Examen UD4 - UD5: API REST con Spring Boot

Nombre:
Fecha:

RA 3: Escribe bloques de sentencias embebidos en lenguajes de marcas, seleccionando y utilizando las estructuras de programación.

RA 4: Desarrolla aplicaciones web embebidas en lenguajes de marcas analizando e incorporando funcionalidades según especificaciones.

RA 5: Desarrolla aplicaciones web identificando y aplicando mecanismos para separar el código de presentación de la lógica de negocio.

ENUNCIADO

Se debe diseñar e implementar un servicio de backend llamado calculadora que exponga una API REST para realizar operaciones matemáticas básicas, manteniendo un registro histórico del rendimiento por cada usuario.

Lógica de Operación

La aplicación debe soportar las siguientes operaciones:

- Suma (operando1 + operando2)
- Resta (operando1 - operando2)
- Multiplicación (operando1 * operando2)
- División (operando1 / operando2)
- Potencia (operando1 elevado a operando2)
- Módulo (Resto de la división: operando1 % operando2)
- Raíz Cuadrada (Solo usa operando1)

Cada una de las operaciones debe ser implementada como una unidad de lógica independiente y reutilizable.

El módulo principal que procesa la solicitud debe acceder e invocar dinámicamente cualquiera de estas unidades.

Nota sobre la raíz cuadrada: para la operación raíz cuadrada, el sistema debe ignorar el valor de operando2 y solo utilizar operando1.

Datos de entrada y salida

Define una estructura de datos para recibir la solicitud que incluya el nombre de la operación y dos operandos numéricos (operando1, operando2).

Define una estructura de respuesta única para todos los resultados. Esta estructura debe comunicar el resultado numérico en caso de éxito o, alternativamente, un código/etiqueta descriptiva del error en caso de fallo, manteniendo el otro campo como nulo. Prohibido utilizar mecanismos de manejo de excepciones para reportar fallos.

Gestión de respuestas y errores

El endpoint de la API debe devolver una respuesta que refleje el resultado del proceso. El sistema debe manejar explícitamente los siguientes casos de error de negocio:

- DIVISION_POR_CERO: ocurre en la división y el módulo si operando2 es cero.
- OPERANDO_NEGATIVO: ocurre si se intenta calcular la raíz cuadrada de un número negativo.
- OPERACION_INVALIDA: la operación solicitada no existe.
- OPERANDO_NULO: faltan los operandos necesarios.

La respuesta HTTP debe ser:

- 200 OK: Operación correcta.
- 400 Bad Request: errores de petición (OPERACION_INVALIDA, OPERANDO_NULO).
- 422 Unprocessable Entity: errores de lógica de negocio (DIVISION_POR_CERO, OPERANDO_NEGATIVO).

Persistencia de estado por sesión

La aplicación debe llevar un conteo de las operaciones realizadas. Este conteo debe ser persistente por cada usuario a lo largo de múltiples peticiones.

Se deben registrar dos contadores:

- Número de operaciones finalizadas con éxito.
- Número de operaciones finalizadas con error.

Implemente una opción que permita consultar los valores actuales de estos contadores para la sesión activa del usuario.

Además también se deberá implementar la funcionalidad de resetear los contadores de operaciones con éxito y error a cero.

Ejemplo de funcionamiento

Entrada

```
{
    "operacion": "SUMA",
    "operando1": 15.0,
    "operando2": 7.5
}
```

Salida

200 OK

```
{
    "error": "SIN_ERROR",
    "resultado": 24.0
}
```

Operaciones realizadas

200 OK

```
{
    "operaciones_correctas": 1,
    "operaciones_erroneas": 0
}
```

Entrada

```
{
    "operacion": "DIVISION",
    "operando1": 10.0,
    "operando2": 0.0
}
```

Salida

422 Unprocessable Entity

```
{
    "error": "DIVISION_POR_CERO",
    "resultado": null
}
```

Operaciones realizadas

200 OK

```
{
    "operaciones_correctas": 1,
    "operaciones_erroneas": 1
}
```