****

**Tecnicatura Universitaria en Programación**

**Laboratorio de Computación III**

**Equipo # 01**

**Integrantes:**

**Zurita, Maximiliano - 24291**

**Gianfranco, Pantanetti - Legajo**

**Fernandez, Ramiro – Legajo**

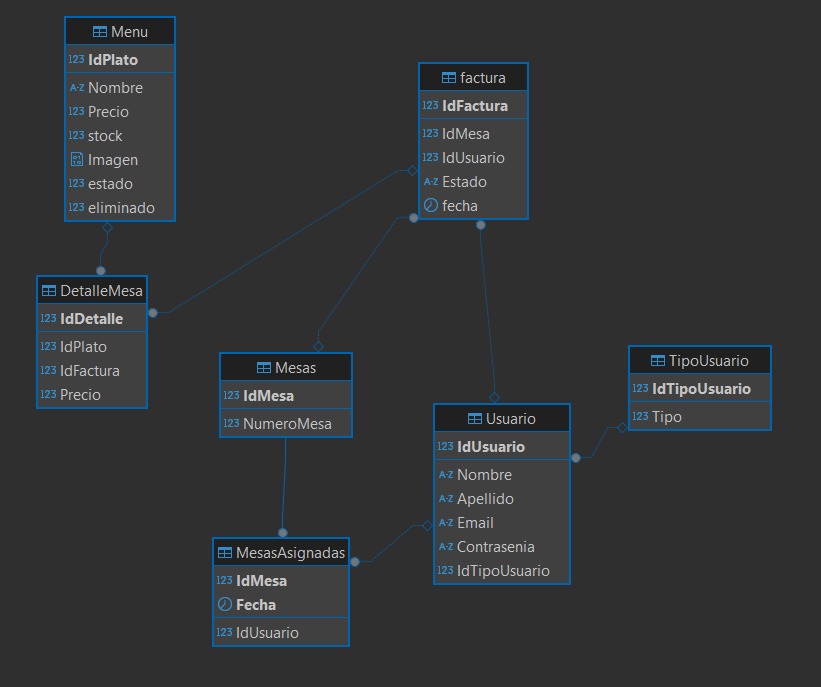
**Explicación del Sistema**

El sistema desarrollado permite gestionar un restaurante de forma eficiente, abarcando desde la administración de usuarios (mozos, clientes, administradores), la gestión de mesas y menús, hasta la facturación y asignación de mozos.  
  
Incluye control de permisos por tipo de usuario, generación de facturas detalladas, y un registro de las órdenes realizadas en cada mesa. Se implementan distintas funcionalidades orientadas a facilitar las tareas operativas del personal y mejorar la atención al cliente.

**Funcionalidades principales**

* **Gestión de usuarios:** Administración de distintos tipos de usuario (administrador, mozo, cliente), con sus datos personales y credenciales.
* **Gestión de mesas:** Alta, baja y modificación de las mesas del restaurante, permitiendo su asignación a clientes y mozos.
* **Gestión de menús:** Administración de productos disponibles en el restaurante, incluyendo su categoría, precio y estado.
* **Toma de pedidos y facturación:** Registro de ventas asociadas a mesas, mozos y clientes, con cálculo automático del total.
* **Historial de ventas:** Consulta detallada de ventas realizadas, con opción de filtrar por usuario, mozo o fecha.

**Diagrama de Entidad-Relación**



**Objetos de Base de Datos clave en el sistema**

**Vista 1: VW\_VentasPorMozo**

Muestra el total de ventas realizadas por cada mozo, agrupadas por fecha.

**CREATE VIEW VW\_VentasPorMozo AS  
SELECT   
 u.IdUsuario AS IdMozo,  
 u.Nombre + ' ' + u.Apellido AS Mozo,  
 CONVERT(date, v.Fecha\_venta) AS Fecha,  
 SUM(v.Suma\_total) AS TotalVendido  
FROM Ventas v  
JOIN Usuario u ON v.IdMozo = u.IdUsuario  
GROUP BY u.IdUsuario, u.Nombre, u.Apellido, CONVERT(date, v.Fecha\_venta);**

**Vista 2: VW\_ProductosMasVendidos**

Devuelve un listado con los productos más vendidos del sistema, ordenados por cantidad.

**CREATE VIEW VW\_ProductosMasVendidos AS  
SELECT   
 p.Nombre,  
 SUM(vp.Cantidad) AS TotalVendido  
FROM VentaProducto vp  
JOIN Producto p ON vp.IdProducto = p.IdProducto  
GROUP BY p.Nombre  
ORDER BY TotalVendido DESC;**

**Vista 3: VW\_ClientesFrecuentes**

Muestra los clientes con más cantidad de compras registradas.

**CREATE VIEW VW\_ClientesFrecuentes AS  
SELECT   
 u.IdUsuario AS IdCliente,  
 u.Nombre + ' ' + u.Apellido AS Cliente,  
 COUNT(v.ID) AS CantidadCompras  
FROM Ventas v  
JOIN Usuario u ON v.IdUsuario = u.IdUsuario  
WHERE u.IdTipoUsuario = 2  
GROUP BY u.IdUsuario, u.Nombre, u.Apellido  
ORDER BY CantidadCompras DESC; CREATE VIEW VW\_ClientesFrecuentes AS  
SELECT   
 u.IdUsuario AS IdCliente,  
 u.Nombre + ' ' + u.Apellido AS Cliente,  
 COUNT(v.ID) AS CantidadCompras  
FROM Ventas v  
JOIN Usuario u ON v.IdUsuario = u.IdUsuario  
WHERE u.IdTipoUsuario = 2  
GROUP BY u.IdUsuario, u.Nombre, u.Apellido  
ORDER BY CantidadCompras DESC;**

**Procedimiento Almacenado 1: SP\_RegistrarVenta**

Inserta una nueva venta con sus productos asociados y actualiza el total.

**CREATE PROCEDURE SP\_RegistrarVenta  
 @IdUsuario INT,  
 @IdMozo INT,  
 @IdMesa INT,  
 @Detalle XML  
AS  
BEGIN  
 BEGIN TRY  
 DECLARE @IdVenta INT;  
 INSERT INTO Ventas (IdUsuario, IdMozo, IdMesa, Fecha\_venta, Suma\_total)  
 VALUES (@IdUsuario, @IdMozo, @IdMesa, GETDATE(), 0);  
   
 SET @IdVenta = SCOPE\_IDENTITY();  
 DECLARE @Total DECIMAL(10,2) = 0;  
 -- Procesar detalle en XML y actualizar total  
 UPDATE Ventas SET Suma\_total = @Total WHERE ID = @IdVenta;  
 END TRY  
 BEGIN CATCH  
 PRINT ERROR\_MESSAGE();  
 END CATCH  
END;**

**Procedimiento Almacenado 2: SP\_AsignarMesaAMozo**

Asigna una mesa a un mozo si está disponible.

**CREATE PROCEDURE SP\_AsignarMesaAMozo  
 @IdMesa INT,  
 @IdMozo INT  
AS  
BEGIN  
 IF EXISTS (SELECT 1 FROM Mesa WHERE IdMesa = @IdMesa AND Estado = 'Libre')  
 BEGIN  
 UPDATE Mesa SET Estado = 'Ocupada', IdMozo = @IdMozo WHERE IdMesa = @IdMesa;  
 END  
 ELSE  
 BEGIN  
 RAISERROR('La mesa no está disponible.', 16, 1);  
 END  
END;**

**Trigger 1: TR\_BloquearVentaSinCliente**

Evita que se inserte una venta para usuarios que no sean clientes.

**CREATE TRIGGER TR\_BloquearVentaSinCliente  
ON Ventas  
INSTEAD OF INSERT  
AS  
BEGIN  
 IF EXISTS (  
 SELECT 1 FROM INSERTED i  
 JOIN Usuario u ON i.IdUsuario = u.IdUsuario  
 WHERE u.IdTipoUsuario <> 2  
 )  
 BEGIN  
 RAISERROR('Solo los clientes pueden realizar compras.', 16, 1);  
 ROLLBACK;  
 END  
 ELSE  
 BEGIN  
 INSERT INTO Ventas SELECT \* FROM INSERTED;  
 END  
END;**

**Trigger 2: TR\_ActualizarEstadoMesa**

Al registrar una venta, cambia automáticamente el estado de la mesa a “Libre”.

**CREATE TRIGGER TR\_ActualizarEstadoMesa  
ON Ventas  
AFTER INSERT  
AS  
BEGIN  
 UPDATE m  
 SET m.Estado = 'Libre'  
 FROM Mesa m  
 JOIN INSERTED i ON m.IdMesa = i.IdMesa;  
END;**

**Links a los recursos**