

# Tecnologías de Software

## Servicios Web RESTfull con Spring Boot

Barrionuevo Cesar, Bracero Alexis, León Jhordy, Sanchez Luis

Universidad de las Fuerzas Armadas - ESPE

Departamento de Eléctrica y Electrónica

Email: cbarrionuevo@espe.edu.ec, aabracer06@espe.edu.ec, jkleon@espe.edu.ec, lsanchez@espe.edu.ec

**Abstract:** The following document details the creation of a Web RESTfull (generation of a software architecture style, which can be used to create web applications respecting HTTP) through Spring Boot, in Netbeans IDE that facilitates the creation of an application.

**Resumen:** En el siguiente documento se detalla la creación de un Web RESTfull (generación de un estilo de arquitectura software, la cual se puede usar para crear aplicaciones web respetando HTTP) mediante Spring Boot, y Netbeans que nos facilita la creación de una aplicación

**Palabras Claves:** Web RESTfull, GET, POST.

### I. INTRODUCCIÓN

En la actualidad el desarrollo de servicios web se ha convertido en un estándar en la programación al momento que se requiere compartir o intercambiar información entre cliente, servidor; de allí nació la necesidad de usar un mecanismo basado en el estilo REST más amigable con el usuario. Debido a esa motivación y buscando la innovación se plantea la alternativa de implementar un servicio web rest con el uso de la infraestructura de Spring Boot, la cual elimina la necesidad de configurar la aplicación a trabajar con el uso de archivos XML.

### II. ESTADO DEL ARTE

Un servicio web es un sistema de software diseñado para soportar la interacción entre máquinas a través de una red. Se podría entender como un conjunto de aplicaciones o tecnologías que pueden interactuar en la web, su principal función es la de intercambiar datos entre sí; tanto los proveedores ofrecen sus servicios y los usuarios solicitan el servicio llamando a estos procedimientos utilizando la web (David Booth, s.f.). En el desarrollo de software propuesto en el documento de (Nguyen, 2015), los servicios IDE se utilizan para ayudar a los desarrolladores en las tareas de programación. Sin embargo, en las aplicaciones web dinámicas, proporcionar servicios IDE para dicho código incrustado es un desafío. Según (Terpak, 2014) REST en un tipo de arquitectura orientada a recursos para el desarrollo de servicios web y gracias al desarrollo de los servicios

web REST, no es necesario realizar una programación exclusiva para cada plataforma, ni limitada por el lenguaje, el hardware la ejecución sino que es suficiente con exponer el servicio web para ser consumido por los clientes. (Gutierrez, 2016) Menciona que Spring Boot es un framework que posee las bibliotecas necesarias para la creación de aplicaciones ejecutables basadas en Spring. Los componentes que posee Spring Boot permiten simplificar los pasos de la selección de las dependencias necesarias para el proyecto a desarrollar y su despliegue en el servidor.

### III. MARCO TEÓRICO

Un servicio REST no es una arquitectura software, sino un conjunto de restricciones con las que podemos crear un estilo de arquitectura software, la cual podremos usar para crear aplicaciones web respetando HTTP. Hoy en día la mayoría de las empresas utilizan API REST para crear servicios. Esto se debe a que es un estándar lógico y eficiente para la creación de servicios web. Según (Alvarez, 2013) las restricciones que definen a un sistema RESTful serían:

#### Cliente/Servidor

Esta restricción mantiene al cliente y al servidor débilmente acoplados.

#### Sin Estado

Aquí se dice que cada petición que recibe el servidor debería ser independiente, es decir, no es necesario mantener sesiones.

#### Cacheable

Debe admitir un sistema de almacenamiento en caché. La infraestructura de red debe soportar una caché de varios niveles.

#### Interfaz Uniforme

Define una interfaz genérica para administrar cada interacción que se produzca entre el cliente y el servidor de manera uniforme, lo cual simplifica y separa la arquitectura.

## Sistema de Capas

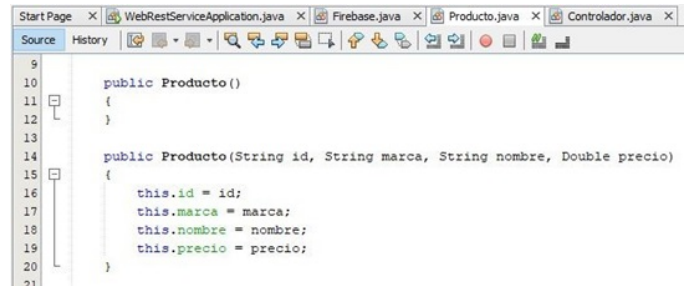
El servidor puede disponer de varias capas para su implementación. Esto ayuda a mejorar la escalabilidad, el rendimiento y la seguridad.

A continuación algunas características de una API REST:

- Las operaciones más importantes que permitirán manipular los recursos son cuatro: GET para consultar y leer, POST para crear, PUT para editar y DELETE para eliminar.
- El uso de hipermedios (término que en el ámbito de las páginas web define el conjunto de procedimientos para crear contenidos que contengan texto, imagen, vídeo, audio y otros métodos de información) para permitir al usuario navegar por los distintos recursos de una API REST a través de enlaces HTML.
- Lo más importante al crear nuestro servicio o API REST es que las respuestas a las peticiones se hagan en XML o JSON, ya que es el lenguaje de intercambio de información más usado.

Figura No 2

- Se crea los constructores.



```

9
10
11 public Producto()
12 {
13 }
14
15 public Producto(String id, String marca, String nombre, Double precio)
16 {
17     this.id = id;
18     this.marca = marca;
19     this.nombre = nombre;
20     this.precio = precio;
21 }

```

Figura No 3

**Nota:** Cabe recalcar que esta es una clase de persistencia ya que se va a tener los mismos atributos que existirán en la base de datos en este caso Fire Base.

- Se crea los getters y setters para posteriormente poder ser llamados.

## IV. DIAGRAMAS

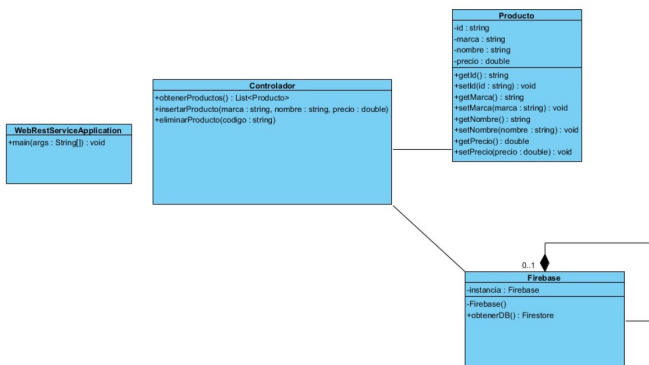
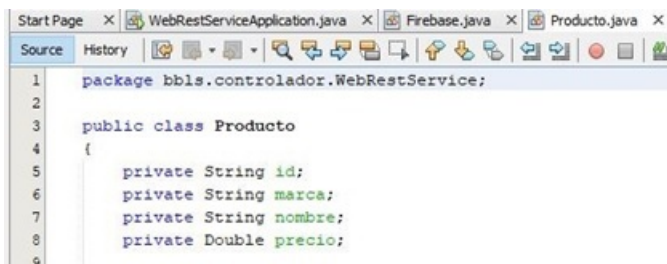


Figura No 1: Diagrama UML.

## V. EXPLICACIÓN DEL CÓDIGO FUENTE

Explicación del código en Java

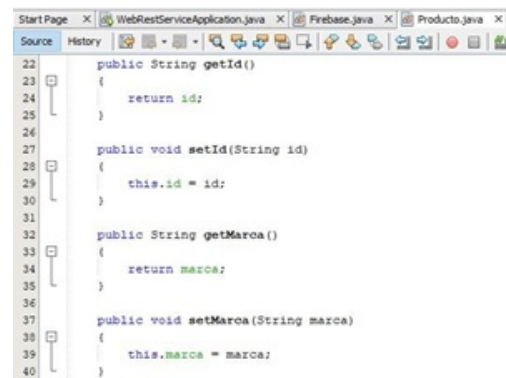
- Se debe crear una clase llamada “Producto” la cual tiene los atributos de un producto como su (id, Marca, Nombre, Precio).



```

1 package bbls.controlador.WebRestService;
2
3
4 public class Producto
5 {
6     private String id;
7     private String marca;
8     private String nombre;
9     private Double precio;

```

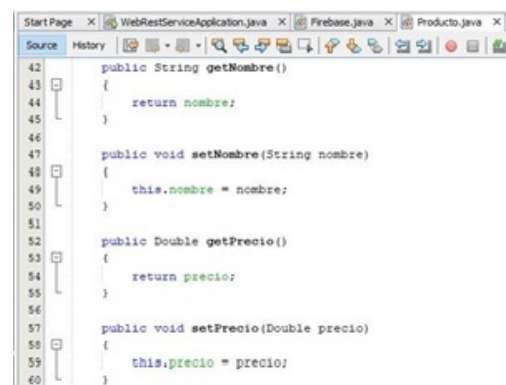


```

22 public String getId()
23 {
24     return id;
25 }
26
27 public void setId(String id)
28 {
29     this.id = id;
30 }
31
32 public String getMarca()
33 {
34     return marca;
35 }
36
37 public void setMarca(String marca)
38 {
39     this.marca = marca;
40 }

```

Figura No 4



```

42 public String getNombre()
43 {
44     return nombre;
45 }
46
47 public void setNombre(String nombre)
48 {
49     this.nombre = nombre;
50 }
51
52 public Double getPrecio()
53 {
54     return precio;
55 }
56
57 public void setPrecio(Double precio)
58 {
59     this.precio = precio;
60 }

```

Figura No 5

- Luego se debe crear una clase llamada FireBase, la cual servirá para realizar la conexión a la base de datos, pero antes se debe colocar las dependencias para poder manejar todas las clases de Firebase.

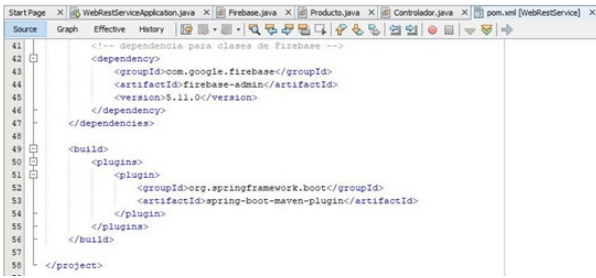


Figura No 6

- Dentro de la clase FireBase se crea un método, en este caso llamado obtenerDB(), este método permite obtener la instancia, cuando no hay nada esta nulo y cuando existe algo retorna a la instancia y asegura que se conecte solo a una base de datos.

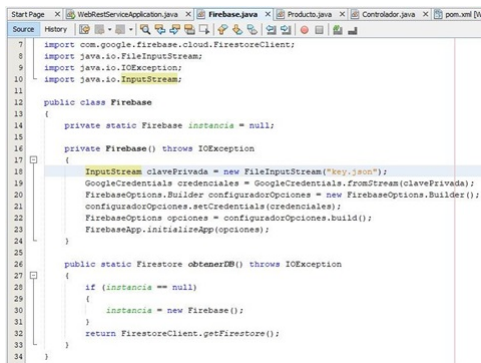


Figura No 7

- En la clase principal creada con el nombre de WebRestServiceApplication, se puede observar el método que ejecuta el servicio; el framework está diseñado para que ejecute ya una base de datos y gestione automáticamente la conexión, en otras palabras para evitar error de compilación por datasource inexistente.

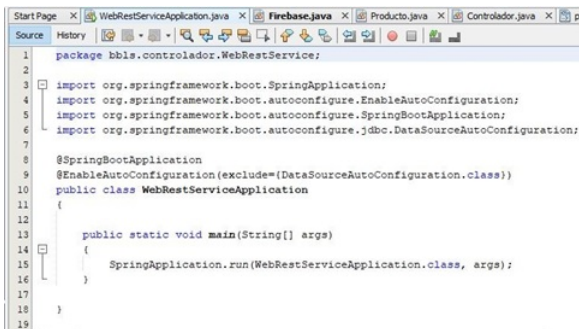


Figura No 8

- A continuación se crea la clase Controlador en la cual se va a desarrollar el servicio rest. El primer fragmento de código sirve para ingresar todos los atributos que contiene un producto para que así se guarden en la base de datos a través del objeto collection. Para consultar todos los atributos que se han ingresado y guardado en la base de datos, luego se procede a hacer una instancia del servicio creando un método, es así que se crea el método obtenerProductos el mismo que retorna todos los productos que están ingresados, se obtiene la instancia de la base de datos y esa base de datos va a tener una colección los cuales se guardan en "resultado".

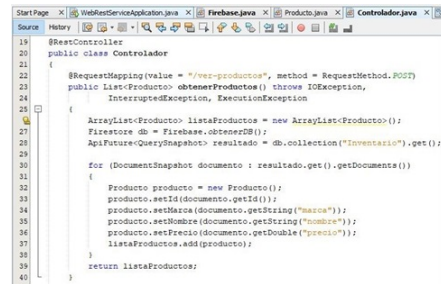


Figura No 9

- Luego de ello dentro Controlador se crea el método insertarProducto que como su nombre lo indica será el encargado de recibir producto por producto y transportarlos a una lista de productos llamado inventario.

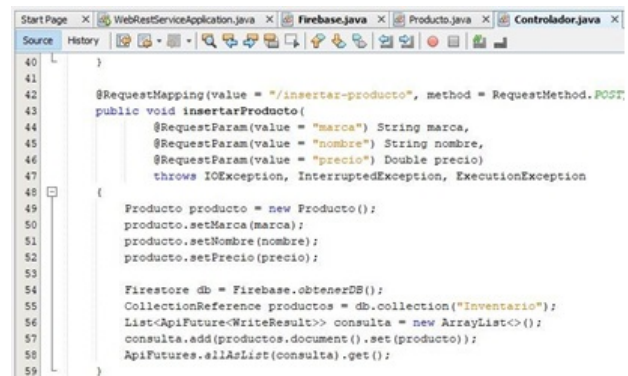


Figura No 10

- Finalmente dentro de la misma clase Controlador, se crea el método llamado eliminarProducto el cual se encarga de eliminar el producto del inventario a través de la id correspondiente a cada producto.

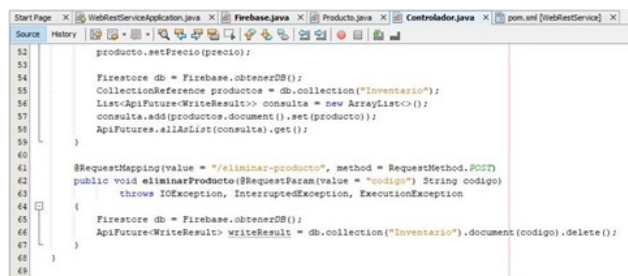


Figura No 11

## VI. CONCLUSIONES

- Luego de realizar la investigación se puede concluir que la importancia de un servicio web Rest se basa en que en la actualidad no existe proyecto o aplicación que no disponga de una API REST para la creación de servicios profesionales a partir de ese software. Hay cientos de empresas que generan negocio gracias a REST y las APIs REST. Sin ellas, todo el crecimiento en horizontal sería prácticamente imposible. Esto es así porque REST es el estándar más lógico, eficiente y habitual en la creación de APIs para servicios de Internet.
- Utilizar Spring Boot es de gran ayuda para los desarrolladores web, ya que le permite al programador

preocuparse en el desarrollo de la aplicación, la creación de dependencias y el despliegue en servidor es cosa del framework.

- El tener acceso a una base de datos (Firebase) para el servicio web creado con Netbeans y con la ayuda de Spring Boot es de gran ayuda e importancia ya que permite verificar que los métodos del servicio web se están ejecutando adecuadamente.

## VII. RECOMENDACIONES

- Se recomienda verificar todos los prerequisites establecidos, en especial la versión del JDK, ya que, sin la versión necesaria, de 64 bits, no se podrá establecer la conexión con la base de datos utilizada.
- Existen otros frameworks como Node.js, pero se recomienda trabajar con Spring Boot ya que se ha desarrollado con Java durante años y en cuanto a seguridad es más robusto.
- Es recomendable trabajar el consumo del servicio web RESTful a través de un programa que nos permita la interacción con el usuario como lo es Postman ya que es mucho más amigable y entendible que trabajar mediante el navegador.

## VIII. REFERENCIAS

- Alvarez, C. (2013). Introducción a servicios REST. *ArquitecturaJava*, 4(3), 34–35, Recuperado de: <https://www.arquitecturajava.com/servicios-rest/>.
- BBVA. (2016). API REST. *API MARKET*, 4(3), 2–3, <https://bbvaopen4u.com/es/actualidad/api-rest-que-es-y-que-son-sus-ventajas-en-el-desarrollo-de-proyectos>.
- David Booth, H. H. (s.f.). *Arquitectura de servicios web*.
- Gutierrez, F. (2016). *Pro Spring Boot*.
- Lazaro, D. (2015). Introducción a los web services. *IEEE*, 5(1).
- Nguyen, H. V. (2015). Varis: Soporte IDE para el código de cliente incorporado en aplicaciones web PHP. *IEEE*, 8(3), 896–906, Recuperado de: <https://ieeexplore.ieee.org/document/7203045/authors/authors>.
- Terpak, J. (2014). La propuesta de servicio web para soporte de cálculos termoquímicos específicos. *IEEE*, 5(1), 63–75, Recuperado de: <https://ieeexplore.ieee.org/document/6843677/authors/authors>.