Bases de Datos II

Trabajo Práctico Integrador: Etapa 2

El objetivo de esta etapa es la confección de un conjunto de consultas HQL que satisfagan una serie de tests implementados por la clase `AirBdbStatisticsServiceTestCase`. Este test case inicializa la base de datos utilizando una clase utilitaria llamada `DBInitializer`, la cual a su vez utiliza la clase existente `AirBdbService` para crear un conjunto de usuarios, propiedades, reservas etc. Estos objetos servirán como un dataset existente para la realización de las consultas.

La especificación de las queries a implementar están descritas en la interfaz `AirBdbStatisticsService`, la cual debe deberá ser extendida por `AirBdbService`, implicando que `AirBdbServiceImpl` (o como se haya llamado la clase que implementa la interfaz `AirBdbService`) deberá implementar cada uno de sus métodos. Estos métodos están descriptos usando el estándar Javadoc y, a su vez, los resultados esperados de las queries luego de haber inicializado la base de datos con `DBInitializer` estarán validados por `AirBdbStatisticsServiceTestCase`.

Para validar la correcta completitud de esta etapa, todos los tests descritos en `AirBdbStatisticsServiceTestCase` deberán pasar satisfactoriamente y la ejecución del comando `mvn clean install` (que incluirá la ejecución de los tests definidos en la Etapa 1) deberá seguir siendo exitosa.

Los métodos a implementar en la interfaz `AirBdbStatisticsService` son los siguientes:

a. `getAllPropertiesReservedByUser(java.lang.String userEmail)`: Obtiene todas las propiedades que fueron reservadas por el usuario con username `userEmail`

b. `getApartmentTop3Ranking()`: Top 3 de `Apartments` con mejores calificaciones promedio

c. `getCitiesThatHaveReservationsBetween(java.util.Date from, java.util.Date to)`: Obtiene las ciudades que han tenido reservas entre las fechas `from` y `to`

d. `getHotmailUsersWithAllTheirReservationsFinished()`: Devuelve los nombres de usuarios de Hotmail (cuyo username termine en @hotmail.com) cuyas reservas estén todas finalizadas

e. `getMatchingUsersThatOnlyHaveReservationsInCities(java.lang.String usernamePart, java.lang.String... cities)`: Devuelve una lista de usuarios que hayan reservado sólo en el conjunto de ciudades cuyos nombres son descriptos en `cities` y cuyo username contenga `usernamePart`

f. `getMostExpensivePrivateRoomReservation()`: Obtiene la reserva de habitación privada (PrivateRoom) más cara de toda la plataforma

g. `getPropertiesThatHaveBeenReservedByMoreThanOneUserWithCapacityMoreThan(int capacity)`: Obtiene todas las propiedades con una capacidad mayor a `capacity` que han sido reservadas por más de un usuario en la plataforma

h. `getReservationsInCitiesForUser(java.lang.String username, java.lang.String... cities)`: Obtiene las reservas del usuario con username `username` en las ciudades `cities`

i. `getTotalRevenueForFinishedReservationsDuringYear(int year)`: Obtiene el importe total facturado por la plataforma en concepto de todas aquellas reservas que han sido finalizadas (es decir que no han sido canceladas ni están en espera de confirmación) durante un año (`year`) específico

j. `getUsersSpendingMoreThan(double amount)`: Obtiene todos los usuarios que han gastado más de `amount` en reservas en la plataforma

k. `getUsersThatReservedMoreThan1PropertyDuringASpecificYear(int year)`: Obtiene todos los usuarios que han reservado más de una vez en el año `year`

l. `getUsersThatReservedOnlyInCities(java.lang.String... cities)`: Obtiene los usuarios que realizaron reservas *sólo* en todas las ciudades cuyos nombres son listados en `cities`