Informe sobre las resoluciones empleadas para los siguientes puntos.

AGREGAR UN ÍTEM A UNA ORDEN YA CREADA

Una orden debería tener ítems de un único proveedor. Es decir, en la misma orden todos los productos deberían ser del mismo proveedor.

El endpoint, informa el id del proveedor seleccionado y el id de la orden creada, a la cual se le quiere agregar un ítem. Y en el body el nuevo ítem.

La orden debe estar en estado “Pending” para que se le pueda agregar un nuevo ítem. En caso contrario larga una excepción.

Además también se controla que el producto del nuevo ítem pertenezca al mismo proveedor “dueño” de la orden, informado en el endpoint. Si el proveedor no es el mismo, se lanzará una excepción.

Si el producto es del proveedor, al ítem se le setea el producto, y a la orden se le agrega el ítem. Además, se actualiza el nuevo precio de la orden.

Al salvar la orden, se actualiza en la base de datos la información relacionada con orden e ítems.

Relación Orden con Item: @OneToMany unidireccional. Es una relación de composición. Desde una orden accedo a los ítems. Al ser una relación @OneToMany unidireccional, utilicé el @JoinColumn para que Hibernate no cree una tabla intermedia y use una clave foránea en la tabla del otro lado de la relación (ítems). El no tener una tabla intermedia mejoraría el rendimiento ya que nos evitaríamos como mínimo un insert y realizar un join para reunir la info entre order e ítems.

Relación Item con Product: @ManyToOne unidireccional. En la tabla ítems se guarda la clave compuesta foránea de producto.

CONFIRMAR UN PEDIDO.

Confirmar o Asignar una orden implica: buscar un repartidor activo y libre, resuelto con una query por el nombre del método. Si se encuentra un repartidor se trae la orden desde el repositorio y a la misma se le asigna el delivery man. En esta asignación el delivery pasa a no estar libre y el estado de la orden cambia de “Pending” a “Assigned”.

Luego se salva la orden y con esta acción se actualizan en la base de datos la info en orden y delivery man.

AÑADIR UNA CALIFICACIÓN A UNA ORDEN YA COMPLETADA

Este endpoint recibe en el path el id de una orden ya creada y en el body una calificación de la misma.

Para poder calificar una orden, esta debe estar en estado “Delivered” y no poseer una calificación previa. En caso contrario se lanzarán las excepciones correspondientes.

Si se cumplen estas condiciones: se trae del repositorio al proveedor de la orden, el cual se obtiene su id tomando el proveedor del producto del primer ítem de la orden.

Al proveedor se le setea la calificación: sumando el nuevo score. Y se le incrementa en uno la cantidad de órdenes calificadas. Asimismo, a la orden se le setea la calificación y se salva la orden. Al salvarse la orden, en la base de datos se actualiza la información de la orden y del proveedor.

Observaciones:

La clase calificación al ser @OneTo One con Order, resolví embeberla en la misma orden y no generar una tabla independiente.

En la clase proveedor me resultó más eficiente que en lugar de tener el promedio de las calificaciones, tenga dos campos: uno la suma de puntos/calificaciones y el otro el total de órdenes punteadas (O cliente que han calificado) y cuando se necesite el promedio hago la división y retorno.

ACTUALIZAR LOS DATOS DE UN PRODUCTO.

Recibe el id del proveedor y el name del producto a actualizar. En el body recibe los datos del producto que se deben actualizar. Podría cambiar el precio e incluso agregarle una nueva categoría de tipo, la cual ya existiría en la base.

El nombre de un producto es único para un proveedor. Por lo cual, la clave de un producto está formado por el id del proveedor y el nombre del producto.

Si cambió el precio: seteo la fecha de fin del precio anterior (ya que cuando el producto se creó, se agregó el precio en el historial con fecha de fin nula) y guardo el nuevo precio en el historial.

Al salvar el producto se replican todas estas actualizaciones en las tablas de producto, historical y la tabla intermedia que genera hibernate para las relaciones @ManyToMany entre producto y producto type.

Relación entre producto y producto type es @ManyToMany Unidireccional navegable desde producto a producto type. Ya que la clase principal es producto, la otra clase suma información al producto y se utiliza principalmente en las búsquedas. Además de que la lista de producto type de un producto no representaría problemas al traerla toda a memoria, es de esperar que sean solo unos pocos tipos. No así del otro lado de la relación.

Relación entre producto y el historial de precios es @OneToMany unidireccional de producto al historial.

ELIMINAR UN PRODUCTO DE LOS OFRECIDOS POR UN PROVEEDOR.

Recibe el id del proveedor y el name del producto a eliminar.

Si el producto figura en al menos una orden, el producto no será eliminado físicamente. Se desactivará. A fin de realizar esta operación se agregó una propiedad “active” en la clase producto.

En el caso que el producto no figure en ninguna orden, se eliminará el producto. (Toda la tupla).

La api siempre le mostrará al usuario productos activos.

OBTENER TODOS LOS PROVEEDORES DE UN CIERTO TIPO

Recibe en el path el id del proveedor type.

Resuelta con una query “named” en el repositorio de proveedores.

Relación entre el proveedor y proveedor type es @ManyToMany Unidireccional navegable desde proveedor a proveedor type. Ya que la clase principal es proveedor, la otra clase suma información al proveedor y se utiliza principalmente en las búsquedas. Además de que la lista de type de un proveedor no representaría problemas al traerla toda a memoria, es de esperar que sean solo unos pocos tipos. No así del otro lado de la relación.

OBTENER TODOS LOS PRODUCTOS Y SU TIPO, DE UN PROVEEDOR ESPECÍFICO.

Recibe en el path id del proveedor.

Resuelta con una query HQL, en la cual junto productos y sus tipos. Y filtro por proveedor. (la clave del producto contiene al proveedor además de su nombre).

La relación entre proveedor y productos es @OneToMany Unidireccional. Navegable desde proveedor.

OBTENER LAS ÓRDENES CON MÁS PRODUCTOS DE UN PROVEEDOR ESPECÍFICO.

Este endpoint recibe el id del proveedor.

Resuelto con una query HQL, (en la cual uso las clases en lugar de las tablas). Implementada en un repositorio customizado.

En el caso de haber más de un orden con el mismo valor máximo, devuelve una lista con sus ids de que igualan dicho valor.

OBTENER LA ORDEN DE MAYOR PRECIO TOTAL DE UN DÍA DADO.

Recibe la fecha como parámetro.

Resuelta con una query “named” en el repositorio de órdenes.

La consulta tiene en cuenta todas las órdenes emitidas en el día, sin considerar el estado de las mismas.

OBTENER LOS DIEZ REPARTIDORES CON MAYOR PUNTAJE.

Resuelto con una query “nombrada”, la cual consulta los delivery men activos y los ordena por el score en forma descendiente, tomando solo los primeros diez.

OBTENER LOS DIEZ PROVEEDORES QUE MÁS ÓRDENES DESPACHARON.

Resuelto con una query HQL. Implementada en un repositorio customizado.

En la query reúno ordenes, ítems, productos y proveedores. Las ordenes deben tener un estado “Delivered”. La lista resultante se entrega ordenada por cantidad de ordenes despachas en forma descendiente.

OBTENER LOS PRECIOS DE UN PRODUCTO ENTRE DOS FECHAS DADAS.

Recibe en el path el id del proveedor y el name del producto a consultar. Las dos fechas que componen el inicio y fin del período son recibidas como parámetros.

Resuelta con una query nativa. En la cual, junto las tablas de producto e historical.

Las condiciones por cumplir son: La fecha de comienzo de historical debe ser menor o igual que la fecha de fin del intervalo y la fecha de fin de historical debe ser mayor que la de comienzo del intervalo o nula (indicando que ese precio es el actual).

OBTENER EL PRECIO PROMEDIO DE LOS PRODUCTOS DE CADA TIPO, PARA TODOS LOS TIPOS.

Resuelto con una query HQL. Implementada en el repositorio de productos..

En la query reúno productos y productos types. Agrupando por tipo de productos y obteniendo en la misma consulta el precio promedio para cada tipo.

OBTENER LA INFORMACIÓN DE LOS PROVEEDORES QUE TENGAN AL MENOS UNA CALIFICACIÓN DE UNA ESTRELLA (LA MÁS BAJA). ES NECESARIO TAMBIÉN EL NÚMERO DE ESTAS CALIFICACIONES QUE EL PROVEEDOR POSEE.

Resuelto con una query HQL. Implementada en el repositorio de proveedores.

Si el número de calificaciones de un proveedor es mayor cero, el proveedor tendrá una calificación mínima de una estrella, ya que la calificación de los clientes van de una a cinco estrellas.

OBTENER LOS PROVEEDORES QUE OFREZCAN PRODUCTOS DE TODOS LOS TIPOS.

Resuelto con una query HQL. Implementada en el repositorio de proveedores.

En la query reúno proveedores , productos y productos types. Esta query recibe como parámetro la cantidad de tipos existentes en el repositorio de productos types. Este valor lo utiliza para realizar el filtrado.