Resumen Técnico de la Prueba Precios Text

martes, 12 de septiembre de 2023 0:07

Descripción General

El objetivo del proyecto era construir una API RESTful utilizando Spring Boot para gestionar precios de productos en una cadena de tiendas. La API toma una fecha de aplicación, un identificador de producto y un identificador de cadena como parámetros de entrada y devuelve el precio aplicable junto con metadatos adicionales.

Características Técnicas y Funcionalidades

1. **Inicialización de Proyecto con Spring Initializr**: Utilización de Spring Initializr para la configuración inicial del proyecto, incluyendo las dependencias necesarias.
2. **Enfoque Api-First**: Diseño de la API antes de la implementación para asegurar que cumpla con los requisitos especificados en la prueba técnica.
3. **Desarrollo Guiado por Comportamiento (BDD)**: Uso del lenguaje Gherkin para definir escenarios de prueba, seguido de la implementación del código necesario para hacer que esos escenarios pasen.
4. **Arquitectura N-Layer**: Debido a la simplicidad del proyecto, se optó por una arquitectura n-layer en lugar de una arquitectura hexagonal.
5. **Creación de Pruebas**: Desarrollo de pruebas para validar los diferentes casos de uso descritos en la especificación de la prueba.
6. **Implementación de la Lógica de Negocios**: Desarrollo del código necesario para satisfacer los escenarios de prueba definidos.
7. **Colección Postman**: Disponibilidad de una colección de Postman para facilitar la exploración y prueba del servicio.
8. **Manejo Centralizado de Excepciones**: Implementación de un GlobalExceptionHandler para manejar excepciones de manera uniforme y centralizada.
9. **Excepciones Personalizadas**: Creación de una excepción personalizada para situaciones donde no se encuentra un precio aplicable.

Mejoras Adicionales

1. **Logging con SLF4J**: Incorporación de logs con configuraciones dinámicas para facilitar el monitoreo y diagnóstico.
2. **Pruebas Unitarias con Mockito**: Implementación de pruebas unitarias utilizando el framework Mockito para asegurar la calidad del código.
3. **Dockerización**: Adición de un Dockerfile y configuraciones de docker-compose para facilitar el despliegue con un solo comando.
4. **Manejo Amigable de Errores de API**: Implementación de respuestas de error más amigables para mejorar la experiencia del usuario.
5. **Rate Limiting**: Inclusión de una funcionalidad de limitación de tasa para proteger contra el abuso del API.

Tecnologías Utilizadas

- Spring Boot
- H2 Database (In-memory)
- Mockito
- Gherkin (para BDD)
- Cucumber

- Swagger
- Docker
- SLF4J
- Postman

Funcionalidades y Mejoras Contempladas pero No Implementadas

1. **Autenticación y Autorización**: Incorporación de JWT para restringir el acceso a ciertas partes de la API.
2. **Cacheo de Resultados**: Utilización de un sistema como Redis para mejorar el rendimiento del servicio.
3. **Internacionalización (i18n)**: Soporte para múltiples idiomas y formatos, especialmente para los mensajes de error y las monedas.
4. **Integración Continua y Despliegue Continuo (CI/CD)**: Implementación de pipelines de CI/CD para automatizar las pruebas y el despliegue.
5. **Monitorización y Telemetría**: Uso de herramientas como Grafana y Prometheus para monitorización en tiempo real.
6. **Pruebas de Estrés y Carga**: Evaluación del comportamiento del sistema bajo condiciones extremas.
7. **Balanceo de Carga**: Implementación de un balanceador de carga para mejorar la eficiencia del servicio.