UNIVERSIDAD TECNOLÓGICA NACIONAL FACULTAD REGIONAL MAR DEL PLATA

PROGRAMACION III – COMISION 6 Y 7

DOCENTE: DANIEL DIAZ

1er Parcial – 21/4

PROGRAMACION III

Sistema de Gestión de Venta de Bebidas en una Distribuidora

Una pequeña **distribuidora de bebidas** necesita un sistema para gestionar la venta de sus productos a diferentes kioscos y almacenes. El dueño quiere mantener un registro digital de:

1. Las **bebidas** que tiene en stock.
2. Los **clientes** que le compran (generalmente kioscos, almacenes o supermercados).
3. Los **pedidos** realizados por los clientes.

Cada pedido puede involucrar una o varias bebidas. El sistema debe permitir registrar, consultar, y actualizar pedidos.

ESTRUCTURA DE LA BASE DE DATOS:

```sql
-- Tabla: Bebida
CREATE TABLE Bebida (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    tipo ENUM('GASEOSA', 'AGUA', 'JUGO', 'CERVEZA') NOT NULL,
    precioUnitario DECIMAL(10,2) NOT NULL CHECK (precioUnitario > 0),
    stock INT NOT NULL CHECK (stock >= 0)
);

-- Tabla: Cliente
CREATE TABLE Cliente (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    localidad VARCHAR(100) NOT NULL,
    tipoCliente ENUM('KIOSCO', 'ALMACEN', 'SUPERMERCADO') NOT NULL
);

-- Tabla: Pedido
CREATE TABLE Pedido (
    id INT AUTO_INCREMENT PRIMARY KEY,
    idCliente INT NOT NULL,
    idBebida INT NOT NULL,
    cantidad INT NOT NULL CHECK (cantidad > 0),
    estado ENUM('PENDIENTE', 'PREPARADO', 'ENTREGADO') NOT NULL DEFAULT 'PENDIENTE',
    FOREIGN KEY (idCliente) REFERENCES Cliente(id),
    FOREIGN KEY (idBebida) REFERENCES Bebida(id)
);
```

UNIVERSIDAD TECNOLÓGICA NACIONAL FACULTAD REGIONAL MAR DEL PLATA

PROGRAMACION III – COMISION 6 Y 7

DOCENTE: DANIEL DIAZ

Se provee una clase **DBConnection** que establece conexión a distribuidora_bebidas utilizando MySQL.

¡IMPORTANTE!

---DEPENDIENDO LA COMPUTADORA EN LA QUE ESTEN REALIZANDO EL PARCIAL (COMPUTADORA PERSONAL O DEL LABORATORIO) DEBERAN MODIFICAR LAS CREDENCIALES CORRESPONDIENTES A LA CONEXIÓN A LA BASE DE DATOS---

Credenciales del servidor de la UTN:

Connection Name: utnbdd

IP: 172.16.1.64

User: bdd1

Password: bdd1

¡NO OLVIDAR IMPORTAR LA LIBRERÍA/DRIVER DE MYSQL AL PROYECTO!

REQUISITOS FUNCIONALES:

1. **RF01 - CRUD de Bebidas:**
   El sistema debe permitir realizar operaciones de alta, baja, modificación y consulta de bebidas, incluyendo nombre, tipo, precio unitario y stock disponible.

2. **RF02 - Filtrar bebidas por tipo:**
   El usuario debe poder listar todas las bebidas de un tipo específico (por ejemplo, todas las de tipo "CERVEZA").

3. **RF03 - Filtrar bebidas con bajo stock:**
   El sistema debe permitir mostrar todas las bebidas cuyo stock sea menor a un valor configurable (por ejemplo, stock < 30).

4. **RF04 - Buscar clientes por localidad:**
   El sistema debe permitir obtener todos los clientes registrados en una localidad específica.

5. **RF05 - Listar pedidos por cliente:**
   El usuario debe poder consultar todos los pedidos realizados por un cliente determinado, filtrando por su id. Mostrar los detalles del cliente encontrado y la lista de sus pedidos realizados.

6. **RF06 - Cambiar el estado de un pedido.**
   Verificar primeramente que el pedido a modificar exista y que la transición entre estados sea válida, es decir, **Pendiente --> Preparado** o de **Preparado -->Entregado**

7. **RF07 - Validar pedidos antes de registrarlos (Implementar el registro de un pedido):**
   El sistema debe lanzar una excepción personalizada (PedidoInvalidoException) si:
   - La cantidad de bebidas indicada por el usuario es negativa.
   - El cliente que realiza el pedido no existe.
   - La bebida que desea pedir no existe.
   - Un pedido intenta registrar una cantidad superior al stock disponible de la bebida.

   Si el pedido es exitoso, se deberá actualizar el stock de la bebida solicitada.
   El Estado del pedido a la hora de registrarlo deberá ser PENDIENTE por defecto.

8. **RF08 – Mostrar la cantidad de pedidos agrupados por estado.**

REQUISITOS NO FUNCIONALES

- Los repositorios deben implementar el patrón Singleton para acceder a la instancia única de la conexion.
- Las interfaces deben estar separadas de su implementación.
- Todo filtrado, agrupamiento o transformación debe resolverse usando **Programación Funcional** y no consultas SQL.
- Todos los datos se deberán recorrer y manipular con el uso de Programación funcional y no mediante bucles tradicionales.
- El código debe ser limpio, organizado y seguir buenas prácticas de diseño.
- Manejar adecuadamente la excepción NullPointer mediante el uso de Optionals en las clases Service.

TABLA DE PUNTAJES	
RF01	**15**
RF02	**5**
RF03	**5**
RF04	**5**
RF05	**15**
RF06	**15**
RF07	**25**
RF08	**15**