Bases de Datos II

Trabajo Práctico Integrador 3

ETAPA 2: Queries

Fecha de Publicación: 02/06
Fecha de Entrega: 16/06 (en conjunto con la etapa anterior)

Introducción

En esta segunda etapa del trabajo se deberán realizar consultas sobre el modelo persistido en MongoDB. Esta funcionalidad está especificada en la interfaz `DBliveryStatisticsService` y validada mediante la nueva clase de tests `DBliveryStatisticsTestCase` y los datos de prueba se cargarán desde el nuevo `DBInitializer`, ahora adaptado para funcionar con MongoDB.

Para resolver este trabajo deberán implementar las consultas en el **repositorio**. Se puede utilizar tanto la búsqueda simple mediante la funcionalidad ***find*** o búsquedas más complejas mediante el ***aggregation framework***, ambos soportados por el cliente de MongoDB para java.

Modificaciones necesarias en el modelo

1. Para facilitar las consultas geográficas sobre los pedidos, es necesario realizar una adaptación a la clase `Order` que consiste en:
   a. Agregar la siguiente variable de instancia
      ```
      private Point position;
      ```
   b. Agregar al constructor el siguiente código para inicializar dicha variable
      ```
      Position pos = new Position(coordX, coordY);
      this.position = new Point(pos);
      ```
   c. Incluir los siguientes *imports*:
      ```
      import com.mongodb.client.model.geojson.Point;
      import com.mongodb.client.model.geojson.Position;
      ```

2. En el método anotado con `@BeforeAll` dentro de la clase `DBliveryStatisticsTestCase`, se crea ahora un índice para la colección de las instancias de `Order`. **Revisar que el nombre de la colección coincida** con el de su implementación; por defecto es **'order'**, si su colección tiene el mismo nombre no hace falta hacer nada en este punto.

Consultas

Las consultas a implementar son las siguientes. La interfaz `DBliveryStatisticsService` contiene más detalles en formato Javadoc.

1. **`getAllOrdersMadeByUser(String username)`**
   Obtiene todas las ordenes realizadas por el usuario con username username

2. **`getTopNSuppliersInSentOrders(int n)`**
   Obtiene los n proveedores que más productos tienen en ordenes que están siendo enviadas

3. **`getPendingOrders()`**
   Obtiene el listado de las órdenes pendientes

4. **`getSentOrders()`**
   Obtiene el listado de las órdenes enviadas y no entregadas

5. **`getDeliveredOrdersInPeriod(Date startDate, Date endDate)`**
   Obtiene todas las órdenes entregadas entre dos fechas

6. **`getDeliveredOrdersForUser(String username)`**
   Obtiene todas las órdenes entregadas para el cliente con username username

7. **`getBestSellingProduct()`**
   Obtiene el producto con más demanda

8. **`getProductsOnePrice()`**
   Obtiene los productos que no cambiaron su precio

9. **`getSoldProductsOn(Date day)`**
   Obtiene los productos vendidos en un day

10. **`getMaxWeigth()`**
    Obtiene el producto más pesado

11. **`getOrderNearPlazaMoreno()`**
    Obtiene las órdenes que fueron realizadas a menos de 400 metros del centro de Plaza Moreno. Sus coordenadas son: [-34.921236,-57.954571]

Cambios en el proyecto

La infraestructura del proyecto provista se encuentra en el mismo repositorio del TP previo, **en la rama `mongo`**:

https://github.com/juliangrigera/bdlivery/tree/mongo

Presenta los siguientes cambios respecto de la etapa 1:

- Se agrega un nuevo método que pueden utilizar en el repositorio llamado `getObjectsAssociatedWith()`.
- Hay una nueva clase de tests `DBliveryStatisticsTestCase` , con los test que comprueban la implementación de la interfaz `DBliveryStatisticsService`.
- La clase `DBInitializer` que en esta etapa se utilizará para cargar los datos de prueba fue adaptada para funcionar con MongoDB.

Al igual que en la etapa 1, desde la línea de comandos debe obtenerse un build exitoso en donde pasen todos los tests.

Requisitos detallados que deben satisfacerse en esta etapa

- Escribir todo el código necesario para que la aplicación compile y todos los tests de `DBliveryMongoTestCase` y `DBliveryStatisticsTestCase` pasen. Esto implicará implementar la interfaz `DBliveryStatisticsService` en su implementación concreta de la interfaz `DBliveryService`. Es decir, el servicio implementará **ambas** interfaces.

- El código tiene que estar subido en el mismo repositorio sobre el branch `practica3`.