# Prueba Técnica Precios Info

**Ramiro Perez Sanz**

## Descripcíon

Este proyecto desarrolla una API RESTful con Spring Boot para gestionar y calcular precios en diversas cadenas de tiendas. La API toma tres parámetros clave (fecha, ID de producto e ID de cadena), procesa estos datos mediante una lógica de negocio compleja y algoritmos de priorización, y devuelve un precio final y metadatos relacionados en un objeto JSON.

1. **Configuración Inicial a través de Spring Initializ**r: Se emplea Spring Initializr para establecer la configuración básica del proyecto, añadiendo las dependencias requeridas.
2. **Diseño Orientado a la API**: Se prioriza la definición de la API antes de proceder con la implementación, garantizando que se ajuste a los requisitos de la evaluación técnica.
3. **Codificación de la Lógica Empresarial**: Implementación del código que cumple con los escenarios de prueba previamente definidos.
4. **Biblioteca Postman Disponible**: Se proporciona una colección de Postman para simplificar la interacción y el testeo del servicio web.
5. **Gestión Unificada de Excepciones**: Utilización de un GlobalExceptionHandler para tratar las excepciones de forma coherente y centralizada.
6. **Excepciones a Medida**: Desarrollo de una excepción específica para casos en los que no se pueda determinar un precio adecuado.
7. **Se configura un archivo YAML:** Para la integración de OpenAPI, permitiendo la visualización e interacción con la API mediante la interfaz de Swagger en un navegador web.
8. **Se emplean objetos de transferencia de datos (DTO)**: Se realizan operaciones de mapeo para facilitar la integración entre las distintas capas de la aplicación.

Mejoras

1. **Logging con SLF4J**: Incorporación de logs con configuraciones dinámicas para facilitar el monitoreo y diagnóstico.
2. **Dockerización**: Adición de un Dockerfile y configuraciones de docker-compose para facilitar el despliegue con un solo comando.
3. **Manejo Amigable** de Errores: Implementación de respuestas de error customizadas para la experiencia del usuario.
4. **Ampliacion de datos** : Se crea los CRUD de dos tablas detectadas.

Mejoras no implementadas

1. **Caching de Precios**: Implementación de una estrategia de almacenamiento en caché para mejorar el rendimiento de las consultas de precios más frecuentes.
2. **Autenticación y Autorización**: Incorporación de mecanismos de seguridad como OAuth2 o JWT para restringir el acceso a la API basado en roles o permisos.
3. **Rate Limiting**: Introducción de límites de tasa de solicitud para proteger la API contra el abuso y garantizar un servicio equitativo para todos los usuarios.

Notas

Tambien se esta añadiendo el Id y se deberia hacer autoincrementable desde la base de  datos para acotar el tiempo.

Por los tiempos no he podido corregir el tema de la Z para la zona horaria, pero sé que es crucial para garantizar la sincronización precisa de los microservicios y evitar problemas relacionados con la consistencia de datos. Planeo abordar este asunto implementando una solución que utilice la biblioteca java.time para manejar las zonas horarias de manera efectiva.