UNIVERSIDAD ANDINA DEL CUSCO

FACULTAD DE INGENIERÍA Y ARQUITECTURA
ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

TRABAJO GRUPAL

Entregable "Sistema Ventas"

ASIGNATURA : DESARROLLO DE PLATAFORMA DE SOFTWARE

DOCENTE : ESPETIA HUAMANGA, Hugo

INTEGRANTES : SALIZAR ROZAS, Max

ESCOBAR BANDA, Alessandro

PACURI CARRIÓN, Shavely

CRUZ OLIVARES, Javier

CUSCO – PERÚ

2020

PRESENTACIÓN

En este presente trabajo realizaremos la continuación del trabajo en clase de un "Sistema de Ventas" que consta de cinco tablas Cliente que permite ingresar al cliente con usuario y contraseña, Vendedor: el cual le permitirá al vendedor ver los productos por categoría, emitir la boleta y tener en un formato de PDF y Excel todos los datos el reporte completo de ventas, ,Producto: en el cual irá todos los los detalles de cada producto, Categoría: tendrá la categoría de cada producto y Boleta: la cual emitirá el reporte de boleta.

La plataforma virtual se realizó en Visual Studio utilizando el lenguaje de C# y para la parte de diseño usamos Bootstrap.

DESARROLLO

1. ENCRIPTACIÓN MEDIANTE SHA-1

Se implementó el encriptado mediante el algoritmo SHA-1 el cual es un miembro de la familia SHA que constituye sus diversas versiones.

Para ello usamos de referencia una serie de códigos provistos en las sesiones de clases previas. Dicho código funciona sobre el botón de login.

```
using System.Security.Cryptography;
```

esta librería es esencial para el funcionamiento de la encriptación

```
1 referencia
private string generarClaveSHA1(string cadena)
{

    UTF8Encoding enc = new UTF8Encoding();
    byte[] data = enc.GetBytes(cadena);
    byte[] result;

    SHA1CryptoServiceProvider sha = new SHA1CryptoServiceProvider();

    result = sha.ComputeHash(data);


    StringBuilder sb = new StringBuilder();
    for (int i = 0; i < result.Length; i++)
    {

        // Convertimos los valores en hexadecimal
        // cuando tiene una cifra hay que rellenarlo con cero
        // para que siempre ocupen dos dígitos.
        if (result[i] < 16)
        {
            sb.Append("0");
        }
        sb.Append(result[i].ToString("x"));
    }

    //Devolvemos la cadena con el hash en mayúsculas para que quede más chuli :)
    return sb.ToString().ToUpper();
}

0 referencias
protected void Login1_Authenticate(object sender, AuthenticateEventArgs e)
{
    string contrasena = Login1.Password;
    Login1.FailureText = generarClaveSHA1(contrasena);
}
```

Estas líneas de código nos permite encriptar una cadena de texto que en este caso es la contraseña del usuario. Estas líneas de código acompañadas de código SQL como se ve a continuación, proveen una encriptación total y por lo tanto, una mayor seguridad a la información de nuestros usuarios.

```
Create Procedure LoginUsuario
    @Name nvarchar(50),
    @Pass nvarchar(50),
    @Result bit Output
As
    Declare @PassEncode As nvarchar(300)
    Declare @PassDecode As nvarchar(50)
Begin
    Select @PassEncode = Pass From Login Where Name = @Name
    Set @PassDecode = DECRYPTBYPASSPHRASE('password', @PassEncode)
End

Begin
    If @PassDecode = @Pass
        Set @Result = 1
    Else
        Set @Result = 0
End

Go
```

Imagen obtenida de: http://www.7sabores.com/blog/encriptar-datos-sql-server

Los resultados son:

CodCliente	Apellidos	Nombres	Direccion	Usuario	Contrasena
C001	JIMENEZ AGUILAR	JORGE	AV SOL 234	jimenez@gmail.com	40BD001563085FC35165329EA1FF5C5ECBDBBEEF
C002	PERALTA ARAOZ	MIRIAM	URB VISTA ALEGRE J-34	peralta1@gmail.com	5F6955D227A320C7F1F6C7DA2A6D96A851A8118F

En caso de que hubiese algún problema con la contraseña del usuario, este puede acceder a un cambio o recuperación mediante su "llave". De otro modo, no habría manera de recuperarlo.

2. IMPLEMENTACIÓN DE MANTENIMIENTO DE LAS TABLAS

Se implementó el mantenimiento de las cuatro tablas, en cada una de ellas se implementó los procedimientos almacenados: Listar, Actualizar, Agregar, Eliminar, Buscar y en el caso de la Tabla Cliente y la de Vendedor incluimos el procedimiento almacenado Actualizar Contraseña.

- CLIENTE

Listar

```
--ListarVendedor
if OBJECT_ID('spListarCliente') IS NOT NULL
    drop proc spListarCliente
go

create proc spListarCliente
as
begin
    select Apellidos + '' + Nombres as Datos, Usuario from TVendedor
end
go

exec spListarCliente
go
```

Login Cliente

```
if OBJECT_ID('spLoginCliente') is not null
    drop proc spLoginCliente
go
create proc spLoginCliente
@Usuario varchar(50),@contrasena varchar(50)
as
begin
    if exists(select Usuario from TCliente where Usuario=@Usuario)
        if exists(select Contrasena from TCliente where Usuario=@Usuario and Contrasena=@contrasena)
            begin
                declare @DatosCliente varchar(50)
                set @DatosCliente =(select Apellidos+' '+Nombres from TCliente where Usuario=@Usuario
                and Contrasena=@contrasena)
                if exists(select  Usuario from TCliente where Usuario=@Usuario
                and Contrasena=@contrasena)
                select CodError=0, Mensaje=@DatosCliente
            end
        else select CodError=1, Mensaje='Error: Usuario y/o Contrasenas incorrectas'
    else select CodError=1,Mensaje='Error: Usuario no existe en la base de datos'
end
go

exec spLoginCliente 'jimenez@gmail.com','123'
go
```

Agregar

```
48 if OBJECT_ID('spAgregarCliente') is not null
49     drop proc spAgregarCliente
50  go
51
52 create proc spAgregarCliente
53  @CodCliente varchar(4),
54  @Apellidos varchar(30),
55  @Nombres varchar(30),
56  @Direccion varchar(50),
57  @Usuarios varchar(50),
58  @Contrasena varchar(200)
59  as
60 begin
61      ---Que el usuario no existe
62      if not exists(select Usuario from TCliente where Usuario = @Usuarios)
63      begin
64          insert into TCliente values (@CodCliente,@Apellidos,@Nombres,@Direccion,
65          @Usuarios,cast(@Contrasena as varbinary(200)))
66          select CodError = 0, Mensaje = 'Usuario insertado correctamente'
67      end
68      else
69          select CodError = 1, Mensaje = 'Error: El usuario ya existe'
70  end
71  go
72
73
74 exec spAgregarCliente 'C011','A','N','D','a@gmail.com','40BD001563085FC35165329EA1FF5C5ECBDBBEEF'
```

Actualizar

```
80  --PA para Modificar Cliente
81 if OBJECT_ID('spActualizarCliente') is not null
82     drop proc spActualizarCliente
83  go
84 create proc spActualizarCliente
85  @CodCliente varchar(30),
86  @Apellidos varchar(50),
87  @Nombres varchar(50)
88
89  as
90 begin
91      --Validar CodCliente
92      if exists(select CodCliente from TCliente where CodCliente = @CodCliente)
93
94
95          begin
96              begin try
97              update TCliente set Apellidos=@Apellidos where CodCliente=@CodCliente
98              update TCliente set Nombres=@Nombres where CodCliente=@CodCliente
99              select CodError = 0, Mensaje= 'Cliente actualizado con exito'
100             end try
101             begin catch
102                 select CodError = 1, Mensaje= 'Error, no se pudo actualizar cliente'
103             end catch
104         end
105
106     else select CodError=1, Mensaje='Error: CodCliente no existe'
107 end
108  go
109
```

Eliminar

```
112  --PA para Eliminar Cliente
113 if OBJECT_ID('spEliminarCliente') is not null
114      drop proc spEliminarCliente
115  go
116 create proc spEliminarCliente
117  @CodCliente varchar(30)
118  as
119 begin
120      --Validar CodCliente
121      if exists(select CodCliente from TCliente where CodCliente = @CodCliente)
122          begin
123              begin try
124              delete from TCliente where CodCliente=@CodCliente
125              select CodError = 0, Mensaje= 'Cliente Eliminado con exito'
126              end try
127              begin catch
128              select CodError = 1, Mensaje= 'Error, no se pudo Eliminar cliente'
129              end catch
130          end
131      else select CodError=1, Mensaje='Error: CodCliente no existe'
132  end
133  go
134
135  execute spEliminarCliente 'C011'
136  go
137
```

Buscar

```
138  --PA para buscar Cliente
139 if OBJECT_ID('spBuscarCliente') is not null
140      drop proc spBuscarCliente
141  go
142 create proc spBuscarCliente
143  @Texto varchar(50), @Criterio varchar(20)
144  as
145 begin
146      begin try
147
148          if(@Criterio ='CodCliente')
149              select CodCliente, Usuario from TCliente where CodCliente = @Texto
150          else if(@Criterio ='Apellidos')
151              select Apellidos, CodCliente, Usuario from TCliente where Apellidos LIKE '%'+@Texto+'%'
152          else if(@Criterio ='Usuario')
153              select Usuario, CodCliente from TCliente where Usuario =@Texto
154          else select CodError = 0, Mensaje= 'Cliente Encontrado'
155      end try
156      begin catch
157      select CodError = 1, Mensaje= 'Error, no se encontro'
158      end catch
159  end
160  go
161
162  execute spBuscarCliente 'jimenez@gmail.com','Usuario'
163  go
164
```

Actualizar Contraseña

```
167 if OBJECT_ID('spActualizarContrasenaCliente') is not null
168     drop proc spActualizarContrasenaCliente
169  go
170
171 create proc spActualizarContrasenaCliente
172  (
173     @Usuario varchar(30),
174     @Contrasena varchar(200),
175     @ContrasenaNueva varchar(200)
176  )
177  as
178 begin
179     --Validar que el usuario exista
180     if exists(select * from TCliente where Usuario = @Usuario)
181     begin
182     --Validar la contrasena correcta
183         if exists(select * from TCliente where Contrasena = @Contrasena and Usuario = @Usuario)
184         begin
185             --Validad que la contrasena no sea la misma
186             if not exists(select * from TCliente where Contrasena = @ContrasenaNueva and Usuario = @Usuario)
187             BEGIN
188                 update TCliente set Contrasena=cast(@ContrasenaNueva as varbinary(200)) where Usuario=@Usuario
189                 select CodError = 0, Mensaje= 'Contrasena Cliente Actualizada'
190             END
191             else select CodError = 1, Mensaje= 'La contrasena no puede ser la misma que la anterior'
192         end
193         else select CodError = 1, Mensaje= 'Error: La cotrasena no es la misma'
194     end
195     else select CodError = 1, Mensaje= 'Error: No existe el usuario'
196 end
197  go
```

- VENDEDOR

Agregar

```
 8   --AgregarVendedor
 9  if OBJECT_ID('spAgregarVendedor') IS NOT NULL
10      drop proc spAgregarVendedor
11   go
12
13  create proc spAgregarVendedor
14   @CodVendedor varchar(4), @Apellidos varchar(30),
15   @Nombres varchar(30), @Usuario varchar(50),
16   @Contrasena varchar(50)
17   as
18  begin
19      --1.- CodVendedor no existe
20      if not exists(select CodVendedor from TVendedor where CodVendedor=@CodVendedor)
21          --2.- Usuario no existe
22          if not exists(select Usuario from TVendedor where Usuario=@Usuario)
23              begin
24                  insert into TVendedor values(@CodVendedor,@Apellidos,@Nombres,
25                  @Usuario, @Contrasena)
26                  select CodError = 0, Mensaje = 'Insertado Correctamente Vendedor'
27              end
28          else select CodError = 1, Mensaje = 'ERROR: Usuario ya Existe'
29      else select CodError = 1, Mensaje = 'ERROR: Usuario ya Existe'
30   end
31   go
32
33   exec spAgregarVendedor 'V005','PACURI CARRION','SHAVELY','shavely@gmail.com','123'
```

Listar

```
--ListarVendedor
if OBJECT_ID('spListarVendedor') IS NOT NULL
    drop proc spListarVendedor
go

create proc spListarVendedor
as
begin
    select CodVendedor, Apellidos + '' + Nombres as Datos, Usuario from TVendedor
end
go

exec spListarVendedor
go
```

Eliminar

```
--EliminarVendedor
if OBJECT_ID('EliminarVendedor')is not null
    drop proc spEliminarVendedor
go

create proc spEliminarVendedor
@CodVendedor varchar (4)
as
begin
    --Validar que el codVendedor exista
    if exists(select CodVendedor from TVendedor where CodVendedor = @CodVendedor)
        begin
            declare @Usuario varchar (50)
            set @Usuario = (select Usuario from TVendedor where CodVendedor=@CodVendedor)
            begin
                begin tran TransEliminar
                begin try
                    delete from TVendedor where CodVendedor = @CodVendedor
                    delete from TVendedor where Usuario = @Usuario
                    commit tran TransEliminar
                    select CodError = 0, Mensaje = 'Vendedor Eliminado Correctamente'
                end try
                begin catch
                    rollback tran TransEliminar
                    select CodError = 1, Mensaje = 'ERROR: Problemas en la Transaccion'
                end catch
            end
        end
    else select CodError = 1, Mensaje = 'ERROR: Codigo de Vendedor no existe'
end
go

execute spEliminarVendedor 'V005'
```

Actualizar

```
------------PA actualizar Vendedor
if OBJECT_ID('spActualizarVendedor') is not null
    drop proc spActualizarVendedor
go
create proc spActualizarVendedor
@CodVendedor varchar(4),
@Apellidos varchar(50),
@Nombres varchar(50),
@Usuario varchar(50),
@Contrasena varchar(50)
as
begin
    --Validar que el codigo Vendedor Exista
    if exists (select CodVendedor from TVendedor where CodVendedor = @CodVendedor)
    begin

        begin
            begin tran TransActualizar
            begin try
                update TVendedor set Apellidos = @Apellidos, Nombres = @Nombres,
                Usuario = @Usuario, Contrasena = @Contrasena where CodVendedor = @CodVendedor

                commit tran TransActualizar
                select CodError = 0, Mensaje = 'Vendedor Actualizado'
            end try
            begin catch
                rollback tran TransActualizar
                select CodError = 1, Mensaje = 'Error:Problemas en la transaccion'
            end catch
        end
    end
    else select CodError = 1, Mensaje = 'Error:Cod Vendedor no existe'
end
```

Buscar

```
-- Buscar en la tabla Vendedor
if OBJECT_ID('spBuscarVendedor') is not null
    drop proc spBuscarVendedor
go

create proc spBuscarVendedor
@Texto varchar(50), @Criterio varchar(10)
as
begin
    if(@Criterio = 'CodVendedor')--Busqueda exacta
        select CodVendedor,Apellidos,Nombres,Usuario,Contrasena from TVendedor where CodVendedor = @Texto
    else if(@Criterio = 'Nombre')
        select CodVendedor,Apellidos,Nombres,Usuario,Contrasena from TVendedor where (Apellidos LIKE '%' + @Texto + '%')
                                                                                  or (Nombres LIKE '%' + @Texto + '%')
                                                                                  or (Usuario LIKE '%' + @Texto + '%')
end
go

exec spBuscarVendedor 'V005','CodVendedor'
go
```

- **CATEGORÍA**

Agregar

```
if OBJECT_ID('spAgregarCategoria') is not null
    drop proc spAgregarCategoria
go

create proc spAgregarCategoria
(
    @Cod varchar(4),
    @Nombre varchar(50),
    @CatPadre varchar(4)
)
as
begin
    --Validar que no exista la categoria
    if not exists (select CodCategoria from TCategoria where CodCategoria = @Cod)
    begin
        --Validar que la categoria padre exista
        if exists(select CodCategoria from TCategoria where CodCategoria = @CatPadre or @CatPadre = NULL)
        begin
            begin tran TransAgregar -- Iniciar la transaccion
            begin try
                insert into TCategoria values(@Cod,@Nombre,@CatPadre)
                commit tran TransAgregar
                select CodError = 0, Mensaje = 'Categoria Agregada'
            end try
            begin catch
                rollback tran TransAgregar
                select CodError = 1, Mensaje = 'Error:Problemas en la transaccion'
            end catch
        end
            else select codError = 1,Mensaje = 'Error: la categoria padre no existe'
    end
        else select codError = 1,Mensaje = 'Error: la categoria ya existe'
end
```

Listar

```
---Listar Categoria
if OBJECT_ID('spListarCategoria') is not null
    drop proc spListarCategoria
go

create proc spListarCategoria
as
begin
    select CodCategoria,Nombre,CategoriaPadre from TCategoria
end
go

exec spListarCategoria
go
```

Eliminar

```
if OBJECT_ID('spEliminarCategoria') is not null
    drop proc spEliminarCategoria
go

create proc spEliminarCategoria
(
    @Cod varchar(4)
)
as
begin
    --Verificar que exista la categoria
    if exists(select CodCategoria from TCategoria where CodCategoria = @Cod)
    BEGIN
        --Verificar que no sea categoria padre
        if not exists(select CategoriaPadre from TCategoria where CategoriaPadre = @Cod)
        begin
            begin tran TransEliminar -- Iniciar la transaccion
            begin try
                delete from TCategoria where CodCategoria=@Cod
                commit tran TransEliminar
                select CodError = 0, Mensaje = 'Categoria Eliminado'
            end try

            begin catch
                rollback tran TransEliminar
                select CodError = 1, Mensaje = 'Error:Problemas en la transaccion'
            end catch
        end
        else select codError = 1,Mensaje = 'Error: La categria es padre'
    END
    else select codError = 1,Mensaje = 'Error: La categoria no existe'
end
go
```

Actualizar

```
if OBJECT_ID('spActualizarCategoria') is not null
    drop proc spActualizarCategoria
go

create proc spActualizarCategoria
(
    @Cod varchar(4),
    @Nombre varchar(50),
    @CatPadre varchar(4)
)
as
begin
    --Verificar que la categoria Exista
    if exists(select CodCategoria from TCategoria where CodCategoria = @Cod)
    BEGIN
        begin tran TransActualizar
            begin try
                update TCategoria set Nombre = @Nombre, CategoriaPadre = @CatPadre where CodCategoria=@Cod

                commit tran TransActualizar
                select CodError = 0, Mensaje = 'Categoria actualizada'
            end try

            begin catch



                rollback tran TransActualizar
                select CodError = 1, Mensaje = 'Error:Problemas en la transaccion'

            end catch
    END
    else select codError = 1,Mensaje = 'Error: La categoria no existe'
```

Buscar

```
---Buscar Categoria
if OBJECT_ID('spBuscarCategoria') is not null
    drop proc spBuscarCategoria
go

create proc spBuscarCategoria
@Texto varchar(50), @Criterio varchar(30)
as
begin
    if(@Criterio = 'CodCategoria')
        select CodCategoria,Nombre,CategoriaPadre from TCategoria where CodCategoria = @Texto
    else if(@Criterio = 'Nombre')
        select CodCategoria,Nombre,CategoriaPadre from TCategoria where Nombre Like '%'+@Texto+'%'
end
go

/**exec spBuscarCategoria 'al','Nombre'
go
exec spBuscarCategoria 'C003','CodCategoria'
go*/
```

- **PRODUCTO**

Listar

```
---Listar Producto
if OBJECT_ID('spListarProducto') is not null
    drop proc spListarProducto
go

create proc spListarProducto
as
begin
    select CodProducto,Nombre,UnidadMedida,Precio,Stock,CodCategoria from TProducto
end
go

exec spListarProducto
go
```

Agregar

```
if OBJECT_ID('spAgregarProducto') is not null
    drop proc spAgregarProducto
go

create proc spAgregarProducto
(
    @CodProducto varchar(4),
    @Nombre varchar(50),
    @UnidadMedida varchar(30),
    @Precio real,
    @Stock int,
    @CodCategoria varchar(4)
)
as
begin
    --Validar que el codProducto no se repita
    if not exists(select CodProducto from TProducto where CodProducto = @CodProducto)
    begin
        --Validar que exista su categoria
        if exists(select CodCategoria from TCategoria where CodCategoria = @CodCategoria)
        begin
            begin tran TransAgregar -- Iniciar la transaccion
            begin try
                insert into TProducto values(@CodProducto,@Nombre,@UnidadMedida,@Precio,@Stock,@CodCategoria)
                commit tran TransAgregar
                select CodError = 0, Mensaje = 'Porducto Agregado'
            end try
            begin catch
                rollback tran TransAgregar
                select CodError = 1, Mensaje = 'Error:Problemas en la transaccion'
            end catch
```

Eliminar

```
--Eliminar Producto
if OBJECT_ID('spEliminarProducto') is not null
    drop proc spEliminarProducto
go
create proc spEliminarProducto
(
    @CodProducto varchar(4)
)
as
begin
    --Verificar que el producto exista
    if exists(select * from TProducto where CodProducto = @CodProducto)
    begin
        begin tran TransEliminar -- Iniciar la transaccion
            begin try
                delete from TProducto where CodProducto = @CodProducto
                commit tran TransEliminar
                select CodError = 0, Mensaje = 'Producto Eliminado'
            end try

            begin catch
                rollback tran TransEliminar
                select CodError = 1, Mensaje = 'Error:Problemas en la transaccion'
            end catch
    end
    else select codError = 1,Mensaje = 'Error: El producto no existe'
end
go
```

Actualizar

```
--Actualizar Producto
 if OBJECT_ID('spActualizarProducto') is not null
    drop proc spActualizarProducto
go

create proc spActualizarProducto
(
    @CodProducto varchar(4),
    @Nombre varchar(50)
)
as
begin
    --Validar que exista el producto
    if exists(select * from TProducto where CodProducto = @CodProducto)
    begin
        begin tran TransActualizar
            begin try
                update TProducto set Nombre = @Nombre where CodProducto=@CodProducto
                commit tran TransActualizar
                select CodError = 0, Mensaje = 'Producto actualizado'
            end try

            begin catch
                rollback tran TransActualizar
                select CodError = 1, Mensaje = 'Error:Problemas en la transaccion'

            end catch
    end
    else select codError = 1,Mensaje = 'Error: El producto no existe'
end
go
```

Buscar

```
---Buscar Producto
if OBJECT_ID('spBuscarProducto') is not null
    drop proc spBuscarProducto
go

create proc spBuscarProducto
@Texto varchar(50), @Criterio varchar(30)
as
begin
    if(@Criterio = 'CodProducto')
        select CodProducto,Nombre,UnidadMedida,Precio,Stock,CodCategoria from TProducto where CodProducto = @Texto
    else if(@Criterio = 'Nombre')
        select CodProducto,Nombre,UnidadMedida,Precio,Stock,CodCategoria from TProducto where Nombre Like '%'+@Texto+'%'
end
go
```

3. IMPLEMENTACIÓN DEL PROCESO BOLETA

4. IMPLEMENTAR EL CAMPO DE CAMBIAR CONTRASEÑA EN LA TABLA DE CLIENTE Y VENDEDOR EN EL CUAL USAMOS CONTROLES DE VALIDACIÓN

Para el cambio de contraseña de la Tabla Cliente y Vendedor usamos un procedimiento almacenado Actualizar Contraseña el cual nos permite cambiar la contraseña del Cliente y del Vendedor.

```
167 if OBJECT_ID('spActualizarContrasenaCliente') is not null
168     drop proc spActualizarContrasenaCliente
169  go
170
171 create proc spActualizarContrasenaCliente
172  (
173     @Usuario varchar(30),
174     @Contrasena varchar(200),
175     @ContrasenaNueva varchar(200)
176  )
177  as
178 begin
179     --Validar que el usuario exista
180     if exists(select * from TCliente where Usuario = @Usuario)
181     begin
182     --Validar la contrasena correcta
183         if exists(select * from TCliente where Contrasena = @Contrasena and Usuario = @Usuario)
184         begin
185             --Validad que la contrasena no sea la misma
186             if not exists(select * from TCliente where Contrasena = @ContrasenaNueva and Usuario = @Usuario)
187             BEGIN
188                 update TCliente set Contrasena=cast(@ContrasenaNueva as varbinary(200)) where Usuario=@Usuario
189                 select CodError = 0, Mensaje= 'Contrasena Cliente Actualizada'
190             END
191             else select CodError = 1, Mensaje= 'La contrasena no puede ser la misma que la anterior'
192         end
193         else select CodError = 1, Mensaje= 'Error: La cotrasena no es la misma'
194     end
195     else select CodError = 1, Mensaje= 'Error: No existe el usuario'
196 end
197  go
```

5. **IMPLEMENTAR EN EL SISTEMA PARA EL VENDEDOR VER EL REPORTE DE SUS VENTAS POR SEMANA EL CUAL SE EXPORTAR A EXCEL Y PDF.**

6. **DISEÑO**

Aquí hicimos uso de Jquery para poder agregar animación y contextura a las los web form y bootstrap para los distintos elementos trabajados dentro de los mismos.

Por parte de los Jquery, trabajamos de la mano con CSS para poder darle mayor estética al web form.

CONCLUSIONES

- Gracias a este trabajo pudimos ver como el manejo adecuado de contraseñas mediante la encriptación se traduce en una seguridad robusta y bien estructurada a la hora de gestionar usuarios.
- Asimismo, el cambio de contraseña mediante un formulario y no mediante base de datos, aumenta la seguridad puesto que es solo el usuario dueño de dicha contraseña quien la conoce.
- Realizar este tipo de Proyectos nos capacita como estudiantes para que en un futuro podamos responder a las necesidades profesionales gracias al estudio y preparación como Ingenieros de Sistemas.
- El uso de bootstrap y jQuery para el diseño supuso un reto a la hora de encontrar información y aplicarla sobre el proyecto.

SUGERENCIAS

- El manejo del diseño mediante bootstrap y/o Jquery simplifica mucho el trabajo de diseño cuando se aplica bien, por lo que en la medida que sea posible su uso es recomendable.
- Siempre tener en mente y no olvidar resguardar la integridad de nuestros usuarios, para ello el cuidar sus datos es fundamental por lo que la encriptación en todos sus niveles debe trabajarse siempre en los sistemas con información sensible.

REFERENCIAS BIBLIOGRÁFICAS

Animated Login Form in Asp.net Tutorial. (2017, Septiembre 28). Animated Login. https://www.youtube.com/watch?v=-UE4yrt5xWI

Robayo, D. (2017, Julio 05). *Estilos con Bootstrap para ASP.NET Web Form*. Estilos con Bootstrap. https://www.youtube.com/watch?v=jwXKCUVzEfw

Sitio Web de Login y Registro con ASP.NET. (2018, Abril 30). Registro con ASP.NET. https://www.youtube.com/watch?v=UX5J-uZdTmg