**ANEXO EXAMEN DE CERTIFICACIÓN**

| | |
|---|---|
| Plan de Estudio | Desarrollo Aplicaciones Fullstack Java Trainee |
| Anexo | Caso BikeShop Catalog & Inventory |

# Caso "BikeShop Catalog & Inventory"

"BikeShop" es una empresa dedicada a la comercialización bicicletas, accesorios y repuestos. Es una tienda de nicho que busca satisfacer las necesidades de un exclusivo segmento de deportistas y entusiastas de alto rendimiento, para lo cual ofrece una gran variedad de productos con un excelente nivel de calidad.

La empresa se creó a inicio de los noventas como un emprendimiento de un grupo de amigos entusiastas de los deportes ciclísticos. La empresa ha tenido un crecimiento paulatino y hoy en día cuenta con 3 locales en Estados Unidos y una platafoma web que recibe pedidos de los clientes pero que no ha tenido mayores cambios durante los últimos años.

Dado que hoy en día las personas están optando por medios de transporte ecológicos, la empresa apuesta por un aumento de las ventas para los próximos 5 años razón por la cual se ha propuesto renovar sus sistemas de inventario, órdenes y catálogo web.

Para esto, el CTO (Chief Tecnological Officer) ha formado un equipo de proyectos de primera línea del cual usted forma parte como desarrollador full-stack Java. El equipo de proyectos también lo conforma un Jefe de Proyectos, un Diseñador UX/UI, un Diseñador Web, un Analista Funcional, un Desarrollador Mobile, y un Arquitecto de Software.

El proyecto busca, dentro de otras cosas, ordenar el sistema de inventario y catálogo de productos que utilizan los funcionarios para la gestión de los pedidos y ventas. A continuación, se listan los requisitos funcionales de alto nivel del sistema:

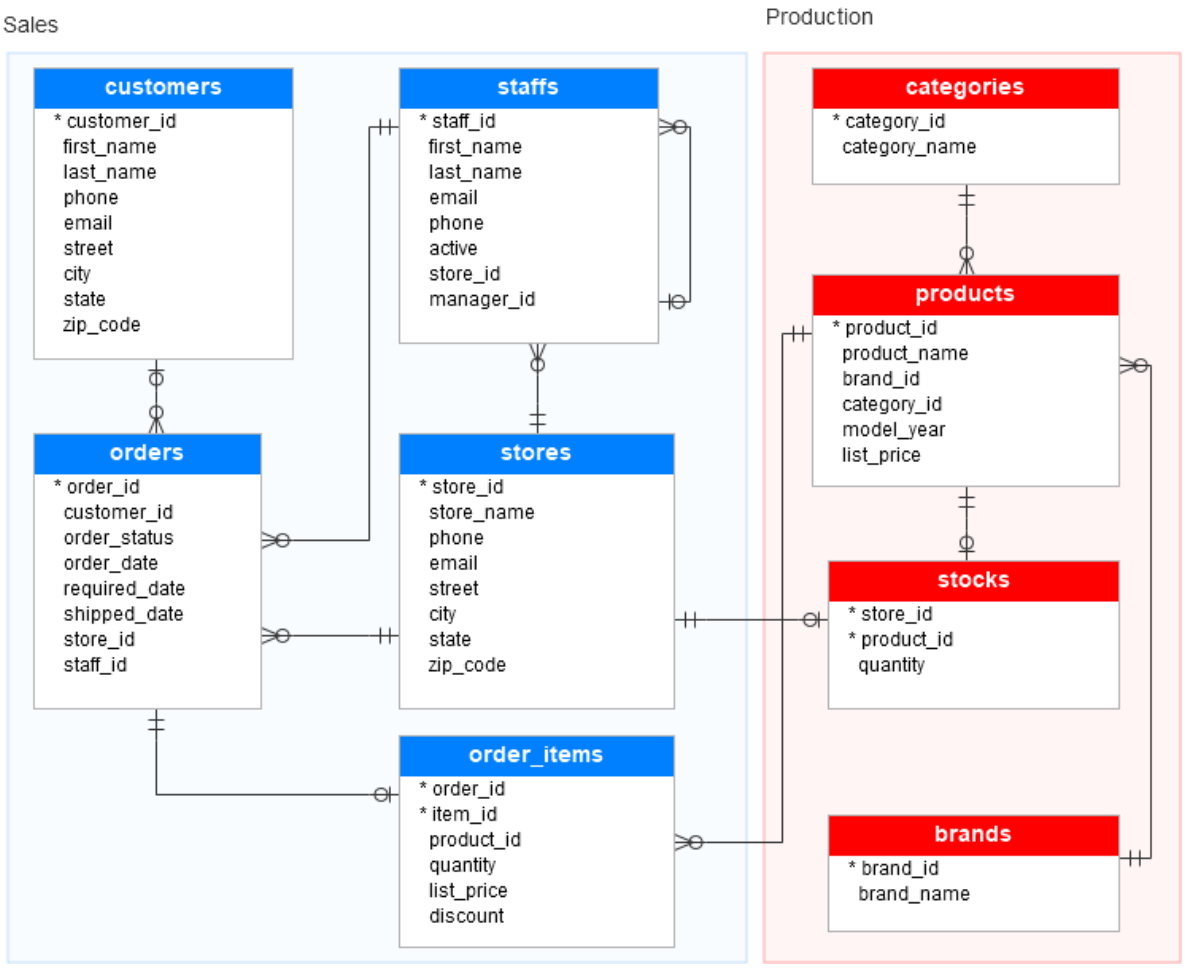
- El sistema debe permitir navegar la estructura del catálogo de productos
- El sistema debe permitir visualizar el inventario de productos por tienda
- El sistema debe permitir el cálculo de descuentos y promociones al momento de ingresar una orden

A la fecha, ha transcurrido gran parte del proyecto y se tiene el siguiente avance:

- Ya se cuenta con un prototipo funcional del aplicativo
- Existe un modelo de datos diseñado
- Existe una base de datos con datos de prueba
- Existe una aplicación web desarrollada con Spring Framework que desarrolla algunas de las funcionalidades requeridas

## Modelo de Datos

A continuación, se presenta el modelo de datos diseñado por el arquitecto en conjunto con el analista funcional.




Como se puede observar, el modelo tiene un área de trabajo con entidades relacionadas con el modelo de producción (catálogo e inventario) y otro grupo de entidades relacionadas con las ventas y órdenes.

La tabla *products* corresponde al maestro de productos de la compañía, en donde cada registro de producto tiene asociado una categoría de la tabla *categories* y una marca de la tabla *brands*. La tabla *stocks* contiene la cantidad del producto en existencia en cada una de las tiendas.

La tabla *orders* contiene la información de las órdenes que han sido cursadas, en donde el campo order_status señala el estado de la orden y los campos order_date, required_date y shipped_date almacenan las fechas de las órdenes en sus distintas etapas. Por otra parte, las órdenes están asociadas a la tabla *order_items* que almacena la información de los productos, cantidades, precios y descuentos que se consignaron al momento de cursar la órden.

Cada orden, además, está asociada a un cliente (tabla customers) en donde se almacena la información general de cada cliente tal como nombre, dirección, teléfono, etc. Asimismo, cada orden está asociada a una tienda (tabla *stores*) y a un vendedor (tabla staff).

El maestro de tiendas (*stores*) contiene información de cada sucursal. Cada sucursal, a su vez, maneja un grupo de vendedores y managers los cuales se encuentran almacenados en la tabla *staffs*. El maestro de empleados (tabla *staff*) almacena una relación jerárquica entre los empleados a través del campo *manager*_id.

## Requerimientos a Desarrollar

El jefe de proyectos, quien lleva un control meticuloso de las actividades del proyecto, le ha solicitado a Usted que realice las siguientes tareas:

1. Realizar consultas a la base de datos
2. Construir un algoritmo de cálculo de stock de seguridad
3. Construir una unidad de pruebas para verificar los algoritmos de cálculo de stock de seguridad
4. Crear un Monitor de Inventario
5. Crear una API REST que disponibilice la información del Monitor de Inventario

A continuación, se especifica con mayor detalle cada uno de los requerimientos:

### 1. Realizar consultas a la base de datos

El Chief Sales Manager ha solicitado algunos reportes de información, razón por la cual el jefe del proyecto le ha encargado a usted que realice algunas consultas en la base de datos, para la extracción de cierta información necesaria para el negocio, por mientras se termina el desarrollo de la aplicación. **Para esto, cree un package en su proyecto Java con nombre "consultas", cree un archivo por cada una de ellas, identificando claramente de qué consulta se trata** (ejm: Consulta-A.sql, Consulta-B.sql, …etc).

a) Se requiere un listado de productos con sus precios, de aquellos productos cuyo modelo es 2016, ordenado alfabéticamente por nombre. El reporte debe tener la siguiente forma:

| product_id | product_name ▲ 1 | model_year | list_price |
|---|---|---|---|
| 21 | Electra Cruiser 1 (24-Inch) - 2016 | 2016 | 269.99 |
| 13 | Electra Cruiser 1 (24-Inch) - 2016 | 2016 | 269.99 |
| 22 | Electra Girl's Hawaii 1 (16-inch) - 2015/2016 | 2016 | 269.99 |
| 14 | Electra Girl's Hawaii 1 (16-inch) - 2015/2016 | 2016 | 269.99 |
| 23 | Electra Girl's Hawaii 1 (20-inch) - 2015/2016 | 2016 | 299.99 |
| 15 | Electra Moto 1 - 2016 | 2016 | 529.99 |
| 24 | Electra Townie Original 21D - 2016 | 2016 | 549.99 |

b) Se requiere un inventario de productos para mujeres (aquellos que dice Ladies en el nombre) ordenado por precio descendente. El reporte debe tener la siguiente forma:

| product_id | product_name | model_year | list_price ▼ 1 |
|---|---|---|---|
| 199 | Electra Townie Commute Go! Ladies' - 2018 | 2018 | 2999.99 |
| 252 | Electra Townie Commute Go! Ladies' - 2018 | 2018 | 2999.99 |
| 195 | Electra Townie Go! 8i Ladies' - 2018 | 2018 | 2599.99 |
| 253 | Electra Townie Go! 8i Ladies' - 2018 | 2018 | 2599.99 |
| 259 | Electra Amsterdam Royal 8i Ladies - 2018 | 2018 | 1199.99 |
| 81 | Electra Amsterdam Fashion 7i Ladies' - 2017 | 2017 | 1099.99 |

c) Se requiere un reporte con la cantidad de productos de cada categoría, ordenado de mayor a menor cantidad. El reporte debe tener la siguiente forma:

| category_id | category_name | count(*) ▼ 1 |
|---|---|---|
| 3 | Cruisers Bicycles | 78 |
| 6 | Mountain Bikes | 60 |
| 7 | Road Bikes | 60 |
| 1 | Children Bicycles | 59 |
| 2 | Comfort Bicycles | 30 |
| 5 | Electric Bikes | 24 |
| 4 | Cyclocross Bicycles | 10 |

d) Se requiere un reporte con la cantidad de inventario de productos por marca, ordenado descendentemente. El reporte debe tener la siguiente forma:

| brand_id | brand_name | sum(st.quantity) |
|---|---|---|
| 3 | Heller | 3378 |
| 6 | Strider | 2654 |
| 1 | Electra | 2608 |
| 7 | Sun Bicycles | 2091 |
| 2 | Haro | 1258 |
| 5 | Ritchey | 1108 |
| 4 | Pure Cycles | 414 |

e) Se requiere un inventario para la tienda Santa Cruz Bike de los productos que tienen en existencia en la categoría Electric Bikes. El reporte debe tener la siguiente manera:

| product_id | product_name ▲ 1 | quantity |
|---|---|---|
| 191 | Electra Loft Go! 8i - 2018 | 21 |
| 198 | Electra Townie Commute Go! - 2018 | 1 |
| 199 | Electra Townie Commute Go! Ladies' - 2018 | 2 |
| 192 | Electra Townie Go! 8i - 2017/2018 | 20 |
| 195 | Electra Townie Go! 8i Ladies' - 2018 | 1 |
| 60 | Sun Bicycles ElectroLite - 2017 | 19 |
| 9 | Trek Conduit+ - 2016 | 11 |
| 185 | Trek Conduit+ - 2018 | 3 |
| 186 | Trek CrossRip+ - 2018 | 1 |
| 190 | Trek Dual Sport+ - 2018 | 21 |
| 193 | Trek Lift+ - 2018 | 30 |

## 2. Construir un algoritmo de cálculo del stock de seguridad

Un objetivo importante de este proyecto es solucionar el problema actual en la cual los vendedores muchas veces venden productos que no se encuentran en existencia, a pesar que el sistema le indica que quedan algunas unidades. Esto sucede principalmente porque en las tiendas a veces hay

cambios de productos que no se registran en el sistema, por lo tanto, el valor informado no siempre es el real.

Para darle solución al problema anterior, se define el concepto de stock de seguridad. El stock de seguridad es la cantidad mínima de inventario que puede tener un producto para realizar una venta. Es decir, si un producto actualmente registra 10 unidades en el sistema, y se define un stock de seguridad de 2 unidades, sólo se podrán generar órdenes por 8 unidades y el sistema no permitirá vender las últimas dos unidades.

Se le solicita que desarrolle un algoritmo cálculo del stock de seguridad con distintas lógicas de cálculo. Posteriormente, en la medida que se aprecien buenos resultados, se irán extendiendo e incorporando nuevas lógicas y reglas de negocio.

Construya dos algoritmos de cálculo de stock de seguridad, uno que realice un **cálculo simple** y otro que realice un **cálculo complejo**.

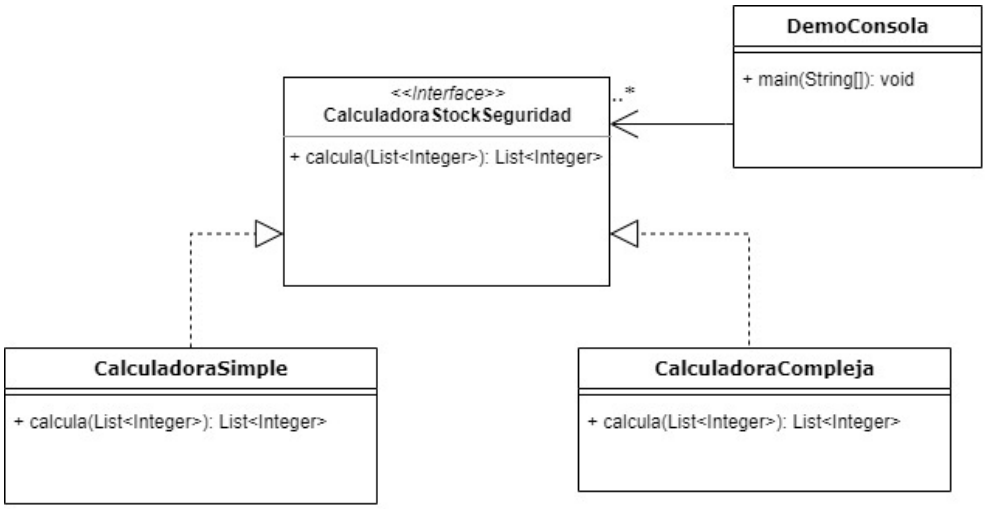
Las reglas del algoritmo de **cálculo simple** son las siguientes:

- El stock de seguridad de un producto siempre va a tener un valor fijo de 2 unidades

Las reglas del algoritmo de **cálculo complejo** son las siguientes:

- El stock de seguridad de un producto va a ser por defecto de 2 unidades, no obstante,
- Si el producto tiene más de 20 unidades entonces al stock de seguridad se le deberá agregar el 10% del stock del producto con el resultado redondeado. Por ejemplo, si un producto tiene 30 unidades, su stock de seguridad sería 2 + 30*0.1 = 5 unidades
- Si un producto tiene stock mayor o igual a 10 y menor a 20, entonces el cálculo será el stock de seguridad inicial más el 5% del stock del producto con resultado redondeado. Por ejemplo, si un producto tiene 12 unidades, su stock de seguridad sería 2 + 12*0.05 = 3 unidades

Ambos algoritmos deben recibir un listado de valores enteros correspondiente a los stocks a verificar, y retornar un listado de enteros con el valor del stock de seguridad. El arquitecto del proyecto le sugiere que realice un diseño de clases similar al del siguiente diagrama:




Para hacer una demostración de los algoritmos, cree una aplicación de consola que genere 5 valores de stock aleatorios entre 0 y 50, y que posteriormente realice el cálculo de los stocks de seguridad utilizando las dos implementaciones creadas anteriormente (algoritmo simple y complejo). A continuación, se presenta un ejemplo de la ejecución:

```
----------------------------------------
Demostración Calculo de Stock de Seguridad
----------------------------------------

Tomando 5 valores de stock aleatorios...
7 26 39 12 44

Stocks de Seguridad con Algoritmo Simple: 2 2 2 2 2
Stocks de Seguridad con Algoritmo Complejo: 2 5 6 3 7
```

Considere hacer un diseño de clases polimórfico, mediante interfaces, en donde deberá crear una interfaz y dos clases concretas que implementen dicha interfaz. Una de ellas que implemente el algoritmo de cálculo simple y otra que implemente el algoritmo de cálculo complejo. Asimismo, una clase que sea la que ejecuta la aplicación de consola y genera el listado de stock con valores aleatorios para entregárselo a cada uno de los algoritmos.

Genere dentro de su proyecto en eclipse un package con nombre representativo que tenga las clases mencionadas.

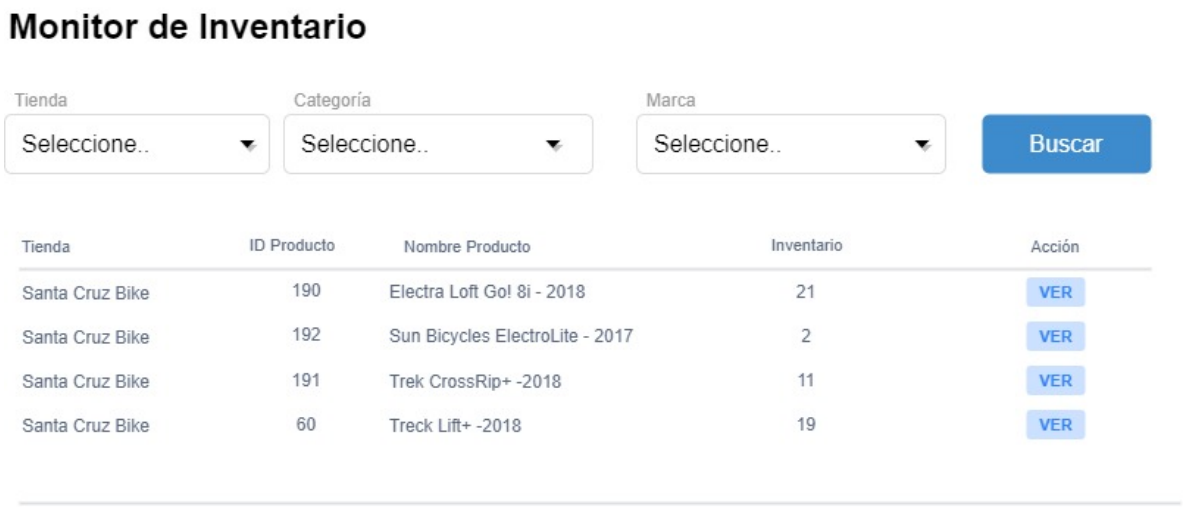
## 3. Construir una unidad de pruebas para verificar el correcto cálculo de los descuentos

Construya una clase de pruebas en Java que permita verificar el correcto funcionamiento del algoritmo de “cálculo complejo” del stock de seguridad, considerando al menos los siguientes tests:

- Tests que considere casos normales, con distinta cantidad y valores de stock.
- Tests que considere condiciones de borde, por ejemplo, qué pasa cuando viene un stock cero, u otra condición de excepción.

## 4. Crear Monitor de Ordenes

Se requiere crear una página web dinámica que despliegue el listado de órdenes, tal como se detalla en la siguiente imagen mock-up.




Se pide:

a) Desplegar el listado de tiendas en el primer combobox, ordenado alfabéticamente, con valores que provengan de la base de datos
b) Desplegar el listado de categorías en el segundo combobox, ordenado alfabéticamente, con valores que provengan de la base de datos
c) Desplegar el listado de las categorías en el tercer combobox, ordenado alfabéticamente, con valores que provengan de la base de datos
d) Desplegar el lisado de los productos y su inventario (de acuerdo a mock) de acuerdo a los filtros seleccionados (tienda, categoría, marca), a partir de la información de la base de datos.

Para realizar el requerimiento, el arquitecto le señala lo siguiente:

- Utilizar jsp y taglibs jstl para el despliegue de la vista (u otra tecnología de vista)
- Utilizar bootstrap para los elementos

- Que los elementos se ajusten a distintos tamaños de pantalla

## 5. Crear una API REST que disponibilice el listado de productos

Disponibilice un servicio REST que permita obtener la misma información del Monitor de Inventario. Recuerde que el servicio podría recibir como la tienda, categoría y marca.