RECUPERACIÓN DE INFORMACIÓN Y RECOMENDACIONES EN LA WEB

ENTREGA FINAL 2018

GRUPO 13

NAHIR TOLEDO OLIVERA 4.940.601-9
NICOLÁS SÁNCHEZ CARRERAS 4.514.131-2
PABLO PIAZZE DE LOS SANTOS 4.024.148-8

DOCENTE: LIBERTAD TANSINI

UNIVERSIDAD DE LA REPÚBLICA | FACULTAD DE INGENIERÍA

CONTENIDO

1. Introducción

En el presente documento se abordará la aplicación de las técnicas provistas por el curso para recuperación de información de la web aplicadas a un problema de interés general.
El informe describe una primera aproximación a una solución integral planteada por el grupo para uno de los problemas que aqueja a la población mundial en los tiempos corrientes, donde el consumismo mundial plantea nuevos desafíos día a día.

2. Problema y Motivación

En los últimos tiempos, se está prestando mucha atención en disminuir el desperdicio de alimentos en los hogares. Creemos que una de las posibles causas del desperdicio, es el desconocimiento por parte de las personas de qué comidas se pueden elaborar con los ingredientes disponibles en el hogar por pocos que sean.

También existen nuevos diagnósticos médicos, mediante los cuales se han descubierto ciertos síntomas de enfermedades producidas por la no tolerancia hacia ciertos alimentos (celíacos, intolerancia a la lactosa, etc.), así como nuevas prácticas que rechazan la utilización y consumo de ingredientes/productos de determinado origen como el animal (veganismo, vegetarianismo, etc.).

Por otro lado, hoy en día existen múltiples plataformas en las cuales las personas comparten información a nivel global (Facebook, Twitter, Youtube, Instagram, etc.). Dentro de esa información se encuentra la publicación de recetas de comida, las cuales de ser centralizadas pueden aportar una alternativa interesante a consultar por parte de los hogares al momento de decidir qué elaborar con los ingredientes disponibles. Esta misma plataforma puede permitir a los usuarios finales consultar distintas preparaciones compatibles con el tipo de alimentación que desean llevar adelante.

3. Enfoque de la solución

El enfoque que se plantea en esta solución, es presentar al usuario una aplicación mediante la cual pueda consultar distintas recetas/elaboraciones en base a una lista de ingredientes disponibles y otros criterios de utilidad.

Existen infinidades de lugares en los cuales se publican recetas de comida en la actualidad, como una primera aproximación, nos enfocaremos en centralizar la información de un par de páginas web las cuales fueron estudiadas previamente para evaluar su factibilidad.

Las páginas sobre los cuales recuperaremos la información es la siguiente:

- Cocina Casera y Fácil:
  - https://www.cocinacaserayfacil.net/todas-las-recetas/
- Recetas Comunidad Celíacos del Uruguay:
  - http://comunidadceliaca.org/recetas

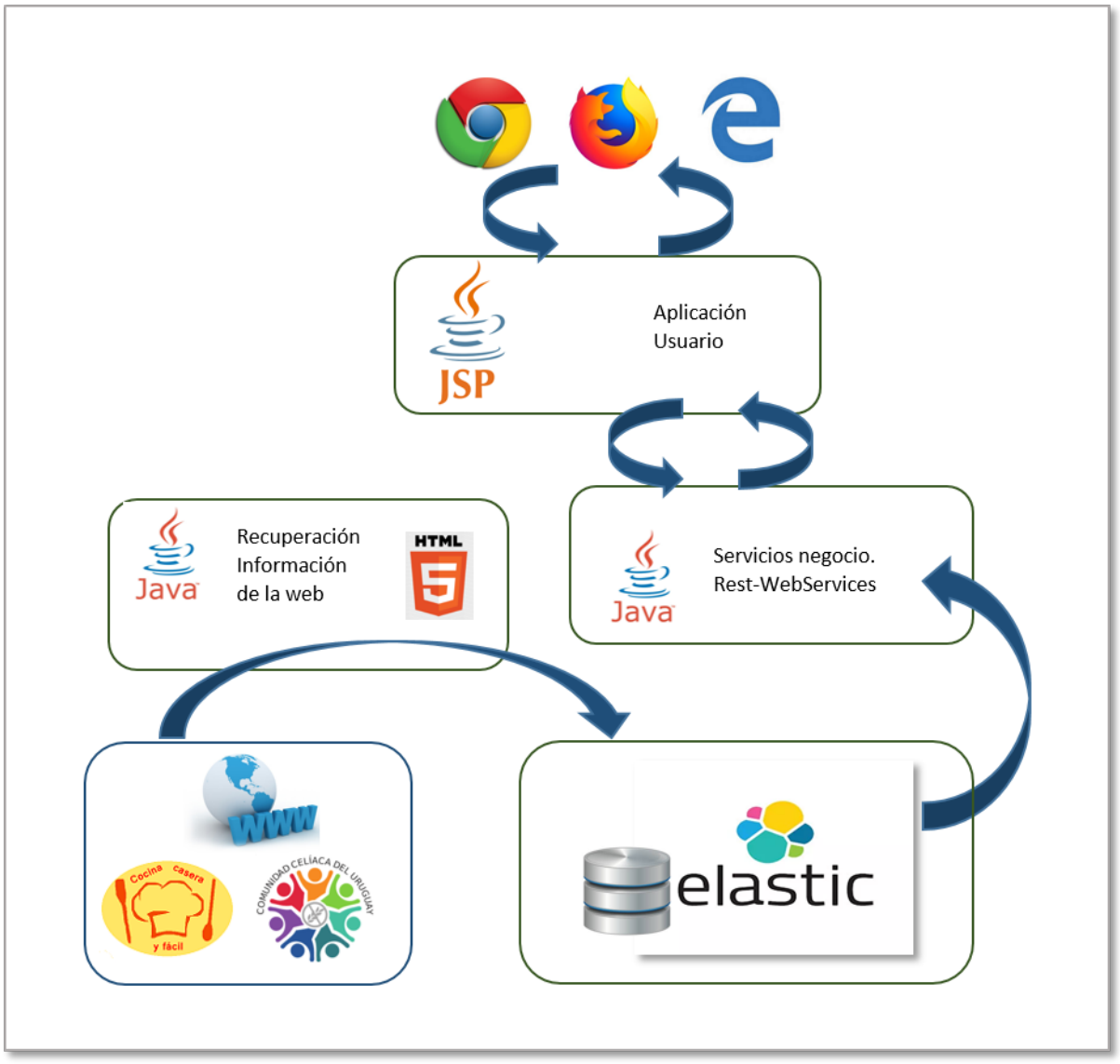
El usuario contará con la posibilidad de buscar recetas utilizando los siguientes criterios:

- Nombres de ingredientes incluidos
- Nombres de ingredientes excluidos
- Celíacos (indicativo de comida apta para celíacos)

A modo de ejemplo: un usuario con rechazo a los “huevos” podrá consultar qué preparaciones incluyen “acelga, cebolla, morrón”, pero no incluya “huevos”.
El sistema buscará dentro de sus documentos posibles aquellos que incluyan la palabra “acelga”, “cebolla” y “morrón”, por otro lado los que contengan “huevos” y restará al primer grupo aquellos que pertenezcan al segundo grupo recuperado. En caso de existir documentos que cumplan esas condiciones, desplegará en pantalla las recetas correspondientes.
Un usuario celíaco, podrá consultar unicamente recetas categorizadas aptas para celíacos.

4. Diseño e implementación

El diseño de la aplicación se ha dividido en distintos componentes, cada uno de los cuales se encarga de un problema en particular. A continuación se presenta un esquema general de la solución y una breve descripción de cada uno de los componentes:



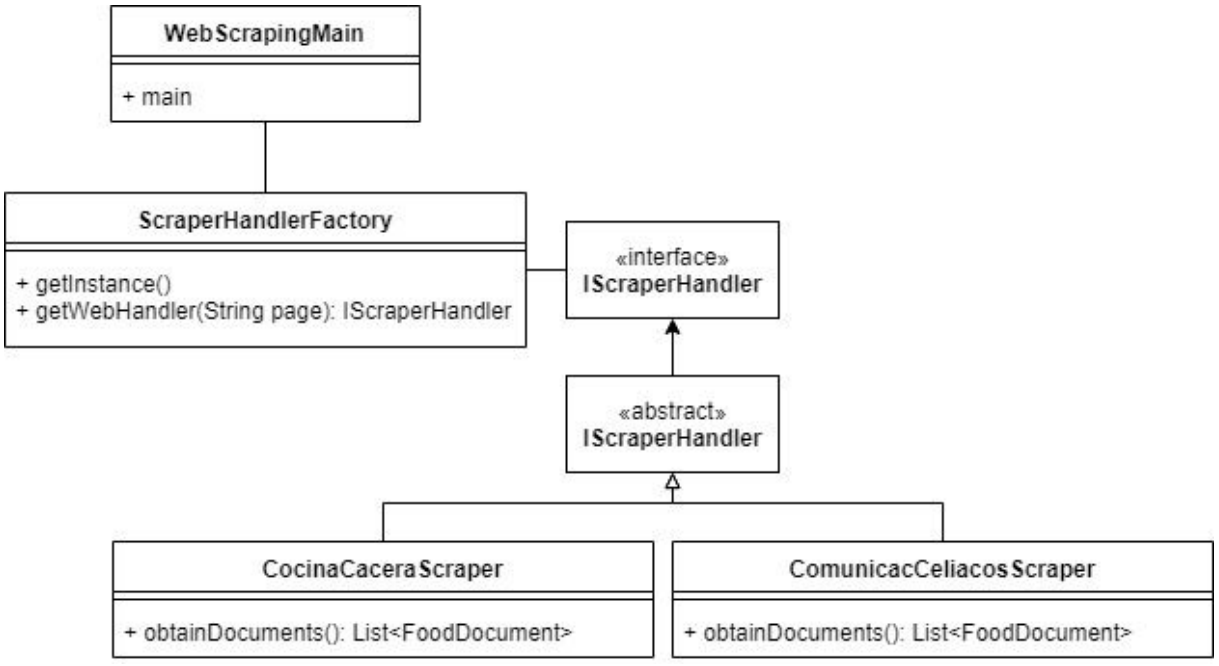
4.1. Recuperación Información de la web

Este servicio será el encargado de recuperar información de la web y almacenarla en la base de datos provista por ElasticSearch.
En base a las páginas web definidas, utilizando las librerías jsoup[1], procesa las páginas html recuperando de ellas cada una de las recetas.
Para cada una de las recetas procesadas, se recupera la siguiente información (post):

- Nombre
- Ingredientes
- url de la publicación
- indicativo de celíaco

Dependiendo de la página de la cual estemos recuperando la información, se completarán indicativos el indicativo de celíaco. Por ejemplo, al recuperar información del grupo de comidas aptas para celíacos, almacenaremos junto al documento un indicativo de apta para celíaco.
Esta información será recuperada en formato JSON [5] plano, el cual se almacenará en la base de datos de ElasticSearch[2] con su respectivo índice que llamaremos "*foodindex*" (la herramienta ElasticSearch y su aplicación en java a través de su API se explicará en los siguientes puntos de este documento).

El proceso tiene los siguientes componentes:



- **WebScrapingMain**
  Es el proceso principal, el cual en base a una lista configurada de páginas web, solicita para cada una de ellas la lista de recetas que contiene y el resultado obtenido lo almacena en la base de datos elasticSearch.
- **ScraperHandlerFactory**
  Es la clase encargada de instanciar en base al identificador de una página web, la clase encargada de realizar el scraping de las recetas.
- **IScraperHandler**
  Interfaz de la cual extienden los distintos tipos de clases encargadas del scraping.
- **ScraperHandler**
  Clase que extiende de la interfaz del punto anterior, la cual tiene implementada una serie de métodos comunes para todos los scapers.
- **CocinaCaceraScraper**
  Clase encargada de realizar el scraping de la web "www.cocinacaserayfacil.net".
  En una primera instancia abre la página principal de recetas "https://www.cocinacaserayfacil.net/todas-las-recetas/" recorriendo la lista de resetas existentes y para cada una de ellas abre el link asociado a la reseta.
  Cada link a una receta es abierto utilizando la libreria jsoup (por ejemplo https://www.cocinacaserayfacil.net/aguacates-rellenos-de-atun-y-huevo/), el cual es procesado para recuperar el nombre y la lista de ingredientes de la reseta.
  Este proceso se repite para cada una de las recetas de la página principal.
- **ComunicacCeliacosScraper**
  Al igual que la clase anterior, esta clase se encargará de procesar las recetas de la web "http://comunidadceliaca.org/recetas ". De la página principal obtiene cada uno de los links a las distintas recetas, las cuales son accedidas para procesar la información de las mismas.

4.2.Elastic Search

Motor de búsqueda de texto completo, distribuido y con capacidad de Stemming [7] y Fuzzy matching [8] para mejorar la precisión de los filtros sobre las publicaciones con errores lexicográficos.
Elasticsearch es un servidor de búsqueda basado en Lucene [3]. Provee un motor de búsqueda de texto completo, distribuido y con capacidad de multi-tenencia con una

interfaz web RESTful [4] y con documentos JSON [5]. Elasticsearch está desarrollado en Java y está publicado como código abierto bajo las condiciones de la licencia Apache [6].

Ventajas de usar Elasticsearch:

- Construir sobre la base de Lucene, biblioteca de recuperación de información.
- Búsqueda de texto completo.
- Fuzzy Searching, permite buscar con errores de ortografía (lexicográficos).
- Autocompletado y búsqueda instantánea.
- Orientado a documentos, almacena entidades complejas del mundo real como documentos JSON estructurados e indexa todos los campos de forma predeterminada, con un resultado de rendimiento superior.
- Velocidad, puede ejecutar consultas complejas extremadamente rápido.
- Escalabilidad, sistema distribuido por naturaleza y puede escalar horizontalmente, lo que permite extender los recursos y equilibrar la carga entre los nodos de un clúster.
- Búsqueda estructurada, no tiene esquemas, acepta documentos JSON e intenta detectar la estructura de datos, indexar los datos y hacer que se pueda buscar.
- Registro de datos, registra cualquier cambio realizado en registros de transacciones en múltiples nodos en el clúster para minimizar la posibilidad de pérdida de datos.
- Afinación precisa de consultas, tiene una poderosa DSL basada en JSON, que permite a los equipos de desarrollo construir consultas complejas y afinarlas para recibir los resultados más precisos de una búsqueda.
- API relajado, las acciones se pueden realizar utilizando una API Restful simple.
- Enfoque distribuido, Los índices se pueden dividir en fragmentos, con cada fragmento capaz de tener cualquier cantidad de réplicas. Las operaciones de enrutamiento y reequilibrio se realizan automáticamente cuando se agregan nuevos documentos.
- Uso de facetas, búsqueda facetada es más sólida que una búsqueda de texto típica, lo que permite a los usuarios aplicar una cantidad de filtros a la información e incluso tener un sistema de clasificación basado en los datos. Esto permite una mejor organización de los resultados de búsqueda.
- Multi Alquiler, múltiples clientes o usuarios con colecciones separadas de documentos, y un usuario nunca debería ser capaz de buscar documentos que no le pertenecen.

4.3.Servicio Negocio

El servicio de negocio será un web service REST, recibe como entrada la información de los criterios de búsqueda (lista de ingredientes a buscar, lista de ingredientes que no quiero usar e indicativo de celíaco), buscar las recetas que cumplan los criterios deseados, procesar información extra de ser necesario y retornar la lista final de recetas.

Con la información recibida realizará la siguiente secuencia de actividades:

1. Utilizando la lista de ingredientes y el indicativo, consultará (mediante elastic) los tag que contienen todas las palabras correspondiente a cada ingrediente y que estén con o sin el indicativo ind_celiac dependiendo del valor especificado.
2. Utilizando la lista de ingredientes que no se desean utilizar, consultará los tag que contienen al menos una de las palabras de cada ingrediente, y el indicativo de la misma forma que el punto anterior.
3. De la lista obtenida en el punto 1, eliminará cada uno de los tags obtenidos en el punto 2. De esta forma obtiene la lista final de tags que cumplen con los criterios de entrada.

Los servicios de negocio implementados en java se comunican con el motor de búsqueda Elasticsearch para realizar las consultas según los filtros seleccionados.

Esto se implementa instanciando en una clase java un cliente de tipo TransportClient de la librería org.elasticsearch.client.transport.TransportClient, que conecta de forma remota a un clúster de Elasticsearch utilizando el módulo de transporte. La dirección que se agrega a este cliente es la http://localhost:9300/:

```
//inicializamos el cliente elastic
InetAddress addr = null;
try {
        addr = InetAddress.getByName("localhost");
} catch (UnknownHostException e) {
        e.printStackTrace();
}
Settings settings = Settings.builder().put("cluster.name", "elasticsearch").build();
TransportClient client = new PreBuiltTransportClient(settings);
client.addTransportAddress(new TransportAddress(addr, 9300));
```

El puerto es el 9300 porque el negocio se encuentra implementado en Java, pero por lo general el puerto utilizado para la llamada al Elasticsearch es el 9200 [10].

Por ejemplo si se desea buscar recetas que cumplan los siguientes criterios:

- contenga el string a, el b y el c (donde a, b y c son ingredientes)
- no contenga el string not_a, ni el not_b, ni el not_c (donde not_a, not_b y not_c son ingredientes)
- considere el booleano 'celiaco'

Se debe implementar el siguiente código java:

```
//El nombre del índice definido al cargar la BBDD del Elasticsearch es "foodindex".
BoolQueryBuilder queryBuilder = QueryBuilders.boolQuery().must(QueryBuilders.matchAllQuery());
                        queryBuilder.must(QueryBuilders.matchQuery("ingredients", a));
                        queryBuilder.must(QueryBuilders.matchQuery("ingredients", b));
                        queryBuilder.must(QueryBuilders.matchQuery("ingredients", c));
                        queryBuilder.mustNot(QueryBuilders.matchQuery("ingredients", not_a));
                        queryBuilder.mustNot(QueryBuilders.matchQuery("ingredients", not_b));
                        queryBuilder.mustNot(QueryBuilders.matchQuery("ingredients", not_c));
                        queryBuilder.must(QueryBuilders.matchQuery("indCeliac", celiaco));

                        SearchResponse response = client.prepareSearch("foodindex")
                                        .setTypes("food")
                                        .setSearchType(SearchType.DFS_QUERY_THEN_FETCH)
                                        .setQuery(queryBuilder)
                                .setFrom(fromRtn).setSize(sizeRtn)
                                .setExplain(false) //si está prendido recupera en el JSON explicación del
resultado recuperado
                                .execute()
                                .actionGet();
```

La consulta en Elasticsearch será la siguiente (esto es transparente para nosotros al utilizar la api para java):

```
{
  "bool" : {
    "must" : [
     {
       "match_all" : {
         "boost" : 1.0
       }
     },
     {
       "match" : {
         "ingredients" : {
           "query" : "huevo",
           "operator" : "OR",
           "prefix_length" : 0,
           "max_expansions" : 50,
           "fuzzy_transpositions" : true,
           "lenient" : false,
```

```
      "zero_terms_query" : "NONE",
      "auto_generate_synonyms_phrase_query" : true,
      "boost" : 1.0
    }
   }
  },
  {
   "match" : {
    "ingredients" : {
      "query" : "azúcar",
      "operator" : "OR",
      "prefix_length" : 0,
      "max_expansions" : 50,
      "fuzzy_transpositions" : true,
      "lenient" : false,
      "zero_terms_query" : "NONE",
      "auto_generate_synonyms_phrase_query" : true,
      "boost" : 1.0
    }
   }
  },
  {
   "match" : {
    "ingredients" : {
      "query" : "vainilla",
      "operator" : "OR",
      "prefix_length" : 0,
      "max_expansions" : 50,
      "fuzzy_transpositions" : true,
      "lenient" : false,
      "zero_terms_query" : "NONE",
      "auto_generate_synonyms_phrase_query" : true,
      "boost" : 1.0
    }
   }
  },
  {
   "match" : {
    "indCeliac" : {
      "query" : true,
      "operator" : "OR",
      "prefix_length" : 0,
      "max_expansions" : 50,
      "fuzzy_transpositions" : true,
      "lenient" : false,
      "zero_terms_query" : "NONE",
      "auto_generate_synonyms_phrase_query" : true,
      "boost" : 1.0
```

```
      }
     }
    }
   ],
   "must_not" : [
    {
     "match" : {
      "ingredients" : {
       "query" : "harina",
       "operator" : "OR",
       "prefix_length" : 0,
       "max_expansions" : 50,
       "fuzzy_transpositions" : true,
       "lenient" : false,
       "zero_terms_query" : "NONE",
       "auto_generate_synonyms_phrase_query" : true,
       "boost" : 1.0
      }
     }
    },
    {
     "match" : {
      "ingredients" : {
       "query" : "sal",
       "operator" : "OR",
       "prefix_length" : 0,
       "max_expansions" : 50,
       "fuzzy_transpositions" : true,
       "lenient" : false,
       "zero_terms_query" : "NONE",
       "auto_generate_synonyms_phrase_query" : true,
       "boost" : 1.0
      }
     }
    },
    {
     "match" : {
      "ingredients" : {
       "query" : "almidón",
       "operator" : "OR",
       "prefix_length" : 0,
       "max_expansions" : 50,
       "fuzzy_transpositions" : true,
       "lenient" : false,
       "zero_terms_query" : "NONE",
       "auto_generate_synonyms_phrase_query" : true,
       "boost" : 1.0
      }
```

```
      }
     }
    ],
    "adjust_pure_negative" : true,
    "boost" : 1.0
  }
}
```

La consulta es una Bool Query donde se le especifican los elementos must y must_not que se deben matchear.
En la consulta se buscó por los campos ingredients e indCeliac. Para el campo ingredients se busca que contengan los ingredientes a, b y c y que no contengan los ingredientes not_a, not_b y not_c; para el campo indCeliac se busca que el booleano coincida con el recibido desde el cliente web.

Elasticsearch retorna todos los posts (“hits”) que coincidan con la consulta:

```
//JSON resultado de la búsqueda
{
  "hits": {
    "total": 2,
    "hits": [
      {
        "_index": "foodindex",
        "_type": "food",
        "_id": "4",
        "_source": {
          "id": 4, "name": "Plantillas o vainillas - Comunidad Celíaca del Uruguay",
        "ingredients": "<receta 4>",  "url": "http://comunidadceliaca.org/recetas/plantillas-o-vainillas-215",
          "image": null,  "indCeliac": true
        }
      },
      {
        "_index": "foodindex",
        "_type": "food",
        "_id": "3",
        "_score": 0.6931472,
        "_source": {
          "id": 3,  "name": "Tiramisu - Comunidad Celíaca del Uruguay",  "ingredients"<receta 3>",
          "url": "http://comunidadceliaca.org/recetas/tiramisu-216",  "image": null,  "indCeliac": true
        }
      }
    ]
  }
}
```

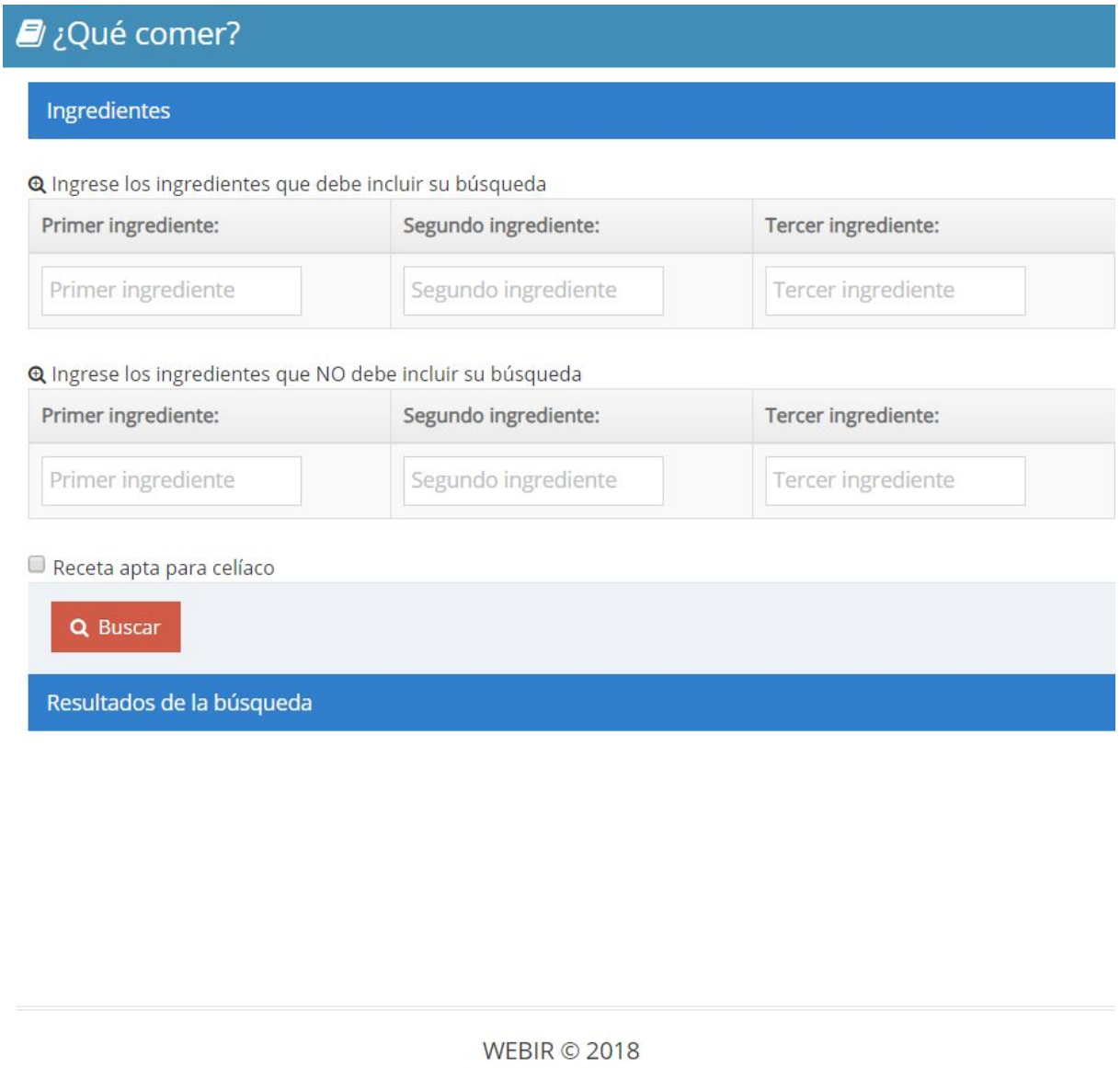
4.4. Aplicación Usuario

Esta aplicación se conecta a través del web service REST enviando el JSON [5] de la consulta deseada, mencionado en el punto anterior, con el motor de búsqueda de ElasticSearch para obtener la información de todas las publicaciones que cumplen con los filtros seleccionados. La llamada al web service REST desde la página JSP se realizará utilizando JQuery AJAX [9].

Se le permitirá al usuario de la aplicación ingresar manualmente la siguiente información:

- Listado de nombres de ingredientes incluidos en las recetas
- Listado de nombres de ingredientes no incluidos en las recetas
- Indicativo de Celíaco

A continuación se presenta un prototipo de la interfaz de usuario:



El botón de buscar disparará la consulta deseada según los filtros seleccionados. Los listados de ingredientes (incluidos y no incluidos), tendrán cada uno su correspondientes botones de agregar (+) y eliminar (-) registro de la grilla. En la sección listado de recetas se mostrará el resultado de la búsqueda.

5. Evaluación y resultados

El procesar información de una página web, dependiendo de cómo esté diseñada la misma, determina la complejidad del procesamiento que deba llevarse adelante. Es por ello que si la página no tiene un estándar en su información, puede llevar a que el procesamiento de la misma no se realice en su totalidad.
Con el objetivo de medir la efectividad, es que se analizaron los dos procesos de scraping, obteniendo para el primero de ellos (Comida Casera) un 98% de eficacia en la obtención de la información, y para el segundo (Comunidad Celíacos) un 80% de eficacia, dado que este último no tiene un diseño unificado para las distintas páginas de recetas.
De todas formas, dado el elevado número de recetas que existen en cada página, el número de procesadas es significativamente bueno.

Por otro lado, se evaluó la efectividad de las consultas realizadas utilizando elasticSearch para lo cual se implementaron distintas búsquedas, obteniendo buenos indicadores de eficacia en las mismas.

6. Conclusiones

Centralizar la información de diferentes fuentes en una única aplicación, facilita la tarea de los usuarios al momento de buscar recetas con ciertas características.
La utilización de jsoup facilita la obtención de la información y la herramienta elasticSearch provee un potente motor de búsquedas que optimiza las consultas, dotando al aplicativo de una buena performance y capacidad de adaptar las consultas a las distintas necesidades.

7. Trabajo Futuro

Sería interesante que el proyecto se extienda, añadiendo y diversificando las fuentes orígenes que alimentan la base de datos de recetas.
Por ejemplo, recuperar recetas de distintas organizaciones las cuales se especializan en alimentación orientada a personas con distintas enfermedades. Con esto, el aplicativo contará con mayor público objetivo y será de mayor utilidad, evitando que las personas deban recorrer varias páginas en busca de la información.

Con el mismo objetivo que el punto anterior, podría alimentarse de información obtenida de distintos grupos abiertos de Facebook, utilizando la Api de Facebook para java; para lo cual se debe obtener las autorizaciones correspondientes de Facebook y los propietarios de los grupos.

Otros dos puntos interesantes a tener en cuenta, sería que el proceso principal encargado de recuperar las recetas de las distintas fuentes, tenga la capacidad de procesar nuevos post que aparezcan, de modo de ir retroalimentando la información de forma constante. Junto

con esto, agregar inteligencia al proceso, para que distinga recetas duplicadas unificando las mismas y pueda categorizarlas para darle más valor agregado al usuario. Un ejemplo de categorización sería: Entradas, Platos Principales, Postres, etc.

Por último, otro punto que queda como trabajo a futuro que el usuario pueda ingresar una cantidad ilimitada de ingredientes para la búsqueda de recetas.

8. Referencias

[1] Jsoup - https://jsoup.org/
[2] ElasticSearch - https://www.elastic.co/products/elasticsearch
[3] Lucene - https://es.wikipedia.org/wiki/Lucene
[4] RESTful - https://en.wikipedia.org/wiki/Representational_state_transfer
[5] JSON - https://en.wikipedia.org/wiki/JSON
[6] Apache License - https://en.wikipedia.org/wiki/Apache_License
[7] Stemming - https://es.wikipedia.org/wiki/Stemming
[8] Fuzzing - https://en.wikipedia.org/wiki/Fuzzing
[9] JQuery AJAX - http://api.jquery.com/jQuery.ajax/
[10] ElasticSearch Java api - https://www.elastic.co/guide/en/elasticsearch/client/java-api/6.4/java-docs.html
[11] ElasticSearch Java api search - https://www.elastic.co/guide/en/elasticsearch/client/java-api/6.4/java-search.html
[12] Usar ElasticSearch : Ventajas, casos prácticos y bibliografía - https://apiumhub.com/es/tech-blog-barcelona/usar-elasticsearch-ventajas-libros/
[13] Guía ElasticSearch en Java (Last modified October 20, 2018): https://www.baeldung.com/elasticsearch-java