****

**Tecnicatura Universitaria en Programación**

**Base de Datos II**

**Equipo # 29**

**Integrantes:**

**Correa Gimenez, Leandro Sebastian - 23908**

**Jesus Esteban Ballico – 30884**

**Facundo Palomanes – 28799**

Explicación del Sistema

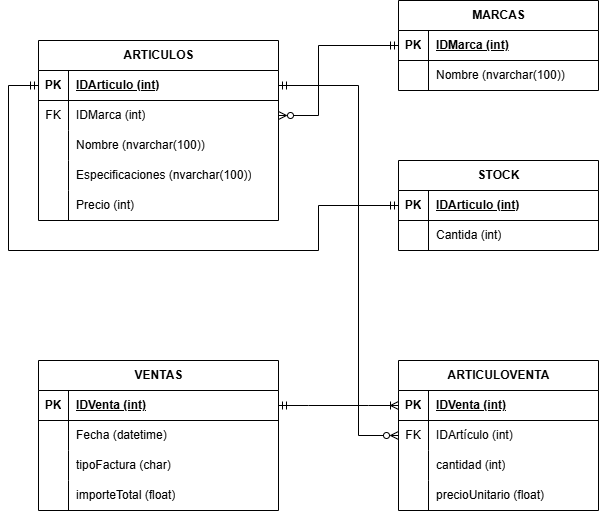
El sistema trata de un punto de venta de carácter general, orientado a pequeños y medianos comercios como kioskos, almacenes y locales de carácter general. El propósito del mismo es facilitar la administración del local, contando con diversas funcionalidades para registrar los movimientos del día a día, y en base a ellas, obtener métricas y reportes para llevar un control del modelo de negocio.

### Funcionalidades principales

* **Gestión del stock:** Permite realizar la carga de artículos al inventario disponible, gestionar su stock disponible del mismo para la venta, asignarle un nombre, marca, categoría, precio.
* **Registro de ventas:** Permite registrar una venta.
* **Reportes y métricas:** A través de las ventas registradas, permite obtener reportes claves para el negocio, como productos más vendidos, ingresos totales por día / semana / mes, reporte de stock actual.
* **Ordenes de compra:** En base al stock que va resultando, permite realizar ordenes de compras de los artículos que tengan un stock por debajo del umbral normal.

El sistema está construido con una base de datos relacional que almacena toda la información necesaria para gestionar estas funcionalidades. Se han implementado tres vistas, dos procedimientos almacenados y dos triggers.

## Diagrama de Entidad Relación



## Objetos de Base de Datos clave en el sistema

### Vista 1

Explicación vista 1

|  |
| --- |
| -- Codigo |

### Vista 2

La vista **VW\_MostrarInformacionVentas** nos devuelve un listado de todos los ítems de todas las ventas realizadas, para luego poder aplicar más filtros sobre el resultado.

La misma nos trae las siguientes columnas:

IDVenta, Fecha, Artículo, Cantidad, Precio Unitario, SubTotal, TipoFactura

La vista simplifica los JOINS, para que podamos filtrar detalles de una venta por su ID, y ver los artículos de esa venta, ver los artículos que se vendieron en un lapso de tiempo, sacar conclusiones de negocio mediante los SubTotales, etc.

|  |
| --- |
| CREATE VIEW VW\_MostrarInformacionVentas AS  SELECT  V.IDVenta, V.Fecha, A.Nombre as 'Artículo', AV.Cantidad, AV.PrecioUnitario, AV.Cantidad \* AV.PrecioUnitario AS 'SubTotal', V.TipoFactura  FROM ArticulosVenta AS AV  INNER JOIN Articulos AS A ON AV.IDArticulo = A.IDArticulo  INNER JOIN Ventas AS V ON AV.IDVenta = V.IDVenta; |

### Vista 3

Aquí se explica el propósito de la Vista 2 del sistema.

|  |
| --- |
| -- codigo |

### Procedimiento Almacenado 1

El procedimiento **SP\_AgregarArticulo** permite dar de alta un artículo nuevo al negocio, y establecerle un posible precio de venta, lo que significa que lo pone disponible para luego cargarle stock, si lo tuviera. Por defecto, establece stock en 0.

El mismo valida que no se inserte un precio negativo, y que previamente esté registrada la marca del producto en el sistema.

|  |
| --- |
| CREATE PROCEDURE SP\_AgregarArticulo(  @nombreArticulo nvarchar(100),  @idMarca int,  @precioArticulo float  )  AS  BEGIN TRANSACTION  BEGIN TRY  IF (@precioArticulo < 0) BEGIN  RAISERROR('No se puede insertar un precio de artículo negativo', 16, 1)  END  DECLARE @marcaID int  SET @marcaID = 0;  SELECT @marcaID = M.IDMarca FROM MARCAS AS M WHERE M.IDMarca = @idMarca;  IF (@marcaID = 0) BEGIN  RAISERROR('Número de ID de marca no existe en la base de datos', 16, 1)  END  INSERT INTO Articulos (Nombre, IDMarca, Precio) VALUES (@nombreArticulo, @idMarca, @precioArticulo);  INSERT INTO Stock VALUES (@@IDENTITY, 0);    COMMIT TRANSACTION;  END TRY  BEGIN CATCH  ROLLBACK TRANSACTION;  PRINT ERROR\_MESSAGE();  END CATCH |

### Procedimiento Almacenado 2

Aquí se explica el propósito del Procedimiento Almacenado 2 de la base de datos.

### 

### Trigger 1

Se utiliza el trigger TR\_SolicitudDiaLibre que permite registrar una nueva solicitud de día libre por parte de un empleado aplicando la siguiente regla de negocio que automatiza el rechazo de las mismas dadas las siguientes condiciones:

* Se rechaza automáticamente si el empleado solicita un día libre en el mismo mes en el que fue sancionado.
* Se rechaza automáticamente si el empleado solicita un día libre con fecha consecutiva a otro día libre solicitado previamente y que haya sido aprobado.

|  |
| --- |
| CREATE TRIGGER TR\_SolicitudDiaLibre ON SolicitudDeDiasLibres AFTER INSERT AS BEGIN  DECLARE @IDEmpleado INT, @FechaSolicitud DATE;   -- Obtener los datos de la nueva solicitud  SELECT @IDEmpleado = IDEmpleado, @FechaSolicitud = FechaSolicitud  FROM INSERTED;  -- Resto de la lógica del trigger para aplicar los rechazos automáticos a partir de las reglas de negocio.  END; |

### Trigger 2

Aquí se explica el propósito del Trigger 2 de la base de datos.

## Links a los recursos

**Script de creación de base de datos con datos**

<https://github.com/Salchy/TP-integrador-dbii>

**Video demo del sistema (hasta 25 minutos)**

<https://youtube.com/usuario/VideoDemo>