

---

# "Chasqui-SERVER API"

**Versión 1**

**Robert Cabrera Lara**

**04 agosto 2022**

**Santa Cruz - Bolivia**

# Tabla de Contenido

|  |           |
|--|-----------|
| <b>1. Introducción</b>   | <b>1</b>  |
| <b>1.1 Propósito</b>   | <b>1</b>  |
| <b>2. Código and Datos</b>   | <b>1</b>  |
| <b>2.1 Repositorio GitHub</b>                                      | <b>1</b>  |
| <b>2.2 Datos</b>   | <b>1</b>  |
| <b>3. Detalle de Implementación</b>                                | <b>2</b>  |
| <b>3.1 Detalle Herramientas utilizadas</b>                         | <b>2</b>  |
| <b>3.1.1 Docker</b>  | <b>2</b>  |
| <b>3.1.2 ElasticSearch</b>   | <b>3</b>  |
| <b>3.1.3 Kibana</b>  | <b>4</b>  |
| <b>3.1.4 SprintBoot</b>  | <b>4</b>  |
| <b>3.2 Detalle Arquitectura y Diagramas</b>                        | <b>6</b>  |
| <b>3.2.1 Arquitectura</b>  | <b>6</b>  |
| <b>3.2.2 Diagrama de Paquetes de ChasquiServer API</b>             | <b>7</b>  |
| <b>4. Demostración end-points Chasqui-Server API</b>               | <b>7</b>  |
| <b>5. Detalle flujo de cada end-point</b>                          | <b>8</b>  |
| <b>5.1 Guardar Producto</b>  | <b>8</b>  |
| <b>5.2 Buscar Producto - Sugerencias</b>                           | <b>9</b>  |
| <b>6. Evidencia de consistencia Eventual</b>                       | <b>10</b> |
| <b>6.1 Búsqueda del producto “papaliza” en Elastic</b>             | <b>10</b> |
| <b>6.2 Búsqueda del producto “papaliza” en ChasquiServer API</b>   | <b>11</b> |
| <b>6.3 Inserción del producto “papaliza” via ChasquiServer API</b> | <b>12</b> |
| <b>6.4 Búsqueda del producto “papaliza” en Elastic</b>             | <b>13</b> |
| <b>6.5 Búsqueda del producto “papaliza” en ChasquiServer API</b>   | <b>14</b> |
| <b>6.6 Búsqueda de Producto “pap4lza” via ChasquiServer API</b>    | <b>15</b> |
| <b>7. Conclusiones</b>   | <b>16</b> |
| <b>8. Referencias</b>  | <b>16</b> |

# 1. Introducción

## 1.1 Propósito

El presente documento tiene como propósito detallar la arquitectura y funcionalidad provista en chasqui-server API.

# 2. Código and Datos

## 2.1 Repositorio GitHub

El código está disponible en: <https://github.com/Trebor006/chasqui-server>

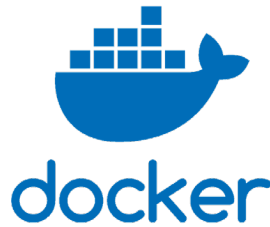
## 2.2 Datos

- Los datos importados a Elastic Search se encuentran en la carpeta
  - “**src/main/resources/static/**”
- En este folder se puede encontrar dos archivos un csv y un txt, el archivo utilizado para el bulk insert es el "products\_bulk\_data.txt"
- Para información adicional puede consultar el archivo **redme.md**, en él se detalla todo el proceso realizado para funcionar con ElasticSearch.

## **3. Detalle de Implementación**

### **3.1 Detalle Herramientas utilizadas**

#### **3.1.1 Docker**



#### **Cómo funciona Docker**

Docker le proporciona una manera estándar de ejecutar su código. Docker es un sistema operativo para contenedores. De manera similar a cómo una máquina virtual virtualiza (elimina la necesidad de administrar directamente) el hardware del servidor, los contenedores virtualizan el sistema operativo de un servidor. Docker se instala en cada servidor y proporciona comandos sencillos que puede utilizar para crear, iniciar o detener contenedores.

En este proyecto Docker se lo utilizara para levantar contenedores para Elasticsearch y Kibana

- En el proyecto se encuentra el archivo `docker_compose.yml` que contiene los comandos necesarios para la descarga de las imágenes necesarias.
- Para levantar las imágenes solo es suficiente con ejecutar el comando:
  - **“docker compose up”**

### 3.1.2 ElasticSearch



Elastic permite integrarse con una variedad de aplicaciones además de que se pueden instalar plugins.

- Para el desarrollo de ChasquiServer API se hizo utilización de un plugin para el **Phonetic Analysis**.

#### **¿Para qué se usa Elasticsearch?**

La velocidad y escalabilidad de Elasticsearch y su capacidad de indexar muchos tipos de contenido significan que puede usarse para una variedad de casos de uso:

- Búsqueda de aplicaciones
- Búsqueda de sitio web
- Enterprise Search
- Logging y analíticas de log
- Métricas de infraestructura y monitoreo de contenedores
- Monitoreo de rendimiento de aplicaciones
- Análisis y visualización de datos geoespaciales
- Analítica de Seguridad
- Analítica de Negocios

#### **¿Cómo funciona Elasticsearch?**

Los datos sin procesar fluyen hacia Elasticsearch desde una variedad de fuentes, incluidos logs, métricas de sistema y aplicaciones web. La ingesta de datos es el proceso mediante el cual estos datos son parseados, normalizados y enriquecidos antes de su indexación en Elasticsearch. Una vez indexados en Elasticsearch, los usuarios pueden ejecutar consultas complejas sobre sus datos y usar agregaciones para recuperar resúmenes complejos de sus datos. Desde Kibana, los usuarios crean visualizaciones poderosas de sus datos, comparten dashboards y gestionan el Elastic Stack.

### **¿Qué es un índice de Elasticsearch?**

Un índice de Elasticsearch es una colección de documentos relacionados entre sí. Elasticsearch almacena datos como documentos JSON. Cada documento correlaciona un conjunto de claves (nombres de campos o propiedades) con sus valores correspondientes (textos, números, Booleanos, fechas, variedades de valores, geolocalizaciones u otros tipos de datos).

#### **3.1.3 Kibana**



### **¿Para qué se usa Kibana?**

Kibana es una herramienta de visualización y gestión de datos para Elasticsearch que brinda histogramas en tiempo real, gráficos circulares y mapas. Kibana también incluye aplicaciones avanzadas, como Canvas, que permite a los usuarios crear infografías dinámicas personalizadas con base en sus datos, y Elastic Maps para visualizar los datos geoespaciales

#### **3.1.4 SprintBoot**



Spring Boot es una tecnología que nos permite crear aplicaciones autocontenidas, con esto nos podemos olvidar de la arquitectura y enfocarnos únicamente en desarrollo, delegando a Spring

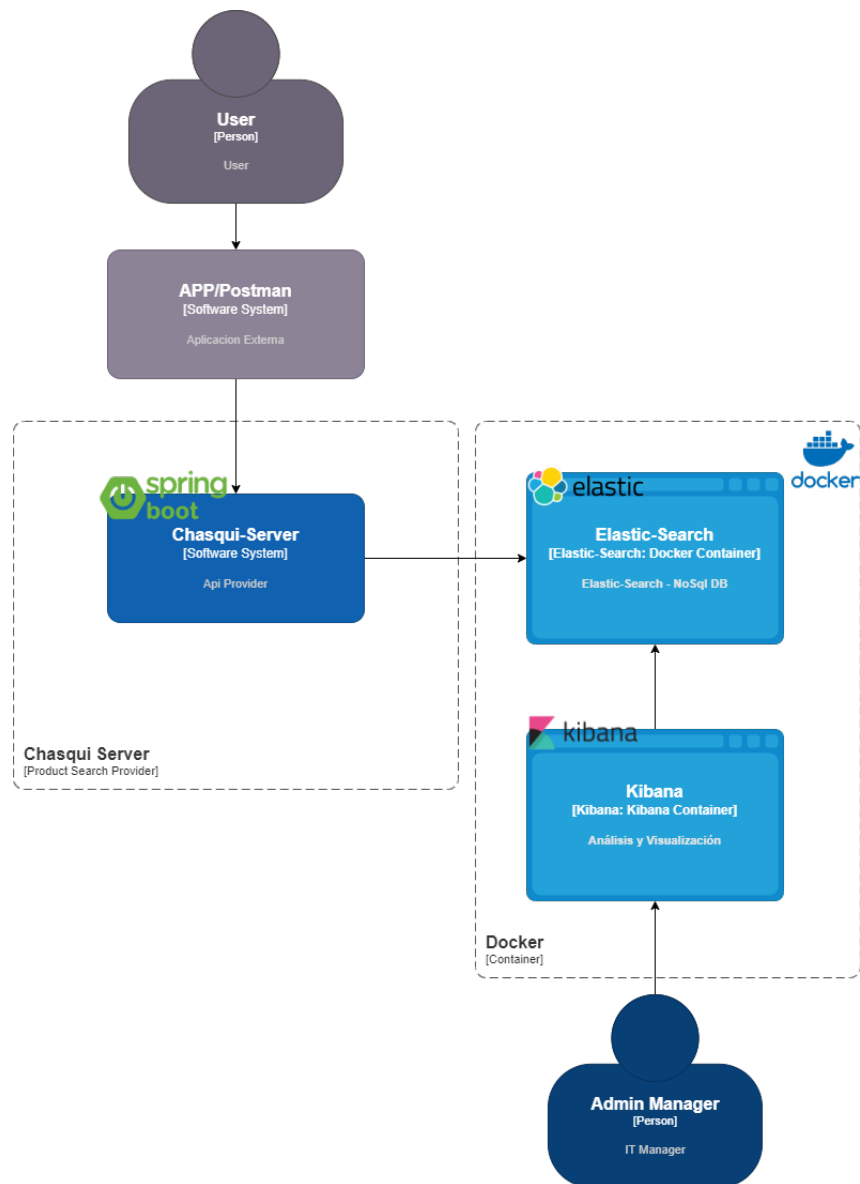
Boot labores como configuración de dependencias, desplegar nuestro servicio o aplicación a un servidor de aplicaciones y enfocarnos únicamente en crear nuestro código.

Para esto Spring Boot utiliza internamente un servidor de aplicaciones embebido, por defecto utiliza Tomcat, y no solo esto, Spring Boot también nos provee un completo gestor de dependencias como maven o gradle, configuraciones automáticas y mucho más para que nuestra aplicación quede a la medida.

Spring Boot facilita la creación de aplicaciones independientes basadas en Spring de grado de producción que puede "simplemente ejecutar".

## 3.2 Detalle Arquitectura y Diagramas

### 3.2.1 Arquitectura

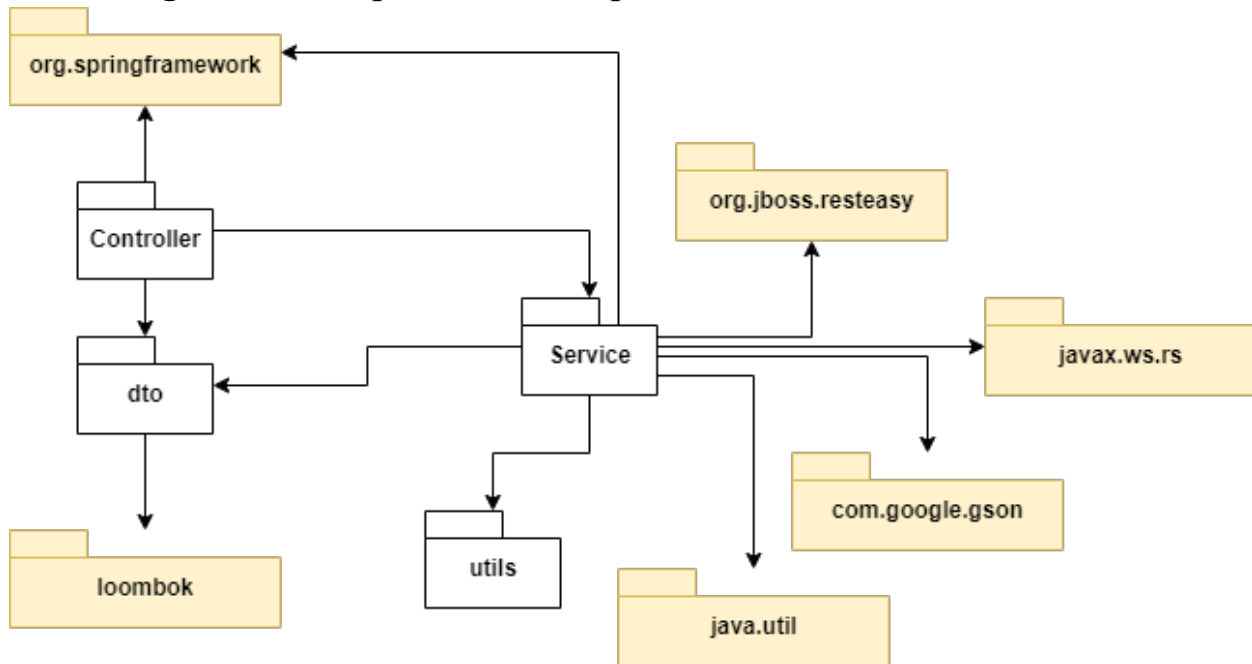


En este diagrama se puede notar que se utiliza:

- SprintBoot: para desplegar la aplicación Chasqui-Server API, que se conecta a los servicios expuestos por Elastic
- Docker: para levantar los contenedores de Elastic y Kibana
- Elastic: Contenedor de Elastic que publica el puerto 9200, para realizar las consultas. Y contiene la BD, además de que se le ha configurado el plugin.
- Kibana: para la administración y visualización de los datos de Elastic



### 3.2.2 Diagrama de Paquetes de ChasquiServer API



El proyecto ChasquiServer API, está compuesto por 4 paquetes, los cuales a su vez depende de librerías externas:

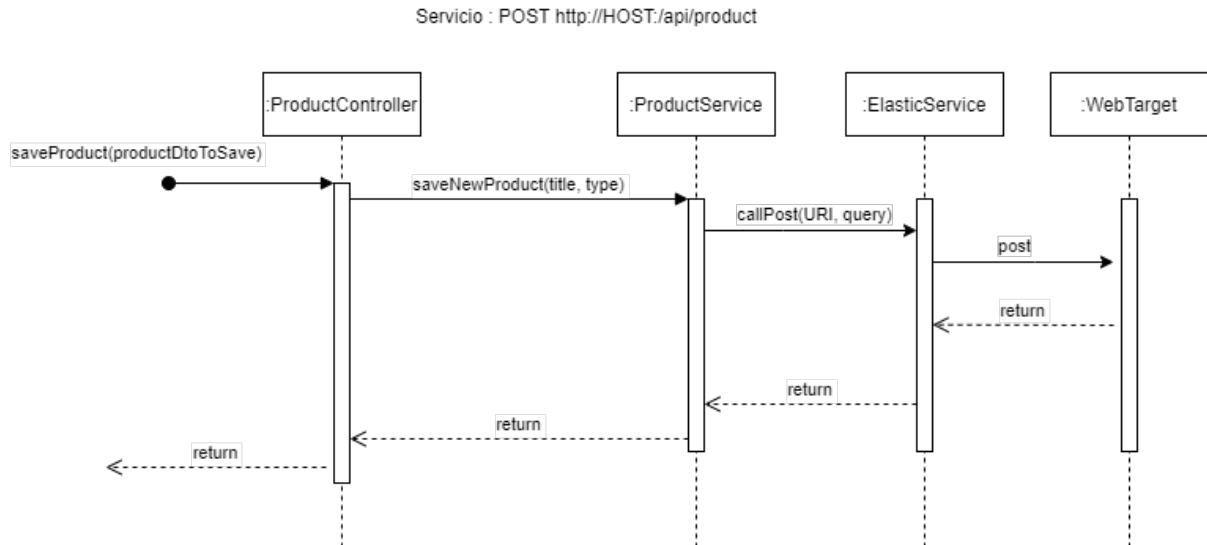
- controller: En este paquete se exponen los servicios para Products, el cual consume los paquetes Service y dto.
- service: Este paquete hace uso de librerías externas para poder consumir los servicios rest expuestos por ElasticSearch, esto lo hace mediante **resteasy**, **gson** y **javax.ws.rs**
- dto: contiene clases para obtener la información de ElasticSearch
- utils: Clases utilitarias.

## 4. Demostración end-points Chasqui-Server API

- Revisar el punto 6

## 5. Detalle flujo de cada end-point

### 5.1 Guardar Producto



El servidor ChasquiServer expone un servicio rest tipo POST “/api/product”

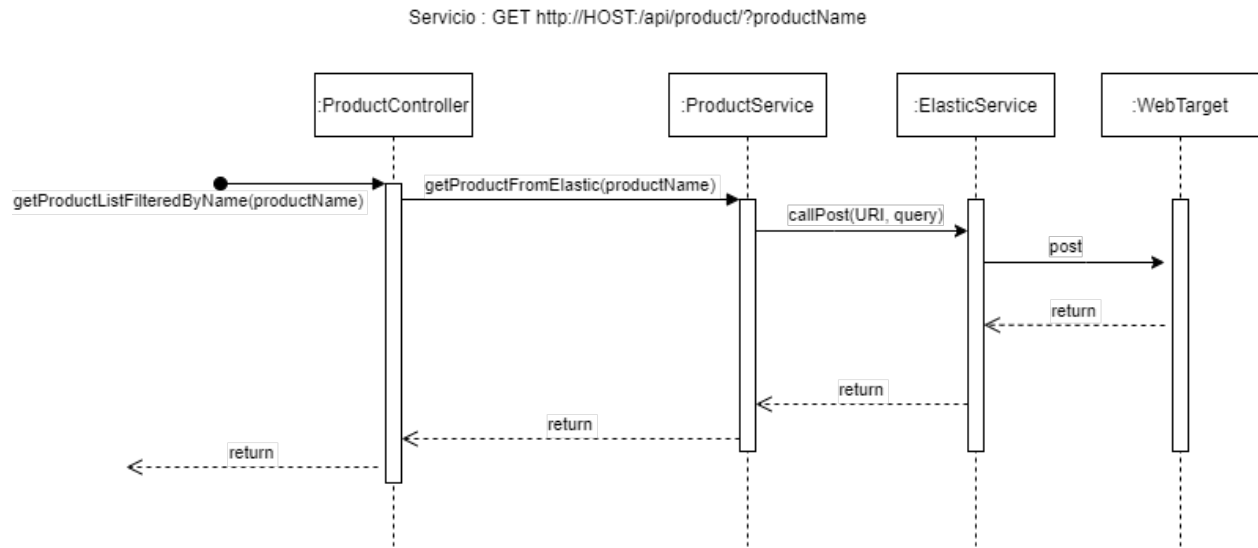
Este servicio espera un objeto:

```
{
  "title": "papaliza",
  "type": "hortaliza"
}
```

ProductController para la información a ProductService en donde se realizan ciertas operaciones para que se pueda hacer uso de la llamada callPost de ElasticService.

En ElasticService se realiza la llamada a los servicios rest de ElasticSearch, en este proceso se llama específicamente: POST http://ELASTIC\_HOST/products/\_doc, mediante las clases de RestEasy.

## 5.2 Buscar Producto - Sugerencias



El servidor ChasquiServer expone un servicio rest tipo GET “/api/producto/?productName=”  
Este servicio espera un queryParams: productName = “”.

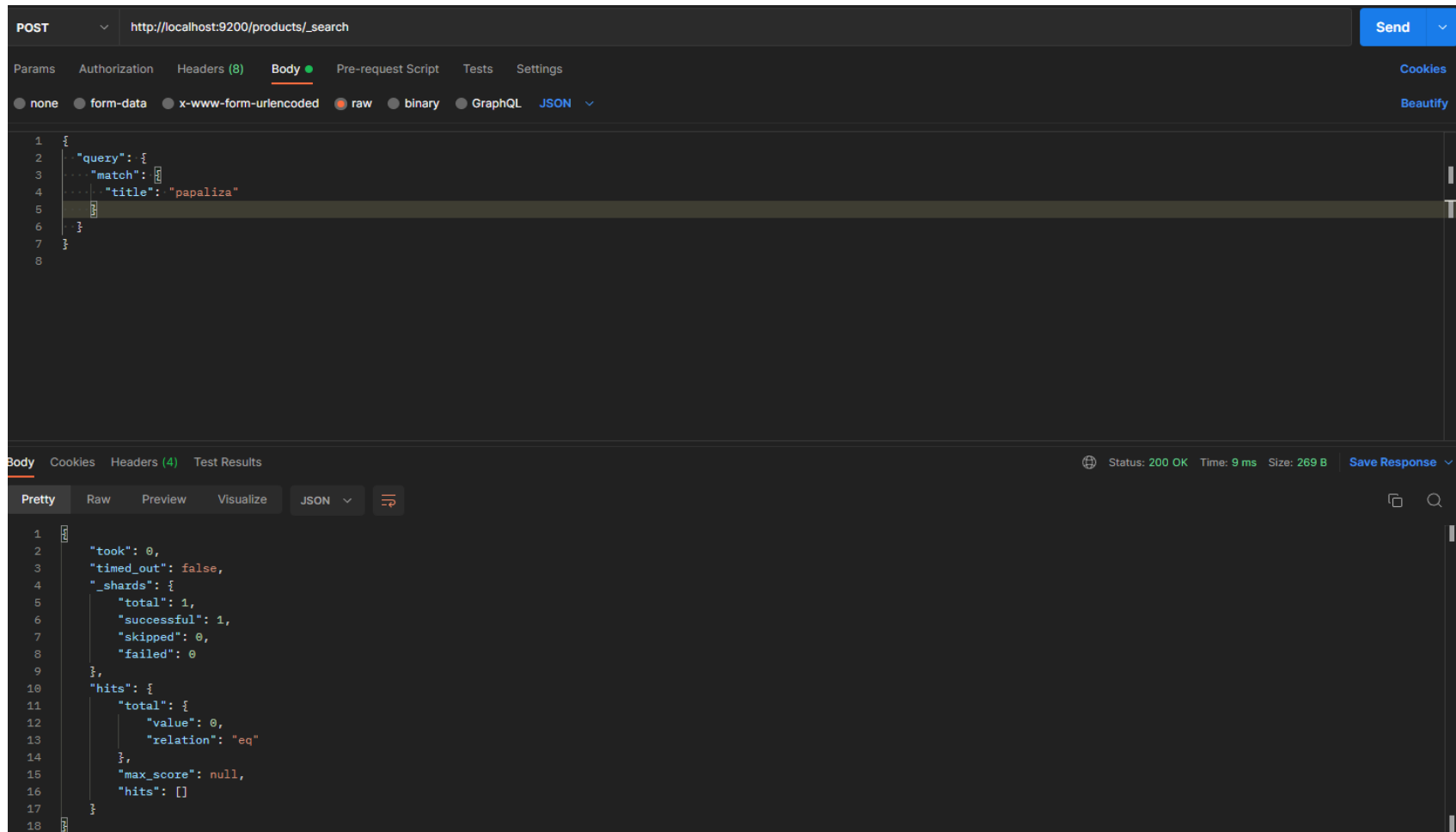
ProductController para la información a ProductService en donde se realizan ciertas operaciones para que se pueda hacer uso de la llamada callPost de ElasticService.

En ElasticService se realiza la llamada a los servicios rest de ElasticSearch, en este proceso se llama específicamente: POST `http://ELASTIC_HOST/products/_search`, mediante las clases de RestEasy.

Este end-point está configurado en ElasticSearch, con un índice llamado **products** que tiene las configuraciones necesarias para aplicar búsqueda vía **Phonetic Analysis**.

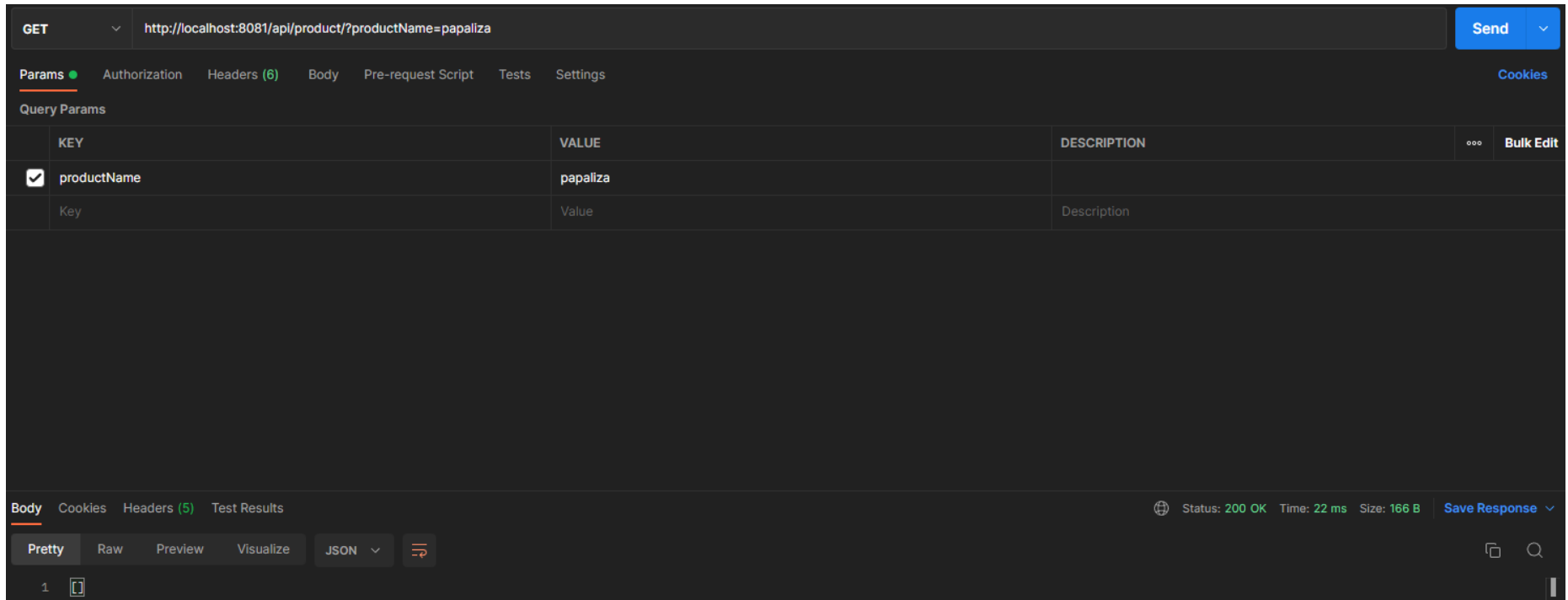
## 6. Evidencia de consistencia Eventual

### 6.1 Búsqueda del producto “papaliza” en Elastic



Como se puede notar el resultado de la consulta en **hits**: { **hits**: [] } es una lista vacía, ya que no se encontró el producto

## 6.2 Búsqueda del producto “papaliza” en ChasquiServer API



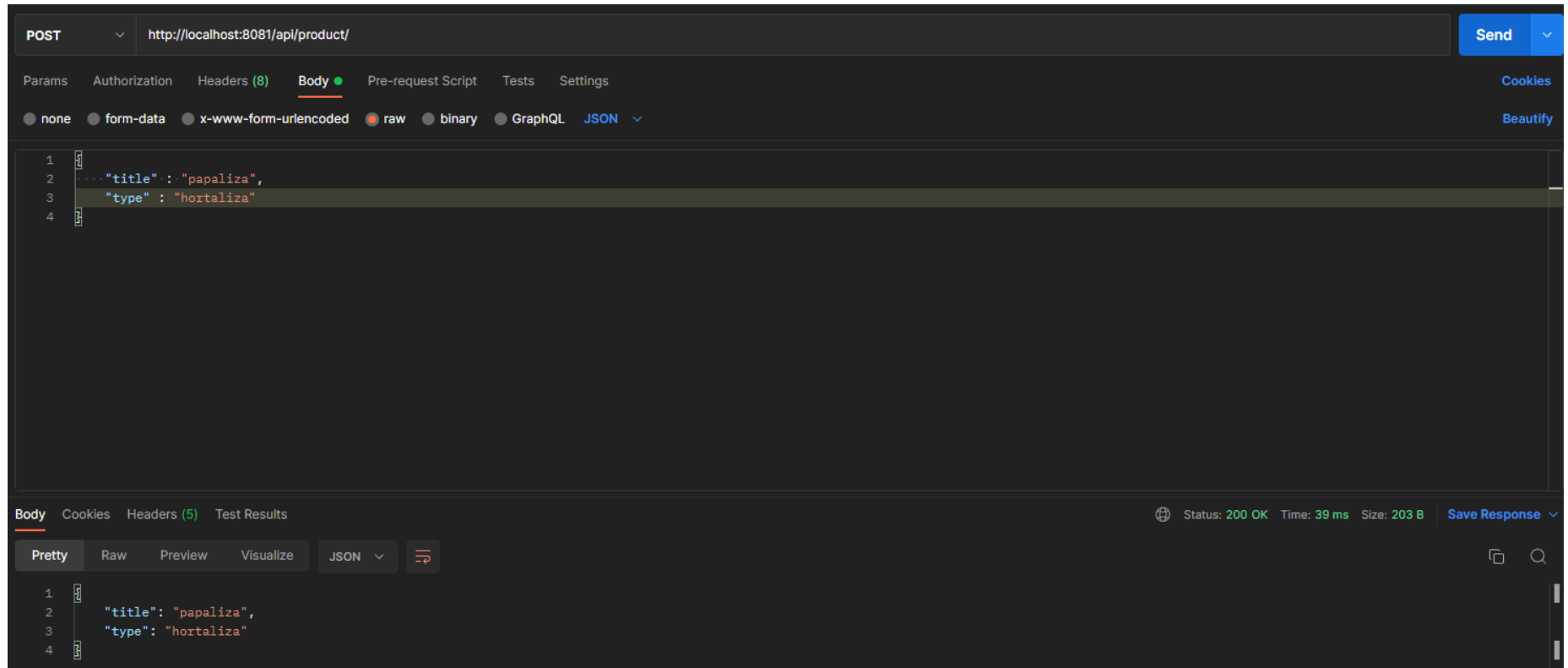
The screenshot displays a REST client interface with the following details:

- Method:** GET
- URL:** http://localhost:8081/apl/product/?productName=papaliza
- Params Tab:** Active, showing a table of query parameters.
- Query Params Table:**

|                                     | KEY         | VALUE    | DESCRIPTION |
|-------------------------------------|-------------|----------|-------------|
| <input checked="" type="checkbox"/> | productName | papaliza |             |
|                                     | Key         | Value    | Description |
- Body Tab:** Active, showing an empty response area.
- Status:** 200 OK, Time: 22 ms, Size: 166 B
- Response Format:** Pretty (JSON)

El resultado de la llamada es una lista vacía, lo que indica que no se encontró el producto en ElasticSearch.

### 6.3 Inserción del producto “papaliza” vía ChasquiServer API



Como se puede observar la respuesta es un 200, lo que quiere decir que se pudo realizar la inserción del nuevo producto en ElasticSearch.

## 6.4 Búsqueda del producto “papaliza” en Elastic

The screenshot displays a REST client interface with a POST request to `http://localhost:9200/products/_search`. The request body is a JSON object:

```
1 {
2   "query": {
3     "match": {
4       "title": "papaliza"
5     }
6   }
7 }
8
```

The response status is 200 OK, with a time of 10 ms and a size of 354 B. The response body is a JSON object:

```
11 {
12   "total": {
13     "value": 1,
14     "relation": "eq"
15   },
16   "max_score": 7.5190673,
17   "hits": [
18     {
19       "_index": "products",
20       "_id": "3xgrbIIBJW4iIr0J7U1k",
21       "_score": 7.5190673,
22       "_source": {
23         "title": "papaliza",
24         "type": "hortaliza"
25       }
26     }
27   ]
28 }
```

Como se puede notar el resultado de la consulta en **hits**: { **hits**: [ { **source**: { **title**, **type**}} ] }, lo que evidencia se encontró el producto

## 6.5 Búsqueda del producto “papaliza” en ChasquiServer API

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** http://localhost:8081/api/product/?productName=papaliza
- Params:** Query Params table with one entry: 

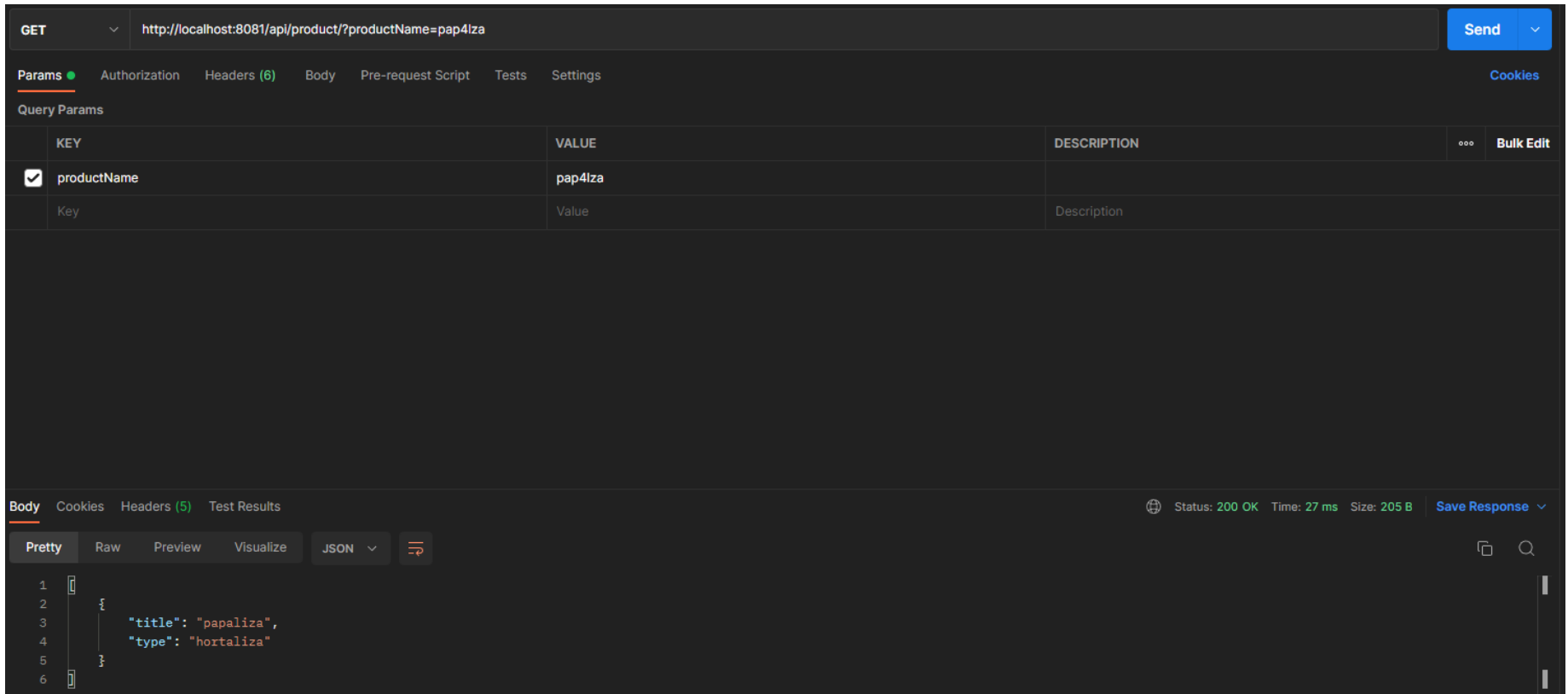
| KEY         | VALUE    | DESCRIPTION |
|-------------|----------|-------------|
| productName | papaliza |             |
- Body:** JSON response: 

```
{  "title": "papaliza",  "type": "hortaliza"}
```
- Status:** 200 OK
- Time:** 22 ms
- Size:** 205 B

Como se puede observar, ahora el API devuelve una lista con el producto encontrado en ElasticSearch.



## 6.6 Búsqueda de Producto “pap4lza” vía ChasquiServer API



The screenshot displays a REST client interface with the following details:

- Method:** GET
- URL:** http://localhost:8081/api/product/?productName=pap4lza
- Params:** A table showing query parameters.

| KEY   | VALUE   | DESCRIPTION |
|---|---------|-------------|
| <input checked="" type="checkbox"/> productName | pap4lza |             |
| Key   | Value   | Description |

The response body is shown in JSON format:

```
1 {  
2     
3   "title": "papaliza",  
4   "type": "hortaliza"  
5 }  
6
```

Response status: 200 OK, Time: 27 ms, Size: 205 B.

Como se puede observar, se ha realizado deliberadamente una búsqueda con el nombre del producto mal escrito, y se obtiene una sugerencia realizada desde Elasticsearch.

## 7. Conclusiones

- Elastic Search es una BD NoSQL potente, que realiza búsquedas complejas en millones de registros de forma eficiente.
- Elastic brinda muchas facilidades al permitir integrar plugin, como es el caso de **Phonetic Analysis**.
- Existen una variedad de algoritmos para realizar Phonetic Analysis, se debe realizar todas las configuraciones necesarias para poder obtener los resultados lo mas acertados posibles, ya que una mala configuración en el índice producirá datos no esperados.
- Docker facilita bastante el tener una BD desplegada, la mayoría de los gestores de BD proveen un mecanismo para su integración con Docker, tal como es el caso de ElasticSearch y Kibana.
- Elastic provee los mecanismos para conectarse vía Servicios Rest, por lo que uno puede conectarse desde cualquier cliente, en el transcurso del desarrollo de este proyecto se hizo el consumo de estos servicios vía Kibana para hacer las configuraciones de índices y los imports masivos, vía postman para hacer las pruebas de forma directa, y vía RestEasy en Java dentro de ChasquiServer API.

## 8. Referencias

- <https://levelup.gitconnected.com/docker-compose-made-easy-with-elasticsearch-and-kibana-4cb4110a80dd>
- <https://www.elastic.co/guide/en/elasticsearch/plugins/current/analysis-phonetic.html>
- <https://www.linkedin.com/pulse/elasticsearch-phonetic-algorithms-daniel-ranallo/>
- [https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-index\\_.html](https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-index_.html)
- <https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-bulk.html>