Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Diseño e implementación de un extractor de noticias automatizado

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Gilabert Perea, Xavier

Tutor: Sáez Barona, Sergio

Tutor externo: García Martínez, María Mercedes

Curso 2020-2021

Resumen

El presente trabajo de final de grado se enfoca, para la industria del lenguaje, en el diseño e implementación de un extractor de noticias automatizado. Esta aplicación tiene la funcionalidad tanto de extraer automáticamente noticias de las fuentes que nosotros le insertamos al sistema, y categorizarlas en diferentes sectores y categorías. Además, podemos valorar las noticias en una escala de relevancia para que el sistema aprenda y en un futuro sepa por ella misma cuales son las noticias que nos pueden interesar más y cuáles no. Gracias a esta aplicación podemos analizar una a una las noticias de multitud de páginas web de las que podemos cambiar el título y tener un pequeño resumen para guardarlo en nuestra base de datos.

Con esta herramienta se abre una oportunidad para el ahorro de tiempo muy importante a la hora del minado de datos que nos pueda interesar, ya sea por idioma o por categoría. La herramienta incorpora mucha variedad de noticias de muchas fuentes diferentes. En un futuro es posible que la extracción no sea únicamente de noticias que puede aplicándola a otro tipo de texto siendo más útil para más gente.

Palabras clave: Extracción automática de texto, Noticias, Categorización, Base de datos, Automatización, Minería de datos, Inteligencia Artificial.

Resum

El present treball de final de grau s'enfoca, per a la indústria del llenguatge, en el disseny i implementació d'un extractor de notícies automatitzat. Aquesta aplicació té la funcionalitat tant d'extraure automàticament notícies de les fonts que nosaltres li inserim al sistema, i categoritzar-les en diferents sectors i categories. A més, podem valorar les notícies en una escala de rellevància perquè el sistema aprenga i en un futur sàpia per ella mateixa quals són les notícies que ens poden interessar més i quins no. Gràcies a aquesta aplicació podem analitzar una a una les notícies de multitud de pàgines web de les quals podem canviar el títol i tindre un xicotet resum per a guardar-lo en la nostra base de dades.

Amb aquesta ferramenta s'obri una oportunitat per a l'estalvi de temps molt important a l'hora del minat de dades que ens puga interessar, ja siga per idioma o per categoria. La ferramenta incorpora molta varietat de notícies de moltes fonts diferents. En un futur és possible que l'extracció no siga únicament de notícies que pot aplicant-la a un altre tipus de text sent més útil per a més gent.

Paraules clau: Extracció automàtica de text, Notícies, Categorització, Base de dades, Automatització, Mineria de dades, Intel·ligència Artificial.

Abstract

This final degree project focuses on the design and implementation of an automated news extractor for the language industry. This application has the functionality to both automatically extract news from the sources we insert into the system and categorise them into different sectors and categories. In addition, we can rate the news on a scale of relevance so that the system learns and, in the future, knows for itself which are the news that may interest us more and which are not. Thanks to this application we can analyse one by one the news of a multitude of web pages of which we can change the title and have a small summary to save it in our database.

This tool opens up a very important time-saving opportunity when it comes to mining data that may be of interest to us, either by language or by category. The tool incorporates a wide variety of news from many different sources. In the future it is possible that the extraction will not only be of news but can be applied to other types of text making it more useful for more people.

Keywords: Automatic text extraction, News, Categorisation, Database, Automation, Data mining, Artificial Intelligence.

Tabla de contenidos

Contenido

Ilustraciones

Tablas

1. Introducción

La introducción del proyecto va a estar dividida en 4 partes diferenciadas: Descripción, Motivación, Planificación y Estructura. Primero describiremos de que trata el proyecto, a continuación, la motivación para realizar este trabajo, después la planificación que he llevado a cabo para la realización del proyecto y, por último, la estructura que tiene la memoria.

1.1. Descripción

El proyecto trata de una herramienta web encargada de extraer y guardar en una base de datos las partes de noticias de multitud de fuentes que nosotros mismo configuraremos. Esta herramienta es capaz de extraer el título, el resumen o entradilla, la imagen, el texto completo y el HTML de la web de la noticia. HTML es un lenguaje que se utiliza en las páginas de Internet que se trata de un formato abierto que permite ordenar y etiquetar diversos documentos dentro de una lista.

La herramienta también tiene la capacidad de poder evaluar las noticias una a una para cambiar el título o el resumen, si así lo ve necesario el usuario y evaluarlo por categorías a las cuales pertenece esa noticia. La aplicación permite añadir infinitas fuentes de las cuales poder extraer noticias.

1.2. Motivación

Hoy en día, las empresas necesitan cantidades muy grandes de datos, los cuales analizan y los tratan para conseguir mejores resultados en sus proyectos y además utilizándolos, por ejemplo, en entrenamientos de máquinas automatizadas y obtener mejores resultados y mejor precisión y eficiencia.

Por ello se ha visto una oportunidad en la que se puede aprender y adquirir conocimientos en el desarrollo web tan importante en la informática, con esta página se puede conseguir grandes cantidades de datos en poco tiempo y diariamente, también se puede observar la gran diversidad de temas e idiomas que hay en los datos, ya que hay mucha diversidad de todas las fuentes que hemos añadido.

Este proyecto ha sido un proyecto real en la empresa en la cual estoy realizando las practicas, Pangeanic, donde hay un cliente interesado en esta herramienta. Además, a la empresa le ayuda a conseguir cantidades de datos para los entrenamientos de máquinas de traducción automática.

1.3. Planificación

El proyecto sigue una planificación la cual se va siguiendo durante el proceso hasta llegar a la meta que nos interesa. Pasos que seguir:

1. Minar datos: Pensamos una nueva manera de minar datos y que datos podían ser aquellos que nos interesaban minar. Nuestra decisión final fue las noticias y los diarios digitales, también porque una empresa cliente estaba interesada en ese tema.

2. Método de minado de datos: Después de saber qué es lo que nos interesa minar decidimos cómo hacerlo. La decisión fue con un crawler (), el cual sabemos que es muy utilizado para extraer información de las páginas web y se puede configurar para que obtenga lo que necesitemos.

3. Implementación: Fase de implementación e instalación de la aplicación y del crawler, para conseguir que extrajera lo que nos interesaba y lo guardase en una base de datos.

4. Diseño de la herramienta: Diseñar la página web y la interfaz con la cual trabajaremos para que sea cómoda y amigable para todos los usuarios.

5. Comprobaciones y actualizaciones: Comprobar y evaluar con el cliente las especificaciones requeridas, las cuales se iban implementando y mostrando al cliente para satisfacer sus necesidades.

6. Mantenimiento: Mantener la herramienta en funcionamiento y corregir errores que pueden surgir por los cambios en las webs de las fuentes que hemos insertado.

1.4. Estructura

La estructura de la memoria se va a dividir en los siguientes apartados:

-Introducción: Primer apartado en el que se describe brevemente el proyecto y cómo surgió la idea de realizarlo.

-Estado del arte: Apartado en el cual analizaremos el estado actual y la historia de los métodos, programas y lenguajes que se han utilizado durante el proyecto.

-Descripción del proyecto: En esta sección se describe con mucho más en detalle el proyecto, todas las funciones, los objetivos y como se ha llevado a cabo.

-Análisis: En este apartado se explica el propósito, utilidades y facilidades de la aplicación, los requisitos de uso y los casos de uso.

-Diseño: Contiene una representación del diagrama de clases y análisis de toda la base de datos.

-Desarrollo: Análisis de la estructura del código y fragmentos.

-Interfaz: Explicación de la interfaz a nivel de usuario de la herramienta y elementos de cada pantalla.

-Pruebas y Evaluación: Análisis de las pruebas realizadas sobre la aplicación y la evaluación para el correcto funcionamiento de la herramienta.

-Conclusión y Trabajo futuro: Conclusión del proyecto y trabajo futuro explicando las funciones que se pueden implementar en la herramienta.

2. Estado del arte

El estado del arte es el estudio de los estados actuales de las técnicas y tecnologías que están relacionadas con el proyecto para la posterior elección de estas.

2.1. Introducción

En este apartado se analizan los métodos, tecnologías y herramientas que han sido utilizados en el proyecto. Ya que nuestra herramienta es una aplicación web vamos a hablar sobre la historia de las páginas web y lo que han avanzado desde que se crearon.

Se explica en qué consisten estas herramientas, las utilidades que tienen y en el estado en la que están actualmente y lo que han avanzado y evolucionado desde que existen.

2.2. Tecnologías desarrollo web

Hace un tiempo, las páginas web eran sencillas y sin con poco contenido, pero poco a poco se han desarrollado y la tecnología ha avanzado creando herramientas nuevas. Consiguiendo crear complejas páginas web con muchos datos y contenido que hace años no nos podíamos imaginar.

Vamos a ver la historia sobre las páginas Web, sus inicios y como han ido evolucionando hoy en día. También veremos las tecnologías que se utilizan para realizar el desarrollo web.

2.2.1. Historia

Tim Berners-Lee, un científico británico, inventó la World Wide Web (WWW) en 1989, mientras trabajaba en el CERN[1]. La Web se concibió y desarrolló originalmente para satisfacer la demanda de intercambio automatizado de información entre científicos de universidades e institutos de todo el mundo.

La idea básica de la WWW era fusionar las tecnologías en evolución de las computadoras, las redes de datos y el hipertexto en un sistema de información global poderoso y fácil de usar [1].

Imagen 1: Primera página web

Esta imagen representa a la World Wide Web. El primer sitio web se puso en marcha en noviembre de 1992. El sitio web se anunció públicamente (mediante una publicación en el grupo de noticias de Usenet) el 6 de agosto de 1991. El primer sitio web estaba 100% basado en texto. El hipertexto azul predeterminado era el único toque de color. En los primeros días de la web, publicar una página era emocionante por sí solo.

Esta página contenía enlaces a información sobre el proyecto WWW en sí, incluida una descripción del hipertexto, detalles técnicos para crear un servidor web y enlaces a otros servidores web a medida que estaban disponibles.

Sólo unos pocos usuarios tenían acceso a una plataforma informática NeXT en la que se ejecutaba el primer navegador, pero pronto comenzó el desarrollo de un navegador "en modo online" más simple, que podía ejecutarse en cualquier sistema. Fue escrito por Nicola Pellow durante sus prácticas de estudiante en el CERN.

En 1991, Berners-Lee lanzó su software WWW. Incluía el navegador "en modo de línea", software de servidor web y una biblioteca para desarrolladores. En marzo de 1991, el software estuvo disponible para los trabajadores que usaban computadoras del CERN. Unos meses más tarde, en agosto de 1991, anunció el software WWW en los grupos de noticias de Internet y el interés en el proyecto se extendió por todo el mundo [2].

Solo dos años después del lanzamiento de la WWW, se presentó a ALIWEB. ALIWEB (Archie Like Indexing for the WEB) el cual se considera el primer motor de búsqueda web. ALIWEB abrió sus puertas en noviembre de 1993, proporcionando a los usuarios enlaces útiles al mejor contenido de la web. En solo dos años, comienza a ver cómo el diseño cobra vida. El objetivo de ALIWEB era ayudar a los usuarios a encontrar información útil. Querían que los usuarios se sintieran atraídos por los enlaces del sitio. Usando un fondo de color, llamaron la atención sobre los elementos más importantes de la página.

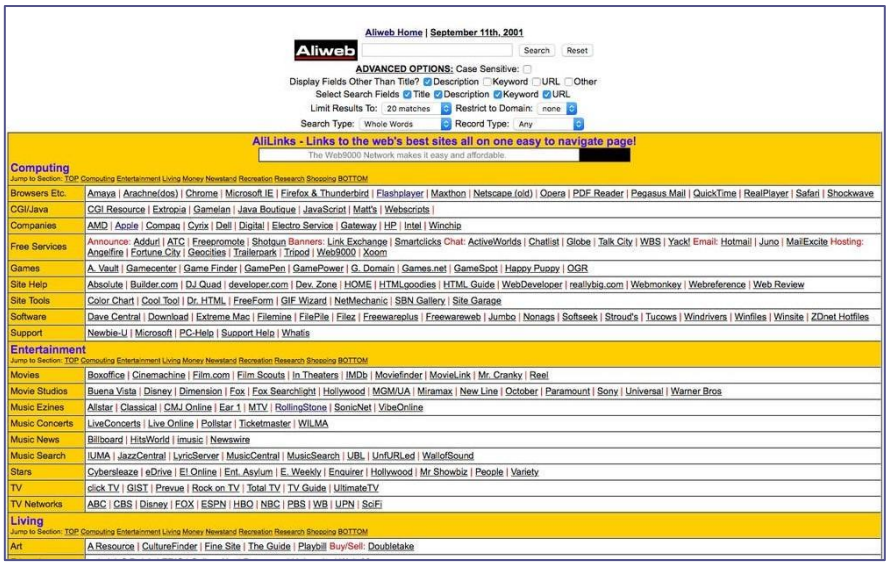

Imagen 2: Primer buscador web

En 1993, las empresas empezaron a darse cuenta del potencial publicitario e informativo de la WWW. A parte de las webs universitarias se crearon las primeras webs comerciales. MTV fue la primera en lanzar su sitio Web. VJ Adam Curry dirigió el proyecto de manera no oficial y personal. Esta fue la imagen que se vio cuando se lanzó su sitio en 1993. Se ve una gran diferencia con otros sitios de años atrás. Según un estudio del investigador del MIT Matthew Gray, a finales de 1993, había 623 sitios web. Internet estaba empezando a despegar y también el diseño de los sitios web empezaba a ser más visible y amigable [3].

Imagen 3: Primera página comercial

En los siguientes años la tecnología despegó y en poco tiempo se vieron muchos cambios significativos que ayudaban a tener mejores y más útiles páginas web. Pronto se incorporó el JavaScript ayudando a superar las limitaciones del HTML estático al permitir traer algo de movimiento a la web. Esto dio origen a la ventana "emergente". El problema es que tiene que cargarse en la parte superior de la página existente, lo que

hace que los sitios se carguen más lentamente. Muchas de las primeras funciones de JavaScript se pudieron realizar posteriormente a través de CSS.

Flash cambió el panorama del diseño de sitios web. Por primera vez, los diseñadores pudieron crear cualquier forma, agregar animación y desarrollar sitios más atractivos que nunca con una sola herramienta. La página final compactaría toda la información en un solo archivo para cargar. El problema principal era que no todos los usuarios web tenían un complemento Flash instalado y los sitios Flash tardaban mucho más en cargarse. La era de Flash nos trajo páginas de bienvenida animadas.

Actualmente los especialistas en marketing entienden que el sitio web de una empresa suele ser la primera oportunidad que tendrán para conectarse con un cliente potencial. Y todo en esta es importante, los colores, el logotipo, la claridad de la web, páginas que atraigan al cliente y que su contenido sea relevante y lo relacionen con sus productos o servicios. También el tiempo de carga en una web tiene que ser rápido, ya que la gente no quiere esperar en un sitio lento cuando hay millones de páginas [3].

Durante estos últimos años, sobre todo los que hemos vivido en una pandemia, se ha visto la importancia que tiene internet y las webs en la sociedad hoy en día. Todo el mundo está conectado a internet y diariamente se utiliza como herramienta de trabajo o social.

Todas las empresas saben que en estos tiempos es necesario tener una página web para retener a sus clientes y llegar a nuevos clientes de todo el mundo. Los desarrolladores web son la solución perfecta para que las empresas creen una puerta de entrada digital para mostrar sus servicios y vender sus productos.

Como hemos podido ver, en tan solo 30 años los sitios web han crecido inmensamente, desde la creación de la primera página web se han ido creando y aplicando tecnologías para que fuese mejorando y fuese atractivo y útil para la gente. Y como se puede ver, en la actualidad las páginas web pueden hacer muchas más acciones que antes y pueden albergar muchos más datos e información, y por eso, la mayoría de las empresas tienen una página web que saben que es importante cuidarla y que tiene que ser atractiva y llamativa para sus clientes o usuarios.

2.2.2. React

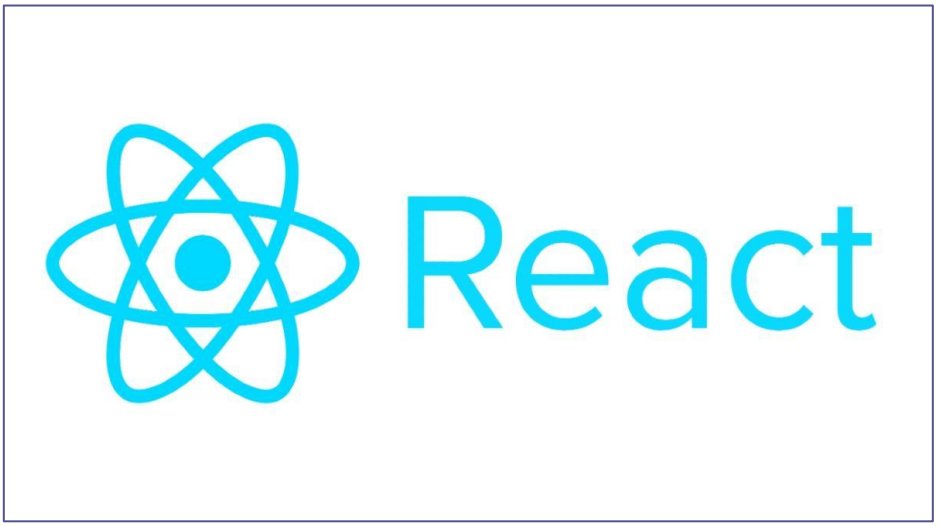

Imagen 4: Logo React

React.js es la biblioteca de JavaScript de código abierto más popular para crear aplicaciones web. Se utiliza para crear Interfaces de Usuario (UI) específicamente para aplicaciones de una sola página. React puede gestionar la capa de visualización de aplicaciones web y móviles. React también nos permite crear componentes de UI reutilizables. React fue creado por primera vez por Jordan Walke, un ingeniero de software que trabaja para Facebook. React se implementó por primera vez en el suministro de noticias de Facebook en 2011 y en Instagram.com en 2012.

React permite a los desarrolladores crear grandes aplicaciones web que pueden cambiar datos, sin recargar la página. El objetivo principal de React es ser rápido, escalable y simple. Funciona solo en las interfaces de usuario de la aplicación. Esto corresponde a la vista en la plantilla MVC (Modelo-Vista-Controlador) que es el patrón en el diseño de software para implementar interfaces de usuario, datos y lógicas [4].

Las propiedades de React.js incluyen lo siguiente:

- Es declarativo.
- Es simple.
- Se basa en componentes.
- Es compatible con el lado del servidor.
- Es extenso.
- Es rápido.
- Es fácil de aprender.

Las siguientes características de ReactJS son muy útiles al crear la interfaz de usuario.

The Virtual DOM

El modelo de objeto de documento o manipulación DOM (Document Object Model) es una de las partes más importantes de la web. El DOM es la estructura de objetos que genera el navegador cuando se carga un documento y se puede alterar mediante Javascript para cambiar dinámicamente los contenidos y aspecto de la página [5].

Pero actualizar el DOM es muy lento, incluso más lento que muchas operaciones de JavaScript. La mayoría de los marcos de JavaScript actualizan todo el DOM, lo que lo hace más lento. Bueno, no es necesario actualizar todo el DOM, en su lugar, estos marcos deberían actualizar solo la parte del DOM que se requiere para actualizar. Esto es lo que hace el DOM virtual.

El DOM virtual es una copia ligera del DOM original. Conservando todas las propiedades del DOM real. Cuando se realiza la actualización, se actualiza todo el DOM virtual. Esto suena ineficiente, pero el DOM virtual es mucho más rápido que el original. Una vez que se completa la actualización, ReactJS compara el DOM virtual actualizado y el DOM original actualizado previamente, esto se conoce como diferenciación. Después de comparar ambos DOM, ReactJS sabe exactamente dónde está la diferencia. Por lo tanto, actualiza el DOM original solo donde estaba la diferencia. Esta es la razón por la que la manipulación DOM de ReactJS es mucho más rápida que otros marcos [6].

One-way Data Binding

La vinculación de datos unidireccional significa que a través de toda la aplicación los datos fluyen solo en una dirección. Esto le da un mejor control sobre él. Los datos se pasan del componente principal al componente secundario a través de accesorios de solo lectura. Estos accesorios no se pueden enviar de vuelta al componente principal. Así es como funciona el enlace de datos unidireccional. Sin embargo, el componente secundario puede comunicarse con el componente principal para actualizar el estado mediante funciones de devolución de llamada [7].

Componentes

Una página web en ReactJS se divide en varios componentes. Cada componente define una vista o una parte de una vista. ReactJS está basado en componentes. La lógica de los componentes está escrita en JavaScript en lugar de en plantillas, por lo que es fácil pasar los datos a través de la aplicación y mantener el estado fuera del DOM [8].

Declaraciones condicionales

Una de las mejores características de ReactJS es que podemos usar declaraciones condicionales dentro de JSX, el cual se explica más adelante. Esto ayuda mucho al mostrar datos en el navegador de acuerdo con las condiciones [6].

Ciclo de vida de los componentes

Los métodos de ciclo de vida proporcionados por ReactJS son muy útiles durante el desarrollo. Cada componente de ReactJS tiene un ciclo de vida propio. Son una serie de métodos que se ejecutan en diferentes etapas de ejecución:

1ª Etapa) Montado: en esta etapa se crea el componente y se pone en el DOM. Tiene cuatro métodos integrados que se llaman en orden:

- constructor: es propio de JavaScript y se ejecuta al realizar la instancia de la clase, en este método se debe inicializar los estados del componente, enlazar eventos utilizando *bind* y declarar variables globales que sirvan dentro del componente. Este método recibe *props* como parámetro, el cual son las propiedades del componente.

- *componenteWillMount():* este método se llama justo antes de que el componente sea montado, aquí se cambian los estados y no se debe hacer llamados a API's. Este ciclo no recibe parámetros.

- render: este ciclo contiene toda la estructura del componente, todo aquello que se tiene que renderizar. Aquí se recomienda utilizar las *props*. No recibe parámetros.

- *componentDidMount():* este ciclo se ejecuta una vez el componente está montado, cuando se muestra en nuestro front-end. Aquí realizamos suscripciones a eventos y llamamos a API's. Este ciclo no recibe parámetros.

2ª Etapa) Actualización: Esta etapa ocurre cuando el componente es actualizado, ya que ha recibido nuevos datos y cambiará su representación. Tiene cuatro métodos integrados que se llaman en orden:

- *componentWillReceiveProps():* Se ejecuta al recibir un cambio en las propiedades del componente, aquí deberemos realizar los cambios en el estado basándonos en las nuevas propiedades.

- *shouldComponentUpdate():* Este ciclo nos permite indicar si nuestro componente tiene que renderizar nuevamente, con esto conseguimos, apoyándonos en validaciones, mejorar el *performance* de nuestra aplicación. Recibe parámetros.

- *componentWillUpdate():* se ejecutará antes de hacer el re-renderizado, si no se ha utilizado el *shouldComponentUpdate()* o este no ha devuelto *true* anteriormente. Recibe parámetros actualizados.

- *componentDidUpdate():* Se ejecutará justo después del re-renderizado. Tiene la misma función que el *componentDidMount()*

3ª Etapa) Desmontaje: En esta etapa se cierran todos aquellos procesos que se hayan podido quedar abiertos, o variables inicializadas y que están en la memoria para que no ocupen espacio. Tiene un método:

- *componentWillUnmount():* Se ejecuta antes de que el componente sea desmontado, aquí se deben quitar todos los procesos que puedan haberse quedado pendientes. No recibe ningún parámetro [9].

JSX

JSX es la extensión de la sintaxis de JavaScript. Su sintaxis es similar a HTML. Los componentes de ReactJS están escritos en JSX. JSX puede verse como una combinación de JavaScript y XML. Su sintaxis es muy simple, lo que facilita mucho la escritura de componentes [10].

JSX es un componente de ReactJS que ayuda a crear elementos ReactJS de manera muy eficiente. Es una extensión de JavaScript. JSX incluye tanto lógica como marcado. A diferencia de AngularJS, no necesitamos crear archivos separados para la lógica y el marcado, lo que permite ahorrar mucho tiempo [6].

```
const element = <h1>Hello, world! </h1>;
```

Como se puede observar en el código anterior la sintaxis es muy similar a la de HTML, pero también usa variables similares a JavaScript. Esto es JSX, donde almacenamos en una variable una etiqueta de HTML, lo que resulta mucho más cómodo y fácil a la hora de programar.

Hooks

Los *hooks* son la nueva característica introducida en la versión React 16.8. Te permite usar el estado y otras características de React sin escribir una clase. Los *hooks* son las funciones que "enganchan" al estado de React y las características del ciclo de vida de los componentes de la función. No funciona dentro de las clases [11].

Los *hooks* son compatibles con versiones anteriores, lo que significa que no contienen cambios importantes. Además, no reemplaza el conocimiento de los conceptos de React [12].

Únicamente tiene 2 reglas:

- No llamarlos dentro de bucles, condiciones o funciones anidadas. Siempre tienen que usarse en el nivel superior de las funciones de React.
- No llamarlos desde funciones regulares de JavaScript.

Los hooks más utilizados son: *useState* y *useEffect*:

useState

```
const [state, setState] = useState (initialState);
```

Nueva forma de declarar un estado en la aplicación React. Con esta sintaxis conseguimos configurar el estado con un valor inicial y tener una función para poder actualizarlo.

```
const [contador, setCount] = useState ( 0 );
```

En nuestro ejemplo, tenemos un estado que es un contador, el cual le hemos indicado que el valor inicial es 0. Y tenemos una función, *setCount*, que dentro del componente nos permitirá actualizar el estado cuando queramos.

useEffect

Este *hook* nos permite realizar efectos secundarios en los componentes de la función. Al utilizar hooks dejamos de utilizar los métodos de ciclo de vida de los componentes. Este Hook equivale a los métodos: *componentDidMount(), componentDidUpdate()* y *componentWillUnmount()* [11].

```
useEffect(() => {
      document.title = 'Ha hecho clic en $ {count} veces';
},[])
```

En nuestro ejemplo, el *hook* tiene la misma función que *componentDidMount()* y *componentDidUpdate()*. Justo antes de ser montado o re-renderizado, el título de la página en el navegador cambia por el valor indicado.

2.2.4. NodeJS

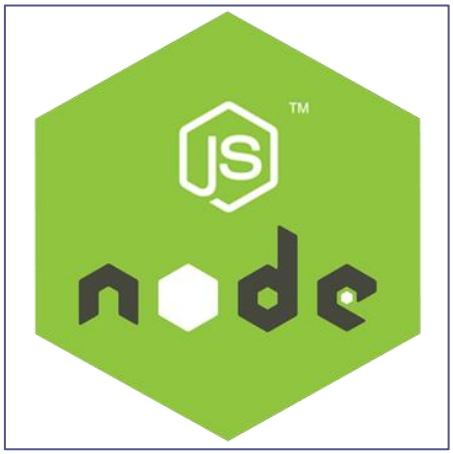

Imagen 5: Logo NodeJS

Node.js es una plataforma del lado del servidor construida sobre el motor JavaScript de Google Chrome. Es un entorno de ejecución multiplataforma de código abierto para desarrollar aplicaciones de red del lado del servidor.

Se creó con el objetivo de poder construir plataformas web con una alta escalabilidad porque los entornos tradicionales sufrían cuando había una alta cantidad de usuarios. Las aplicaciones se escriben en JavaScript y se pueden ejecutar en todos los sistemas operativos [13].

Las características principales son las siguientes:

- Asíncronas y controladas por eventos: las API de la biblioteca Node.js son asíncronas, por lo tanto, no tienen bloqueo. Un servidor basado en Node.js no espera a que la API le devuelva los datos, sino que pasa a la siguiente API y con un mecanismo de notificación por eventos ayuda a obtener la respuesta de la llamada a la API anterior.
- Muy rápido: Como está construida sobre el motor JavaScript V8 de Google Chrome, la biblioteca es muy rápida en la ejecución de código.
- Un solo subproceso, pero altamente escalable: Node.js utiliza un modelo de un solo subproceso con bucle de eventos. El mecanismo de eventos ayuda al servidor a responder sin bloqueos y hace que el servidor sea altamente escalable en comparación con los servidores tradicionales que crean subprocesos limitados para manejar solicitudes.
- Sin almacenamiento en búfer: las aplicaciones Node.js nunca almacenan en búfer ningún dato. Estas aplicaciones simplemente generan los datos en trozos [14].

2.2.5. Web Crawler

Imagen 6: Representación Web Crawler

Apareció con un propósito militar para permitir la comunicación segura entre instituciones a distancia. En internet es de vital importancia el contenido a través de enlaces o hipervínculos. En 1998 una empresa decidió analizar y organizar los enlaces para que sea mucho más accesible para el usuario. Esa empresa fue Google y creo un ejecito de crawlers [15].

Un crawler, también conocido como rastreador o araña, es un programa que saca y analiza la información de las páginas web automáticamente. Su objetivo es perseguir todos los enlaces que encuentra en un sitio web, para así crear una red con muchas páginas web interconectadas entre sí [16].

Los crawlers se relacionan con las arañas ya que su trabajo es ir con sus "patas" explorando los enlaces que tiene el sitio web, recogiendo toda la información relevante y almacenando una copia en la base datos.

Funcionan de la siguiente manera:

1. El crawler visita una serie de URLs, les extrae la información y descarga su código HTML. Lo único que hacen es leer el código y hacen una copia del HTML que es lo que almacenaremos en la base de datos.
2. Los crawlers identifican los enlaces que contienen las URLs que ha visitado y los añaden a la lista para la próxima vez que se visiten.
3. Llega un momento en el que los crawlers dejan de buscar nuevos enlaces, porque hay infinidad de enlaces. Suele ocurrir sobre el nivel 3.
4. Una vez terminado el rastreo y almacenamiento, el algoritmo analizará el contenido para extraer solo aquello que nos es interesante.
5. Con la información se crea un índice.
6. Cada cierto tiempo los crawlers vuelven a analizar las mismas URLs por si ha habido cambios y así actualizar la información.

Con estas funciones podemos observar que los crawlers son excelentes herramientas para extraer información de las páginas web. De las cuales podemos conseguir únicamente los datos que nos interesa y guardarlos en nuestra base de datos [15].

Se suelen distinguir dos tipos de crawlers:

- Los de buscadores: Google, Bing, Yandex…
- Los de herramientas SEO como Screaming Frog, Ahrefs [17].

El primer crawler que se publicó fue RBSE que se basa en dos programas, el primero mantiene la base de datos relacional y el segundo, descarga las páginas web.

Google crawl, publicado en 1998, está basado en C++ y Python, este recorre internet extrayendo información de los dominios y analizando los datos son nuevos o ya estaban la última vez que pasó por allí. Si es nuevo, añade la información a la base de datos.

En la actualidad, hay multitud de crawlers que se utilizan para analizar y extraer distintas cosas [18].

2.2.6. DBeaver

Para la administración y control de la base de datos se ha utilizado la aplicación DBeaver.

Imagen 7: Logo DBeaver

DBeaver es herramienta de administración de bases de datos multiplataforma de código abierto y un software cliente de SQL. DBeaver se puede utilizar para acceder a cualquier base de datos o aplicación en la nube que tenga un controlador ODBC o JDBC, como Oracle, SQL Server, MySQL, Salesforce o MailChimp [19].

Con esta aplicación podemos manejar y controlar múltiples bases de datos, como trasladar datos de una a otra o hacer copias de ellas.

Se creó en 2010 y el objetivo era que fuese libre y de código abierto con una atractiva interfaz para el usuario. También debían incluir funciones que usan los desarrolladores de bases de datos frecuentemente. Poco a poco se convirtió en una herramienta que utilizaba mucha de la comunidad de código abierto.

Algunas de las características que tiene son:

- Ejecución de consultas SQL.
- Navegador de datos con gran cantidad de funciones.
- Multiplataforma (Linux, Windows, MacOS X, Solaris).
- Idioma (inglés, ruso, chino, alemán e italiano).
- Editar y navegar la estructura de base de datos.
- Exportación y migración de datos [20].

2.2.7. CSS

Imagen 8: Logo CSS

CSS (Cascading Style Sheets) es un lenguaje de diseño creado para dar aspecto y presentación a documentos escritos en código HTML. Se encarga de la apariencia de una página web. Con este lenguaje podemos controlar el color del texto, el tamaño, el estilo, las imágenes o los colores de fondo, es decir, todo aquello visual y que hace más atractivo la página web y no tan monótono [21].

Se trata de un lenguaje basado en reglas, los usuarios definen las reglas que van a aplicarse a elementos particulares o a grupos de elementos de la página web. Estos objetos que se van a presentar suelen estar designados por su tipo, clase, *Tag* o identificador, para que se apliquen las reglas correspondientes con ellos [22].

La versión actual es CSS3, la cual fue publicada en 2011, y ofrece la posibilidad de dar un aspecto más moderno a las páginas HTML.

3. Descripción del proyecto

En la descripción del proyecto se explica más en detalle lo comentado anteriormente. Para conocer más a fondo la herramienta y cuáles son las razones para realizar este proyecto.

3.1. Definición

El proyecto, como se ha comentado anteriormente, va a tratar sobre un extractor de noticias automatizado que se basará en un crawler el cual inspeccionará, rastreará y obtendrá la información que creamos que es relevante para una noticia. Esta información consta de 6 partes:

1) Título: Título de la noticia que vamos a obtener.

2) Resumen: Una pequeña entradilla sobre la noticia. Puede haber ocasiones en las que la web no tenga una entradilla en las noticias.

3) Imagen: Imagen que corresponde a la noticia en sí. Como con el resumen, hay veces que las noticias no vienen con una imagen asignada.

4) URL: Enlace a la noticia completa y particular de la cual vamos a obtener el texto.

5) HTML: HTML de la página web de la noticia.

6) Texto: Texto completo de la noticia que estamos extrayendo.

Al extraer todas estas partes de cada noticia, las noticias se almacenarán en la base de datos. La aplicación permitirá al usuario ver todas esas noticias y categorizarlas por los distintos temas que puede tratarse la noticia. Estas categorías también son modificables y se pueden crear tantas como el usuario quiera.

En esta herramienta se podrán añadir todas las fuentes que el usuario quiera, añadiendo un JSON que corresponda a esa página web. La herramienta leerá las fuentes, si todo es correcto, y podrá minar las noticias de esas fuentes.

3.2. Objetivos

Los objetivos de la aplicación son:

Extraer y evaluar noticias

Obviamente al ser un extractor de noticias, el objetivo principal es extraer todas las noticias posibles que hay en internet, más en particular en las fuentes que nosotros insertemos.

Una vez extraídas las noticias, otro objetivo de la aplicación es categorizar y evaluar estas noticias. Al evaluar la noticia, podemos ponerle las categorías a las cuales pertenece la temática de la noticia y ponerle un número que significará la relevancia que considere el usuario, como también cambiar el título y resumen si así lo ve oportuno.

También podemos visualizar las noticias individualmente para ver que los datos que hemos extraído son los correctos y no ha habido problemas. Pudiendo eliminarlas si ha ocurrido algún error.

Administrar fuentes

Al ser una herramienta que extrae de cualquier página web de internet, uno de los objetivos y funciones que más le pueden interesar a los usuarios que utilicen la herramienta es la posibilidad de añadir cualquier página web que queramos.

Podemos crear las fuentes indicando en que parte de la página web se encuentra la información que necesitamos para completar los datos de la noticia. Esta función de normal la suele hacer el administrador ya que suele ser el que más facilidad y conocimiento para añadir una fuente nueva, por lo que las funciones de crear solo tendrían permisos los usuarios con rol de administrador.

Asimismo, una vez creadas estas fuentes podremos editarlas porque pueden cambiar en un futuro y empezar a dar problemas, como también eliminarlas o ejecutarlas individualmente para comprobar que estas funcionan.

Ahorro de tiempo

Este es uno de los objetivos principales de la aplicación. Minimizar y optimizar el tiempo en la informática es vital para sacar mayor rendimiento a nuestra empresa.

El trabajo que hace la herramienta si se intentara hacer manualmente sería imposible ir página por página y noticia por noticia extrayendo la información que nos interesa, ya que esto nos llevaría muchísimo tiempo y no tendría sentido invertirlo en esta tarea.

Al ser una herramienta que extrae de cualquier página web de internet, uno de los objetivos y funciones que más le pueden interesar al usuario que utilice la herramienta es la posibilidad de añadir cualquier página web que queramos. Podemos configurarla indicando en que parte de la página web se encuentra la información necesaria.

También se podría intentar programar un Script que cogiese una lista de URLs e intentará extraer toda esta información, pero el tiempo de ejecución sería muy elevado y tener la máquina encendida tanto tiempo no saldría rentable.

Por lo que con nuestra aplicación podemos optimizar este tiempo y automatizar a la máquina para que la haga varias veces al día y consiga extraer cada vez que aparezcan nuevas noticias, y solo "perderíamos" el tiempo de introducir una nueva fuente. Podremos conseguir muchas noticias al día de manera automática, solo teniendo que entrar a la aplicación veremos la cantidad que se han extraído.

Obtención de datos

En la época en la que estamos, los datos son muy importantes, con ellos podemos hacer que una máquina aprenda y se automatice para hacer determinadas cosas, simplemente insertando datos que nos interesan y variando su salida para que poco a poco ella vaya aprendiendo.

Con esta herramienta esta tarea se soluciona y la hace mucho más sencillo. Podemos obtener muchos datos diariamente y de forma más o menos constante, ya que todos los días salen noticias nuevas.

Además, con esta herramienta podremos hacer una elección sobre aquellos datos que nos interesará obtener, por ejemplo, si nos interesan datos de determinado idioma o categoría. Por lo que conseguiremos solo datos que nos interesarán.

Y por último, todos esos datos que en internet están cada uno en una web y dispersos por internet los conseguimos tener reunidos todos en la base de datos y organizados para manejarlos cuando nos haga falta y solo los que nos hagan falta.

3.3. Alcance

Este proyecto no está planteado para que tenga un alcance mundial, ya que se ha planteado como una herramienta para el uso particular de una empresa. Por lo tanto, está muy enfocada a las necesidades de la empresa.

Esta empresa es la que lo utilizará y a la única que, de momento, se le dará acceso. Los usuarios serán los empleados de esta que diariamente la usarán y trabajarán con ella. También tendrán acceso los administradores para modificar y tratar algunos aspectos de la aplicación, como las fuentes o si hay algún error para que todo vaya bien.

Aunque desde el principio se planteó para que este proyecto fuese para una única empresa, al desarrollarla hemos visto una oportunidad para obtener datos de cualquier URL y en un futuro, podría llegar a venderse o hacerla *open Source*.

Por otro lado, también es posible que la empresa la utilice para también conseguir diariamente datos de distintas URLs, que como ya hemos comentado antes son bastantes valiosos ahora mismo.

3.4. Proceso software

En este apartado vamos a hablar de como se ha planteado el proyecto y que proceso hemos seguido. Es decir, la gestión que hemos llevado a cabo durante su ciclo de vida. Todos los proyectos deben entregar un producto definido en una fecha definida.

Para la gestión de un proyecto solemos ver 2 tipos de gestión: clásica/tradicional o ágil.

La gestión tradicional o clásica es una metodología establecida desde el principio y que tiene una secuencia fija que se realiza de forma consistente y en el orden que se define. Estas partes son: Planificación, ejecución, seguimiento y cierre. El enfoque pone atención en los procesos lineales, la documentación, la planificación por adelantado y la priorización. Con este método se plantea un tiempo y presupuesto variable.

La gestión ágil es un enfoque general utilizado sobre todo para desarrollo de software. Los requisitos no son claros desde el inicio y se van cambiando a medida que va avanzando y probando el producto. Este tipo de gestión se basa en el trabajo en equipo, las tareas a corto plazo, la colaboración y la flexibilidad para responder a los cambios lo más rápido posible [23].

Este proyecto, a diferencia de la gestión clásica, se basa en ciclos cortos, que son llamados *Sprints*, que busca objetivos parciales que es una manera de trabajar más flexible y adaptable a las necesidades del cliente y el proyecto [24].

La siguiente tabla muestra la diferencia entre los tipos de gestión que hemos hablado:

Características	Gestión Tradicional	Gestión Ágil
Escala de proyectos	Grandes	Pequeños
Requisitos	Definidos desde el inicio	Dinámicos
Implicación del cliente	Baja	Alta
Participación del cliente	Se involucran al principio, pero cuando empieza la ejecución dejan de hacerlo	Participan desde el principio y mientras se realiza el proyecto
Gestión de escalado	Cuando hay un problema, se pasa a los gerentes del proyecto	Cuando ocurre un problema, todo el equipo trabaja para resolverlo
Planificación	Se planifica desde el principio con detalle	Se planifica de Sprint en Sprint
Revisiones y aprobaciones	Las revisiones y las aprobaciones son constantes por parte de los líderes del proyecto	Las revisiones se realizan después de cada entrega

Tabla 1: Diferencias gestión tradicional y ágil

Una vez vistas las diferencias entre las 2 maneras de gestionar que tenemos y viendo las ventajas que tiene cada uno, se ha decidido que la manera más apropiada y que mejor se puede adaptar a nuestro proyecto sería la gestión ágil.

Esta gestión nos permitirá tener mayor flexibilidad, ya que durante el proyecto se pueden probar cosas diferentes a lo que desde el inicio estaba planteado. En esta metodología nos centraremos más en el producto que en seguir una estructura rígida.

Además, con esta manera los clientes pueden estar al tanto en todo momento mientras va avanzando el proyecto, por lo que pueden tener más feedback y conseguir una mejor retroalimentación. Después de cada entrega los clientes pueden ir viendo si eso les va convenciendo o puede que les surjan otras preferencias y se pueden comentar para que se vayan implementando [23].

En nuestro caso, el objetivo final está claro, pero se va trabajando en *Sprints* para que el cliente pueda ir viendo como avanza y que cosas le van llamando más la atención o quiere cambiar alguna cosa que no le guste del todo. Con esta metodología podemos ir a la par con los clientes y así consiguiendo al final un producto que a ellos les guste y no tener que cambiar cosas al final, si fuese el caso de una gestión tradicional.

4. Análisis

Después de hablar más detalladamente del propósito del proyecto y que utilidades y facilidades nos va a aportar, vamos a ver una especificación de requisitos de acuerdo con el estándar IEEE 830.

Primeramente, se va a escribir la Especificación de Requisitos Software (ERS) con respecto al estándar IEE 830, como también el propósito, ámbito del sistema, definiciones, acrónimos y abreviaturas.

Propósito

La intención del ERS es la de dejar claro cuál es el software que se debe utilizar. También recoge las necesidades tanto del cliente como del usuario a la hora de utilizar la herramienta y así evitar inconvenientes cuando se haga uso de esta.

Ámbito del sistema

Como ya se ha comentado a lo largo de la memoria, la herramienta se encargará de explorar y extraer noticias de las fuentes o URLs que nosotros le insertemos para su posterior evaluación y categorización. Como también crear categorías, editar noticias y eliminarlas y gestionar las fuentes de las cuales se extraen las noticias.

Definiciones, Acrónimos y Abreviaturas

Estos son los términos que se han utilizado en el ERS:

- DBeaver: Herramienta de base de datos multiplataforma gratuita para desarrolladores, administradores de bases de datos, analistas y todas las personas que necesitan trabajar con bases de datos [19].
- JSX: Una extensión de la sintaxis de JavaScript, se utiliza para describir la interfaz de usuario. Se utiliza con React [12].
- React: Es una librería de JavaScript declarativa, eficiente y flexible para construir interfaces de usuario [25].
- Web crawler: Son máquinas que buscan datos en internet, analizan el contenido lo guardan en bases de datos [26].

Perspectiva del producto

La herramienta que se ha desarrollado no depende de ningún software adicional, ni forma parte de un sistema mayor. Al tratarse de una aplicación web solo se deberá acceder al enlace que se proporcionará y obviamente, disponer de un navegador web para poder acceder a este. Para la utilización de la base de datos y el manejo de estos hará falta instalar el DBeaver para visualizar los datos que se vayan guardando.

Funciones del producto

El objetivo, ya definido de la aplicación será extraer y guardar noticias para su posterior evaluación, modificación y categorización. Las funciones que puede hacer son:

- Crear, borrar, ejecutar y editar fuentes de extracción.
- Crear, borrar y editar Tags de categorización.
- Consultar las extracciones que se han hecho.
- Consultar todas las noticias extraídas.
- Filtrar las noticias usando filtros.
- Ver, evaluar, editar, categorizar y borrar las noticias.
- Consultar resumen de la herramienta.

Características de los Usuarios

El uso de esta herramienta va dirigido únicamente a 2 tipos de usuarios: el cliente o usuario de la empresa y al administrador.

El cliente será el usuario básico que utilice la herramienta diariamente. Este podrá utilizar todas las funciones que se han dicho en el punto anterior, aunque para la creación de fuentes se necesitan conocimientos informáticos, por lo que de normal esa función la utilizará más comúnmente un administrador. Sin embargo, la idea es proporcionar los conocimientos básicos para que el cliente sea capaz de utilizar esa función. El resto de las funciones no necesitan conocimientos informáticos por lo que puede ser utilizadas sin ningún problema.

El administrador, como hemos dicho antes, tendrá acceso a todas las funciones, pero al no ser el usuario principal de la herramienta no utilizará las funciones excepto la función de crear fuentes. El administrador tiene acceso para crear, modificar y editar los usuarios que puedan entrar en la herramienta. Su rol de usuario sirve para hacer cambios en la herramienta, por lo que necesita tener conocimientos avanzados en informática y sobre todo de React y JSX que son principalmente la base de la herramienta. También es el que tiene acceso a la base de datos y puede ver todos los datos que consigue la herramienta.

Restricciones

Para que la aplicación funcione, al ser una aplicación web no es necesario que se cumplan muchas condiciones:

- Ordenador con cualquier sistema operativo (Linux, Windows, MacOS)
- Navegador web actualizado (Google Chrome, Firefox, Safari, Internet Explorer…).

Requisitos funcionales

Los requisitos funcionales, son aquellos requisitos que debe cumplir la herramienta para que se considere que es totalmente funcional para lo que se había propuesto la aplicación, en nuestro caso serían:

- La herramienta debe permitir el inicio de sesión de los usuarios creados.
- El usuario puede navegar y ver todas las pestañas creadas de la aplicación.
- El usuario puede listar, editar/valorar, filtrar y borrar las noticias extraídas.
- El usuario puede ver las noticias individualmente con todos sus atributos correspondientes.
- El sistema debe permitir crear, editar, listar y borrar las fuentes del crawler.
- El sistema debe ser capaz de extraer noticias de una única fuente que el usuario ejecute.
- El sistema debe ser capaz de automáticamente, en determinadas horas, extraer noticias de todas las fuentes creadas.
- El sistema debe permitir listar, activar y desactivar las fuentes del API.
- El sistema debe permitir crear, editar, listar y borrar las tags de categorización.
- El usuario puede visualizar una lista con todas las extracciones que ha realizado el sistema.
- El sistema debe permitir cerrar la sesión

Requisitos futuros

Después de realizarse la aplicación básica de extracción de noticias, una nueva funcionalidad muy interesante es la traducción automática de las noticias que no estén en el idioma que el usuario las necesite. Como también que la herramienta por si sola cree un resumen e inserte una imagen cuando las noticias extraídas no la contengan.

4.1. Casos de uso

En este apartado vamos a detallar los casos de uso de la herramienta relacionados con los requisitos funcionales de la aplicación. Todos los casos de uso van a tener una serie de características:

Identificación: Código que se usará como referencia, único en cada caso de uso.

Nombre: Nombre que se le pone al caso de uso.

Descripción: Explicación del caso de uso.

Entrada: Condición que se debe dar antes de ejecutar el caso de uso.

Proceso: Acciones que realiza el sistema.

Salida: Resultado de la ejecución del caso de uso.

Identificación	CU01
Nombre	Inicio de sesión
Descripción	El sistema pide un correo y una contraseña para utilizar la herramienta.
Entrada	Se requiere que el servidor este iniciado, se introduce un correo y una contraseña.
Proceso	El sistema comprueba los datos que se han insertado.
Salida	Si los datos son correctos, se empieza a utilizar la aplicación. Si son incorrectos, saltará un error y pedirá otra vez los datos.

Tabla 2: Caso de Uso CU01. Inicio de sesión

Identificación	CU02
Nombre	Listar noticias
Descripción	Obtenemos una lista con todas las noticas extraídas.
Entrada	Se debe iniciar sesión para listar.
Proceso	El sistema busca todas las noticas que hay en la base de datos.
Salida	Se ve la lista con todas las noticias.

Tabla 3: Caso de Uso CU02. Listar noticias

Identificación	CU03
Nombre	Ver Noticia
Descripción	Ver los detalles de una noticia en particular.
Entrada	La noticia debe estar en la lista de las noticias extraídas y se hace clic sobre el botón de “ver”.
Proceso	El sistema comprueba el id de la noticia seleccionada y consigue sus atributos.
Salida	Se abre una nueva pestaña mostrando todos los atributos de esta.

Tabla 4: Caso de Uso CU03. Ver noticia

Identificación	CU04
Nombre	Evaluar noticia
Descripción	Se evalúa y edita la información de una noticia.
Entrada	Seleccionamos con el botón de “editar” la noticia que se desea editar/evaluar.
Proceso	El sistema comprueba que se han introducido bien los nuevos datos y se aplican sobre la base de datos, si no son correctos salta un error y pide reintroducir esos datos.
Salida	La noticia estará modificada en la lista con los nuevos datos introducidos.

Tabla 5: Caso de Uso CU04. Evaluar noticia

Identificación	CU05
Nombre	Borrar noticia
Descripción	Se borra de la lista una noticia.
Entrada	Seleccionamos con el botón de “borrar” la noticia que se desea eliminar de la lista.
Proceso	El sistema comprueba el id de la noticia seleccionada y le cambia el atributo de *remove* a *true*. De la base de datos no se borra la noticia, sino que no se muestra por la lista.
Salida	La noticia seleccionada ya no aparecerá en el listado de noticias.

Tabla 6: Caso de Uso CU05. Borrar noticia

Identificación	CU06
Nombre	Filtrar noticias
Descripción	Se muestra una lista de noticias con unos determinados filtros.
Entrada	Seleccionamos los filtros que queremos para filtrar la lista.
Proceso	El sistema selecciona las noticias que cumplan los atributos seleccionados en los filtros.
Salida	Se obtiene una lista con solo las noticias que cumplen esos requisitos.

Tabla 7: Caso de Uso CU06Filtrar noticias

Identificación	CU07
Nombre	Listar fuentes/sources
Descripción	Obtenemos una lista con todas las fuentes que tiene nuestra base de datos.
Entrada	Se debe iniciar sesión para listar.
Proceso	El sistema busca todas las fuentes que hay en la base de datos.
Salida	Lista todas las fuentes.

Tabla 8: Caso de Uso CU07. Listar fuentes

Identificación	CU08
Nombre	Crear fuente
Descripción	Se crea y configura una nueva fuente en el sistema.
Entrada	Se introducen los datos necesarios desde la pestaña de fuentes para poder crearla.
Proceso	El sistema comprueba que los datos son correctos y crea una nueva fuente en la base de datos, si no son correctos salta un error y pide reintroducir esos datos.
Salida	La nueva fuente aparecerá en el listado de fuentes.

Tabla 9: Caso de Uso CU08. Crear fuente.

Identificación	CU09
Nombre	Editar fuente
Descripción	Se edita los atributos de una fuente que se ha creado previamente.
Entrada	Seleccionamos con el botón de “editar” la fuente que se desea editar.
Proceso	El sistema comprueba que se han introducido bien los nuevos datos y se aplican sobre la base de datos, si no son correctos salta un error y pide reintroducir esos datos.
Salida	La fuente estará modificada en la lista con los nuevos datos introducidos.

Tabla 10: Caso de Uso CU09. Editar fuente

Identificación	CU10
Nombre	Ejecutar fuente
Descripción	Se ejecuta una fuente en particular.
Entrada	Seleccionamos con el botón de “ejecutar” la fuente que se desea ejecutar.
Proceso	El sistema comprueba el id de la fuente y desde el servidor ejecuta su función para extraer las noticias.
Salida	Si la configuración de la fuente funciona, aparecerá en pantalla una ventana con los atributos de las noticias extraídas. En caso contrario, saldrá un mensaje de error y dejará de ejecutarse la fuente.

Tabla 11: Caso de Uso CU10. Ejecutar fuente

Identificación	CU11
Nombre	Borrar fuente
Descripción	Se borra de la lista una fuente.
Entrada	Seleccionamos con el botón de “borrar” la fuente que se desea eliminar de la lista.
Proceso	El sistema comprueba el id de la fuente seleccionada y le cambia el atributo de *remove* a *true*. De la base de datos no se borra la fuente, sino que se deja de mostrar por la lista.
Salida	La fuente seleccionada ya no aparecerá en el listado de fuentes.

Tabla 12: Caso de Uso CU11. Borrar fuente

Identificación	CU12
Nombre	Listar fuentes/sources API
Descripción	Obtenemos una lista con todas las fuentes del API que tiene nuestra base de datos.
Entrada	Se debe iniciar sesión para listar.
Proceso	El sistema busca todas las fuentes que hay en la base de datos que pertenezcan a la API.
Salida	Lista todas las fuentes de la API.

Tabla 13: Caso de Uso CU12Listar fuentes API

Identificación	CU13
Nombre	Activar fuente API
Descripción	Activa una fuente de la lista de la API.
Entrada	Seleccionamos el botón de *active* en la lista de fuentes de la API.
Proceso	El sistema cambia el valor de *active* a *true* en la base de datos de la fuente seleccionada.
Salida	La fuente seleccionada ahora aparecerá activada en la lista.

Tabla 14: Caso de Uso CU13. Activar fuente API

Identificación	CU14
Nombre	Listar Tags
Descripción	Obtenemos una lista con todas las Tags que tiene nuestra base de datos.
Entrada	Se debe iniciar sesión para listar.
Proceso	El sistema busca todas las Tags que hay en la base de datos.
Salida	Lista todas las Tags.

Tabla 15: Caso de Uso CU14. Listar Tags

Identificación	CU15
Nombre	Borrar Tags
Descripción	Se borra de la lista una Tag.
Entrada	Seleccionamos con el botón de “borrar” la Tag que se desea eliminar de la lista.
Proceso	El sistema comprueba el id de la Tag seleccionada y le cambia el atributo de *remove* a *true*. De la base de datos no se borra la Tag, sino que se deja de mostrar por la lista.
Salida	La Tag seleccionada ya no aparecerá en el listado de fuentes.

Tabla 16: Caso de Uso CU15. Borrar Tag

Identificación	CU16
Nombre	Editar Tag
Descripción	Se edita los atributos de un Tag que se ha creado previamente.
Entrada	Seleccionamos con el botón de “editar” la Tag que se desea editar.
Proceso	El sistema comprueba que se han introducido bien los nuevos datos y se aplican sobre la base de datos, si no son correctos salta un error y pide reintroducir esos datos.
Salida	La Tag estará modificada en la lista con los nuevos datos introducidos.

Tabla 17: Caso de Uso CU16. Editar Tag

Identificación	CU17
Nombre	Crear Tag
Descripción	Se crea y configura una nueva Tag en el sistema.
Entrada	Se introducen los datos necesarios desde la pestaña de Tags para poder crearla.
Proceso	El sistema comprueba que los datos son correctos y crea una nueva Tag en la base de datos, si no son correctos salta un error y pide reintroducir esos datos.
Salida	La nueva Tag aparecerá en el listado de fuentes.

Tabla 18: Caso de Uso CU17. Crear Tag

Identificación	CU18
Nombre	Listar Extracciones
Descripción	Obtenemos una lista con todas las ejecuciones y extracciones que se han hecho.
Entrada	Se debe iniciar sesión para listar.
Proceso	El sistema busca todos los registros de extracciones en la base de datos.
Salida	Lista con todos los registros de extracciones y ejecuciones realizadas.

Tabla 19: Caso de Uso CU18. Listar extracciones

Identificación	CU19
Nombre	Visualizar Resumen
Descripción	Observamos una pantalla con un pequeño resumen de la herramienta.
Entrada	Se debe iniciar sesión para visualizar.
Proceso	El sistema busca y cuenta todos los datos necesarios para el resumen.
Salida	Pantalla con números de las noticias extraídas, fuentes y tags creados. Y una gráfica con la cantidad de noticias con cada tag e idioma. Mostrando las últimas 5 noticias obtenidas.

Tabla 20: Caso de Uso CU19. Visualizar resumen

Identificación	CU20
Nombre	Cerrar sesión
Descripción	Cerrar la sesión una vez se termine de utilizar la herramienta.
Entrada	El usuario debe de haber iniciado sesión correctamente
Proceso	El sistema cierra la sesión actual iniciada.
Salida	Se muestra la pantalla inicial para volver a iniciar sesión.

Tabla 21: Caso de Uso CU20. Cerrar sesión

Un diagrama de caso de uso es un tipo de diagrama UML (Unified Modelling Language) en el que se representa los distintos procesos, usos e interacciones de los diferentes usuarios que existen en el sistema [27]. Para la representación de este diagrama vamos a utilizar la página web, Ludichart [28]. La figura 9 muestra los casos de uso que tiene nuestra herramienta y que usuario interactúa con cada uno de ellos.

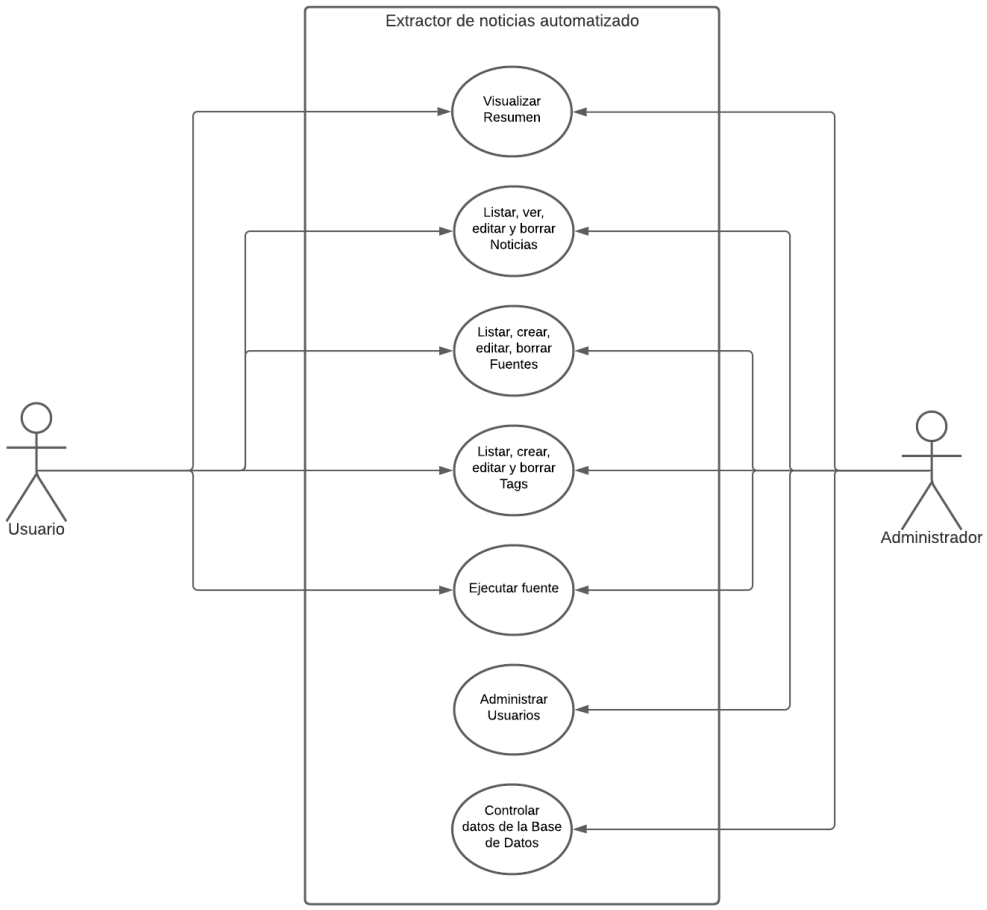

Figura 9: Diagrama Casos de uso

4.2. Diagrama de clases

Los diagramas de clases son uno de los tipos de diagramas más útiles en UML, ya que trazan claramente la estructura de un sistema concreto al modelar sus clases, atributos, operaciones y relaciones entre objetos.

La figura 10 representa el diagrama de clases de nuestro sistema de una manera simplificada, el inicio de lo que será el diagrama de clases completo.

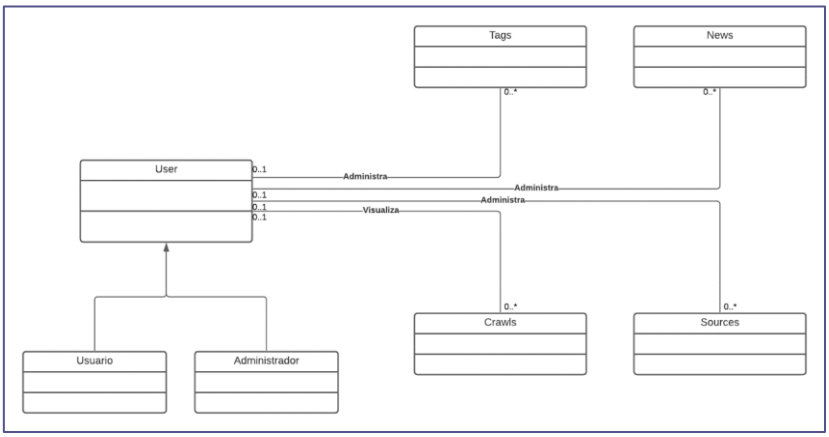

Figura 10: Diagrama de clases simplificado

4.3. Ejemplo de uso concreto

El cliente desea añadir una nueva fuente o URL desde la cual poder obtener las noticias de esa página web y poder evaluarlas para ver si son relevantes usando la funcionalidad de categorizarlas y editar los campos que ellos vean oportunos. Para ello hará falta un administrador para la configuración de esta nueva fuente y un usuario que se encargará de evaluar las noticias. La herramienta requiere de las siguientes funcionalidades:

1. Creación de la nueva fuente: El administrador deberá configurar la nueva fuente entrando a esta URL y escogiendo los elementos relevantes para que la herramienta únicamente extraiga aquellos atributos que nos interesan.

2. Ejecución de la fuente: El administrador, una vez configurada la fuente, la ejecutará para que, si esta es correcta, obtenga todas las noticias de esa página web y la guarde en la base de datos.

3. Evaluación de noticias: El usuario accederá a la lista de noticias en las cuales se encontrarán las nuevas noticias que hemos obtenido. Irá una por una evaluándolas, viendo si le interesan o si hay algún error en el título o resumen para poder corregirlo. Además, las categorizará dependiendo de a qué tema pertenece cada noticia.

4. Actualización de las noticias: Las noticias que ya están evaluadas se actualizarán en la base datos con los nuevos atributos que se han insertado.

5. Diseño

El diseño de la herramienta se puede dividir en 3 capas: presentación, persistencia y lógica.

5.1. Maquetas

Para el diseño de la aplicación a nivel interfaz, se han realizado varias maquetas con algunas de las ideas o conceptos que se querían incluir en las páginas o ventanas de la herramienta.

Vamos a documentar algunas de las mockups que se han realizado de manera manual y que hemos seguido como guía para la realización de la aplicación.

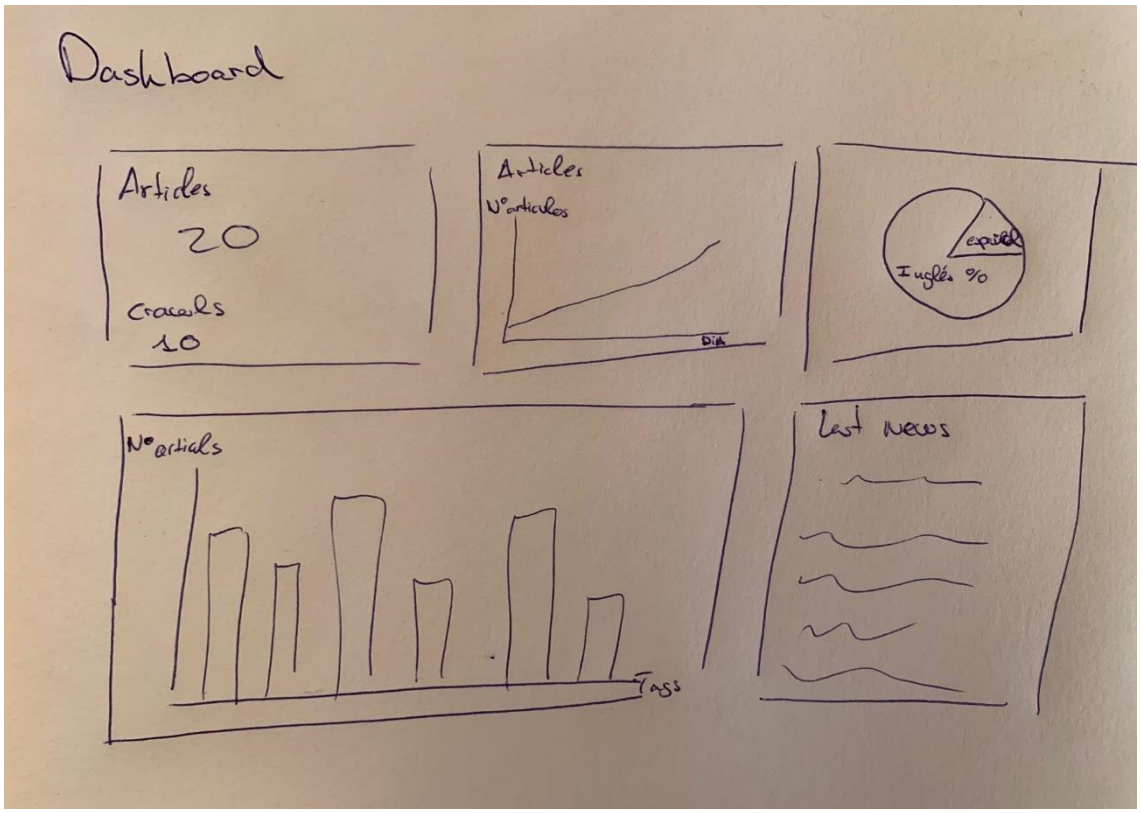

Figura 11: Mockup Dashborad

La figura 11 representa la maqueta de lo que es la pantalla principal de la aplicación, la cual está asignada para mostrar un resumen de la herramienta y los datos que hemos conseguido de ella.

Se pueden apreciar 5 componentes. En los cuales tendremos un contador de los artículos que hemos extraído, cuantos hemos conseguido cada día, una gráfica con porcentajes de los lenguajes de las noticias, una gráfica de barras con los tags asignados a las noticias y por último, las ultimas noticias que han sido extraídas.

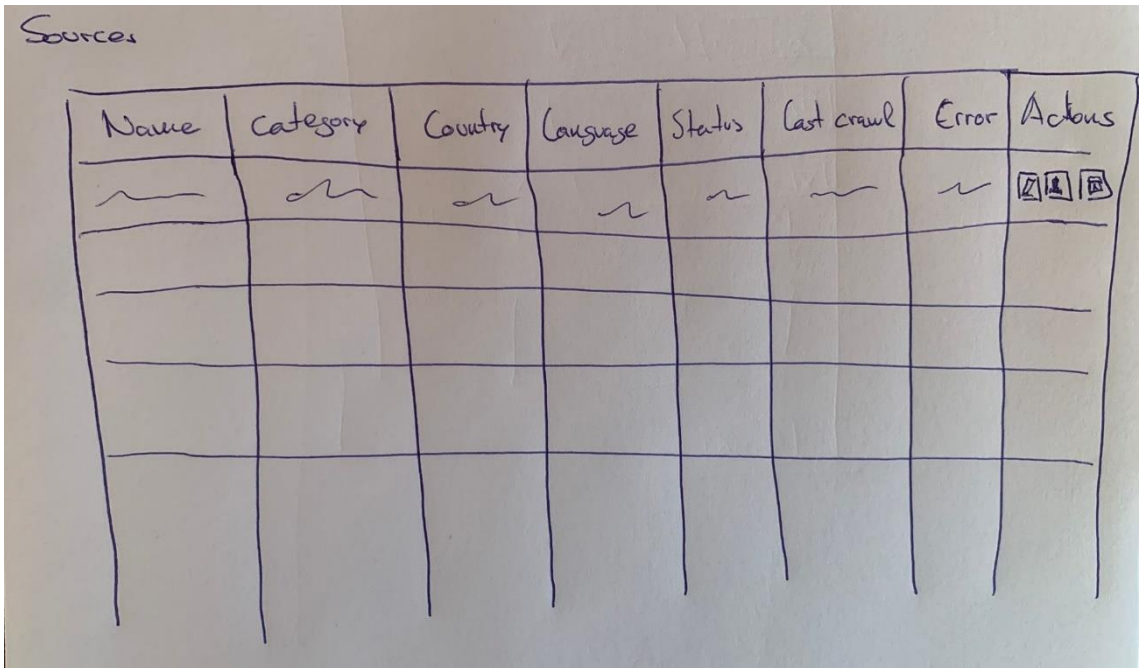

Figura 12: Mockup lista fuentes

En la figura 12 podemos observar la pantalla que contiene la lista de fuentes. Esta constará de una tabla con todos los atributos divididos por columnas. La última columna representa con tres botones las acciones que podemos hacer con cada fuente: editar, ejecutar y borrar.

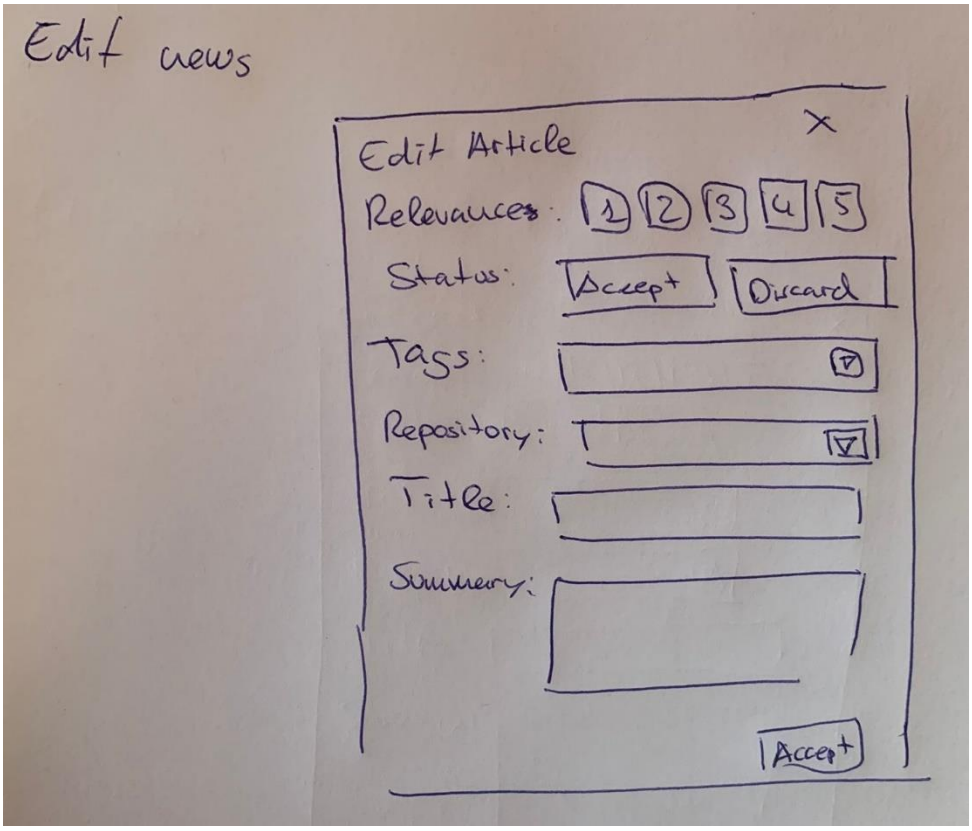

Figura 13: Mockup editar noticia

La figura 13 muestra la maqueta de la ventana para editar/evaluar una noticia que ha sido extraída. Esta cuenta con botones para asignarle una relevancia y para aceptar o descartar la noticia. También podemos ver dos desplegables para seleccionar las Tags y los repositorios de las noticias. Y para finalizar, dos campos de texto para escribir su nuevo título y resumen si el usuario vio necesario cambiarlos.

5.2. Diagrama de clases

Los diagramas de clases son diagramas estructurales que recogen las clases de los objetos y las asociaciones que tienen entre ellos. Representan la estructura y la relación entre los distintos objetos que hay en el sistema. La estructura consta de atributos, clases y métodos [29].

Para realizar la representación del diagrama de clases de nuestro sistema hemos seguido el estándar UML. Aquí se encuentra más detallada del visto anteriormente en el análisis.

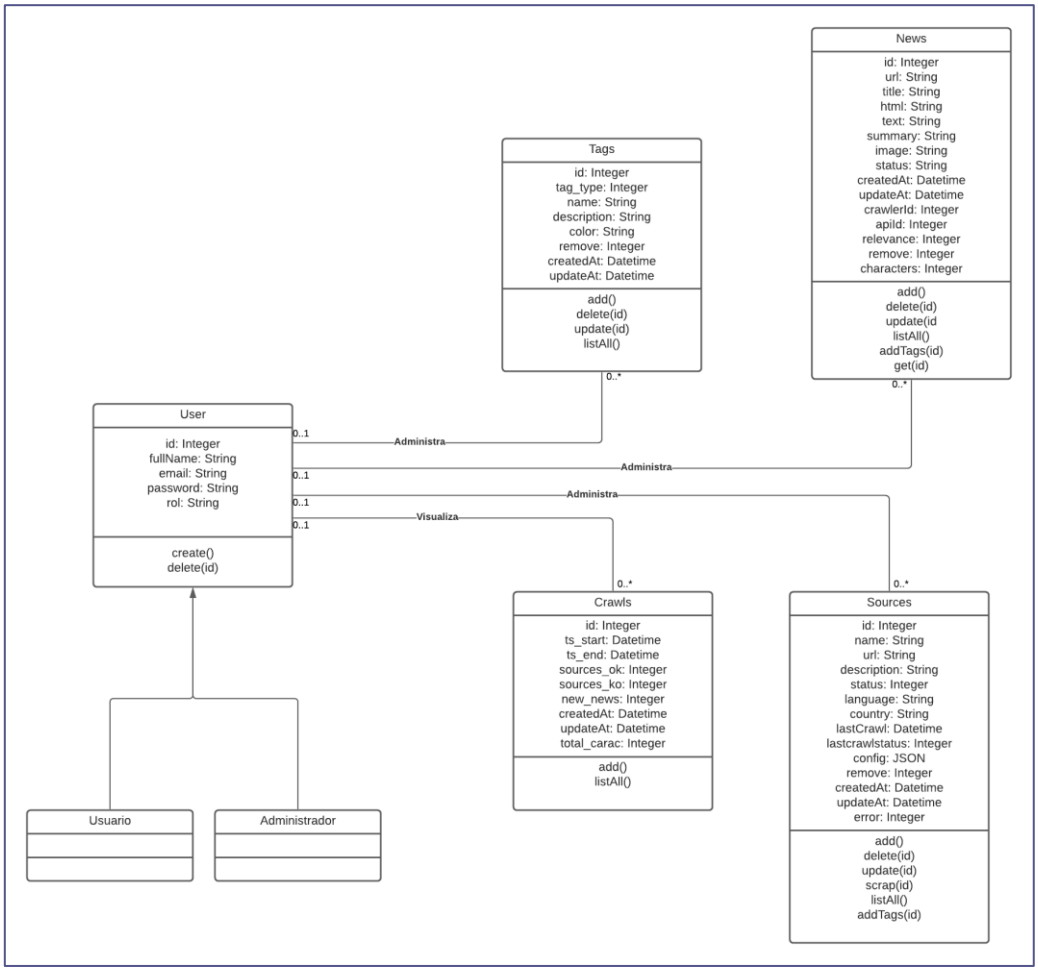

Figura 14: Diagrama de clases detallado

5.3. Base de datos

La base de datos es un elemento muy importante en el diseño de nuestra herramienta. Necesitamos que esté bien organizada para que la información que tenemos se guarde correctamente y de una forma clara.

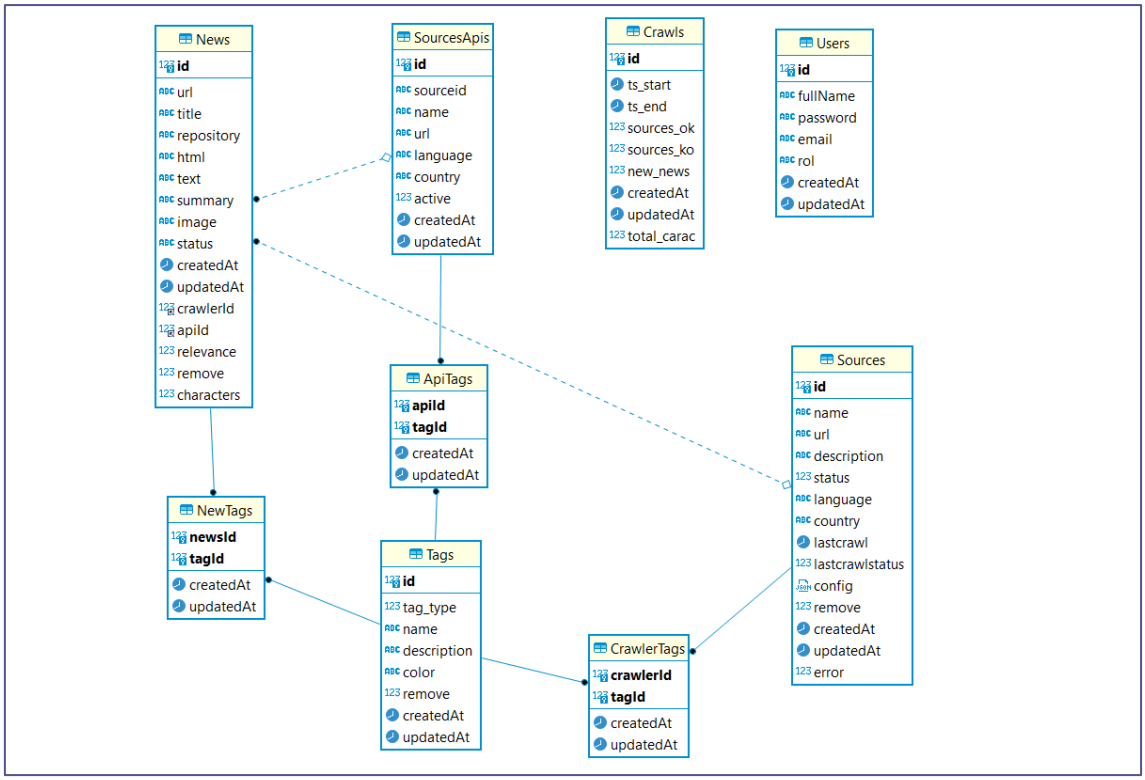

Figura 15: Representación base de datos

La Figura 15 representa el modelo relacional de nuestra base de datos.

Un modelo relacional muestra la organización del almacenamiento de datos en tablas compuesta por filas (o tuplas) y columnas (o campos). El usuario lo comprende con poco conocimiento. También están representadas las relaciones que hay entre las distintas tablas [30].

Nuestra base de datos contiene las siguientes tablas:

News: Tabla sobre la que se guardan las noticias, contiene toda la información que queremos extraer de las noticias.

SourcesApi: Tabla donde están guardados las fuentes de una API predeterminada y que contiene muchas fuentes de Internet servibles para extraer noticias.

Sources: Tabla que contiene las fuentes creadas por el usuario, y toda la configuración y atributos que necesitamos para que la extracción de esta fuente sea útil.

Tags: Tabla que guarda todas las Tags que se han creado por los usuarios, contiene todas las características que necesitamos.

CrawlerTags: Tabla que funciona como relación entre las fuentes creadas por los usuarios y los tags que les asignamos, su clave es una tupla con el id de la fuente y el id de la Tag que le asignamos.

ApiTags: Esta tabla es semejante a la anterior, pero relaciona las fuentes de la API con las Tags.

NewsTags: Esta tabla funciona como las dos tablas anteriores, pero en este caso son las noticias las que están relacionadas con las Tags a las que pertenece.

Crawls: Tabla que representa las extracciones que hace el sistema y guarda información de cada extracción que hacemos.

Users: Tabla que contiene todos los usuarios y sus acreditaciones para poder utilizar la herramienta.

5.4. Esquema relacional asociado

Ahora hablaremos en detalle de cada una de las tablas a través de un esquema relacional asociado al modelo que hemos visto en el apartado anterior.

News (id: int, url: string, title: string, repository: int, html: string, text: string, summary: string, image: string, status: string, createdAt: date, updatedAt: date, crawlerId: int, apiId: int, relevance: int, remove: int, characters: int)

CP: {id}

VNN: {url, title, html, text, status, createdAt, updatedAt, relevance, remove}

VNN: {crawlerId, apiId}

CAj: {crawlerId} -> Sources {id}

CAj: {apiId} -> SourcesApi {id}

SourcesApi (id: int, sourceid: string, name: string, url: string, language: string, country: string, active: int, createdAt: date, updatedAt: date)

CP: {id}

VNN: {sourceid, name, url, language, country, active, createdAt, updatedAt}

Sources (id: int, name: string, url: string, description: string, status: int, language: string, country: string, lastcrawl: date, lastcrawlstatus: int, config: JSON, remove: int createdAt: date, updatedAt: date, error: int)

CP: {id}

VNN: {name, description, status, language, country, lastcrawlstatus, config, remove, createdAt, updatedAt}

Tags (id: int, tag_type: int, name: string, description: string, color: string, remove: int, createdAt: date, updatedAt: date)

CP: {id}

VNN: {tag_type, name, description, color, remove, createdAt, updatedAt}

CrawlerTags (crawlerId: int, tagId: int, createdAt: date, updatedAt: date)

CP: {crawlerId, tagId}

VNN: {createdAt, updatedAt}

CAj: {crawlerId} -> Sources {id}

CAj: {tagId} -> Tags {id}

ApiTags (apiId: int, tagId: int, createdAt: date, updatedAt: date)

CP: {apiId, tagId}

VNN: {createdAt, updatedAt}

CAj: {apiId} -> SourcesApis {id}

CAj: {tagId} -> Tags {id}

NewTags (newsId: int, tagId: int, createdAt: date, updatedAt: date)

CP: {newsId, tagId}

VNN: {createdAt, updatedAt}

CAj: {newsId} -> News {id}

CAj: {tagId} -> Tags {id}

Crawls (id: int, ts_start: date, ts_end: date, sources_ok: int, sources_ko: int, new_news: int, createdAt: date, updatedAt: date, total_carac: int)

CP: {id}

VNN: {ts_start, ts_end, sources_ok, sources_ko, new_news, createdAt, updatedAt}

Users (id: int, fullName: string, password: string, email: string, rol: string, createdAt: date, updatedAt: date)

CP: {id}

VNN: {fullName, password, email, rol, createdAt, updatedAt}

Uni: {email}

5.5. Diagrama de alcance

Estos diagramas corresponden a la capa de negocio del diseño de la herramienta. En estos diagramas vamos a hablar de las funciones que pueden hacer los tipos de usuarios que tenemos en la aplicación.

En las figuras se muestra las acciones que pueden realizar y cuales deben hacer antes para llegar a algunas acciones.

En primer lugar, veremos el diagrama de caso de uso del usuario (ver Figura 16) el cual necesita iniciar sesión para entrar en la aplicación y luego accede a todas las funciones de la aplicación.

Y, en segundo lugar, tenemos al administrador (Figura 17), el cual tiene las mismas funciones que el usuario, únicamente añadiéndole la administración de la base de datos, a la cual el usuario normal no tiene acceso. Desde el manejo de la base de datos puede añadir o eliminar usuarios y sus permisos. También controla todos los datos que se obtienen desde la herramienta.

Como hemos comentado antes, en ocasiones para hacer una acción debe realizar otras. En estos diagramas se puede observar que para realizar la acción “Ver Noticia”, anteriormente se ha tenido que hacer la función “Listar Noticias” y también para realizar esta función el usuario debe realizar “Inicio de sesión”.

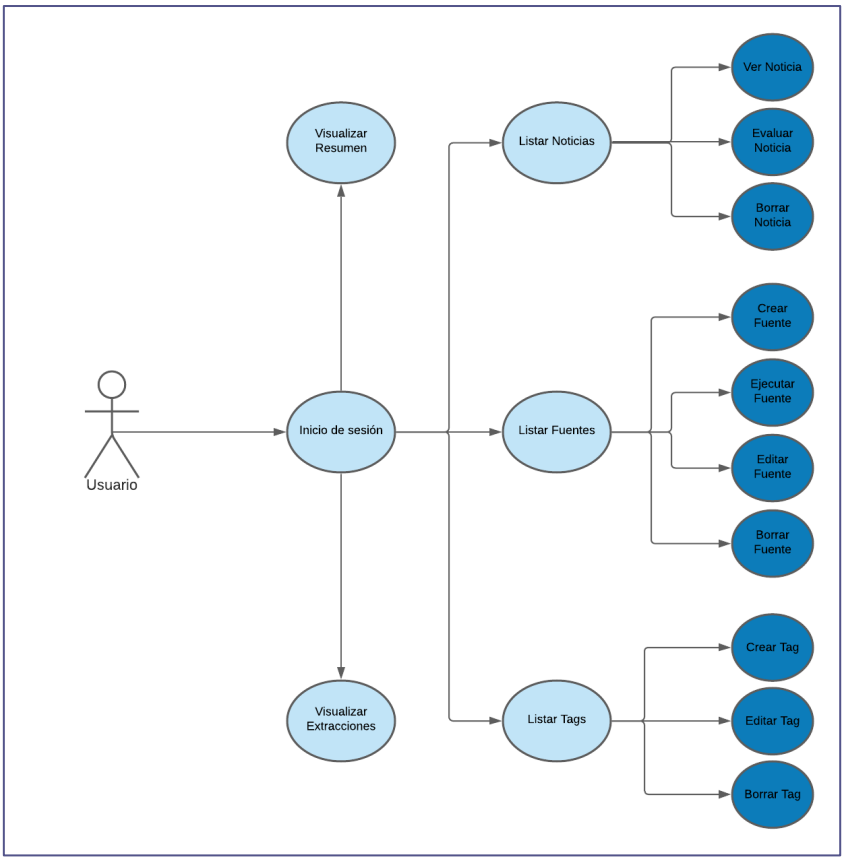

Figura 16: Diagrama Usuario

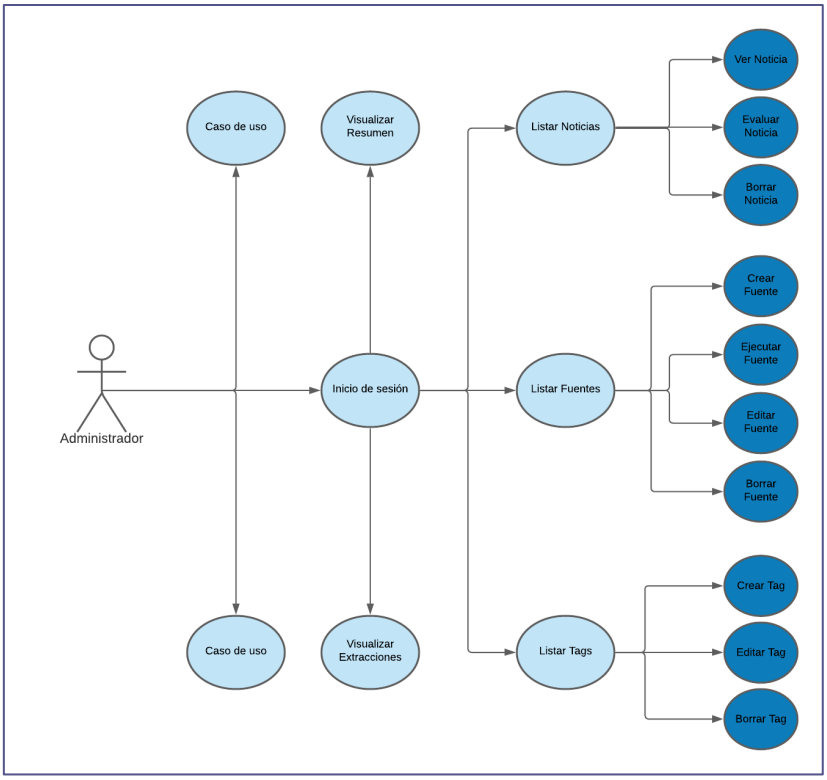

Figura 17: Diagrama administrador

6. Desarrollo

En este apartado veremos cómo se ha llevado a cabo el desarrollo de la aplicación, para ello primero vamos a ver que tecnologías, herramientas y lenguajes hemos utilizado para el proceso. Y luego analizaremos la estructura entera del código de la herramienta.

6.1. Tecnologías usadas

En el apartado 2.2. hemos descrito todas las tecnologías que nos han servido para el desarrollo web. En esta sección, explicaremos en qué se han utilizado cada una de estas tecnologías.

Visual Studio Code

Es el editor que hemos utilizado para crea y modificar el código de una forma más práctica. En Visual Studio Code hemos configurado tanto la parte del Cliente como la parte del Servidor.

Nuestros ficheros están escritos en varios lenguajes de programación, el principal, JSX, que es el propio de React, y también CSS y JSON que nos complementan la configuración de la aplicación.

DBeaver

Es la aplicación que hemos utilizado para la gestión de nuestra base de datos, donde podemos observar todos los datos que obtenemos y gestionarlos a nuestro gusto.

Esta herramienta nos da mucha facilidad para visualizar cualquier dato que nos interese. Para consultas dentro de la aplicación utilizamos SQL como lenguaje.

Navegador web

La visualización de la herramienta requiere de un navegador que lo ejecuta. En nuestro caso hemos utilizado Google Chrome, aunque también es compatible con cualquier otro como Firefox o Microsoft Edge.

Web crawler

El Web crawler es la tecnología más importante de nuestra herramienta, la cual caracteriza nuestra aplicación porque es la que consigue que obtengamos todos los datos que queremos de cada noticia. Está programado en JavaScript.

6.2. Estructura de la herramienta

En este apartado detallaremos la estructura que se ha realizado a nivel de código del proyecto. El código de esta herramienta se divide entre Cliente y Servidor. Cada uno de estos se divide en carpetas. (Ver Figura 18)

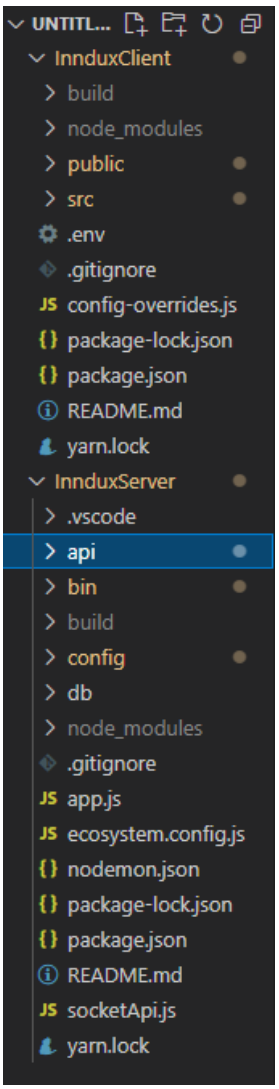

Figura 18: Estructura de todo el código

Como se puede observar en la Figura 18 se distinguen dos carpetas “InnduxClient” y “InnduxServer” que representa el front-end (Cliente) y la otra el back-end (Servidor).

Cliente

La parte del cliente corresponde al front-end de la herramienta, la parte que el cliente ve cuando utiliza la aplicación. El front-end es la parte visible de la herramienta con la cual el usuario interactúa [31].

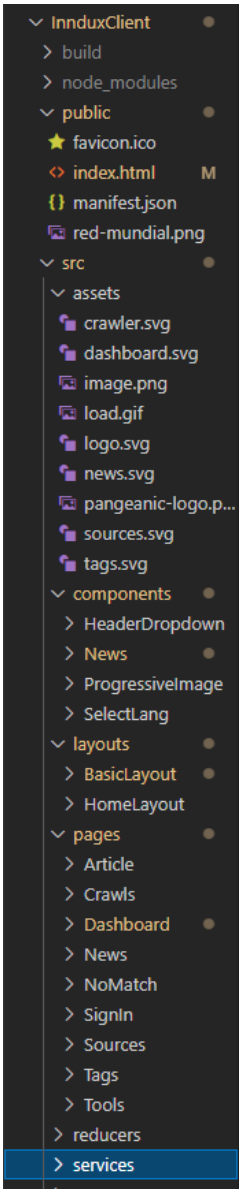

Figura 19: Estructura del código del Cliente

En la Figura 19 se puede observar la estructura completa del Cliente que está dividida en las siguientes carpetas:

- Build: en esta carpeta se genera nuestro componente minificado y transpilado a ES5, el cual usarán las aplicaciones cuando lo instalen via NPM.
- Node_modules: aquí es donde se instalan las dependencias de nuestro componente.
- Public: En esta carpeta están los recursos que son accesibles de forma pública desde el navegador web del cliente, sin necesidad de estar autenticado o registrado. Aquí se encuentran las imágenes estáticas que se cargan en la web y el HTML que utiliza el código JavaScript como plantilla.
- Scr: en esta carpeta se encuentra todo el código fuente que se ejecuta. Se distribuye en varias subcarpetas:
    - Assets: Contiene las imágenes que se utilizan en la aplicación.
    - Components: Contiene componentes específicos que se encuentran en varias pantallas de la aplicación.
    - Layouts: Contiene los componentes de la pantalla de inicio y el layout principal de la herramienta.
    - Pages: Se subdivide en carpetas, una por cada pestaña que hay en la aplicación
- Resto de archivos: Son archivos de configuración de los diferentes frameworks que se ejecutan en el sistema, el que tiene más importancia es el package.json, ya que es el encargado de recopilar todos los paquetes y las dependencias de NPM.

Una aplicación en React puede ser una aplicación que tenga una sola página, es decir, que hay un único archivo JavaScript que se encarga de la lógica de las interfaces del cliente, o también puede ser una aplicación de múltiples páginas que se dividen los apartados de la página en diferentes archivos y cada uno gestiona su lógica.

En esta herramienta se ha utilizado la aplicación con solo una página ya que tiene beneficios para el sistema:

-Mejor **velocidad**, debido a que solo es una aplicación no se requiere recargar y actualizar cada vez que cambias de un apartado a otro y tener que solicitar otro archivo JavaScript por lo que mejora significativamente la velocidad de uso del sistema.

-Mejor funcionamiento en **dispositivos móviles** ya que en los móviles es muy importante el tiempo de carga y que la aplicación esté lo más optimizada posible.

El funcionamiento de este tipo de aplicaciones de una sola página se basa en un archivo HTML que carga un archivo JavaScript con el código compilado [32].

Una vez cargado el archivo compilado, se carga el código que hay en index.js, el cual contiene la librería Router React que es la que resuelve las rutas de la aplicación web. Cuando se resuelve la ruta, se cargan los componentes asignados a la misma que contienen la lógica y la interfaz que se ha de mostrar al usuario [33].

La Figura 20 se muestra el uso de esta librería, donde están todas las rutas de la herramienta y donde se cargarán los componentes correspondientes para que el cliente vea la interfaz tal y como la hemos creado.

Cada componente está formado por diferentes funciones y variables. La función más importante es la que tiene el mismo nombre que el componente y se encarga de devolver los datos que se requieren.

```
ReactDOM.render(
    <Provider store={store}>
        <ConnectedRouter history={history}>
            <IntlProvider locale="en" messages={messages['en']}>
                <ConfigProvider locale={enUS}>
                    <Router>
                        <div>
                            <Route exact path="/signin" component={SignIn} />
                            <Route exact path="/admin" render={() => (
                                <Redirect to="/admin/dashboard" />
                            )} />
                            <Route exact path="/" render={() => (
                                <Redirect to="/signin" />
                            )} />
                            <Route exact path="/admin/dashboard" component={Dashboard} />
                            <Route exact path="/admin/news" component={News} />
                            <Route exact path="/admin/news/article/:id" component={Article}
/>
                            <Route exact path="/admin/sources" component={Sources} />
                            <Route exact path="/admin/tags" component={Tags} />
                            <Route exact path="/admin/crawls" component={Crawls} />
                            <Route exact path="/admin/tools" component={Tools} />


                            {/* <Route component={NoMatch} /> */}
                        </div>
                    </Router>
                </ConfigProvider>
            </IntlProvider>
        </ConnectedRouter>
    </Provider>,
    document.getElementById("root")
);
```

Figura 20: Uso de la librería Cliente

Servidor

La parte del servidor corresponde al back-end de la herramienta, la parte que el cliente no ve que utiliza la aplicación. El back-end es la parte del desarrollo web que se encarga de la lógica de una página funcione, son acciones que pasan dentro de la web pero que no se pueden ver [34]. En el back-end también se administra la comunicación con la base de datos y servidores.

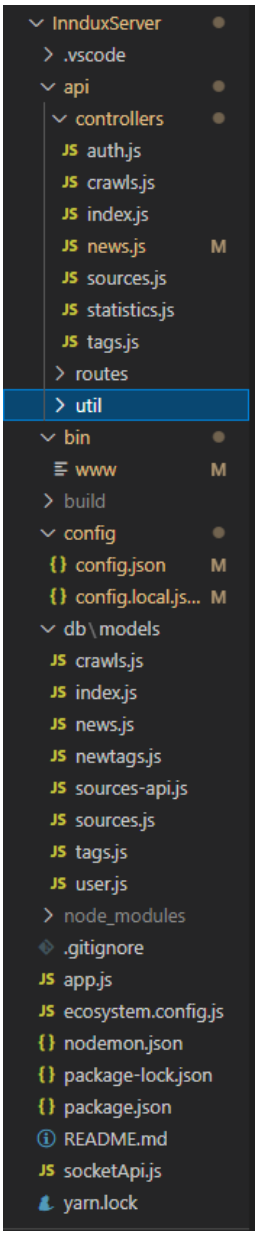

Figura 21: Estructura del código del Servidor

La Figura 21 muestra la estructura completa del Servidor, el cual está dividido en las siguientes carpetas:

- .vscode: Contiene un archivo de configuración que se crea al elegir un entorno para nuestro servidor.
- api: Archivos JavaScript que sirven para la configuración del servidor de las pestañas que hay conteniendo funciones y comprobaciones con la base de datos. Se divide en carpetas:
    - controllers: Configuración de funciones para cada pantalla.
    - routers: Configuración de todas las rutas que tiene la herramienta.
    - util: configuración del crawler.
- bin: Configuración del puerto donde se ejecutará la aplicación.
- build: Al igual que en el cliente, en esta carpeta se genera nuestro componente.
- config: Contiene archivos JSON para la configuración y conexión con la base de datos.
- bd: Configuración de cada tabla de la base de datos para obtener los atributos con su tipo de cada tabla.
- node_modules: Contiene las dependencias del Servidor.
- Resto de archivos: Como en el Cliente, es configuración de los diferentes frameworks, y el importante es el package.json.

En esta aplicación web el trabajo de la parte del servidor es la comunicación con el servidor, la conexión con la base de datos y gestión de los datos. Además, es la parte que trabaja el crawler, el que lo ejecuta y el que extrae las noticias y las funciones que corresponden con la extracción.

6.2.1. Crawler

El crawler es el corazón de la herramienta, es el que consigue a través de sus funciones obtener todos los datos que queremos para nuestra herramienta. Analiza todas las webs y extrae las noticias de las páginas para luego guardarlas en nuestra base de datos.

La configuración del crawler se basa en un código de JavaScript en el cual se definen las funciones para que pueda conseguir únicamente los datos que nos interesan extraer.

El scraping que realiza nuestro crawler se basa en dos fases o niveles, y cada nivel cuenta con una función para tomar los datos que nos interesan en cada nivel.

Primer nivel

En este primer nivel el crawler accede a la URL de la página web, en la que se encuentran todas las noticias que queremos extraer. Con la función de este nivel tomaremos la primera parte de la información de la noticia (Title, summary, image y URL de la noticia en detalle).

La Figura 22 contiene un trozo de código de la función que corresponde al primer nivel y donde se puede observar que con el selector de "root" y más tarde el selector "url" conseguimos la URL de las noticias individuales. Existe un condicional por si la URL es una dirección incompleta, ya que es común encontrarse eso en algunas páginas web. Para solucionar esto le añadimos una "base" que está escrita en la configuración JSON para que se complete.

```
exports.webScrapingFirstLvl = (source) => {
  return new Promise(function (resolve) {
      const c = new Crawler({
        maxConnections: 10,
        // This will be called for each crawled page
        callback: (error, res, done) => {
          if (error) {
            console.error(error)
            resolve([])
          } else {
            var $ = res.$
            const articles = []

            if (res.options.article.root) {
              let rootSelector = $(`${res.options.article.root}`)
              for (let i = 0; i < rootSelector.length; i++) {

                let url = $(rootSelector)
                  .find(`${res.options.article.url}`)
                  .eq(i)
                  .attr('href')

                //If the url doesn't contain http
                if(url){

                  if (res.options.article.base && !url.includes('http')
) {

                    url = res.options.article.base + '' + url
                  }
                }
```

Figura 22: Fragmento función primer nivel crawler

Segundo nivel

Este nivel corresponde a la segunda parte del scraping, cuando accedemos a la URL obtenida en el primer nivel y observamos la noticia completa. Con la función del segundo nivel tomaremos la información particular de la noticia, es decir, el HTML de la de la página web y el texto que corresponde al cuerpo de la noticia completa.

La Figura 23 contiene un trozo del código de la función donde se extraen los datos que deseamos extraer. En este trozo del código podemos observar que obtenemos el HTML y el Texto de la noticia.

```
exports.webScrapingSecondLvl = (article) => {
  return new Promise(function (resolve, reject) {

    const c = new Crawler({
      maxConnections: 10,
      // This will be called for each crawled page
      callback: (error, res, done) => {
        if (error) {
          console.error(error)
          resolve({ html: '', text: '' })
        } else {

            for (let i = 0; i < res.options.base.length; i++) {
              var $ = res.$

              try{
                let html = $(`${res.options.base[i]}`).html()
                let text = $(`${res.options.base[i]}`).text()
```

Figura 23: Fragmento función segundo nivel crawler

Una vez ejecutadas con éxito las dos funciones de scraping en los dos niveles, el servidor es el encargado de guardar todos los datos extraídos en la base de datos. Esto lo consigue a través de una función con la cual crea una nueva noticia si esta no existe ya en la base de datos. La comprobación de que no sea repetida se hace utilizando la URL ya que cada noticia tiene una propia y no cambia. Si la noticia no existe, el sistema crea una nueva noticia en la base de datos con los datos que acabamos de extraer en los atributos correspondientes.

El código de la Figura 24 corresponde a la comprobación y creación de las noticias en nuestra base de datos. Como se puede observar, primero busca con la función *findAll()* las noticias que tengan la misma URL que la noticia que vamos a insertar y si no encuentra ninguna la inserta creando una nueva noticia con la función *create()* y dentro de la función le da el valor correspondiente a cada atributo de la noticia. Por último, se le añade los tags correspondientes, que inicialmente son los tags que tiene la fuente de la cual se ha extraído la noticia que estamos creando.

```
for (let j = 0; j < resp.length; j++) {
          let doc = await News.findAll({
            where: {
              url: resp[j].url,
            },
          })
          let carac = resp[j].title.length + resp[j].text.length + resp
[j].summary.length
          console.log("**************************")
          console.log("**************************")
          console.log("Noticia nueva")
          console.log("Numero de caracteres: " + carac + " Y " + tCarac
)
          console.log("**************************")
          console.log("**************************")
          try{
          if (doc.length === 0 && resp[j].html != null) {
            tCarac = tCarac +  resp[j].title.length + resp[j].title.len
gth + resp[j].summary.length
            const ndoc = await News.create({
              url: resp[j].url,
              title: resp[j].title,
              html: resp[j].html,
              text: resp[j].text,
              summary: resp[j].summary,
              image: resp[j].image,
              status: 'NEW',
              relevance: 0,
              crawlerId: source.id,
              characters: carac
            })
            if (ndoc) {
              const array = []

              tags.forEach((element) => {
                array.push({
                  newsId: ndoc.id,
                  tagId: element.id,
                })
              })

              await NewTags.bulkCreate(array)
            }
          }
          } catch(e) {
            console.log("ERROR AL INSERTAR LA NOTICIA")
            console.log(e)
          }
        }
```

Figura 24: Función para añadir noticias a la base de datos

7. Interfaz

En esta sección vamos a ver y analizar toda la interfaz de la herramienta.

Una interfaz es un punto donde los usuarios interactúan con el sitio web en el que navegan. La interfaz de usuario es una parte principal para la creación de un sitio web atractivo . Un buen diseño de interfaz de usuario representa una combinación perfecta de diseño visual, diseño de interacción y arquitectura de la información [35].

En primer lugar, al acceder a la herramienta vemos la pantalla para iniciar sesión, la cual pide un email y una contraseña asociada a este. (Ver Figura 25)

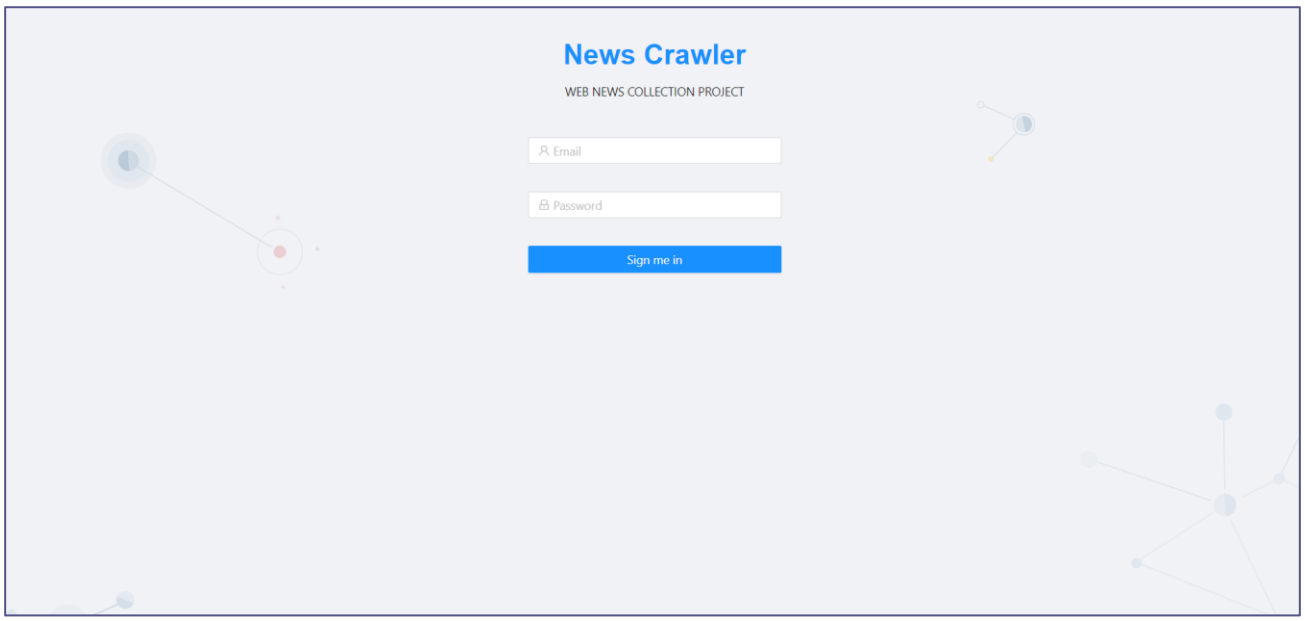

Figura 25: Inicio de Sesión

Una vez iniciado correctamente la sesión, accedemos a la primera pantalla, la cual es la pestaña de Dashboard que muestra un resumen de cómo está funcionando la herramienta.

Dashboard

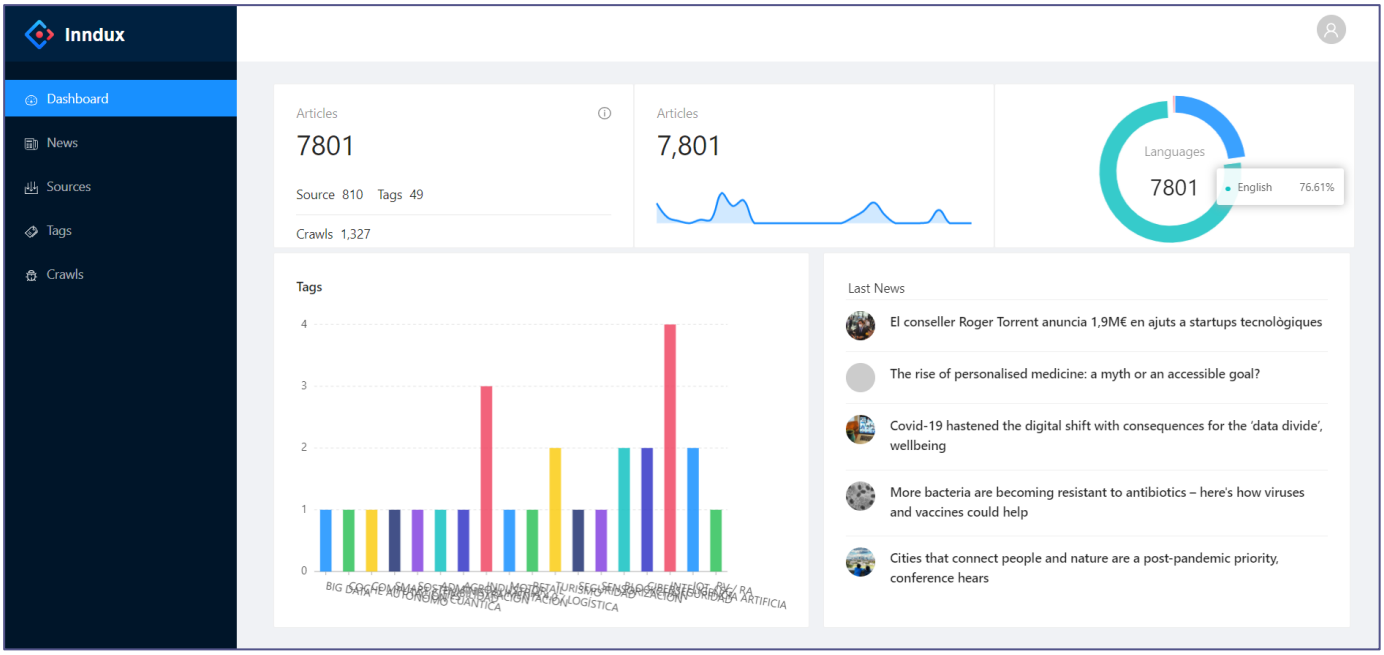

Figura 26: Dashboard

En el Dashboard o Panel de la aplicación podemos ver varios apartados (viéndolos de arriba a abajo y de izquierda a derecha):

-Contador de artículos, Sources (fuentes), Tags y Crawls totales que hemos creado u obtenido.

-Línea temporal de artículos, en la cual podemos ver cada día cuántos artículos se han extraído.

-Gráfica idiomas, donde se puede observar el porcentaje de noticias que hay en cada idioma.

-Gráfica Tags, donde están ilustradas la cantidad de noticias que tienen esos tags asociados.

-Últimas noticias, muestra las 5 últimas noticias que nuestra herramienta ha extraído.

News

El menú “News” contiene todas las noticias que la herramienta ha extraído:

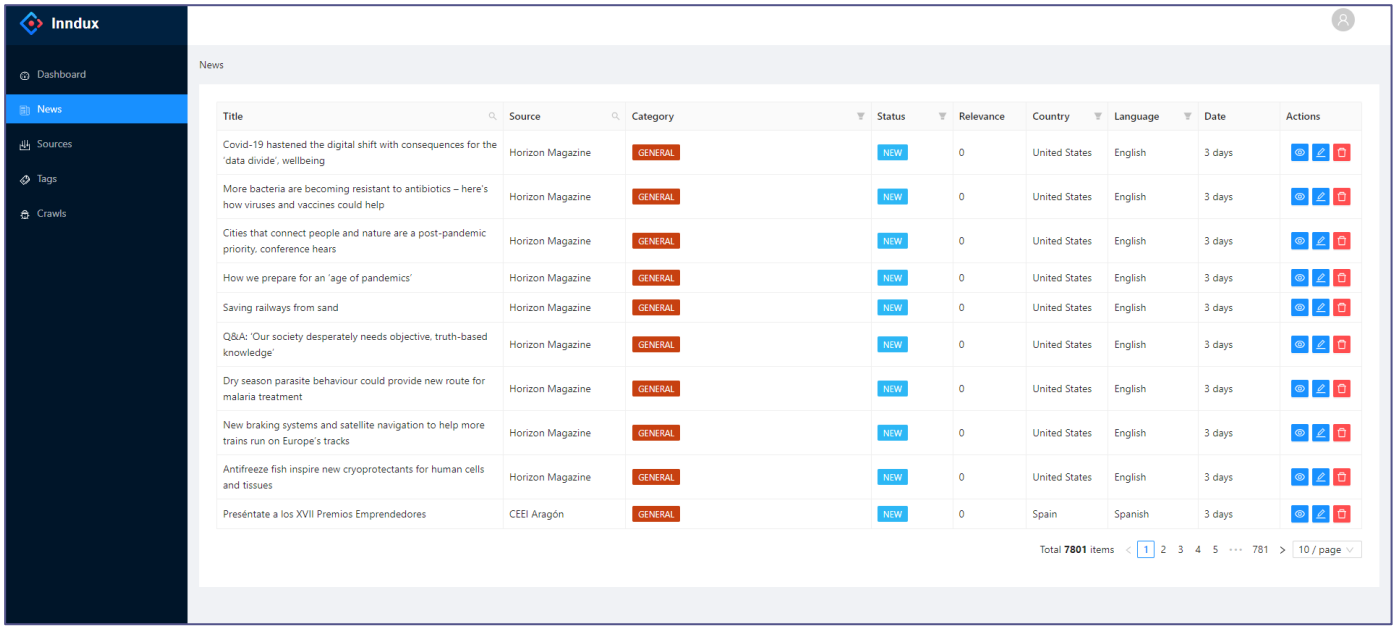

Figura 27: News

En el menú News, podemos observar los atributos principales de la noticia con los detalles de su extracción, como es:

- La fuente de la cual se extrae.
- La categoría de esta noticia.
- El estado de evaluación en el cual se encuentra (New, Accept o Discard).
- Relevancia que se le otorga a la noticia.
- País de donde viene la página web.
- Idioma en el que está la noticia.
- El tiempo que hace que se ha extraído la noticia.

Además, podemos hacer tres acciones con los tres botones que tienen cada una de las noticias: verlas en detalle, editarlas/evaluarlas y eliminarlas.

En la parte inferior, podemos apreciar que nos muestra el total de noticias y que podemos variar la cantidad de artículos que nos muestra por página, por defecto son 10 por página.

Article

Dentro del menú de News, si le damos a observar una noticia en particular, pasamos al siguiente menú en el que vemos todos los datos de la noticia: título, resumen, imagen y texto.

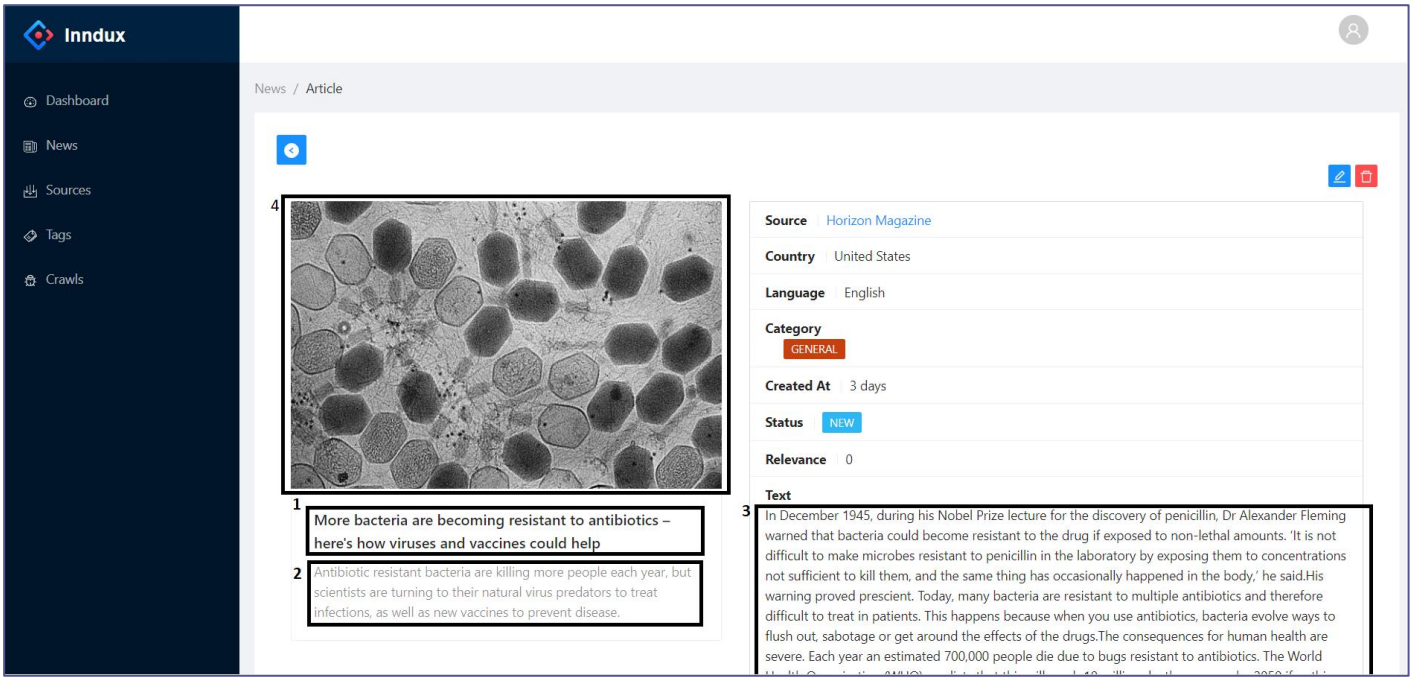

Figura 28: Article

La Figura 28 muestra ver en detalle la noticia seleccionada, con todos los atributos que se mostraban en el menú superior más cuatro que corresponden a los datos de la noticia:

1. Título.
2. Resumen o entradilla.
3. Texto completo de la noticia.
4. Imagen.

En este submenú también tenemos los botones de editar y de eliminar para optimizar el tiempo y la usabilidad de la aplicación y no tener que volver a la pantalla anterior para hacerlo.

Por último, tenemos un botón arriba a la izquierda que sirve para volver a la pantalla donde se encuentran todas las noticias.

Edit News

Cada noticia se puede evaluar o editar con el botón asignado para ello. Si seleccionamos editar una noticia nos saldrá la siguiente ventana.

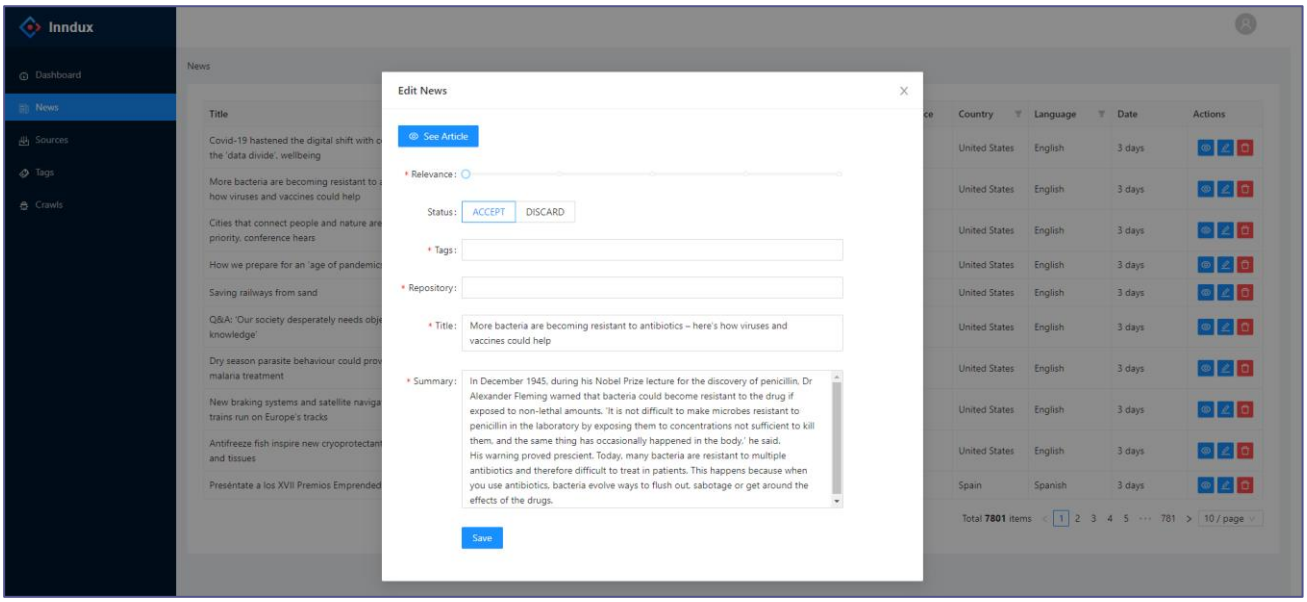

Figura 29: Edit News

En esta ventana emergente podemos ver todos los campos que podemos evaluar o editar. En primer lugar, tenemos un botón el cual nos llevará a la noticia en su página web, para leerla en su sitio original.

Después tenemos un formulario con los siguientes campos:

-Relevance: Relevancia que el usuario considera que tiene esa noticia para el interés de la empresa.

-Status: Estado que le da a la noticia, si la acepta o la rechaza. Si la noticia es rechazada, no será obligatorio rellenar ningún campo si ya la hemos descartado. Si la aceptamos, todos los campos son obligatorios.

-Tags: Desplegable de opciones donde podemos seleccionar entre las tags creadas todas las que creamos que tienen relación con la noticia a evaluar.

-Repository: Desplegable con los dos repositorios que tiene la empresa, la cual indicará la base de datos utilizada.

-Title: Título de la noticia, se inicializa con el que se extrae, pero el usuario puede decidir cambiarlo.

-Summary: Resumen, la herramienta genera un resumen a partir del texto completo que extrae de la noticia, pero el usuario puede modificarlo.

Una vez rellenado todo, el sistema guarda la noticia en la base de datos con los nuevos atributos.

Sources Crawler

En el menú “Sources crawler” podemos encontrar todas las fuentes que han sido creadas por un usuario. (Ver Figura 30)

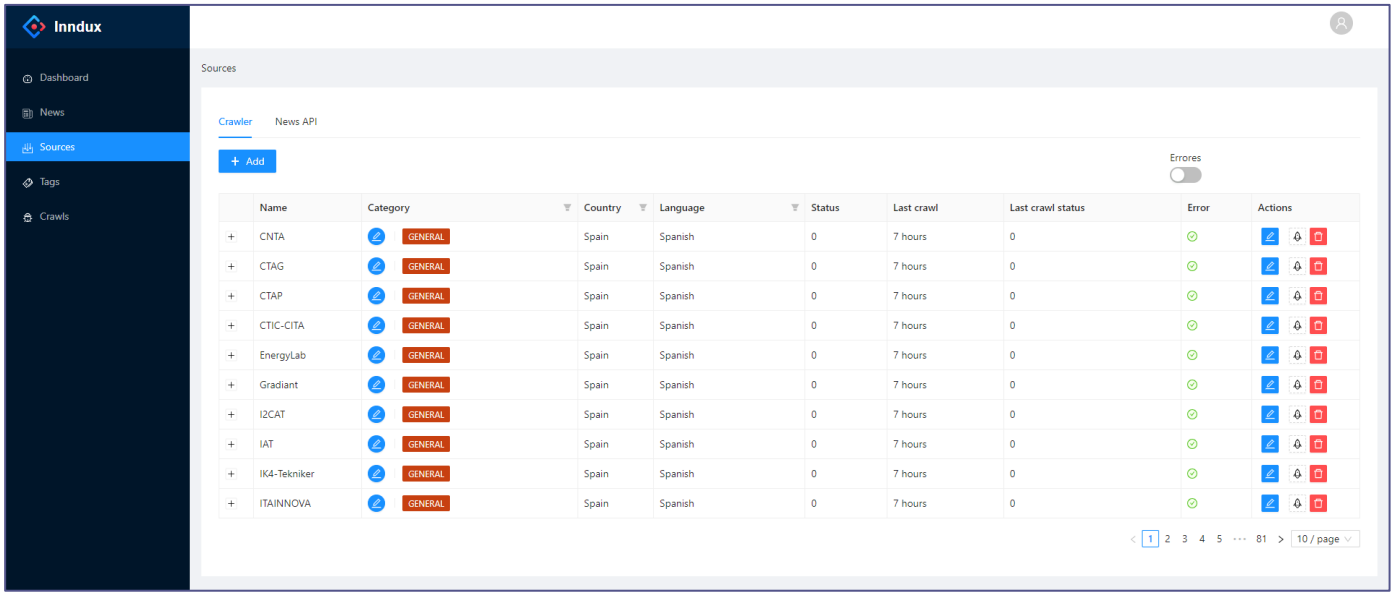

Figura 30: Sources crawler

Podemos ver todos los atributos de cada una de las fuentes creadas.

Además, tenemos tres botones de acciones para cada una de estas: Editar, Ejecutar y Borrar. En este menú también tenemos la opción de crear nuevas fuentes con el formulario “Create Source” explicado en el siguiente apartado.

Tenemos un botón que nos sirve para filtrar las fuentes y observar solo las fuentes con errores, ya que a veces las páginas web cambian y tenemos que reconfigurar algunas fuentes.

Al ejecutar una fuente, el sistema ejecuta el crawler y extrae las noticias de la fuente seleccionada. Una vez ejecutada nos sale este resultado que muestra los atributos extraídos de todas las noticias de esa fuente. (Ver Figura 31)

Figura 31: Ejecución de una fuente

Create Source

Para crear una fuente, desde el menú de Sources le damos al botón “Add”, lo que nos mostrará un formulario con todo lo necesario para que una fuente funcione.

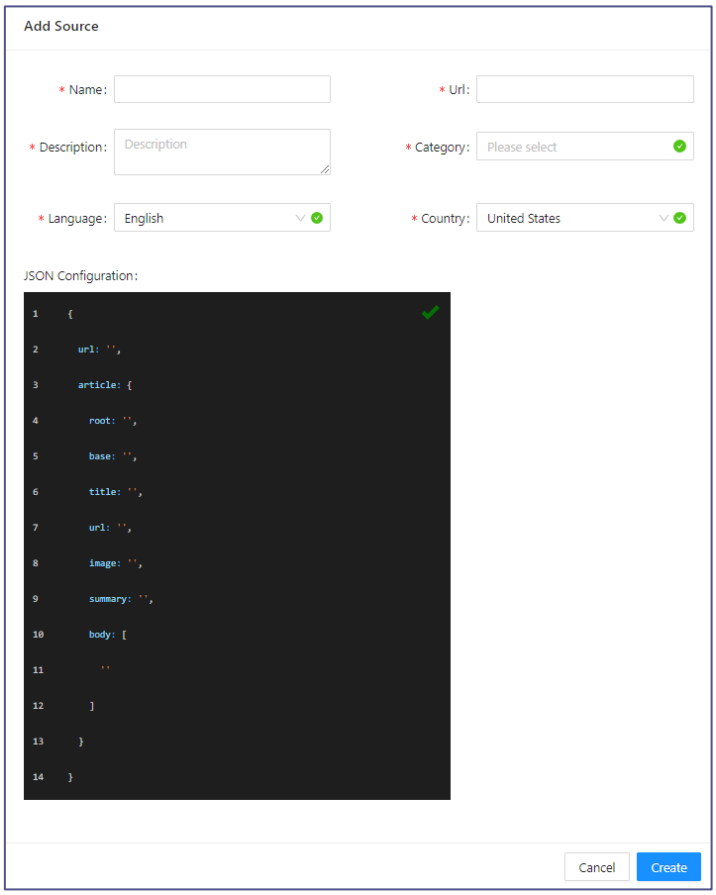

Figura 32: Add Source

En el formulario se pueden ver los siguientes campos:

-Name: Nombre de la fuente.

-Url: URL de la página web.

-Description: Descripción de la fuente.

-Category: Tag que le atribuimos de inicio.

-Language: Idioma de las noticias.

-Country: País del cual proviene la página web.

-JSON configuration: JSON donde escribimos los selectores adecuados para que consiga extraer los atributos que nos interesan de la página web.

Una vez completado todo, el sistema crea la fuente y la guarda en la base de datos. Este formulario es idéntico cuando se quiere editar una fuente.

Sources API

El menú “Sources API” muestra todas las fuentes que vienen del API ya instalado de internet, el cual se llama “newsAPI” y contiene páginas web de noticias de todo el mundo.

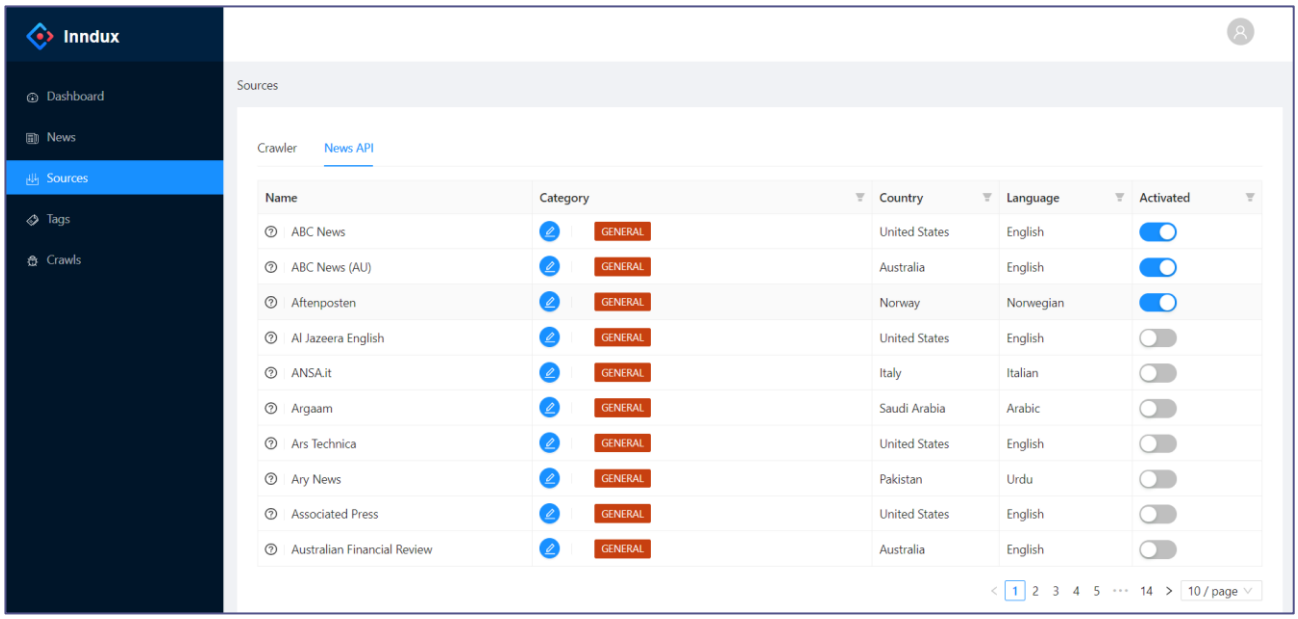

Figura 33: Sources API

En la Figura 33 se pueden observar los atributos de cada fuente, y en la última columna todos tienen un botón para activar o desactivar la fuente, esto sirve para cuando se haga la extracción automática solo consideré las fuentes que están activas.

Tags

La Figura 34 muestra todas las Tags que se encuentran en la base de datos y que a su vez han sido creadas por los usuarios.

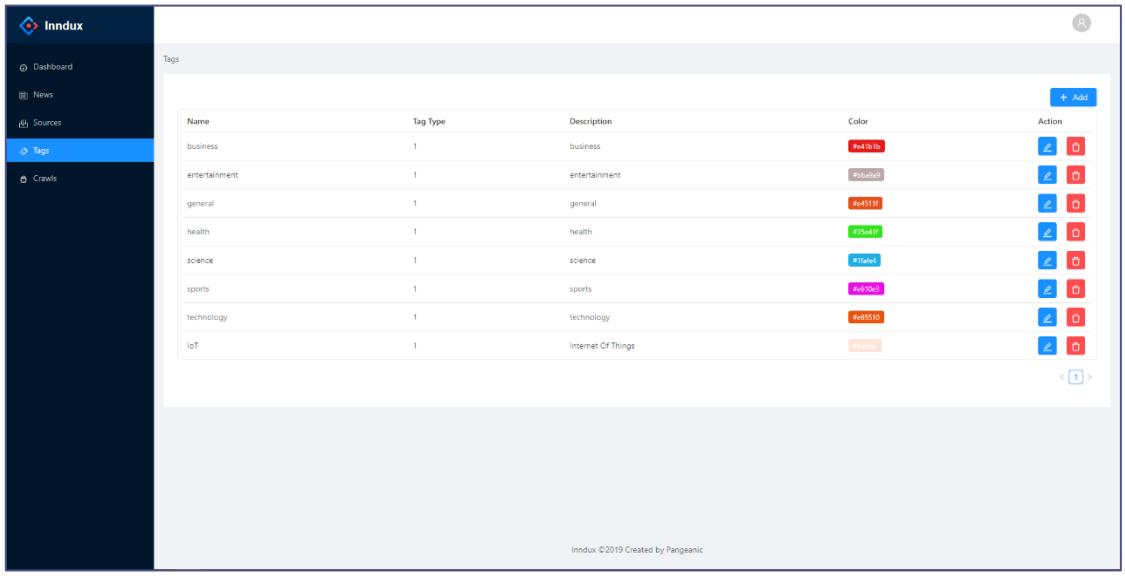

Figura 34: Tags

Cada Tag tiene una clase de atributos:

-Name: Nombre del Tag.

-Tag Type: Valor numérico que se le da para diferenciar algunos Tags, como podrían ser los Tags de categoría y los que representan el repositorio.

-Description: Una breve descripción.

-Color: Color que caracteriza cada Tag.

La última columna tiene dos acciones para cada tag: editarlos o borrar.

En este menú también se encuentra el botón “Add” para añadir nuevas Tags.

Add Tag

A la hora de añadir una Tag, nos saldrá un formulario para rellenar con todas las características propias de las Tags.

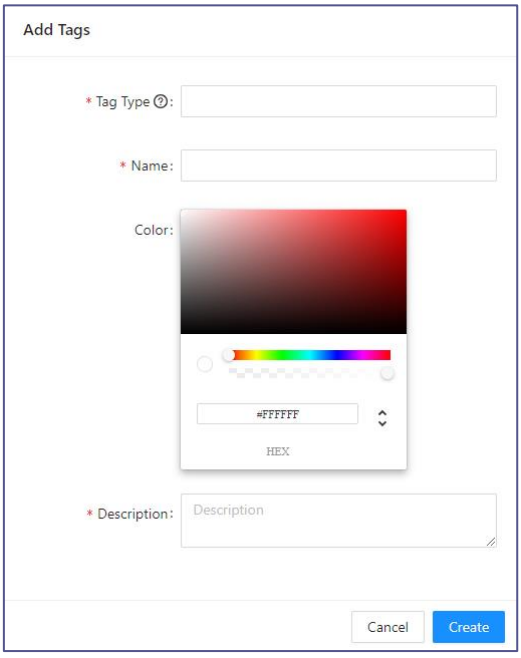

Figura 35: Add Tag

En este formulario se completan las mismas características que se pueden ver en la pantalla de Tags.

Si todos los datos son introducidos correctamente, el sistema guardará la Tag en la base de datos y se mostrará en la pantalla de todas las Tags.

Crawls

El último menú correspondo a “Crawls” donde tenemos el historial de extracciones, el cual muestra cada vez que se ejecuta una extracción, tanto automática con todas las fuentes de la herramienta como manual de una sola fuente.

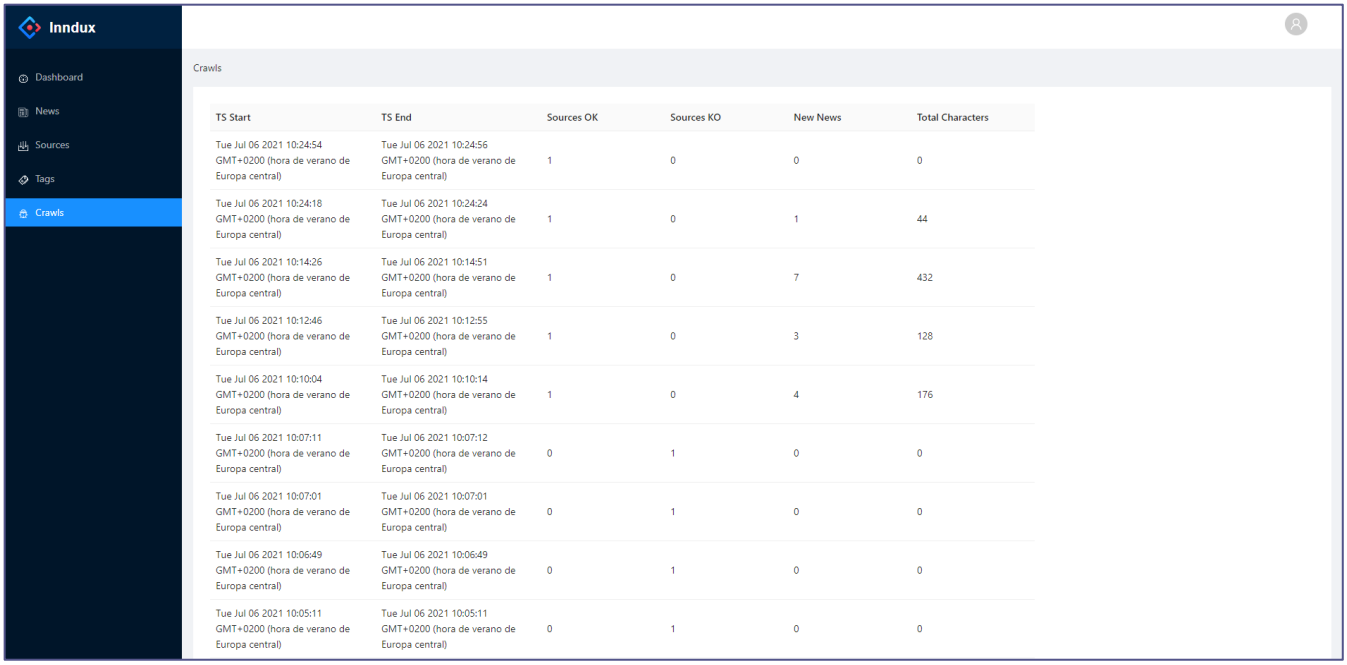

Figura 36: Crawls

Esta pantalla como las anteriores está dividida en columnas con atributos para cada una de las extracciones:

-TS start: Indica el momento de inicio de la extracción.

-TS end: Indica el momento del fin de la extracción.

-Sources OK: Señala las fuentes que han funcionado en la extracción.

-Sources KO: Señala las fuentes que han dado algún error en la extracción y no se ha podido extraer de esta información.

-New News: Cantidad de noticias nuevas que hemos obtenido en la extracción.

-Total Characters: Total de caracteres de las nuevas noticias.

8. Pruebas y Evaluación

Una parte importante en cualquier proyecto es la fase de pruebas y evaluación. Con pruebas intentamos probar todo tipo de combinaciones y errores para ver que la herramienta es capaz de responder correctamente. Es interesante que estas pruebas las hagan personas externas para que busquen en fallo. Y la evaluación por parte de los clientes es necesario para saber que se han conseguido las necesidades pedidas.

8.1. Pruebas

Las pruebas de software son comprobaciones que permiten verificar la calidad y el buen funcionamiento de un producto software. Mediante técnicas experimentales podemos descubrir los errores que puede tener la aplicación y corregirlos.

Las pruebas de software se separan en casos de prueba que son un conjunto de condiciones o variables las cuales determinarán si el requisito de la aplicación es parcial o completamente satisfactorio.

La característica principal de un escrito de caso de prueba es que hay una entrada conocida y una salida esperada. La entrada conocida debe probar una precondición y la salida esperada debe probar una postcondición.

Para la validación de esta herramienta hemos realizado un conjunto de pruebas sobre cada funcionalidad posible en la aplicación. Al finalizar la implementación de cada funcionalidad se ha realizado un caso de prueba sobre él para comprobar que funciona bien y corregir errores.

La tabla 23 muestra todos los casos de prueba que hemos realizado para la comprobación de cada funcionalidad de la aplicación.

Código	Descripción
PRUEBA 01	Inicio de sesión
PRUEBA 02	Navegar por distintas pantallas
PRUEBA 03	Ver noticias individualmente
PRUEBA 04	Volver a la lista de noticias
PRUEBA 05	Ir a la URL de la noticia original
PRUEBA 06	Evaluar las noticias
PRUEBA 07	Cambiar status y sus restricciones
PRUEBA 08	Cerrar ventana de evaluación
PRUEBA 09	Eliminar noticias
PRUEBA 10	Abrir formulario para fuente de crawler
PRUEBA 11	Añadir categorías de una fuente
PRUEBA 12	Modificar fuente de crawler
PRUEBA 13	Seleccionar categorías en el formulario de fuentes
PRUEBA 14	Crea fuente crawler
PRUEBA 15	Ejecutar fuente de crawler
PRUEBA 16	Eliminar fuente de crawler
PRUEBA 17	Ver detalles de fuente de crawler
PRUEBA 18	Activar fuente API
PRUEBA 19	Filtrar fuentes crawler erróneas
PRUEBA 20	Añadir Tag
PRUEBA 21	Modificar Tag
PRUEBA 22	Eliminar Tag
PRUEBA 23	Cerrar sesión

Tabla 22: Casos de prueba

Como ejemplo de esta serie de pruebas, vamos a explicar varios casos de pruebas a continuación que permitan ver la correcta funcionalidad a la hora de crear una fuente, rellenar todo su formulario y ejecutarla para ver que es correcta.

PRUEBA 10: Abrir formulario para fuente de crawler	
Caso de prueba: CP-10-01	**Nombre:** Usuario abre el formulario de una fuente de crawler
Descripción: Desde el menú de fuentes crawler un usuario abre el formulario para crear una nueva fuente	
Actividades	
Entorno de ejecución: - El usuario debe existir en la tabla “Users” de la base de datos	
Acciones: - Entrar en la pestaña de fuentes crawler y pulsar el botón “Add”	
Resultados	
Resultado esperado: Se abre una ventana con un formulario para crear una fuente	
Resultado real: SATISFACTORIO	

Tabla 23: Caso de prueba formulario

PRUEBA 13: Seleccionar categorías en el formulario de fuentes	
Caso de prueba: CP-13-01	**Nombre:** Usuario selecciona las categorías en el formulario
Descripción: En la ventana del formulario, en el apartado de categorías un usuario seleccionará una o varias categorías	
Actividades	
Entorno de ejecución: - El usuario debe existir en la tabla "Users" de la base de datos - Las categorías deben existir en la tabla "Tags"	
Acciones: - Seleccionar el campo "Category" en el formulario - Seleccionar la o las categorías que se vean oportunas del desplegable con todos los Tags que hay.	
Resultados	
Resultado esperado: Todas las categorías seleccionadas aparecerán en el formulario	
Resultado real: SATISFACTORIO	

Tabla 24: Caso de prueba categorías

PRUEBA 14: Crear fuente crawler	
Caso de prueba: CP-14-01	**Nombre:** Usuario crea una fuente de crawler
Descripción: Un usuario crea una nueva fuente con unos parámetros únicos.	
Actividades	
Entorno de ejecución: - El usuario debe existir en la tabla "Users" de la base de datos - Las categorías deben existir en la tabla "Tags" - El fichero JSON debe tener una sintaxis correcta	
Acciones: - Pulsar desde el formulario ya rellenado el botón "Create"	
Resultados	
Resultado esperado: La fuente se añadirá a la base de datos y se mostrará en la lista de fuentes	
Resultado real: SATISFACTORIO	

Tabla 25: Caso de pruebas crear fuente

PRUEBA 15: Ejecutar fuente crawler	
Caso de prueba: CP-15-01	**Nombre:** Usuario ejecuta una fuente de crawler
Descripción: Un usuario desde el menú de fuentes crawler, ejecuta una fuente individualmente.	
Actividades	
Entorno de ejecución: - El usuario debe existir en la tabla “Users” de la base de datos - La fuente tiene que existir en la tabla “Sources”	
Acciones: - Pulsar en la fuente que queremos ejecutar el botón: 	
Resultados	
Resultado esperado: La fuente se ejecutará desde la base de datos y se mostrará por pantalla las noticias extraídas.	
Resultado real: SATISFACTORIO	

Tabla 26: Caso de prueba ejecutar fuente

8.2. Resultados

En esta sección vamos a mostrar visualmente algunas de las pruebas que hemos realizado sobre la aplicación para ver que el funcionamiento es correcto, en concreto, desde la creación de una nueva fuente hasta ejecutarla y comprobar que hemos extraído correctamente las noticias de esta nueva fuente.

La primera prueba que vamos a realizar es comprobar que la ventana para crear una nueva fuente funciona de manera correcta. Para ello, en el menú “Sources crawler” pulsaremos el botón “Add” (ver Figura 37)

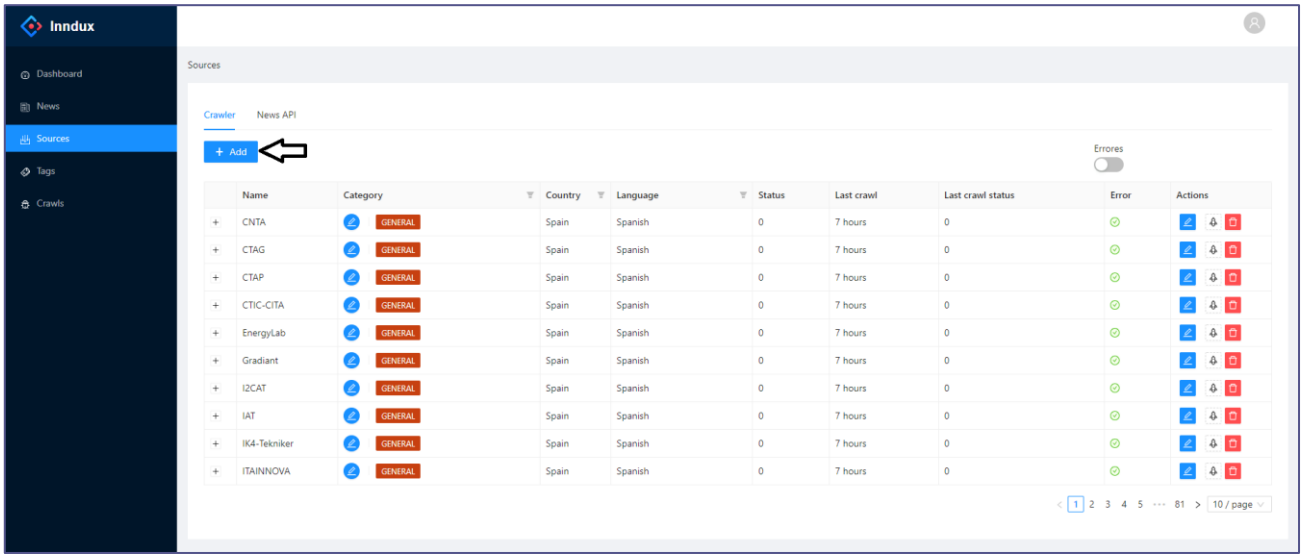

Figura 37: Ilustración de botón "Add"

Al pulsar el botón aparece una nueva ventana sobrepuesta al menú, la cual corresponde al formulario para crear una nueva fuente. Para comprobar su funcionamiento hemos hecho diversas pruebas.

La siguiente prueba ha sido intentar crearla sin rellenar ningún dato para que no nos lo permita y nos aparezca un error.

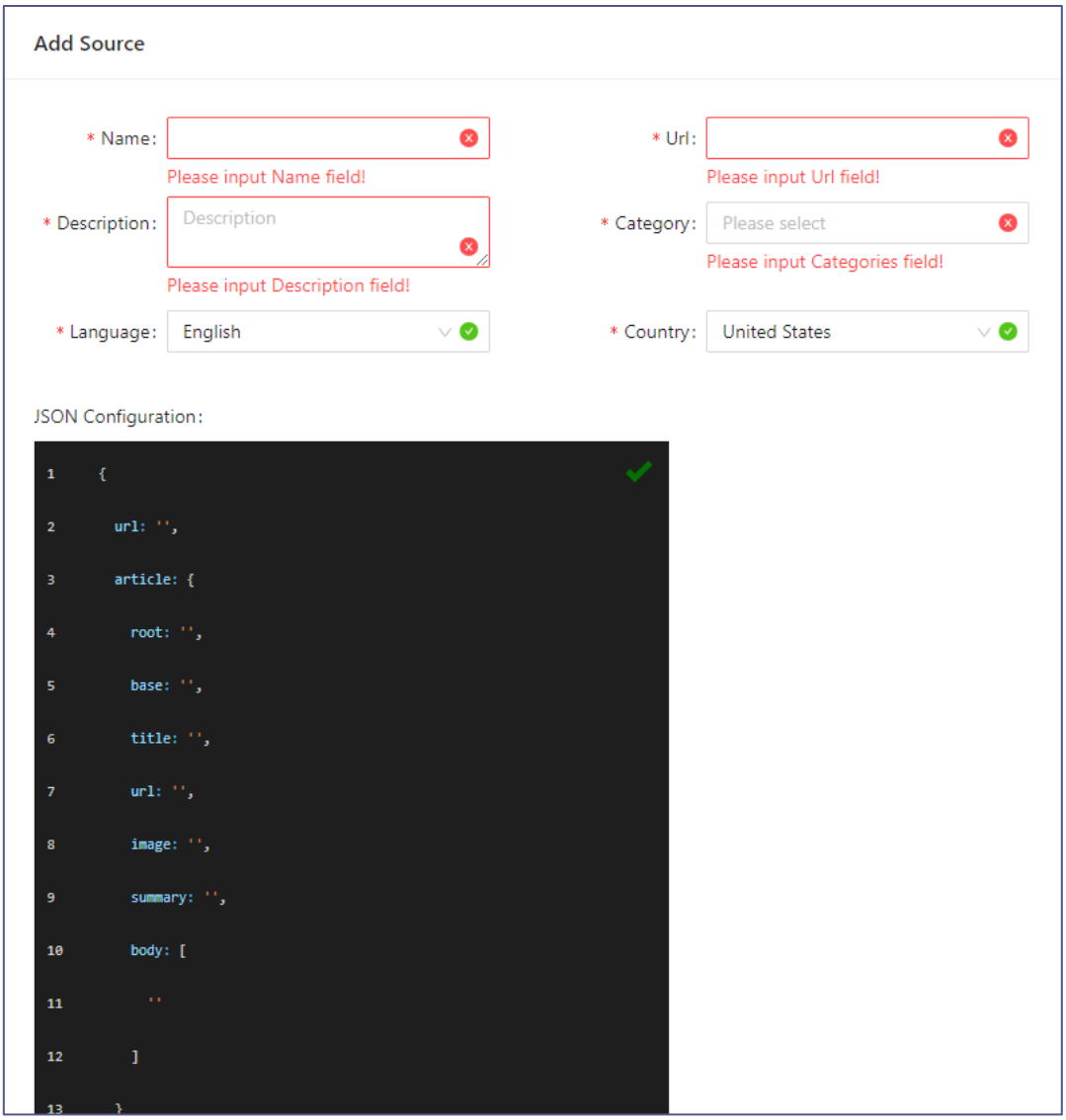

Figura 38: Error al añadir una imagen

En la Figura 38 observamos que en aquellos campos que no están completados nos muestra un mensaje de error porque es obligatorio rellenar estos campos para crear una fuente y con estos avisos hace saber al usuario lo que es necesario rellenar.

Una vez comprobado que si algún campo obligatorio es necesario nos salta el error y nos avisa para rellenar correctamente el formulario. El campo “category” corresponde a las Tags que le asignamos a esta nueva fuente, por lo que al pulsar sobre este campo debe salir un desplegable con todas las Tags que tenemos en la base de datos.

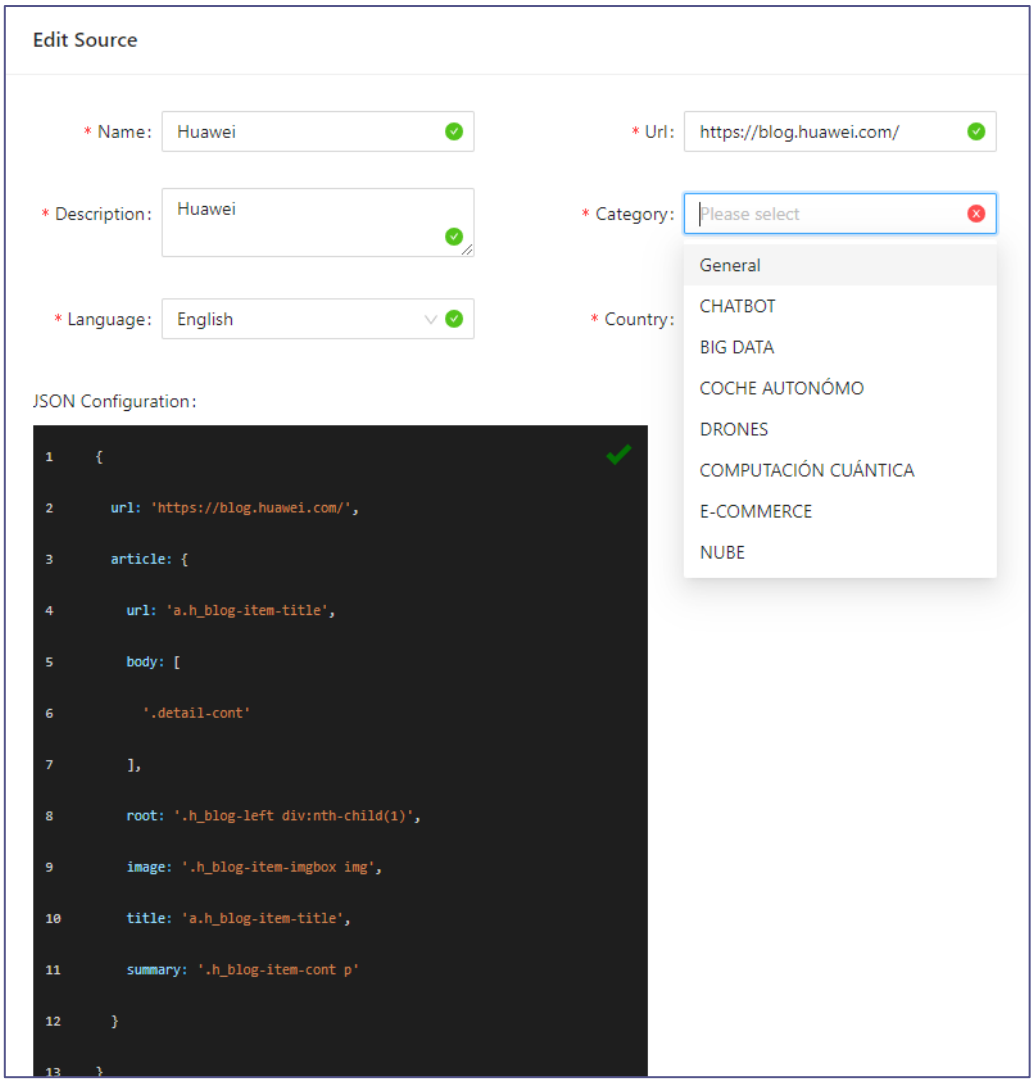

Figura 39: Desplegable Tags

La Figura 39 representa el buen funcionamiento de la herramienta para mostrar el desplegable de los Tags que tenemos creados en la base de datos.

El resto de los campos los rellenará el usuario para que sea todo correcto y se pueda crear la fuente. El campo más complicado es la configuración JSON, ya que hay que rellenarlo con los selectores específicos de la página web de donde vamos a extraer las noticias. Cómo completar esta configuración está detallado en el Anexo I.

Después de rellenar todos los campos de manera correcta, la herramienta guarda en la base de datos esta nueva fuente y se muestra en la lista de fuentes. La siguiente prueba es ejecutar la fuente y ver el resultado de la extracción.

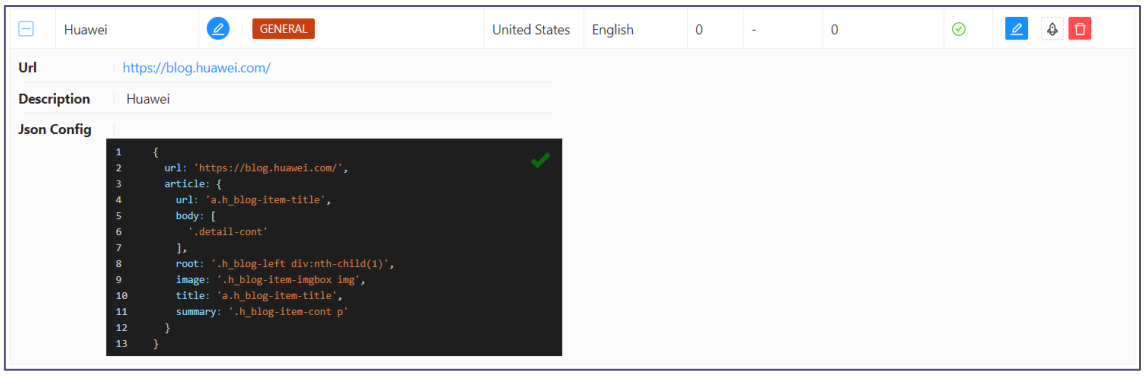

Figura 40: Fuente creada

La Figura 40 representa que se ha creado correctamente la fuente y que está guardado en la base de datos. El siguiente paso será ejecutarlo para comprobar que la configuración JSON funciona para la página web.

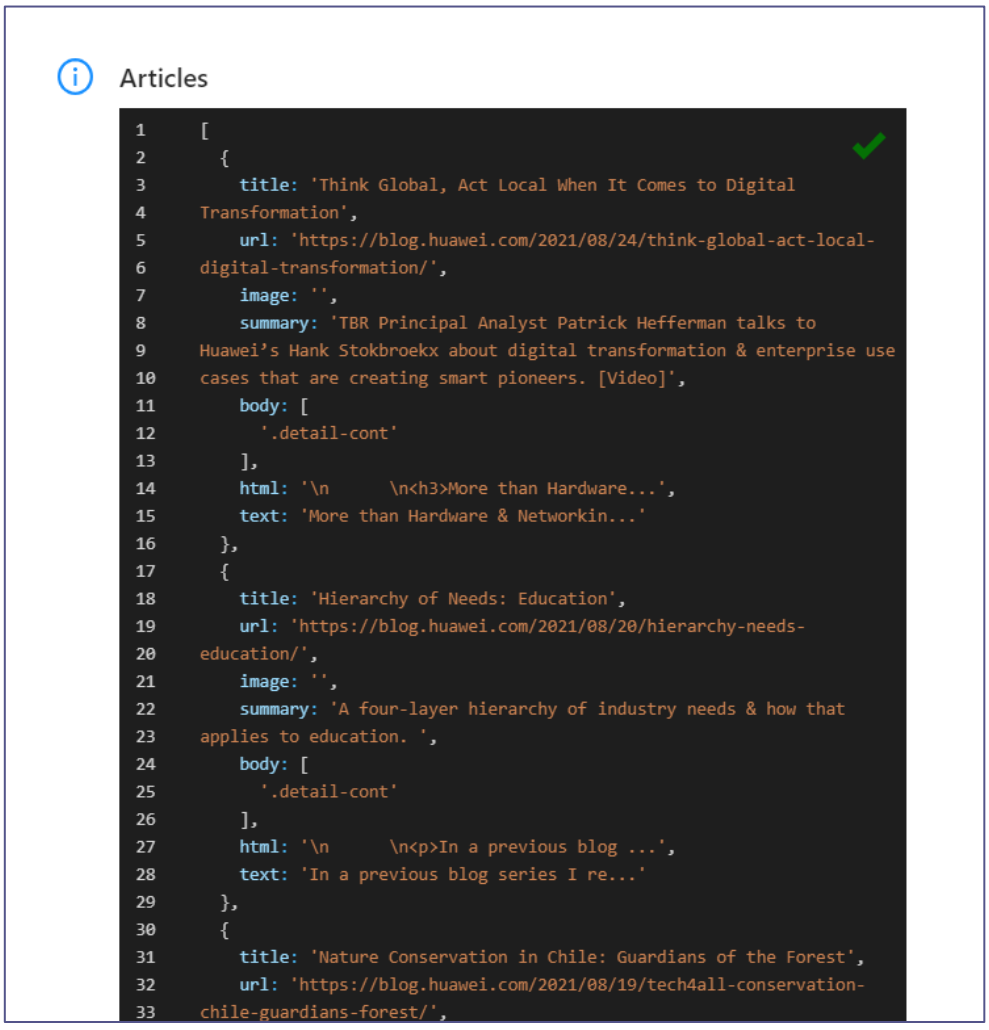

Figura 41: Ejecución de una fuente

La Figura 41 muestra la correcta ejecución de la fuente ya que se puede ver los atributos de cada noticia que queremos extraer.

La última prueba que vamos a hacer al sistema es la de comprobar si se ha guardado correctamente las noticias en la base de datos y podemos verlas de manera individual. Para ello vamos a ir al menú de “News” donde están las noticias extraídas.

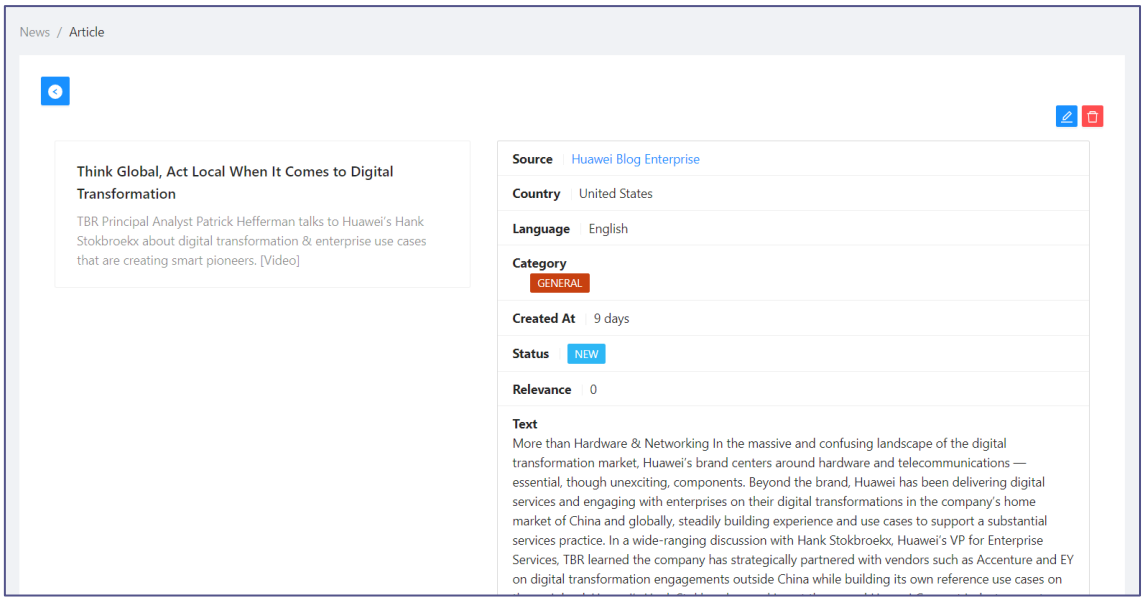

Figura 42: Noticia extraída

En la Figura 42 podemos observar que tenemos la noticia que hemos extraído con todos los atributos que le corresponden. Con esto podemos afirmar el buen funcionamiento de la creación de fuentes y la creación de estas.

8.3. Evaluación

La evaluación de programas y aplicaciones es un proceso que trata de valorar de manera sistemática y objetiva la pertinencia, el rendimiento y el éxito de los programas y proyectos concluidos y en curso.

En nuestro caso esta evaluación la ha llevado a cabo el cliente, el cual comprobaba los cambios que se iban haciendo en la aplicación y probaba para su propio uso.

Como se ha comentado anteriormente, en este proyecto hemos seguido una gestión ágil, la cual hace que los clientes estén más involucrados en el proyecto día a día y valoren cada cambio que se va haciendo.

En nuestro caso, los clientes tenían acceso a la herramienta en todo momento y podían ver todo cambio que hacíamos. Una vez cada dos semanas hacíamos reuniones en las cuales comentábamos los cambios, que les parecían y si les convenía o les gustaría tener algo de otra manera.

El feedback del cliente ha sido constante durante todo el proyecto mediante correo electrónico y reuniones hemos conseguido que la evaluación de la aplicación sea constante y con cambios poco a poco llegue a ser la aplicación deseada y que fuese lo más cómoda para ellos ya que iban a ser los usuarios habituales de esta.

Además, desde la parte del cliente se ha hecho saber la satisfacción por el trabajo hecho, donde. Con esta herramienta van a ahorrar mucho tiempo y coste en empleados ya que la aplicación consigue extraer noticias a una velocidad y cantidad incapaces para el ser humano.

9. Conclusión y trabajo futuro

Para finalizar el proyecto vamos a realizar una conclusión sobre todo del proyecto y hacer una visión general sobre este. También observar y comentar los cambios y avances que se han hecho de formación y personalmente.

Haremos una conclusión y resumen del proyecto y explicaremos el trabajo futuro de esta herramienta.

9.1. Conclusión

Para finalizar el proyecto vamos a realizar una conclusión sobre todo del proyecto y hacer una visión general sobre este. También observar y comentar los cambios y avances que se han hecho de formación y personalmente.

Con lo referente a los objetivos que se plantearon desde el principio del proyecto, se han cumplido satisfactoriamente, ya que la herramienta hoy en día es funcional y tiene implementadas todas las funciones que se pedían al principio.

La decisión de utilizar ReactJS ha sido la mejor decisión en el proyecto. Las páginas y aplicaciones web están creciendo y es por ello por lo que no basta con poseer conocimientos de una sola tecnología. La demanda de este tipo de aplicaciones es cada vez mayor y cada vez se delega más y más carga en el lado del cliente que en el servidor, debido al aumento de la potencia de procesamiento en los terminales. Es por ello por lo que las nuevas tecnologías, frameworks o librerías para el desarrollo de aplicaciones web facilitan el trabajo de desarrollar un producto eficiente sin por ello tener que invertir mucho tiempo. ReactJS justamente proporciona estas funcionalidades a este proyecto. Y no solo eso, ReactJS hace que el desarrollo sea más natural permitiendo una mejor cohesión en el código. La otra razón positiva de utilizar ReactJS es la facilidad de modificar cada componente individualmente y parte de estos. Esta razón será mucho más valiosa en la fase de mantenimiento por si se quiere modificar únicamente algún componente no tener que cambiar el resto del código, únicamente el que nos interesa moldear.

Lo más complejo del proyecto ha sido tener que comprender y aprender a utilizar ReactJS desde cero, ya que este no lo había utilizado ni visto hasta ahora, como su lenguaje particular, JSX. No obstante, creo que he conseguido superar el problema consiguiendo aprender lo suficiente para poder llevar a cabo todas las funciones de la aplicación. Aunque sé que me queda mucho por aprender en este ámbito he dado un paso gigantesco en este campo con este proyecto.

Aun teniendo que aprender un nuevo lenguaje y utilizar mucho tiempo en su aprendizaje, me ha parecido una experiencia muy buena para mi formación, como también el llevar un proyecto real a mi cargo pasando por todas las fases que este implica. Además, tener que comunicarme con el cliente para hablar los requisitos o hablar de los avances de la

aplicación o futuro de esta, me ha parecido una vivencia enriquecedora para mi personal y profesionalmente hablando.

Como conclusión, ha sido un proyecto difícil desde el principio y horas dedicadas, pero mirando con retrospectiva ha sido una gran ayuda para mi formación y además se ha conseguido sacar el proyecto satisfactoriamente.

9.2. Trabajo Futuro

Como se ha comentado anteriormente, el proyecto ya consigue hacer todas las funciones que inicialmente se pedían con éxito. Por lo tanto, es totalmente funcional y el cliente ya está empezando a utilizarlo.

Aun así, en un futuro ya se han comentado varias funciones que se pueden implementar y ser útiles dentro de la herramienta.

Una de las posibles mejores es la de la traducción automática, la cual el usuario elegiría qué idioma o idiomas le interesa tener las noticias. La herramienta al extraer una noticia traduciría esta y la guardaría traducida en la base de datos.

Otra funcionalidad que se podría incorporar es que la herramienta sea capaz de, leyendo todo el texto de la noticia, hacer un resumen de esta noticia automáticamente. Como también, si no hay imagen en esa noticia, buscar mediante palabras claves de la noticia una imagen relacionada con esas palabras en una librería de imágenes públicas.

Por último, otra función que se ha comentado es la categorización automática, que al extraer las noticias y con las palabras claves de esta, la herramienta sea capaz de saber a qué categoría o categorías pertenece la noticia.

Como se puede ver, la visión a largo plazo de esta aplicación es la automatización total de esta. La fase final de esta sería que la herramienta fuese capaz de extraer, categorizar, resumir, traducir y evaluar las noticias automáticamente sin la interacción de una persona física, solamente para una revisión final.

10. Bibliografía

[1] *Las páginas web cumplen 25 años, esta es su historia*. (s. f.). ComputerHoy. https://computerhoy.com/noticias/internet/paginas-web-cumplen-25-anos-49980

[2] *A short history of the Web*. (s. f.). CERN. https://home.cern/science/computing/birth-web/short-history-web

[3] CPBI, R. S. (2021, 13 de mayo). *The History of Website Design: 30 Years of Building the Web [2021 Update]*. Inbound Marketing Agency | SEO FIRM | SMA Marketing. https://www.smamarketing.net/blog/the-history-of-website-design

[4] *What And Why React.js*. (s. f.). C# Corner - Community of Software and Data Developers. https://www.c-sharpcorner.com/article/what-and-why-reactjs/

[5] *Qué es el DOM*. (s. f.). Home de DesarrolloWeb.com. https://desarrolloweb.com/articulos/que-es-el-dom.html

[6] *Top React JS Features - Pros and Cons - Angular JS vs React*. (s. f.). Woocommerce Product addons. https://acowebs.com/react-js-features/

[7] Sufiyan, T. (2020, 4 de junio). *What is ReactJS: Introduction To React and Its Features*. Simplilearn.com. https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs

[8] *ReactJS Features - javatpoint*. (s. f.). www.javatpoint.com. https://www.javatpoint.com/react-features

[9] Ortega, J. M. (2018, 3 de junio). *Entendiendo los ciclos de vida en React*. Medium. https://medium.com/@jmz12/entendiendo-los-ciclos-de-vida-8a70abb3b51a

[10] *Presentando JSX – React*. (s. f.). React – Una biblioteca de JavaScript para construir interfaces de usuario. https://es.reactjs.org/docs/introducing-jsx.html

[11] *React Hooks - javatpoint*. (s. f.). www.javatpoint.com. https://www.javatpoint.com/react-hooks

[12] *Un vistazo a los Hooks – React*. (s. f.). React – Una biblioteca de JavaScript para construir interfaces de usuario. https://es.reactjs.org/docs/hooks-overview.html

[13] *Acerca | Node.js*. (s. f.). Node.js. https://nodejs.org/es/about/

[14] *Node.js - Introduction*. (s. f.). The Biggest Online Tutorials Library - AppML, AI with Python, Behave, Java16, Spacy. https://www.tutorialspoint.com/nodejs/nodejs_introduction.htm

[15] *Crawlers: qué son, cómo funcionan y afectan a tu servidor*. (s. f.). Blog de Linube. https://linube.com/blog/crawlers-que-son/

[16]*¿Qué es un crawler o arañas de la web y qué hacen? ⋆ Agencia Posicionamiento Web - The Lift Company*. (s. f.). Agencia Posicionamiento Web - The Lift Company. https://theliftco.eu/que-es-un-crawler-o-aranas-de-la-web-y-que-hacen/

[17] *Qué es Crawler - Definición, significado y ejemplos*. (s. f.). Arimetrics. https://www.arimetrics.com/glosario-digital/crawler

[18] *▷ Crawlers: Qué son y Cómo funcionan | Explicado Fácil*. (s. f.). Artilet. https://www.artilet.com/diccionario-seo/crawlers/

[19] *DBeaver Community | Free Universal Database Tool*. (s. f.). DBeaver Community | Free Universal Database Tool. https://dbeaver.io/

[20] *Mock Data Generation in DBeaver · dbeaver/dbeaver Wiki*. (s. f.). GitHub. https://github.com/dbeaver/dbeaver/wiki/Mock-Data-Generation-in-DBeaver

[21] *What is CSS?* (s. f.). The Biggest Online Tutorials Library - AppML, AI with Python, Behave, Java16, Spacy. https://www.tutorialspoint.com/css/what_is_css.htm

[22] *¿Qué es el CSS? - Aprende sobre desarrollo web | MDN*. (s. f.). MDN Web Docs. https://developer.mozilla.org/es/docs/Learn/CSS/First_steps/What_is_CSS

[23] *GESTIÓN ÁGIL vs GESTIÓN TRADICIONAL DE PROYECTOS ¿CÓMO ELEGIR? - Escuela de Negocios FEDA*. (s. f.). Bienvenidos a Escuela de Negocios FEDA - Escuela de Negocios FEDA. https://www.escueladenegociosfeda.com/blog/50-la-huella-de-nuestros-docentes/471-gestion-agil-vs-gestion-tradicional-de-proyectos-como-elegir

[24] *Gestión AGIL vs gestión clásica ¿Cual es mejor para gestionar proyectos?* (s. f.). Recusos en project management. https://www.recursosenprojectmanagement.com/gestion-clasica-vs-gestion-agile/

[25]*Tutorial: Introducción a React – React*. (s. f.). React – Una biblioteca de JavaScript para construir interfaces de usuario. https://es.reactjs.org/tutorial/tutorial.html

[26] *¿Qué es un web crawler? Cómo las arañas web optimizan Internet*. (s. f.). IONOS Digitalguide. https://www.ionos.es/digitalguide/online-marketing/marketing-para-motores-de-busqueda/que-es-un-web-crawler/

[27] *El diagrama de casos de uso en UML*. (s. f.). IONOS Digitalguide. https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/diagrama-de-casos-de-uso/

[28] *Software de diagramaciÃ³n en lÃnea y soluciÃ³n visual | Lucidchart*. (s. f.). Lucidchart. https://www.lucidchart.com/pages/es

[29] *Diagrama de Clases - manuel.cillero.es*. (s. f.). manuel.cillero.es. https://manuel.cillero.es/doc/metodologia/metrica-3/tecnicas/diagrama-de-clases/

[30] *Modelo relacional en la gestión de bases de datos*. (s. f.). Tips de Logicalis. https://blog.es.logicalis.com/analytics/conceptos-basicos-del-modelo-relacional-en-la-gestion-de-bases-de-datos

[31] Platzi: Cursos online profesionales de tecnología. (s. f.). Platzi. Recuperado 5 de agosto 2021, de https://platzi.com/blog/que-es-frontend-y-backend/

[32] *Qué es React. Por qué usar React*. (s. f.). Home de DesarrolloWeb.com. https://desarrolloweb.com/articulos/que-es-react-motivos-uso.html

[33] *React Router: Declarative Routing for React*. (s. f.). ReactRouterWebsite. https://reactrouter.com/

[34] *¿Qué es el Back End y Front End? ⚡20%Dto en servicios de Desarrollo*. (s. f.). Agencia Inbound Marketing Madrid. https://nestrategia.com/desarrollo-web-back-end-front-end/#:~:text=En%20otras%20palabras,%20el%20Back,la%20comunicación%20con%20el%20servidor

[35] *What is User Interface Design (UI) and why is it important?* (s. f.). F5 Studio. https://f5-studio.com/articles/what-is-user-interface-design-and-why-is-it-important

Anexo I: Configuración JSON

Para la configuración de una nueva fuente se ha comentado que uno de los requisitos es completar un fichero JSON. En el presente anexo vamos a ver como conseguimos rellenar el fichero JSON con los atributos que nos interesa partiendo de una página web.

En primer lugar, hay que saber cuáles son los selectores CSS y cuál es la función para cada uno de ellos para seleccionar la parte que nos interesa de nuestra URL.

El fichero JSON cuenta con distintos atributos los cuales concuerdan con los atributos que queremos extraer, y así configurar un selector específico para cada atributo.

Figura 43: Configuración JSON

La Figura 43 muestra los atributos, vamos a explicar individualmente como conseguimos sus selectores de una fuente real.

Para nuestro ejemplo utilizamos la URL: https://news.sap.com/type/feature-article/

Entramos desde nuestro navegador a esa URL y con la herramienta de inspeccionar conseguimos ver el código fuente de la página y seleccionar la parte del código que nos interesa.

URL

En este campo vamos a poner la URL de la página Web de la cual queremos extraer las noticias.

En nuestro ejemplo:

```
url: 'https://news.sap.com/type/feature-article/',
```

Figura 44: URL Ejemplo real

Root

Este dato hace referencia a donde se encuentra la raíz de la cual vamos a sacar las noticias, es decir, la parte de la página Web que contiene las noticias.

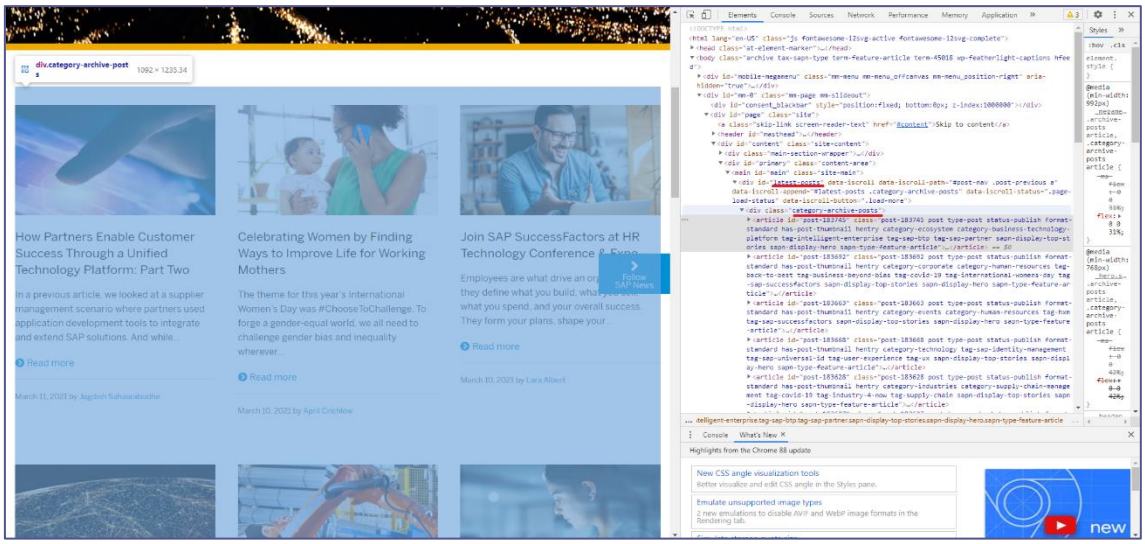

Imagen 45: Inspeccionar Root

La Figura 45 muestra dónde está la sección que contiene las noticias es "*div class= category-archive-posts*" y su padre en este caso es "*div id= latest-posts*". Con estos datos podemos sacar el selector necesario.

El padre hace referencia al elemento que contiene la sección más en detalle donde se encuentran las noticias. En este ejemplo:

```
root: '#latest-posts div:nth-child(1)',
```

Figura 46: Root Ejemplo real

Title

En este apartado buscamos donde está el título de la noticia. Para ello, lo primero que tenemos que hacer es encontrar el apartado donde se separan las noticias de manera individual y analizamos sobre una noticia (ver Figura 47).

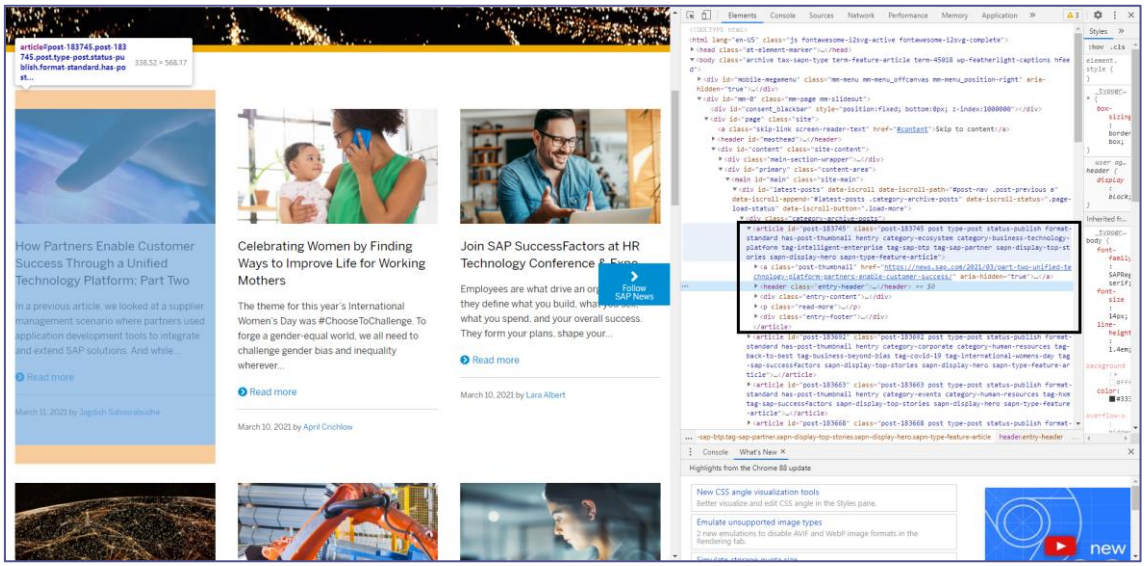

Figura 47: Inspeccionar Noticia

Una vez separado el código que contiene la noticia solo buscaremos el texto que corresponde al título (ver Figura 48).

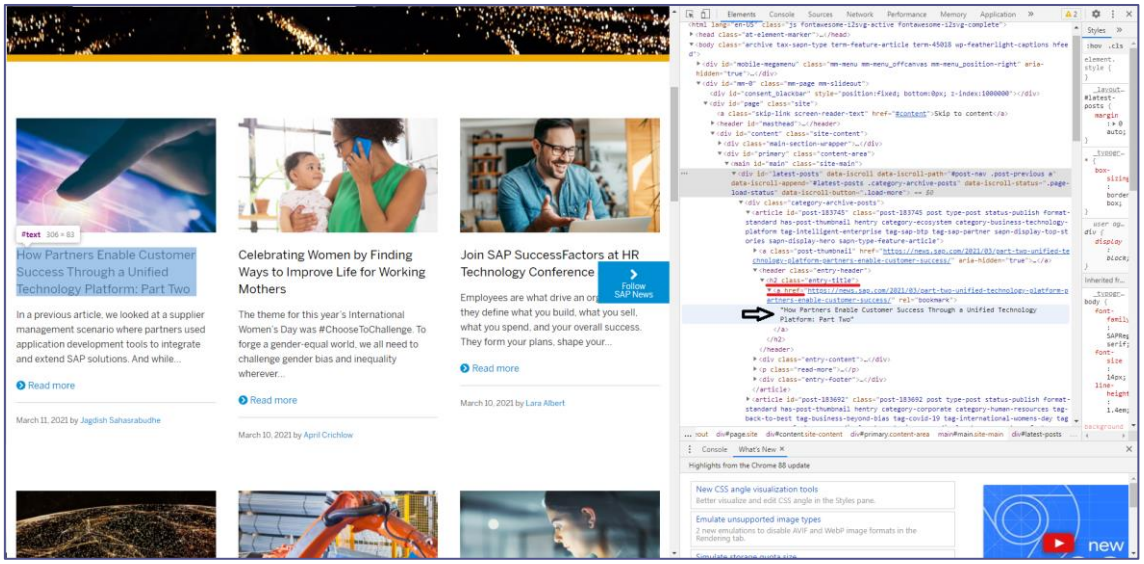

Figura 48: Inspeccionar Título

Como se puede apreciar, podemos ver que el título se encuentra en la etiqueta HTML "*h2*" y dentro de la etiqueta "*a*".Por lo que así quedaría en la configuración de nuestro ejemplo:

```
title: 'h2>a',
```

Figura 49: Title Ejemplo real

URL

Esta segunda URL, que se encuentra dentro de los corchetes de “article”, hace referencia a la URL de la noticia en particular.

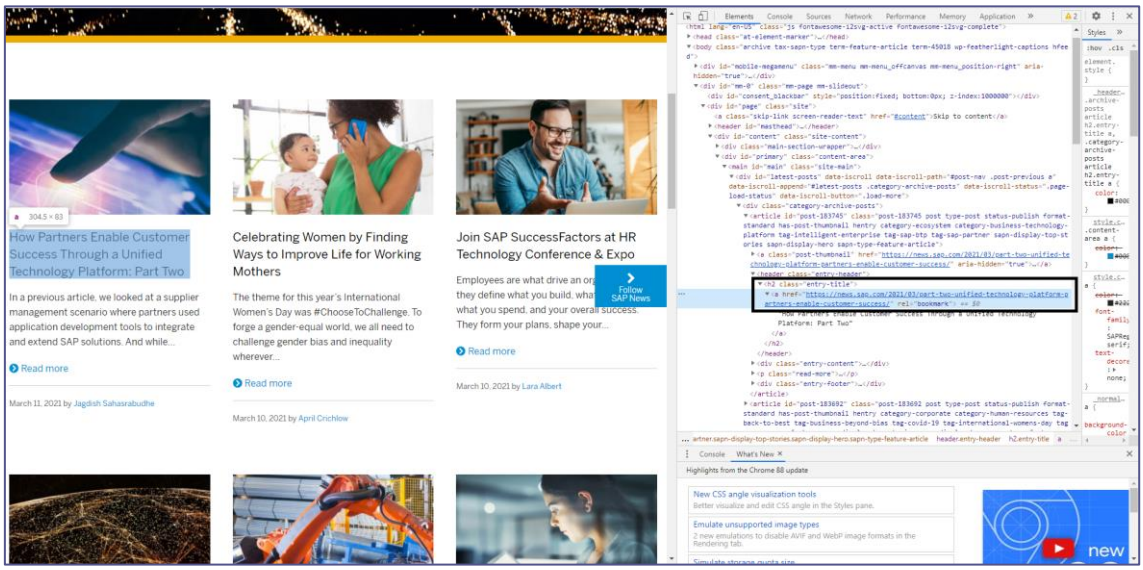

Figura 50: Inspeccionar URL

En la Figura 50 buscamos el enlace en la parte del código que corresponde a la noticia y este será el que nos llevará al artículo de la noticia.En nuestro ejemplo quedaría así:

```
url: 'h2>a',
```

Figura 51: URL Noticia Ejemplo real

Summary

Este atributo es el resumen o entradilla que encontramos en cada noticia en la página en la que se encuentran todas las noticias (ver Figura 52).

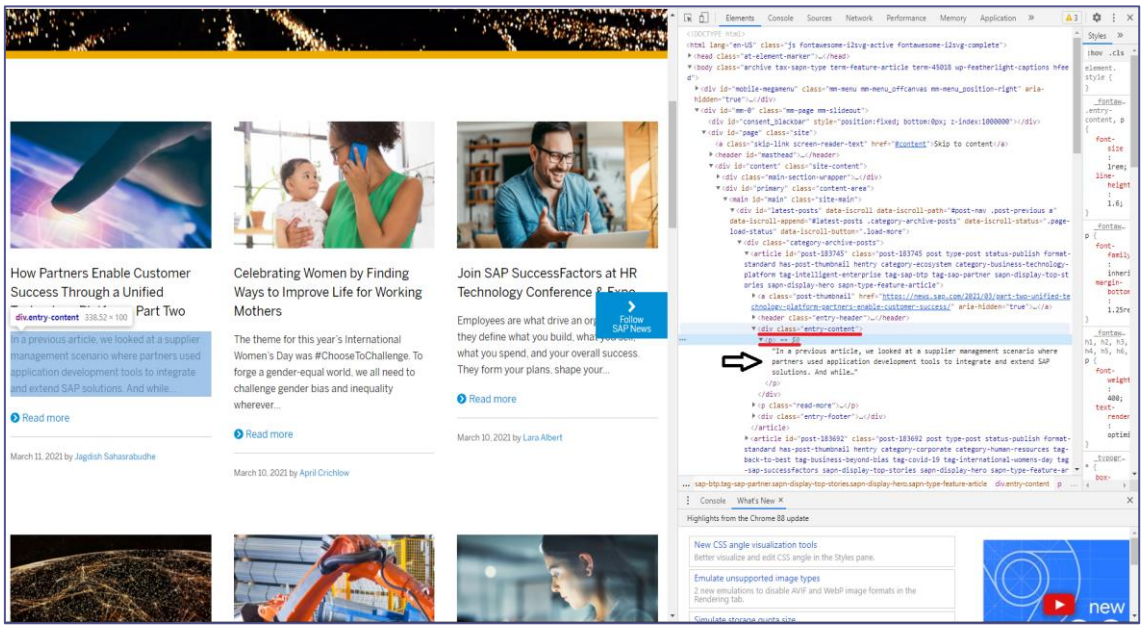

Figura 52: Inspeccionar Resumen

Como en los apartados anteriores, escribimos, con la mayor exactitud posible, el selector donde está contenido el resumen. En nuestro caso:

```
summary: '.entry-content p'
```

Figura 53: Summary Ejemplo real

Image

Buscamos la imagen que corresponde a la noticia (ver Figura 54).

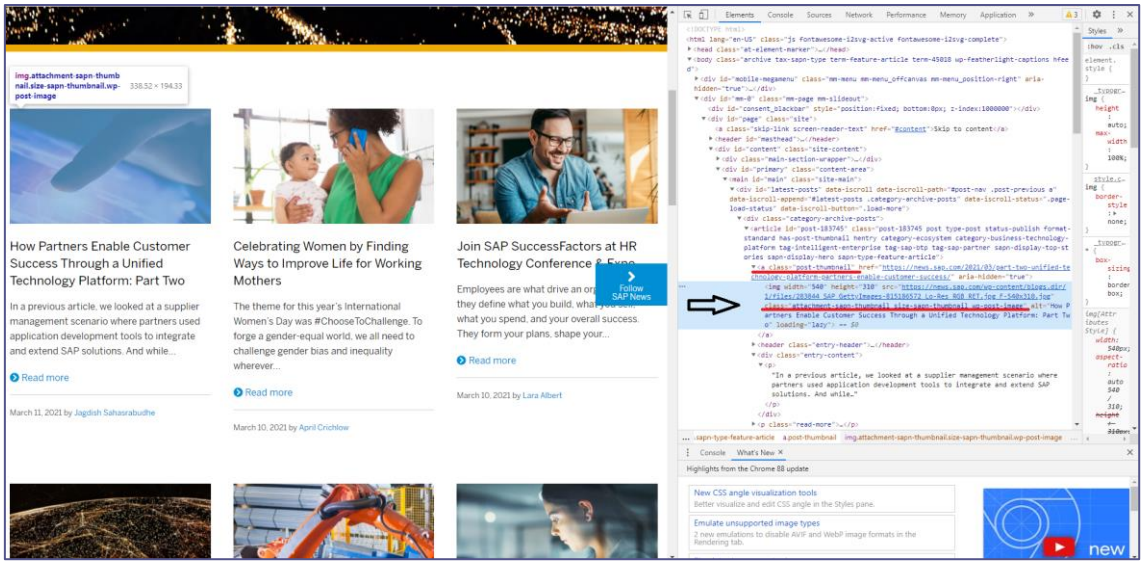

Figura 54: Inspeccionar Imagen

Y como muestra la imagen, escribimos los selectores que mejor definan donde se encuentra la imagen. En nuestro ejemplo real escribimos:

```
image: 'a img.wp-post-image',
```

Figura 55: Image Ejemplo real

Hay ocasiones en las que las noticias de una página web no contienen una imagen con relación a la noticia o el resumen de la noticia en sí o incluso se pueden dar ambas situaciones. Este es un ejemplo de otra página web real en la que ocurre lo comentado (ver Figura 56).

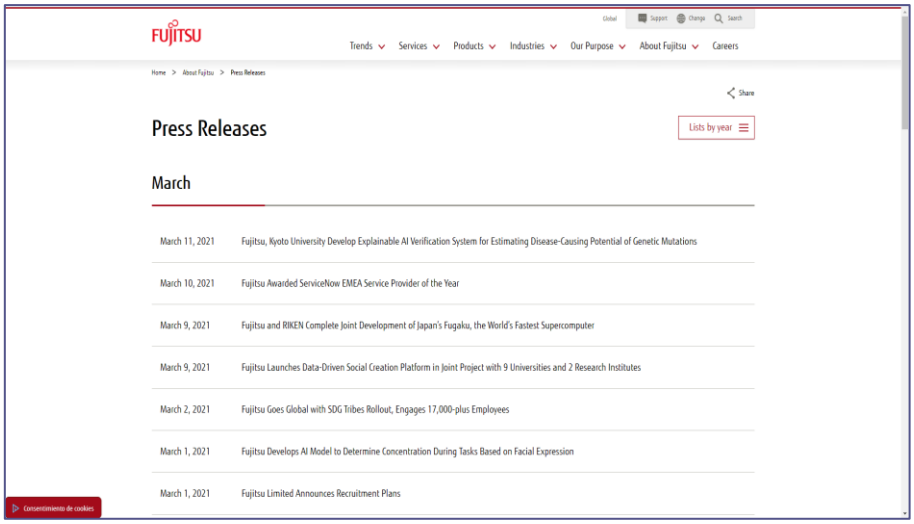

Figura 56: Ejemplo Página Web

Como se puede apreciar en estas noticias no hay ninguna imagen ni un resumen, únicamente un título. En estos casos en la configuración JSON los valores de estos campos se les asigna *null*.

```
image: null,

summary: null,
```

Figura 57: Configuración sin valores

Body

Este campo se compone de un array donde los selectores serán aquellos que contengan todo el texto de la noticia. Primero entramos en la noticia, todas las noticias de la misma página tienen el texto en el mismo selector, por lo que entramos en la que queramos y otra vez con la herramienta de inspeccionar obtenemos el código de la página web y así buscar el selector oportuno (ver Figura 58).

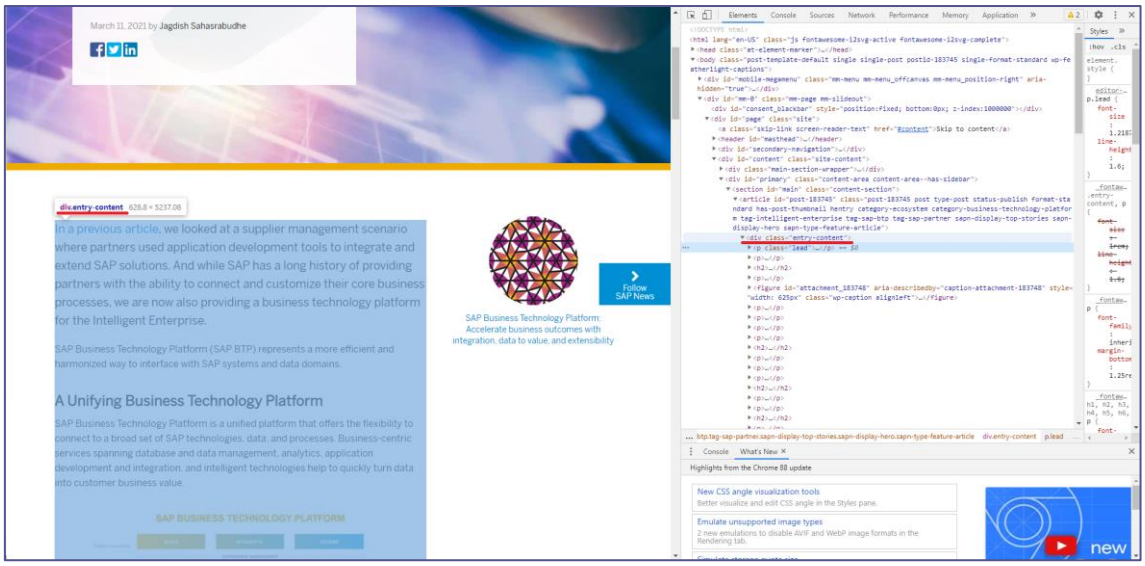

Figura 58: Inspeccionar Texto

Y como en el resto de los apartados, escribimos el selector más preciso que contenga el cuerpo de la noticia (ver Figura 59).

```
body: [

  '.entry-content'

],
```

Figura 59: Body Ejemplo real

Y con esto hemos completado todos los datos necesarios del fichero JSON para que funcione. Nuestro ejemplo real queda así:

```
{
  url: 'https://news.sap.com/type/feature-article/',
  article: {
    url: 'h2>a',
    body: [
      '.entry-content'
    ],
    root: '#latest-posts div:nth-child(1)',
    image: 'a img.wp-post-image',
    title: 'h2>a',
    summary: '.entry-content p'
  }
}
```

Figura 60: JSON Ejemplo real

Una vez hecho el fichero JSON y rellenado el resto de los campos de la ventana de creación de la fuente, pulsaremos el botón Create

Con esto hemos generado una nueva fuente de la cual poder extraer noticias. En la lista de *sources* se mostrará esta nueva fuente.

Ahora solo queda ejecutar el crawler con esta fuente para comprobar que la configuración sea correcta. Para ello, pulsamos el botón en acciones de ejecutar:

Una vez ejecutado, si todo es correcto, nos aparece una ventana de las noticias que hemos extraído con la estructura de un fichero JSON.

Ejecutando nuestra fuente que hemos configurado en el ejemplo, vemos por pantalla lo siguiente:

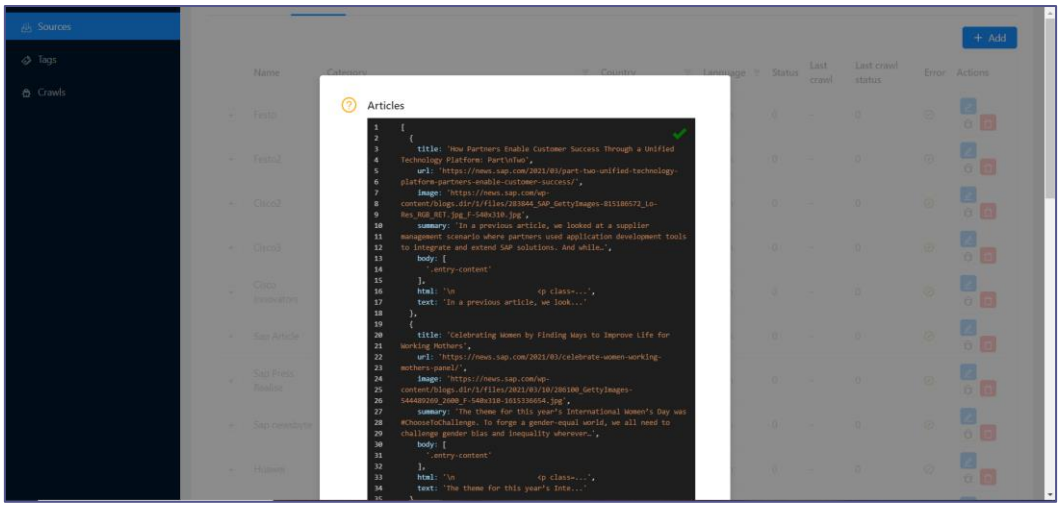

Figura 61: Salida de ejecución

Y ahora en la pestaña de *News* encontraremos las noticias que hemos extraído de esta nueva fuente. Aparecerán las primeras ya que han sido las últimas en extraerse esperando que un usuario las evalúe o descarte.

Title	Source	Category	Status	Relevance	Country	Language	Date	Actions
Succeeding Against the Odds	Sap Article	GENERAL	NEW	0	United States	English	a few seconds	
AGL Employees Go Digital with New iOS App and SAP BTP	Sap Article	GENERAL	NEW	0	United States	English	a few seconds	
Advancing Digital Skills When Needed Most	Sap Article	GENERAL	NEW	0	United States	English	a few seconds	
Personalized Medicine Shouldn't Be Cost Prohibitive: Life Science Business Networks Can Help	Sap Article	GENERAL	NEW	0	United States	English	a few seconds	
German Supermarket Launches Tiny Store for the Digital Age	Sap Article	GENERAL	NEW	0	United States	English	a few seconds	
Navigating Climate Change with Supply Chain	Sap Article	GENERAL	NEW	0	United States	English	a few seconds	

Figura 62: Noticias extraídas