**Análisis, diseño y pruebas**

Control de versiones

|  |  |  |  |
| --- | --- | --- | --- |
| Versión | Fecha | Autor | Descripción |
| 1.0 | 10/09/2023 | Mario Fernández Puerto | Primera versión del documento |

# Introducción

Este documento proporciona la descripción de los requisitos y detalles del diseño para la creación de un nuevo endpoint API REST solicitada por el cliente para la consulta de precios.

## Petición

La petición expresa es la siguiente:

En la base de datos de comercio electrónico de la compañía disponemos de la tabla PRICES que refleja el precio final (pvp) y la tarifa que aplica a un producto de una cadena entre unas fechas determinadas. A continuación se muestra un ejemplo de la tabla con los campos relevantes:

PRICES

-------

BRAND\_ID START\_DATE END\_DATE PRICE\_LIST PRODUCT\_ID PRIORITY PRICE CURR

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

1 2020-06-14-00.00.00 2020-12-31-23.59.59 1 35455 0 35.50 EUR

1 2020-06-14-15.00.00 2020-06-14-18.30.00 2 35455 1 25.45 EUR

1 2020-06-15-00.00.00 2020-06-15-11.00.00 3 35455 1 30.50 EUR

1 2020-06-15-16.00.00 2020-12-31-23.59.59 4 35455 1 38.95 EUR

Campos:

BRAND\_ID: foreign key de la cadena del grupo (1 = ZARA).

START\_DATE , END\_DATE: rango de fechas en el que aplica el precio tarifa indicado.

PRICE\_LIST: Identificador de la tarifa de precios aplicable.

PRODUCT\_ID: Identificador código de producto.

PRIORITY: Desambiguador de aplicación de precios. Si dos tarifas coinciden en un rago de fechas se aplica la de mayor prioridad (mayor valor numérico).

PRICE: precio final de venta.

CURR: iso de la moneda.

Se pide:

Construir una aplicación/servicio en SpringBoot que provea una end point rest de consulta tal que:

Acepte como parámetros de entrada: fecha de aplicación, identificador de producto, identificador de cadena.

Devuelva como datos de salida: identificador de producto, identificador de cadena, tarifa a aplicar, fechas de aplicación y precio final a aplicar.

Se debe utilizar una base de datos en memoria (tipo h2) e inicializar con los datos del ejemplo, (se pueden cambiar el nombre de los campos y añadir otros nuevos si se quiere, elegir el tipo de dato que se considere adecuado para los mismos).

Desarrollar unos test al endpoint rest que validen las siguientes peticiones al servicio con los datos del ejemplo:

- Test 1: petición a las 10:00 del día 14 del producto 35455 para la brand 1 (ZARA)

- Test 2: petición a las 16:00 del día 14 del producto 35455 para la brand 1 (ZARA)

- Test 3: petición a las 21:00 del día 14 del producto 35455 para la brand 1 (ZARA)

- Test 4: petición a las 10:00 del día 15 del producto 35455 para la brand 1 (ZARA)

- Test 5: petición a las 21:00 del día 16 del producto 35455 para la brand 1 (ZARA)

Se valorará:

Diseño y construcción del servicio.

Calidad de Código.

Resultados correctos en los test.

# Requisitos identificados

|  |  |  |
| --- | --- | --- |
| Petición de servicio | Requisitos usuario | Área |
| REQ\_FUN\_001 | Utilización BBDD H2 | Tecnología |
| REQ\_FUN\_002 | Utilización Spring Boot | Tecnología |
| REQ\_FUN\_003 | Creación de Entidad “PRICES” con los campos siguientes: BRAND\_ID: foreign key de la cadena del grupo (1 = ZARA).  START\_DATE , END\_DATE: rango de fechas en el que aplica el precio tarifa indicado.  PRICE\_LIST: Identificador de la tarifa de precios aplicable.  PRODUCT\_ID: Identificador código de producto.  PRIORITY: Desambiguador de aplicación de precios. Si dos tarifas coinciden en un rago de fechas se aplica la de mayor prioridad (mayor valor numérico).  PRICE: precio final de venta.  CURR: iso de la moneda. | Diseño |
| REQ\_FUN\_004 | Creación de Entidad “BRAND” (cadena) como consecuencia del anterior requisito | Diseño |
| REQ\_FUN\_005 | Parámetros de entrada del API REST de consulta: START\_DATE, PRODUCT\_ID y BRAND\_ID | Diseño |
| REQ\_FUN\_006 | Parámetros de salida del API REST de consulta: PRODUCT\_ID, BRAND\_ID, PRICE\_LIST, START\_DATE, END\_DATE y PRICE | Diseño |
| REQ\_FUN\_007 | Ejecución de las pruebas siguientes:  - Test 1: petición a las 10:00 del día 14 del producto 35455 para la brand 1 (ZARA)  - Test 2: petición a las 16:00 del día 14 del producto 35455 para la brand 1 (ZARA)  - Test 3: petición a las 21:00 del día 14 del producto 35455 para la brand 1 (ZARA)  - Test 4: petición a las 10:00 del día 15 del producto 35455 para la brand 1 (ZARA)  - Test 5: petición a las 21:00 del día 16 del producto 35455 para la brand 1 (ZARA)  Se entiende que se refiere al mes de junio y al año 2020. | Testing |

# Descripción de los cambios

## Módulos impactados

|  |  |  |  |
| --- | --- | --- | --- |
| Versión Software | Módulo | Descripción | Fecha |
| 1.0 | PricesController.java | Controlador REST | 10/09/2023 |
| 1.0 | Brand.java | Entidad que representa la tabla BRANDS | 10/09/2023 |
| 1.0 | Price.java | Entidad que representa la entidad PRICES | 10/09/2023 |
| 1.0 | PriceDTO.java | Data Transfer Object que representa la información a devolver a la vista | 10/09/2023 |
| 1.0 | BrandsRepository.java | Repositorio que implementa los métodos de obtención de datos del modelo de BRANDS. | 10/09/2023 |
| 1.0 | PricesRepository.java | Repositorio que implementa los métodos de obtención de datos del modelo de PRICES. | 10/09/2023 |
| 1.0 | BrandsService.java | Servicio del modelo de BRANDS que se comunica con el controlador. | 10/09/2023 |
| 1.0 | PricesService.java | Servicio del modelo de PRICES que se comunica con el controlador, teniendo lógica de filtrado de la información. | 10/09/2023 |
| 1.0 | ProductPriceServiceApplication.java | Clase de Spring Boot que contiene la carga inicial de los datos proporcionados. | 10/09/2023 |
| 1.0 | SpringFoxConfig.java | Fichero de configuración para SwaggerUI (OpenAPI) | 10/09/2023 |
| 1.0 | application.properties | Fichero de configuración de la aplicación. | 10/09/2023 |
| 1.0 | apidoc.yaml | Fichero YAML con la documentación del API REST. | 10/09/2023 |
| 1.0 | pom.xml | Fichero de configuración de dependencias MAVEN | 10/09/2023 |

## Descripción técnica ampliada por modulo modificado

El diagrama de diseño de la aplicación es el siguiente:



En relación con los módulos indicados en el punto anterior tenemos la estructura siguiente:



El proceso de construcción de la solución ha sido ir codificando en Java al mismo tiempo que se construía el documento “apidoc.yaml”.

Se ha seguido la guía siguiente para generar la documentación:

<https://www.baeldung.com/swagger-2-documentation-for-spring-rest-api>

Se optó por crear la entidad Price.java con todos los atributos de la tabla PRICES, para no tener pérdida de información, pero para retornar a la vista el resultado de la búsqueda, para ceñirnos al REQ\_FUN\_006, se creó una clase PriceDTO.java (Data Transfer Object), que cumple con los requisitos de salida.

Por tener la lógica de filtrado en el modelo, se optó por implementar dicha lógica en la clase PricesService.java, dentro del método findByBrandIdAndStartDateAndProductId. Consistente en:

1. Se obtiene una lista con todos los precios que tienen el productId y brandId indicados.
2. De la lista obtenida en el punto anterior, se obtiene una lista de precios vigentes según el startDate indicado.
3. De la lista obtenida anterior, se obtiene un solo precio que será el que mayor priority tenga.

Con esto se logra obtener un único precio para: la fecha indicada, el productId y el brandId indicados.

En la fase de pruebas, se detectó un problema con las franjas horarias. Para solventarlo, se estableció la zona horaria por defecto en el application.properties:

spring.jackson.time-zone=Europe/Madrid

Se ha descartado implementar rate limiting ni el uso de Hateoas, puesto que no hay requisitos al respecto y el volumen de datos de la información devuelta a la vista representa no es grande.

Los datos de acceso a la base de datos configurada son:

Ruta: http://localhost:8080/h2-console

URL: jdbc:h2:mem:testdb

Usuario: profile

Contraseña: pass

Interfaz de usuario gráfica

Descripción generada automáticamente

Tabla

Descripción generada automáticamente con confianza media

Al ser un proyecto de prueba, sin carga de datos confidenciales o sensibles a GDPR se ha descartado el empleo de contraseñas fuertes o a la comunicación segura de los datos de acceso a los datos.

# Pruebas

A continuación, se documentan las pruebas realizadas. Han sido ejecutadas con Postman, se adjunta la exportación de las pruebas en la ruta “docs”.

Tal y como se indica en el requisito REQ\_FUN\_007, se ejecutan las pruebas solicitadas. A continuación, se adjuntan evidencias:

Test1:

petición a las 10:00 del día 14 del producto 35455 para la brand 1 (ZARA)

Captura de pantalla de un celular

Descripción generada automáticamente

Test2:

petición a las 16:00 del día 14 del producto 35455 para la brand 1 (ZARA)

Captura de pantalla de un celular

Descripción generada automáticamente

Test3:

petición a las 21:00 del día 14 del producto 35455 para la brand 1 (ZARA)

Captura de pantalla de un celular

Descripción generada automáticamente

Test4:

petición a las 10:00 del día 15 del producto 35455 para la brand 1 (ZARA)

Captura de pantalla de un celular

Descripción generada automáticamente

Test5:

petición a las 21:00 del día 16 del producto 35455 para la brand 1 (ZARA)

Captura de pantalla de un celular

Descripción generada automáticamente