Almacén

Estamos desarrollando una servicio para la gestión del almacenamiento de productos en una tienda.

Estos productos se dividen en tres categorías:

- Normales: No pueden producir ni ganancias ni pérdidas.
- Añejos: Pueden producir ganancias en base a la antigüedad de sus existencias.
- Perecederos: Producen producir pérdidas en base a la fecha de caducidad de sus existencias.

El servicio dispone de los siguientes endpoints:

- GET /Store => obtiene inventario de productos y sus existencias.
- PATCH /Store/SetStockPrice => actualiza precio (pérdida/ganancia) de las existencias.
- GET /Store/balance => obtiene el balance de pérdidas y ganancias generadas por el almacenamiento de los productos.

Reglas para calcular pérdida/ganancia:

Añejos => no aplica pérdida.

La ganancia se calcula en base a estos criterios:

- Edad: años transcurridos desde su fecha de fabricación.
- AñosAlmacenado: años transcurridos desde la fecha de entrada en almacén.
- AñosCalculo = edad - añosAlmacenado
    - AñosCalculo 1-5 => se aplica un beneficio del 5% sobre el precio.
    - AñosCalculo 6-10 => se aplica un beneficio del 10% sobre el precio.
    - AñosCalculo > 10 => se aplica beneficio por coeficiente (1 + (AñosCalculo/100)) sobre el precio.

Perecederos => no aplica ganancia.

La pérdida se calcula en base a estos criterios:

- <= 1 día para caducar => se aplica pérdida = precio.
- Entre 2 y 3 días para caducar => se aplica pérdida = precio / 2.
- Entre 4 y 5 días para caducar => se aplica pérdida = precio / 4.

Normales => no aplica ni pérdida ni ganancia.

Objetivos a realizar

1.- Refactorizar el código del método "GetBalance" que calcula el balance de pérdidas/ganancias dentro del controlador "StoreController.cs" con el objetivo de facilitar su legibilidad y mantenimiento.

2.- Añadir código en el método "SetStocksPrice" del controlador "StoreController.cs" para actualizar el precio de todo el inventario de la tienda aplicando pérdida o ganancia a una fecha opcionalmente dada (por defecto fecha actual) y categoría opcional (por defecto todas).

En el desarrollo de los puntos anteriores, tener en cuenta lo siguiente:

- No se podrán cambiar la firma de los métodos actuales de la clase "StoreController.cs"
- No se podrá cambiar ninguna clase de la carpeta "Data"
- No se podrán alterar los test existentes en la clase "StoreControllerTests.cs"
- Se podrán crear nuevas clases y organizarlas como se considere conveniente.
- Las métodos públicos de las nuevas clases creadas deberán cubrirse mediante tests unitarios.
- Todos los tests deberán pasar con el código final presentado.