**ÍNDICE GENERAL**

[1. Introducción](#_Toc454867310) 2

[1.1. Antecedentes](#_Toc454867311) 2

[1.2. Definición del problema](#_Toc454867312) 2

[1.3. Objetivos](#_Toc454867313) 3

[1.3.1. Objetivo general](#_Toc454867314) 3

[1.3.2. Objetivos específicos](#_Toc454867315) 3

[1.4. Innovación tecnológica](#_Toc454867316) 3

[1.5. Justificación 4](#_Toc454867317)

[1.6. Metodología de desarrollo](#_Toc454867318) 4

[1.6.1. Matriz de objetivos](#_Toc454867314) 5

[1.6.2. Diagrama de Gantt](#_Toc454867315) 7

[1.7. Alcance …8](#_Toc454867319)

1.8. Bibliografía……………………………………………………………………………………………………………………………….8

Introducción

La integración de las tecnologías de información y las comunicaciones (TICs), ha contribuido a un mayor acceso a la información.

Hoy en día el acceso a la información se realiza por medio de la web. La naturaleza de la web es distribuida en relación a la información, la cantidad de la información va creciendo de manera progresiva día tras día sobre todo en la gestión de los documentos más específicamente en la publicación de edictos digitalizados. Esta situación ha generado la necesidad de crear un motor de búsqueda enfocado a la búsqueda de edictos digitalizados.

Los edictos son documentos que se publican en el periódico de manera cotidiana, ellos mismos representan una fuente de información importante que es manejada de una manera muy ineficiente dado que contienen datos importantes que no pueden ser encontrados directamente por el interesado dado su cantidad de contexto y el número de periódicos que hay en el país.

Dada esta problemática, este trabajo propone el desarrollo de un motor de búsqueda basada en bases de datos no relacionales aplicando Elasticsearch para la búsqueda de los edictos digitalizados proveyendo la información de manera más rápida y eficiente.

* 1. Antecedentes

La gestión de los documentos es un área muy importante dada la extensa cantidad de información que se maneja a través de ellos, información se va acumulando documentos desde que el hombre comenzó a escribirlos y a la necesidad de almacenarlos en bases de datos se vuelve imperiosa, esto teniendo en cuenta que los tipos y áreas de los documentos son variados una de esas áreas se encuentra en el aspecto legal más específicamente con la publicación de los edictos de prensa.

Edicto en el derecho moderno, es el mandato o decreto publicado con autoridad magistrado, juez o autoridad administrativa que dispone la observancia de ciertas reglas en algún asunto. También se determina así a los anuncios fijados en lugares públicos de las ciudades, villas o edificios gubernamentales sobre algún asunto para que sea notorio y de conocimiento general. ​En la práctica forense del derecho es un tipo de comunicación procesal. En nuestro caso nosotros trabajaremos con edictos de prensa digitalizados.

El crear un motor de búsqueda para realizar búsquedas en esa gran cantidad de información que contienen los edictos y las cuales son almacenadas en bases de datos y encontrar cualquier información de una manera más directa aprovechando la eficiencia de Elasticsearch combinado con Spring data para la implementación del motor de búsqueda.

* 1. Definición del problema

Los edictos son documentos publicados por un medio impreso un periódico de maneras distintas y con gran cantidad de información haciendo difícil encontrar la información de interés como ser los nombres, carnet de identidad, direcciones y otros, el interesado debe realizar una búsqueda manual y exhaustiva leyendo todo el contexto del edicto.

Se debe de leer una gran cantidad de información de manera exhaustiva dado la cantidad de edictos en un periódico y sin mencionar la calidad pésima del texto impreso que se tiene que se tiene que tratar de encontrar que varía de empresa a empresa además de tener que conseguir los periódicos de otros departamentos diferentes al que uno se encuentra haciendo bastante extensa teniendo que perder mucho tiempo. En el peor de los casos no logrando encontrar la información y teniendo que sufrir las consecuencias del mismo.

* 1. Objetivo
     1. Objetivo general

Implementar un motor de búsqueda de edictos digitalizados basado Elasticsearch para facilitar la búsqueda de la información.

* + 1. Objetivos específicos
* Implementar la arquitectura del motor de búsqueda en el lado del servidor y en el lado del cliente.
* Implementar un algoritmo de indexación de tablas relacionadas de base de datos y pasar los datos a colección de documentos en Elasticsearch.
* Implementar el algoritmo de búsqueda para la búsqueda de texto completo y búsqueda avanzada.
* Implementar un módulo de reportes para la presentación de los resultados y el monitoreo de los edictos.
* Desarrollar un módulo de registro y carga de nuevos edictos para su posterior búsqueda.
  1. Innovación tecnológica

Se desarrolló un motor de búsqueda de edictos almacenados en una base de datos no relacional en Elasticsearch el cual estará implementado enteramente en la parte del back end en lenguaje orientado a objetos Java haciendo uso de framework Spring Boot añadiendo la dependencia Spring Data para poder manejar los Documentos persistidos en un motor de datos Elasticsearch y generar un algoritmo de búsqueda tanto por texto como para búsquedas avanzadas, generando un api Restful, para que sea consumido por una aplicación cliente basado en un framework JavaScript llamado Angular.

Elasticsearch es un motor de búsqueda orientado a documentos que nos permite indexar una gran cantidad de datos para poder realizar consultas sobre ellos posteriormente.

Entre otras cosas nos permite hacer búsquedas de texto completo, búsquedas aproximadas, facetas y de resultado.

Las ventajas tecnológicas que nos provee Elasticsearch son:

* **Acceso en tiempo real**: Elasticsearch nos permite acceder sobre los datos que se están modificando en tiempo real.
* **Escalabilidad**: Gracias a su diseño nos permite escalar de forma horizontal e ir escalando nuestros servidores según nuestras necesidades.
* **Alta disponibilidad**: Los clúster de Elasticsearch son capaces de detectar qué nodos están fallando y reorganizarse para hacer que los datos sean siempre accesibles.
* **Multi-Tenant:** Nos permite operar sobre distintos índices al mismo tiempo y así potenciar nuestras búsquedas.
* **No utiliza esquemas:**Permite trabajar sin una estructura fija de base de datos.
* **Orientado a documentos:** Las entidades de Elasticsearch se almacenan como archivos JSON estructurados donde todos los campos son indexados y podemos incluir todos los índices en una misma consulta.
* **API:** Elasticsearch nos proporciona API Restful en JSON junto con API para diferentes lenguajes.
* **Búsquedas basadas en texto:**Elasticsearch está basado en Lucene, lo que incrementa las capacidades de búsqueda de texto, soportando geo localización, autocompletado.
* **Gestión de conflictos:** Previene la pérdida de datos al editar simultáneamente los registros.

Spring Data es un proyecto de SpringSource cuyo propósito es unificar y facilitar el acceso a distintos tipos de tecnologías de persistencia, tanto a bases de datos relacionales como a las del tipo NoSQL.

Spring ya proporcionaba soporte para JDBC, Hibernate, JPA, JDO o MyIbatis, simplificando la implementación de la capa de acceso a datos, unificando la configuración y creando una jerarquía de excepciones común para todas ellas.

Y ahora, Spring Data viene a cubrir el soporte necesario para distintas tecnologías de bases de datos NoSQL y, además, integra las tecnologías de acceso a datos tradicionales, simplificando el trabajo a la hora de crear las implementaciones concretas.

* 1. Justificación

Justificación técnica del por qué se está implementando un motor de búsqueda, es porque no hay un motor de búsqueda que este enfocados a la búsqueda de edictos que minimice el encontrar un determinada información por el interesado, además del aprovechamiento de Elasticsearch que es una de las mejores tecnologías para realizar búsquedas.

Justificación económica porque al usuario que haga uso de la aplicación tendrá beneficios en el proceso de búsqueda de edictos de su interés minimizando el tiempo y logrando responder de manera pertinente y ágil con los edictos que coinciden con la búsqueda de su interés.

Justificación social el proyecto puede tener una orientación de apoyo en la sociedad Boliviana, mejorando la calidad de las personas respondiendo de manera rápida a la necesidad que tenga y otorgándole más tiempo a las personas que tengan la necesidad de buscar alguna información de su interés en los edictos en algún periódico a nivel nacional.

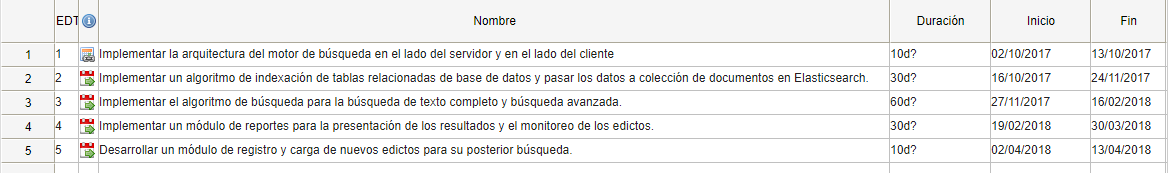
* 1. Metodología de desarrollo

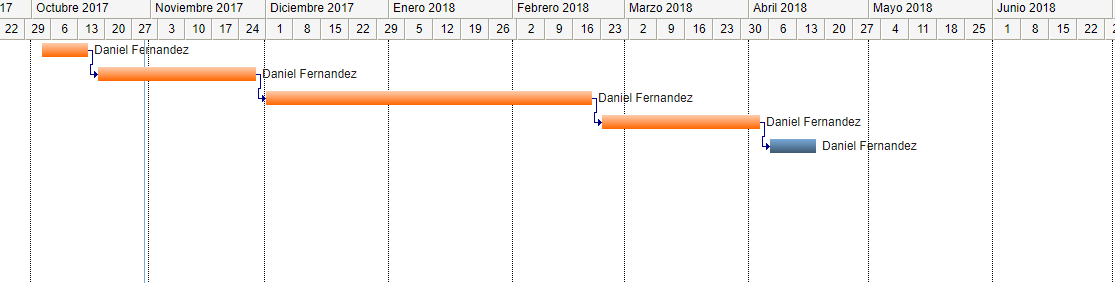
El proyecto se desarrollará de manera incremental, en la cual cada módulo o subsistema se tomará como un incremento de la aplicación. Cada incremento seguirá este proceso, la primera es realizar la implementación de un algoritmo de indexación para poder llevar la información de la base de datos relacional a la base de datos de Elasticsearch, la segunda etapa la implementación de un algoritmo de búsqueda por texto, y después pasaría a la etapa de implementación de un algoritmo de búsqueda avanzada para realizar búsquedas más específicas, por último en la quinta etapa se realizara la aplicación cliente para realizar las búsquedas por el usuario y mostrar la información de los resultados de las búsquedas.

* + 1. Matriz de objetivos

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Objetivos | Actividades a Realizar | Resultados | Nro. Días Provistos | Fecha de Inicio | Fecha de Conclusión |
| Implementar la arquitectura del motor de búsqueda en el lado del servidor y en el lado del cliente.  . | * Creación de nuevo proyecto en Spring Boot. * Instalación de dependencias (Spring Data, Elasticsearch). * Configuración de propiedades para conexión con Elasticsearch * Creación de modulo cliente en Angular | * Proyecto creado en Spring Boot, con dependencias instaladas y configuración para la conexión a Elasticsearch | 10 | 02/10/2017 | 13/10/2017 |
| Implementar un algoritmo de indexación de tablas relacionadas de base de datos y pasar los datos a colección de documentos en Elasticsearch. | * Creación de algoritmo de migración de datos desde una base de datos relacional * Creación de la base de datos a través de los objetos persistentes Spring Data y JPA | * Base de datos no relacional en Elasticsearch * Datos migrados a la base de datos | 30 | 16/10/2017 | 24/11/2017 |
| Implementar el algoritmo de búsqueda para la búsqueda de texto completo y búsqueda avanzada. | * Diseño de algoritmo de búsqueda. * Implementación del algoritmo en Spring Boot y Spring Data para búsquedas en Elasticsearch. * Diseñar el algoritmo de búsqueda con campos establecidos. * Implementación de algoritmo en Java Spring Boot tomando como criterios obligatorios los parámetros. | * Implementación de algoritmo para búsqueda avanzada con criterios | 60 | 25/11/2017 | 16/02/2018 |
| Implementar un módulo de reportes para la presentación de los resultados y el monitoreo de los edictos. | * Implementar la interfaz de para la presentación de los resultados de las búsquedas. * Implementar un visor de edictos digitalizados en pdf | * Interfaz para la presentación de los resultados y la visualización de los edictos digitalizados | 30 | 19/02/2018 | 30/02/2018 |
| Desarrollar un módulo de registro y carga de nuevos edictos para su posterior búsqueda. | * Implementar un módulos de registro de los edictos digitalizados. * Registrar los nuevos edictos en la base de datos de Elasticsearch. | * Módulo de registro de edictos digitalizados | 10 | 02/04/2018 | 13/04/2018 |

* + 1. Diagrama de Gantt





* 1. Alcance
* Se contempla únicamente el diseño la implementación del algoritmo de búsqueda.
* Para para la implementación se hará uso de Elasticsearch para realizar almacenar los datos.
* El sistema no contempla la gestión de los usuarios ni todo lo que respecta a los usuarios.
* El sistema no contempla la digitalización de los edictos.

1. MARCO TEORICO
   1. Área de Conocimiento del proyecto

Para un mejor entendimiento de los componentes que integran el sistema se realizara de una descripción detallada de cada uno de ellos.

* + 1. Arquitectura

La programación está basada en una arquitectura RESTFUL que separa el sistema en dos áreas importantes el lado del cliente que es la que interactúa con los usuarios del sistema la que genera todas las solicitudes, y el lado del servidor que es la que se conecta con la base de datos siendo este el más importante del sistema generando un API RESTFUL que  genera las respuestas para las solicitudes del Cliente.

2.1.1.1. Características del API RESTFUL

El sistema es un servicio desarrollado en distintos lenguajes de programación y también con una arquitectura variada pero como base principal para la comunicación de los servicios entre sistemas distribuidos, sistema tiene como arquitectura principal el modelo de servicios RESTFUL el cual será descrito a detalle.

2.1.1.2. Servicios RESTFUL

La Transferencia de Estado Representacional (REST - Representational State Transfer) fue ganando amplia adopción en toda la web como una alternativa más simple a SOAP y a los servicios web basados en el Lenguaje de Descripción de Servicios Web (Web Services Descripción Language - WSDL). Veamos los principios de REST para entender más esta tecnología.

2.1.1.3. REST

REST define un set de principios arquitectónicos por los cuales se diseñan servicios web haciendo foco en los recursos del sistema, incluyendo cómo se accede al estado de dichos recursos y cómo se transfieren por HTTP hacia clientes escritos en diversos lenguajes.

Una implementación concreta de un servicio web REST sigue cuatro principios de diseño fundamentales:

* Utiliza los métodos HTTP de manera explícita
* No mantiene estado
* Expone URIs con forma de directorios
* Transfiere XML, JavaScript Object Notation (JSON), o ambos

REST hace que los desarrolladores usen los métodos HTTP explícitamente de manera que resulte consistente con la definición del protocolo. Este principio de diseño básico establece una asociación uno-a-uno entre las operaciones de crear, leer, actualizar y borrar y los métodos HTTP. De acuerdo a esta asociación:

* Se usa POST para crear un recurso en el servidor
* Se usa GET para obtener un recurso
* Se usa PUT para cambiar el estado de un recurso o actualizarlo
* Se usa DELETE para eliminar un recurso

Este modelo sigue una comunicación entre un cliente y un servidor ambos comunicándose bajo responsabilidades por ambas partes.

2.1.1.3. Responsabilidad del servidor

Genera respuestas que incluyen enlaces a otros recursos para permitirle a la aplicación navegar entre los recursos relacionados. Este tipo de respuestas tiene enlaces embebidos. De la misma manera, si la petición es hacia un padre o un recurso contenedor, entonces una respuesta REST típica debería también incluir enlaces hacia los hijos del padre o los recursos subordinados, de manera que se mantengan conectados.

Genera respuestas que indican si son susceptibles de caché o no, para mejorar el rendimiento al reducir la cantidad de peticiones para recursos duplicados, y para lograr eliminar algunas peticiones completamente. El servidor utiliza los atributos Cache-Control y Last-Modified de la cabecera en la respuesta HTTP para indicarlo.

2.1.1.4. Responsabilidades del cliente de la aplicación

Utiliza el atributo Cache-Control del encabezado de la respuesta para determinar si debe cachear el recurso (es decir, hacer una copia local del mismo) o no. El cliente también lee el atributo Last-Modified y envía la fecha en el atributo If-Modified-Since del encabezado para preguntarle al servidor si el recurso cambió desde entonces. Esto se conoce como *GET Condicional*, y ambos encabezados van de la mano con la respuesta del servidor 304 (No Modificado) y se omite al recurso que se había solicitado si no hubo cambios desde esa fecha. Una respuesta HTTP 304 significa que el cliente puede seguir usando la copia local de manera segura, evitando así realizar las peticiones GET hasta tanto el recurso no cambie.

Envía peticiones completas que pueden ser servidas en forma independiente a otras peticiones. Esto implica que el cliente hace uso completo de los encabezados HTTP tal como está especificado por la interfaz del servicio web, y envía las representaciones del recurso en el cuerpo de la petición. El cliente envía peticiones que hacen muy pocas presunciones sobre las peticiones anteriores, la existencia de una sesión en el servidor, la capacidad del servidor para agregar contexto a una petición, o sobre el estado de la aplicación que se mantiene entre las peticiones.

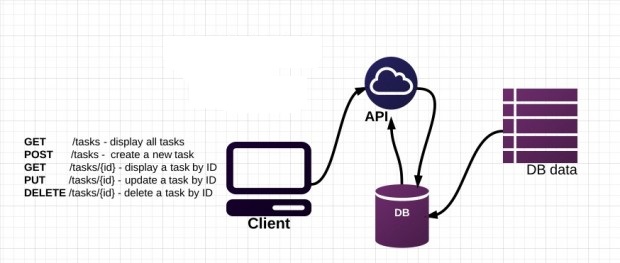
Esta colaboración entre el cliente y el servicio es esencial para crear un servicio web REST sin estado. Mejora el rendimiento, ya que ahorrar ancho de banda y minimiza el estado de la aplicación en el servidor.

2.1.1.5. REST transmite xml, json o ambos

La transmisión de los datos a través de JSON para la comunicación entre el cliente y el servidor es realizado por objetos json por el protocolo HTTP y con el uso de MIME para la negociación de los contenidos, el cual permite que los clientes puedan escoger el formato de los datos puedan leer, y minimiza el acoplamiento de datos entre el servicio y las aplicaciones que lo consumen. Los tipos MIME más usados son:

|  |  |
| --- | --- |
| **MIME-Type** | **Content-Type** |
| **JSON** | application/json |
| **XML** | application/xml |
| **XHTML** | application/xhtml+xml |

Tabla 1: Tipos de content type manejados en un request a una url



**Fig. 1:** Diagrama de modelo RestFul

* + 1. Elasticsearch

Elasticsearch es un servidor de búsqueda basado en Lucene. Provee un motor de búsqueda de texto completo, distribuido y con capacidad de multi-tenencia con una interfaz web RESTful y con documentos JSON.

Elasticsearch es una plataforma de búsqueda en tiempo casi real. Lo que esto significa es que hay una ligera latencia (normalmente un segundo) desde el momento en que indexa un documento hasta que se convierte en buscable.

2.1.2.1. Cluster

Un cluster es una colección de uno o más nodos (servidores) que, juntos contienen todos sus datos y proporcionan capacidades de indexación y búsqueda federadas a través de todos los nodos. Un cluster se identifica por un nombre único por defecto es Elasticsearch.

Este nombre es importante porque un modo solo puede formar parte de un cluster si el modo está configurado para unirse al cluster mediante su nombre.

2.1.2.2. Node

Un nodo es un único servidor que forma parte del clúster, almacena los datos y participa en las capacidades de indexación y búsqueda del clúster. Al igual que un clúster, un nodo se identifica con un nombre que, por defecto, es un Identificador Universalmente Único al azar (UUID) que se asigna al nodo en el inicio. Puede definir cualquier nombre de nodo que desee si no desea el valor predeterminado.

2.1.2.3. Index

Un índice es una colección de documentos que tienen características algo similares. Por ejemplo, puede tener un índice para datos de clientes, otro índice para un catálogo de productos y otro índice para datos de pedido. Un índice se identifica por un nombre (que debe estar en minúsculas) y este nombre se utiliza para referirse al índice al realizar operaciones de indización, búsqueda, actualización y eliminación con los documentos que contiene.

2.1.2.4. Type

Dentro de un índice, puede definir uno o más tipos. Un tipo es una categoría / partición lógica de su índice cuya semántica depende completamente de usted. En general, se define un tipo para documentos que tienen un conjunto de campos comunes. Por ejemplo, supongamos que ejecuta una plataforma de blogs y almacena todos sus datos en un único índice.

2.1.2.5. Document

Un documento es una unidad básica de información que se puede indexar. Por ejemplo, puede tener un documento para un solo cliente, otro documento para un solo producto y otro para un solo pedido. Este documento se expresa en JSON (JavaScript Object Notation), que es un omnipresente formato de intercambio de datos de Internet.

Dentro de un índice / tipo, puede almacenar tantos documentos como desee. Tenga en cuenta que, aunque un documento reside físicamente en un índice, en realidad un documento debe estar indexado / asignado a un tipo dentro de un índice.

2.1.2.6. Shards y Replicas

Un índice potencialmente puede almacenar una gran cantidad de datos que pueden exceder los límites de hardware de un único nodo. Por ejemplo, un único índice de mil millones de documentos que ocupan 1 TB de espacio en disco puede no caber en el disco de un solo nodo o puede ser demasiado lento para servir las solicitudes de búsqueda desde un solo nodo.

Para resolver este problema, Elasticsearch proporciona la capacidad de subdividir su índice en múltiples piezas denominadas fragmentos. Cuando crea un índice, simplemente puede definir el número de fragmentos que desee. Cada fragmento es en sí mismo un "índice" completamente funcional e independiente que puede alojarse en cualquier nodo del clúster.

* + 1. Edicto

Edicto es el mandato o decreto publicado con autoridad de un magistrado. El concepto procede del latín *edictum*, un vocablo que se utilizaba para nombrar al pronunciamiento de los magistrados romanos sobre cuestiones relativas a su competencia.

También se puede definir como un documento mediante el cual se da a conocer un mandato expedido por una autoridad jurídica competente. Se puede definir mejor por medio de la siguiente imagen:

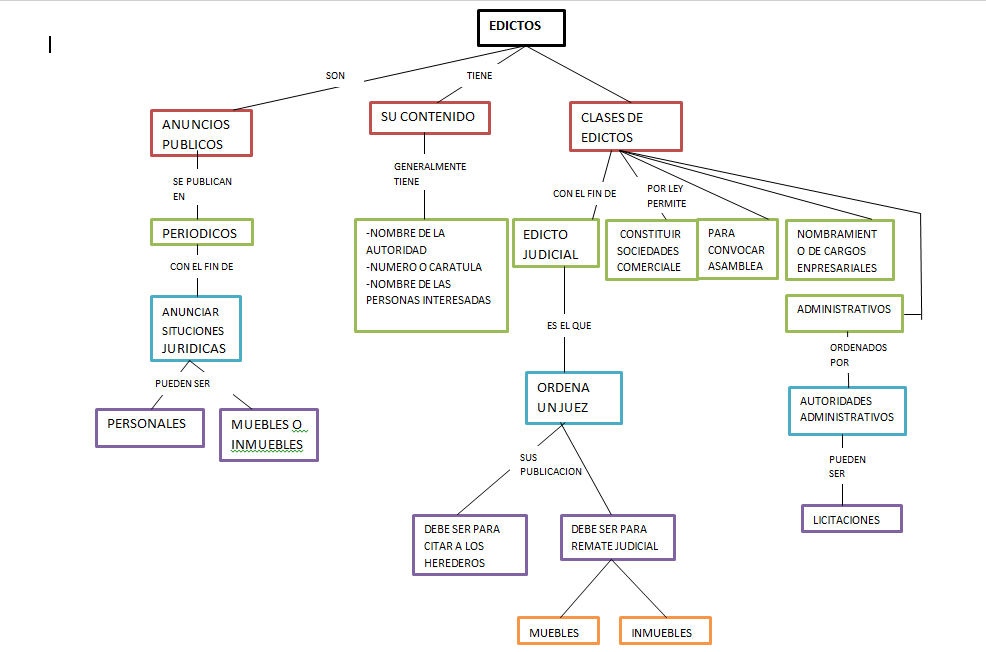


Fig. 1.1:Mapa conceptual de los edictos y su clasificación

Los edictos no son numerados de 1 a 3 en su publicación por lo cual el demandado no sabe si es el primero o el último porque solo tiene 30 días desde la primera publicación para contestar

* 1. Plataforma de Desarrollo

Para poder entender mejor la plataforma en la que se va desarrollar el sistema, vamos hacer un desglose de los componentes que se han de usar para la plataforma de desarrollo.

* + 1. Integración Spring Boot y Elasticsearch

Uno de los framework más usados del mercado es Spring MVC sobre todo para desarrollo de sistemas web basados en el lenguaje java, tiene una gran cantidad de documentación para poder realizar cualquier sistema lo cual lo pone como anfitrión para el desarrollo de este sistema, se decidió usar Spring Boot, porque tiene una muy fácil configuración y puesta en marcha libre de configuraciones en XML eso hecho te permite que te enfoques más en el desarrollo de la aplicación. Esto permite realizar una API RestFul para poder realizar las búsquedas en el sistema.

Para poder realizar una conexión con los datos o la base de datos vamos usar una de las dependencias de Spring Boot que nos permite persistir la información, Spring Data es la dependencia que ayudara a nuestra aplicación poder obtener y persistir la información de nuestra base datos, en nuestro caso vamos hacer uso de Elasticsearch que es servidor de búsquedas anteriormente mencionada que también almacena la información en documentos.

Para poder realizar las búsquedas y poder interactuar con el servidor Elasticsearch instalamos sus dependencias para Spring y poder implementar el algoritmo.

* + 1. Esquema General de Motor

El esquema general del motor de búsqueda debe responder a una cliente que realiza las solicitudes al API de búsqueda tanto de búsqueda por texto completo y avanzada siguiente un flujo como se muestra en la siguiente figura:

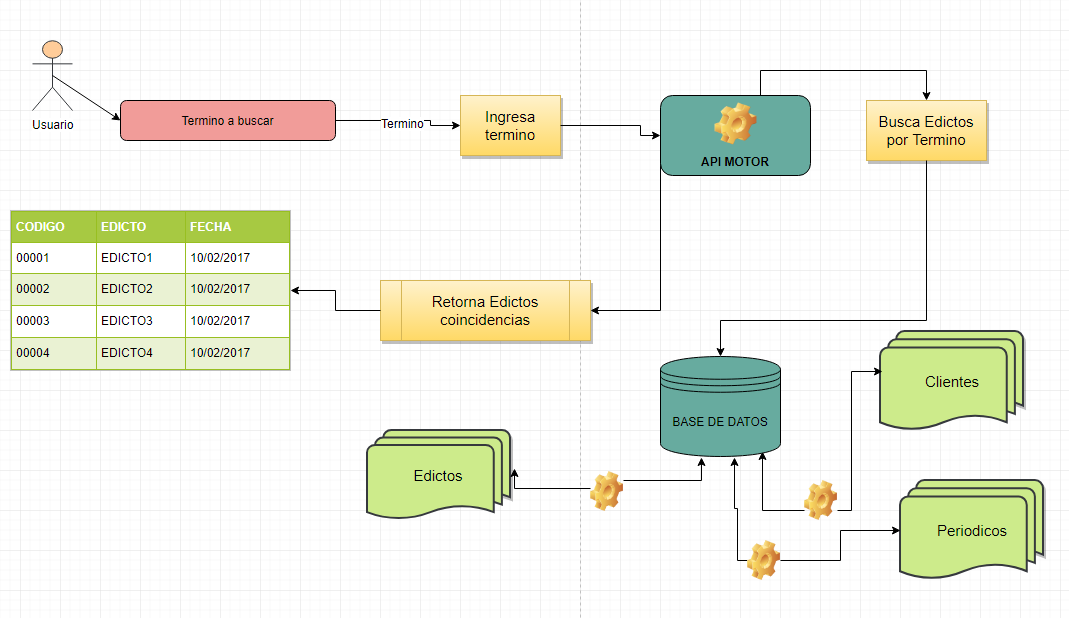
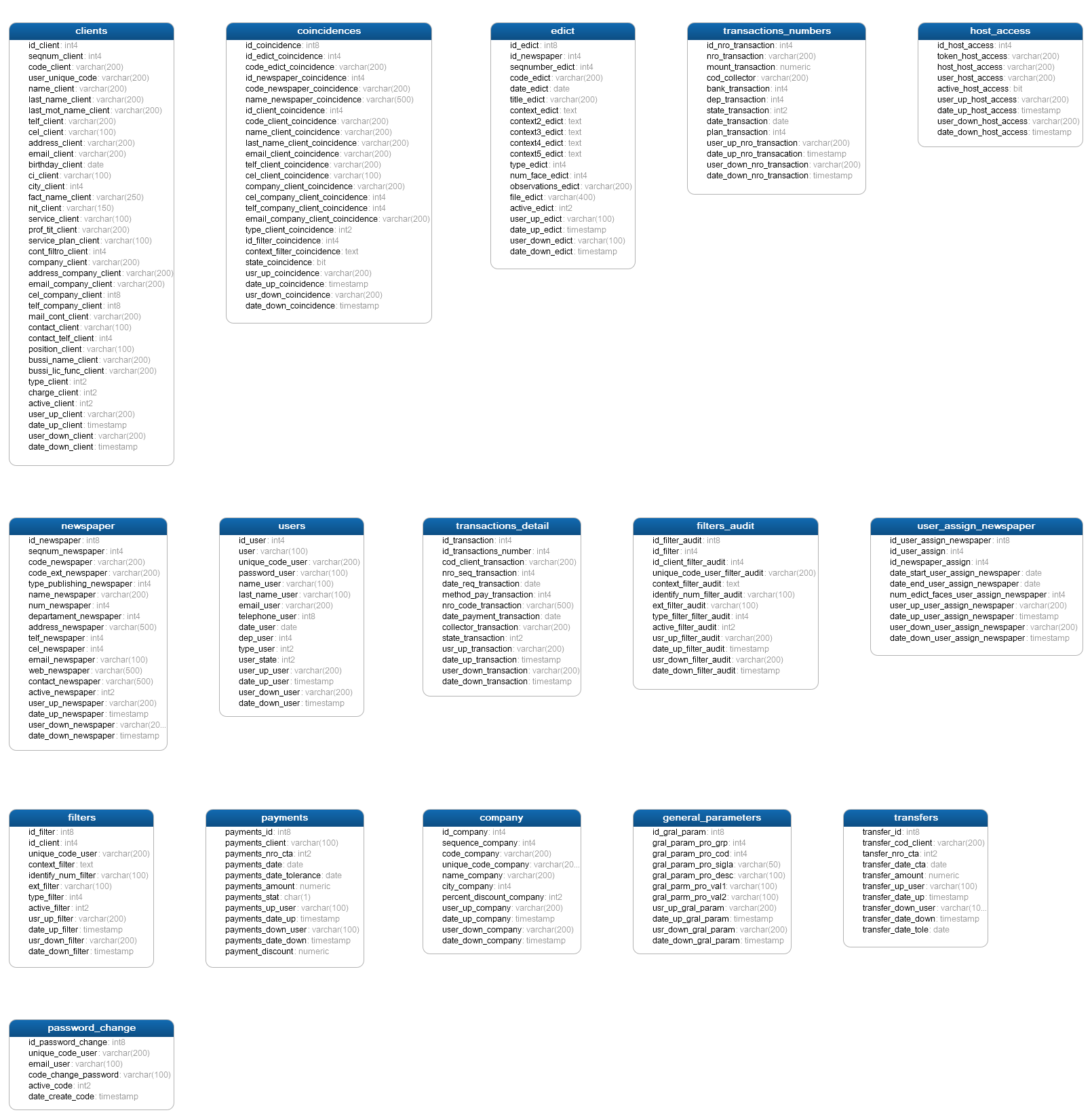


Fig.3: Esquema de algoritmo de búsqueda

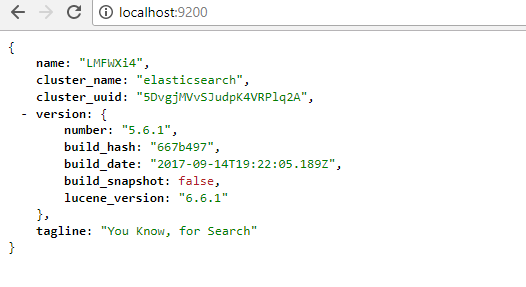
* 1. Administrador de las Base de Datos

El administrador de las bases de datos dado que es una base de datos no relacional no cuenta actualmente. Para poder obtener una base de datos lo primero que se hizo es el de migrar los datos desde una base de datos relacional y pasarlo por el algoritmo de indexación para llevarlo al servidor de Elasticsearch en la siguiente imagen se muestra el modelo actual de la base de datos:



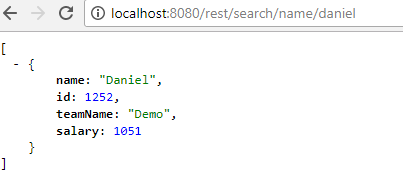
**Fig. 5:** Modelo de base de datos relacional

La manera de poder revisar los documentos indexados en el servidor Elasticsearch es a través de consultar al API que viene por defecto el cual nos retornara una documento con la información del como la entiende Elasticsearch la siguiente imagen muestra la primero ruta



**Fig. 6:** Consulta al API principal con las características de Elasticsearch

Para poder entender mejor el funcionamiento del sistema se preparó un ejemplo con información que básica almacenada en el servidor de Elasticsearc la siguiente figura nos muestra los datos que contiene el índice Persona que se creó desde la nueva API desarrollada con Spring Boot y Spring Data.



**Fig. 7:** API de ejemplo de un índice persona realizando una búsqueda por el nombre

* 1. Bibliografía
* Phillip Webb, Dave Syer, Josh Long, Stéphane Nicoll, Rob Winch, Andy Wilkinson, Marcel Overdijk, Christian Dupuis, Sébastien Deleuze, Michael Simons. (2012-2017). Documentacion spring boot. 25-09-2017, de Documentacion spring boot Sitio web: <https://docs.spring.io/spring-boot/docs/current/reference/html/>
* Pivotal Software. (15-10-2015). spring boot. 25-09-2017, de Spring boot Sitio web: <https://projects.spring.io/spring-boot/>
* Gerard Auladell. (28-11-2016). ¿Qué es Elasticsearch?. 25-09-2017, de Elasticsearch Sitio web: <https://www.drauta.com/que-es-elasticsearch>
* Elasticsearch. (11-09-2017). Elasticsearch documentacion. 25-09-2017, de Elasticsearch Documentacion Sitio web: <https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html>
* Elasticsearch. (2011). ElasticSearch documentacion. Julio 2011, de ElasticSearch Sitio web: [https://www.elastic.co](https://www.elastic.co/)
* Elasticsearch. (2016). A Distributed RESTful Search Engine. Julio 2016, de ElasticSearch Sitio web: <https://github.com/elastic/elasticsearch>
* ElasticSearch. (2015). Spring Data Elasticsearch. Junio 2015, de Elasticsearch Sitio web: <https://docs.spring.io/spring-data/elasticsearch/docs/current/reference/html/>
* ElasticSearch Spring Data. (2015). Spring Data Elasticsearch. Junio 2015, de ElasticSearch Sitio web: <https://github.com/spring-projects/spring-data-elasticsearch>
* Hector Estrada. (2015). Clases de Notificacion. Octubre, 2015, de Clases de Notificacion Sitio web: <http://tareasjuridicas.com/2015/10/05/clases-notificacion/>
* Vinneth Mohan. (2015). Building Your Own E-Commerce Solution. En Elasticsearch Blueprints(40). Livery Place: Pack Publishing.