**Evaluación Técnica de Programación**

| AUTOR | VERSIÓN |
|---|---|
| Alberto Salazar Trejo | Versión 24.4 |

**PUNTUACIÓN:**

La organización de la evaluación y la distribución de puntos en los diferentes ejercicios propuestos se resume en la siguiente tabla:

| EJERCICIOS | PUNTOS |
|---|---|
| 1. Conocimientos SQL | 10 |
| 2. Ejercicio práctico: BD | 15 |
| 3. Ejercicio práctico: Desarrollo | 50 |
| 4. Documentación | 25 |
| **TOTAL** | **100** |

**NOTA:** A partir de la recepción de la evaluación contará con 72 horas para enviarla.

## 1. CONOCIMIENTOS SQL

**1.1)** Describe el funcionamiento general de la sentencia **JOIN**.
R= La sentencia JOIN funciona para obtener datos de dos o mas tablas por medio de unas condiciones de campos en comun

**1.2)** ¿Cuáles son los tipos de **JOIN** y cuál es el funcionamiento de los mismos?
R=**LEFT JOIN**: Se utiliza para unir dos tablas, devolviendo todos los registros de la tabla izquierda o principal y las coincidencias de la tabla derecha. En caso de que no haya coincidencias devolvera un NULL como respuesta
**RIGTH JOIN:** Se utiliza para unir dos tablas pero ahora devolvera todos los registros de la tabla secundaria o derecha y solo las coincidencias de la tabla izquierda. Llenando las no coincidencias con NULL
**INNER JOIN:** Une dos o mas tablas, devolviendo solo aquellas en las que hay coinicdencias en las columnas especificadas en la condición de unión.
**FULL JOIN:** Une dos tablas y devuelve todos los registros de ambas, tanto si existen coincidencias como si no. Como en el caso de LEFT y RIGTH regresa un NULL cuando no hay coincidencias
**CROSS JOIN:** Se usa para combianr cada fila de una tabla con todas las filas de otra tabla, teniendo como resultado todas las combinaciones posibles entre ambas. Ademas de que no requiere condición de union.

**1.3) ¿**Cuál el funcionamiento general de los **TRIGGER** y qué propósito tienen?

**R=** Es un bloque de sentencias sql que se ejecuta automaticamente cuando se realiza ya sea INSERT, UPDATE y DELETE dentro de la base de datos y el proposito es tener un mayor control del acciones dentro de la base de datos ya que se puede realizar auditorias, validación, seguridad, sincronización, entre otros.

**1.4)** ¿Qué es y para qué sirve un **STORED PROCEDURE**?

R= Es un bloque de sentencia preparado que puede guardarse con la finalidad de que pueda ser ejecutado una y otra vez, tiene la caracteristica de que se le puede pasar parametros. Sirve para optimizar la reutilización de sentencias, ademas de centralizar la lógica de negocio dentro de la bd.

Considerando que las siguientes tablas:

| productos | |
|---|---|
| PK | idProducto INT(6) |
| | nombre VARCHAR(40) |
| | precio DECIMAL(16,2) |

| ventas | |
|---|---|
| PK | idVenta INT(6) |
| FK1 | idProducto INT(6) |
| | cantidad INT(6) |

Tienen los siguientes datos insertados:

| idProducto | nombre | precio |
|---|---|---|
| 1 | LAPTOP | 3000.00 |
| 2 | PC | 4000.00 |
| 3 | MOUSE | 100.00 |
| 4 | TECLADO | 150.00 |
| 5 | MONITOR | 2000.00 |
| 6 | MICROFONO | 350.00 |
| 7 | AUDIFONOS | 450.00 |

| idVenta | idProducto | cantidad |
|---|---|---|
| 1 | 5 | 8 |
| 2 | 1 | 15 |
| 3 | 6 | 13 |
| 4 | 6 | 4 |
| 5 | 2 | 3 |
| 6 | 5 | 1 |
| 7 | 4 | 5 |
| 8 | 2 | 5 |
| 9 | 6 | 2 |
| 10 | 1 | 8 |

Hacer las consultas necesarias para:

**1.5)** Traer todos los productos que tengan una venta.

```
SELECT v.idProducto, p.nombre FROM ventas v INNER JOIN productos p ON p.idProducto = v.idProducto GROUP BY v.idProducto HAVING SUM(v.cantidad) = 1;
```

**1.6)** Traer todos los productos que tengan ventas y la cantidad total de productos vendidos.

```
SELECT v.idProducto, SUM(v.cantidad) AS total FROM ventas v INNER JOIN productos p ON p.idProducto = v.idProducto WHERE v.cantidad >= 1 GROUP BY v.idProducto;
```

**1.7)** Traer todos los productos (independientemente de si tienen ventas o no) y la suma total ($) vendida por producto.

```
SELECT p.idProducto, p.nombre, SUM(v.cantidad) AS cantidad,SUM(v.cantidad * p.precio) AS total FROM productos p LEFT JOIN ventas v ON v.idProducto = p.idProducto GROUP BY p.idProducto;
```

**ESCENARIO: Leer el siguiente caso detenidamente para trabajar los puntos 2, 3 y 4 de tu evaluación.**

Una empresa necesita un sistema para administrar su inventario en el almacén. Este sistema debe tener las siguientes características:

- Debe tener un **inicio de sesión**.

- Debe tener un **módulo** para ver el **inventario** de la empresa.

◦ Se debe poder agregar nuevos productos al inventario. Al agregar un producto, la cantidad inicial será 0.
◦ Se debe poder aumentar el inventario de los productos (**Entrada de productos**). Si intentas disminuir la cantidad de inventario actual, mostrará un mensaje de error.
◦ Se debe poder dar de baja un producto. La baja no elimina el registro, solo actualiza el estatus. Los productos dados de baja se pueden activar nuevamente. ◦ Se pueden ver los productos activos e inactivos.

- Debe haber un **módulo** para sacar(restar) inventario del almacén (**Salida de productos**).

◦ Solo se pueden ver los productos activos.
◦ No se puede sacar una cantidad mayor de un producto de la que está en inventario. Si se intenta hacer, mostrará un mensaje de error.

- Debe haber un de **módulo** con el **historial** de movimientos de "Entrada" y "Salida" productos.

◦ El listado de movimientos se debe poder filtrar por tipo de movimiento (entrada o salida).
◦ Cada movimiento debe registrar quién lo realizó.
◦ Cada movimiento debe tener la fecha y hora en que se realizó el movimiento.

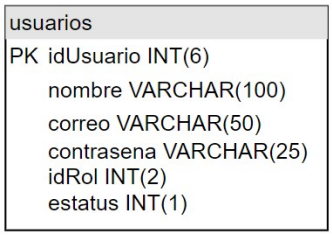
- El sistema tendrá 2 roles y los permisos de los mismos se describen a continuación:

| Permiso | Administrador | Almacenista |
|---|---|---|
| Ver módulo inventario | ✓ | ✓ |
| Agregar nuevos productos | ✓ | ✗ |
| Aumentar inventario | ✓ | ✗ |
| Dar de baja/reactivar un producto | ✓ | ✗ |
| Ver módulo para Salida de productos | ✗ | ✓ |
| Sacar inventario del almacén | ✗ | ✓ |
| Ver módulo del histórico | ✓ | ✗ |

**2. EJERCICIO PRÁCTICO: BD**

**2.1)** Crea un diagrama relacional de BD para el escenario descrito anteriormente.
**2.2)** Hacer el script para crear las tablas del punto anterior (en el punto 4.2 se especificará dónde debe anexarse dicho script).

**NOTA:** Actualmente solo se cuenta con la tabla de "usuarios". Falta considerar las tablas para saber identificar el tipo de rol, los productos y para el histórico.




## 3. EJERCICIO PRÁCTICO: DESARROLLO

**3.1)** Crear una aplicación **WEB** para el escenario que se planteó previamente.

- Preferiblemente utilizar el diseño **MVC** para su desarrollo.
- Preferiblemente utilizar **JAVA**.
- Utilizar **MySQL** o **SQLServer**.

## 4. DOCUMENTACIÓN:

**4.1)** Se necesita que el código se suba en un repositorio de **GITHUB** y esté público para que pueda ser descargado.

**4.2)** Los scripts generados para la evaluación deben encontrarse en una carpeta llamada **SCRIPTS**, dentro del repositorio de GITHUB.

**4.2)** Dentro del repositorio, debe haber un **README.md** donde se especifiquen los datos relevantes para el desarrollo:

- IDE utilizado.
- Versión del lenguaje de programación utilizado.
- DBMS utilizado y su versión.
- Lista de pasos para correr su aplicación.

**4.3)** Haga un **video** donde se muestren los siguientes escenarios dentro de su aplicación:
Mostrar que los usuarios con el rol de **Administrador** pueden:

- Iniciar de sesión.
- Visualizar el histórico y filtrarlo.
- Registrar productos nuevos.
- Dar de baja productos.
- Reactivar productos.
- Agregar existencias a los productos.
- No pueden acceder al módulo para salida de material.

Mostrar que los usuarios con el rol de **Almacenista** pueden:

- Iniciar de sesión.
- No pueden visualizar el histórico.
- Solo pueden visualizar el módulo de inventario.
- Puede acceder al módulo para salida de material y sacar material.

Mostrar los mensajes de error que se consideraron en el escenario planteado.

**NOTA**: Dicho video se puede compartir a través de Google Drive.