

**LABORATORIO DE COMPUTACIÓN III**

**TRABAJO INTEGRADOR FINAL (TIF)**

**PETS**

**Integrantes:**

-Máximo Tomás Canedo

-Jean Pierre Esquen

-María Olivia Hanczyc

-Ezequiel Alejandro Martínez

-Javier Andrés Torales

**Cuerpo docente:**

-Ing. Ariel Hernán Herrera

-TSP. Diego Ariel Español

Tecnicatura Universitaria en Programación – 2º año – 1º cuatrimestre 2023

ÍNDICE

[INTRODUCCIÓN 3](#_Toc139635336)

[D.E.R. 4](#_Toc139635337)

[CÓDIGO 5](#_Toc139635338)

[JEAN PIERRE ESQUEN 6](#_Toc139635339)

[JAVIER ANDRÉS TORALES 9](#_Toc139635340)

[MÁXIMO TOMÁS CANEDO 12](#_Toc139635341)

[EZEQUIEL ALEJANDRO MARTÍNEZ 16](#_Toc139635342)

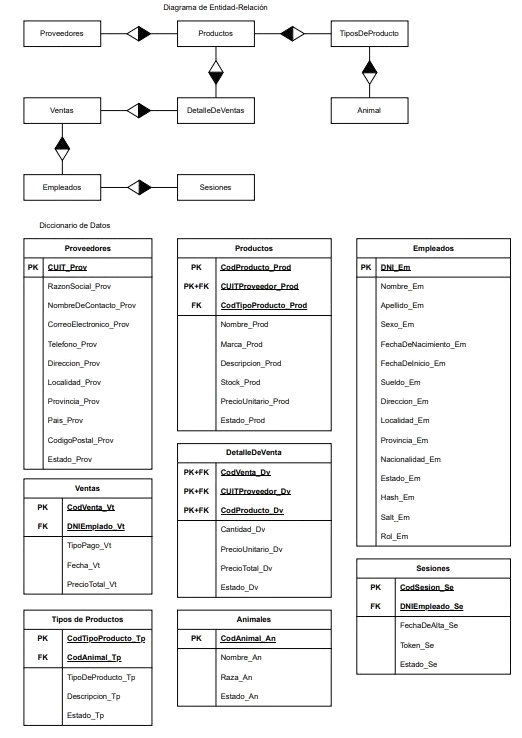
[MARÍA OLIVIA HANCZYC 20](#_Toc139635343)

[SCRIPT 23](#_Toc139635344)

# INTRODUCCIÓN

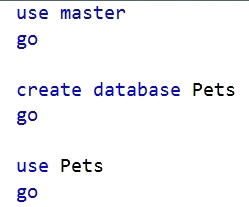
Nuestro proyecto consiste en una aplicación web para gestionar las ventas de una tienda de artículos para animales, llamada Pets. Para ello, creamos una base de datos, que nos posibilita almacenar: la información sobre cada venta y detalle de la venta, los productos y tipos de productos, sobre los animales a los cuales están destinados nuestros productos, los datos de los proveedores y de los empleados (incluyendo aquellos que sean administradores), y las sesiones de cada empleado en la aplicación.

# D.E.R.



# CÓDIGO

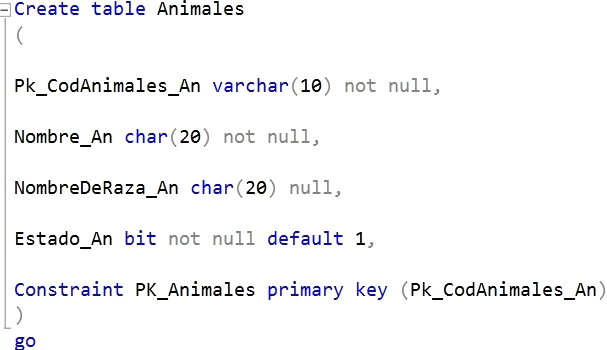
Aquí mostraremos primero una captura de pantalla de la creación de la base de datos Pets.



La creación de tablas, procedimientos almacenados, triggers y carga de registros en cada tabla también se mostrarán en capturas de pantalla, y ordenadas según el integrante que las haya realizado.

## JEAN PIERRE ESQUEN

**-Tablas:**



En esta tabla llamada Animales se guardan todos los animales y alguna raza de animales. Están guardados con un código de animal como clave primaria. La función de esta tabla es tener información del animal.

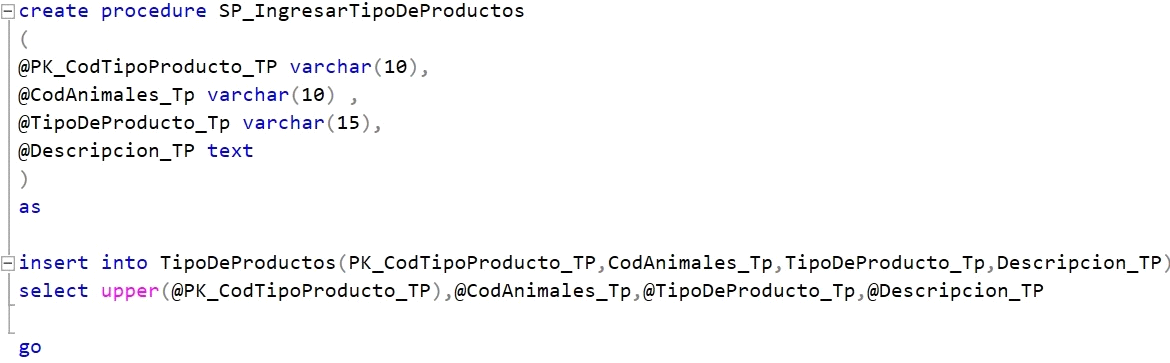
Imagen que contiene Texto

Descripción generada automáticamente

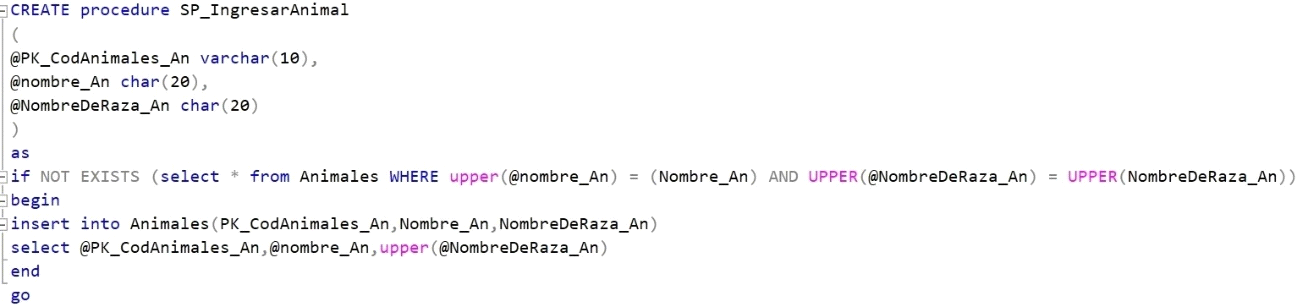
La tabla TipoDeProductos se encarga de registrar los diferentes productos. Esta tabla está conectada con Animales para saber para qué animal es el producto y esto lo conectamos con la foreign key que es CodAnimales\_Tp. También tiene en uno de sus campos con check para que se elija la categoría del producto que puede ser Comida, Salud, Ropa, Accesorios e Higiene. Esta tabla la llamamos TipoDeProducto\_Tp. Como último, la clave primaria es PK\_CodTipoProducto\_TP que es un código que generamos.

**-Procedimientos almacenados:**

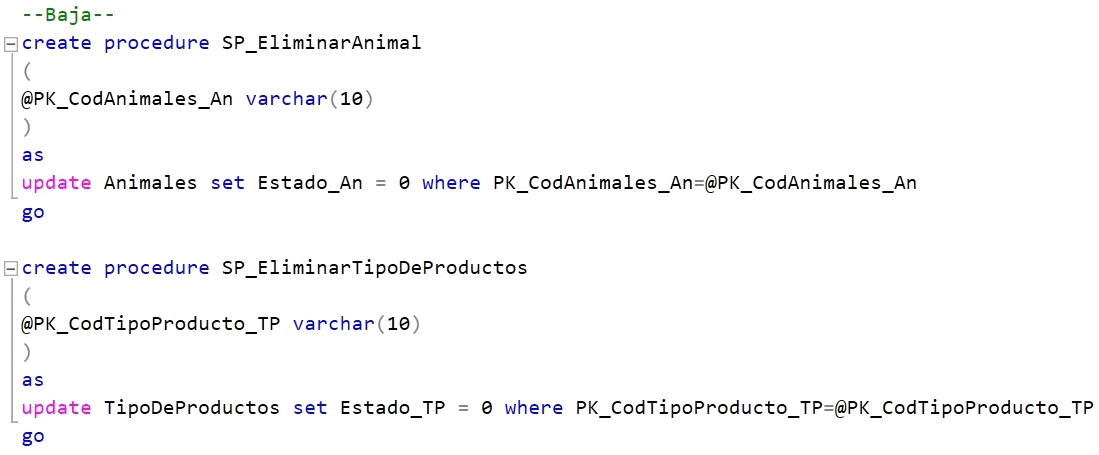
Estos son los procedimientos de ingresar:



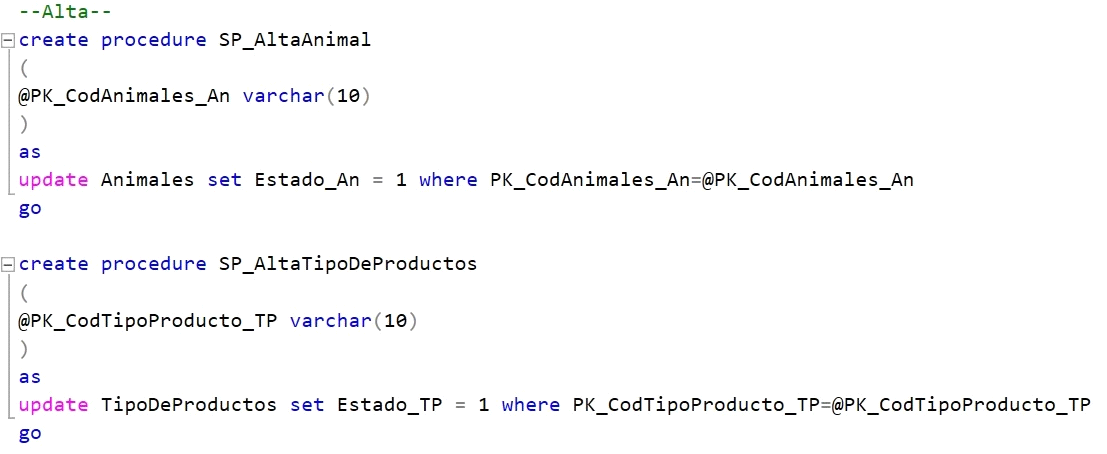
Se ingresa el tipo de producto. Al ingresar se cambia la clave primaria a mayúscula con upper().



Sp\_IngresarAnimal: se ingresan los datos del animal y se compara para que no se repita el animal junto con la raza. Al ingresar el animal se pone en mayúscula con la función upper el nombre de la raza. Para validar que no se ingrese el mismo animal tanto como Nombre y NombreDeRaza se ponen en mayúscula.



Tanto en Animal como en TipoDeProducto se da la baja lógica. Esto se hace porque si otras tablas tienen algún campo de estas tablas puede producir errores. Y esto se logra cambiando el estado a 0.



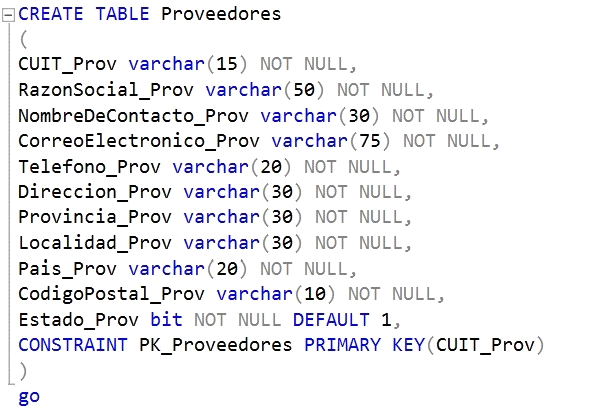
Tanto en Animal como en TipoDeProducto se da de alta lógica cambiando el estado a 1. El alta se da si en algún momento se vuelve a utilizar la información.



Tanto en Animal como en TipoDeProducto cambia los datos menos el estado y la clave primaria. Esto sirve para actualizar o corregir algún error al ingresar el dato.

## JAVIER ANDRÉS TORALES

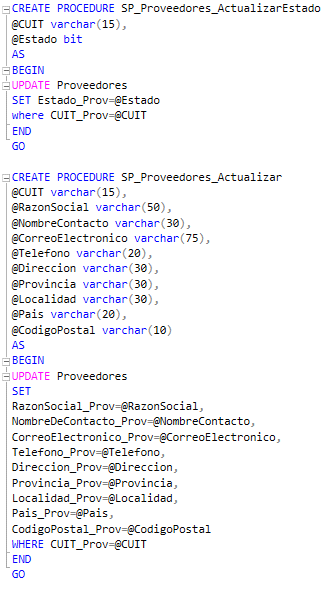
**-Tabla:**



La tabla que yo realicé fue Proveedores. La cual está compuesta por 10 campos y todos ellos del tipo varchar (cada campo tiene su propio tamaño) y NOT NULL para evitar que se ingresen registros sin algún campo en específico. Por último, está la restricción CONSTRAINT en donde se ubica la PRIMARY KEY de la tabla Proveedores, por lo cual la PRIMARY KEY seria PK\_Proveedores apuntando al campo CUIT\_Prov.

**-Procedimientos almacenados:**





Luego, realicé 3 procedimientos almacenados, que serían SP\_Proveedor\_Crear, SP\_Proveedores\_ActualizarEstado, SP\_Proveedores\_Actualizar. Estos se encargarían de crear, actualizar estado que es usado para hacer un borrado lógico de un registro y el ultimo para actualizar un registro. Todos utilizan variables.

**-Trigger:**



Por último, está el Trigger TR\_Proveedores\_PrevenirEliminar que se encarga de prevenir que se hagan consultas del tipo DELETE, básicamente no está permitido eliminar registros en esta tabla. La única forma de hacerlo es mediante un borrado lógico.

## MÁXIMO TOMÁS CANEDO

Yo me encargué de armar las tablas Empleados y Sesiones.

La tabla Empleados contiene toda la información necesaria sobre un empleado.

Observaciones:

* Los empleados pueden tener dos roles: Administrador y Normal.
* “Fecha de Inicio” refiere a la fecha en la que el empleado fue contratado.
* En vez de guardar la clave en formato de texto plano hice lo siguiente:

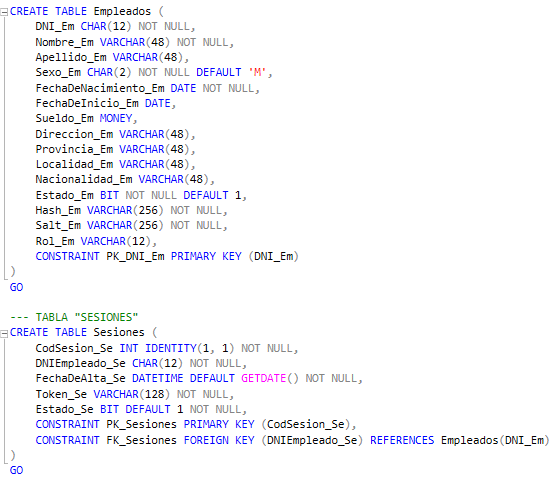
Cuando el usuario crea una cuenta con una clave nueva, se recibe esa clave nueva, se le añade un salt, que son caracteres aleatorios, con la finalidad de fortalecer dicha clave, y luego, al resultado obtenido le aplico Hash. Esto último consiste en aplicar a la cadena (clave + salt) una serie de operaciones que vuelven a la cadena inicial en una cadena de longitud fija y (casi) imposible de deshacer.

En esta tabla se almacenan el Hash y el Salt generados, de forma que cuando el usuario inicia sesión, se recibe la contraseña ingresada, se le añade el salt, y si el hash resultante es estrictamente idéntico al hash almacenado en la tabla Empleados, consideramos que la clave es correcta.

La tabla Sesiones contiene la información sobre todas las sesiones activas e inactivas.

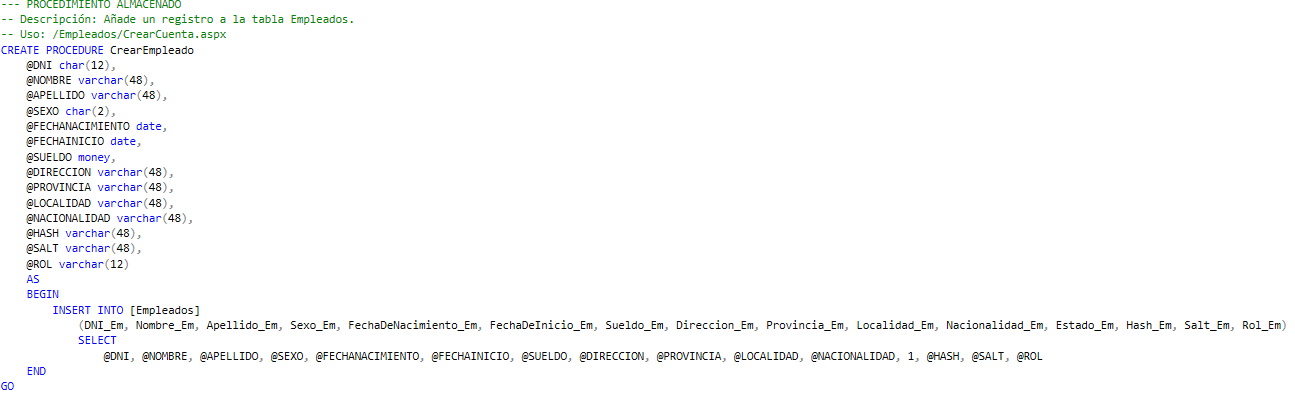
Luego de que el usuario inicia sesión exitosamente, se guardan sus datos dentro de un objeto JSON, y a ese objeto se lo encripta mediante el uso de una frase secreta. La cadena resultante, se almacena entre las cookies del usuario y cada vez que necesite realizar una acción sensible (Como por ejemplo, iniciar una venta o insertar un nuevo producto), se usará el token provisto por el usuario, se lo desencripta, se obtienen sus datos, se verifica que el token no haya expirado y que siga autorizado (El usuario puede revocar tokens / Cerrar sesiones), y de estar todo bien se procede.

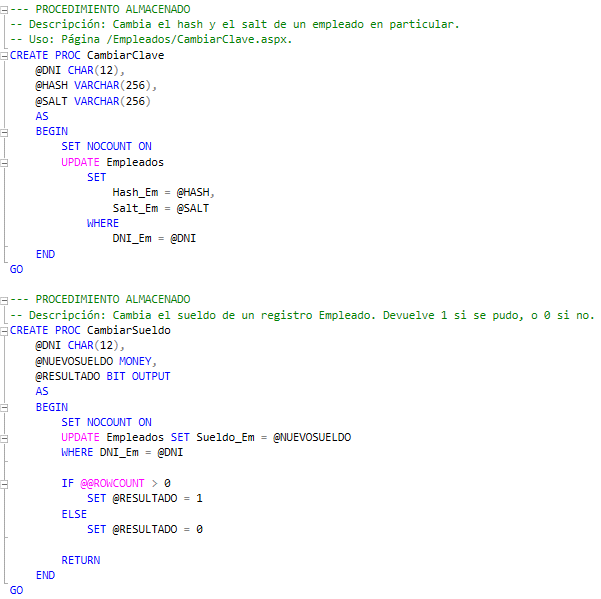
**-Tablas:**

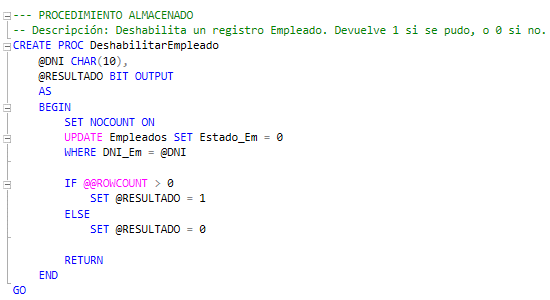


**-Procedimientos almacenados:**

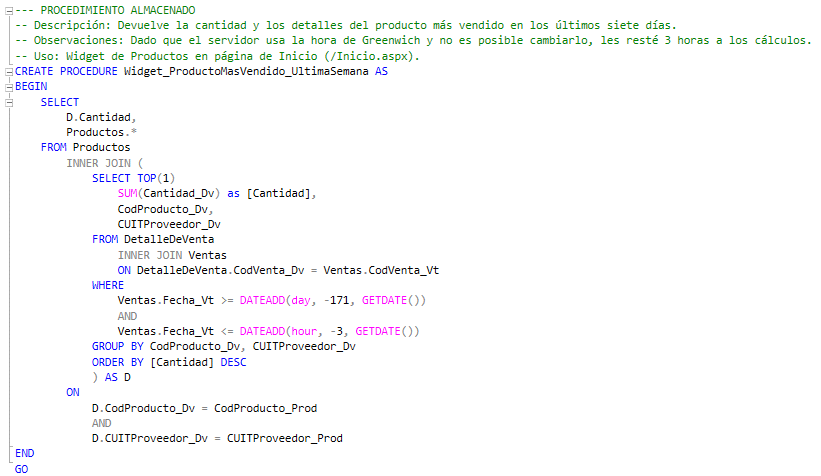
Para Tabla Empleados:

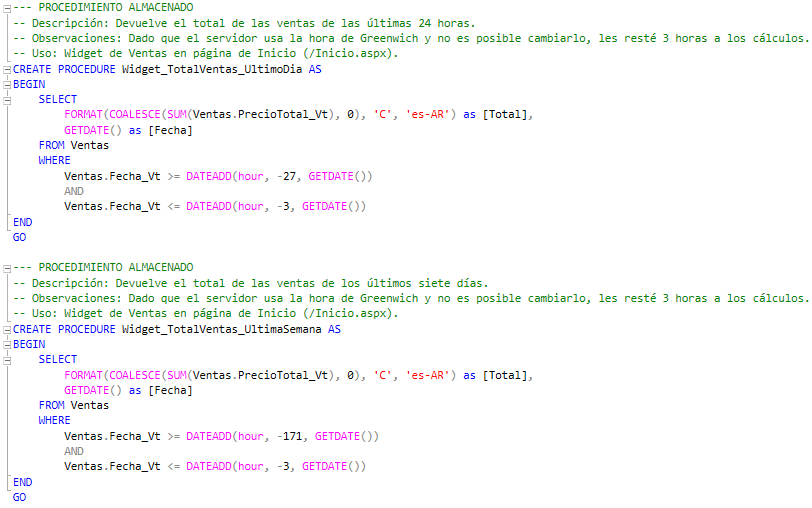


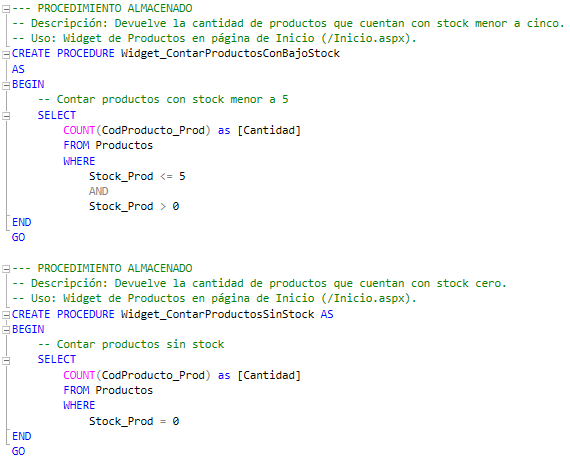




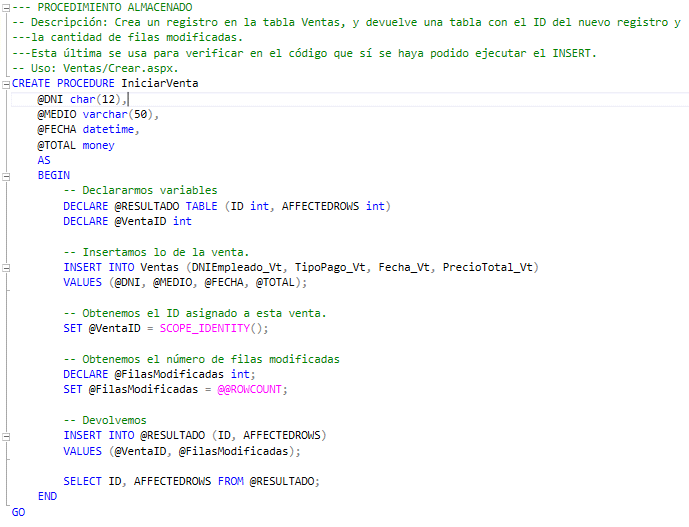
Estos procedimientos son para uso de los Widgets en la página de Inicio.

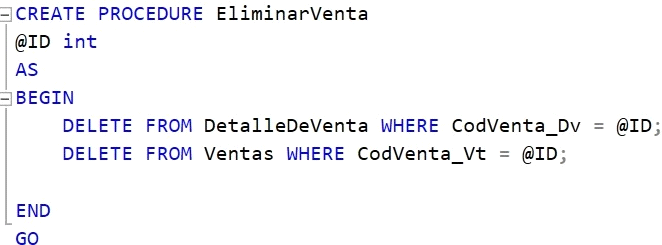






Para la tabla VENTAS:





Triggers:

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente**

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente**

## EZEQUIEL ALEJANDRO MARTÍNEZ

**-Tablas:**

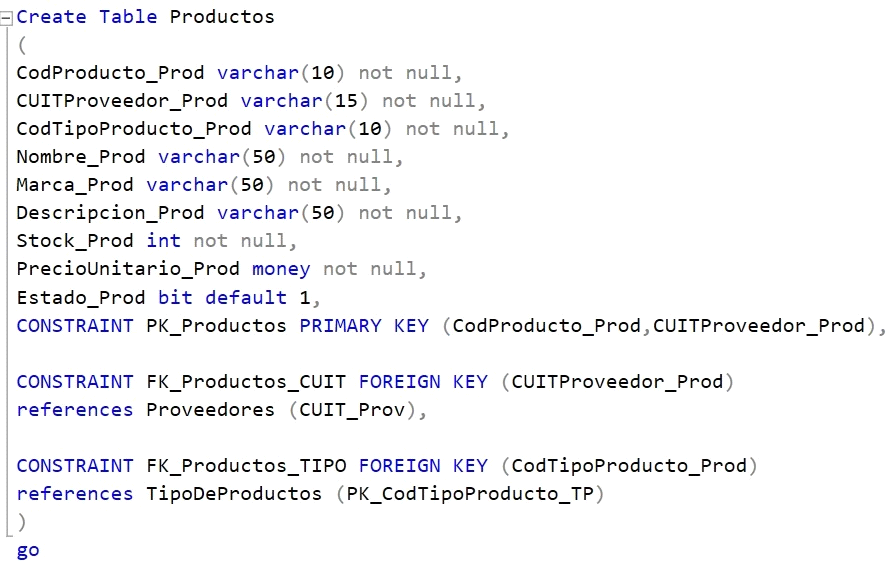


Tabla Productos, donde las claves primarias son “CodProducto\_Prod” y “CUITProveedor\_Prod”, permitiendo asi que existan varios productos con un mismo codigo siempre y cuando el CUIT del proveedor sea diferente. “CuitProveedor” y “CodTipoProducto” son claves foráneas.

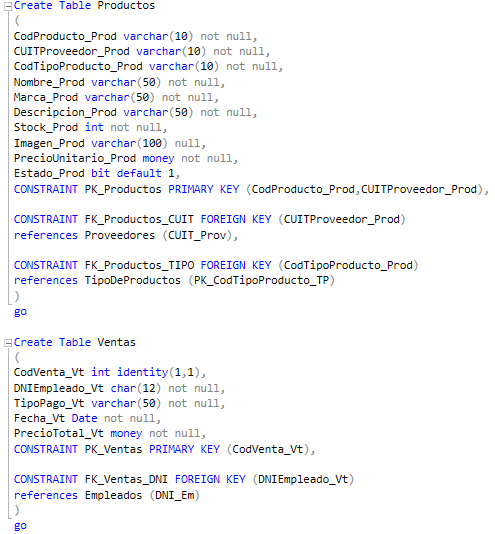
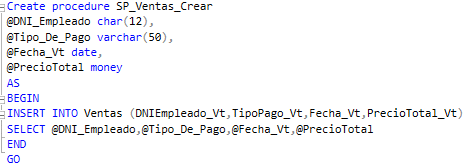
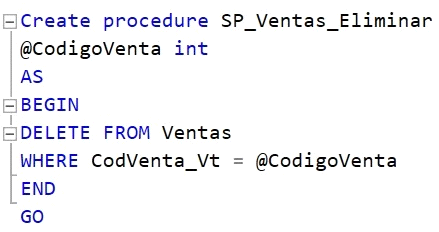


Tabla Ventas, aquí la clave primaria es el “CodVenta\_Vt”, aquí no se maneja nada de los productos ya que en una misma venta puede haber varios productos vendidos, eso se maneja en **Detalle de Ventas**. Tambien en este caso la única clave foránea que posee es la de “DNIEmpleado\_Vt”.

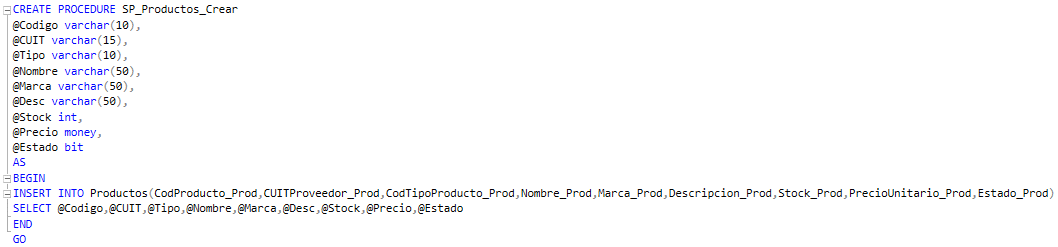
**-Procedimientos almacenados:**

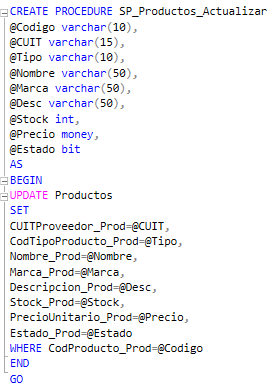


Procedimiento para crear un registro nuevo de una venta desde el ámbito de programación. Ya que la página web posee un apartado para Crear o Eliminar registros.



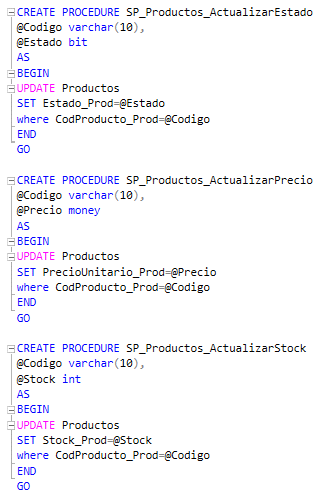
Procedimiento para eliminar un registro Venta desde el ámbito de programación. Este procedimiento fue actualizado a medida que avanzábamos el trabajo por Máximo Canedo para que también borre los registros de DetalleDeVenta.

Procedimiento utilizado para agregar un nuevo producto a la lista desde la página web.

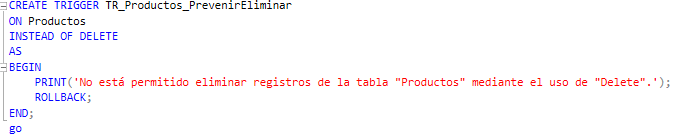


Procedimiento que permite al usuario administrador de la página web editar ciertos campos de un producto.

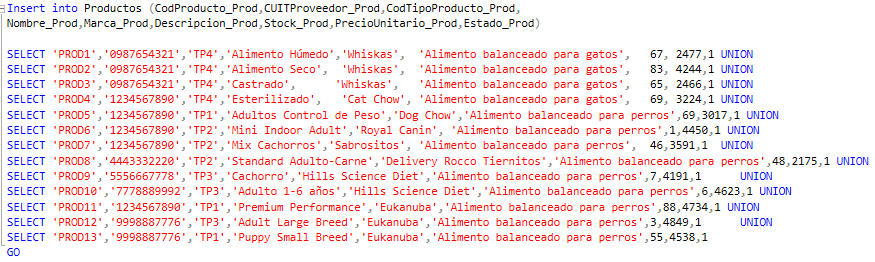
Nota: antes de crear el SP Actualizar, existían los siguientes SP los cuales con el tiempo fueron descartados y reemplazados por el procedimiento **SP\_Productos\_Actualizar** el cual permitía actualizar todo en una misma función sin necesidad de crear varios SP.

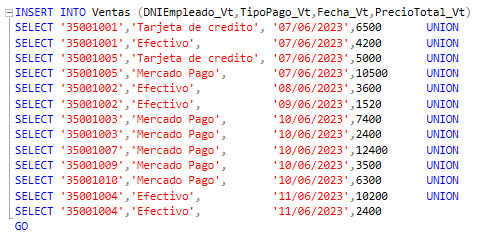


**-Trigger:**



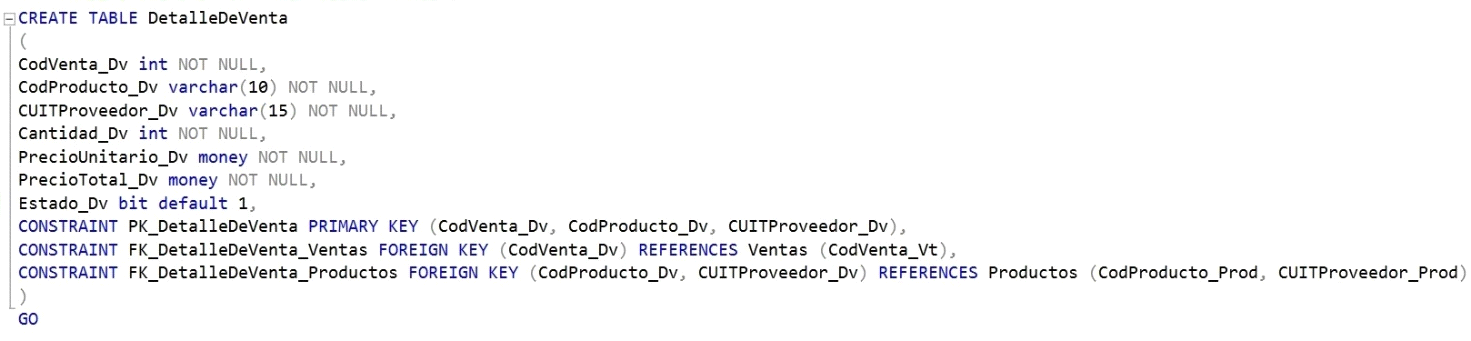
**-Carga de datos:**





## MARÍA OLIVIA HANCZYC

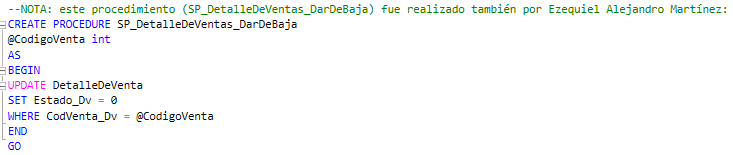
**-Tabla:**

La tabla DetalleDeVenta contiene los campos CodVenta\_Dv (entero, no nulo), CodProducto\_Dv (varchar(10), no nulo), CUITProveedor\_Dv (varchar(15), no nulo), Cantidad\_Dv (entero, no nulo), PrecioUnitario\_Dv (money, no nulo), PrecioTotal\_Dv (money, no nulo), Estado\_Dv (bit, con valor por defecto 1). Los campos no pueden tener datos nulos ya que en esta tabla se registra información relacionada a cada venta realizada.

Tiene una clave primaria concatenada, conformada por los campos CodVenta\_Dv, CodProducto\_Dv y CUITProveedor\_Dv. Tiene como claves foráneas: el campo CodVenta\_Dv, que hace referencia al campo CodVenta\_Vt (clave primaria de la tabla Ventas), y los campos CodProducto\_Dv y CUITProveedor\_Dv, que hacen referencia a los campos CodProducto\_Prod y CUITProveedor\_Prod (clave primaria concatenada de la tabla Productos).

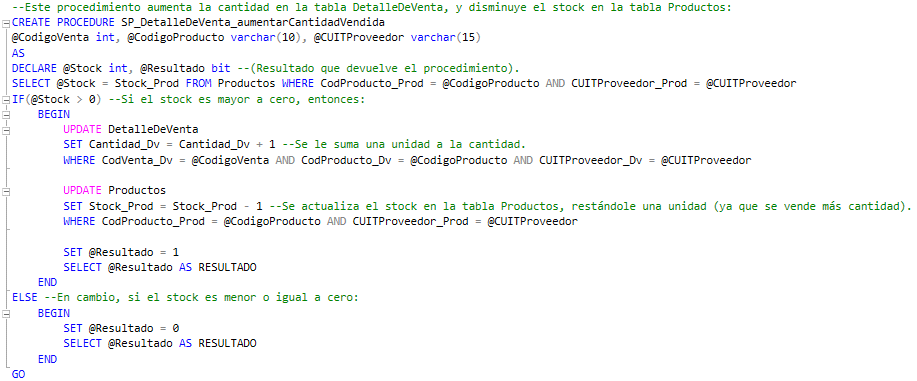
**-Procedimientos almacenados:**

El procedimiento almacenado SP\_DetalleDeVenta\_Agregar me permite insertar un registro al final de la tabla DetalleDeVenta, teniendo en cuenta el precio unitario de la tabla Productos, correspondiente al código de producto enviado por parámetro. Para calcular el precio total, se multiplica la cantidad enviada por parámetro por el precio unitario obtenido previamente. El estado al momento de registrar el detalle de la venta siempre tendrá valor true o 1.



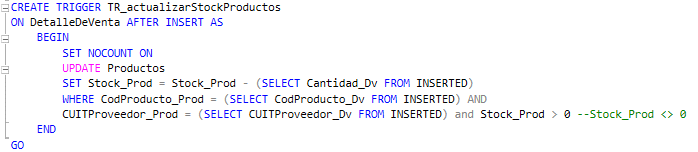
El procedimiento almacenado SP\_DetalleDeVenta\_DarDeBaja establece el estado en false o 0 que corresponda al código de la venta enviado por parámetro. Esto realiza una baja lógica.





Estos dos últimos procedimientos se utilizan para modificar la cantidad vendida en un detalle de venta, ya sea restándola o sumándola de a 1 unidad. Se crearon para darle funcionalidad a 2 botones (uno para la resta y otro para la suma) de la página /Empleado/VerFactura.aspx de la aplicación web (esta opción solo está disponible para aquellos empleados que sean administradores o para aquel que sea el gestor de la venta). En cada procedimiento, primero se verifica si el stock del producto en cuestión (comparando el código del producto y el CUIT del proveedor contra dichos datos enviados por parámetros) es mayor a 0. De ser así, se realizan los cálculos indicados y el resultado devuelve true. En caso contrario, el resultado devuelve false.

**-Trigger:**



Este trigger se ejecuta cada vez que se inserte un registro en la tabla DetalleDeVenta. Lo que hace es restarle la cantidad vendida al stock del producto correspondiente al registro agregado.

**-Carga de datos:**



Aclaración: estos fueron los primeros registros que inserté. Luego, con el procedimiento almacenado SP\_DetalleDeVenta\_Agregar fuimos añadiendo nuevos registros.

## SCRIPT

use master

go

create database Pets

go

use Pets

go

/\*\*\*\* TABLAS \*\*\*\*/

---- TABLAS CREADAS POR JEAN ESQUEN ----

Create table Animales

(

Pk\_CodAnimales\_An varchar(10) not null,

Nombre\_An char(20) not null,

NombreDeRaza\_An char(20) null,

Estado\_An bit not null default 1,

Constraint PK\_Animales primary key (Pk\_CodAnimales\_An)

)

go

create TABLE TipoDeProductos

(

PK\_CodTipoProducto\_TP varchar(10) NOT NULL,

CodAnimales\_Tp varchar(10) NOT NULL,

TipoDeProducto\_Tp varchar(15) NOT NULL CHECK(TipoDeProducto\_Tp='Comida' OR TipoDeProducto\_Tp='Accesorios' OR TipoDeProducto\_Tp='Ropa' OR TipoDeProducto\_Tp='Higiene' OR TipoDeProducto\_Tp='Salud'),

Descripcion\_TP text NOT NULL,

Estado\_Tp bit not null default 1,

Constraint PK\_TipoDeProductos primary key (PK\_CodTipoProducto\_TP),

constraint FK\_TipoDeProductosxAnimales foreign key (CodAnimales\_Tp)

references Animales(Pk\_CodAnimales\_An)

)

go

---- TABLAS CREADAS POR JAVIER ANDRÉS TORALES ----

CREATE TABLE Proveedores

(

CUIT\_Prov varchar(15) NOT NULL,

RazonSocial\_Prov varchar(50) NOT NULL,

NombreDeContacto\_Prov varchar(30) NOT NULL,

CorreoElectronico\_Prov varchar(75) NOT NULL,

Telefono\_Prov varchar(20) NOT NULL,

Direccion\_Prov varchar(30) NOT NULL,

Provincia\_Prov varchar(30) NOT NULL,

Localidad\_Prov varchar(30) NOT NULL,

Pais\_Prov varchar(20) NOT NULL,

CodigoPostal\_Prov varchar(10) NOT NULL,

Estado\_Prov bit NOT NULL DEFAULT 1,

CONSTRAINT PK\_Proveedores PRIMARY KEY(CUIT\_Prov)

)

go

---- TABLAS CREADAS POR MÁXIMO CANEDO ----

CREATE TABLE Empleados (

DNI\_Em CHAR(12) NOT NULL,

Nombre\_Em VARCHAR(48) NOT NULL,

Apellido\_Em VARCHAR(48),

Sexo\_Em CHAR(2) NOT NULL DEFAULT 'M',

FechaDeNacimiento\_Em DATE NOT NULL,

FechaDeInicio\_Em DATE,

Sueldo\_Em MONEY,

Direccion\_Em VARCHAR(48),

Provincia\_Em VARCHAR(48),

Localidad\_Em VARCHAR(48),

Nacionalidad\_Em VARCHAR(48),

Estado\_Em BIT NOT NULL DEFAULT 1,

Hash\_Em VARCHAR(256) NOT NULL,

Salt\_Em VARCHAR(256) NOT NULL,

Rol\_Em VARCHAR(12),

CONSTRAINT PK\_DNI\_Em PRIMARY KEY (DNI\_Em)

)

GO

--- TABLA "SESIONES"

CREATE TABLE Sesiones (

CodSesion\_Se INT IDENTITY(1, 1) NOT NULL,

DNIEmpleado\_Se CHAR(12) NOT NULL,

FechaDeAlta\_Se DATETIME DEFAULT GETDATE() NOT NULL,

Token\_Se VARCHAR(128) NOT NULL,

Estado\_Se BIT DEFAULT 1 NOT NULL,

CONSTRAINT PK\_Sesiones PRIMARY KEY (CodSesion\_Se),

CONSTRAINT FK\_Sesiones FOREIGN KEY (DNIEmpleado\_Se) REFERENCES Empleados(DNI\_Em)

)

GO

---- TABLAS CREADAS POR EZEQUIEL ALEJANDRO MARTíNEZ ----

Create Table Productos

(

CodProducto\_Prod varchar(10) not null,

CUITProveedor\_Prod varchar(15) not null,

CodTipoProducto\_Prod varchar(10) not null,

Nombre\_Prod varchar(50) not null,

Marca\_Prod varchar(50) not null,

Descripcion\_Prod varchar(50) not null,

Stock\_Prod int not null,

PrecioUnitario\_Prod money not null,

Estado\_Prod bit default 1,

CONSTRAINT PK\_Productos PRIMARY KEY (CodProducto\_Prod,CUITProveedor\_Prod),

CONSTRAINT FK\_Productos\_CUIT FOREIGN KEY (CUITProveedor\_Prod)

references Proveedores (CUIT\_Prov),

CONSTRAINT FK\_Productos\_TIPO FOREIGN KEY (CodTipoProducto\_Prod)

references TipoDeProductos (PK\_CodTipoProducto\_TP)

)

go

Create Table Ventas

(

CodVenta\_Vt int identity(1,1),

DNIEmpleado\_Vt char(12) not null,

TipoPago\_Vt varchar(50) not null,

Fecha\_Vt Date not null,

PrecioTotal\_Vt money not null,

CONSTRAINT PK\_Ventas PRIMARY KEY (CodVenta\_Vt),

CONSTRAINT FK\_Ventas\_DNI FOREIGN KEY (DNIEmpleado\_Vt)

references Empleados (DNI\_Em)

)

go

---- TABLAS CREADAS POR MARÍA OLIVIA HANCZYC ----

CREATE TABLE DetalleDeVenta

(

CodVenta\_Dv int NOT NULL,

CodProducto\_Dv varchar(10) NOT NULL,

CUITProveedor\_Dv varchar(15) NOT NULL,

Cantidad\_Dv int NOT NULL,

PrecioUnitario\_Dv money NOT NULL,

PrecioTotal\_Dv money NOT NULL,

Estado\_Dv bit default 1,

CONSTRAINT PK\_DetalleDeVenta PRIMARY KEY (CodVenta\_Dv, CodProducto\_Dv, CUITProveedor\_Dv),

CONSTRAINT FK\_DetalleDeVenta\_Ventas FOREIGN KEY (CodVenta\_Dv) REFERENCES Ventas (CodVenta\_Vt),

CONSTRAINT FK\_DetalleDeVenta\_Productos FOREIGN KEY (CodProducto\_Dv, CUITProveedor\_Dv) REFERENCES Productos (CodProducto\_Prod, CUITProveedor\_Prod)

)

GO

/\*\*\*\* PROCEDIMIENTOS ALMACENADOS \*\*\*\*/

----------------------------------------------------------------------------------------------------------------------------------------------

/\*-- PROCEDIMIENTOS CREADOS POR JEAN ESQUEN ---\*/

--Agregar--

create procedure SP\_IngresarTipoDeProductos

(

@PK\_CodTipoProducto\_TP varchar(10),

@CodAnimales\_Tp varchar(10) ,

@TipoDeProducto\_Tp varchar(15),

@Descripcion\_TP text

)

as

insert into TipoDeProductos(PK\_CodTipoProducto\_TP,CodAnimales\_Tp,TipoDeProducto\_Tp,Descripcion\_TP)

select upper(@PK\_CodTipoProducto\_TP),@CodAnimales\_Tp,@TipoDeProducto\_Tp,@Descripcion\_TP

go

CREATE procedure SP\_IngresarAnimal

(

@PK\_CodAnimales\_An varchar(10),

@nombre\_An char(20),

@NombreDeRaza\_An char(20)

)

as

if NOT EXISTS (select \* from Animales WHERE upper(@nombre\_An) = (Nombre\_An) AND UPPER(@NombreDeRaza\_An) = UPPER(NombreDeRaza\_An))

begin

insert into Animales(PK\_CodAnimales\_An,Nombre\_An,NombreDeRaza\_An)

select @PK\_CodAnimales\_An,@nombre\_An,upper(@NombreDeRaza\_An)

end

go

--Baja--

create procedure SP\_EliminarAnimal

(

@PK\_CodAnimales\_An varchar(10)

)

as

update Animales set Estado\_An = 0 where PK\_CodAnimales\_An=@PK\_CodAnimales\_An

go

create procedure SP\_EliminarTipoDeProductos

(

@PK\_CodTipoProducto\_TP varchar(10)

)

as

update TipoDeProductos set Estado\_TP = 0 where PK\_CodTipoProducto\_TP=@PK\_CodTipoProducto\_TP

go

--Alta--

create procedure SP\_AltaAnimal

(

@PK\_CodAnimales\_An varchar(10)

)

as

update Animales set Estado\_An = 1 where PK\_CodAnimales\_An=@PK\_CodAnimales\_An

go

create procedure SP\_AltaTipoDeProductos

(

@PK\_CodTipoProducto\_TP varchar(10)

)

as

update TipoDeProductos set Estado\_TP = 1 where PK\_CodTipoProducto\_TP=@PK\_CodTipoProducto\_TP

go

--Actualizar--

create procedure SP\_ActualizarTipoProducto

(

@PK\_CodTipoProducto\_TP varchar(10),

@CodAnimales\_Tp varchar(10),

@TipoDeProducto\_Tp varchar(15),

@Descripcion\_TP text

)

as

update TipoDeProductos set CodAnimales\_Tp = @CodAnimales\_Tp,TipoDeProducto\_Tp = @TipoDeProducto\_Tp,Descripcion\_TP = @Descripcion\_TP

where PK\_CodTipoProducto\_TP = @PK\_CodTipoProducto\_TP

go

create procedure SP\_ActualizarAnimales

(

@PK\_CodAnimales\_An varchar(10),

@nombre\_An char(20),

@NombreDeRaza\_An char(20)

)

AS

update Animales set Nombre\_An=@nombre\_An,NombreDeRaza\_An = upper(@NombreDeRaza\_An) where PK\_CodAnimales\_An=@PK\_CodAnimales\_An

go

----------------------------------------------------------------------------------------------------------------------------------------------

/\*-- PROCEDIMIENTOS CREADOS POR MÁXIMO CANEDO ---\*/

--- PROCEDIMIENTO ALMACENADO

-- Descripción: Añade un registro a la tabla Empleados.

-- Uso: /Empleados/CrearCuenta.aspx

CREATE PROCEDURE CrearEmpleado

@DNI char(12),

@NOMBRE varchar(48),

@APELLIDO varchar(48),

@SEXO char(2),

@FECHANACIMIENTO date,

@FECHAINICIO date,

@SUELDO money,

@DIRECCION varchar(48),

@PROVINCIA varchar(48),

@LOCALIDAD varchar(48),

@NACIONALIDAD varchar(48),

@HASH varchar(48),

@SALT varchar(48),

@ROL varchar(12)

AS

BEGIN

INSERT INTO [Empleados]

(DNI\_Em, Nombre\_Em, Apellido\_Em, Sexo\_Em, FechaDeNacimiento\_Em, FechaDeInicio\_Em, Sueldo\_Em, Direccion\_Em, Provincia\_Em, Localidad\_Em, Nacionalidad\_Em, Estado\_Em, Hash\_Em, Salt\_Em, Rol\_Em)

SELECT

@DNI, @NOMBRE, @APELLIDO, @SEXO, @FECHANACIMIENTO, @FECHAINICIO, @SUELDO, @DIRECCION, @PROVINCIA, @LOCALIDAD, @NACIONALIDAD, 1, @HASH, @SALT, @ROL

END

GO

--- PROCEDIMIENTO ALMACENADO

-- Descripción: Cambia el hash y el salt de un empleado en particular.

-- Uso: Página /Empleados/CambiarClave.aspx.

CREATE PROC CambiarClave

@DNI CHAR(12),

@HASH VARCHAR(256),

@SALT VARCHAR(256)

AS

BEGIN

SET NOCOUNT ON

UPDATE Empleados

SET

Hash\_Em = @HASH,

Salt\_Em = @SALT

WHERE

DNI\_Em = @DNI

END

GO

--- PROCEDIMIENTO ALMACENADO

-- Descripción: Cambia el sueldo de un registro Empleado. Devuelve 1 si se pudo, o 0 si no.

CREATE PROC CambiarSueldo

@DNI CHAR(12),

@NUEVOSUELDO MONEY,

@RESULTADO BIT OUTPUT

AS

BEGIN

SET NOCOUNT ON

UPDATE Empleados SET Sueldo\_Em = @NUEVOSUELDO

WHERE DNI\_Em = @DNI

IF @@ROWCOUNT > 0

SET @RESULTADO = 1

ELSE

SET @RESULTADO = 0

RETURN

END

GO

--- PROCEDIMIENTO ALMACENADO

-- Descripción: Deshabilita un registro Empleado. Devuelve 1 si se pudo, o 0 si no.

CREATE PROC DeshabilitarEmpleado

@DNI CHAR(10),

@RESULTADO BIT OUTPUT

AS

BEGIN

SET NOCOUNT ON

UPDATE Empleados SET Estado\_Em = 0

WHERE DNI\_Em = @DNI

IF @@ROWCOUNT > 0

SET @RESULTADO = 1

ELSE

SET @RESULTADO = 0

RETURN

END

GO

--- PROCEDIMIENTO ALMACENADO

-- Descripción: Devuelve la cantidad y los detalles del producto más vendido en los últimos siete días.

-- Observaciones: Dado que el servidor usa la hora de Greenwich y no es posible cambiarlo, les resté 3 horas a los cálculos.

-- Uso: Widget de Productos en página de Inicio (/Inicio.aspx).

CREATE PROCEDURE Widget\_ProductoMasVendido\_UltimaSemana AS

BEGIN

SELECT

D.Cantidad,

Productos.\*

FROM Productos

INNER JOIN (

SELECT TOP(1)

SUM(Cantidad\_Dv) as [Cantidad],

CodProducto\_Dv,

CUITProveedor\_Dv

FROM DetalleDeVenta

INNER JOIN Ventas

ON DetalleDeVenta.CodVenta\_Dv = Ventas.CodVenta\_Vt

WHERE

Ventas.Fecha\_Vt >= DATEADD(day, -171, GETDATE())

AND

Ventas.Fecha\_Vt <= DATEADD(hour, -3, GETDATE())

GROUP BY CodProducto\_Dv, CUITProveedor\_Dv

ORDER BY [Cantidad] DESC

) AS D

ON

D.CodProducto\_Dv = CodProducto\_Prod

AND

D.CUITProveedor\_Dv = CUITProveedor\_Prod

END

GO

--- PROCEDIMIENTO ALMACENADO

-- Descripción: Devuelve el total de las ventas de las últimas 24 horas.

-- Observaciones: Dado que el servidor usa la hora de Greenwich y no es posible cambiarlo, les resté 3 horas a los cálculos.

-- Uso: Widget de Ventas en página de Inicio (/Inicio.aspx).

CREATE PROCEDURE Widget\_TotalVentas\_UltimoDia AS

BEGIN

SELECT

FORMAT(COALESCE(SUM(Ventas.PrecioTotal\_Vt), 0), 'C', 'es-AR') as [Total],

GETDATE() as [Fecha]

FROM Ventas

WHERE

Ventas.Fecha\_Vt >= DATEADD(hour, -27, GETDATE())

AND

Ventas.Fecha\_Vt <= DATEADD(hour, -3, GETDATE())

END

GO

--- PROCEDIMIENTO ALMACENADO

-- Descripción: Devuelve el total de las ventas de los últimos siete días.

-- Observaciones: Dado que el servidor usa la hora de Greenwich y no es posible cambiarlo, les resté 3 horas a los cálculos.

-- Uso: Widget de Ventas en página de Inicio (/Inicio.aspx).

CREATE PROCEDURE Widget\_TotalVentas\_UltimaSemana AS

BEGIN

SELECT

FORMAT(COALESCE(SUM(Ventas.PrecioTotal\_Vt), 0), 'C', 'es-AR') as [Total],

GETDATE() as [Fecha]

FROM Ventas

WHERE

Ventas.Fecha\_Vt >= DATEADD(hour, -171, GETDATE())

AND

Ventas.Fecha\_Vt <= DATEADD(hour, -3, GETDATE())

END

GO

--- PROCEDIMIENTO ALMACENADO

-- Descripción: Devuelve la cantidad de productos que cuentan con stock menor a cinco.

-- Uso: Widget de Productos en página de Inicio (/Inicio.aspx).

CREATE PROCEDURE Widget\_ContarProductosConBajoStock

AS

BEGIN

-- Contar productos con stock menor a 5

SELECT

COUNT(CodProducto\_Prod) as [Cantidad]

FROM Productos

WHERE

Stock\_Prod <= 5

AND

Stock\_Prod > 0

END

GO

--- PROCEDIMIENTO ALMACENADO

-- Descripción: Devuelve la cantidad de productos que cuentan con stock cero.

-- Uso: Widget de Productos en página de Inicio (/Inicio.aspx).

CREATE PROCEDURE Widget\_ContarProductosSinStock AS

BEGIN

-- Contar productos sin stock

SELECT

COUNT(CodProducto\_Prod) as [Cantidad]

FROM Productos

WHERE

Stock\_Prod = 0

END

GO

--- PROCEDIMIENTO ALMACENADO

-- Descripción: Crea un registro en la tabla Ventas, y devuelve una tabla con el ID del nuevo registro y la cantidad de filas modificadas. Esta última se usa para verificar en el código que sí se haya podido ejecutar el INSERT.

-- Uso: Ventas/Crear.aspx.

CREATE PROCEDURE IniciarVenta

@DNI char(12),

@MEDIO varchar(50),

@FECHA datetime,

@TOTAL money

AS

BEGIN

-- Declaramos variables

DECLARE @RESULTADO TABLE (ID int, AFFECTEDROWS int)

DECLARE @VentaID int

-- Insertamos lo de la venta.

INSERT INTO Ventas (DNIEmpleado\_Vt, TipoPago\_Vt, Fecha\_Vt, PrecioTotal\_Vt)

VALUES (@DNI, @MEDIO, @FECHA, @TOTAL);

-- Obtenemos el ID asignado a esta venta.

SET @VentaID = SCOPE\_IDENTITY();

-- Obtenemos el número de filas modificadas

DECLARE @FilasModificadas int;

SET @FilasModificadas = @@ROWCOUNT;

-- Devolvemos

INSERT INTO @RESULTADO (ID, AFFECTEDROWS)

VALUES (@VentaID, @FilasModificadas);

SELECT ID, AFFECTEDROWS FROM @RESULTADO;

END

GO

CREATE PROCEDURE EliminarVenta

@ID int

AS

BEGIN

DELETE FROM DetalleDeVenta WHERE CodVenta\_Dv = @ID;

DELETE FROM Ventas WHERE CodVenta\_Vt = @ID;

END

GO

----------------------------------------------------------------------------------------------------------------------------------------------

/\*-- PROCEDIMIENTOS CREADOS POR JAVIER ANDRÉS TORALES ---\*/

CREATE PROCEDURE SP\_Proveedor\_Crear

@CUIT varchar(15),

@RazonSocial varchar(50),

@NombreContacto varchar(30),

@CorreoElectronico varchar(75),

@Telefono varchar(20),

@Direccion varchar(30),

@Provincia varchar(30),

@Localidad varchar(30),

@Pais varchar(20),

@CodigoPostal varchar(10)

AS

BEGIN

INSERT INTO Proveedores(CUIT\_Prov,RazonSocial\_Prov,NombreDeContacto\_Prov,CorreoElectronico\_Prov,Telefono\_Prov,Direccion\_Prov,Provincia\_Prov,Localidad\_Prov,Pais\_Prov,CodigoPostal\_Prov)

SELECT @CUIT,@RazonSocial,@NombreContacto,@CorreoElectronico,@Telefono,@Direccion,@Provincia,@Localidad,@Pais,@CodigoPostal

END

GO

CREATE PROCEDURE SP\_Proveedores\_ActualizarEstado

@CUIT varchar(15),

@Estado bit

AS

BEGIN

UPDATE Proveedores

SET Estado\_Prov=@Estado

where CUIT\_Prov=@CUIT

END

GO

CREATE PROCEDURE SP\_Proveedores\_Actualizar

@CUIT varchar(15),

@RazonSocial varchar(50),

@NombreContacto varchar(30),

@CorreoElectronico varchar(75),

@Telefono varchar(20),

@Direccion varchar(30),

@Provincia varchar(30),

@Localidad varchar(30),

@Pais varchar(20),

@CodigoPostal varchar(10)

AS

BEGIN

UPDATE Proveedores

SET

RazonSocial\_Prov=@RazonSocial,

NombreDeContacto\_Prov=@NombreContacto,

CorreoElectronico\_Prov=@CorreoElectronico,

Telefono\_Prov=@Telefono,

Direccion\_Prov=@Direccion,

Provincia\_Prov=@Provincia,

Localidad\_Prov=@Localidad,

Pais\_Prov=@Pais,

CodigoPostal\_Prov=@CodigoPostal

WHERE CUIT\_Prov=@CUIT

END

GO----------------------------------------------------------------------------------------------------------------------------------------------

/\*-- PROCEDIMIENTOS CREADOS POR MARÍA OLIVIA HANCZYC ---\*/

CREATE PROCEDURE SP\_DetalleDeVenta\_Agregar

@CodigoVenta int,

@CodigoProducto varchar(10),

@CUITProveedor varchar(15),

@Cantidad int

AS

BEGIN

DECLARE @PrecioUnitario money

--Obtener el precio unitario de la tabla Productos:

SELECT @PrecioUnitario = PrecioUnitario\_Prod

FROM Productos

WHERE CodProducto\_Prod = @CodigoProducto

--Insertar el registro en la tabla DetalleDeVenta:

INSERT INTO DetalleDeVenta (CodVenta\_Dv, CodProducto\_Dv, CUITProveedor\_Dv, Cantidad\_Dv, PrecioUnitario\_Dv, PrecioTotal\_Dv, Estado\_Dv)

SELECT @CodigoVenta, @CodigoProducto, @CUITProveedor, @Cantidad, @PrecioUnitario, (@Cantidad \* @PrecioUnitario), 1

END

GO

--NOTA: este procedimiento (SP\_DetalleDeVentas\_DarDeBaja) fue realizado también por Ezequiel Alejandro Martínez:

CREATE PROCEDURE SP\_DetalleDeVentas\_DarDeBaja

@CodigoVenta int

AS

BEGIN

UPDATE DetalleDeVenta

SET Estado\_Dv = 0

WHERE CodVenta\_Dv = @CodigoVenta

END

GO

--Este procedimiento disminuye la cantidad en la tabla DetalleDeVenta, y aumenta el stock en la tabla Productos:

CREATE PROCEDURE SP\_DetalleDeVenta\_disminuirCantidadVendida

@CodigoVenta int, @CodigoProducto varchar(10), @CUITProveedor varchar(15)

AS

DECLARE @Stock int, @Resultado bit --(Resultado que devuelve el procedimiento).

SELECT @Stock = Stock\_Prod FROM Productos WHERE CodProducto\_Prod = @CodigoProducto AND CUITProveedor\_Prod = @CUITProveedor

IF(@Stock > 0) --Si el stock es mayor a cero, entonces:

BEGIN

UPDATE DetalleDeVenta

SET Cantidad\_Dv = Cantidad\_Dv - 1 --Se le resta una unidad a la cantidad.

WHERE CodVenta\_Dv = @CodigoVenta AND CodProducto\_Dv = @CodigoProducto AND CUITProveedor\_Dv = @CUITProveedor

UPDATE Productos

SET Stock\_Prod = Stock\_Prod + 1 --Se actualiza el stock en la tabla Productos, sumándole una unidad (ya que se vende menos cantidad).

WHERE CodProducto\_Prod = @CodigoProducto AND CUITProveedor\_Prod = @CUITProveedor

SET @Resultado = 1

SELECT @Resultado AS RESULTADO

END

ELSE --En cambio, si el stock es menor o igual a cero:

BEGIN

SET @Resultado = 0

SELECT @Resultado AS RESULTADO

END

GO

--Este procedimiento aumenta la cantidad en la tabla DetalleDeVenta, y disminuye el stock en la tabla Productos:

CREATE PROCEDURE SP\_DetalleDeVenta\_aumentarCantidadVendida

@CodigoVenta int, @CodigoProducto varchar(10), @CUITProveedor varchar(15)

AS

DECLARE @Stock int, @Resultado bit --(Resultado que devuelve el procedimiento).

SELECT @Stock = Stock\_Prod FROM Productos WHERE CodProducto\_Prod = @CodigoProducto AND CUITProveedor\_Prod = @CUITProveedor

IF(@Stock > 0) --Si el stock es mayor a cero, entonces:

BEGIN

UPDATE DetalleDeVenta

SET Cantidad\_Dv = Cantidad\_Dv + 1 --Se le suma una unidad a la cantidad.

WHERE CodVenta\_Dv = @CodigoVenta AND CodProducto\_Dv = @CodigoProducto AND CUITProveedor\_Dv = @CUITProveedor

UPDATE Productos

SET Stock\_Prod = Stock\_Prod - 1 --Se actualiza el stock en la tabla Productos, restándole una unidad (ya que se vende más cantidad).

WHERE CodProducto\_Prod = @CodigoProducto AND CUITProveedor\_Prod = @CUITProveedor

SET @Resultado = 1

SELECT @Resultado AS RESULTADO

END

ELSE --En cambio, si el stock es menor o igual a cero:

BEGIN

SET @Resultado = 0

SELECT @Resultado AS RESULTADO

END

GO

----------------------------------------------------------------------------------------------------------------------------------------------

/\*-- PROCEDIMIENTOS CREADOS POR EZEQUIEL ALEJANDRO MARTíNEZ ---\*/

Create procedure SP\_Ventas\_Crear

@DNI\_Empleado char(12),

@Tipo\_De\_Pago varchar(50),

@Fecha\_Vt date,

@PrecioTotal money

AS

BEGIN

INSERT INTO Ventas (DNIEmpleado\_Vt,TipoPago\_Vt,Fecha\_Vt,PrecioTotal\_Vt)

SELECT @DNI\_Empleado,@Tipo\_De\_Pago,@Fecha\_Vt,@PrecioTotal

END

GO

Create procedure SP\_Ventas\_Eliminar

@CodigoVenta int

AS

BEGIN

DELETE FROM Ventas

WHERE CodVenta\_Vt = @CodigoVenta

END

GO

CREATE PROCEDURE SP\_Productos\_Crear

@Codigo varchar(10),

@CUIT varchar(15),

@Tipo varchar(10),

@Nombre varchar(50),

@Marca varchar(50),

@Desc varchar(50),

@Stock int,

@Precio money,

@Estado bit

AS

BEGIN

INSERT INTO Productos(CodProducto\_Prod,CUITProveedor\_Prod,CodTipoProducto\_Prod,Nombre\_Prod,Marca\_Prod,Descripcion\_Prod,Stock\_Prod,PrecioUnitario\_Prod,Estado\_Prod)

SELECT @Codigo,@CUIT,@Tipo,@Nombre,@Marca,@Desc,@Stock,@Precio,@Estado

END

GO

CREATE PROCEDURE SP\_Productos\_ActualizarEstado

@Codigo varchar(10),

@Estado bit

AS

BEGIN

UPDATE Productos

SET Estado\_Prod=@Estado

where CodProducto\_Prod=@Codigo

END

GO

CREATE PROCEDURE SP\_Productos\_ActualizarPrecio

@Codigo varchar(10),

@Precio money

AS

BEGIN

UPDATE Productos

SET PrecioUnitario\_Prod=@Precio

where CodProducto\_Prod=@Codigo

END

GO

CREATE PROCEDURE SP\_Productos\_ActualizarStock

@Codigo varchar(10),

@Stock int

AS

BEGIN

UPDATE Productos

SET Stock\_Prod=@Stock

where CodProducto\_Prod=@Codigo

END

GO

CREATE PROCEDURE SP\_Productos\_Actualizar

@Codigo varchar(10),

@CUIT varchar(15),

@Tipo varchar(10),

@Nombre varchar(50),

@Marca varchar(50),

@Desc varchar(50),

@Stock int,

@Precio money,

@Estado bit

AS

BEGIN

UPDATE Productos

SET

CUITProveedor\_Prod=@CUIT,

CodTipoProducto\_Prod=@Tipo,

Nombre\_Prod=@Nombre,

Marca\_Prod=@Marca,

Descripcion\_Prod=@Desc,

Stock\_Prod=@Stock,

PrecioUnitario\_Prod=@Precio,

Estado\_Prod=@Estado

WHERE CodProducto\_Prod=@Codigo

END

GO

/\*\*\*\* Triggers \*\*\*\*/

/\*--- TRIGGER CREADO POR MARÍA OLIVIA HANCZYC ---\*/

CREATE TRIGGER TR\_actualizarStockProductos

ON DetalleDeVenta AFTER INSERT AS

BEGIN

SET NOCOUNT ON

UPDATE Productos

SET Stock\_Prod = Stock\_Prod - (SELECT Cantidad\_Dv FROM INSERTED)

WHERE CodProducto\_Prod = (SELECT CodProducto\_Dv FROM INSERTED) AND

CUITProveedor\_Prod = (SELECT CUITProveedor\_Dv FROM INSERTED) and Stock\_Prod > 0

END

GO

/\*-- TRIGGER CREADO POR EZEQUIEL ALEJANDRO MARTíNEZ ---\*/

CREATE TRIGGER TR\_Productos\_PrevenirEliminar

ON Productos

INSTEAD OF DELETE

AS

BEGIN

PRINT('No está permitido eliminar registros de la tabla "Productos" mediante el uso de "Delete".');

ROLLBACK;

END;

go

/\*--- TRIGGER CREADO POR JAVIER ANDRÉS TORALES ---\*/

CREATE TRIGGER TR\_Proveedores\_PrevenirEliminar

ON Proveedores

INSTEAD OF DELETE

AS

BEGIN

PRINT('No está permitido eliminar registros de la tabla "Proveedores" mediante el uso de "Delete".');

ROLLBACK;

END;

go

---- Integrantes ----

select upper('Javier Andres Torales'), upper('María Olivia Hanczyc '),upper('Ezequiel Alejandro Martinez'),upper('Máximo Canedo'),upper('jean esquen')

--- CARGA DE DATOS ---

--- TABLA "EMPLEADOS"

INSERT INTO Empleados (DNI\_Em, Nombre\_Em, Apellido\_Em, Sexo\_Em, FechaDeNacimiento\_Em, FechaDeInicio\_Em, Sueldo\_Em, Direccion\_Em, Provincia\_Em, Localidad\_Em, Nacionalidad\_Em, Estado\_Em, Hash\_Em, Salt\_Em, Rol\_Em)

    -- Martínez, José Eduardo, empleado, clave "potato".

    SELECT '35001000 ', 'José Eduardo', 'Martínez', 'M', '1987-05-05', '2015-08-08', 140005.0000, 'Av. Dardo Rocha 3700', 'Buenos Aires', 'Tigre', 'Argentina', 1, 'qgIaxwvYzFgmGcdeclvkU98Ikqcq1afLWfUOuOrAeyU=', 'XptIldDnqJPQvgJ6YzMbng==', 'NORMAL' UNION

    -- Da Silva, Héctor, administrador, clave "potato".

    SELECT '35001001 ', 'Héctor', 'Da Silva', 'M', '1984-06-20', '2017-05-01', 540006.0000, 'Av. Lisboa 277', 'Santa Catarina', 'Forquilhinhas', 'Brasil', 1, 'IGHm07tKvD1BgLys6LC18QNJtfQYyiMddtL5Zb/zkOU=', 'DOpM4P3M93P5tuKih62IpQ==', 'ADMIN' UNION

    -- Mónaco, Libia, administradora, clave "potato".

    SELECT '35001002 ', 'Libia', 'Mónaco', 'F', '1977-08-24', '2015-08-10', 430000.0000, 'Chacabuco 471', 'Atacama', 'Copiapó', 'Chile', 1, '5uvjvvMTQfy0Y0/GAevNaNMJQSbQZuI85FG1vRhbVfo=', 'WWQlLau+76cwO1T2yJJRSA==', 'ADMIN' UNION

    -- Diet, María, empleada, clave "potato".

    SELECT '35001003 ', 'María', 'Diet', 'F', '1969-07-07', '2015-08-11', 460000.0000, 'Av. Manuel Belgrano 620', 'Tierra del Fuego', 'Río Grande', 'Argentina', 1, 'YOka37HQhJpe7RMbiRFSMdnTcLc7wN3bnYReya07xN4=', 'A67UnLoL1c+EqIn4i7FeGQ==', 'NORMAL' UNION

    -- Viera, Rodrigo, administrador, clave "potato".

    SELECT '35001004 ', 'Rodrigo', 'Viera', 'M', '1988-08-12', '2016-01-10', 360000.0000, 'Martín Carrera 420', 'Nuevo León', 'Monterrey', 'México', 0, 'Orgb2BAlu1rrcvTVG/Wo6McT5ReLoLdnNeoglDTVNtA=', 'zQEv0XKue6cH73kE0IxoRw==', 'ADMIN' UNION

    -- O Connor, Mary, empleada, clave "potato".

    SELECT '35001005 ', 'Mary', 'O Connor', 'F', '1965-12-12', '2016-02-15', 510300.0000, 'Stott Terrace 63', 'Northern Territory', 'Alice Springs', 'Australia', 1, 'N8ih6jqvD2rn1w/eQrKhKNDk8LsZnLLHQzegPBU6oC4=', 'gJwKfdjbnldtJqc48Wks0w==', 'NORMAL' UNION

    -- Pireira, Atima, empleada, clave "potato".

    SELECT '35001006 ', 'Atima', 'Pireira', 'F', '1991-09-11', '2015-07-30', 470000.0000, '8 Oguntuga St', 'Igbobi', 'Lagos', 'Nigeria', 1, 'JqB3G3yoJ9MdYo9HuwcM0tZhOM57ed3HDFpbeqaHRz4=', 'yRarIp6DGZWIb6kwJDtcgw==', 'NORMAL' UNION

    -- Jackson, Andrew, empleado, clave "potato".

    SELECT '35001007 ', 'Andrew', 'Jackson', 'M', '1980-01-05', '2015-08-20', 430550.0000, '5000 53 Ave.', 'Alberta', 'Warburg', 'Canadá', 1, '3y/yEkN/lNspHEsrwHnn6+BHlEqzN/yNVchXEz6aTZo=', '6WpSE7DK0luunBf3XsdWtg==', 'NORMAL' UNION

    -- Oliveira, Alejandro, empleado, clave "potato".

    SELECT '35001008 ', 'Alejandro', 'Oliveira', 'M', '1991-09-01', '2015-11-11', 380000.0000, 'Av. Portugal 8', 'Murcia', 'Lorca', 'España', 1, 'l23J2ZaVnST1wbrhN0Rw/r0cGemMR4gp1oKGian/po8=', 'LQnFaTooyioejjzRgbmU9A==', 'NORMAL' UNION

    -- Roderick Theodore, empleado, clave "potato".

    SELECT '35001009 ', 'Theodore', 'Roderick', 'M', '1985-10-10', '1985-10-10', 525800.0000, '4108 Kings Lane', 'Tennessee', 'Nashville', 'Estados Unidos', 0, 'krizqk0m5tZv/MF5FfoFYraeOr5ZCPIJO3jxCudWBoM=', 'gZyvrySPZDBhLzRqIigsrQ==', 'NORMAL' UNION

    -- Escudero, Inés, empleada, clave "potato".

    SELECT '35001010 ', 'Inés', 'Escudero', 'F', '1960-10-24', '1960-10-24', 489090.0000, 'Belgrano 1340', 'Buenos Aires', 'San Fernando', 'Argentina', 0, 'JpozyNv45dmyFiv+GotcB5k+ewaxJu4LxcYT7EcpO80=', 'IZR9x2eDOlH0Yqh2NXOyng==', 'NORMAL' UNION

    -- Marte, Alex, empleado, sin clave. \*

    SELECT '35007001 ', 'Alex', 'Marte', 'M', '1987-01-01', '2014-05-05', 340000.0000, 'Av. Waloyo Yamoni 1213', 'Buenos Aires', 'Tandil', 'Argentina', 1, '', '', 'NORMAL' UNION

    -- Dime, Dimmy, administrador, clave "potato".

    SELECT '45009001 ', 'Dimmy', 'Dime', 'M', '1990-01-05', '2015-05-01', 456009.0000, 'Washington 2040', 'Ciudad Autónoma de Buenos Aires', 'Belgrano R.', 'Argentina', 1, 'B6YFeo2luElLt+Zq9IQP/h88WR1wolPs4j/n6wif5uI=', 'rnAVCf9Mts1yxwnQNlJBXA==', 'ADMIN' UNION

    -- Herrera, Carlos, empleado, clave "potato".

    SELECT '75801800 ', 'Carlos', 'Herrera', 'M', '2000-12-12', '2014-04-14', 145000.0000, 'Av. Larralde 1250', 'Buenos Aires', 'Tigre', 'Argentina', 1, '6aLuqHe9vAObT/6zfFhzZia+n6jQU8lD2j95JHYFchs=', 'jabS49NEkyRzQLSUemuuUQ==', 'NORMAL' UNION

-- Observaciones:

-- - Los últimos tres registros fueron creados mediante la plataforma.

-- - Los hashes y salts fueron generados automáticamente mediante la plataforma. No se ingresan a mano, sino que para esta consulta se copiaron y pegaron sus valores, pero a la hora de insertarlos por primera vez, se dejaban estos campos en blancos para luego ser rellenados.

-- - En el caso del DNI 35.007.001 (Marte, Alex) nunca se le generó una clave dado que lo utilicé en el TP de programación para realizar algunas pruebas de seguridad.

--- TABLA "SESIONES"

INSERT INTO Sesiones (CodSesion\_Se, DNIEmpleado\_Se, FechaDeAlta\_Se, Token\_Se, Estado\_Se)

    SELECT NULL, '35001001', '2023-06-29 01:32:11.600', 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkbmkiOiIzNTAwMTAwMSIsImZlIjoiMjkvMDYvMjAyMyAxOjMyOjExIiwiZXhwIjoxNjg4MDQxOTMxLCJpc3MiOiJUUEkiLCJhdWQiOiJQUk9HIn0.6uryrfUmXki2GwpMZYusCghWWZVWLZgpRQpr21ClqEQ', 1

-- Observaciones:

-- - Esta tabla nunca se rellenó manualmente, está pensada para uso interno del TP de programación. El insert es para dar una idea del tipo de registros que contiene.

-- - CodSesion\_Se es autonumérico, por eso se inserta con valor NULL.

-- - Token\_Se contiene un token generado por la plataforma una vez que el usuario inicia sesión. Se usa para autenticarlo en el futuro.

-- - Estado\_Se es un bit que indica si el token está activo, o si fue deshabilitado por el usuario.