
This is the **published version** of the bachelor thesis:

Ramallo López, Joan; Ortega Gil, Marc, dir.; Ortega Gil, Marc, dir. Aplicación Web Para La Reordenación De Resultados Sobre Búsquedas en Elasticsearch. 2023. (Enginyeria Informàtica)

This version is available at <https://ddd.uab.cat/record/280701>

under the terms of the  license

Aplicación web para la reordenación de resultados sobre búsquedas en ElasticSearch

Joan Ramallo López

Resumen- Se busca desarrollar una herramienta para el departamento de marketing de una empresa dedicada a la venta de productos en e-commerce, que pueda controlar la forma en cómo se muestran los productos dentro de su página web. La herramienta debe dar la posibilidad de ordenar, según su criterio, la lista de productos devuelta a raíz de las búsquedas que realicen los clientes. La herramienta debe ser visual e interactiva, y enseñar de forma inmediata los resultados de las modificaciones en la lista de productos. Debe ofrecer la posibilidad de reordenar los resultados manualmente, por referencia de producto o por estacionalidad/temporada. Además de poder modificar el estado de la base de datos, sea añadiendo más productos o editándolos. La herramienta debe ser flexible y ser capaz de funcionar sobre cualquier base de datos no relacional basada en el motor de búsqueda ElasticSearch. Actuará como una segunda capa intermedia entre la página web con la búsqueda de productos y la base de datos.

Palabras clave- Docker, ElasticSearch, índice,

Abstract- It seeks to develop a tool for the marketing department of a company dedicated to the sale of products in e-commerce that can control the way in which the products are displayed within its web page. The tool must give the possibility to sort, according to its criteria, the list of products returned as a result of the searches carried out by the customers. The tool must be visual and interactive, and immediately show the results of modifications to the product list. It must offer the possibility to reorder the results manually, by product reference or by season. In addition to being able to modify the state of the database, either by adding more products or editing them. The tool must be flexible and able to work on any non-relational database based on the ElasticSearch search engine. It will act as a second intermediate layer between the web page with product search and the database.

Keywords- Docker, ElasticSearch, Index,

1 Contexto del trabajo

En el creciente mundo del comercio electrónico, la búsqueda de productos en las páginas web se ha convertido en una función esenci-

al para el éxito de las tiendas online. La alta competencia que caracteriza a este sector implica que optimizar esta herramienta sea vital para aumentar las ventas y la fidelidad de los clientes. La capacidad de mostrar resultados rápidos y precisos, así como el producto buscado, incrementa la probabilidad de que los potenciales compradores tomen la decisión de adquirirlo en la página web y regresen en el futuro.

En la actualidad, las compras online han experimentado un crecimiento exponencial, convirtiéndose en una opción popular para que los consumidores adquieran una amplia variedad de productos desde

- E-mail de contacto: 1531610@uab.cat
- Menció n realizada: Tecnologías de la Información
- Trabajo tutorizado por: Marc Ortega Gil
- Departamento de Ingeniería de la Información y Comunicaciones
- Curso 2022/2023

sus hogares. Sin embargo, el aumento constante de la competencia y la expansión del comercio electrónico exigen que las tiendas online ofrezcan una experiencia de búsqueda efectiva y satisfactoria para sus clientes.

En este escenario, los motores de búsqueda desempeñan un papel fundamental en la navegación y descubrimiento de productos en las tiendas online. Estas potentes herramientas permiten a los usuarios realizar consultas y explorar catálogos de productos de manera eficiente, agilizando el proceso de búsqueda y ayudando a encontrar rápidamente los artículos deseados.

Existen circunstancias en las cuales el motor de búsqueda no puede adaptarse plenamente a los requisitos y condiciones de una empresa, y es necesario que intervenga el factor humano. Por ejemplo, en situaciones en las que se requiere patrocinar un catálogo específico acordado contractualmente con proveedores, o en eventos organizados por la empresa que demandan que determinados productos ocupen las primeras posiciones de la lista de resultados.

La herramienta propuesta en este artículo tiene como objetivo facilitar y potenciar dicho enfoque, al proporcionar una capa intermedia entre la página web de interés y el motor de búsqueda. Su finalidad es permitir que los usuarios personalicen de manera más exhaustiva los resultados obtenidos a partir de las búsquedas realizadas.

Este documento describe las características de la aplicación web desarrollada, presentando su funcionamiento y las decisiones adoptadas durante su desarrollo. Se explican en profundidad tanto su estructura interna como las funcionalidades que ofrece.

También se exponen los objetos y requisitos que se han tenido en cuenta para el desarrollo del proyecto y se detallan las diferentes funcionalidades y características que la aplicación debe poseer para satisfacer las necesidades identificadas.

Posteriormente, se presenta la metodología utilizada en el proceso de desarrollo de la aplicación web. Se destacan las técnicas y herramientas utilizadas durante el desarrollo.

2 Estado del arte

Los motores de búsqueda son herramientas que han evolucionado significativamente en los últimos años, adaptándose a las crecientes demandas y necesidades de los consumidores en el entorno del comercio electrónico.

Los motores de búsqueda en tiendas online utilizan algoritmos avanzados de indexación y búsqueda para recopilar y organizar grandes volúmenes de información sobre los productos disponibles. Estos algoritmos permiten una búsqueda rápida y eficiente, incluso en catálogos con miles o millones de productos. También tienen la capacidad de ofrecer

resultados personalizados y recomendaciones basadas en el comportamiento del usuario y en datos demográficos. Utilizando técnicas de filtrado colaborativo y aprendizaje automático, pudiendo sugerir productos adaptados a los intereses y preferencias individuales de cada usuario.

Además, utilizan búsqueda semántica, que va más allá de las simples palabras clave y tiene en cuenta el contexto y la intención del usuario. Esto permite una comprensión más precisa de las consultas de búsqueda y una correspondencia más relevante entre los productos y las necesidades del usuario.

Dado que la velocidad de carga y respuesta es crucial para la satisfacción del usuario, los motores de búsqueda en tiendas online se están enfocando en optimizar la velocidad y el rendimiento. Esto implica la implementación de técnicas de almacenamiento en caché, la optimización del código y el uso de arquitecturas escalables para manejar grandes volúmenes de datos.

Un ejemplo destacado en el campo es Algolia. Algolia es una herramienta especializada en la personalización de búsquedas que ha demostrado ofrecer excelentes resultados al aumentar la frecuencia de clics y compras en los sitios web que la utilizan. Algolia destaca por su capacidad para adaptar los resultados de búsqueda a las preferencias y necesidades individuales de cada usuario. Esto se logra mediante la utilización de técnicas avanzadas de filtrado colaborativo y aprendizaje automático, lo que permite ofrecer recomendaciones altamente relevantes y una experiencia de búsqueda personalizada.

3 Requisitos

En esta sección se concretan los requisitos funcionales que debe cumplir la aplicación para satisfacer las necesidades que se intentan cubrir.

La aplicación debe mostrar la lista de productos que contiene la base de datos. También deberá incorporar un campo de búsqueda que permita filtrar productos, al igual que se realizaría en una página web de compra online. Como ya se ha comentado, el motor de búsqueda utilizado para filtrar los productos será Elasticsearch. Debido a que el usuario realizará modificaciones sobre las búsquedas, se dejará seleccionar entre dos opciones de búsqueda: Teniendo en cuenta las modificaciones o sin tenerlas en cuenta.

Las modificaciones que debe poder realizar el usuario son tres.

- Posicionar manualmente un producto en una posición concreta de la lista de resultados.
- Excluir o esconder un producto de la lista de resultados.

- Permitir que productos de una determinada categoría ocupen las primeras posiciones del listado.

Las modificaciones se realizarán a partir de una búsqueda. Es decir, no serán modificaciones aplicadas siempre, sino que dependerán de la búsqueda realizada.

Además, la herramienta debe poder crear eventos calendarizados que permitan modificar la lista de resultados basándose en un rango de fechas. Por ejemplo, se podría definir que productos de la categoría “navidad” se mostrasen en las primeras posiciones del listado del 25 de noviembre al 15 de diciembre. Terminada la campaña, estos productos ocuparían la posición “habitual” en el listado.

Por último, un requisito importante es que la aplicación debe ser visual e intuitiva. Debe permitir posicionar de forma visual los productos. Es por eso que un producto debe poder posicionarse en una lista mediante drag-and-drop.

4 Objetivos

El objetivo del proyecto es desarrollar una aplicación web funcional que brinde a los usuarios la capacidad de manipular y gestionar la lista de productos en su base de datos. Para lograrlo, se deben desarrollar el Frontend, el Backend y la base de datos no relacional.

4.1 Frontend

El Frontend es la aplicación web que sirve como interfaz gráfica entre el usuario y la lógica de ordenación de productos. La aplicación web debe tener tres pantallas principales; Una pantalla para realizar búsquedas sobre los productos, otra pantalla para definir y manipular las reglas y una tercera para mostrar el conjunto de reglas definidas y administrarlas.

La interfaz de búsqueda tiene como objetivo mostrar de manera visual el contenido de la base de datos y ver los resultados finales que se mostrarían en la página web de interés. Pudiendo realizar diferentes búsquedas y modificar el catálogo. Esta interfaz tiene en cuenta todas las reglas que aplican según la búsqueda e indica qué reglas se están aplicando en todo momento.

La interfaz de reglas es la encargada de definir las reglas que modificarán el orden de aparición de los productos en la lista. Su enfoque está en mostrar una interfaz clara e intuitiva que permita gestionar fácilmente el listado de reglas. Por ende, no permitirá modificar el catálogo. Es un requisito importante que la interfaz de reglas sea lo más visual e interactiva posible. Los productos mostrados deben poder ser ordenados arrastrándolos a la posición deseada y debe generarse una regla acorde a

esto. Además de que cada producto tendrá la opción de fijarse en la posición actual que ocupa.

La interfaz de gestión de reglas permite administrar con facilidad las reglas definidas por el usuario. Muestra el listado de reglas y permite activarlas o desactivarlas, cambiar sus propiedades o eliminarlas.

4.2 Backend

La función del Backend es simple. Debe funcionar como una API web a la que la aplicación web enviará peticiones con tal de enviar y recibir datos. Debe ser capaz de gestionar correctamente el catálogo y aplicar las reglas definidas por el usuario. La API además tendrá conexión con la base de datos no relacional. Solo accederá a ella para realizar tres principales acciones:

- Obtener lista de productos a partir de la búsqueda del usuario.
- Añadir, modificar o eliminar productos.
- Añadir, modificar o eliminar reglas.

Como ya se ha mencionado, a partir del criterio de búsqueda que realice el usuario desde la aplicación web, la API debe ser capaz de aplicar las reglas oportunas y devolver el listado de productos.

La respuesta de la API debe seguir el mismo formato de mensajes que utiliza Elasticsearch, ya que la aplicación web es una capa intermedia entre la página web y la base de datos.

4.3 Base de datos

Se necesitará una base de datos no relacional para almacenar la información sobre productos y reglas. En este caso, se empleará el motor de búsqueda Elasticsearch como mecanismo de acceso a los datos y para facilitar la interacción con ellos.

El propósito principal de la base de datos será permitir las operaciones de inserción, modificación y eliminación de productos, así como la capacidad de agregar, modificar y eliminar reglas.

La estructura de almacenamiento de los datos se basará en ficheros JSON y se organizará mediante la creación de índices. Este enfoque es adecuado, ya que Elasticsearch es especializado en el procesamiento y búsqueda eficiente de este tipo de ficheros.

5 Infraestructura

En esta sección se indica la infraestructura del proyecto. Con el fin de implementar el proyecto, se ha utilizado contenedores Docker. Cada componente principal: Frontend, Backend y Base de Datos, se ejecuta en su propio contenedor, utilizando imágenes y volúmenes independientes.

La elección de Docker se basa principalmente en su capacidad de portabilidad, ya que los contenedores son ligeros, portátiles y pueden ser fácilmente trasladados entre diferentes entornos. Además, Docker proporciona beneficios en términos de consistencia, escalabilidad, aislamiento y velocidad.

El Frontend de la aplicación utiliza Nginx como servidor web HTTP, el cual se ejecuta dentro de un contenedor Docker. Nginx cumple el papel de intermediario entre el navegador del cliente y el Frontend de la aplicación, gestionando las solicitudes y respuestas HTTP. Nginx recopila los archivos estáticos del volumen definido en su configuración. La aplicación está escuchando en el puerto 8080.

El Backend utiliza la imagen oficial de Node.js para crear el contenedor de Docker. Tiene un volumen definido donde se guarda el código fuente. Está escuchando peticiones por el puerto 3000.

Se utiliza la imagen oficial de Elasticsearch por el motor de búsqueda que ofrece y por su capacidad de indexación, la cual es utilizada para almacenar los productos y reglas. Tiene un volumen definido para guardar los elementos indexados. Elasticsearch responde sobre el puerto 9200.

Por encima de los contenedores, se encuentra el sistema operativo de la máquina host, el cual es responsable de ejecutar los tres contenedores [Fig. 1]. Para poner en marcha los Dockers, hay que ejecutar los siguientes comandos en una terminal, teniendo el daemon de Docker activo:

- 1 Docker-compose up
- 2 Docker-compose build

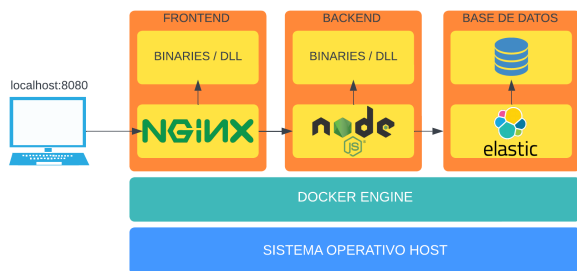


Fig. 1: Infraestructura de la aplicación

6 Reglas

Para controlar las modificaciones sobre las búsquedas se ha definido un motor de reglas, en la que cada regla es un conjunto de cambios definido por el usuario. Este apartado define en detalle las propiedades que puede tener una regla.

Cada regla está dividida en dos partes: Triggers y Eventos. Los triggers son las condiciones que se deben cumplir para que la regla aplique sobre la búsqueda y los Eventos son los cambios que se realizarán sobre la lista de resultados. Hay tres tipos

de triggers: Por condición de búsqueda, franja de tiempo y categoría.

- Condición de búsqueda: Es el trigger principal, si la búsqueda realizada cumple con la Query definida, la regla aplicará. Se puede seleccionar la condición a cumplir, siendo las opciones: "Contains", "Is", "Starts with" y "Ends with" más el texto.
- Franja de tiempo: Permite la gestión de eventos calendarizados. Permite definir un intervalo entre dos fechas. Si la fecha en que se realiza la búsqueda está en la franja de tiempo indicada, la regla aplicará.
- Categoría: Permite controlar que una regla solo se aplique si el usuario ha seleccionado una categoría dentro de la página web. Por ejemplo, si el usuario ha seleccionado la categoría de ropa.

Cada trigger puede definirse más de una vez en cada regla. Si no se define un trigger de uno de los tres tipos, se considerará que siempre se cumple. Entre diferentes triggers hay una relación AND, y entre triggers del mismo tipo hay una relación OR. Por ejemplo, si se definen varias franjas de tiempo, con encontrarse dentro de una de las franjas es suficiente. Una regla sin ningún trigger definido se aplicará siempre.

Los eventos pueden ser de cuatro tipos: Fijar un producto en una posición, esconder un producto, promover una categoría o filtrar resultados.

- Fijar: Permite fijar en una posición concreta de la lista un producto.
- Esconder: Elimina de la lista de resultados el producto.
- Promover: Permite que los productos de la categoría indicada ocupen posiciones más altas dentro del listado de productos.
- Filtrar: Condiciona la búsqueda para que solo devuelva los productos que cumplan el filtro.

Al igual que los triggers, se pueden definir tantos de cada tipo como se necesite. Aun así, el motor utiliza su lógica para evitar conflictos entre eventos y casos excepcionales. En el caso de que se intente fijar y esconder un producto simultáneamente, prevalecerá el evento de esconder. Además, un producto fijado se añadirá al final de la lista de resultados si la posición es mayor al número de resultados. Por último, un producto ocultado no ocupa una posición real en la lista. Si un producto es escondido, el resto de productos avanzan una posición, menos los productos fijados.

Más de una regla puede estar activa a la vez y crear conflictos entre sí, en caso de que dos reglas quieran fijar un producto en una posición en concreto, dependerá del orden en que se adquieren las reglas

de la base de datos para determinar qué producto ocupará esa posición.

7 Índices

Se han definido dos índices en la base de datos no relacional, uno para guardar los productos y otra para guardar las reglas.

índice productos:

```

1  {
2    ID: {type: 'keyword'},
3    title: {type: 'search_as_you_type'},
4    image: {type: 'keyword'},
5    description: {type: 'search_as_you_type'}
6    URL: {type: 'keyword'},
7    creationDate: {type: 'date'},
8    categories: {
9      properties: {
10       category: { type: 'keyword' },
11      },
12    },
13    punctuation: { type: 'number' },
14  },

```

índice reglas:

```

1  {
2    lastUpdate: {type: 'date'},
3    enabled: {type: 'boolean'},
4    ruleID: {type: 'keyword'},
5    description:{type: 'text'},
6    validity: {
7      properties: {
8        startDate: {type: 'date'},
9        endDate: {type: 'date'},
10      },
11    },
12    conditions: {
13      properties: {
14        conditionID: {type: 'keyword'},
15        conditionText: {type: 'text'},
16      },
17    },
18    categories: {
19      properties: {
20        categoryText: {type: 'keyword'},
21      },
22    },
23    consequences: {
24      properties: {
25        objectPin: {
26          properties: {
27            objectID: {type: 'keyword'},
28            position: {type: 'integer'},
29          },
30        },
31        objectHide: {
32          properties: {
33            objectID: {type: 'keyword'},
34          },
35        },
36        categoryBoost: {
37          properties: {
38            categoryID: {type: 'keyword'},
39            boostValue: {type: 'integer'},
40          },
41        },
42        filterResult: {
43          properties: {
44            filterField: {type: 'keyword'},
45            filterCond: {type: 'keyword'},

```

```

46         filterText:{type: 'text'}
47       },
48     },
49   },
50 },
51 },

```

El tipo de dato 'search_as_you_type' es un tipo de dato que se comentará más adelante. En esencia equivale a type: 'text', pero ofrece propiedades de búsqueda adicionales.

En el índice de reglas, las propiedades validity, conditions y categories corresponden a los tres tipos de triggers, mientras que las cuatro propiedades dentro de consequences responden a los tipos de eventos. Los campos objectID y categoryID hacen referencia a los campos ID y categories del índice de productos.

8 Metodología

Para el desarrollar el proyecto se ha utilizado la metodología Agile. Al ser un proyecto realizado individualmente, un enfoque iterativo e incremental es óptimo para desarrollar paso a paso el software necesario. Utilizar esta metodología aporta principalmente flexibilidad, ya que fácilmente puede haber cambios en la planificación, fácil gestión al utilizar ciclos de desarrollo cortos, mayor transparencia y mejor reacción ante los resultados, ya que durante el desarrollo habrá módulos funcionales que podrán ser valorados en calidad y desempeño.

Para la coordinación con el tutor, se han realizado reuniones de seguimiento mensuales para informar del estado del proyecto y recibir feedback de posibles cambios o mejoras que pueden realizarse. También para informar de cambios en la planificación y discutir problemas encontrados.

9 Planificación

Para desarrollar el proyecto se ha empezado definiendo los Dockers. Se pusieron en funcionamiento con el mínimo de código necesario para que se comunicasen entre sí. Para, de este modo, asegurarse de que las imágenes estaban bien configuradas. Una vez en funcionamiento, se empezó a desarrollar el Frontend y el Backend de manera lineal para construir el motor de reglas [Fig. 2].

Paralelamente, cuando estuvo disponible el contenido de la base de datos, se implementó en el Backend el proceso de inserción de datos en el índice de productos.

Por último, cuando todas las funcionalidades estuvieron implementadas, se procedió a crear una buena interfaz de usuario y a la corrección de bugs.

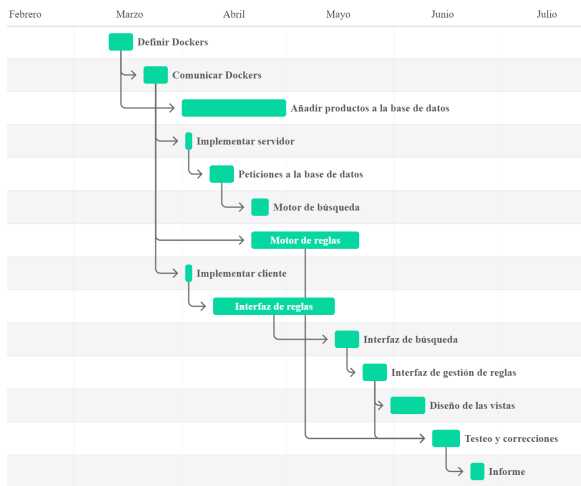


Fig. 2: Planificación inicial

10 Implementación del Frontend

El Frontend se ha desarrollado utilizando Vue.js. Vue.js es un Framework de JavaScript de código abierto para la construcción de interfaces de usuario y aplicaciones de una sola página. Se ha elegido esta tecnología, ya que está especializada en crear componentes. La intención ha sido definir un componente para los productos de manera que puedan ser manipulados como un conjunto de iguales independiente de sus características. Y del mismo modo, definir un componente para las reglas, de manera que pueda normalizarse su manipulación y encapsular las características propias de cada una.

El componente principal controla las tres vistas principales de la aplicación. Consta de una barra de navegación que permite cambiar de vista y controla cuál de los tres componentes correspondientes a la vista se muestra en cada momento [Fig. 3].

```

1 <template>
2   <div class="MAINCLASS">
3     <div class="navBar">
4       <navigationBar/>
5     </div>
6     <div class="addRuleView" v-if="navTab === 0">
7       <ruleCreationView/>
8     </div>
9     <div class="rulesView" v-if="navTab === 1">
10      <ruleView/>
11    </div>
12    <div class="productView" v-if="navTab === 2">
13      <searchView/>
14    </div>
15  </template>

```

10.1 Interfaz de reglas

La pestaña de reglas sirve para definir las reglas. La pantalla está dividida en tres partes: El listado de productos, la sección de triggers y la sección de eventos. A partir de la búsqueda realizada, se muestra la lista de resultados, cada artículo del listado es un componente interactivo. Posee dos botones, uno para fijar el producto en su posición y otro para

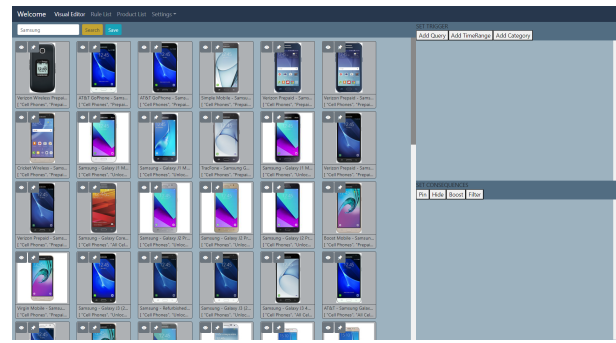


Fig. 3: Interfaz de reglas

esconder el producto [Fig. 4]. Además, puede ser arrastrado gracias a la propiedad drag-and-drop de la lista hasta la posición deseada para fijarlo en esa posición. Cada producto muestra su imagen, título y categorías.

Para hacer la reordenación de la lista se ha utilizado Vue.Draggable [2]. Vue.Draggable es un complemento de Vue.js que permite la implementación de funcionalidades de arrastrar y soltar elementos.



Fig. 4: Ejemplo de un objeto fijado y otro oculto

A la derecha de la vista está la sección para definir las características de la regla. Cada botón corresponde a un trigger o un evento. Al dar clic a un botón, se añade a la lista de trigger o de eventos la función seleccionada. También se añade a la lista de eventos un elemento al fijar un producto, ya sea con su respectivo botón o al arrastrarlo a una posición o al esconderlo.

Todos los elementos tienen la opción de editar y eliminar. Al editar un elemento, se abre un diálogo de texto que permite editar las propiedades. Se ha creado usando Modals de Bootstrap [1]. Se ha utilizado Vue.datepicker.ui [4] para gestionar los rangos de tiempo. Al editar el trigger se muestran dos calendarios, uno para seleccionar la fecha de inicio y otro para la fecha de fin [Fig. 5] [Fig. 6].

Para las llamadas HTTP a la API se ha utilizado Axios [3]. Axios es una biblioteca que facilita el manejo de solicitudes HTTP. Se ha utilizado gracias a su facilidad de uso y manejo de promesas.

Por último, para guardar la regla, está el botón Save. Al pulsarlo se muestra un diálogo que enseña el JSON que será guardado en el índice de reglas. También permite añadirle una descripción e indicar



Fig. 5: Listado con reglas y eventos definidos

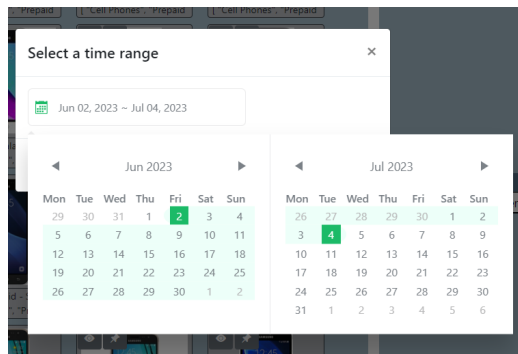


Fig. 6: Diálogo para modificar el rango de tiempo

si la regla estará activa o inactiva.

10.2 Interfaz de gestión de reglas

La función de la pantalla de gestión de reglas tiene como objetivo administrar las reglas ya creadas. Muestra una lista de todas las reglas de la base de datos. De cada regla se muestra su descripción, si está deshabilitada o no, su ID y la última fecha de modificación [Fig. 7].

Adicionalmente, las reglas tienen cuatro botones: Tienen la opción de habilitarse o deshabilitarse, mostrar los detalles de la regla, editar la regla o eliminar la regla. Editar una regla carga la regla en la pantalla de reglas, mientras que mostrar detalles de la regla enseña el JSON guardado en la base de datos.

10.3 Interfaz de búsqueda

La pestaña de búsqueda sirve para recrear los resultados que se verían en la página web de compras. Tiene un buscador para realizar búsquedas y muestra los resultados. Los productos pueden ser

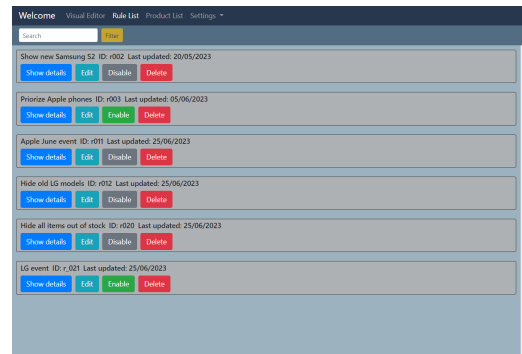


Fig. 7: Interfaz de gestión de reglas

editados, eliminados y se pueden crear nuevos. A la derecha de la pantalla se muestra en todo momento las reglas que se están aplicando [Fig. 8].

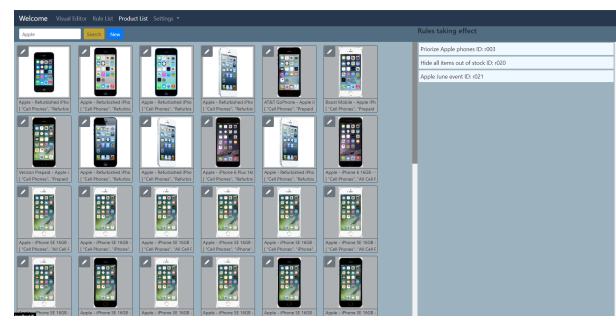


Fig. 8: Interfaz de búsqueda

11 Implementación del Backend y Elasticsearch

El Backend se ha desarrollado utilizando Node.js [5] debido a sus ventajas en términos de velocidad, eficiencia y escalabilidad. Además, su facilidad de uso permite un desarrollo rápido de prototipos. Para montar el servidor se ha utilizado Koa [6], un framework de Node.js que facilita la creación de servidores al añadir una capa de abstracción.

El servidor recibe todas las peticiones del Frontend y las procesa, accediendo a los índices de la base de datos. Para acceder a la base de datos se ha utilizado el framework de Elasticsearch para Node.js [7]. También tiene los procesos que hacen funcionar al motor de reglas.

Gracias a que el índice de productos tiene las propiedades title y description del tipo 'search_as_you_type' [8], la búsqueda realizada tiene propiedades de predicción, y permite la visualización de resultados en tiempo real mientras se escribe.

Para que la predicción funcione, es necesario especificar en el cuerpo de la solicitud las siguientes propiedades: "title", "title._2gram" y "title._3gram" para el campo "title" y "description", "description._2gram" y "description._3gram" para el campo "description".

```

1  async function search(term, offset = 0) {
2    var index = 'products'
3    var type = '_doc'
4    const body = {
5      from: offset,
6      query: {
7        multi_match: {
8          query: term,
9          type: "bool_prefix",
10         fields: [
11           "title",
12           "title._2gram",
13           "title._3gram",
14         ]
15       }
16     },
17     highlight: { fields: { text: {} } },
18     size: 100
19   }
20   return client.search({ index, type, body })
21 }

```

Para obtener la base de datos provisional que se ha utilizado para realizar pruebas y validar el correcto funcionamiento de las reglas, se desarrolló un script escrito en Ruby [9] que recorre todos los registros de la base de datos de ejemplo que proporciona Algolia, y cada registro es guardado en un fichero.

```

1  require 'json'
2  require 'algoliasearch'
3
4  Algolia.init(application_id: '1DNSPFYD99',
5               api_key: 'secretApiKey')
6
7  index = Algolia::Index.new('dev_Phones')
8  hits = []
9  index.browse do |hit|
10    hits.push(hit)
11  end
12  File.write('products', hits.to_json)

```

Para añadirlos en la base de datos, en el Backend se desarrolló un proceso que lee el fichero y lo carga en el índice de productos.

12 Evaluación de resultados

Luego de meses de desarrollo se ha conseguido desarrollar una aplicación web que cubre con los requisitos iniciales. Tiene una interfaz sencilla, intuitiva y visual que era uno de los aspectos más importantes que cubrir. La aplicación es funcional y puede ser usada fácilmente en entornos reales.

La herramienta tiene potencial para crecer y cubrir otros campos relacionados con el mundo del e-commerce. Es posible desarrollar módulos de análisis de datos y gestión de múltiples bases de datos simultáneamente, lo que brinda oportunidades adicionales de crecimiento y adaptabilidad.

En un futuro, se planea desarrollar funcionalidades adicionales que mejoran el uso de la aplicación y que quedan fuera del alcance del proyecto. Por ejemplo, un proceso para añadir, a partir de un fichero, productos de manera masiva en un índice.

13 Conclusiones

En este proyecto se ha desarrollado el "full stack" de una aplicación. Lo cual me ha servido para ser más versátil y conocedor de las diferentes capas de una aplicación.

Además, he trabajado con Docker y Elasticsearch, dos herramientas muy potentes y ampliamente utilizadas en el mundo profesional que no he podido conocer a fondo durante mis años de universidad.

Después de finalizar este proyecto, puedo concluir diciendo que el resultado ha sido exitoso y se ha logrado elaborar una aplicación web de calidad. Realizar este trabajo ha supuesto un reto que me ha servido para expandir mis conocimientos y adquirir habilidades blandas, al tener que responsabilizarme y dirigir un proyecto por mí mismo. Además, me ha aportado confianza y ha mejorado mis competencias como ingeniero informático.

Agradecimientos

Quiero dar mi gratitud a mi tutor Marc Ortega por toda la ayuda en el desarrollo de este proyecto y por guiarme en las decisiones de desarrollo que he tomado. También a mis amigos y compañeros cercanos que me han apoyado y ayudado estos últimos meses.

Referencias

- [1] Bootstrap modal. Bootstrap [Online] Available: <https://getbootstrap.com/docs/4.0/components/modal/>
- [2] Vue.draggable. GitHub. [Online] Available: <https://github.com/SortableJS/Vue.Draggable>
- [3] Llamadas HTTP con Axios. Axios. [Online] Available: <https://axios-http.com/es/docs/intro>
- [4] Vue.datepicker.ui. GitHub. [Online] Available: <https://github.com/edisdev/vue-datepicker-ui>
- [5] About Node.js. Node. [Online] Available: <https://nodejs.org/en>
- [6] Koa for Node.js. Koa. [Online] Available: <https://koajs.com/>
- [7] Elasticsearch JavaScript Client. Elastic. [Online] Available: <https://www.elastic.co/guide/en/elasticsearch/client/javascript-api/current/index.html>
- [8] Using "search-as-you-type". Elastic [Online] Available: <https://docs.elastic.co/search-ui/guides/using-search-as-you-type>
- [9] Lenguaje de Programación Ruby. Ruby [Online] Available: <https://www.ruby-lang.org/es/>