

Análisis y desarrollo de un sistema de integración y comparación de noticias



Grado en Ingeniería Multimedia

Trabajo Fin de Grado

Autor:

Adrián Tomás Vañó

Tutor/es:

Gustavo Candela Romero

María Dolores Sáez Fernández



Universitat d'Alacant
Universidad de Alicante

Mayo 2019

Análisis y desarrollo de un sistema de integración y comparación de noticias

Autor

Adrián Tomás Vañó

Directores

Gustavo Candela Romero

Departamento de Lenguajes y Sistemas Informáticos

María Dolores Sáez Fernández

Departamento de Lenguajes y Sistemas Informáticos



GRADO EN INGENIERÍA MULTIMEDIA



Escuela
Politécnica
Superior



Universitat d'Alacant
Universidad de Alicante

ALICANTE, 1 de junio de 2019

Resumen

Los consumidores de prensa digital tienen acceso a mucha más información y muchas más fuentes informativas que nunca. El auge, crecimiento y popularidad de las tecnologías que dan soporte a este tipo de contenido es cada vez más notorio [Luis Muñoz, 2017] y cuenta con mejores características día tras día. En la actualidad, acceder a la información es muy sencillo, lo cual acarrea una contraparte: la sobrecarga informativa. Esta misma facilidad de consumo causa que el usuario se vea abrumado por el impacto generado por la cantidad de información que recibe y la atención que dedica a ello. Los usuarios quieren estar al corriente de demasiados temas y la tecnología debe dar una respuesta responsable a ello.

Este auge de la información conlleva que se hayan multiplicado problemas que antes existían de manera aislada, como lo es la manipulación de la información. La mayoría de los medios imprimen en su contenido su línea de pensamiento y para el usuario, que busque formar una opinión lo más objetiva posible, se convierte en una gran inversión de tiempo navegar entre distintas fuentes y acceder a la misma pieza de información.

Por tanto, las situaciones que se han detectado y a las que se pretende ofrecer una respuesta son:

- La dificultad del consumidor de prensa digital para contrastar información en distintos medios.
- El desaprovechamiento del tiempo que invierte el consumidor en leer contenido de su interés.

La propuesta que plantea este trabajo de final de grado parte del uso de tecnologías de análisis de lenguaje natural, que mejoren las características de un agregador de contenido a través del análisis del texto. Para ello se ha implementado una API que cuenta con un analizador de RSS y un extractor de entidades, utilizadas para hallar la similitud entre noticias. Se ha definido un esquema de datos que utilice un indexador para almacenar el contenido y, por último, se ha diseñado e implementado una capa de visualización para mostrar las noticias con múltiples opciones de filtrado.

Para llevar a cabo el desarrollo se ha utilizado contenido de medios de prensa digital nacionales con los que realizar las extracciones de datos, indexaciones y visualizaciones.

Preámbulo

En la actualidad existe una imperiosa necesidad de estar informado todo el tiempo acerca de la mayoría de los temas que suceden en el planeta. Dicha necesidad se ve acrecentada por el alcance, extensión y uso de las tecnologías de la información y la realidad de que cualquier hecho que se quiera conocer está inmediatamente disponible con solo un par de búsquedas o *clicks*.

Como usuario se es conocedor de la existencia de la gran diferencia de opinión entre los medios de prensa que sirven esta información. Cada editorial refleja sus intereses en el contenido que publica y para el lector, resulta una odisea e inversión de tiempo localizar y leer una misma noticia para contrastar información.

Esta intrincada situación, acompañada de mi ambición personal por conocer tecnologías distintas a las que habitualmente se utilizan y el interés en desarrollar una aplicación de principio a fin, dan lugar a la implementación y documentación de este proyecto.

Por lo tanto, se plantea un escenario de planificación, investigación, diseño, construcción y visualización de una aplicación capaz de mostrar noticias de distintas fuentes de prensa digital y relacionarlas entre sí por similitud para que el usuario pueda contrastar la información.

Quiero dedicar unas líneas para agradecer a mis tutores, sin los cuales el presente documento no sería más que una amalgama de ideas sin orden ni concierto.

A mi familia por mantenerme con vida todo este tiempo.

A Lluna por haber sido mi compañera y a mi tribu por su comprensión en mis ausencias.

Finalmente, a ESINEC por haberme apoyado en el sprint final.

*Just because something doesn't do
what you planned it to do
doesn't mean it's useless*

Thomas Edison

Índice general

Resumen	v
1. Introducción	1
2. Objetivos	3
2.1. Generales	3
2.2. Específicos	3
3. Marco Teórico	5
3.1. Servicios de contenido de medios digitales	5
3.1.1. Agregadores RSS	5
3.1.2. Servicios de seguimiento de prensa digital	6
3.2. Marco legal	6
3.2.1. Uso de contenido de medios digitales	6
3.2.2. Conflicto AEDE con Google Noticias	7
3.3. Tecnologías para el desarrollo	7
3.3.1. Lenguajes de programación	7
3.3.2. Tecnologías para la implementación del back-end	8
3.3.3. Tecnologías para la implementación del front-end	13
4. Modelo de negocio	15
5. Metodología	17
5.1. Hitos e iteraciones	17
5.1.1. Hito 1	17
5.1.2. Hito 2	18
5.2. Herramientas de administración del proyecto	19
5.3. Herramientas de desarrollo	20
6. Análisis y especificación	23
6.1. Especificación requisitos funcionales	23
6.2. Casos de uso	24
6.3. Diagrama de actividad	24
7. Diseño	27
7.1. Arquitectura seleccionada	27
7.2. Tecnologías	28
7.3. Diagrama de componentes	28

7.4. Mockups	29
7.4.1. Hito 1	30
7.4.2. Hito 2	31
8. Implementación	33
8.1. Hito 1	33
8.1.1. Iteración I	33
8.1.2. Iteración II	33
8.1.3. Iteración III	38
8.1.4. Iteración IV	41
8.2. Hito 2	43
8.2.1. Iteración VI	44
8.2.2. Iteración VII	46
9. Despliegue en producción	49
10. Resultados	51
11. Conclusiones	55
11.1. Conclusiones	55
11.2. Líneas de trabajo futuras	55
Bibliografía	57
A. Anexo A. Esquema de base de datos	59
B. Anexo B. Requisitos funcionales	61
C. Anexo C. Archivos Docker	69

Índice de figuras

3.1. Lenguajes más populares en la encuesta de 2018 de Stack Overflow	8
3.2. Motores de búsqueda más populares [db engines, 2019]	10
3.3. Soporte oficial de idiomas en CoreNLP [Stanford, 2019]	11
4.1. Lean Canvas de Contrast	16
5.1. Espacio de trabajo para Contrast	19
5.2. Tablero para Contrast	20
5.3. Integración IFTT de Todoist con Trello	21
6.1. Casos de uso	24
6.2. Diagrama de actividad de la función ampliar noticia	25
6.3. Diagrama de actividad de la función filtrar	26
7.1. Stack tecnológico utilizado por Contrast	27
7.2. Diagrama de componentes	29
7.3. Prototipo de interfaz para página de inicio y resultado búsqueda	30
7.4. Prototipo de interfaz para noticia ampliada	30
7.5. Diseño final de interfaz para página de inicio y resultado búsqueda	31
7.6. Diseño final de interfaz para noticia ampliada	32
8.1. Interfaz de usuario gráfica de Apache Solr	35
8.2. Demostración de extracción de entidades en CoreNLP	36
8.3. Demostración de extracción de entidades en Freeling	36
10.1. Vista de inicio de Contrast	52
10.2. Vista de noticia ampliada de Contrast	53

Índice de tablas

B.1. Requisito funcional Listar noticias	61
B.2. Requisito funcional Buscar noticias	61
B.3. Requisito funcional Filtrar noticias por periódico	62
B.4. Requisito funcional Filtrar noticias por rango de fechas	63
B.5. Requisito funcional Filtrar noticias por entidades	64
B.6. Requisito funcional Ampliar noticia	64
B.7. Requisito funcional Buscar noticias similares	65
B.8. Requisito funcional Indexar noticias	65
B.9. Requisito funcional Mantener formato	66
B.10.Requisito funcional Obtener noticias	66
B.11.Requisito funcional Facetar por contenido	67
B.12.Requisito funcional Pagar noticias	67
B.13.Requisito funcional Reconocer entidades	68

Índice de Listados

8.1. Universal Feed Parser	34
8.2. Rome Tools	34
8.3. rss-parser	35
8.4. Declaración de un botón en Vuetify	37
8.5. Declaración de un botón en Bootstrap	37
8.6. Funciones de Rome para la consecución del esquema mínimo de datos . .	38
8.7. Funciones de Rome para la extracción de los campos adicionales para el esquema de datos óptimo	38
8.8. Eliminación de etiquetas HTML del cuerpo de la noticia	39
8.9. Definición del modelo de datos	39
8.10. Interfaz del repositorio Solr	40
8.11. Configuración de conexión a cliente Solr	40
8.12. Controlador de la API de Contrast	41
8.13. Proxy inverso del front-end al back-end	42
8.14. Clase de reconocimiento de entidades	42
8.15. Uso del extractor de entidades y convergencia de ambos datos	43
8.16. Campo de datos personalizado	44
8.17. Endpoints finales de obtención de noticias	45
8.18. Clase final extractora de entidades	46
9.1. Configuración para puesta en producción de Vue.js	50
A.1. Esquema de datos mínimo utilizado	59
A.2. Esquema de datos óptimo utilizado	59
C.1. Archivo docker-compose.yml	69
C.2. Dockerfile de Solr	70
C.3. Dockerfile de Vue.js	70
C.4. Dockerfile de Java	70

1. Introducción

El consumo de prensa digital de pago aumentó un 157,89 % de 2011 a 2016. Así se refleja en el estudio de uso y actitudes de consumo de contenidos digitales [Luis Muñoz, 2017] publicado por el Observatorio Nacional de Telecomunicaciones y Sociedad de la Información (ONTSI). En el año 2016, el 66,2 % de la población adulta española consumió contenido de medios de prensa digital y el 48,4 % de la población lo realizó a diario, ya que la lectura de noticias y artículos digitales es el tipo de consumo de contenidos digitales más frecuente. Adicionalmente, la mayoría de lectores de noticias digitales lo realizan de manera gratuita y, además, desde sitios web distintos a los originales que ofrecen la información.

Por otro lado, se considera la pluralidad de la diversidad de opinión y en especial su extremo: la manipulación de información en los medios de prensa [Laura Uche, 2017]. El consumidor de noticias, sin importar el formato, se ve expuesto a recibir una narración subjetiva de la realidad, incluso viéndose, en cierto modo, forzado, a contrastar la información entre distintos medios. Este hecho parece haberse incrementado en los últimos años debido a la propagación de las TIC y a la aparente necesidad de estar informado y por tanto formar una opinión sobre cualquier tema público o privado.

Los consumidores de contenido de medios de prensa digital que desean obtener una determinada información de distintos medios se enfrentan a la necesidad de acceder a cada uno de los sitios webs y realizar la búsqueda o viceversa, por lo que se ven afectados al no gozar de la posibilidad de consumir contenido de diversos medios en un mismo portal.

A la hora de realizar un análisis de opinión o seguimiento sobre un tema concreto, los usuarios deben contratar servicios especializados para ello, pues las posibilidades gratuitas se limitan al número de búsquedas, alcance, etc de unas determinadas palabras clave pero no tienen en cuenta el contenido ni semántica de la información en cuestión.

La propuesta que plantea este trabajo de final de grado, es el desarrollo de una aplicación que proporcione una respuesta parcial o completa a las situaciones mencionadas anteriormente.

En primer lugar se pretende implementar un agregador de contenido de diversas fuentes que permita la lectura unificada de noticias de medios digitales en un único lugar. Esto facilitará la experiencia de usuario en el consumo de contenido digital, debido a que se permanecerá en un único lugar para la lectura de noticias y se disminuirán los tiempos y números de interacciones necesarias del usuario.

Para dotar la aplicación con características avanzadas, se propone utilizar una herramienta de análisis de lenguaje natural para comparar y relacionar las noticias entre sí. La herramienta será utilizada para la obtención de palabras clave a través de las entidades presentes en el texto, de tal manera que se facilite la equiparación entre contenidos

independientemente de las palabras clave que haya definido el medio digital.

Para el desarrollo del proyecto, se ha propuesto analizar y estudiar la situación actual sobre el uso de contenidos de medios digitales y sus aspectos legales. Del mismo modo, se investigarán distintas propuestas de tecnologías que den una solución óptima para la correcta implementación de la aplicación.

Bajo este contexto y con la anterior propuesta expuesta, se plantea la implementación de una aplicación denominada Contrast, haciendo referencia al contraste entre información, clave principal del desarrollo del proyecto.

2. Objetivos

2.1. Generales

El objetivo general del proyecto es crear un lector de prensa digital capaz de relacionar noticias según su contenido, de tal manera que, partiendo de una noticia determinada, se realice una selección de noticias ordenadas por similitud.

2.2. Específicos

A continuación se detallan los objetivos específicos que parten del objetivo general. Se han desgranado en ítems independientes entre sí en la mayor medida posible, de tal manera que su consecución sea medible.

1. Aplicar una metodología de desarrollo de software iterativa en la implementación del proyecto.
2. Estudiar los servicios y herramientas existentes que realizan funciones similares a las propuestas en el objetivo general. Analizar las tecnologías sobre las que implementar el proyecto de manera idónea.
3. Investigar los aspectos legales acerca del uso de contenido de medios digitales de terceros.
4. Implementar un recuperador y analizador de contenido RSS que sirva para obtener los medios de los que se vale la aplicación.
5. Utilizar un motor de búsqueda para indexar las noticias obtenidas mediante RSS. Definir un esquema de datos mínimo y óptimo adecuado al contexto de la aplicación.
6. Construir una API que actúe como herramienta de transacción de noticias con la base de datos. Dicha API debe proveer métodos para recuperar y filtrar contenido.
7. Utilizar una herramienta de procesamiento de lenguaje natural que proporcione los extractores necesarios para obtener información relevante que permita determinar la similitud entre noticias.
8. Diseñar un visor de noticias que permita la lectura de estas a través de su comunicación con la API. Dicho visor debe implementar diferentes opciones para el filtrado de las noticias, así como tener una interfaz que promueva la usabilidad.

3. Marco Teórico

El siguiente capítulo trata el estado de las tecnologías existentes que ofrecen un servicio similar al que plantea Contrast en alguna de sus funcionalidades. Por otra parte, se trata el contexto en el que se engloba la aplicación a desarrollar.

3.1. Servicios de contenido de medios digitales

En este apartado se detallan dos posibles conjuntos de servicios sobre los que Contrast comparte cierta similitud en cuanto a funcionalidad. Los servicios citados están disponibles en internet.

3.1.1. Agregadores RSS

Existe una gran cantidad de aplicaciones con diferentes características y funcionalidades, aunque similares en uso. Dichos servicios permiten la búsqueda de fuentes de contenido para su lectura y la suscripción a temas de interés, que a su vez suscribe a varias fuentes relacionadas con el tema.

- **Feedly**¹ es el lector de contenido más popular actualmente. Cuenta con una interfaz clara y simple y permite consumir contenido en diversos formatos: vídeo, noticias de periódicos, tweets, publicaciones de blogs, alertas Google Keywords y por supuesto, RSS. Posee herramientas de integración con servicios de terceros como Evernote o Trello y permite crear tableros compartidos con otros usuarios.
- **Flipboard**² ha sido el agregador de noticias más utilizado durante mucho tiempo y una red social en sí. Posee integraciones con las principales redes sociales para leer contenido de ellas y compartir noticias en las redes desde la aplicación. Tiene un diseño cuidado que asemeja el uso de un magazín físico en cuanto a estructura de las secciones y el deslizamiento de hojas.
- **Inoreader**³ se define entre los usuarios como la alternativa a los anteriores cuando hay algún aspecto entre ellos que no termina de encajar con el usuario. Este lector de contenido posee herramientas para etiquetar y organizar automáticamente la información entrante en busca de ahorro de tiempo por parte del usuario.

¹<https://feedly.com/>

²<https://flipboard.com/>

³<https://www.inoreader.com/>

3.1.2. Servicios de seguimiento de prensa digital

El seguimiento de medios consiste en la detección de toda información publicada sobre un tema concreto. Estos datos se analizan en pos de categorizarlos y mejorar la distribución de los mismos. Los clientes que contratan estos servicios buscan mejorar su presencia digital al mismo tiempo que ser notificados de cualquier mención a su empresa y a su reputación.

- **Pressclipping**⁴: esta empresa ofrece análisis de medios y evaluación de noticias, además del propio seguimiento de medios. Cuenta además con soluciones para la gestión de las relaciones públicas y del marketing estratégico para medir con precisión el alcance y reputación de una marca.
- **Puntonews**⁵: servicio que permite buscar noticias tanto de periódico, radio como de televisión y exportarlas a PDF.
- **MyNews**⁶: hemeroteca digital desde 1996. Dispone de seguimiento de formatos audiovisuales y redes sociales, al igual que análisis de presencia y reputación de marca.

3.2. Marco legal

3.2.1. Uso de contenido de medios digitales

La siguiente sección detalla los aspectos legales a tener en consideración a la hora de utilizar el contenido de medios de prensa digital. Para ello se han investigado las licencias de distintos periódicos españoles, encontrando grandes diferencias entre ellos.

Por una parte, existen medios que utilizan una licencia Creative Commons, eldiario.es,⁷ particularmente, utiliza la más abierta: la BY-SA⁸. Esta licencia permite copiar, difundir y remezclar el contenido en internet, incluso para fines comerciales siempre y cuando se utilice el mismo tipo de licencia. Esto no exime de citar y enlazar la fuente y el autor del contenido que se utilice.

Otros medios como EL PAÍS,⁹ especifican con claridad la prohibición sobre la reproducción, distribución y comunicación pública de sus contenidos para fines comerciales sin la autorización del medio.

El periódico EL MUNDO,¹⁰ define los mismos derechos que el medio anterior solo que su prohibición se aplica para cualquier fin, con el añadido de la oposición a que pueda ser citado la reproducción de su contenido. En general, este es el tipo de licencia que más se ha encontrado durante el estudio de esta sección.

⁴<https://www.pressclipping.com/>

⁵<https://www.puntonews.com/es>

⁶<https://www.mynews.es/>

⁷<https://www.eldiario.es/licencia/>

⁸<https://creativecommons.org/licenses/by-sa/3.0/deed.es>

⁹<https://elpais.com/estaticos/aviso-legal/index.html>

¹⁰<https://www.elmundo.es/registro/avisolegal.html>

3.2.2. Conflicto AEDE con Google Noticias

A continuación se describe uno de los sucesos más relevantes, que ha acontecido en España en la última década, al respecto del uso de contenido de medios digitales.

Google Noticias es un agregador y buscador de noticias de medios de prensa digital que cuenta con más de 70 ediciones en 35 idiomas y que genera más de 10.000 millones de interacciones al mes. Esta herramienta, nacida en 2002, no está disponible en España desde hace unos años.

El 5 de noviembre de 2014, el Partido Popular aprueba la nueva ley de propiedad intelectual,¹¹ causando que Google se viese obligado a pagar una tasa exigida por la Asociación de Editores de Diarios Españoles (AEDE). Dicho de otra manera, ahora tenía que pagar por mostrar cualquier fragmento del contenido de los editores españoles.

Cabe destacar que Google no tenía un modelo de negocio con este servicio, pues no mostraba anuncios, tan solo enlazaba los titulares con la noticia en el medio digital.

La ley aprobada entraba en vigor el 1 de enero de 2015 y el 16 de diciembre Google anunciaba que retiraba el servicio de noticias en España. Los editores contrastaron tiempo después que la plataforma de Google les aportaba tráfico y por lo tanto ingresos por anuncios, por ello pidieron al gobierno y a la Unión Europea que intervinieran, abogando que el gigante de la tecnología tenía la posibilidad de negociar la tasa que debía pagar. Google declinó la propuesta. En el período en que se redacta este documento, el exilio de Google Noticias en España sigue existiendo, al igual que la ley que lo forzó.

3.3. Tecnologías para el desarrollo

En esta sección se detallan las tecnologías sobre las que puede servirse el proyecto para su desarrollo. Por un lado se abarcan los lenguajes de programación, por otro las herramientas pertenecientes al back-end y por último las que atañen al front-end.

3.3.1. Lenguajes de programación

Se han analizado los lenguajes más populares en la última encuesta publicada en Stack Overflow,¹² que permiten tanto el desarrollo front-end como back-end, aunque se elijan opciones distintas para cada apartado.

- **Javascript**¹³ fue el primer lenguaje de scripting para web. Es uno de los lenguajes más usados en web tanto para el back-end como para el front-end, integrado en la gran cantidad de frameworks que se han construido sobre él. Con la salida de ECMAScript 6 se han solucionado muchos problemas que acarreaba el uso de Javascript y se han añadido nuevas funcionalidades que lo convierten en aún más interesante. Tiene una de las bases de usuarios más sólidas entre todos los lenguajes.

¹¹<https://www.boe.es/boe/dias/2014/11/05/pdfs/BOE-A-2014-11404.pdf>

¹²<https://insights.stackoverflow.com/survey/2018/>

¹³<https://www.javascript.com/>



Figura 3.1.: Lenguajes más populares en la encuesta de 2018 de Stack Overflow

- **Java**¹⁴ es un lenguaje con madurez y con muchos proyectos construidos sobre él que dependen de la instalación de Java en la máquina local. Se caracteriza por la solidez, confiabilidad, extensibilidad y rapidez. Existen multitud de frameworks y plugins que operan sobre Java y la documentación y comunidad es más que extensa. A este razonamiento se le suma el hecho de que es el lenguaje para el desarrollo en Android.
- **Python**¹⁵ es un lenguaje en alza desde los últimos años. Está diseñado partiendo de la intuitividad y del aproximamiento al lenguaje humano. La popularidad del lenguaje se debe al crecimiento del big data y del internet de las cosas, ya que ofrece algunas herramientas para el prototipado y las analíticas. Es un lenguaje ideal para principiantes debido a su flexibilidad.

<https://www.quora.com/Which-language-has-the-best-future-prospects-Python-Java-or-JavaScript> <https://stackshare.io/stackups/java-vs-javascript-vs-python>

3.3.2. Tecnologías para la implementación del back-end

El siguiente apartado contiene un sumario de tecnologías que facilitan la implementación y puesta en funcionamiento del back-end del proyecto.

¹⁴<https://www.oracle.com/technetwork/java/javase/overview/index.html>

¹⁵<https://www.python.org/>

Analizador RSS

Un analizador RSS es una herramienta que recolecta contenido RSS de distintas fuentes y lo traduce en una clase u objeto sobre el que se puede interactuar para obtener información específica. En este caso, se ha analizado la herramienta más popular para este cometido en cada uno de los lenguajes citados anteriormente.

- **rss-parser**¹⁶ es una librería mínima para convertir contenido RSS XML en objetos Javascript. Se compone de una función asíncrona que utiliza en un capa inferior un conversor de XML a objeto Javascript llamado `xml2js`¹⁷.
- **Universal Feed Parser**¹⁸ es un módulo Python para descargar y parsear feeds. Es un plugin simple, autocontenido en un único fichero y con una única función primaria para parsear. Tiene una documentación extensa y funciones adicionales para sanitizar y parsear otros elementos contenidos en el feed. Trata también con el formato Atom.
- **Rome**¹⁹ es un framework de Java de código abierto para tratar con RSS y Atom. Incluye analizadores, generadores y conversores entre ambos formatos. Los analizadores, el objeto de interés de Contrast, devuelve generalmente un objeto genérico que permite trabajar sin importar el formato de entrada o salida. Está compuesto por distintos módulos según se realice una acción concreta u otra. Posee una documentación sólida y es ampliamente utilizado para este fin.

Motores de búsqueda e indexadores

Los sistemas de recuperación de información con índice invertido,²⁰ son sistemas caracterizados por tener una estructura de datos capaz de manejar grandes volúmenes de información orientados a texto. Esta funcionalidad está presente en los motores de búsqueda que se han investigado y que además son los tres más populares.

- **Apache Solr**²¹ está construido sobre **Apache Lucene**²². **Lucene** es una potente librería de recuperación de información. Se encarga de construir el índice invertido mencionado anteriormente, una estructura especializada en emparejar documentos de texto con términos de consulta. **Lucene**, al igual que otros motores de esta clase, se distingue por la escalabilidad, el rápido despliegue, el manejo de grandes volúmenes de datos, la optimización orientada a búsqueda y esta a su vez, en textos. **Solr** por tanto es un servidor open source construido con la librería de **Lucene**, que incluye en una interfaz web de administración, APIs en distintos

¹⁶<https://www.npmjs.com/package/rss-parser>

¹⁷<https://www.npmjs.com/package/xml2js>

¹⁸<https://pythonhosted.org/feedparser/>

¹⁹<https://rometools.github.io/rome/>

²⁰<http://www.sisorbe.com/ExamenFinalRI/invertidos.html>

²¹<https://lucene.apache.org/solr/>

²²<https://lucene.apache.org/>

18 systems in ranking, December 2018


















Rank			DBMS	Database Model	Score		
Dec 2018	Nov 2018	Dec 2017			Dec 2018	Nov 2018	Dec 2017
1.	1.	1.	Elasticsearch 	Search engine	144.70	+1.24	+24.92
2.	2.	 3.	Splunk	Search engine	82.18	+1.81	+18.39
3.	3.	 2.	Solr	Search engine	61.35	+0.47	-4.95
4.	4.	4.	MarkLogic	Multi-model 	14.28	+0.81	+3.13
5.	5.	5.	Sphinx	Search engine	7.82	+0.46	+1.79
6.	6.	6.	Microsoft Azure Search	Search engine	5.67	+0.36	+1.56
7.	7.	7.	Algolia	Search engine	3.62	-0.13	+0.57
8.	8.	 9.	Google Search Appliance	Search engine	2.93	+0.03	+0.20
9.	9.	 8.	Amazon CloudSearch	Search engine	2.64	-0.11	-0.09
10.	10.	10.	Xapian	Search engine	0.60	-0.08	-0.02
11.	11.	11.	CrateDB 	Multi-model 	0.44	-0.06	-0.15
12.	12.	12.	SearchBlox	Search engine	0.24	0.00	+0.01
13.	13.	 14.	searchxml	Multi-model 	0.09	-0.01	+0.09
14.	14.	 13.	DBSight	Search engine	0.06	+0.00	+0.05
15.	15.	 14.	Manticore Search	Search engine	0.04	-0.01	+0.04
16.			FinchDB	Multi-model 	0.03		
17.	 16.	 14.	Exorbyte	Search engine	0.01	+0.01	+0.01
18.	 16.	 14.	Indica	Search engine	0.00	±0.00	±0.00

Figura 3.2.: Motores de búsqueda más populares [db engines, 2019]

lenguajes para la realización de consultas y una serie de mejoras funcionales sobre las características base de **Lucene**.

- **Elasticsearch**²³, también construido sobre **Lucene**, ha basado su modelo en una API REST Web al ser un motor de búsqueda más reciente y que por ello, ha ganado popularidad desde la última mitad de década. **Elasticsearch** tiene una destacable ejecución en las consultas analíticas y es apreciado por ser muy intuitivo, especialmente entre desarrolladores que se inician en este tipo de tecnologías. Esta tecnología es muy valorada por su escalabilidad.
- **Splunk**²⁴ es un motor de búsqueda y plataforma orientado a las analíticas para el Big Data, es por ello que se ha desestimado su análisis.

Herramientas de procesamiento de lenguaje natural

Las tecnologías de procesamiento de lenguaje natural son un conjunto de herramientas lingüísticas, que permiten extraer información relativa a características léxicas, morfo-sintácticas y semánticas de un determinado texto. Existen muchas propuestas en este ámbito, entre ellas destacan las siguientes.

- **CoreNLP**²⁵ es un proyecto de la Universidad de Stanford que provee una serie

²³<https://www.elastic.co/es/>

²⁴<https://www.splunk.com/>

²⁵<https://stanfordnlp.github.io/CoreNLP/>

de herramientas de análisis de lenguaje humano. Proporciona instrumentos para un amplio análisis gramático, además de extractores muy diversos. CoreNLP [Manning et al., 2014] posee una documentación extensa ya que es ampliamente utilizado, complementándose además con actualizaciones frecuentes que mejoran el producto. Posee soporte oficial para seis idiomas (entre ellos el español), aunque no están disponibles todas las funciones, como se puede observar en la figura 3.3 de la página 11. La integración con otros proyectos se realiza a través de una serie de APIs en distintos lenguajes. Esta herramienta es utilizada por la Biblioteca Virtual Miguel de Cervantes,²⁶ concretamente el analizador sintáctico.

ANNOTATOR	AR	ZH	EN	FR	DE	ES
Tokenize / Segment	✓	✓	✓	✓		✓
Sentence Split	✓	✓	✓	✓	✓	✓
Part of Speech	✓	✓	✓	✓	✓	✓
Lemma			✓			
Named Entities		✓	✓		✓	✓
Constituency Parsing	✓	✓	✓	✓	✓	✓
Dependency Parsing		✓	✓	✓	✓	
Sentiment Analysis			✓			
Mention Detection		✓	✓			
Coreference		✓	✓			
Open IE			✓			

Figura 3.3.: Soporte oficial de idiomas en CoreNLP [Stanford, 2019]

- **OpenNLP**²⁷ es un proyecto Apache, consistente en una librería de aprendizaje automático cuyo objetivo es el procesamiento del lenguaje humano en texto. Posee soporte para las tareas NLP más comunes. No tiene soporte al uso de distintos lenguajes, pero posee una serie de modelos entrenados en diferentes idiomas para propósitos determinados. En el caso del español encontramos cuatro modelos entrenados en reconocimiento de entidades: personas, organizaciones, localizaciones

²⁶<http://data.cervantesvirtual.com/analizador-sintactico-automatico>

²⁷<https://opennlp.apache.org/>

y misceláneo. Cualquier extractor de interés que no esté en los modelos tendría que ser entrenado para su propósito.

- **Freeling**²⁸ es un proyecto de la Universitat Politècnica de Catalunya. Se trata de una librería con una serie de herramientas open source para el análisis de lenguaje humano. Está escrita en C++ y posee una extensa variedad de utilidades, entre ellas las de interés para el proyecto. Tiene soporte para muchos idiomas y dialectos españoles, pero carece de una comunidad sólida que utilice la herramienta de manera asidua, al igual que una documentación no tan extensa como las propuestas anteriores.

Frameworks de desarrollo

Para esta sección se ha detallado un framework por cada lenguaje de programación que se ha propuesto previamente. Esta selección de frameworks ha sido realizada bajo criterios de popularidad, mantenimiento, calidad en la documentación, extensibilidad e integración con el resto de herramientas propuestas.

- **Spring Boot**²⁹ es un framework basado en Spring para Java. Abstrae el proceso de elección de dependencias y despliegue del proyecto, realizando en cambio un proceso sencillo y automatizado. El objetivo principal de esta herramienta es proporcionar un conjunto de herramientas para construir rápidamente aplicaciones en el potente entorno Spring. Se trata de un framework maduro, con resultados profesionales e integraciones con otros productos de Spring que añaden más funcionalidades. Cabe destacar que posee una capa de abstracción³⁰ para el motor de búsqueda Solr, al igual que otra para Elasticsearch³¹, aunque la primera está más consolidada.
- **Express.js**³² es un framework rápido, minimalista y flexible para el entorno de ejecución Node.js para Javascript. Proporciona un conjunto sólido de características para realizar APIs a través de métodos y utilidades HTTP y middleware que garantizan una creación rápida y sencilla.
- **Django**³³ es un framework full-stack para Python que facilita rapidez y claridad, ya que se hace cargo de muchos conceptos que conlleva el desarrollo web. Es una herramienta altamente escalable debido a su flexibilidad y proporciona ayuda a los desarrolladores para evitar muchos errores comunes de seguridad.

²⁸<http://nlp.lsi.upc.edu/freeling/node/1>

²⁹<https://spring.io/>

³⁰<https://spring.io/projects/spring-data-solr>

³¹<https://spring.io/projects/spring-data-elasticsearch>

³²<https://expressjs.com/>

³³<https://www.djangoproject.com/>

3.3.3. Tecnologías para la implementación del front-end

El siguiente apartado contiene un sumario de tecnologías que facilitan la implementación del front-end y diseño de interfaces del proyecto. Por cada una de estas, se han detallado dos o tres propuestas en función de la popularidad, extensión y uso.

Frameworks de desarrollo

Prácticamente la totalidad de propuestas para el desarrollo del front-end se basan en Javascript, pues es un lenguaje web interpretado ejecutado en el lado del cliente. En este caso, se detalla un conjunto de frameworks con gran popularidad en la actualidad.

- **Angular**³⁴ es una plataforma para diseñar aplicaciones multidispositivo creada por Google que facilita el desarrollo de aplicaciones web SPA, proporcionando herramientas para trabajar con los elementos de la página web de una manera óptima. Tiene una gran comunidad detrás y una documentación extensa, pero en cierto modo fuerza al usuario a aprender TypeScript.
- **Vue**³⁵ es un framework progresivo con un ecosistema que se plantea cercano, versátil y con gran rendimiento. Esta herramienta, diseñada para construir interfaces de usuario reactivas, se creó con el propósito de ser fácil de integrar con otras librerías o proyectos a través del desacople de las partes que lo conforman. Es un framework relativamente nuevo que mejora algunos conceptos de Angular y React, a través de una documentación clara y detallada.
- **React**³⁶ es una librería para construir interfaces de usuario reactivas SPA. Esta herramienta de Facebook permite la creación de aplicaciones web potentes que requieran un intercambio de datos constante. Tiene una curva de aprendizaje algo elevada que recompensa con eficiencia y responsividad.

Librerías de diseño de interfaces y componentes

En este apartado se detallan tres librerías de diseño de interfaces, que complementan el front-end con animaciones y elementos preconstruidos. La selección ha sido realizada en base al número de estrellas que acumula cada proyecto en Github.

- **Bootstrap**³⁷ es la librería de diseño más popular para construir aplicaciones web responsive pensadas en el mobile first. Dispone elementos que al utilizarlos aplican diferentes propiedades HTML, CSS y JS con un esfuerzo mínimo. Posee una gran comunidad detrás que además contribuye con plugins extra.
- **Vuetify**³⁸ es una librería de componentes semánticos para Vue.js específicamente. Su objetivo es proporcionar componentes claros, significantes y reutilizables que

³⁴<https://angular.io/>

³⁵<https://vuejs.org/>

³⁶<https://reactjs.org/>

³⁷<https://getbootstrap.com/>

³⁸<https://vuetifyjs.com/>

faciliten la implementación de una aplicación. Está construido de acuerdo con las especificaciones de Material Design.³⁹

- **Semantic UI**⁴⁰ posee el enfoque en realizar más semántico el proceso de construcción de una página. La principal funcionalidad consiste en utilizar principios del lenguaje natural para hacer el código más legible y fácil de entender. Posee una documentación bien organizada y con múltiples guías para empezar.

³⁹<https://material.io/design/>

⁴⁰<https://semantic-ui.com/>

4. Modelo de negocio

Tras el estudio del contexto del proyecto, el análisis de las aplicaciones similares existentes y la investigación de las tecnologías para llevarlo a cabo, se propone la realización del estudio de viabilidad del proyecto con el afán de hacer una declaración de intenciones acerca de los puntos clave que diferencian la aplicación a implementar. El estudio se lleva a cabo para reforzar los objetivos propuestos y conocer en qué aspectos hay que incidir en el desarrollo para el funcionamiento planteado.

Para el cometido propuesto se utilizará el lienzo Lean Canvas, [Maurya, 2012] una herramienta muy extendida para diseñar el escenario de negocio en el que se va a mover la aplicación; especialmente útil en casos centrados en el servicio ofrecido. El Lean Canvas realizado queda reflejado en la figura 4.1.

PROBLEM Dificultad para leer noticias de distintos periódicos en un mismo lugar Dificultad para contrastar información entre periódicos El seguimiento de medios para usos no-comerciales es caro	SOLUTION Agregador de prensa digital de diferentes fuentes Uso de tecnologías de análisis de lenguaje natural para comparar el contenido Seguimiento de medios básico, relacionando noticias entre sí	UNIQUE VALUE PROPOSITION Visor para lectura de prensa digital de distintos medios con sugerencias de noticias a través del procesamiento de lenguaje natural	UNFAIR ADVANTAGE Uso de procesamiento de lenguaje natural para comparar noticias entre sí	CUSTOMER SEGMENTS Lectores de prensa digital Periodistas Instituciones con necesidades de recuperación de información
EXISTING ALTERNATIVES Agregadores de contenido que permiten la lectura de noticias Para contrastar la información hay que acceder a cada medio por separado El seguimiento de medios se hace a través de servicios de pago	KEY METRICS Búsquedas realizadas Accesos a la aplicación	HIGH-LEVEL CONCEPT Comparador de opinión en prensa	CHANNELS Redes sociales Product Hunt Boca a boca Instituciones universitarias	EARLY ADOPTERS Lectores de prensa digital
COST STRUCTURE Alojamiento Difusión Tiempo de desarrollo			REVENUE STREAMS Opción Premium con adición de fuentes personalizadas, posibilidad de realizar recortes, subrayados, etc Venta de aplicación para uso privado en instituciones	

Figura 4.1.: Lean Canvas de Contrast

5. Metodología

El proyecto ha seguido un modelo de desarrollo incremental mediante dos hitos compuestos por iteraciones cada dos semanas. Al finalizar cada iteración se ha realizado una revisión con los tutores del proyecto con el fin de analizar el avance, responder dudas surgidas durante el sprint y planificar el siguiente. Al finalizar el primer hito, con un desarrollo funcional de la mayoría de características, se empezó el segundo desde el principio evitando los errores realizados en el primer hito y mejorando la calidad del código. El trabajo realizado en cada hito e iteración se detalla en el siguiente apartado.

5.1. Hitos e iteraciones

Este apartado trata la duración y tareas propuestas en cada iteración. Se especifican las tareas a grandes rasgos, sin entrar en detalle.

5.1.1. Hito 1

El primer hito ha consistido en realizar una primera implementación del proyecto, probando las diferentes propuestas tecnológicas y analizando las interacciones e integraciones entre ellas.

Iteración I

La primera iteración ha analizado el problema a tratar así como las tecnologías para abordarlo y qué se quiere mostrar. Concretamente las tareas han sido:

- Establecer los objetivos.
- Estudiar el marco teórico y documentarlo.
- Establecer la metodología de trabajo.
- Describir los requisitos funcionales.

Iteración II

La segunda iteración ha consistido en especificar ciertas bases del proyecto y construir una implementación inicial de algunas partes independientes probando diferentes tecnologías. Las tareas han sido:

- Prototipar las interfaces.

- Probar y escoger las tecnologías para el desarrollo y documentarlas.
- Construir el analizador RSS.
- Definir los casos de uso de la aplicación.

Iteración III

La tercera iteración ha pretendido conseguir una API funcional mínima que interactuase con el motor de base de datos. Las tareas han sido:

- Definir el modelo de negocio.
- Construir la API con CRUD mínimo.
- Definir el esquema de datos provisional e integrar el motor de búsqueda.

Iteración IV

En la cuarta iteración se ha integrado un framework de front-end que muestre los resultados y la herramienta de procesamiento de lenguaje natural. El objetivo era llegar a una primera versión semi-funcional.

- Construir el front-end mínimo y funcional e integrarlo con la API para mostrar los resultados.
- Documentar la implementación inicial del proyecto.
- Integrar la herramienta de procesamiento de lenguaje natural.

5.1.2. Hito 2

El segundo hito ha supuesto una extensión del primero, ampliando y mejorando las características ya implementadas en funcionalidades finales que usaría la aplicación en su primera versión.

Iteración V

- Estudiar e integrar las tecnologías CI/CD (Integración continua/Distribución continua) y contenedores.
- Definir y documentar la arquitectura final seleccionada.
- Definir y documentar el diseño final de la aplicación.

Iteración VI

- Detallar el esquema de datos final y optimizar la indexación.
- Crear los endpoints finales de la API, integrando facetados y aspectos del motor de búsqueda.

Iteración VII

- Optimizar el uso de la herramienta de NLP para disminuir el coste computacional.
- Extender el front-end para mostrar el diseño final.

5.2. Herramientas de administración del proyecto

A continuación se detallan las herramientas que se han empleado para la organización del proyecto:

- **Workona**¹: extensión del navegador Chrome que facilita la gestión de las pestañas. Su principal funcionalidad es crear espacios de trabajo en el que se guardan las pestañas actualmente abiertas de una sesión a otra. Esto permite cambiar el dispositivo de trabajo fácilmente sin perder el contenido que se estaba visitando en ese momento. Al mismo tiempo también permite cambiar el proyecto en el que se esté trabajando con un solo clic.

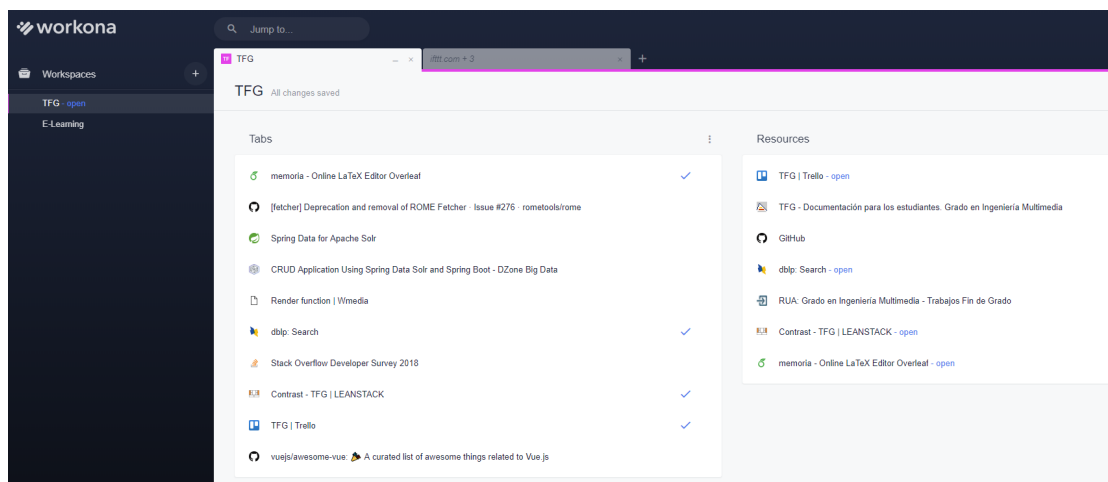


Figura 5.1.: Espacio de trabajo para Contrast

- **Trello**²: aplicación web que posibilita la creación de tableros con listas de tareas. Dichas tareas pueden ser etiquetadas y comentadas y se les pueden añadir archivos adjuntos. Los tableros permiten el flujo libre de tareas entre una lista y otra. En el caso del proyecto, se ha utilizado una metodología de gestión de trabajo Kanban, por lo tanto se han creado tres listas: por hacer, en proceso y finalizado. Se han añadido dos listas adicionales: una para acumular las tareas al finalizar una iteración y otra para definir tareas que surgen durante el proceso de desarrollo y así asignarlas en una iteración próxima.

¹<https://workona.com/>

²<https://trello.com/>

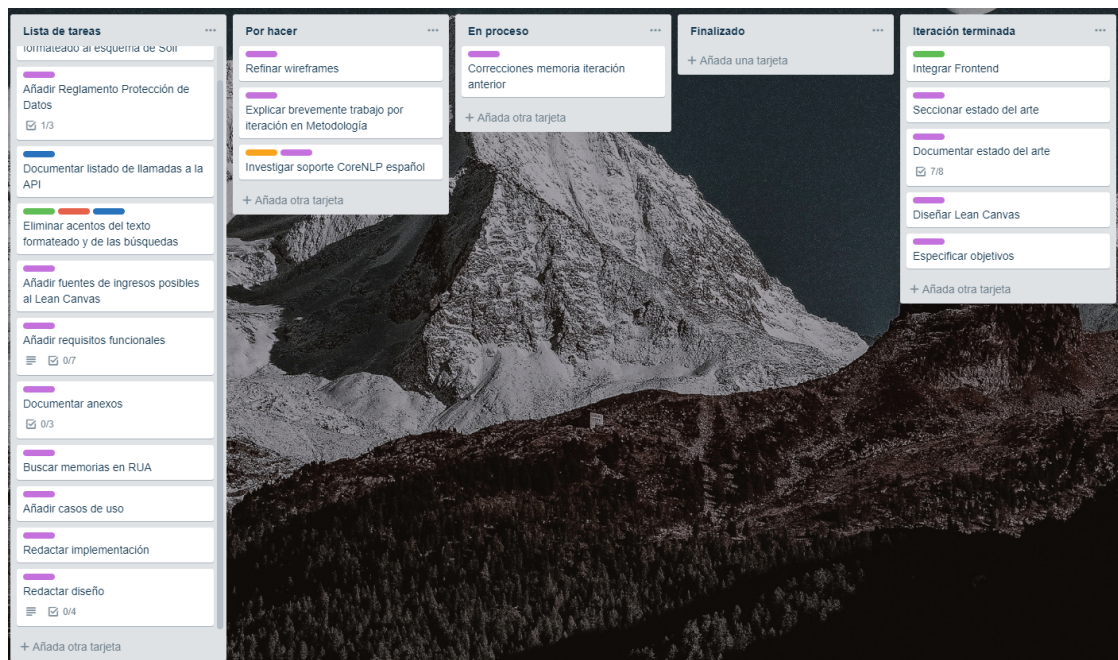


Figura 5.2.: Tablero para Contrast

- **Todoist**³: aplicación web que facilita la organización de tareas en el día a día. En el caso del proyecto es utilizada para organizar las tareas personales junto a las del trabajo de final de grado. Esta aplicación es utilizada junto a **IFTT**⁴ (If This Then That), que posibilita la integración de Todoist y Trello, de tal manera que cada vez que se cree una tarjeta en Trello, se añade una tarea en Todoist. La aplicación es utilizada bajo la filosofía GTD⁵ (Getting Things Done).

5.3. Herramientas de desarrollo

En esta sección se especifican las herramientas utilizadas para el desarrollo del proyecto:

- **Visual Studio Code**⁶: editor de código fuente multiplataforma, open-source que incluye muchas funcionalidades en forma de extensiones para trabajar con todos y cada uno de los lenguajes existentes. Su característica principal es el IntelliSense, que autocompleta las instrucciones mientras se escribe y destaca elementos con los que se interactúan.

³<https://todoist.com/>

⁴<https://ifttt.com/>

⁵https://es.wikipedia.org/wiki/Getting_Things_Done

⁶<https://code.visualstudio.com/>



Figura 5.3.: Integración IFTT de Todoist con Trello

- **GitKraken**⁷: cliente de Git⁸ con una interfaz visual que automatiza ciertos procesos internos de las instrucciones y permite un uso intuitivo de Git. Destaca por el uso de la técnica de *arrastrar y soltar* para interactuar con los *commits*.
- **Postman**⁹: herramienta que facilita el desarrollo, prueba y validación de elementos en una API. Nació como una extensión de Chrome que evolucionó en multiplataforma; hoy en día se ha convertido en un *must-have* del desarrollo web.
- **Overleaf**¹⁰: servicio de LaTeX¹¹ colaborativo en línea con el que redactar y publicar documentos académicos de una manera más rápida, visualizando el resultado tras cada modificación automáticamente.

⁷<https://www.gitkraken.com/>

⁸<https://git-scm.com/>

⁹<https://www.getpostman.com/>

¹⁰<https://www.overleaf.com/>

¹¹<https://www.latex-project.org/>

6. Análisis y especificación

6.1. Especificación requisitos funcionales

De acuerdo con el estándar IEEE 830, los requisitos funcionales del proyecto han sido identificados y se encuentran recogidos en el Anexo B. A continuación se detalla una lista resumida de ellos:

- **RF01: Listar noticias.** La aplicación mostrará un listado de noticias recientes.
- **RF02: Buscar noticias.** La aplicación obtendrá un listado de noticias según los parámetros introducidos por el usuario en el campo de búsqueda.
- **RF03: Filtrar noticias por periódico.** La aplicación obtendrá un listado de noticias con los periódicos seleccionados.
- **RF04: Filtrar noticias por rango de fechas.** La aplicación obtendrá un listado de noticias según el rango de fechas definido por el usuario.
- **RF05: Filtrar noticias por entidades.** La aplicación obtendrá un listado de noticias según las entidades seleccionadas por el usuario.
- **RF06: Ampliar noticia.** La aplicación mostrará un botón en cada noticia para ampliar su vista y facilitar su lectura.
- **RF07: Buscar noticias similares.** La aplicación buscará noticias similares a la seleccionada.
- **RF08: Indexar noticias.** La aplicación indexará las noticias que obtenga del analizador.
- **RF09: Mantener formato.** La aplicación mantendrá el formato original de las noticias en su visualización.
- **RF10: Obtener noticias.** La aplicación obtendrá noticias de distintos medios.
- **RF11: Facetar por contenido.** La aplicación realizará distintos facetados que permitan complementar la información mostrada por los filtrados.
- **RF12: Pagarinar noticias.** La aplicación mostrará un conjunto de noticias por página y permitirá avanzar y retroceder entre ellas.

6.2. Casos de uso

La figura 6.1 describe los casos de uso diseñados a partir de los requisitos funcionales descritos en el Anexo B. Como se puede observar, el usuario puede listar noticias, buscar en ellas, filtrarlas y ampliar una noticia determinada. Esta última acción implica listar otras noticias que tengan semejanza con la seleccionada.

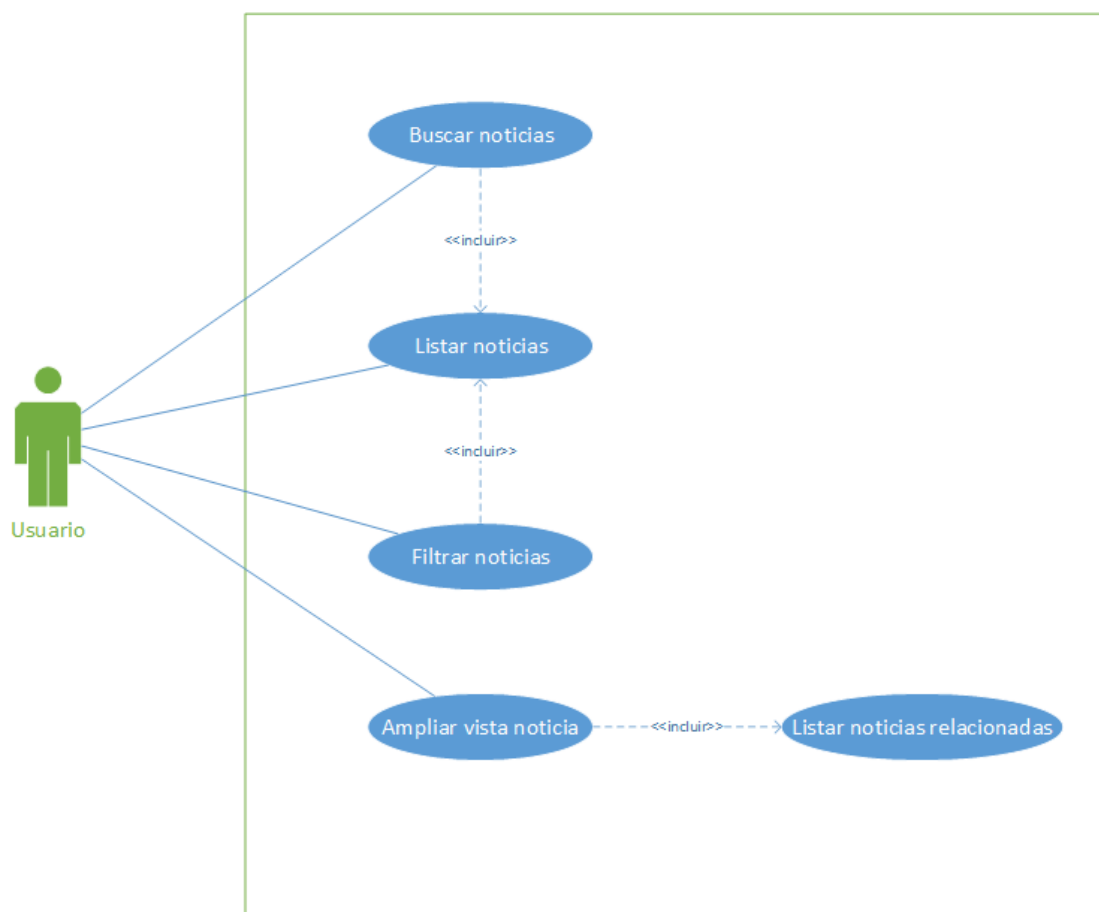


Figura 6.1.: Casos de uso

6.3. Diagrama de actividad

A continuación se describe el flujo de control entre el proceso de ampliar la vista de una noticia (figura 6.2) y el de filtrar noticias (figura 6.3).

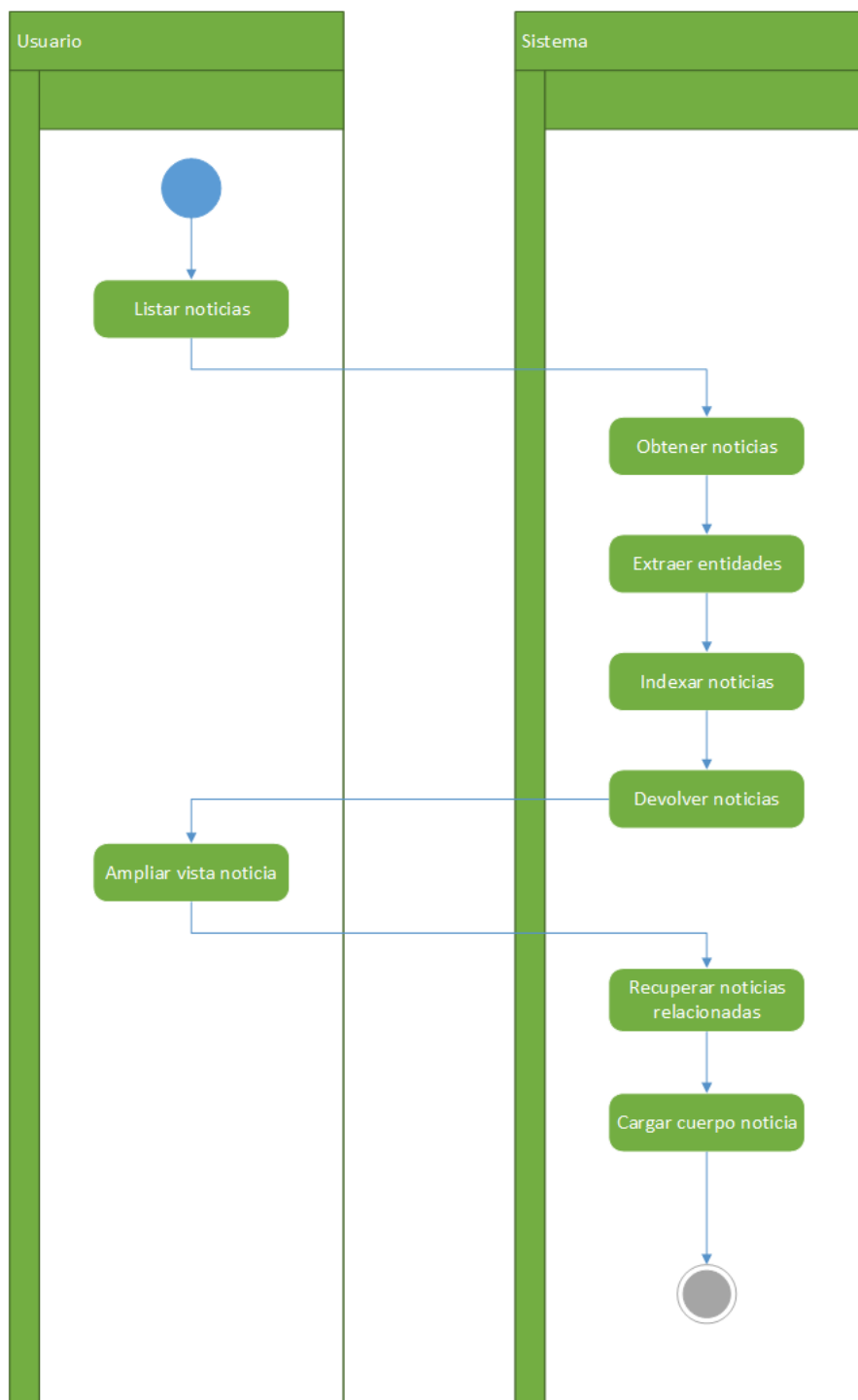


Figura 6.2.: Diagrama de actividad de la función ampliar noticia

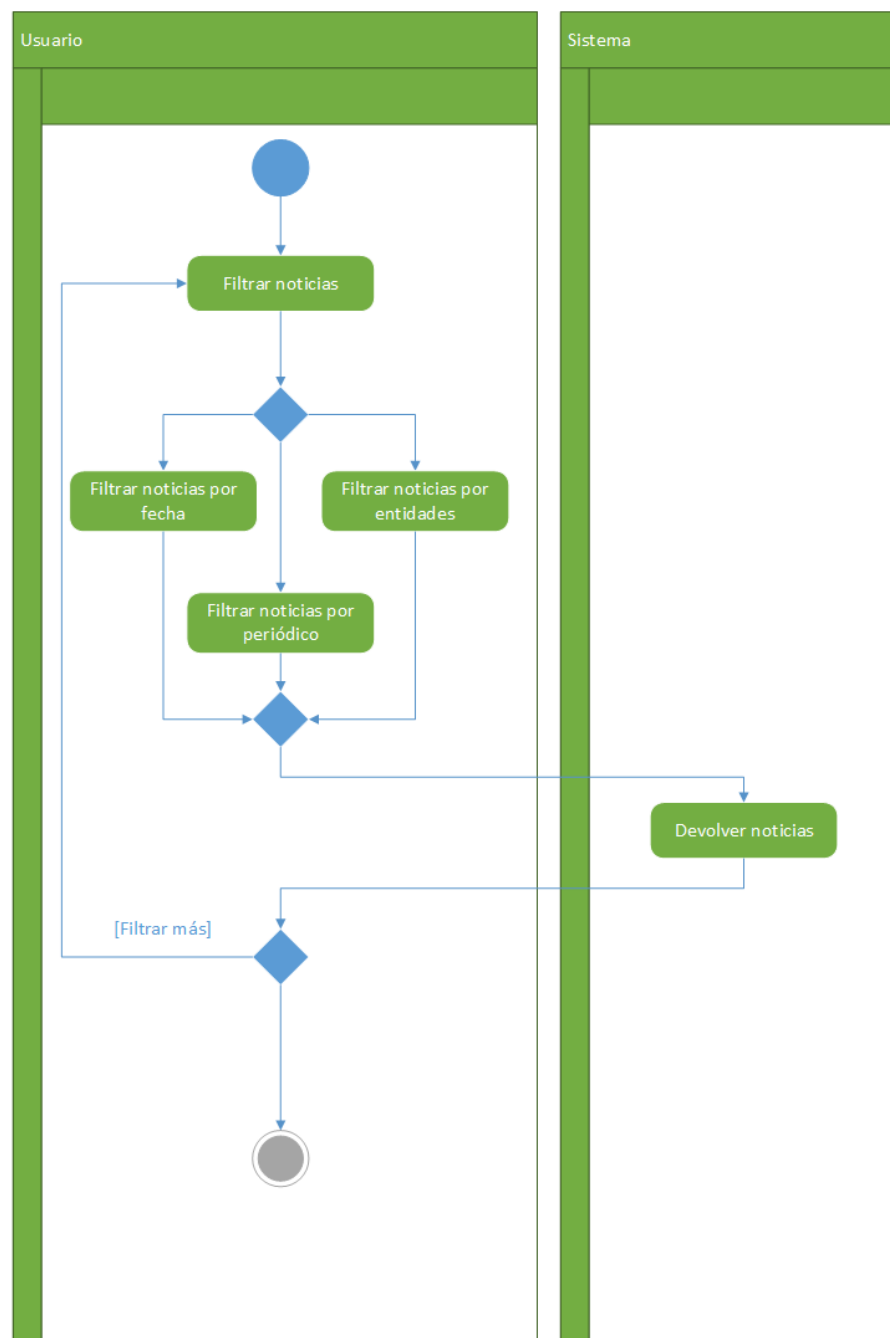


Figura 6.3.: Diagrama de actividad de la función filtrar

7. Diseño

7.1. Arquitectura seleccionada

En la figura 7.1 se detalla el conjunto de tecnologías seleccionadas para el desarrollo del proyecto. Cabe destacar que algunas se han omitido, ya que sirven de soporte a otras que sí que se han detallado.

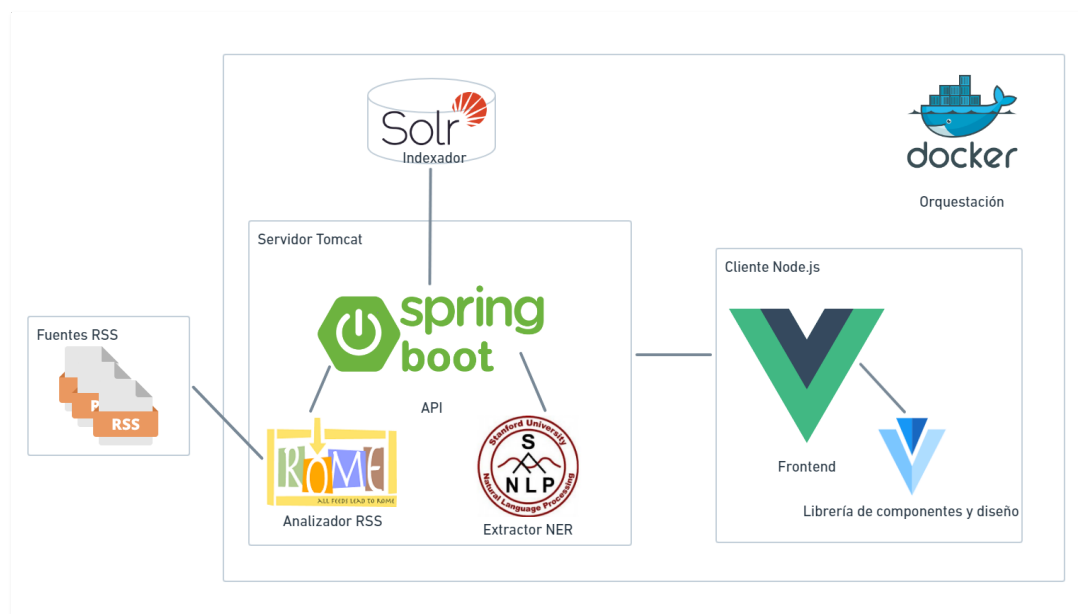


Figura 7.1.: Stack tecnológico utilizado por Contrast

Como se puede observar en la figura 7.1, la aplicación toma su contenido de fuentes externas a la aplicación. Se describen dos contextos. El del back-end, un servidor Tomcat, contiene la API implementada con Spring Boot y las dependencias de Rome Tools y Stanford CoreNLP. El contexto del front-end, un servidor Node.js, contiene a la capa de visualización implementada con Vue.js y la librería de componentes Vuetify. Por otro lado se cuenta con Solr como indexador y con Docker que da soporte a la ejecución de Solr, el back-end y el front-end, ya que son partes independientes contenerizadas.

7.2. Tecnologías

El siguiente apartado justifica las tecnologías escogidas entre las analizadas en la sección 3, Marco Teórico. Por una parte, se ha optado por utilizar Java y por lo tanto Spring Boot, en el back-end por la gran compatibilidad que ofrece con el resto de tecnologías que se tenían que integrar con él, por su madurez y por su extensa documentación. Esta decisión también se ha visto condicionada por:

- Tener una capa de abstracción para tratar directamente con el motor de elección de base de datos.
- Tener la posibilidad de utilizar CoreNLP mediante una dependencia del proyecto, simplificando su uso e integración.

Dado que se ha escogido Java, la elección de Rome como analizador era muy clara. Esto es debido a que se integra del mismo modo como una dependencia más del proyecto, con el añadido de ser además la herramienta más potente de las tres analizadas.

Entre los motores de búsqueda la elección ha sido más difusa, de hecho se ha realizado el desarrollo prácticamente paralelamente sobre ambas opciones. Esto ha sido posible gracias a que la capa de abstracción de Spring Boot sobre Apache Solr y Elasticsearch, permitía con modificaciones mínimas cambiar entre un motor u otro. Finalmente se ha escogido Solr dado que posibilitaba crear previamente el esquema de datos mediante una sencilla declaración de parámetros en un XML.

Como herramienta de procesamiento de lenguaje natural, la balanza se ha inclinado hacia CoreNLP por las razones que se detallan a continuación:

- Se ha integrado con el back-end escogido como una dependencia más, simplificando su uso.
- Posee una mayor documentación que el resto analizado, facilitando el entendimiento del uso de los extractores.
- Es la herramienta más utilizada de las tres analizadas en este ámbito.

Por último, el front-end se ha construido con Vue.js. Dado que la aplicación requiere de pocas interfaces y pocos elementos en esta, este framework posibilita un rápido desarrollo con componentes predefinidos sobre los que tan solo había que enlazar los datos. Encima de este se ha utilizado Vuetify como librería de diseño y componentes, que al ser nativa de Vue, ha permitido embellecer el resultado sin realizar excesivas modificaciones.

7.3. Diagrama de componentes

La figura 7.2 describe el diagrama de componentes a alto nivel del sistema empleado por el proyecto.

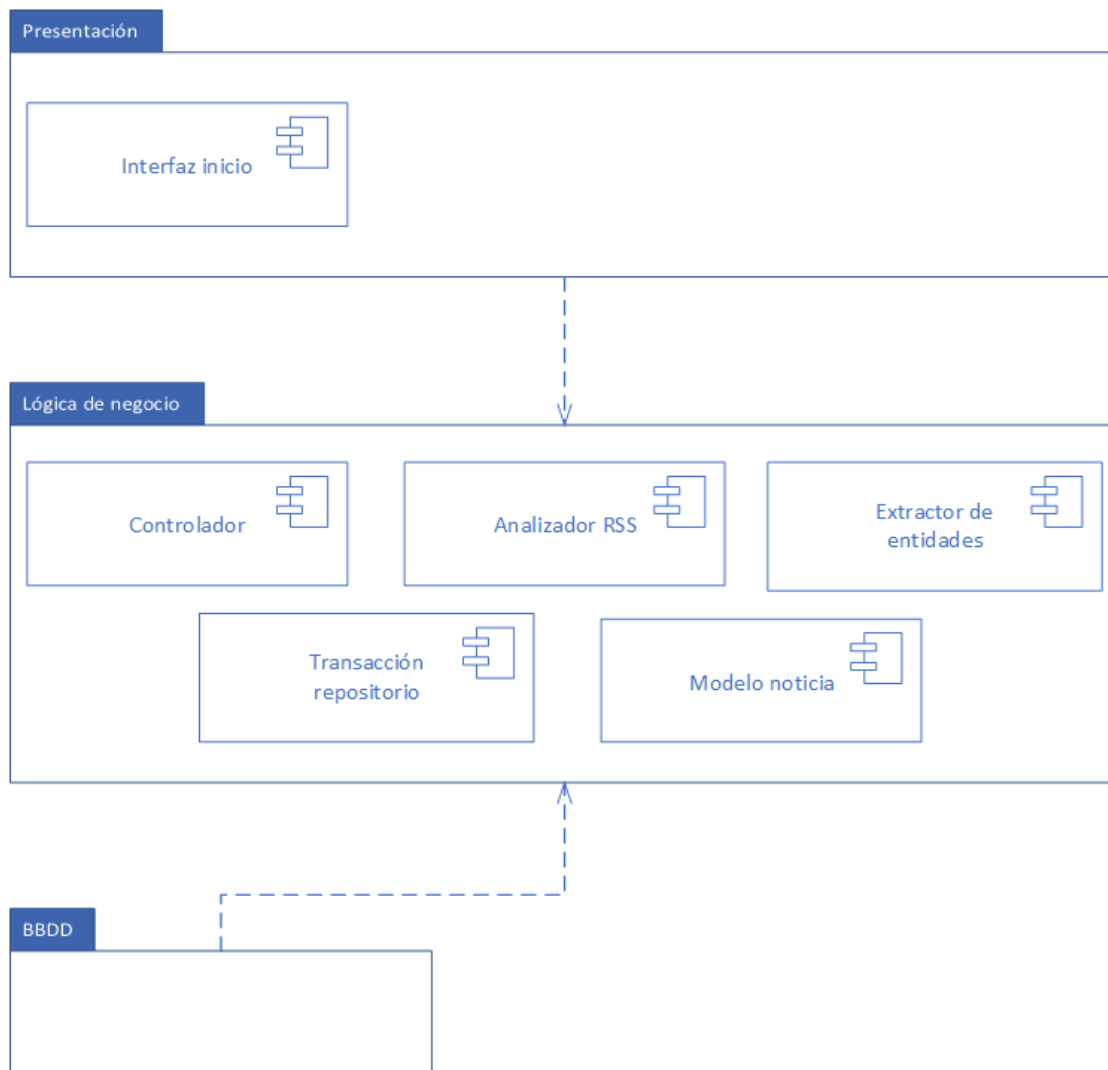


Figura 7.2.: Diagrama de componentes

7.4. Mockups

En la siguiente sección se detallan los mockups que se realizaron para ambos Hitos. El primero plasma un esbozo de la idea original, sin entrar en detalles de su posterior implementación y el segundo, en cambio, muestra la idea refinada con un aspecto similar al de su implementación final.

7.4.1. Hito 1

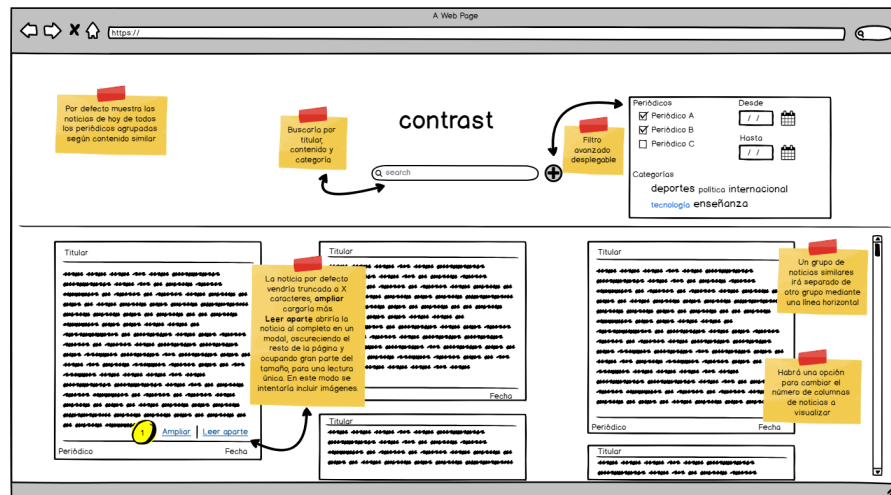


Figura 7.3.: Prototipo de interfaz para página de inicio y resultado búsqueda

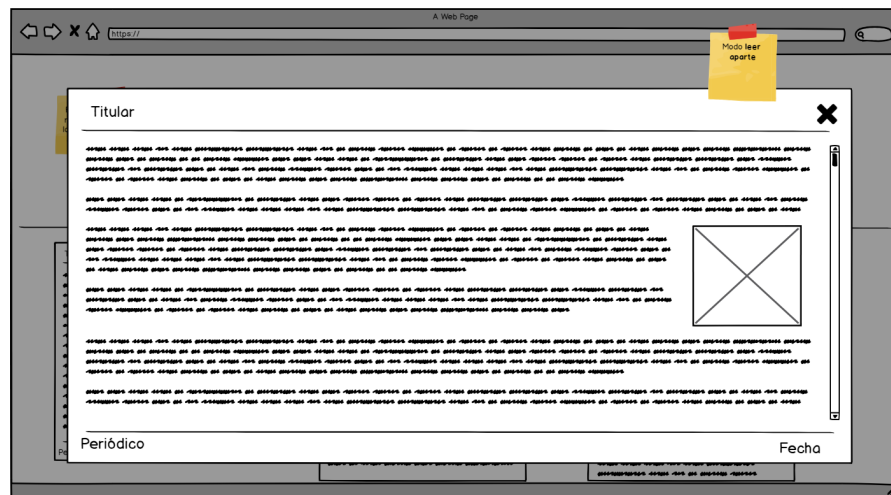



Figura 7.4.: Prototipo de interfaz para noticia ampliada

7.4.2. Hito 2

contrast



Titular

Periódico
Fecha

Lorem ipsum dolor sit amet,
consectetur adipiscing elit. Quisque
blandit felis non justo volutpat congue.
Nullam cursus nulla id ex imperdiet
malesuada. Phasellus eu enim
consequat, ornare justo id...

Leer más

Titular

Periódico
Fecha

Lorem ipsum dolor sit amet,
consectetur adipiscing elit. Quisque
blandit felis non justo volutpat congue.
Nullam cursus nulla id ex imperdiet
malesuada. Phasellus eu enim
consequat, ornare justo id...

Leer más

Periódico

☐ Periódico 1 (X)

☒ Periódico 2 (X)

☒ Periódico 3 (X)

Fecha

DD/MM/YYYY

DD/MM/YYYY

Etiquetas seleccionables

[Etq \(X\)](#) [Etq \(X\)](#)

[Etq \(X\)](#) [Etq \(X\)](#)

X representa el facetado por ese parámetro

Figura 7.5.: Diseño final de interfaz para página de inicio y resultado búsqueda

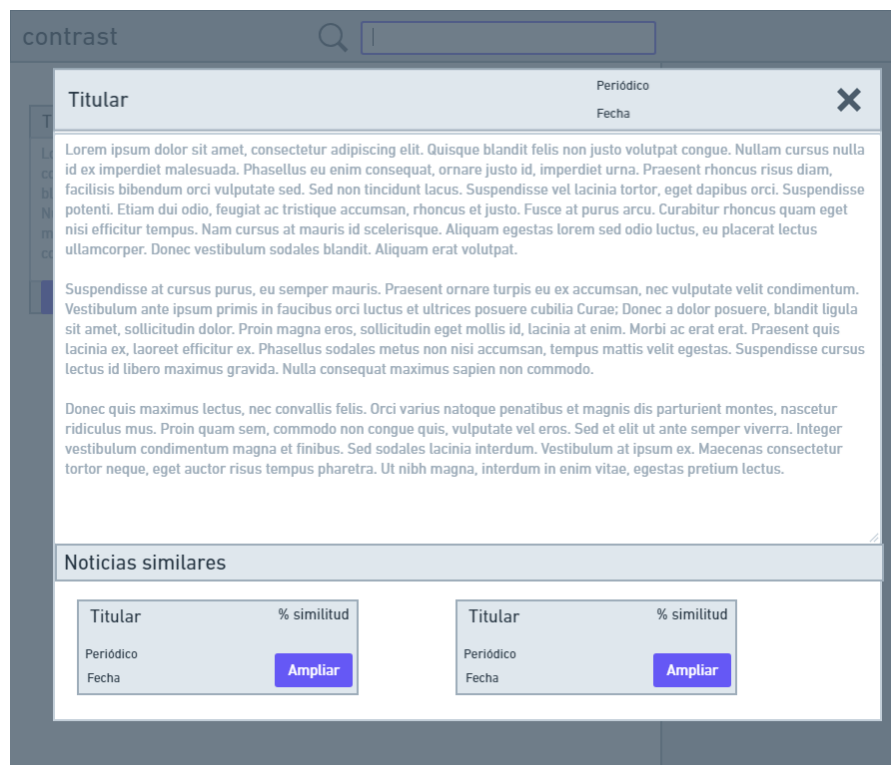


Figura 7.6.: Diseño final de interfaz para noticia ampliada

8. Implementación

El siguiente capítulo detalla el avance conseguido en cada iteración, la problemática encontrada y qué pasos se han seguido para resolverla.

8.1. Hito 1

8.1.1. Iteración I

En primer lugar se elaboraron unos objetivos tangibles y alcanzables que cumplieran las necesidades del proyecto y los intereses personales. Acto seguido se estableció una metodología de trabajo basada en la recopilación de documentación detallada en el capítulo 5.

El estudio del marco teórico en el que se engloba el proyecto ha sido realizado en distintas etapas. La primera de ellas ha consistido en la recolección de información acerca de qué productos se encuentran en el mismo nicho de mercado y cumplen funciones similares. Para ello se ha realizado el registro en cada una de las plataformas y se ha utilizado con el mismo conjunto de fuentes durante un par de días. Este proceso ha resultado muy interesante ya que no solo sirvió para determinar qué se quería mostrar, sino que también se utilizó para observar cómo mostrar esa información. Algunos servicios no han podido utilizarse debido a que eran de pago, en este caso se observaron notas de prensa y contenido multimedia relacionado con el servicio.

El siguiente paso ha sido la investigación del marco legal en el que se engloba el proyecto. Este es un aspecto importante dado que Contrast se vale de contenido protegido de terceros. Se realizó una búsqueda de licencias y avisos legales para la reproducción del contenido en sitios de terceros y se además se documentaron sucesos relevantes al respecto, como el conflicto AEDE.

El paso posterior ha consistido en el estudio de las tecnologías con las que se ha desarrollado el proyecto. Para ello se plantearon diversas propuestas en cada uno de los apartados, según la investigación en plataformas técnicas especializadas.

Tras esto, se desgranaron los objetivos en diferentes requisitos funcionales que la aplicación debería satisfacer y se completaron con observaciones extraídas de las plataformas analizadas.

8.1.2. Iteración II

La segunda iteración comenzó describiendo los casos de uso. Tras finalizar el paso anterior, se procedió a prototipar la primera versión de las interfaces a un bajo nivel, que incluiría todos los requisitos funcionales que se pudieran mostrar visualmente junto a

elementos de diseño observados en los servicios analizados en el 3. Para ello se realizaron dos prototipos sencillos con las dos vistas principales que albergaría la aplicación. Por un lado se creó la visualización principal o por defecto, con dos áreas significativas claramente diferenciadas, una extensión para el listado de noticias y otra para la búsqueda y filtrado de estas. En la segunda visualización se mostró el resultado de ampliar una noticia para su lectura extendida. Cabe destacar que este primer prototipado se utilizó para volcar las ideas que se tenían al respecto en la implementación del proyecto, sin tener en cuenta elementos de diseño específicos. Es por ello que se utilizaron múltiples anotaciones textuales de varios elementos para clarificar su utilidad final.

El paso siguiente consistió en probar las diferentes tecnologías propuestas para aumentar la documentación en la memoria y así poder tomar una decisión respecto a ellas en su uso para la implementación del proyecto. Dado que los lenguajes de programación ya eran conocidos y habían sido utilizados en otros proyectos, no se destinó a tiempo a tratar con ellos específicamente ya que en el uso de las tecnologías seleccionadas basadas en ellos habría tiempo suficiente para decantarse por uno o por otro. Es por ello que se abordó en primer lugar las tecnologías que darían soporte al back-end y por ende se empezó a testear las diferentes librerías de recuperación y parseo de contenido RSS, con ejemplos mínimos proporcionados por la propia documentación de la tecnología pero que satisfacen los intereses del proyecto.

En el extracto de código 8.1 se detalla el proceso de obtención de contenido de un feed desde una URL en la librería Universal Feed Parser de Python.

Listado 8.1: Universal Feed Parser

```
1 import feedparser
2 d = feedparser.parse('URL-DEL-FEED')
```

El código incluido en 8.2 contiene el proceso que utiliza la librería Rome Tools de Java para obtener contenido de un feed en una URL.

Listado 8.2: Rome Tools

```
1 import java.net.URL;
2 import java.io.InputStreamReader;
3 import com.rometools.rome.feed.synd.SyndFeed;
4 import com.rometools.rome.io.SyndFeedInput;
5 import com.rometools.rome.io.XmlReader;
6
7 public class FeedReader {
8
9     public static void main(String[] args) {
10         if (args.length==1) {
11             try {
12                 URL feedUrl = new URL("URL-DEL-FEED");
13
14                 SyndFeedInput input = new SyndFeedInput();
```

```

15         SyndFeed feed = input.build(new XmlReader(feedUrl)
16             );
17     }
18     catch (Exception ex) {
19         ex.printStackTrace();
20         System.out.println("ERROR: "+ex.getMessage());
21     }
22 }
23 }

```

La pieza de código en 8.3 describe los pasos necesarios para recuperar contenido de un feed en una URL con la librería rss-parser de Javascript.

Listado 8.3: rss-parser

```

1 let Parser = require('rss-parser');
2 let parser = new Parser();
3
4 (async () => {
5     let feed = await parser.parseURL('URL-DEL-FEED');
6 })();

```

Tal y como se puede observar, las propuestas basadas en Python y Javascript son más sencillas y legibles dado a la falta de tipado ya que son lenguajes orientados a una mayor velocidad y flujo. Por otro lado la propuesta con Java es autoexplicativa debido a justo lo contrario que las anteriores; a pesar de ser más extensa, se observa con mejor definición el proceso y su control.

Tras esta prueba de concepto, se pasó a investigar las propuestas de los motores de búsqueda. Splunk fue descartado debido a su alta orientación a Big Data, así que la dualidad estaba entre Solr y Elasticsearch. Dado que ambos están contruidos sobre Apache Lucene comparten muchos aspectos entre ellos, especialmente en cuanto a funcionalidad, pero teniendo en cuenta que la experiencia en estos motores es inexistente, se optó por empezar el desarrollo con aquel que fuese más intuitivo de utilizar. En este aspecto Solr sufrió una discriminación positiva ya que cuenta con una GUI que permite realizar muchas operaciones básicas. En el caso de Elasticsearch, se puede obtener una GUI a través de front-ends desarrollados por terceros, sin soporte oficial.

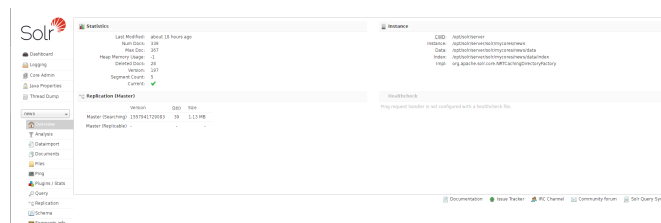


Figura 8.1.: Interfaz de usuario gráfica de Apache Solr

Elasticsearch también ha sido instalado para que, una vez definido el esquema de datos, realizar pruebas de indexación y recuperación sobre ambos motores y así poder obtener una experiencia real del proceso que se requiere para el proyecto.

Por otra parte, se probaron las herramientas de procesamiento de lenguaje natural seleccionadas. Tanto CoreNLP como Freeling, proporcionan una demo web en la que puedes insertar texto y extraer características de él, aunque las opciones disponibles no permiten personalización. En ambas se introdujo la misma frase: 'Esto es una prueba de extracción de entidades para el TFG de Ingeniería Multimedia de la Universidad de Alicante de Adrián Tomás.'

Stanford CoreNLP 3.9.2 (updated 2018-11-29)

— Text to annotate —
 Esto es una prueba de extracción de entidades para el TFG de Ingeniería Multimedia de la Universidad de Alicante de Adrián Tomás.

— Annotations —
☒ named entities ☒ sentiment

— Language —
 Spanish

Submit

Named Entity Recognition:

1 Esto es una prueba de extracción de entidades para el TFG de Ingeniería Multimedia de la Universidad de Alicante de Adrián Tomás.

Sentiment:

1 Esto es una prueba de extracción de entidades para el TFG de Ingeniería Multimedia de la Universidad de Alicante de Adrián Tomás.

Figura 8.2.: Demostración de extracción de entidades en CoreNLP

Sentence 1

Esto	es	una	prueba	de	extracción	de	entidades	para	el	TFG de Ingeniería Multimedia de la Universidad de Alicante de Adrián Tomás	.
esto	ser	uno	prueba	de	extracción	de	entidad	para	el	tfg de ingeniería multimedia de la universidad de alicante de adrián tomas	Fp
P000000	V5IP350	D10F50	NCF5000	SP	NCF5000	SP	NCF5000	DA0MS0	NP000000		

▼ CONLL format

1	Esto						esto			P000000	PD	pos=pronoun type=demonstrative num=singular	-
2	es						ser			V5IP350	V5I	pos=verb type=sentiauxiliary mod=indicative tense=present person=3 num=singular	-
3	una						uno			D10F50	O1	pos=determiner type=indefinite gen=feminine num=singular	-
4	prueba						prueba			NCF5000	NC	pos=noun type=common gen=feminine num=singular	-
5	de						de			SP	SP	pos=adposition type=preposition	-
6	extracción						de			NCF5000	NC	pos=noun type=common gen=feminine num=singular	-
7	de						de			SP	SP	pos=adposition type=preposition	-
8	entidades						entidad			NCF5000	NC	pos=noun type=common gen=feminine num=plural	-
9	para						para			SP	SP	pos=adposition type=preposition	-
10	el						el			DA0MS0	DA	pos=determiner type=article gen=masculine num=singular	-
11	TFG de Ingeniería Multimedia de la Universidad de Alicante de Adrián Tomás						tfg de ingeniería multimedia de la universidad de alicante de adrián tomas			NP000000	NP	pos=noun type=proper ne=class=organization	B-ORG
12	.						.			Fp	Fp	pos=punctuation type=period	-

Figura 8.3.: Demostración de extracción de entidades en Freeling

En la extracción de entidades se puede constatar que Freeling considera todo el contenido a partir de la palabra 'TFG' como una sola entidad, de la cual sería muy difícil extraer información relevante y que, las entidades obtenidas por CoreNLP, se acercan bastante a las esperadas dado que reconoce los distintos elementos clave de la frase. Los resultados obtenidos no son exactos ni finales ya que la versión de demostración no cuenta con todas las características que ofrece la herramienta al completo. Particularmente, CoreNLP expone las entidades de tipo *ORGANIZATION* y *MISC* en la frase de demostración que, pese a que no son del tipo esperado, separa las entidades entre sí de manera adecuada. En los casos de Freeling y OpenNLP se halla además que la documentación tiende a ser escueta y que su presencia en los foros especializados es menor. Con todo ello se escoge CoreNLP sobre las demás propuestas.

Por último para terminar con la prueba de las tecnologías seleccionadas para el back-

end, se repasan los frameworks sobre los que se desarrollaría la API. Inicialmente en este apartado no se le da excesiva importancia a la elección del framework en sí, dado que la implementación planteada es sencilla. Es por ello que se decide priorizar la integración del framework con las tecnologías descritas anteriormente para facilitar el desarrollo del proyecto. En este sentido Spring Boot contiene una capa de abstracción para ambos motores de búsqueda propuestos y además tanto la herramienta de parseo de RSS (Rome Tools), como la de extracción de entidades (CoreNLP) se integran en el proyecto como una dependencia más en el archivo pom.xml.

Si se quisiera utilizar Express.js, habría que realizar las llamadas a la API del motor de búsqueda manualmente, el analizador de RSS se integraría fácilmente y habría que realizar llamadas a la API proporcionada por CoreNLP, la cual además habría que ejecutarla en un servidor.

Django en este sentido pecaría de la misma situación que Express.js, pues son frameworks ligeros pensados para escalarlos a medida y que por ello, las integraciones son algo más costosas aunque más personalizables.

Una vez expuestos los razonamientos de cada una de las tecnologías probadas anteriormente, se decide utilizar Spring Boot, Rome y CoreNLP en la parte del back-end.

La etapa siguiente era la prueba y decisión de tecnologías para el desarrollo del front-end. Las propuestas están construidas en Javascript y con unos principios muy similares, ya que comparten una gran cantidad de características y en muchos aspectos la transición de uno a otro tiene una curva de aprendizaje ligera. Los tres son frameworks jóvenes pero altamente utilizados por la comunidad, poseen extensa documentación y están en constante evolución, es por ello que la decisión en este campo ha consistido más en una preferencia personal.

Dado que Vue.js es el más nuevo entre ellos, fue construido basándose en las otras dos opciones y es la más sencilla de las propuestas en cuanto a implementación, se ha decidido utilizarlo como framework para el front-end.

En cuanto a las librerías de diseño y componentes, tanto Bootstrap como Vuetify ofrecen integraciones sencillas en la arquitectura del proyecto, teniendo la segunda opción una mayor definición semántica ya que parte de los componentes naturales de Vue.js. En el caso de Foundation, su integración es más compleja ya que requiere componentes de terceros para apoyar su funcionalidad al completo.

Listado 8.4: Declaración de un botón en Vuetify

```
1 <v-btn color="success">Success</v-btn>
```

Listado 8.5: Declaración de un botón en Bootstrap

```
1 <button type="button" class="btn btn-primary">Primary</button>
```

Es por ello que se decide utilizar Vuetify como librería de componentes y diseño ya que facilita la nomenclatura y coherencia semántica del front-end.

Una vez escogido el stack de tecnologías con las que se va a desarrollar el proyecto, se procede a construir el analizador RSS con Rome partiendo del ejemplo anterior.

Por un lado se crea un archivo que almacene por línea los enlaces de las fuentes de las cuales se va a recuperar contenido RSS. Posteriormente, utilizando el ejemplo proporcionado por la documentación de Rome y mostrado anteriormente, se construye el analizador imprimiendo en pantalla los resultados de la recuperación y parseo de los RSS. Este es un paso importante, pues ha servido para observar qué campos van a conformar el esquema de datos.

8.1.3. Iteración III

La tercera iteración ha empezado analizando los resultados del analizador construido en la iteración anterior. El objeto de análisis ha sido determinar qué campos de datos van a ser útiles para el desarrollo de la aplicación. Tal y como estaba especificado en los requisitos funcionales, cada noticia debe poseer al menos el titular, la fecha de publicación y el periódico al que pertenece para poder mostrarse. Adicionalmente se define un campo de datos más, las entidades que se reconozcan en la noticia. Los campos descritos anteriormente definen el esquema mínimo de datos para la aceptación de una noticia. Rome proporciona estos datos a través de las siguientes llamadas:

Listado 8.6: Funciones de Rome para la consecución del esquema mínimo de datos

```
1 SyndFeed feed = new SyndFeedInput().build(new XmlReader(stream));
2 String newspaper = feed.getTitle();
3 for (SyndEntry entry : feed.getEntries()) {
4     String headline = entry.getTitle()
5     Set<String> entities = new HashSet<String>();
6     for (SyndCategory cat : entry.getCategories()) {
7         entities.add(cat.getName());
8     }
9     LocalDate publishedDate = entry.getPublishedDate().toInstant()
10        .atZone(ZoneId.systemDefault()).toLocalDate();
11 }
```

Para la conformación del esquema de datos óptimo, es decir, aquel en el que el funcionamiento del objeto en la aplicación sea el idóneo, se necesitaría almacenar dos campos adicionales. Estos son el enlace a la noticia y el cuerpo de la noticia. Se añaden como datos adicionales dado que no todas las fuentes que se probaron proporcionan estos datos y porque son prescindibles para el funcionamiento general pensado de la aplicación.

Listado 8.7: Funciones de Rome para la extracción de los campos adicionales para el esquema de datos óptimo

```
1 entry.getLink();
2 entry.getDescription().getValue();
```

La función para la obtención del cuerpo de las noticias devuelve además todas las etiquetas HTML que contiene el cuerpo original de la noticia, esto es el formato de visualización con el que fue creado, además de los elementos multimedia. Es por ello que debido al RF09 de mantenimiento de formato en el cuerpo de las noticias, se decide utilizar dos campos para el almacenamiento de la descripción de las noticias. El primero almacena el valor obtenido por la función; el segundo almacena el valor obtenido tras eliminar todas las etiquetas HTML. Esto es realizado a través de una expresión regular:

Listado 8.8: Eliminación de etiquetas HTML del cuerpo de la noticia

```
1 entry.getDescription().getValue().replaceAll("<[^>]*>", "")
```

De esta manera, se utiliza el campo sin etiquetas para la recuperación de información y el original para la visualización únicamente.

Una vez concretados los esquemas de datos, se procede a la construcción del modelo que los soporte y la API que interactúe con ellos. Para ello, se incluye la dependencia de Spring Boot Data sobre Solr y siguiendo las líneas propuestas en la documentación de Spring Boot, se empieza por crear el modelo con las variables descritas a continuación.

Listado 8.9: Definición del modelo de datos

```
1 @SolrDocument(collection = "news")
2 public class News {
3     @Id
4     @Field
5     private String id;
6
7     @Field
8     private String newspaper;
9
10    @Field
11    private String headline;
12
13    @Field
14    private LocalDateTime date;
15
16    @Field
17    private String link;
18
19    @Field
20    private String descriptionPlain;
21
22    @Field
23    private String descriptionRaw;
24
25    @Field
26    private Set<String> entities;
27 }
```

Cabe destacar que mediante:

```
1 @SolrDocument(collection = "news")
```

Se especifica que el modelo de datos asociado va a conformar los campos de datos que se van a encontrar en el motor de búsqueda Apache Solr. De este modo, Solr va a crear sus índices a partir de esta definición, con lo que se abstrae la configuración del esquema de datos del motor de búsqueda a la declaración simple de variables en el modelo. Este modo de definición es llamado Schemaless Mode.¹

Posteriormente se construye el repositorio y la configuración que darán soporte a las operaciones transaccionales de inserción y recuperación de información.

Listado 8.10: Interfaz del repositorio Solr

```
1 public interface NewsRepository extends SolrCrudRepository<News,
    String> {}
```

Al ser una extensión del repositorio proporcionado por Solr, permite derivar las queries a través del nombre utilizado en las funciones siguiendo las estrategias definidas en la documentación.²

Listado 8.11: Configuración de conexión a cliente Solr

```
1 @Configuration
2 @EnableSolrRepositories(basePackages = "contrast.contrast")
3
4 @ComponentScan
5 public class SolrConfig {
6
7     @Value("${spring.data.solr.host}")
8     String solrURL;
9
10    @Bean
11    public SolrClient solrClient() throws MalformedURLException,
        IllegalStateException {
12        return new HttpSolrClient.Builder("http://" + solrURL).
            build();
13    }
14
15    @Bean
16    public SolrTemplate solrTemplate(SolrClient client) throws
        Exception {
17        return new SolrTemplate(client);
18    }
19 }
```

¹<https://lucene.apache.org/solr/guide/7.7/schemaless-mode.html>

²<https://docs.spring.io/spring-data/solr/docs/4.0.8.RELEASE/reference/html/#repositories.query-methods.details>

Para completar la API mínima faltaría crear un controlador sobre el que realizar las llamadas. Para una primera implementación que permita probar los resultados, se decide crear las operaciones de guardado, borrado y recuperación total.

Listado 8.12: Controlador de la API de Contrast

```
1 @RestController
2 public class NewsController {
3
4     @Autowired
5     private NewsRepository newsRepository;
6
7     @PostMapping("/populate")
8     public void populateSolr() throws IllegalArgumentException {
9         RSSParser parser = new RSSParser();
10        newsRepository.saveAll(parser.parseFeeds());
11    }
12
13    @DeleteMapping("/dropSolr")
14    public void dropSolr() {
15        newsRepository.deleteAll();
16    }
17
18    @GetMapping("/getNews")
19    public Iterable<News> getAll() {
20        return newsRepository.findAll();
21    }
22 }
```

Esta API es ejecutada con el servidor Tomcat integrado en la propia instalación de Spring Boot, ya que este framework provee muchas facilidades para un desarrollo ágil y personalizable.

Por último para satisfacer el último objetivo de la iteración, se procede a la definición del modelo de negocio mediante la creación de un lienzo de modelo para negocios o Lean Canvas. Este es utilizado para realizar una declaración de intenciones de los objetivos que persigue el proyecto; del mismo modo refuerza la importancia de algunas ideas en su implementación. Gran parte de los aspectos que se subrayan son tenidos en consideración únicamente en caso de que se publicase una versión comercial de la aplicación, ya que el cometido de este trabajo de final de grado es lograr una versión académica del estudio de la implementación.

8.1.4. Iteración IV

Para la implementación inicial del front-end, se desestima integrar directamente la librería de componentes, pues esta es una extensión final del framework base y se prima tener el control sobre la visualización de los resultados por encima de la estética en

esta fase de la implementación. En este sentido se crea una única vista para mostrar el resultado de recuperar todas las noticias indexadas.

A la hora de realizar la llamada se percibe un problema de intercambio de recursos de origen cruzado pues ambos están siendo ejecutados en servidores y puertos distintos. Es por ello que se decide implementar un proxy inverso sobre el front-end para que lance todas las llamadas desde la propia dirección sobre la que se ejecuta la API.

Listado 8.13: Proxy inverso del front-end al back-end

```
1 module.exports = {  
2   devServer: {  
3     proxy: {  
4       '/api': {  
5         target: "http://localhost:8080",  
6         ws: true,  
7         changeOrigin: true  
8       }  
9     }  
10  }  
11 }
```

La decisión de ejecutar ambas partes sobre servidores distintos es debido a motivos de desacople, mantenibilidad y escalabilidad. Por una parte, dado que el entorno de la aplicación es el de desarrollo durante la realización del proyecto, es de gran interés mantener los componentes independientes para poder desarrollar de manera rápida y sin tener en cuenta el conjunto. Del mismo modo la caída de una de las partes no debería implicar la caída del servicio entero y uno de los principios que se persigue con esta decisión es la ortogonalidad de las partes implicadas, para que sean reemplazables sin afectar a la otra. En el capítulo 9 queda definida la estrategia que se utilizaría en un entorno de producción.

Llegada la hora de integrar la herramienta de procesamiento de lenguaje natural, en este caso Stanford CoreNLP, se añade como dependencia en el archivo pom.xml tanto la herramienta como el modelo entrenado para el español y se busca su integración en la fase de indexado de noticias. El objetivo que se pretende es enriquecer el ya definido dato de entidades de tal forma que sea la suma de las categorías proporcionadas por el analizador RSS y las entidades extraídas por la herramienta de procesamiento de lenguaje natural. Esto comporta cambios mínimos en la arquitectura actual, pues solo hay que añadir esta fase en la etapa de creación del objeto noticia tras el parseado.

Listado 8.14: Clase de reconocimiento de entidades

```
1 public class NERTagger {  
2  
3   private StanfordCoreNLP pipeline;  
4  
5   public NERTagger() {  
6     Properties props = new Properties();
```

```
7         try {
8             props.load(IOUTils.readerFromString("StanfordCoreNLP-
               spanish.properties"));
9         } catch (IOException e) {
10             e.printStackTrace();
11         }
12         props.put("annotators", "tokenize,ssplit,pos,lemma,ner");
13         this.pipeline = new StanfordCoreNLP(props);
14     }
15
16     public Set<String> getNERTags(String doc) {
17         Set<String> nerTags = new HashSet<String>();
18         CoreDocument document = new CoreDocument(doc);
19         this.pipeline.annotate(document);
20         for (CoreEntityMention em : document.entityMentions()) {
21             nerTags.add(em.text());
22         }
23         return nerTags;
24     }
25 }
```

Al analizador de RSS se le añade las siguientes líneas para satisfacer la extracción de entidades. Cabe remarcar que se examina tanto el cuerpo de la noticia sin las etiquetas HTML, como el titular de la noticia. Esta decisión es debida a que como se ha mencionado anteriormente, hay noticias que no contienen el cuerpo de ella.

Listado 8.15: Uso del extractor de entidades y convergencia de ambos datos

```
1 NERTagger tagger = new NERTagger();
2 Set<String> entities = new HashSet<String>();
3 for (SyndCategory cat : entry.getCategories()) {
4     entities.add(cat.getName());
5 }
6 entities.addAll(tagger.getNERTags(entry.getDescription().getValue
   ().replaceAll("<[^>]*>", "")
7     + " " + entry.getTitle()));
```

8.2. Hito 2

El segundo Hito parte unas semanas más tarde de finalizar el primer prototipo logrado en el anterior para replantear algunas decisiones arquitecturales y de implementación realizadas, que pueden favorecer o mejorar el desarrollo y/o despliegue del proyecto. El enfoque que se brinda a esta segunda etapa de desarrollo es para conseguir una primera versión de la aplicación con orientación a producción.

En este sentido y teniendo en cuenta que el desarrollo se estaba realizando sobre máquinas distintas con sistemas operativos distintos, se decide utilizar un sistema de

virtualización con contenedores para independizar las dependencias y configuraciones de las diferentes partes implicadas en el proyecto. Esta decisión además facilita el posible despliegue en producción pues se utilizaría el mismo modus operandi. La herramienta que se escoge para posibilitar este escenario es Docker.³

Se crea un archivo de especificación de los servicios que van a utilizarse de manera independiente, pero no aislada entre ellos. Se definen instrucciones para crear contenedores de Apache Solr, de la API Spring Boot en Java y del front-end Vue sobre Node.js. El contenido de estos archivos se detalla en el Anexo C.

Se intenta acoplar en el proyecto la herramienta de integración continua Travis CI⁴ sin éxito, debido a no conseguir correr Apache Solr en el susodicho servicio. Dicha imposibilidad causaba que la API fallase al no poder enlazar los repositorios y que por lo tanto, las pruebas y tests definidos fallasen siempre. En el caso de haber escogido Elasticsearch, Travis CI proporciona soporte para el uso de algunos motores de bases de datos.⁵ El objetivo que se perseguía al intentar integrar este servicio era el de poder realizar pruebas para el correcto funcionamiento del proyecto de manera automatizada.

Se rediseña la interfaz de usuario desde los cimientos, esta vez introduciendo elementos de las últimas especificaciones de diseño y teniendo en cuenta la usabilidad y experiencia de usuario. Para ello se hace un mayor hincapié en la experiencia observada en el uso de las aplicaciones similares detalladas en el Marco Teórico. También se realizan simplificaciones y eliminación de adornos y funciones secundarias que no estaban descritas en los requisitos funcionales y que su aportación es irrelevante para el desarrollo académico de la aplicación.

Por último se documenta y describe la arquitectura final seleccionada contando con las tecnologías escogidas en el Hito 1 y las herramientas de soporte escogidas en esta iteración.

8.2.1. Iteración VI

En los resultados mostrados por las llamadas a la API se observan distintos problemas:

- Los resultados son sensibles a mayúsculas/minúsculas y signos de puntuación.
- Las entidades compuestas son tomadas en cuenta como elementos independientes.

Es por ello que se decide declarar un esquema manual de datos del que partirá Solr, dejando de lado el modo Schemaless. Para ello se crean tipos de datos que integren los filtros⁶ y tokenizadores⁷ que sean de interés para el proyecto.

Listado 8.16: Campo de datos personalizado

³<https://www.docker.com/>

⁴<https://travis-ci.org/>

⁵<https://docs.travis-ci.com/user/database-setup/>

⁶https://lucene.apache.org/solr/guide/6_6/filter-descriptions.html

⁷https://lucene.apache.org/solr/guide/6_6/tokenizers.html


```
1 <fieldType name="text_general_keyword_tokenizer" class="solr.  
  TextField" positionIncrementGap="100" multiValued="true">  
2   <analyzer type="index">  
3     <tokenizer class="solr.KeywordTokenizerFactory"/>  
4     <filter class="solr.ASCIIFoldingFilterFactory"/>  
5     <filter class="solr.LowerCaseFilterFactory"/>  
6   </analyzer>  
7   <analyzer type="query">  
8     <tokenizer class="solr.KeywordTokenizerFactory"/>  
9     <filter class="solr.ASCIIFoldingFilterFactory"/>  
10    <filter class="solr.LowerCaseFilterFactory"/>  
11  </analyzer>  
12 </fieldType>
```

En este caso es de interés utilizar el tokenizador Keyword, que trata el campo de texto entero como un solo token y como filtros el ASCIIFolding para las acentuaciones y el LowerCase para evitar la distinción entre mayúsculas y minúsculas.

El esquema de datos final se describe en el Anexo A.

El siguiente paso ha consistido en realizar facetados en las consultas para enriquecer las vistas mostradas en los filtros del front-end. Dichos facetados están descritos en los requisitos funcionales y deben aplicarse sobre las fechas de publicación, el periódico y las entidades.

En este proceso se descubrió una nueva situación a afrontar pues al realizar los facetados por días, Solr agrupa los resultados teniendo en cuenta horas y minutos por lo que era un resultado indeseado. Esto tenía tres posibles soluciones:

- Crear un nuevo campo de datos que sea una copia del campo fecha de publicación pero truncando el valor al día únicamente.
- Despreciar la hora de publicación forzando manualmente un valor fijo para todas las noticias.
- Agrupar los resultados desde el back-end, editando manualmente la respuesta de Solr.

Si el valor de hora y minuto hubiera sido relevante, se hubiera escogido la primera opción, pero dado que para la funcionalidad propuesta es completamente prescindible, se escogió la segunda y se fijó manualmente en la creación de las noticias que todas estuvieran publicadas a medianoche.

Por lo tanto se crean dos llamadas en el controlador y se elimina la llamada de recuperación de todas las noticias.

Listado 8.17: Endpoints finales de obtención de noticias

```
1 @GetMapping(value = "/content/{page}")  
2 @GetMapping(value = "/content/{newspapers}/{initialDate}/{  
  finalDate}/{entities}/{fragment}/{page}")
```

La primera de ellas se propone para la visualización por defecto, sin aplicar ningún filtrado más que realizar la paginación. En la segunda llamada se incluyen todas las posibilidades de filtrado y es la que se propone para utilizar cuando el usuario interactúa con la aplicación introduciendo sus datos o preferencias de búsqueda de información.

8.2.2. Iteración VII

Esta iteración es planteada para incluir en el front-end la librería escogida de diseño de componentes y para optimizar el uso de CoreNLP pues el reconocimiento de entidades necesita varios segundos para mostrar los resultados.

Para satisfacer el primer cometido se empieza por aislar cada parte del front-end en componentes independientes que puedan ser utilizados en diversas vistas, a pesar de que solo se va a construir una única. De esta manera el código tiene una mejor mantenibilidad y es fácilmente escalable. Hecho esto y siguiendo la documentación de Vuetify, se añade la dependencia en el package.json y las correspondientes importaciones en los archivos de configuración del proyecto. Posteriormente se agregan y extienden componentes presentes en el propio framework de Vue.js, más otros implementados por Vuetify, que terminan aportando unas líneas estéticas con mucha riqueza sin tener que establecer manualmente código CSS o de estilos.

En este proceso se descubre que Vuetify es una librería aún más joven que Vue.js y que por lo tanto puede no satisfacer los intereses de muchos usuarios debido a sus limitaciones. Contrast en este caso, no necesita excesivos componentes para la visualización de los datos que pretende mostrar, por lo tanto el resultado previsto en el diseño no se ha visto alterado en gran medida. Aún así se pueden percibir ciertos cambios del diseño previsto al resultado final, que en absoluto empobrecen la calidad del producto final.

Para reducir el coste computacional y los tiempos del uso de extractor de entidades, se ha tenido que analizar a documentación⁸ a fondo para considerar que aspectos se podían personalizar en la llamada. En primer lugar se han desactivado aquellas características que solo estaban implementadas para el modelo de reconocimiento en inglés. Estas son el reconocimiento de variables de tiempo con SUTime y la aplicación de clasificadores números para distinguir porcentajes, cifras, moneda, etc; que además no son de interés para el proyecto. Posteriormente se ha desactivado un componente que ralentizaba el tiempo de computación, que es el clasificador de grano fino. Por defecto coreNLP realiza clasificaciones por un total de 12 clases distintas que luego son recategorizadas en las 23 clases que da soporte el grano fino. En este caso además se han detallado que clasificadores son de interés, utilizando únicamente 3 de los 12 iniciales: organización, localización y persona. Esta ampliación de código se añade al constructor de la clase extractora de entidades, quedando finalmente así.

Listado 8.18: Clase final extractora de entidades

```
1 public class NERTagger {  
2
```

⁸<https://stanfordnlp.github.io/CoreNLP/ner.html>

```
3     private StanfordCoreNLP pipeline;
4     private List<String> selectedClassifiers;
5
6     public NERTagger() {
7         Properties props = new Properties();
8         try {
9             props.load(IOUtils.readerFromString("StanfordCoreNLP-
10             spanish.properties"));
11         } catch (IOException e) {
12             e.printStackTrace();
13         }
14         props.put("annotators", "tokenize,ssplit,pos,lemma,ner");
15         props.setProperty("ner.useSUTime", "false");
16         props.setProperty("ner.applyNumericClassifiers", "false");
17         props.setProperty("ner.applyFineGrained", "false");
18         this.pipeline = new StanfordCoreNLP(props);
19         this.selectedClassifiers = Arrays.asList("ORGANIZATION", "
20             LOCATION", "PERSON");
21     }
22
23     public Set<String> getNERTags(String doc) {
24         Set<String> nerTags = new HashSet<String>();
25         CoreDocument document = new CoreDocument(doc);
26         this.pipeline.annotate(document);
27         for (CoreEntityMention em : document.entityMentions())
28             if (selectedClassifiers.contains(em.entityType()))
29                 nerTags.add(em.text());
30         return nerTags;
31     }
32 }
```

Una vez realizados los cambios se aprecia que el tiempo de computación pasa de segundos a milisegundos, convirtiendo su uso en una propuesta viable para un entorno de producción.

9. Despliegue en producción

Este capítulo va a describir las consideraciones y acciones que deberían tomarse en caso de que la aplicación implementada fuese lanzada a un entorno real en producción.

El desarrollo en la segunda fase del proyecto, es decir, en el Hito 2, ha ido acompañado de una clara orientación a un entorno y despliegue en producción. El primer gran paso que se produjo fue el de adoptar Docker como herramienta de contenerización para poder desplegar fácilmente el conjunto tecnológico del proyecto en cualquier sistema a través del archivo *docker-compose.yml* descrito en el Anexo C. En caso de utilizar este sistema para un entorno de producción, habría que tener en cuenta los siguientes cambios:¹

- Eliminar los enlaces de los volúmenes al código de la aplicación, de tal manera que el código esté contenido en el contenedor.
- Exponer y cambiar el puerto del front-end en el servidor, mantener ocultos los puertos de la base de datos y API. Docker posee un sistema de autodescubrimiento de puertos para los servicios que ejecuta, así que serían visibles entre ellos.
- Configurar variables de entorno correctamente para que las compilaciones estén adecuadas a un entorno de producción.
- Especificar una política de reinicio añadiendo *restart:always* a los servicios que ejecuta Docker para evitar la inactividad.
- Incluir servicios adicionales como un agregador de registros, un sistema de manejo de cache, etc.

Una vez hechas estas configuraciones, se escogería una tecnología de orquestación para crear nodos y réplicas de la aplicación o sus servicios. Si se utilizara Docker Swarm² como tecnología de orquestación, esta provee una integración³ sencilla del archivo *docker-compose.yml*, con la que con cambios mínimos se tendría el orquestador distribuyendo la aplicación en diversos nodos.

Adicionalmente se utilizaría una herramienta de integración continua y distribución continua para automatizar las etapas de desarrollo, mantenimiento y extensión de la aplicación. De este modo podrían realizarse de manera automática tests que atacasen los *end-points* de la API tras las modificaciones y despliegues en caso de éxito. Un ejemplo de una tecnología de esta clase, que actualmente está en alza, es Jenkins.⁴

¹<https://docs.docker.com/compose/production/>

²<https://docs.docker.com/engine/swarm/>

³<https://docs.docker.com/compose/swarm/>

⁴<https://jenkins.io/>

Yendo a las configuraciones nucleares de ambas partes de la aplicación, front-end y back-end, se pueden considerar diversas mejoras, algunas de las cuales ya están siendo aplicadas. En primer lugar, ambas partes serían ejecutadas sobre el mismo servidor, no sobre servidores separados como se ha hecho durante el desarrollo. Nginx⁵ sería un ejemplo de servidor para ejecutar ambas partes y así unificar ciertas configuraciones. Esta aplicación se añadiría como un servicio más en el *docker-compose.yml* de Docker.

Dado que el back-end está construido con Spring Boot, se podría añadir el módulo *Actuator* que ofrece importantes funcionalidades como métricas, diagnósticos de salud, registros, etc en forma de end-points. La propia documentación del framework lo recomienda⁶ como medida en entornos de producción.

Respecto al front-end, al añadir la pieza de código descrita en el extracto 9.1 al archivo de configuración de Vue, la aplicación quedaría optimizada para su puesta en producción. La documentación provee recomendaciones⁷ adicionales para configuraciones específicas.

Listado 9.1: Configuración para puesta en producción de Vue.js

```
1 module.exports = {  
2   mode: 'production',  
3 }
```

Por último, habría que considerar los aspectos legales del uso de los contenidos de los que se nutre la aplicación, es decir, las licencias. En el apartado 3.2.1 se especifican las medidas que habría que tener en cuenta, aún así, se necesitaría buscar nuevas fuentes de contenido.

⁵<https://www.nginx.com/resources/wiki/>

⁶<https://docs.spring.io/spring-boot/docs/current/reference/html/production-ready.html>

⁷<https://vuejs.org/v2/guide/deployment.html>

10. Resultados

El desarrollo de este proyecto ha dado como resultado una aplicación web que tiene como objetivo visualizar prensa digital de distintos medios y relacionar el contenido entre sí. La aplicación es capaz de obtener noticias de medios digitales, realizar una extracción de entidades para poder comparar noticias e indexarlas. También es capaz de listar y mostrar los resultados a través de una capa de visualización ligada a una API y de realizar distintos filtrados y búsquedas según la preferencia del usuario. Además permite relacionar noticias entre sí para ofrecérselas al usuario.

Este resultado ha sido posible gracias al estudio de las propuestas existentes en el mercado y de la definición de un modelo de negocio. Este proceso se ha visto acompañado por la investigación de qué tecnologías serían las idóneas para la implementación de la propuesta, con un resultado final de más de ocho herramientas involucradas en el desarrollo.

Las tecnologías utilizadas en el producto final se dividen en dos grupos: back-end y front-end. El primer grupo está conformado por una API desarrollada con el framework Spring Boot de Java, acompañada de dos módulos nucleares para la implementación del proyecto: el analizador RSS y la herramienta de análisis de lenguaje natural. La primera se ha conseguido mediante Rome Tools, una potente librería para tratar con contenido RSS y la segunda, con la tecnología de la Universidad de Stanford, CoreNLP, de la que se ha utilizado únicamente el extractor de entidades. Todo el contenido generado en la aplicación se ha indexado en el motor de búsqueda Apache Solr. Por otro lado, el front-end ha sido dispuesto con Vue.js sobre JavaScript y aderazado con Vuetify, una librería de componentes y diseño. Docker ha sido el encargado de enlazar todas las partes para un funcionamiento óptimo, escalable y personalizable.

El desarrollo ha sido realizado en dos hitos. El primero ha tenido como objetivo el desarrollo de un mínimo producto viable con funcionalidades sencillas que sentasen las bases de las integraciones de las tecnologías seleccionadas entre sí. El objetivo del segundo ha sido extender y agregar valor a la aplicación a través de más funcionalidades, redefinición de las existentes y enriquecimiento del producto final.

Al finalizar el desarrollo se ha contemplado y documentado el escenario del despliegue en producción a través del estudio de tecnologías que lo faciliten y el análisis de los parámetros necesarios para optimizar las herramientas en este entorno.

El resultado final se puede observar en las imágenes 10.1 y 10.2 que referencian a las diseñadas en la sección 7.4.



Figura 10.1.: Vista de inicio de Contrast

La Fiscalía pedirá al Supremo la suspensión inmediata de los políticos presos





La Fiscalía del Tribunal Supremo elevará este martes un escrito solicitando que se comunique a las Cortes la suspensión de los políticos presos que han obtenido sus actas de diputados y senador. El escrito señala que una vez que Oriol Junqueras, Jordi Turull, Josep Rull, Jordi Sànchez y Raül Romeva hayan tomado posesión de sus escaños, "se comunique a la Mesa del Congreso y Senado que deben proceder a la aplicación inmediata del artículo 384 bis de la Ley de Enjuiciamiento criminal".

Esa comunicación supondrá, según el Ministerio Fiscal, "la suspensión de los mismos en el ejercicio de sus derechos y deberes como diputados y senador".

La nueva Mesa del Congreso se reunirá mañana para fijar postura sobre la suspensión y, previsiblemente, solicitar un informe de los letrados de la cámara. Si antes de ese encuentro el Supremo decide atender al requerimiento de la Fiscalía, la suspensión de los diputados Oriol Junqueras, Jordi Sànchez, Josep Rull, Jordi Turull, y del senador Raül Romeva podría ser inmediata.

La decisión de la Fiscalía del Supremo se produce minutos después de la tensa sesión de constitución del Congreso, en la que el método elegido por los diputados independentistas para acatar la Constitución ha sido puesta en cuestión por Partido Popular, Ciudadanos y Vox. Estas tres formaciones han anunciado que presentarán escritos solicitando la suspensión de los cuatro diputados en prisión. La petición de las tres formaciones podría ser estéril de producirse antes la orden directa por parte del Tribunal Supremo.

Noticias similares

-  eldiario.es - eldiario.es  21/05/2019
- **La charla de Junqueras con Borrell y el frío saludo con Pedro Sánchez: los presos acaparan la atención en el Congreso**
-  Entidades en común
 - Jordi Turull
 - Mesa
 - Josep Rull
 - Jordi Sànchez
 - Congreso
 - Oriol Junqueras
 - Constitución
 - Jordi Sànchez
 - Cortes
 - Vox

CERRAR

Figura 10.2.: Vista de noticia ampliada de Contrast

11. Conclusiones

11.1. Conclusiones

El objetivo principal del proyecto era crear una prueba de concepto académico sobre un lector de prensa digital capaz de relacionar noticias según su contenido. Se han cumplido los objetivos propuestos en el capítulo 2. Por una parte, se ha aplicado una metodología de desarrollo iterativa, observable en el apartado 5. Se han estudiado los servicios existentes, tecnologías de desarrollo y aspectos legales en el capítulo 3. En último lugar, se ha detallado el proceso y la implementación de la aplicación en el capítulo 8.

Las evidencias de la implementación de los objetivos son los resultados descritos en el capítulo 10. En resumidas cuentas, se ha implementado un analizador de contenido RSS mediante Rome Tools, del cual extrae entidades una herramienta de análisis de lenguaje natural como CoreNLP; ambas partes integradas en una API desarrollada en Spring Boot. Se ha utilizado Apache Solr como indexador y se ha definido un esquema de datos de aceptación de noticias, observable en el Anexo A. Por último se ha implementado una capa de visualización mediante Vue.js con distintas opciones de filtrado y búsqueda.

La consecución de los objetivos propuestos ha supuesto todo un logro pues habían áreas de la arquitectura y de las etapas del proyecto en las que no se poseía experiencia alguna y en las que se ha tenido que recabar una gran cantidad de documentación para hacerle frente. Dicho esfuerzo se ha visto combinado con la finalización de las últimas asignaturas del grado y el inicio de la actividad laboral. La experiencia en la implementación y documentación del proyecto ha resultado muy enriquecedora tanto a nivel personal como académico y enormemente gratificante por el cumplimiento de las metas acordadas. Se han utilizado tecnologías diversas que han servido para esclarecer cuáles han sido y son los puntos fuertes del desarrollo y cuáles los mejorables.

11.2. Líneas de trabajo futuras

Existen diversas vertientes de trabajo futuras dependiendo del contexto en el que se quiera englobar la aplicación.

Por una parte puede ser utilizada como una herramienta de seguimiento de medios privada e interna de una institución o empresa. Tan solo hay que facilitar las fuentes del contenido a indexar y la aplicación está preparada para atender esta funcionalidad.

Por otro lado se puede crear una versión comercial que se nutra de otras fuentes distintas a las utilizadas en esta prueba de concepto académico. La principal consideración de este planteamiento es encontrar fuentes con licencias permisivas para tratar el contenido.

Otra línea de trabajo futura consiste en la creación de unidades temáticas sobre las

que agrupar noticias. Con esta propuesta se crean núcleos temáticos contruidos sobre un conjunto de entidades, que a su vez contienen un grupo de noticias. Las noticias no se listarán ordenadas por fecha de publicación, sino que se muestran agrupadas por unidades temáticas con una gran relación entre su contenido.

Bibliografía

- [db engines, 2019] db engines (2019). Motores de búsqueda más populares.
- [Laura Uche, 2017] Laura Uche, A. C. (2017). Dieciséis ejemplos de propaganda, censura y manipulación en tve.
- [Luis Muñoz, 2017] Luis Muñoz, P. A. (2017). Estudio de uso y actitudes de consumo de contenidos digitales. Technical report, ONTSI.
- [Manning et al., 2014] Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.
- [Maurya, 2012] Maurya, A. (2012). *Running Lean: Iterate from Plan A to a Plan That Works (Lean Series)*. O’Reilly Media.
- [Stanford, 2019] Stanford (2019). Soporte oficial de idiomas de corenlp.

A. Anexo A. Esquema de base de datos

A continuación se detallan los esquemas de datos implementados. En la versión final descrita en el presente documento se utiliza el esquema de datos óptimo.

Listado A.1: Esquema de datos mínimo utilizado

```
1 <field name="entities" type="text_general_keyword_tokenizer"  
  uninvertible="true" multiValued="true" indexed="true" stored  
  ="true"/>  
2 <field name="date" type="pdate" uninvertible="true" multiValued  
  ="false" indexed="true" required="true" stored="true"/>  
3 <field name="headline" type="text_es" uninvertible="true"  
  multiValued="false" indexed="true" required="true" stored="true"/>  
4 <field name="id" type="string" multiValued="false" indexed="true"  
  " required="true" stored="true"/>  
5 <field name="newspaper" type="text_general_gkeyword_tokenizer"  
  uninvertible="true" multiValued="false" indexed="true"  
  required="true" stored="true"/>
```

Listado A.2: Esquema de datos óptimo utilizado

```
1 <field name="entities" type="text_general_keyword_tokenizer"  
  uninvertible="true" multiValued="true" indexed="true" stored  
  ="true"/>  
2 <field name="date" type="pdate" uninvertible="true" multiValued  
  ="false" indexed="true" required="true" stored="true"/>  
3 <field name="descriptionPlain" type="text_es" uninvertible="false"  
  multiValued="false" indexed="true" required="false" stored="true"/>  
4 <field name="descriptionRaw" type="text_general_keyword_tokenizer" uninvertible="false"  
  multiValued="false" indexed="false" stored="true"/>  
5 <field name="headline" type="text_es" uninvertible="true"  
  multiValued="false" indexed="true" required="true" stored="true"/>  
6 <field name="id" type="string" multiValued="false" indexed="true"  
  " required="true" stored="true"/>  
7 <field name="link" type="string" uninvertible="false"  
  multiValued="false" indexed="false" required="true" stored="true"/>
```

```
8 | <field name="newspaper" type="text_general_keyword_tokenizer"  
   |   uninvertible="true" multiValued="false" indexed="true"  
   |   required="true" stored="true"/>
```


B. Anexo B. Requisitos funcionales

Identificador del requisito	RF01
Fuente del requisito	Aplicación web
Nombre	Listar noticias
Descripción	La aplicación mostrará un listado de noticias recientes.
Características	<ul style="list-style-type: none">▪ Las noticias estarán ordenadas según el día de publicación.▪ Cada noticia de la lista contendrá, al menos, el titular, la fecha de publicación y el periódico al que pertenece.▪ Este será el comportamiento por defecto de la carga inicial de la página.

Tabla B.1.: Requisito funcional Listar noticias

Identificador del requisito	RF02
Fuente del requisito	Aplicación web
Nombre	Buscar noticias
Descripción	La aplicación obtendrá un listado de noticias según los parámetros introducidos por el usuario en el campo de búsqueda.
Características	<ul style="list-style-type: none">▪ El término de búsqueda se comparará con el titular, contenido y categorías de las noticias.▪ La búsqueda podrá ser realizada con o sin acentos y obtendrá los mismos resultados.▪ Cada noticia de la lista contendrá, al menos, el titular, la fecha de publicación y el periódico al que pertenece.

Tabla B.2.: Requisito funcional Buscar noticias

Identificador del requisito	RF03
Fuente del requisito	Aplicación web.
Nombre	Filtrar noticias por periódico.
Descripción	La aplicación obtendrá un listado de noticias con los periódicos seleccionados.
Características	<ul style="list-style-type: none">▪ Todos los periódicos disponibles estarán seleccionados por defecto.▪ Las noticias filtradas se ordenarán por fecha de publicación.▪ Cada noticia de la lista contendrá, al menos, el titular, la fecha de publicación y el periódico al que pertenece.▪ El periódico indicará en todo momento cuantas noticias de ese periódico está mostrando.▪ Se podrán aplicar distintos filtros simultáneamente.

Tabla B.3.: Requisito funcional Filtrar noticias por periódico

Identificador del requisito	RF04
Fuente del requisito	Aplicación web.
Nombre	Filtrar noticias por rango de fechas.
Descripción	La aplicación obtendrá un listado de noticias según el rango de fechas definido por el usuario.
Características	<ul style="list-style-type: none">■ Por defecto el rango de fechas estará situado entre la noticia más antigua y la más reciente.■ Cada noticia de la lista contendrá, al menos, el titular, la fecha de publicación y el periódico al que pertenece.■ Los días en que hay noticias tendrán un indicador que marque la cantidad.■ Se podrán seleccionar únicamente fechas que contengan noticias.■ Se podrán aplicar distintos filtros simultáneamente.

Tabla B.4.: Requisito funcional Filtrar noticias por rango de fechas

Identificador del requisito	RF05
Fuente del requisito	Aplicación web.
Nombre	Filtrar noticias por entidades.
Descripción	La aplicación obtendrá un listado de noticias según las entidades seleccionadas por el usuario.
Características	<ul style="list-style-type: none"> ■ Por defecto se mostrarán noticias con todas las combinaciones posibles de entidades. ■ Cada noticia de la lista contendrá, al menos, el titular, la fecha de publicación y el periódico al que pertenece. ■ Se podrán seleccionar cualquier cantidad de entidades, siempre y cuando hayan noticias que las contengan. ■ Se podrán aplicar distintos filtros simultáneamente.

Tabla B.5.: Requisito funcional Filtrar noticias por entidades

Identificador del requisito	RF06
Fuente del requisito	Aplicación web.
Nombre	Ampliar noticia.
Descripción	La aplicación mostrará un botón en cada noticia para ampliar su vista y facilitar su lectura.
Características	<ul style="list-style-type: none"> ■ La aplicación ampliará la vista de la noticia ocupando más espacio en pantalla y bloqueando el resto del contenido. ■ La noticia mostrará, además de los ítems mencionados anteriormente, el cuerpo de la noticia. ■ Esta visualización tendrá elementos accionables que permitan volver a la vista anterior.

Tabla B.6.: Requisito funcional Ampliar noticia

Identificador del requisito	RF07
Fuente del requisito	Aplicación web.
Nombre	Buscar noticias similares.
Descripción	La aplicación buscará noticias similares a la seleccionada.
Características	<ul style="list-style-type: none">▪ Este comportamiento se dará únicamente cuando se amplíe una noticia.▪ Las noticias relacionadas se ordenarán según su similitud con la original.▪ De cada noticia relacionada se mostrará el titular, periódico, fecha de publicación y entidades en común con la seleccionada.

Tabla B.7.: Requisito funcional Buscar noticias similares

Identificador del requisito	RF08
Fuente del requisito	Aplicación web
Nombre	Indexar noticias
Descripción	La aplicación indexará las noticias que obtenga del analizador
Características	<ul style="list-style-type: none">▪ Las noticias se indexarán si se ajustan al esquema de datos definido.▪ De cada noticia se almacenará el periódico, el titular, la fecha de publicación y las entidades como mínimo. Este será el esquema de datos mínimo de aceptación.▪ De cada noticia se almacenará adicionalmente el cuerpo de la noticia y el enlace. Estos datos junto a los especificados en el ítem anterior, conformarán el esquema de datos óptimo.

Tabla B.8.: Requisito funcional Indexar noticias

Identificador del requisito	RF09
Fuente del requisito	Aplicación web
Nombre	Mantener formato
Descripción	La aplicación mantendrá el formato original de las noticias en la visualización
Características	<ul style="list-style-type: none"> ■ Las noticias contendrán las etiquetas HTML del medio original que les dan formato. ■ Las noticias contendrán los elementos multimedia originales del medio digital.

Tabla B.9.: Requisito funcional Mantener formato

Identificador del requisito	RF10
Fuente del requisito	Aplicación web
Nombre	Obtener noticias
Descripción	La aplicación obtendrá noticias de distintos medios
Características	<ul style="list-style-type: none"> ■ Las fuentes de las noticias quedarán definidas en un fichero externo. ■ Las noticias que se obtengan serán analizadas para ajustarse al esquema de datos.

Tabla B.10.: Requisito funcional Obtener noticias

Identificador del requisito	RF11
Fuente del requisito	Aplicación web
Nombre	Facetar por contenido
Descripción	La aplicación realizará distintos facetados que permitan complementar la información proporcionada por los filtrados
Características	<ul style="list-style-type: none">■ El facetado se realizará por fuentes, entidades y fechas de las noticias.■ El resultado del facetado se mostrará junto a los filtros para una visualización óptima.

Tabla B.11.: Requisito funcional Facetar por contenido

Identificador del requisito	RF12
Fuente del requisito	Aplicación web
Nombre	Paginar noticias
Descripción	La aplicación mostrará un conjunto de noticias por página y permitirá avanzar y retroceder entre ellas.
Características	<ul style="list-style-type: none">■ El número de noticias a visualizar por página vendrá determinado por las dimensiones del contenido a mostrar, pero siempre se mantendrá una matriz cuadrada.■ Se proporcionarán elementos visuales sencillos para la navegación entre páginas, del mismo modo que para conocer en qué página se está.

Tabla B.12.: Requisito funcional Paginar noticias

Identificador del requisito	RF13
Fuente del requisito	Aplicación web
Nombre	Reconocer entidades
Descripción	La aplicación utilizará extractores para reconocer entidades en las noticias.
Características	<ul style="list-style-type: none">■ Las entidades extraídas se indexarán junto a las categorías proporcionadas por el recuperador de medios RSS.■ Se hará una selección de las entidades relevantes a indexar en la base de datos.

Tabla B.13.: Requisito funcional Reconocer entidades

C. Anexo C. Archivos Docker

Listado C.1: Archivo docker-compose.yml

```
1  version: "3.7"
2  services:
3    api:
4      image: spring-boot-custom
5      container_name: api
6      build: ./backend
7      ports:
8        - 8080:8080
9      depends_on:
10       - db
11      networks:
12        - contrast-net
13
14    db:
15      image: solr-from-schema
16      build: .
17      container_name: solr
18      entrypoint:
19        - docker-entrypoint.sh
20        - solr-precreate
21        - news
22        - /opt/solr/server/solr/configsets/newsConfig
23      ports:
24        - 8983:8983
25      networks:
26        - contrast-net
27      volumes:
28        - ./data:/opt/solr/server/solr/mycores
29
30    node:
31      image: node-vue
32      container_name: vue
33      build:
34        context: ./frontend
35        args:
36          - NODE_ENV=development
37      ports:
38        - 80:80
39      environment:
```

```
40     - NODE_ENV=development
41     depends_on:
42     - api
43     networks:
44     - contrast-net
45
46 networks:
47     contrast-net:
48     name: contrast-net
49
50 volumes:
51     data:
```

Listado C.2: Dockerfile de Solr

```
1 FROM solr:7.7.1-alpine
2 COPY coredir /opt/solr/server/solr/configsets/newsConfig
```

Listado C.3: Dockerfile de Vue.js

```
1 # build stage
2 FROM node:lts-alpine as build-stage
3 RUN npm i npm@latest -g
4 WORKDIR /app
5 COPY package*.json ./
6 RUN npm install --no-optional && npm cache clean --force
7 COPY . .
8 RUN npm run build
9
10 # production stage
11 FROM nginx:stable-alpine as production-stage
12 COPY --from=build-stage /app/dist /usr/share/nginx/html
13 EXPOSE 80
14 CMD ["nginx", "-g", "daemon off;"]
```

Listado C.4: Dockerfile de Java

```
1 FROM openjdk:8-jdk-alpine as build
2 WORKDIR /workspace/app
3
4 COPY mvnw .
5 COPY .mvn .mvn
6 COPY pom.xml .
7 COPY src src
8
9 RUN ./mvnw install -DskipTests -Dmaven.test.skip=true
10 RUN mkdir -p target/dependency && (cd target/dependency; jar -xf
    ../*.jar)
```

```
11 FROM openjdk:8-jre-alpine
12 VOLUME /tmp
13 ARG DEPENDENCY=/workspace/app/target/dependency
14 COPY --from=build ${DEPENDENCY}/BOOT-INF/lib /app/lib
15 COPY --from=build ${DEPENDENCY}/META-INF /app/META-INF
16 COPY --from=build ${DEPENDENCY}/BOOT-INF/classes /app
17
18 EXPOSE 8080
19 ENTRYPOINT ["java","-XX:+UnlockExperimentalVMOptions","-XX:+
20   UseCGroupMemoryLimitForHeap","-XX:MaxRAMFraction=1","-
   XshowSettings:vm","-cp","app:app/lib/*","contrast.contrast.
   Application"]
```