Ejercicio práctico Java BackEnd Developer

Indicaciones Generales

- El ejercicio debe ser cargado a un repositorio Git público, la dirección del repositorio debe ser enviado por correo a las direcciones indicadas.
- El nombre del proyecto debe poseer como prefijo el nombre y apellido del candidato (EJ: juanPerez_inventarios)
- Lea detenidamente lo solicitado en el ejercicio, recuerde que la solución de cada requerimiento será tomada en cuenta para la evaluación.
- Elabore las pruebas unitarias y de integración que considere necesarias, el ejercicio debe contener mínimo 4 pruebas para que se sea aceptado.
- Para los servicios Mock solicitados, se puede hacer uso de la página: https://mocki.io/fake-json-api
- Puede utilizar la base de datos que desee.
- Posterior a la entrega del ejercicio se agendará una entrevista técnica donde el candidato deberá sustentar la solución planteada, se pedirá realizar pequeños cambios en la aplicación y se realizarán preguntar técnicas relacionadas con la tecnología utilizada.

Aplicación SpringBoot

- La aplicación debe manejar las entidades: Tienda, Producto y Cliente, en caso de requerir mas entidades puede incluirlas.

Producto

- El catálogo de productos debe ser cargado utilizando un servicio Mock y posteriormente guardarlos en la base de datos al levantar la aplicación, el servicio Mock (GET) debe responder lo siguiente:

```
{
  prods: [
    {
      "id": 1,
      "cod": "prod-1",
      "name": "prod-name-1",
      “price”: 5.5,
      "stock": 10
    },
    {
      "id": 2,
      "cod": "prod-2",
      "name": "prod-name-2",
      “price”: 6,
```

```
                "stock": 5
               },
               {
                 "id": 3,
                 "cod": "prod-3",
                 "name": "prod-name-3",
                 “price”: 7.5,
                 "stock": 15
               },
               {
                 "id": 4,
                 "cod": "prod-4",
                 "name": "prod-name-4",
                 “price”: 2.5,
                 "stock": 20
               },
               {
                 "id": 5,
                 "cod": "prod-5",
                 "name": "prod-name-5",
                 “price”: 9.5,
                 "stock": 25
               },
               {
                 "id": 6,
                 "cod": "prod-6",
                 "name": "prod-name-6",
                 “price”: 1.5,
                 "stock": 0
               }
             ]
}
```


- Cree un Endpoint que retorne el código y nombre de todos los productos.
- Cree un Endpoint para actualizar el stock de un determinado producto, considerar que el stock a actualizar no puede ser menor o igual a cero.

Tienda

- Tienda debe poseer la siguiente información: nombre, código (si considera necesario puede añadir mas atributos).
- Las Tiendas deben ser cargadas directamente a base de datos mediante un script (inserte 4 tiendas), esta carga debe ser realizada desde la misma aplicación springBoot.
- Cada Tienda puede vender uno o varios productos, es decir, puede ser que una tienda venda únicamente un producto mientras que otra puede vender todos (cree los servicios necesarios para realizar esta asignación).
- El Stock de productos de cada Tienda se maneja de forma global, por ejemplo, el stock del producto “prod-1” debe ser manejado por todas las tiendas.

```
{
  "id": 1,
  "cod": "prod-1",
  "name": "prod-name-1",
  "stock": 10
},
```


Cliente

- Cliente debe poseer la siguiente información: nombre, identificación (si considera necesario puede añadir mas atributos).
- Se debe realizar un CRUD de clientes (cree los servicios necesarios para realizar este requerimiento), considerar que el nombre e identificación del cliente son campos requerido.

Realizar Pedidos

- Un Cliente puede realizar pedidos de uno/varios productos a una/varias tiendas al mismo tiempo (cree el/los servicios necesarios para cumplir con este requerimiento).
- La aplicación debe controlar la existencia: Cliente, Tienda, Productos al realizar los pedidos.
- Si un Cliente realiza el pedido de un producto, se debe restar la cantidad solicitada del stock general.
- Si un cliente realiza el pedido de un producto y no se encuentra en stock, se debe realizar lo siguiente:
    1) Si el stock faltante es por mas de 10 unidades, se rechaza la transacción indicada el error: “Unidades no disponibles (> 10)”
    2) Si el stock faltante es de: (>5 unidades y <=10 unidades), se debe solicitar stock extra, para esto se simulará dicha petición haciendo uso de un servicio Mock (GET) el cual debe responder lo siguiente:


```
{
"code": "code-prod",
"name":"prod-name",
"stock":10
}
```


El valor del stock que responde el Mock ("stock":10) se lo debe sumar al stock general del producto y la transacción termina de manera satisfactoria.

3) Si el stock faltante está entre 0 y 5 unidades, se acepta la transacción, es decir, el servicio responde de manera inmediata indicando que la transacción se realizó correctamente, **pero de manera asincrónica** se debe invocar al servicio Mock (GET) el cual debe responder lo siguiente:

```
{
"code": "code-prod",
"name":"prod-name",
"stock":5
}
```

El valor del stock que responde el servicio Mock ("stock":5) se lo debe sumar al valor al stock general del producto.

Registro de transacciones

- Por cada tienda se debe guardar el registro de las transacciones realizadas, donde se indique: cliente, producto, cantidad, fecha y hora. (aplique la solución mas genérica posible).

Reportes

- Se debe crear los Endpoints necesarios para obtener la siguiente información:
  - Número de transacciones realizadas agrupadas por Tienda y fecha.
  - Monto Vendido por Tienda y Producto.
- Genere y descargue un reporte de tipo CSV con la información de las transacciones realizadas por un cliente en un rango determinado de fechas, se debe validar la existencia del cliente, que el rango de fechas sea el adecuado.