TP5 Arquitecturas Web 2021
Tudai

Grupo 6

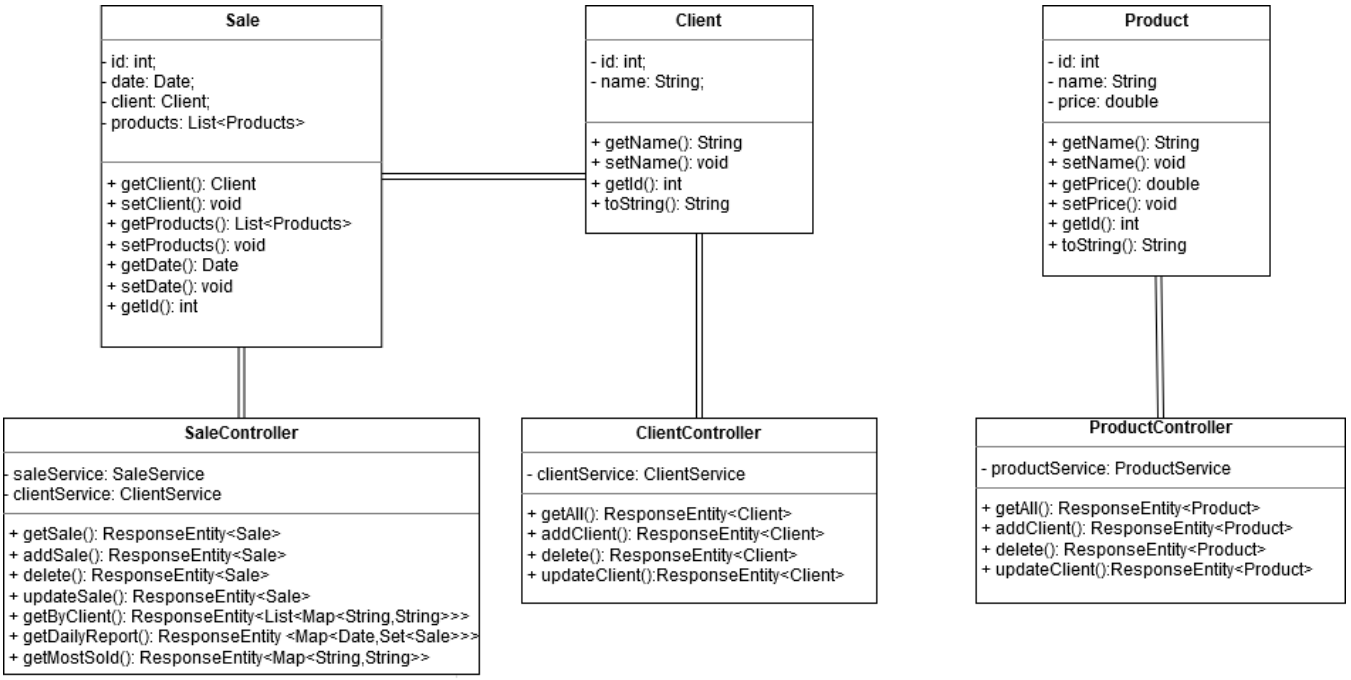
Barthes, Juan

Fuhr, Federico

Genaro, Facundo

Introducción:
Se nos solicitó hacer una aplicación realizada en Springboot para una despensa online.
Esta aplicación debía implementar los servicios básicos de manipulación de entidades (alta, baja y modificaciones) además de cumplir ciertos requisitos.

Base de la aplicación: primero hubo que pensar como íbamos a implementar las entidades, decidimos implementar tres entidades, las cuales fueron Product, Client y Sale.
Dichas entidades tendrían cada una su propio Controller el cual se encargaría de hacer los llamados necesarios a los servicios en el momento de obtener/modificar datos desde la BD.
Los controllers de Product y Client son simples ya que solo implementan los métodos básicos GET, PUT y POST.
La consigna también solicitaba poner un limite de tres unidades por compra por día, esto se implementó en el metodo AddSale en la clase SaleService donde antes de agregar la venta a la base de datos, previamente verificamos utilizando un HashMap donde la key es el ID del producto y el value es la cantidad de veces que este producto está dentro de la venta, una vez hecho esto, si el value del Map es mayor a tres, el elemento no se agrega a la base de datos.
En el siguiente diagrama se puede observar como quedaron las entidades con sus controladores (sin tener en cuenta los Servicios y Repositorios).



Requisito 1:
Generar un reporte donde se indiquen los clientes y el monto total de sus compras.
Para resolver este inciso se implementó un método dentro de SaleService que recibe una lista de todos los clientes existentes en la base de datos, a partir de esa lista de Clientes obtienen todas las ventas correspondientes a cada uno de ellos y luego realiza la suma de los montos de cada una de estas ventas para luego adjuntarla con su respectivo cliente dentro de la variable de retorno.

Requisito 2:
Generar un reporte con las ventas por día.
Para resolver este inciso se decidió hacer una implementación que retorne todas las ventas ordenadas por fecha (en un Map donde la Key es la fecha y el value es un set de ventas). Para empezar obtuvimos todas las fechas de las ventas y las agregamos a un Set de fechas, luego por cada una de esas fechas agregamos a la variable de retorno (un Map) la fecha correspondiente a cada venta como Key y un set de ventas como Value

Requisito 3:
Implementar una consulta para determinar cuál fue el producto más vendido.
Para resolver este inciso decidimos agregar cada una de las ventas a un Mapa donde cada producto sería la Key y el value sería la cantidad de veces que aparece dicho producto en la base de datos (por cada vez que se encontrara ese producto dentro de la base de datos, se le haría +1 al value de la entry que tenga ese producto como Key dentro del mapa).

Cambios para la segunda entrega:
Para la segunda parte de este trabajo se nos solicitó hacer tests de JUnit y documentar la api en swagger.
La documentación con swagger se hizo sin problemas, junto con sus respectivos comentarios. Se accede por la url https://tp-5.herokuapp.com/swagger-ui/
Para los test de JUnit decidimos probar los tres requisitos solicitados en la primer parte del trabajo, los tests fueron correctos y están dentro del trabajo.

Problemas que surgieron:
El principal problema fue pensar la implementación y armado del sistema requerido, ya que en un principio no teníamos claro cuantas entidades teníamos que hacer o en que clases deberíamos abstraer la lógica de negocios, por ejemplo en un principio implementamos mucha logica de negocios dentro del controlador, eso con el tiempo lo fuimos abstrayendo para dejarlo todo en el Service y el controlador quedó más y más limpio hasta llegar a la versión final.

Otra complicación que tuvimos fue con la implementación del limite de compras por día, la idea plasmada en el trabajo practico fue la que más nos convenció y la que mejor resultó de todas las que probamos.
A excepción de esos detalles, el trabajo no tuvo mayores complicaciones.