejemplo, para detectar si un documento es un formulario o no).

El *plugin* **parse-metatags** recibe como entrada los metadatos que se quieren extraer de un documento a través de la propiedad *metatags.names* que se incluye en el fichero “nutch-site.xml”. Este *plugin* en específico es muy útil ya que, gracias a él, conseguimos extraer información de metadatos que no se extraen por defecto (el tipo de documento y su fecha de publicación, por ejemplo).

Una vez se especifica el nombre de los metadatos que se quieren extraer en la propiedad *metatags.names*, un método del *plugin* los lee, los extrae y los añade a la información que se indexará más adelante, como se muestra en la Figura 27.



```
1 public void setConf(Configuration conf) {
2     this.conf = conf;
3     // specify whether we want a specific subset of metadata
4     // by default take everything we can find
5     String[] values = conf.getStrings("metatags.names", "*");
6     for (String val : values) {
7         metatagset.add(val.toLowerCase(Locale.ROOT));
8     }
9 }
```

Figura 27. Método *setConf* del plugin *parse-metatags*. Fuente: propia.

Como se puede apreciar, el método lee la configuración de la propiedad *metatags.names* para obtener los metadatos que se quieren añadir y los almacena en un vector. A continuación, lo recorre y va añadiendo la información extraída en el set de meta etiquetas que más adelante formarán parte del documento final.

index-(basic | metadata | links): al igual que anteriormente, esto es una expresión regular que hace referencia a los tres *plugins* que se van a incluir. Todos ellos tienen en común que preparan el documento a indexar, es decir, con la información del análisis, construyen el documento que más tarde se indexará. El *plugin* index-basic se encarga de construir el documento a indexar con los atributos básicos que se han extraído, es decir,

el título, la URL, el contenido, el host, la estampilla temporal, etc. Tanto el index-metadata como el index-links tienen el mismo comportamiento, pero añadiendo información relativa a los metadatos y a los enlaces entrantes y salientes, respectivamente. Se necesitan incluir ya que consiguen construir el documento JSON final que se indexará con la información fruto del portal web del MAPA. No obstante, la información básica no es suficiente en el ámbito del problema. Por ello, para añadir información del tipo MIME del documento, se ha necesitado modificar el código fuente del index-basic, procedimiento que se muestra en el Anexo C.2.2.

urlnormalizer-basic: este *plugin* se encarga de normalizar las URL, esto es, estandarizar todas ellas. El proceso de normalización es el básico, pudiéndose extender si fuese necesario. Este proceso consiste en quitar el puerto por defecto de las URL y quitar segmentos con puntos en la ruta, por ejemplo, “./” o “../”. El uso de este *plugin* es un buen mecanismo para ir filtrando URL que no se desean.

indexer-elastic: este *plugin* se ejecuta en la última fase del proceso de extracción. Como su nombre indica, se encarga de indexar el documento JSON construido en ElasticSearch, el motor de búsqueda usado. Al igual que el *plugin* parse-metatags, este también necesita especificar sus entradas como propiedades en el fichero de configuración “nutch-site.xml”. Las propiedades son las siguientes: el host (la IP donde se encuentra el servidor de ElasticSearch), el puerto, el nombre del clúster y el nombre del índice. Es imprescindible este *plugin* ya que se encarga de hacer la indexación (la escritura) en el índice de ElasticSearch.

C.2.1 Otras propiedades de configuración de Nutch

Aparte de los *plugins* de Nutch, también son importantes otras propiedades que hacen que el sistema funcione como se desea. Algunas de estas propiedades están íntimamente ligadas a los *plugins* (mencionadas anteriormente), ya que se corresponden con sus entradas, pero hay otras que son ajenas a los puntos de extensión. Se ofrece una breve descripción de estas últimas a continuación.

db.ignore.external.links: si el valor de esta propiedad es *true*, los enlaces que parten del host actual a un host o dominio externo son ignorados. Muy útil ya que solamente necesitamos dominios del MAPA, de nada más. También es de utilidad para que el fichero de expresiones regulares del *plugin* urlfilter-regex no sea demasiado complejo (facilita el sistema de filtro cribando información en una primera instancia). Gracias a esta propiedad, se van cribando las URL antes de iniciar el proceso de filtrado, que es cronológicamente posterior.

db.ignore.internal.links: propiedad exactamente igual a la anterior, pero en vez de ignorar enlaces del host a un dominio externo, ignora enlaces de dominios externos al host. Muy útil también para cumplir el mismo objetivo, hacer el proceso de filtrado más sencillo.

linkdb.ignore.internal.links: esta propiedad es indispensable. Si su valor es *false*, los enlaces que se extraen y que apuntan al mismo host no se ignoran. Este es justo el comportamiento deseado, es decir, se necesita un proceso que navegue por el dominio del MAPA, extrayendo sólo páginas dentro de este marco. A modo de anécdota, esta propiedad causó muchos problemas al principio porque su valor no era explícitamente *false* y, en consecuencia, no se recuperaba la información correcta.

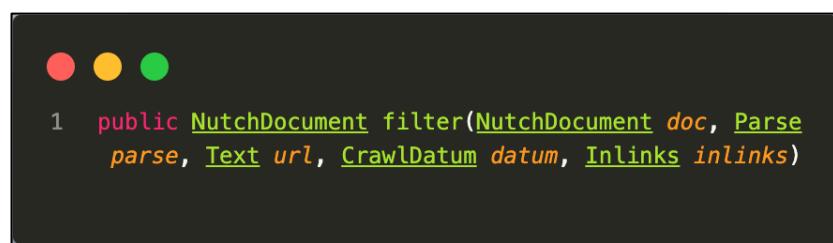
fetcher.server.delay: para no hacer el proceso de extracción extremadamente costoso en tiempo, se incluye esta propiedad y las dos siguientes. En específico, esta propiedad solamente esperará n segundos entre sucesivas peticiones al servidor, siendo n el número de segundos. En nuestro caso, se esperan tres segundos entre sucesivas peticiones, es decir, el tiempo máximo de respuesta que se espera del servidor antes de pasar a la siguiente petición son tres segundos.

fetcher.threads.fetch: esta propiedad dictamina la cantidad de procesos o hilos que se usarán en el proceso de obtención de la información. De esta forma, esta propiedad indica la cantidad de peticiones que se llevan a cabo a la vez. No obstante, el valor depende mucho del hardware con el que se ejecute el *crawler*. Un número alto de hilos requiere un hardware que soporte este gasto. Además, si el número es relativamente alto (127 o más), el proceso de extracción puede convertirse en un inconveniente para el MAPA ya que podría existir cierto riesgo de colapso del portal web. Por todo ello, hay que buscar un balance entre rapidez y respeto.

fetcher.threads.per.queue: esta propiedad indica la cantidad de hilos que acceden a la cola de extracción a la vez. Un número alto indica una fuerte concurrencia en el acceso a la información. De todas maneras, es una propiedad útil ya que acelera el proceso de extracción puesto que más procesos acceden a la cola de URL a la vez.

C.2.2 Modificación de un *Plugin* (index-basic)

En primer lugar, se debe realizar un análisis del código fuente para comprender a la perfección el comportamiento del *plugin*. Este cuenta con un método principal *filter*, cuya cabecera se muestra en la Figura 28.



```
1 public NutchDocument filter(NutchDocument doc, Parse  
    parse, Text url, CrawlDatum datum, Inlinks inlinks)
```

Figura 28. Método *filter* del plugin index-basic. Fuente: propia.

Como se puede apreciar, es un método que recibe la información extraída por anteriores *plugins* (la información analizada, la URL, los metadatos y los enlaces entrantes), la añade a un documento de Nutch y devuelve este tipo de dato (un objeto JSON con las propiedades del documento original).

Precisamente a partir de la URL origen (la ruta completa) podemos hallar el tipo MIME del documento. Para ello, se hace uso de la clase “*MimeUtil*” de Nutch. Esta clase recibe una URL como parámetro y devuelve el tipo MIME de esa URL.

Con toda esta información, ya se puede comenzar a editar el código fuente. Antes que nada, se añade al objeto de configuración de Nutch la inicialización de la utilidad que se va a usar (*MimeUtil*) como se muestra en la Figura 29 en la línea 6.

```
1 public void setConf(Configuration conf) {  
2     this.conf = conf;  
3     this.MAX_TITLE_LENGTH = conf.getInt("indexer.max.title.length", 100);  
4     this.addDomain = conf.getBoolean("indexer.add.domain", false);  
5     this.MAX_CONTENT_LENGTH = conf.getInt("indexer.max.content.length", -1);  
6     this.MIME = new MimeUtil(conf);  
7 }
```

Figura 29. Método *setConf* del plugin *index-basic*. Fuente: propia.

Una vez hecho esto se manipula la función principal *filter*, haciendo uso de la utilidad recientemente declarada, para añadir al documento final el tipo MIME, como se muestra en las líneas 11 y 12 de la Figura 30.

```
1 // add cached content/summary display policy, if available  
2 String caching = parse.getData().getMeta(Nutch.CACHING_FORBIDDEN_KEY);  
3 if (caching != null && !caching.equals(Nutch.CACHING_FORBIDDEN_NONE)) {  
4     doc.add("cache", caching);  
5 }  
6  
7 // add timestamp when fetched, for deduplication  
8 doc.add("tstamp", new Date(datum.getFetchTime()));  
9  
10 // extract MIME type from URL itself  
11 String mimeType = MIME.getMimeType(url.toString()).toString();  
12 doc.add("mimeType", mimeType);  
13  
14 return doc;  
15 }
```

Figura 30. Modificación del método *filter* de *index-basic*. Fuente: propia.

Anexo D Implementación del sistema de persistencia y procesamiento

Este Anexo hace referencia a todo aquello relacionado con la implementación del sistema de persistencia y procesamiento. Se centra en introducir la forma en la que se han implementado las consultas DSL que permiten obtener y actualizar información del sistema y también refleja diferentes aspectos de interés en la implementación de los *scripts* de post proceso, que basan su funcionamiento en este tipo de consultas.

D.1 Implementación de consultas DSL

Como ya se ha presentado a lo largo de este documento, ElasticSearch cuenta con su propio lenguaje de consultas basado en JSON. A la hora de implementar el sistema de post proceso basado en *scripts*, una tarea que se ha repetido constantemente ha sido la de traducir consultas en lenguaje natural a consultas en lenguaje DSL. Se muestran en esta sección unos ejemplos de consultas en lenguaje natural y cómo se han implementado haciendo uso del lenguaje DSL.

El primer ejemplo de consulta consiste en **obtener todos los PDF que no se hayan visitado** (para más adelante visitar sus enlaces entrantes y añadir propiedades a esos documentos). La traducción a consulta DSL se muestra en la Figura 31.

```
1
2 { "_source": "inlinks",
3   "size" : 10000,
4   "query": {
5     "bool": {
6       "must": [
7         { "match": {"mimeType.keyword": "application/pdf"}}
8       ],
9       "must_not": {
10      "exists": {
11        "field": "visitado"
12      }
13    }
14  }
15}
16}
17 }
```

Figura 31. Traducción de la consulta a lenguaje DSL. Fuente: propia.

Esta consulta consta de dos partes. La parte de obtención de los PDF y la parte de que los PDF obtenidos no se hayan visitado todavía. Para esto se utiliza el atributo “bool”, que solamente recupera el documento si los atributos que se encuentran jerárquicamente por debajo de él son verdaderos. Para la parte de obtención de los documentos PDF se usa el atributo “match”, que se asegura de que el campo *mimeType* del documento coincida con el tipo “application/pdf”, el tipo MIME de un PDF. La parte en la que nos aseguramos de que el documento no sea visitado se representa con los atributos “must_not” y “exists”, es decir, no debe existir el campo “visitado”.

Finalmente, la palabra clave “source” sirve para seleccionar el campo que se quiere obtener en la respuesta ya que, por defecto, ElasticSearch devuelve el documento JSON entero, con todos sus campos. En este caso se quiere solamente el campo *inlinks*, que representa los enlaces entrantes al documento (para, más delante, añadir información de que contienen un PDF).

Otro ejemplo interesante de consulta se describe en la siguiente situación. El primer *script* está procesando un PDF. Obtiene los documentos que apuntan a ese PDF (sus enlaces entrantes). En ese caso, **debe añadir la propiedad “tienePDF” con valor true a esos documentos si no la tenían previamente y sumar una unidad al número de PDF que poseen**. Se puede ver que esta consulta cuenta con unos condicionales. Esto no es problema para ElasticSearch ya que tiene un atributo especial, “script”, que permite insertar código en la consulta DSL, como se aprecia en la Figura 32.

```
1 ▼ {  
2   "script": "if (ctx._source.numPDF == null) {  
3     ctx._source.numPDF = 1 } else { ctx._source.numPDF += 1} if  
4     (ctx._source.tienePDF == null) { ctx._source.tienePDF =  
5       \"true\"}  
6 }
```

Figura 32. Consulta de tipo “script”. Fuente: propia.

Como se puede observar, esta consulta contiene únicamente la palabra reservada “script”, gracias a la cual se puede construir la consulta con código. Gracias a esta propiedad, ElasticSearch nos permite llevar a cabo la consulta condicional de la que se ha hablado en el anterior párrafo. Existen dos sentencias condicionales en esta consulta. La primera sentencia *if-else*, incrementa en una unidad la cantidad de PDF que tiene un documento si la propiedad existe. Esta comprobación es imprescindible ya que si se incrementa en una unidad el valor de un campo que no existe, se obtiene un error. La segunda sentencia condicional simplemente añade el campo *tienePDF* con valor *true* si no existía previamente.

Los *scripts* de post proceso han tenido que construir este estilo de consultas para poder completar y enriquecer la información de los documentos.

D.2 Implementación de los *scripts* de post proceso

Tal y como se muestra en la sección del Sistema de persistencia y procesamiento, todos los *scripts* se basan en peticiones de obtención (GET) a ElasticSearch, procesamiento de la información y peticiones de actualización (POST). Por ejemplo, la petición de obtención de la información (para luego trabajar con ella más adelante) en el primer *script* se muestra en la Figura 33.

```

1 conn = http.client.HTTPConnection("localhost", 9200)
2
3 headers = {'Content-type': 'application/json'}
4
5 # ES por defecto nos devuele solo 10 elementos. Ponemos 10.000 de size para
6 # que nos devuelva todos los que tiene.
7 query = {"_source": "inlinks", "size": 10000, "query": {
8     "bool": {"must": [{"match": {"mimeType.keyword": fileType}}], "must_not":
9         [{"exists": {"field": "visitado"}}]}}
10 json_query = json.dumps(query)
11
12 # hacemos el GET con la query
13 conn.request("GET", "/nutch/_search", json_query, headers)
14 resp = conn.getresponse()
15 respuesta_entera = resp.read().decode()

```

Figura 33. Implementación en Python de la petición GET a la API /_search de ElasticSearch para obtener los documentos no visitados del tipo indicado. Fuente: propia.

Como se puede apreciar, la línea 1 define el cliente HTTP que se va a usar para hacer la petición a ElasticSearch. Más adelante, se define la consulta a realizar. En este caso, la consulta obtiene los ficheros específicos en base a la variable “fileType” que no se hayan visitado todavía, es decir, que no se hayan procesado. A continuación, se convierte esa consulta en una consulta JSON (línea 9). Más adelante, en la línea 12, el cliente HTTP realiza la petición GET a la API /nutch/_search de ElasticSearch con el contenido de la consulta y las cabeceras. Finalmente, se recibe la respuesta.

Acto seguido, se obtienen los documentos que lo apuntan y se actualiza la información de éstos añadiéndoles la propiedad de que tienen un PDF (o el fichero en cuestión) y aumentando una unidad el número de ellos mediante una petición POST al API /_update de ElasticSearch. Una vez hecho esto, se marca el fichero como visitado, añadiendo esa propiedad al documento como se muestra en la Figura 34.

```

1 def marcarVisitado(fichero):
2     conn = http.client.HTTPConnection("localhost", 9200)
3     query = {
4         "script": "if (ctx._source.visitado == null) { ctx._source.visitado = \"true\"}"}
5     json_query_visitado = json.dumps(query)
6     endpoint = "/nutch/_update/"
7     headers = {'Content-type': 'application/json'}
8     doc_endpoint = endpoint + urllib.parse.quote_plus(fichero)
9     conn.request("POST", doc_endpoint, json_query_visitado, headers)
10    resp = conn.getresponse()
11    resp.read()

```

Figura 34. Implementación en Python de como se ha marcado como visitado un fichero. Fuente: propia.

La línea 2 abre la conexión HTTP y a continuación, se crea la consulta en la línea siguiente. Esta consulta añade el campo visitado si no estaba ya (aunque sepamos que ese

campo no existe, la comprobación es necesaria). A continuación, se definen las cabeceras y se realiza la petición HTTP POST al API /nutch/_update en la línea 9. Esta función se ha reusado en todos los *scripts* ya que, una vez manipulado un documento, no se debe manipular más.

A parte de las peticiones y consultas, otro tema de implementación interesante es el cribado de la información. Se considera que un documento no es relevante si no contiene información similar al de una base de datos. Por eso, los documentos de tipo formulario siempre permanecerán en el sistema. Sin embargo, hay veces que tanto los documentos de tipo noticia como de tipo contenido, no apuntan a ningún fichero y deberían suprimirse, pero con mucho cuidado (ya que, a priori, podría parecer que un documento no tiene ficheros, pero la razón real es que esos ficheros no se han extraído todavía y, por tanto, no se ha actualizado la información del documento que los apunta). Esto se implementa de la forma que se muestra en la Figura 35.



```

● ● ●

1  def tratarDocumento(doc):
2      tienepdf = "tienePDF" in doc["_source"]
3      tienexml = "tieneXML" in doc["_source"]
4      tienemsword = "tieneMSWORD" in doc["_source"]
5      tienexlsx = "tieneXLSX" in doc["_source"]
6      tiendocx = "tieneDOCX" in doc["_source"]
7      tieneptx = "tienePPTX" in doc["_source"]
8      tieneotros = "tieneOTROS" in doc["_source"]
9
10     contieneAlgo = tienepdf or tienexml or tienemsword or tienexlsx or
11     tiendocx or tieneptx or tieneotros
12
13     # si no contiene nada, lo eliminamos.
14     if (not contieneAlgo):
15         # en este punto lo que se haría es eliminar el documento en cuestión.
16         print("==DELETE== id: " + doc["_id"])
17         conn = http.client.HTTPConnection("localhost", 9200)
18         headers = {'Content-type': 'application/json'}
19         # endpoint de eliminar
20         endpoint = "/nutch/_doc/" + urllib.parse.quote_plus(doc["_id"])
21         print(endpoint)
22         conn.request("DELETE", endpoint, {}, headers)
23         resp = conn.getresponse()
24         resp.read()
25         print("    >>> status: " + str(resp.status))

```

Figura 35. Implementación del cribado de información de tipo Contenido. Fuente: propia.

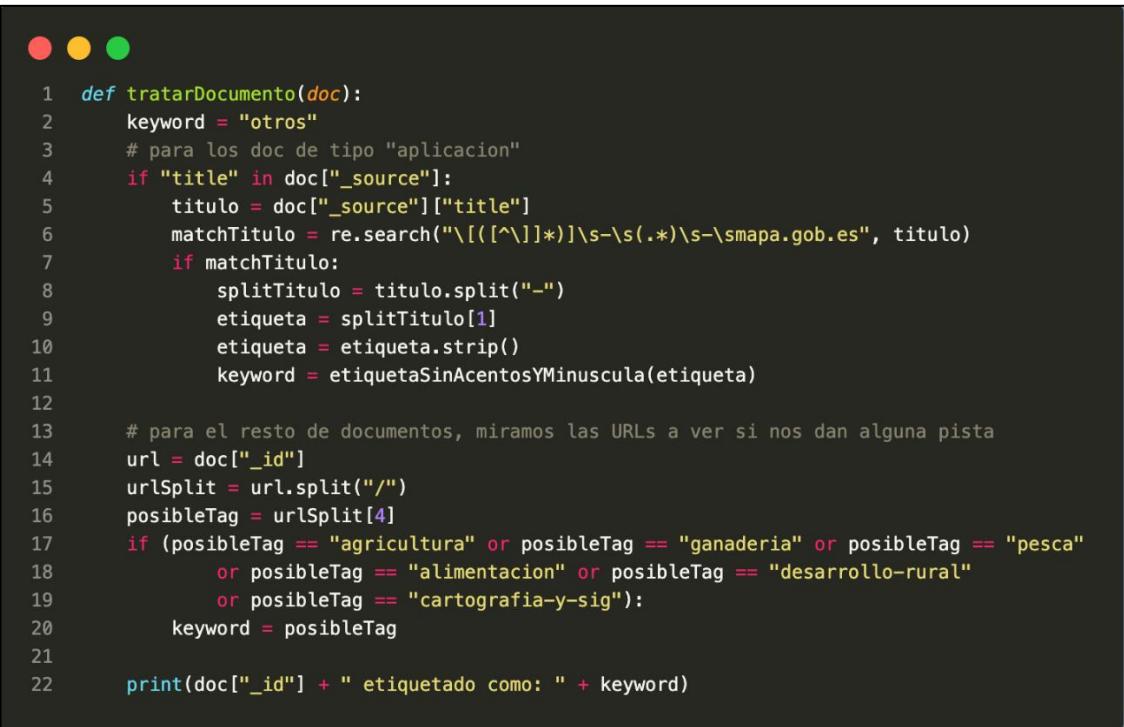
Como se ha comentado previamente, el proceso de cribar la información se hace en base a la existencia de documentos dentro de otros. Esto es, si un documento no contiene (apunta) otros documentos, no es relevante. No es una buena fuente de información por la que luego se busca. La línea 10 de la anterior Figura contiene un valor booleano referente a la presencia o no de documentos. En base a esta variable se hace la criba.

Acto seguido, si el documento no contiene nada relevante, se procede a su eliminación. De la misma forma que en las anteriores ocasiones, se define la conexión HTTP y la

cabecera. Acto seguido, se hace la petición de tipo DELETE al API /nutch/_doc en la línea 21. Este API es igual que el API de /_update ya que recibe en la propia ruta el fichero que deseamos tratar (en este caso, eliminar). Por ello, la línea 19 construye la ruta final de la petición, que será de esta forma: /nutch/_doc/<id del documento a eliminar>. Al ser una petición DELETE, no se envía ningún tipo de cuerpo de petición.

Esta sección de código criba los documentos de tipo “contenido”. Los documentos de tipo “noticia” se eliminan siempre sin hacer la comprobación, ya que no apuntan a otros documentos.

Otro aspecto interesante que comentar es la implementación de la categorización de los documentos. Se muestra el respectivo código en la Figura 36.



```

1 def tratarDocumento(doc):
2     keyword = "otros"
3     # para los doc de tipo "aplicacion"
4     if "title" in doc["_source"]:
5         titulo = doc["_source"]["title"]
6         matchTitulo = re.search("\[\[([^\]]*)\]\s-\s(.*)\s-\smapa.gob.es", titulo)
7         if matchTitulo:
8             splitTitulo = titulo.split("-")
9             etiqueta = splitTitulo[1]
10            etiqueta = etiqueta.strip()
11            keyword = etiquetaSinAcentosYMinuscula(etiqueta)
12
13    # para el resto de documentos, miramos las URLs a ver si nos dan alguna pista
14    url = doc["_id"]
15    urlSplit = url.split("/")
16    posibleTag = urlSplit[4]
17    if (posibleTag == "agricultura" or posibleTag == "ganaderia" or posibleTag == "pesca"
18        or posibleTag == "alimentacion" or posibleTag == "desarrollo-rural"
19        or posibleTag == "cartografia-y-sig"):
20        keyword = posibleTag
21
22    print(doc["_id"] + " etiquetado como: " + keyword)

```

Figura 36. Implementación de la categorización de la información. Fuente: propia.

Como se ha comentado en la sección 4.2 del apartado de implementación, el último *script* (keywords.py) se encarga de clasificar los documentos en base a su categoría. Esto se consigue analizando tanto la URL del documento como su título. Tras estudiar intensamente muchos documentos del MAPA, se encontró cierta información oculta en algunos títulos. Se usa una expresión regular (línea 6) para extraer, si es posible, una categoría de los títulos de los documentos. Algunos documentos no ofrecen información acerca de su categoría en el título, pero sí lo hacen en la propia URL. Para ello, ésta se inspecciona para tratar de encontrar palabras clave ocultas en la misma (líneas 15-20). Las palabras que se buscan son las mismas que en título, es decir, posibles categorías como agricultura, ganadería, pesca, alimentación, etc. En el caso de encontrar esta categoría, se añade al documento un campo denominado “keyword” con el valor en cuestión para que más tarde se puedan hacer filtrados en base a este campo, como se muestra en la Figura 37.

```
● ● ●  
1 conn = http.client.HTTPConnection("localhost", 9200)  
2 headers = {'Content-type': 'application/json'}  
3 endpoint = "/nutch/_update/" + urllib.parse.quote_plus(doc["_id"])  
4 # añadimos también el scoring  
5 query = {"doc": {"etiquetado": "true", "keyword": keyword}}  
6 json_query = json.dumps(query)  
7 # actualizamos campo "children" con este valor  
8 conn.request("POST", endpoint, json_query, headers)  
9 resp = conn.getresponse()  
10 resp.read()  
11 if (resp.status != 200):  
12     print(">>> " + doc["_id"] + str(resp.status))
```

Figura 37. Actualización de la información referente a la categoría. Fuente: propia.

Como se ha comentado previamente, utilizamos una petición HTTP POST al API /nutch/_update (línea 8) para actualizar la información del documento. La consulta (línea 5) simplemente añade al documento el campo con la categoría extraída y el campo que indica que el documento se ha visitado.

Anexo E Implementación del sistema de consulta

Este Anexo hace referencia a todo aquello relacionado con la implementación del sistema de consulta. Se muestra en primer lugar el modelo de datos que se usa para representar la información de los usuarios del sistema. A continuación, se muestra la API que permite gestionar las operaciones que estos usuarios pueden llevar a cabo en el sistema. Finalmente se aportan, con ejemplos, aspectos de implementación de la aplicación web que se consideran interesantes.

E.1 Modelo de datos

Esta sección tiene como objetivo mostrar el modelo de datos que se ha descrito para representar la información de los usuarios. En este contexto, el modelo de datos es sencillo porque la información que se quiere almacenar es únicamente de usuarios, debido a que los documentos del MAPA se encuentran ya almacenados en el motor de búsqueda.

Como se comenta en el apartado de Implementación en la sección 4.3, se desea almacenar el email, contraseña, fecha de registro, fecha del último inicio de sesión y el rol del usuario. Lo que se debe hacer entonces es crear el esquema o modelo de datos. En éste, se especifican todos los campos que pertenecen a un documento y la configuración de dichos campos. Cuando se haga una inserción en la base de datos, se almacenará un objeto JSON que sigue ese determinado esquema y por el que más tarde se podrá consultar.

Como también ocurre en las bases de datos relacionales, cada atributo tiene un tipo de dato (cadena de caracteres, entero, fecha, etc.). Aparte del tipo de dato, también se debe configurar si un campo es único o no para que no se hagan dos inserciones que tengan campos iguales. Esto es útil para el caso del email, es decir, cuando se registre una nueva persona, ésta no se podrá almacenar si ya existe una que tenga el mismo email. Por otro lado, para el rol es muy útil el campo “enum”, que básicamente indica los únicos roles posibles para un usuario y solamente se almacena el usuario si el rol es exactamente uno de ellos. De esta forma, se consigue asegurar siempre la consistencia de la información referente a los roles. Finalmente, cabe destacar que todos los campos son requeridos excepto el del último inicio de sesión, que se considera opcional. La Tabla 4 resume la información relativa al modelo de datos.

Tabla 4. Representación del modelo de datos de un usuario.

Campo	Tipo de dato	Unicidad	Enum	Requerido
email	Cadena de caracteres	Sí		Sí
contraseña	Cadena de caracteres	No		Sí
fecha registro	Fecha	No		Sí
fecha últ. login	Fecha	No		No
rol	Cadena de caracteres	No	Administrador, Editor, Revisor, Usuario	Sí

Para llevar a cabo la definición del modelo de datos se utiliza la librería mongoose¹⁷ de Node.js. Esta librería permite definir esquemas de datos, hacer consultas a la base de datos de forma sencilla y rápida, establecer distintas configuraciones, etc. Esta librería ha sido de mucha utilidad a lo largo de todo el desarrollo del *backend*.

E.2 API de gestión de usuarios

La API que se ha desarrollado permite gestionar los usuarios de la aplicación web. Tal y como se comenta en la sección 3.4, la API ofrece servicios que permiten:

- Iniciar sesión.
- Registrarse.
- Dar «me gusta» a un documento.
- Editar un documento.
- Obtener los usuarios y su rol.
- Obtener el número total de usuarios registrados.
- Cambiar el rol de un usuario.

Todos estos servicios se han documentado con la herramienta Swagger¹⁸. La Figura 38 muestra el resultado.

The screenshot shows the Swagger UI interface for a NodeJS API. At the top, it says "NodeJS API doc 1.0.0 OAS3". Below that, there's a "Servers" dropdown set to "http://localhost:3003/" and an "Authorize" button. The main area is divided into sections: "User" and "Admin".

User

- POST** /users/login Logs new user
- POST** /users/signin Signs in new user
- PUT** /like Likes a document
- PUT** /editar Updates a document
- DELETE** /tagsChildren Deletes field "tagsChildren" of a document

Admin

- GET** /users ADMIN ONLY. Obtains every user-role pair
- GET** /users/total ADMIN ONLY. Obtains total number of registered users
- PUT** /users/{id}/rol Change user's role

Figura 38. Documentación de la API con Swagger. Fuente: propia.

¹⁷ <https://codigofacilito.com/articulos/que-es-mongoose>

¹⁸ <https://swagger.io/>

Como se puede comprobar, existen servicios accesibles por los usuarios (la mayoría) y servicios accesibles solamente por el administrador (todos aquellos relacionados con el panel de administración). Como ejemplo, las Figuras 39 y 40 muestran la documentación del servicio de inicio de sesión.

The screenshot shows a user interface for a REST API documentation tool. At the top, it says "User". Below that, a green button labeled "POST" is followed by the endpoint "/users/login Logs new user". To the right of the endpoint is a small upward arrow icon. Underneath, there's a section titled "Parameters" with a green progress bar underneath it. A "Try it out" button is located to the right. Below this, it says "No parameters". Then there's a section for "Request body" which is marked as "required". To the right of this is a dropdown menu set to "application/json". Below the request body section, there are two links: "Example Value" and "Schema". A large black box contains a JSON example:

```
{  
  "email": "example@gmail.com",  
  "password": "iamexample1234"  
}
```

Figura 39. Documentación del servicio de inicio de sesión (1). Fuente: propia.

Por un tema de legibilidad se ha tenido que dividir la documentación del servicio de inicio de sesión en dos Figuras. La Figura 39 muestra el método y la URI que hay que utilizar para acceder al servicio. Muestra también que no se necesitan parámetros y que la petición requiere un cuerpo que conste del email y la contraseña del usuario que desea iniciar sesión.

La documentación del servicio incluye también las respuestas que devuelve el servidor en función a la petición enviada, como se puede apreciar en la Figura 40 mostrada a continuación. Si existe algún error en el inicio de sesión, el servidor devuelve el código de error 400 acompañado de un mensaje explicativo (contraseña o usuario incorrectos) mientras que, si el problema es uno interno del servidor, entonces se devuelve un código de error 500 con el mensaje del error correspondiente. En el caso de que todo sea correcto, el servidor responde con el código de estado 200 junto al id del usuario y al token.

Code	Description	Links
200	User successfully logged in Media type application/json ▾ Controls Accept header. Example Value Schema <pre>{ "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImV4YW1wbGVAZ21hdWwuY29tIiwiZWFOljoxNjUwNTYzMjQ5LCJleHAiOjE2NTA2NDk2NDI9.f1NcFQ-TIPRbHbdPDJNRnpuqBS2uEpvrJjrElbMW-RQ", "id": "62795e42312401ed8495e677" }</pre>	<i>No links</i>
400	User cannot be logged in (user not registered or incorret pw). Media type application/json ▾ Example Value Schema <pre>{ "message": "User or password is incorrect" }</pre>	<i>No links</i>
500	Internal server error. Media type application/json ▾ Example Value Schema <pre>{ "message": "string" }</pre>	<i>No links</i>

Figura 40. Documentación del servicio de inicio de sesión (2). Fuente: propia.

El id es un extra ya que no se utiliza en este proyecto, pero es muy útil para que, en un futuro, se puedan realizar operaciones de modificación de los datos del usuario puesto que el id es el código que identifica a un usuario en la base de datos. Por otro lado, como el protocolo HTTP no tiene estado, se necesita mantener el estado de la sesión de alguna forma. Esto se consigue mediante el uso de un token.

En este proyecto se ha usado JWT¹⁹, un estándar abierto basado en JSON para la creación de tokens de acceso [20]. Operaciones como el inicio de sesión y el registro no necesitan token ya que en ese punto todavía no se tiene uno. En cambio, sí que es necesario en todas

¹⁹ <https://jwt.io/>

las demás peticiones para poder autenticar al usuario emisor de la petición ya que solamente los usuarios autenticados poseen un token válido. En la Figura 38, las peticiones que necesitan un JWT se pueden identificar con un candado en color gris.

Estos servicios que necesitan autenticación de usuario (token) se implementan haciendo uso de un middleware de autenticación. Este middleware no es más que una función que se ejecuta antes que el controlador. Ésta comprueba si el token que se envía es correcto (corresponde a un usuario autenticado en la aplicación web) o, por el contrario, se trata de un token inválido. Si el token es inválido, el servidor directamente devuelve un código de error, sin ni siquiera ejecutar el controlador correspondiente (no es necesario ya que el usuario no está autenticado porque su token es inválido). Si el token es correcto, entonces se pasa a ejecutar el controlador del servicio.

El controlador del servicio de inicio de sesión es el que se muestra a continuación en la Figura 41.



```
1 // login
2 const login = async (req, res) => {
3   const email = req.body.email;
4   const password = req.body.password;
5   // si no me pasan valores o no son strings...
6   if (
7     email == undefined ||
8     password == undefined ||
9     typeof email != "string" ||
10    typeof password != "string"
11   ) {
12     return res.status(400).send({ message: "Invalid values" });
13   }
14
15 User.findOne({ email }, (err, retrievedUser) => {
16   if (err) {
17     logger.error("Error en BD -> /login");
18     return res.status(500).send({ message: err });
19   }
20   if (!retrievedUser)
21     return res.status(400).send({ message: "Usuario no registrado" });
22   else {
23     // user with 'email' exists. Let's prove pw is correct.
24     if (!bcrypt.compareSync(password, retrievedUser.password)) {
25       return res.status(400).json({
26         message: "Contraseña incorrecta",
27       });
28     }
29     // everything OK. jwt is generated.
30     const accessToken = jwt.sign(
31       { username: email, id: retrievedUser._id },
32       process.env.TOKEN_KEY,
33       {
34         expiresIn: 60 * 60 * 24, // 24 h
35       }
36     );
37     return res
38       .status(200)
39       .send({ token: accessToken, id: retrievedUser._id });
40   }
41 });
42 };
```

Figura 41. Controlador de inicio de sesión. Fuente: propia.

El controlador es el método que se encarga de implementar la funcionalidad específica del servicio. En este caso, la Figura 41 muestra el controlador del inicio de sesión. Después de comprobar que los parámetros recibidos en el cuerpo de la petición son correctos (ll. 7-10), se realiza una petición a la base de datos para encontrar al usuario con el email recibido usando la librería mongoose, como se comenta en el Anexo E.1. Si éste existe, se comprueba que la contraseña es la adecuada (l. 24) y en ese caso, se construye el token correspondiente y se envía como respuesta junto al id (l. 37). En caso contrario, se envía el código de error correspondiente. No obstante, si se recibe un código de error interno procedente de la base de datos, se vuelca el error en un fichero de log para que se tengan evidencias en todo momento de los errores que atraviesa la API (l. 17). Para tener este fichero de registro interno, se ha usado la librería Winston²⁰ de Node.js.

E.3 Aplicación web

Como se menciona al final de la sección 4.3, se ha seguido la guía oficial de React. Una buena práctica que se ha extraído de esa Guía de Estilo es la división de la funcionalidad en componentes. Un componente de React es una pieza individual y autónoma de código que lleva a cabo una funcionalidad [21]. Gracias a estos componentes, se consigue desarrollar un sistema modular y escalable ya que una gran característica de éstos es su reusabilidad. En el contexto del proyecto, se ha dividido la funcionalidad total a desarrollar en pequeñas partes, creando un componente para cada una de ellas. De esta forma, la estructura de ficheros que sigue la aplicación web es la mostrada en la Figura 42.

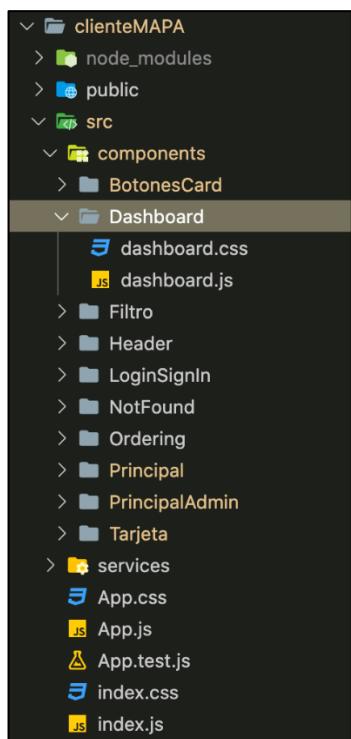


Figura 42. Estructura de ficheros de la aplicación web. Fuente: propia.

²⁰ <https://www.npmjs.com/package/winston>

Como se puede ver, el código es completamente modular. Fuera de los ficheros de índice (que supone el punto de entrada a la aplicación web) y el de aplicación (que actúa de enrutador), se puede apreciar que se tiene un directorio para cada funcionalidad individual, es decir, para la cabecera, para la página principal, para el Dashboard del administrador, para la tarjeta que representa el documento, etc. Así, dentro de cada directorio se puede encontrar el fichero de estilos CSS y el fichero JavaScript que implementa el componente.

Se comentan a continuación puntos de interés relativos a la implementación de un componente. De forma más detallada, un componente recibe un objeto y devuelve un elemento de React que describe lo que debe mostrarse en pantalla [22], como se puede observar en la Figura 43.



```

1  function BotonesCard(props) {
2      //console.log(props.obj); props.obj tiene todos los campos de la card
3      var handleTitleUpdate = props.handleTitleUpdate;
4      var handleScoringUpdate = props.handleScoringUpdate;
5      var handleTypeUpdate = props.handleTypeUpdate;
6      // variable necesaria para que (SIN darle a la ruleta de actualizar de Chrome),
7      // al editar un titulo y acto seguido, volverlo a editar, coja el título nuevo
8      // y no el inicial, como hacía antes. Le paso el estado y fin, me despreocupó.
9      var tituloTarjeta = props.tituloTarjeta;
10     var tipoTarjeta = props.tipoTarjeta;
11     return (
12         <Stack direction="row">
13             <Stack direction="row" spacing={1}>
14                 <ShowTags item={props.obj} />
15             </Stack>
16             <div className="espacioizquierda">
17                 <Stack direction="row" spacing={3}>
18                     <Button
19                         style={{
20                             borderRadius: 25,
21                             backgroundColor: "#f2ad4e",
22                             textTransform: "none",
23                         }}
24                         variant="contained"
25                         startIcon=<EditIcon />
26                         size="small"
27                         onClick={formularioEditar.bind(
28                             this,
29                             props.obj,
30                             tituloTarjeta,
31                             handleTitleUpdate,
32                             tipoTarjeta,
33                             handleTypeUpdate
34                         )}

```

Figura 43. Sección del código del componente BotonesCard. Fuente: propia.

El componente BotonesCard, que se encarga de la funcionalidad de los botones de editar y de «me gusta», recibe un objeto llamado props y devuelve un elemento de React que describe cómo se muestra la información por pantalla. Además, dentro de este componente se llaman sin problemas a otros componentes de React (línea 12, 14, 18,

etc.). De esta forma, se consigue crear un componente relativamente complejo formado por pequeños componentes individuales (botones, contenedores...), modularizando el código.

Aparte del concepto de componente, otra buena práctica de desarrollo de código en React que se ha adoptado es el uso de hooks²¹. Básicamente, los hooks son un mecanismo de gestión del estado en React mediante variables de estado y funciones que modifican su valor. Se han utilizado alrededor de la funcionalidad de editar y dar «me gusta». Cuando el usuario edita o aporta un «me gusta» a un documento, los hooks se encargan de verificar si algo referente al documento ha cambiado para volver renderizar la página con la nueva información.



```
1  function Tarjeta(item) {
2    let titulo_doc = item.item.title;
3    if (titulo_doc === undefined) titulo_doc = "[Title N/A]";
4    const [tit, setTit] = useState(titulo_doc);
5    // HOOK para que cuando se edite el documento, se actualice el título tb.
6    const handleTitleUpdate = (valor) => {
7      setTit(valor);
8    };
9
10   const [rel, setRel] = useState(item.item.userScoring);
11   // HOOK para la relevancia al darle like.
12   const handleScoringUpdate = () => {
13     setRel(rel + 1);
14   };
15
16   const [tipo, setTipo] = useState(item.item.tipo);
17   // HOOK para la relevancia al darle like.
18   const handleTypeUpdate = (tipo) => {
19     setTipo(tipo);
20   };

```

Figura 44. Sección del código del componente Tarjeta. Fuente: propia.

En la Figura 44 se pueden apreciar tres hooks usados: hook para el título (líneas 4-8), hook para la relevancia (líneas 10-14) y hook para el tipo (líneas 16-20). Cada uno de ellos controla una variable y actualiza inmediatamente la interfaz gráfica de forma dinámica si una de estas se ve modificada. En definitiva, son una especie de funciones que se ejecutan en segundo plano y que se encuentran a la espera de un cambio. Cuando ese cambio ocurre, ejecutan el código correspondiente.

Como se ha comentado en la sección 4.3, otro aspecto de interés que ha tenido mucho peso durante el proceso de implementación ha sido el uso de la librería Reactivesearch²². Esta librería ofrece componentes gráficos de React que se conectan con el índice de ElasticSearch. Gracias a estos componentes se pueden construir los filtrados y buscadores

²¹ <https://es.reactjs.org/docs/hooks-intro.html>

²² <https://opensource.appbase.io/reactivesearch/>

requeridos que consultan directamente el índice de ElasticSearch con los documentos extraídos (y procesados) del MAPA. Además, esta librería ofrece una gran cantidad de componentes, que se agrupan en diferentes categorías en base a su finalidad. En esta aplicación se han utilizado componentes que pertenecen a estas cuatro categorías:

- **Result components.** Para mostrar el listado de documentos recuperados al hacer una búsqueda/filtro.
- **List Components.** Para permitir que se realicen las búsquedas y filtrados. Por ejemplo, el componente que permite recuperar documentos atendiendo a los ficheros que contiene es una MultiList.
- **Search Components.** Para las búsquedas en texto (el buscador que se ofrece en la parte superior).
- **Range components.** Para implementar el filtrado por fecha.

Estos componentes tienen una serie de propiedades que permiten llevar a cabo un conjunto extenso de cambios en el comportamiento y en la vista del propio componente. Las propiedades son opcionales excepto dos de ellas: el id del componente y el campo de datos. El primero sirve para hacer referencia al componente más adelante desde otros componentes, mientras que la utilidad del segundo de ellos es indicar el campo de ElasticSearch al que se atacará cuando se ejecute la consulta asociada a ese componente.

Se muestra un ejemplo de componente de tipo *Range* de esta librería en la Figura 45.



The screenshot shows a code editor window with a dark theme. At the top, there are three colored circular icons: red, yellow, and green. The main area contains the following code:

```
1 <DateRange
2   componentId="Fechas"
3   //placeholder="Seleccione una fecha"
4   dataField="fecha_pub"
5   queryFormat="date"
6   placeholder={}
7   start: "Fecha de inicio",
8   end: "Fecha de fin",
9   }
10  title=<FilterFecha />
11  focused={false}
12  showClear={true}
13  //customQuery={DateQuery}
14  />
```

Figura 45. Componente DateRange de la librería Reactivesearch usado en el componente Principal. Fuente: propia.

Podemos apreciar que existe el identificador del componente y el campo de datos al que ataca, que en particular es el campo “fecha_pub” del índice. El resto de las propiedades son opcionales, pero necesarias para cumplir el requisito de filtrado por fechas. Cabe destacar el campo “queryFormat” para definir el tipo de dato referente a la fecha por el que se consultará, ya que hay varios tipos y se debe seleccionar el mismo tipo de dato que se almacena en el índice. Para la propiedad del título, se ha usado un componente de elaboración propia que muestra el título con estilos personalizados. Finalmente, se puede

modificar la consulta que se hace a Elasticsearch gracias a la propiedad “customQuery”, pero como el comportamiento por defecto es el deseado, no ha hecho falta. En conclusión, el componente renderiza en pantalla dos inputs en forma de calendario para que el usuario elija las fechas de publicación de inicio y fin, como se puede apreciar en la Figura 46.

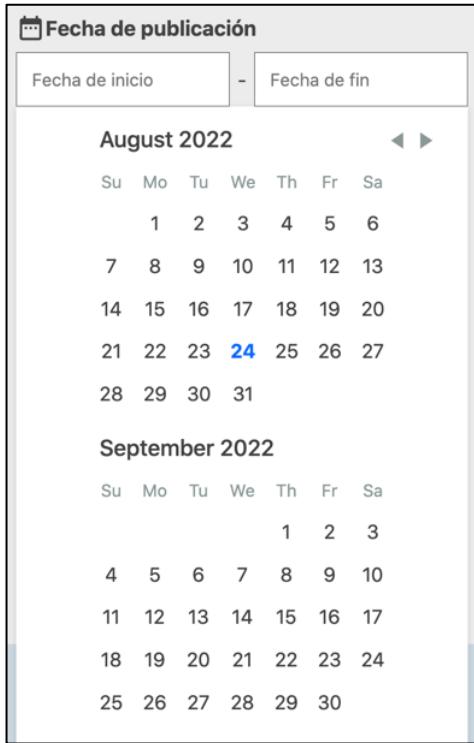


Figura 46. Renderizado del componente de filtrado por fecha. Fuente: propia.

Acto seguido, cuando el usuario elige la fecha de inicio y fin, se forma una consulta que ataca el índice de Elasticsearch (concretamente al campo “fecha_pub”) con las entradas proporcionadas por el usuario. Finalmente, se reciben los documentos que satisfacen ese filtro, se procesan y se muestran en pantalla gracias a un componente de la categoría ResultList.

Otras dos librerías muy destacables que se han utilizado han sido yup²³ y formik²⁴. Su uso permite realizar las validaciones de los valores de entrada que proporciona el usuario en el inicio de sesión y registro. Se usan en combinación para crear una serie de reglas que deben cumplir los datos de entrada para procesarse. Estas reglas se aplican a cada campo del formulario que se quiere validar y se pueden incluir expresiones regulares que debe cumplir la entrada del usuario para que esta sea válida. Las reglas usadas para la validación del registro se muestran en la Figura 47. En el contexto del problema, interesa que nadie se pueda registrar como administrador por lo que se usa la expresión regular de la línea 5 de la Figura 47. Interesa también que el campo de la contraseña tenga como mínimo 8 caracteres (línea 9 de la Figura 47) y, además, se explica que el campo de confirmación de contraseña debe ser idéntico al campo contraseña (línea 14 de la Figura

²³ <https://www.npmjs.com/package/yup>

²⁴ <https://formik.org/>

47). Además, los tres campos son obligatorios, como indica la última regla de cada campo de esta Figura.

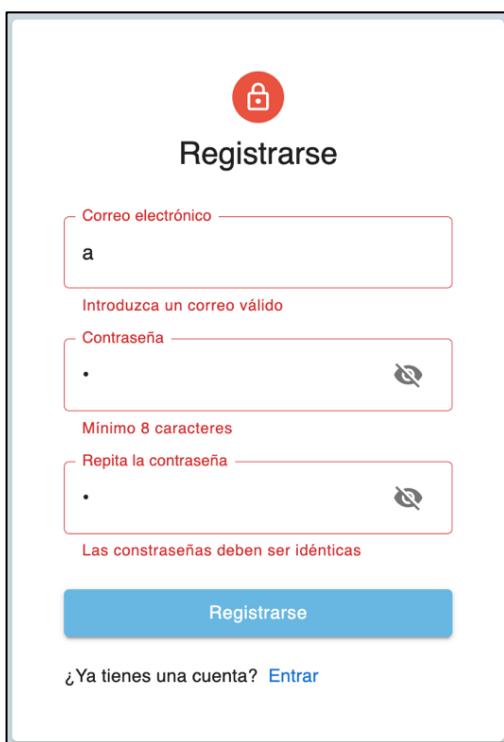


```
1 const validationSchema = yup.object({
2     CampoCorreo: yup
3         .string("Introduzca su correo electrónico")
4         .email("Introduzca un correo válido")
5         .matches(/^(?!admin\b)/i, "Correo inválido")
6         .required("Campo requerido"),
7     CampoContraseña: yup
8         .string("Introduzca su contraseña")
9         .min(8, "Mínimo 8 caracteres")
10        .required("Campo requerido"),
11     ConfirmaContraseña: yup
12         .string("Repita la contraseña")
13         .required("Campo requerido")
14         .oneOf(
15             [yup.ref("CampoContraseña"), null],
16             "Las contraseñas deben ser idénticas"
17         ),
18 });

```

Figura 47. Implementación de las reglas de validación de campos del registro. Fuente: propia.

Ahora, se muestra el resultado correspondiente a estas reglas en la Figura 48.



The registration form is titled 'Registrarse' and features three input fields with validation messages:

- Correo electrónico:** The input field contains 'a' and has a red border. Below it, the error message 'Introduzca un correo válido' is displayed.
- Contraseña:** The input field contains a password and has a red border. Below it, the error message 'Mínimo 8 caracteres' is displayed.
- Repita la contraseña:** The input field contains a password and has a red border. Below it, the error message 'Las contraseñas deben ser idénticas' is displayed.

At the bottom of the form is a blue 'Registrarse' button.

Figura 48. Registro de usuario con validación de campos. Fuente: propia.

Como se puede comprobar, antes de si quiera realizar cualquier tipo de petición a la API de gestión de usuarios, se hace una comprobación de las entradas en el propio cliente de la aplicación web. En este caso se puede comprobar cómo se pide un correo válido, una contraseña de mínimo 8 caracteres, y que además debe coincidir con la siguiente.

Como punto final, y con el objetivo de cumplir con uno de los requisitos no funcionales, se ha usado la librería MUI²⁵ para ofrecer esa homogeneidad de color y estilo a lo largo de todo el desarrollo de la aplicación web. Esta librería ofrece un conjunto de componentes e iconos básicos y también avanzados, altamente personalizables.

Todo el código de la aplicación web, del *backend* y de los *scripts* está disponible en el repositorio de GitHub²⁶.

²⁵ <https://mui.com/>

²⁶ <https://github.com/pabloJordan24/TFG-2022>

Anexo F Pruebas

Este Anexo muestra evidencias de los resultados obtenidos y detalla ejemplos de diferentes pruebas realizadas.

F.1 Resultados obtenidos

Esta sección del Anexo tiene como objetivo mostrar los resultados de toda la aplicación web del sistema final.

Una vez se ejecuta la aplicación web, el usuario puede iniciar sesión como usuario registrado (introduciendo su correo y contraseña) o como usuario invitado (no es necesario autenticarse) tal y como se muestra en la Figura 49.

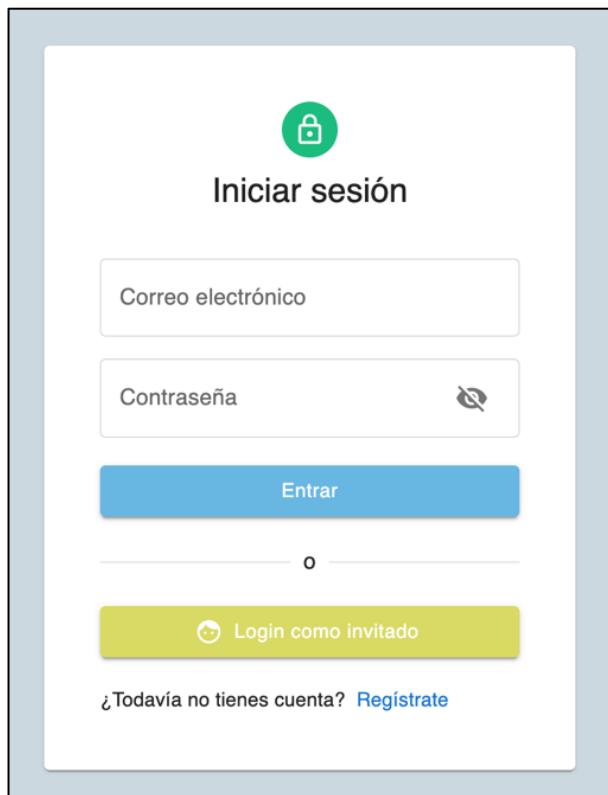


Figura 49. Vista del Inicio de sesión. Fuente: propia.

No obstante, el usuario también se puede registrar si aún no tiene cuenta o no desea entrar como usuario invitado en la aplicación, como se muestra en la Figura 50.

The image shows a registration form titled "Registrarse". It includes fields for "Correo electrónico", "Contraseña", and "Repita la contraseña", each with a visibility toggle icon. A large blue "Registrarse" button is at the bottom, and a link "¿Ya tienes una cuenta? Entrar" is at the bottom left.

Figura 50. Vista del registro. Fuente: propia.

En este caso, para mostrar los resultados, se ha hecho un inicio de sesión como invitado. Una vez hecho esto, se muestra la página principal del portal web (esta vez al completo). Este resultado se puede apreciar en las Figuras 51 y 52.

The screenshot shows the main page of a web portal. At the top, it says "Focused Crawler Endpoint" and "Invitado". On the left, there's a sidebar with "Formatos" (PDF, DOC, XML, etc.) and "Páginas web que contienen..." (PDF, DOCX, XLSX, etc.). The main area has a search bar and a list of 5 de 25 resultados encontrados en 9 ms. The first result is "Ayudas Nacionales" (src: https://www.mapa.gob.es/es/alimentacion/temas/integracion-asociativa/apoyos-integracion/Ayudas_Nacionales.aspx, Relevancia: 5 stars, Fecha de publicación: 2022-07-08). The second result is "BIENESTAR ANIMAL" (src: <https://www.mapa.gob.es/es/ganaderia/temas/produccion-y-mercados-ganaderos/bienestanimal/default.aspx>, Relevancia: 5 stars, Fecha de publicación: 2022-06-03). Both results have "PDF(15)", "MSWORD(2)", "DOCX(5)", "XLSX(5)" buttons and "Editar" and "Like" buttons.

Figura 51. Vista de la página principal del portal web (1). Fuente: propia.

The screenshot displays the main interface of a web portal. On the left, there is a sidebar with several filter options:

- Formato de documento:** Includes checkboxes for PDF (checked), DOCX, XLSX, MSWORD (checked), XML, PPTX, and OTROS. The count for PDF is 1770, DOCX 125, XLSX 115, MSWORD 55, XML 19, PPTX 3, and OTROS 1.
- Tipo de página:** Buttons for Contenido, Noticia, and Formulario.
- Fecha de publicación:** A date range selector from Fecha de inicio to Fecha de fin.
- Categoría:** Radio buttons for Agricultura, Ganadería, Pesca, Alimentación, Desarrollo rural, Cartografía y SIG, and Otros.
- Relevancia:** A slider scale from 0 Estrellas to 5 Estrellas.

The main content area shows search results for different topics, each with a title, source URL, relevance rating (0 stars), publication date, download links (PDF and MSWORD), and edit/like buttons.

- BIENESTAR ANIMAL**
src: <https://www.mapa.gob.es/es/ganaderia/temas/produccion-y-mercados-ganaderos/bienestanimal/default.aspx>
Relevancia: 0 (0)
Fecha de publicación: 2022-06-03
PDF(2) MSWORD(1)
- BIENESTAR ANIMAL**
src: <https://www.mapa.gob.es/es/ganaderia/temas/produccion-y-mercados-ganaderos/bienestanimal/>
Relevancia: 0 (0)
Fecha de publicación: 2022-06-03
PDF(2) MSWORD(1)
- Higiene y trazabilidad**
src: https://www.mapa.gob.es/es/agricultura/temas/sanidad-vegetal/higiene_y_trazabilidad/
Relevancia: 0 (0)
Fecha de publicación: 2020-12-09
PDF(1) MSWORD(1)
- Subvenciones a proyectos de innovación tecnológica en el medio rural**
src: <https://www.mapa.gob.es/es/desarrollo-rural/ayudas-y-subvenciones/subvenciones-innovacion-tecnologica.aspx>
Relevancia: 0 (0)
Fecha de publicación: 2017-12-22
PDF(3) MSWORD(4)

Pagination at the bottom: Prev, 1, 2, 3, 4, 5, Next.

Figura 52. Vista de la página principal del portal web (2). Fuente: propia.

El portal web cuenta con una cabecera en la que se muestra, a mano derecha, el nombre del usuario con la opción de cerrar sesión si se hace clic en el nombre. A continuación, se puede encontrar el propio portal web. A mano izquierda se encuentran los filtros, uno por cada requisito funcional de este tipo que se ha definido. De arriba abajo, se puede ver el filtro por formato de documento, por tipo de fichero que contiene ese documento, por tipo de página, por rango de fechas de publicación, por categoría y, finalmente, por relevancia.

En este pequeño ejemplo, se ha aplicado un filtro por el tipo de fichero que contienen las páginas web. En este caso, se quieren solamente aquellos documentos que contengan PDF y MSWORD. Se observa, a mano derecha, como la lista devuelta por el portal muestra solamente aquellos documentos que contienen PDF y MSWORD (junto a su cantidad, entre paréntesis). No solamente se devuelve esa información, sino también el título, la fuente, la relevancia y fecha de publicación del documento en cuestión. Además, la información devuelta está paginada para que sea más sencilla su lectura.

También, justo encima de la lista devuelta, se encuentran los filtros que se están usando, pudiéndose quitar de forma individual en cualquier momento desde ahí o de forma colectiva haciendo clic en *Clear All*. Justo encima se encuentra el buscador por texto. Es un buscador que cuenta con una funcionalidad de auto sugerencia, es decir, conforme se escribe información, el portal web va sugiriendo documentos que se asemejen al texto introducido. Se muestra un ejemplo en la Figura 53.

The screenshot shows a search interface titled "Focused Crawler Endpoint". On the left, there's a sidebar with sections for "Formatos" (file types) and "Páginas web que contienen..." (web pages containing...). The "Formatos" section lists file extensions with their counts: .pdf (6972), .doc (137), .xml (12), .html (5998), .docx (183), .xlsx (659), and .pptx (6). The "Páginas web que contienen..." section has a search bar with "alimentos" and a list of results: "Premios Alimentos de España", "Última edición Premios Alimentos de España", "Premio Alimentos de España Mejores Quesos", "Premio Alimentos de España, Edición 2020", "Promoción de los Alimentos de España", "Observatorio de Precios de los Alimentos", "Premio "Alimentos de España Mejores Quesos\"", "Premio Alimentos de España - Edición 2021", and "Estudios Cadena de Valor Alimentos Frescos". Below the results is a PDF download link: "tendencias/informesmesesalimentacionseptiembre2017_tcm30-434503.pdf". There are also "Relevancia" and "Fecha de publicación: N/A" filters, and "Editar" and "Like" buttons.

Figura 53. Vista de una búsqueda con auto sugerencia. Fuente: propia.

Se aprecia como al introducir la palabra “alimentos”, el sistema sugiere ciertas páginas que con ese texto se asemejan.

Otro aspecto del portal es la edición. Si se hace clic en el botón de Editar, se muestra un formulario de edición del documento, como se puede observar en la Figura 54.

The screenshot shows the same search interface as Figure 53, but with a modal dialog box in the foreground titled "¡Ayúdanos a mejorar!". The dialog contains fields for "Título del documento" (Ayudas Nacionales), "Tipo de documento" (contenido), and "Ficheros que contiene" (PDF-15,MSWORD-2,XLSX-5,DOCX-5). At the bottom of the dialog are "Cancelar" and "Continuar" buttons. In the background, the search results for "Búsqueda genérica" are visible, including a result for "BIENESTAR ANIMAL" with a relevance of 0 and a publication date of 2022-06-03. There are "Editar" and "Like" buttons for each result.

Figura 54. Vista de la edición de un documento. Fuente: propia.

De esta forma, si el usuario posee el rol de edición, después de introducir sus cambios y continuar con la edición, éstos persisten en el portal web. Sin embargo, si no se tienen los permisos de edición (como es el caso ya que se inició sesión como usuario invitado), el sistema muestra el error de la Figura 55.

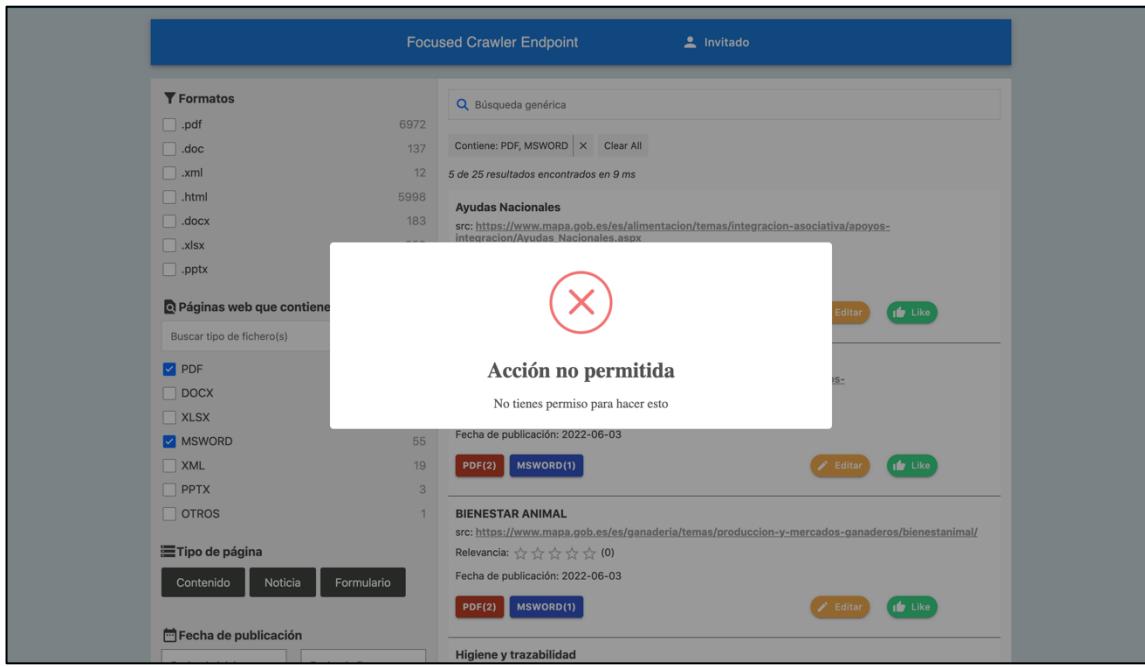


Figura 55. Vista de una acción no permitida. Fuente: propia.

La funcionalidad de «me gusta» también depende del rol del usuario. Si no se tiene permiso, el sistema avisa al usuario con el error de la anterior Figura. Si, por el contrario, se puede llevar a cabo esta acción, entonces el sistema aumenta la relevancia de ese documento y avisa al usuario del éxito como se muestra en la Figura 56. Como se puede observar, para poder realizar esto, se ha cerrado sesión y se ha iniciado de nuevo como usuario administrador.

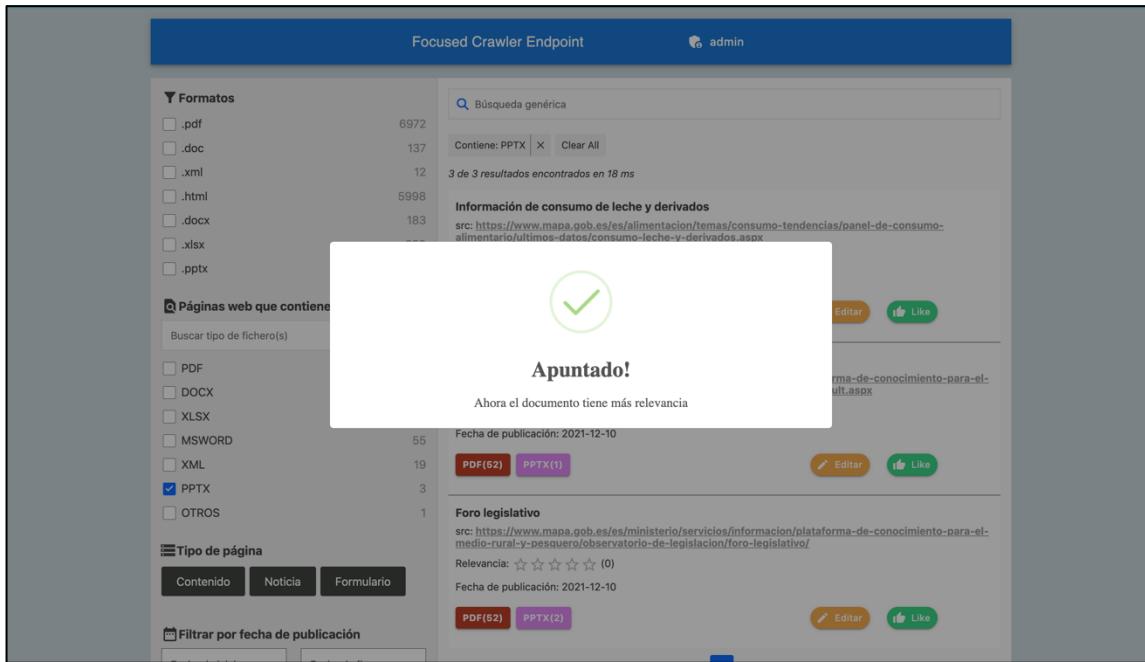
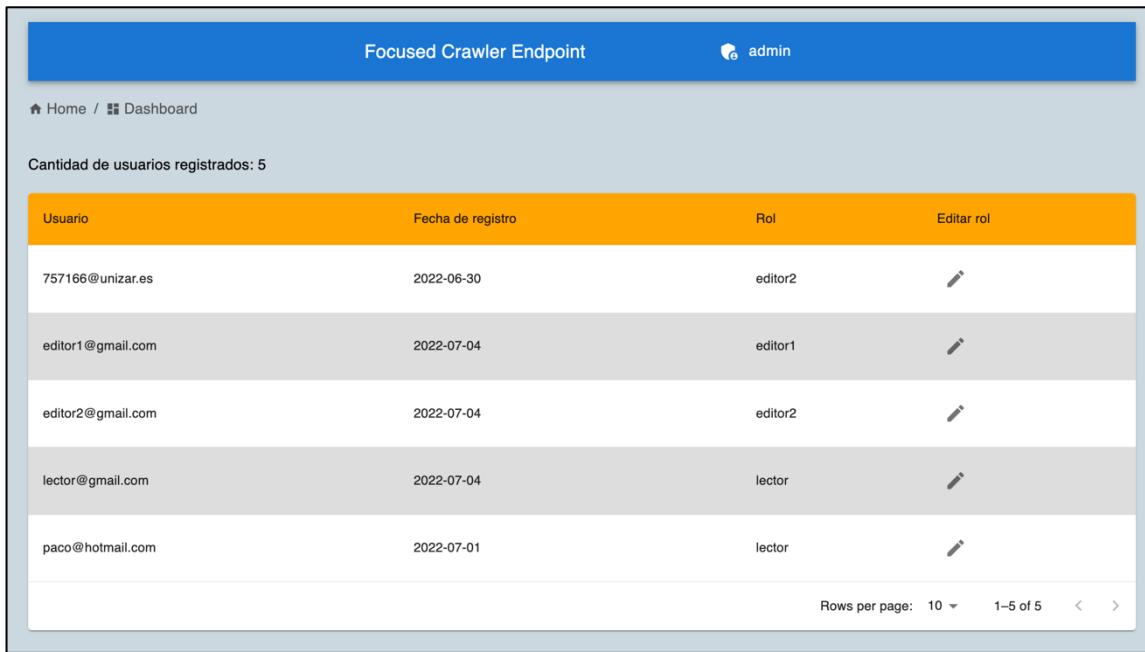


Figura 56. Vista de una acción permitida. Fuente: propia.

Este usuario accede al sistema con un usuario y contraseña y accede al mismo portal web que los demás usuarios, pero con la opción de editar y de aumentar la relevancia de cualquier documento ya que su rol así se lo permite. Además, el administrador puede

acceder a un panel de administración donde puede ver la cantidad de usuarios registrados en el sistema junto a información relevante. Esto se muestra en la Figura 57.



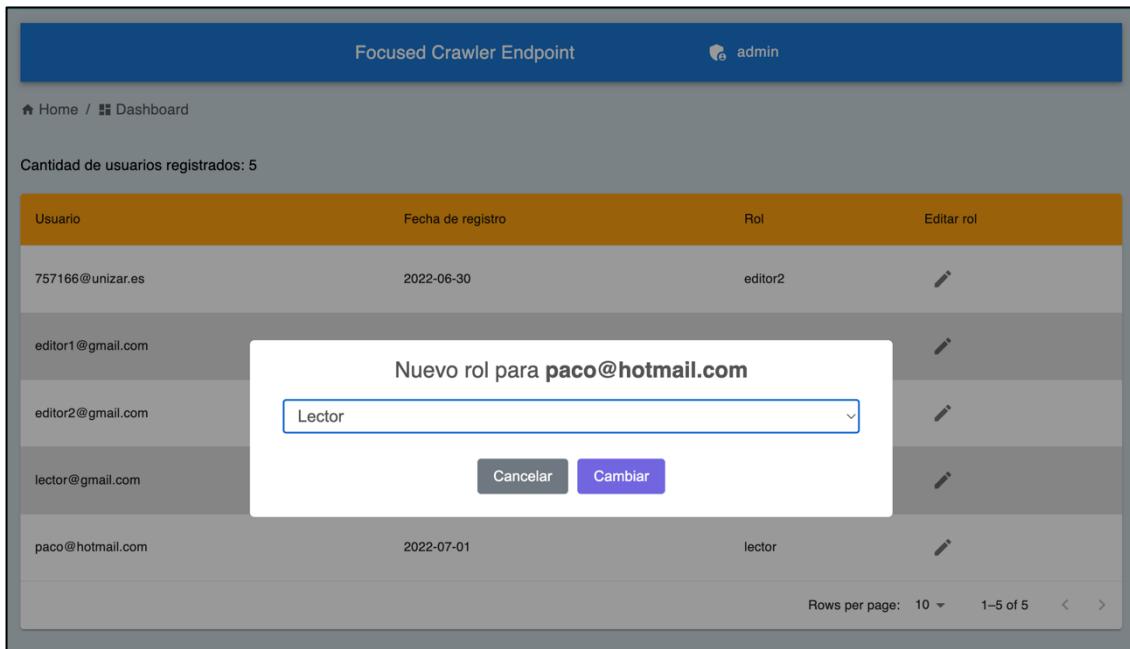
The screenshot shows a web-based administration interface for a crawler endpoint. At the top, a blue header bar displays the title "Focused Crawler Endpoint" and the user "admin". Below the header, a breadcrumb navigation shows "Home / Dashboard". A message "Cantidad de usuarios registrados: 5" is displayed above a table. The table has four columns: "Usuario", "Fecha de registro", "Rol", and "Editar rol". The data rows are:

Usuario	Fecha de registro	Rol	Editar rol
757166@unizar.es	2022-06-30	editor2	
editor1@gmail.com	2022-07-04	editor1	
editor2@gmail.com	2022-07-04	editor2	
lector@gmail.com	2022-07-04	lector	
paco@hotmail.com	2022-07-01	lector	

At the bottom right of the table area, there are pagination controls: "Rows per page: 10" and "1–5 of 5".

Figura 57. Vista del panel de administración. Fuente: propia.

Desde aquí, el administrador puede consultar la información de cualquier usuario, incluyendo su rol actual. Si lo desea, puede editar el rol de cualquier usuario, aumentando así sus privilegios dentro del sistema o disminuyéndolos. Un ejemplo de edición del rol de un usuario se muestra en la Figura 58.



The screenshot shows the same administration interface as Figure 57, but with a modal dialog box overlaid on the user list. The dialog is titled "Nuevo rol para paco@hotmail.com" and contains a dropdown menu with the option "Lector" selected. At the bottom of the dialog are two buttons: "Cancelar" and "Cambiar". The background table remains visible, showing the other registered users.

Figura 58. Vista del cambio de rol desde el panel de administración. Fuente: propia.

Como se puede apreciar, este usuario tiene el rol de Usuario (el más bajo posible, pudiendo únicamente ver el contenido, pero sin la posibilidad de editarlo de ninguna

forma). Entonces, gracias a esta opción, de edición el administrador podría escoger otro rol para este usuario y aumentar sus privilegios en el sistema.

F.2 Proceso de pruebas

Esta sección del Anexo tiene como objetivo detallar el proceso de pruebas mostrando un ejemplo de algunas de las pruebas realizadas. Como ya se ha podido observar en el apartado de Arquitectura del sistema, la aplicación web construida sirve de interfaz gráfica donde se pueden realizar diferentes pruebas para comprobar que el sistema funciona como debería.

A parte de la prueba mostrada en la sección 5.2, otra prueba interesante es comprobar los formularios, es decir, buscar páginas web del MAPA que contengan formularios y comprobar si están así catalogadas en el portal web. Para ello, se puede usar este enlace²⁷ como ejemplo de página web del MAPA con formulario. Para comprobar esto basta con filtrar por formularios en el portal web y por el término “tractores” para ver si aparece ese documento. El resultado de la búsqueda se muestra en la Figura 59.

The screenshot shows a web application titled "Focused Crawler Endpoint" with a user "admin" logged in. On the left, there are three filter panels: "Formatos" (Formats) listing .pdf (9178), .doc (142), .xml (12), .html (6241), .docx (184), .xlsx (774), and .pptx (8); "Páginas web que contienen..." (Web pages containing...) listing PDF (1879), DOCX (137), XLSX (131), MSWORD (53), XML (19), PPTX (4), and OTROS (1); and "Tipo de página" (Type of page) with buttons for Contenido, Noticia, and Formulario, where "Formulario" is selected. The main search area has a search bar with "tractores", a filter "TipoDePagina: Formulario", and a search term "mainSearch: tractores". It displays "5 de 6 resultados encontrados en 5 ms". The results are listed as follows:

- [Tractores agrícolas y estructuras de protección homologadas] - Agricultura - mapa.gob.es
src: <https://www.mapa.gob.es/app/Tractores/CirCon.aspx>
Relevancia: ★ ★ ★ ★ ★ (0)
Fecha de publicación: 2010-10-11
Buttons: HTML, Editar, Like
- [Tractores agrícolas y estructuras de protección homologadas] - Agricultura - mapa.gob.es
src: <https://www.mapa.gob.es/app/Tractores/CorCon.aspx>
Relevancia: ★ ★ ★ ★ ★ (0)
Fecha de publicación: 2010-10-11
Buttons: HTML, Editar, Like
- [Tractores agrícolas y estructuras de protección homologadas] - Agricultura - mapa.gob.es
src: <https://www.mapa.gob.es/app/Tractores/EsCon.aspx>
Relevancia: ★ ★ ★ ★ ★ (0)
Fecha de publicación: 2010-10-11
Buttons: HTML, Editar, Like
- [Tractores agrícolas y estructuras de protección homologadas] - Agricultura - mapa.gob.es
src: <https://www.mapa.gob.es/app/Tractores/TracCon.aspx>

Figura 59. Búsqueda de formularios. Fuente: propia.

Se puede apreciar justo debajo de la caja de búsqueda como se tiene seleccionado el filtro de formularios y el término “tractores” (ya que el formulario que se busca tiene esa temática). Se puede ver en el resultado de la búsqueda como el tercer documento es justo el que buscamos (las URL coinciden), comprobando así que se ha catalogado

²⁷ <https://www.mapa.gob.es/app/Tractores/EsCon.aspx>

correctamente. De paso, se demuestra que el primer resultado²⁸ es también un formulario, como se aprecia en la Figura 60.

The screenshot shows a search interface for 'Tractores agrícolas y estructuras de protección homologadas'. The left sidebar lists categories like 'Ayudas', 'Estación Mecánica Agrícola', and 'Base de datos'. The main area has a search bar and dropdown filters for 'Número Registro' and 'Tipo' (set to 'Cualquiera'). A 'Buscar' button is at the bottom right. The URL in the address bar is https://www.mapa.gob.es/app/Tractores/CirCon.aspx.

Figura 60. Formulario del MAPA. Fuente: propia.

Se han realizado también pruebas para comprobar que los documentos se han categorizado correctamente. Por ejemplo, se ha realizado un filtrado por categoría escogiendo “Alimentación”. Además, para acotar la búsqueda se le ha añadido el filtro de páginas web que contengan ficheros XML. Se muestra el resultado en la Figura 61.

The screenshot shows a search results page with a sidebar for file formats (.pdf, .doc, .xml, etc.) and a main area for 'Páginas web que contienen...'. A search bar filters by 'Categoría: Alimentación' and 'Contiene: XML'. Results include a 'Recopilaciones legislativas monográficas' section with a link to https://www.mapa.gob.es/es/alimentacion/legislacion/recopilaciones-legislativas-monograficas/default.aspx and an 'Últimas disposiciones publicadas' section with a link to https://www.mapa.gob.es/es/alimentacion/legislacion/ultimas-disposiciones-publicadas/default.aspx. Both sections have 'Editar' and 'Like' buttons. The URL in the address bar is https://www.mapa.gob.es/app/FocusedCrawlerEndpoint.aspx.

Figura 61. Filtro de documentos con categoría “alimentación” que contengan XML. Fuente: propia.

²⁸ <https://www.mapa.gob.es/app/Tractores/CirCon.aspx>

Al igual que anteriormente, se puede observar la aplicación de los filtros en las etiquetas situadas debajo de la caja de búsqueda. Los resultados que aparecen son correctos, cumplen los requisitos de los filtros. Después de inspeccionar en la web del MAPA ambos resultados, se puede comprobar que tanto el primero²⁹ como el segundo³⁰ pertenecen a la categoría de alimentación.

Otra prueba importante es la de edición y «me gusta» de documentos. Un ejemplo de prueba realizada es buscar un documento con información incompleta o incorrecta y editarla. Por ejemplo, se puede comprobar que este³¹ documento del MAPA tiene 17 PDF y 11 DOCX. No obstante, en el sistema solamente figuran 14 PDF y 11 DOCX, como se muestra en la Figura 62.

The screenshot shows a search interface for documents on the MAPA website. On the left, there are filters for file type (DOCX checked, XLSX, MSWORD, XML, PPTX, OTROS), page type (Contenido selected, Noticia, Formulario), publication date range (Fecha de inicio to Fecha de fin), category (Agricultura, Ganadería, Pesca, Alimentación, Desarrollo rural, Cartografía y SIG, Otros), and relevance (0 Estrellas to 5 Estrellas). The main area displays four document entries:

- Apoyo a Actuaciones de Cooperación en Proyectos Medioambientales**
src: https://www.mapa.gob.es/es/alimentacion/temas/integracion-asociativa/apoyos-integracion/PNDR_Cooperacion_Medioambientales_M16.5.aspx
Relevancia: 0 (0)
Fecha de publicación: 2020-12-11
PDF(14) DOCX(11) Editar Like
- Audiencia e información pública del Proyecto de Real Decreto sobre el alcance de la declaración de i**
src: <https://www.mapa.gob.es/es/desarrollo-rural/temas/gestion-sostenible-regadios/plan-nacional-regadios/informaciones-publicas/Audiencias-IPP-Proyecto-RD.aspx>
Relevancia: 0 (0)
Fecha de publicación: 2022-03-21
PDF(1) DOCX(1) Editar Like
- Cuerpo de Ingenieros Técnicos en Especialidades Agrícolas (Acceso Libre)**
src: https://www.mapa.gob.es/es/ministerio/servicios/empleo-publico/oposiciones/2020/agricolas_20_Libre.aspx
Relevancia: 0 (0)
Fecha de publicación: 2021-11-15
PDF(3) DOCX(2) Editar Like

At the bottom, there are navigation links: Prev, 1, 2, 3, 4, 5, Next.

Figura 62. Búsqueda de documento específico. Fuente: propia.

Gracias a la función de editar información, se consigue pasar de la Figura 62 a la Figura 63.

²⁹ <https://www.mapa.gob.es/es/alimentacion/legislacion/recopilaciones-legislativas-monograficas/default.aspx>

³⁰ <https://www.mapa.gob.es/es/alimentacion/legislacion/ultimas-disposiciones-publicadas/default.aspx>

³¹ https://www.mapa.gob.es/es/alimentacion/temas/integracion-asociativa/apoyos-integracion/PNDR_Cooperacion_Medioambientales_M16.5.aspx

The screenshot shows a web-based application titled "Focused Crawler Endpoint". At the top right, it says "Invitado". The main area has a search bar with the query "Apoyo a Actuaciones de Cooperación en Proyectos Medioambientales". Below the search bar, there is a message: "mainSearch: Apoyo a Actuaciones de C... X Clear All" and "5 de 4923 resultados encontrados en 25 ms".

Apoyo a Actuaciones de Cooperación en Proyectos Medioambientales
 src: https://www.mapa.gob.es/es/alimentacion/temas/integracion-asociativa/apoyos-integracion/PNDR_Cooperacion_Medioambientales_M16.5.aspx

Relevancia: ★ ★ ★ ★ ★ (0)
 Fecha de publicación: 2020-12-11

PDF(17) **DOCX(11)** **Edit** **Like**

Apoyo a Actuaciones de Cooperación en Suministros de Biomasa
 src: https://www.mapa.gob.es/es/alimentacion/temas/integracion-asociativa/apoyos-integracion/PNDR_Cooperacion_Biomasa_M16.6.aspx

Relevancia: ★ ★ ★ ★ ★ (0)
 Fecha de publicación: 2020-12-11

PDF(13) **DOCX(11)** **Edit** **Like**

Formatos

<input type="checkbox"/> .pdf	6972
<input type="checkbox"/> .doc	137
<input type="checkbox"/> .xml	12
<input type="checkbox"/> .html	5998
<input type="checkbox"/> .docx	183
<input type="checkbox"/> .xlsx	659
<input type="checkbox"/> .pptx	6

Páginas web que contienen...

Buscar tipo de fichero(s)	
<input type="checkbox"/> PDF	1770
<input type="checkbox"/> DOCX	125
<input type="checkbox"/> XLSX	115
<input type="checkbox"/> MSWORD	55
<input type="checkbox"/> XML	19
<input type="checkbox"/> PPTX	3

Figura 63. Documento con información editada. Fuente: propia.

La información se edita y ahora es al 100% consistente con la información del MAPA. Acto seguido, se hace la prueba de la funcionalidad de «me gusta» también. Antes de hacer esta prueba, se debe cerrar sesión y entrar con un usuario con privilegios. Una vez dentro, se aumenta la relevancia de un documento. Ahora, cuando se hace una búsqueda, se puede también filtrar por documentos que sean relevantes. Por ejemplo, los documentos con una o más estrellas de la categoría “Agricultura” se muestran en la Figura 64.

The screenshot shows a web-based application titled "Focused Crawler Endpoint". At the top right, there is a user icon labeled "admin". The main content area has a blue header bar with the text "Búsqueda genérica" and a search input field. Below this, a search filter bar shows "Categoría: Agricultura" and "Relevancia: 1, 5". A message indicates "3 de 3 resultados encontrados en 11 ms".

Result 1: [Consulta de productos fertilizantes] - Agricultura - mapa.gob.es
 src: <https://www.mapa.gob.es/app/consultafertilizante/consultafertilizante.aspx>
 Relevancia: ★☆☆☆☆ (1)
 Fecha de publicación: 2010-10-11

Result 2: [Catálogos nacionales y comunitarios] - Agricultura - mapa.gob.es
 src: <https://www.mapa.gob.es/app/regVar/default.aspx>
 Relevancia: ★★★☆☆ (3)
 Fecha de publicación: 2010-10-11

Result 3: Datos INFOVI año 2017
 src: https://www.mapa.gob.es/es/agricultura/temas/producciones-agricolas/vitivinicultura/infovi_2017.aspx
 Relevancia: ★☆☆☆☆ (1)
 Fecha de publicación: 2021-04-16

On the left sidebar, there are three sections: "Formatos" (listing file types like .pdf, .doc, .xml, etc.), "Páginas web que contienen..." (listing page types like PDF, DOCX, XLSX, etc.), and "Tipo de página" (listing Content, News, and Formulario). Below these is a "Filtrar por fecha de publicación" button.

Figura 64. Documento con dos puntos de relevancia. Fuente: propia.

Se pueden ver documentos de la categoría “Agricultura” que son relevantes para los usuarios. Es decir, quizá alguno de ellos contiene información muy útil relacionada con la Agricultura y que se pueden consultar fácilmente. Se pueden observar que dos documentos son formularios y otro documento es una página web que contiene ficheros PDF y XLSX.

Por último, cabe destacar en esta sección la importancia de la independencia de cada subsistema. En un punto del proceso de pruebas se detectó un error mediante el cual los ficheros que contenían los documentos no aparecían. Tras analizar bien cada subsistema, y ver que todo era correcto, se analizó el portal del MAPA y se detectó un cambio en el nombre de los metadatos. Esto produjo que se tuviera que modificar el fichero de configuración de Nutch, algunas propiedades de ElasticSearch y partes del código de la aplicación web. Gracias a la independencia entre subsistemas, los cambios se pudieron realizar rápido y de forma sencilla (cambiando sólo uno o dos ficheros en cada subsistema).

Anexo G Manual de usuario

Se aprovecha esta sección para adjuntar un pequeño manual de usuario que permite instalar y ejecutar el sistema localmente. Para poder ejecutar el *backend* es necesario tener instalada la herramienta node para poder ejecutar el comando “npm”. Para ello, esta página web contiene información de cómo hacerlo³². Una vez se tiene instalado el entorno de node (que instala por defecto el comando npm también), se revisa que este proceso haya sido exitoso. Para ello, se pide introducir en la terminal los siguientes comandos que se muestran en la Figura 65.

```
> node -v  
> npm -v
```

Figura 65. Comandos de comprobación. Fuente: propia.

Si la respuesta ofrecida por la terminal es la versión de cada uno de ellos, entonces se puede afirmar que las herramientas están instaladas.

El siguiente paso es acceder al repositorio de GitHub³³ y descargar el código contenido en ese repositorio. Se descargará un archivo zip, que hay que descomprimir. En su interior existen tres directorios, uno correspondiente a la aplicación web (“clienteMAPA”), otro correspondiente al *backend* (“backendFC”) y el último, correspondiente a los *scripts* de postproceso (“post-proceso”). Finalmente, para hacerse con el crawler y el motor de búsqueda, descargar y descomprimir el archivo zip que se encuentra disponible en esta URL³⁴.

En este punto, ya se tiene todo el material descargado y solamente queda ejecutar los diferentes subsistemas. En primer lugar, para ejecutar el motor de búsqueda, se introducen los comandos que se muestran en la Figura 66 en una terminal.

```
> cd focused-crawler/crawler/elasticsearch-7.4.2  
> bin/elasticsearch
```

Figura 66. Ejecución de ElasticSearch. Fuente: propia.

Ahora, ya tenemos el motor de búsqueda ejecutándose. Si se quiere acceder a él, se puede hacer desde la URL <http://localhost:9200>. En este dominio se puede ver el estado del servidor. No obstante, hay muchas funcionalidades que se pueden ver, por ejemplo, los índices que hay y la cantidad de documentos existentes en cada uno. Todo esto en la dirección http://localhost:9200/_cat/indices?v.

Para ejecutar la aplicación web, se deben introducir los comandos de la Figura 67 en otra terminal.

³² <https://www.cursogis.com/como-instalar-node-js-y-npm-en-4-pasos/>

³³ <https://github.com/pabloJordan24/TFG-2022>

³⁴ <https://drive.google.com/file/d/1zZ4v90DUrMuekvW7SqM2i4qEGndqRHH/view?usp=sharing>

```
> cd TFG-2022-main/clienteMAPA  
> npm install  
> npm start
```

Figura 67. Ejecución de la aplicación web. Fuente: propia.

Ya se tiene la aplicación web ejecutándose, accesible desde la URL <http://localhost:3000>. Para confirmar el correcto funcionamiento, debería aparecer la vista del inicio de sesión al acceder a la URL previamente mencionada. Finalmente, para ejecutar el *backend*, se deben introducir los comandos de la Figura 68.

```
> cd TFG-2022-main/backendFC  
> npm install  
> npm start
```

Figura 68. Ejecución del backend. Fuente: propia.

Para confirmar el éxito en su ejecución, al acceder a la dirección <http://localhost:3003/>, debe aparecer un mensaje confirmando que la API está ejecutándose. Si se quiere consultar la documentación de esta, acceder a <http://localhost:3003/api-doc>.

Una vez hecho esto, ya se puede comenzar a usar el sistema. Se puede usar como inicio de sesión la cuenta “757166@unizar.es” con contraseña “12345678” que se corresponde con un usuario con privilegios de edición. La cuenta de administrador es “admin@admin.com” con contraseña “1234admin” con privilegios totales. No obstante, también se puede usar el registro y entrar como nuevo usuario.

Si se quiere hacer un proceso de extracción nuevo para cargar datos recientes al portal, se debe contar con cierta memoria en el equipo ya que el proceso así lo requiere.

Antes de nada, se debe instalar Python3, por ejemplo, desde esta URL³⁵. Acto seguido, con el motor de búsqueda ejecutándose, se ejecuta el propio proceso de extracción, introduciendo en otra terminal el comando de la Figura 69.

```
> bin/crawl -i -s urls crawl 1
```

Figura 69. Ejecución de una iteración del proceso de crawl. Fuente: propia.

Dependiendo del equipo y entorno donde se ejecute, puede aparecer el siguiente mensaje al ejecutar el comando de la Figura 69: “Error: JAVA_HOME is not set”. Para

³⁵ <https://www.python.org/downloads/>

solucionarlo, es necesario añadir la variable de entorno “JAVA_HOME” al equipo, con los comandos que aquí se muestran³⁶.

Tras esperar un tiempo (será largo), se deben ejecutar los *scripts* de post proceso de Python, uno a uno. Para ello, desde otra terminal, se ejecutan los comandos de la Figura 70.

```
> cd TFG-2022-main/post-proceso  
> python3 postProcessing.py  
> python3 postProc_parent.py  
> python3 addScoring.py  
> python3uitarNoticias.py  
> python3 keywords.py
```

Figura 70. Ejecución de los procesos de análisis en Python. Fuente: propia.

Tras hacer esto, se vuelve a ejecutar la aplicación web y el *backend*, siguiendo los comandos que se muestran en las Figuras 67 y 68. En este punto, habrá más información en el sistema por la que se podrá buscar.

³⁶ <https://www.baeldung.com/java-home-on-windows-7-8-10-mac-os-x-linux>