Obligatorio 1 PortLog - Programación 3
Grupo N3B
2020

José Ignacio Rossi Faval
237974

Valentin Barreneche
218588

Tabla de Contenido

Requerimientos Funcionales

Se requiere la creación de un sistema en el cual la compañía de importaciones PortLog puede llevar un registro de sus Clientes, Productos, e Importaciones almacenadas en sus almacenes, hacer una previsión de la ganancia que cada uno de sus Clientes le generará, y generar archivos con sus registros.

El Sistema contará con dos tipos de usuarios, “admin” y “deposito”, que podrán acceder al sistema con sus credenciales y cerrar sesión. Cuenta con las siguientes funcionalidades:

-Alta de nuevos Usuarios por parte de los usuarios de tipo “admin”, de los cuales se conoce su cedula, su contraseña (la cual debe tener mínimo 6, incluir un número, una mayúscula y una minúscula), y su rol (“admin” o “deposito”).

-Alta de nuevos Clientes por ambos tipo de usuarios, de los cuales se conoce su nombre, su RUT (número único de 12 dígitos), y la fecha en que comenzaron su relación con PortLog.

-Alta de nuevos Productos por ambos tipos de usuario, de los que se conoce un código, su nombre, el peso por unidad y el cliente que lo importa.

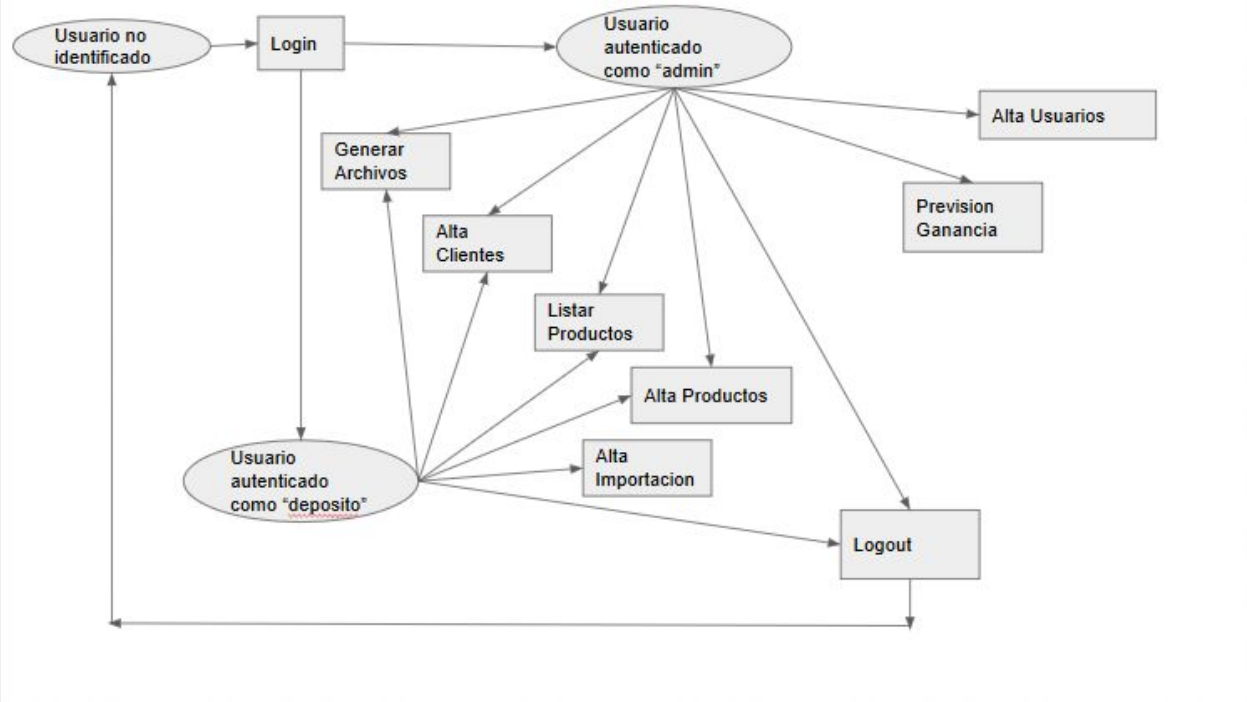
-Listar todos los productos por ambos tipos de usuario a través de un WCF.

-Alta de nuevas Importaciones por los usuarios de tipo “deposito” a través de un WCF. De estas importaciones se conoce la fecha de ingreso al depósito y su fecha de salida prevista, el producto importado, la cantidad, y el precio por unidad de producto.

-Prevision de la Ganancia por Cliente por los usuarios de tipo “admin”. El calculo se realiza según el monto total de la mercancía que PortLog tiene almacenada al momento para un cliente en particular, considerando la cantidad de días transcurridos entre la fecha en que fue ingresada y la fecha prevista de salida. PortLog cobra un porcentaje diario, de ese monto. Los clientes que tengan más de una determinada cantidad de años de antigüedad se benefician con un porcentaje de descuento.

-Generación de Archivos con los registros de todos los Usuarios, Clientes, Importaciones, Descuentos, y Productos del sistema.

Diagrama con Casos de Uso



Casos de Uso Narrativos

Hacer Login

Descripcion: Permite a un Usuario No-Autenticado hacer login con sus credenciales y acceder al sistema, el cual mostrará distintas funcionalidades dependiendo del rol del usuario autenticado.

Actores: Usuario No-Autenticado

Precondiciones: El Usuario está en la página de Inicio de Sesión.

Flujo Normal:

1. El actor llena el campo Usuario.
2. El actor llena el campo Password.
3. El sistema verifica la existencia del usuario y su contraseña.

Flujo/s Alternativo/s:

1. El actor ingresa un nombre de usuario no existente.
1.1 Se muestra el mensaje de error acorde.

2. El actor ingresa una contraseña que no es válida para el nombre de usuario.
2.1 Se muestra el mensaje de error acorde.

Flujo/s Excepcional/es:

La comunicación con la Base de Datos se corta durante la llamada. Se muestra un error.

Pos Condiciones

El Usuario es llevado a la página de Inicio de Sesión, donde se muestran las funcionalidades de acuerdo a su rol.

Hacer Logout

Descripcion: Permite a un Usuario Autenticado cerrar su sesión.

Actores: Cualquier Usuario Autenticado.

Precondiciones: El Usuario está autenticado.

Flujo Normal:

1. El Usuario hace click en el botón Cerrar Sesión.

Pos Condiciones

El Usuario es llevado a la página de Login.

Alta Usuario

Descripcion: Permite a un Usuario Autenticado con rol de “admin” crear nuevos usuarios.

Actores: Usuario Autenticado con rol “admin”.

Precondiciones: El Usuario “admin” está loggeado.

Flujo Normal:

1. El Usuario hace click en Alta Usuario y es llevado a la vista correspondiente.
2. El usuario llena el campo cédula.
3. El usuario llena el campo password.
4. El usuario elige un rol para el nuevo usuario.
5. El usuario hace click en el botón Crear.

Flujo/s Alternativo/s:

1. El usuario ingresa una cedula de menos de 6 o más de 8 dígitos .
1.1 Se muestra el mensaje de error acorde.

2. El actor ingresa una contraseña que no tiene al menos 6 caracteres, una minúscula, una mayúscula, y un número.
2.1 Se muestra el mensaje de error acorde.

3. El actor no solecciona un rol.
3.1 Se muestra el mensaje de error acorde.

4. El usuario ingresa la cedula de un usuario ya existente.
4.1 Se muestra el mensaje de error acorde.

Flujo/s Excepcional/es:

La comunicación con la Base de Datos se corta durante la llamada. Se muestra un error.

Pos Condiciones

El nuevo usuario es creado con éxito. Se muestra un mensaje de éxito.

Alta Cliente

Descripcion: Permite a un Usuario Autenticado crear nuevos Clientes.

Actores: Cualquier usuario autenticado.

Precondiciones: El Usuario está loggeado.

Flujo Normal:

1. El Usuario hace click en Alta Cliente y es llevado a la vista correspondiente.
2. El usuario llena el campo Rut.
3. El usuario llena el Nombre.
4. El usuario llena el campo Fecha de Comienzo de Relación.
5. El usuario hace click en el botón Crear.

Flujo/s Alternativo/s:

1. El usuario ingresa un rut de menos de 12 caracteres.
1.1 Se muestra el mensaje de error acorde.

2. El usuario ingresa un rut de un cliente que ya existe.
2.1 Se muestra el mensaje de error acorde.

3. El usuario no llena el campo nombre.
3.1 Se muestra el mensaje de error acorde.

4. El Usuario no llena el campo Fecha de Comienzo de Relación.
4.1 Se muestra el mensaje de error acorde.

Flujo/s Excepcional/es:

La comunicación con la Base de Datos se corta durante la llamada. Se muestra un error.

Pos Condiciones

El nuevo Cliente es creado con éxito. Se muestra un mensaje de éxito.

Alta Producto

Descripcion: Permite a un Usuario Autenticado crear nuevos Productos.

Actores: Cualquier usuario autenticado.

Precondiciones: El Usuario está loggeado.

Flujo Normal:

1. El Usuario hace click en Alta Producto y es llevado a la vista correspondiente.
2. El usuario llena el Peso por Unidad del Producto.
3. El usuario llena el campo Nombre.
4. El usuario elige el cliente que importa ese producto.
5. El usuario hace click en el botón Crear.

Flujo/s Alternativo/s:

1. El usuario ingresa un Peso con valor 0 o vacio..
1.1 Se muestra el mensaje de error acorde.

2. El usuario ingresa el campo Nombre de un producto en vacio.
2.1 Se muestra el mensaje de error acorde.

Flujo/s Excepcional/es:

La comunicación con la Base de Datos se corta durante la llamada. Se muestra un error.

Pos Condiciones

El nuevo Producto es creado con éxito. Se muestra un mensaje de éxito.

Listar Productos

Descripcion: Permite a un Usuario Autenticado listar nuevos Productos.

Actores: Cualquier usuario autenticado.

Precondiciones: El Usuario está loggeado y el servicio WCF debe estar levantado.

Flujo Normal:

1. El Usuario hace click en Listar Productos y es llevado a la vista correspondiente.

Flujo/s Excepcional/es:

1. La comunicación con la Base de Datos se corta durante la llamada. Se muestra un error.
2. La comunicación con el servicio WCF está inactiva. Se muestra un error.

Pos Condiciones

El sistema lista los Productos de acuerdo a los datos correspondiente en la Base de Datos.

Alta Importacion

Descripcion: Permite a un Usuario con rol “deposito” generar nuevas Importaciones de un Producto.

Actores: Usuario Autenticado con rol “deposito”.

Precondiciones: El Usuario con rol “deposito” está loggeado y en la página de Listar Productos.

Flujo Normal:

1. El Usuario hace click en el botón “Hacer Importación” del Producto correspondiente y es llevado a la vista de Alta Importacion.
2. El Usuario ingresa la Fecha de Ingreso al deposito.
3. El Usuario ingresa la Fecha de Salida Prevista.
4. El Usuario ingresa la cantidad de Producto importado.
5. El Usuario ingresa el Precio Por Unidad del Producto importado.

Flujo/s Alternativo/s:

1. El Usuario no ingresa una Fecha de Ingreso.
1.1 Se muestra el mensaje de error acorde.

2. El usuario no ingresa una Fecha de Salida Prevista.
2.1 Se muestra el mensaje de error acorde.

3. El Usuario no ingresa Cantidad del Producto importado.
3.1 Se muestra el mensaje de error acorde.

4. El Usuario no ingresa Precio Por Unidad del Producto importado.
4.1 Se muestra el mensaje de error acorde.

Flujo/s Excepcional/es:

La comunicación con la Base de Datos se corta durante la llamada. Se muestra un error.

Pos Condiciones

La nueva importación es creada con éxito. Se muestra un mensaje de éxito.

Calcular Ganancia para Cliente

Descripcion: Permite a un Usuario con rol “admin” calcular la Previsión de ganancias generadas por un Cliente en particular.

Actores: Usuario Autenticado con rol “admin”.

Precondiciones: El Usuario con rol “deposito” está loggeado y en la página de Listar Productos.

Flujo Normal:

1. El Usuario hace click en el botón “Prevision Ganancia” y es llevado a la vista correspondiente.
2. El Usuario hace click en el botón “Calcular Ganancia” del Cliente requerido.
3. El Usuario es llevado a la vista correspondiente.

Flujo/s Excepcional/es:

La comunicación con la Base de Datos se corta durante la llamada. Se muestra un error.

Pos Condiciones

La nueva vista muestra el resultado de la operación de la ganancia prevista para el Cliente.

Generar Archivos

Descripcion: Permite a un Usuario Autenticado generar archivos con toda la información del sistema.

Actores: Cualquier Usuario Autenticado.

Precondiciones: El Usuario está loggeado.

Flujo Normal:

1. El Usuario hace click en el botón “Generar Archivos” y es llevado a la vista correspondiente.

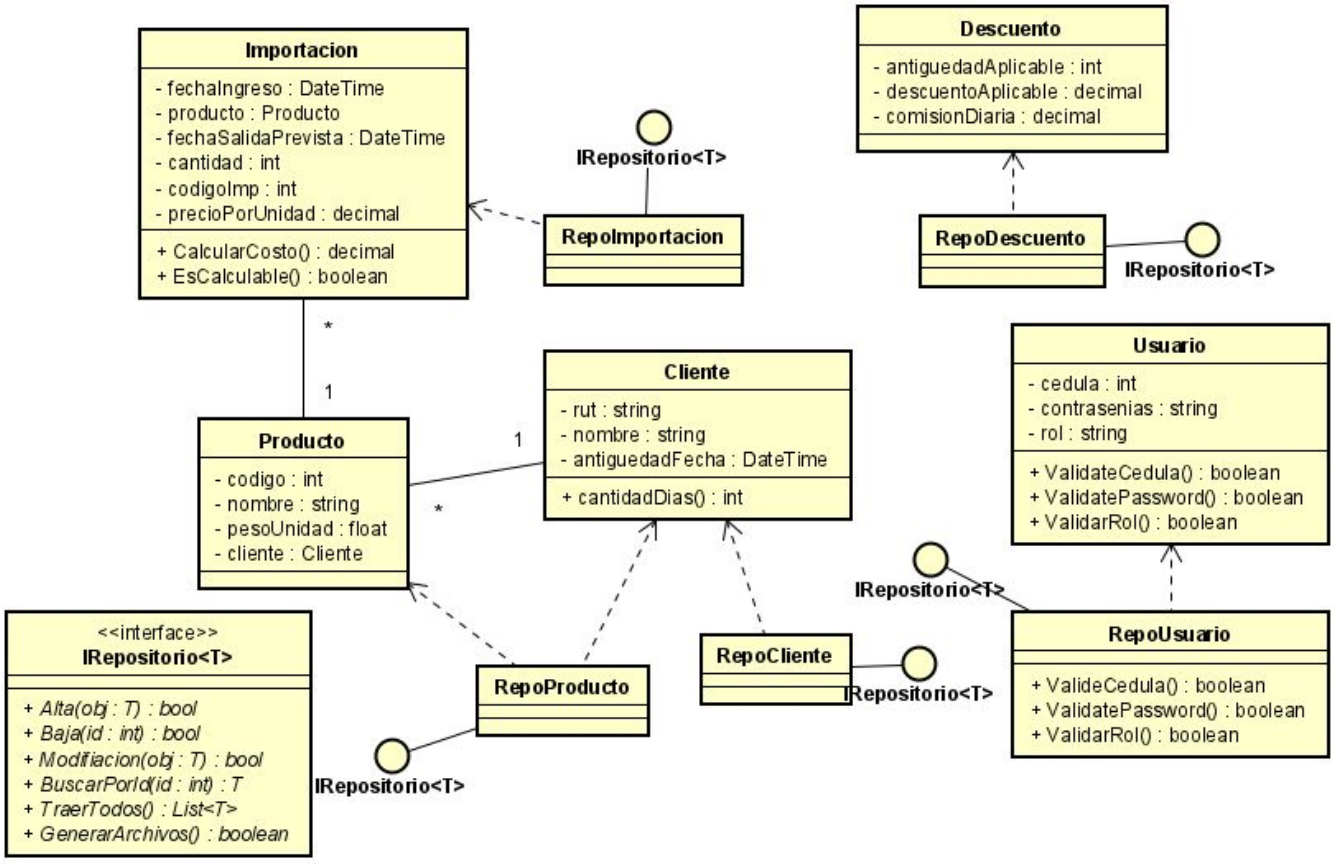
Flujo/s Excepcional/es:

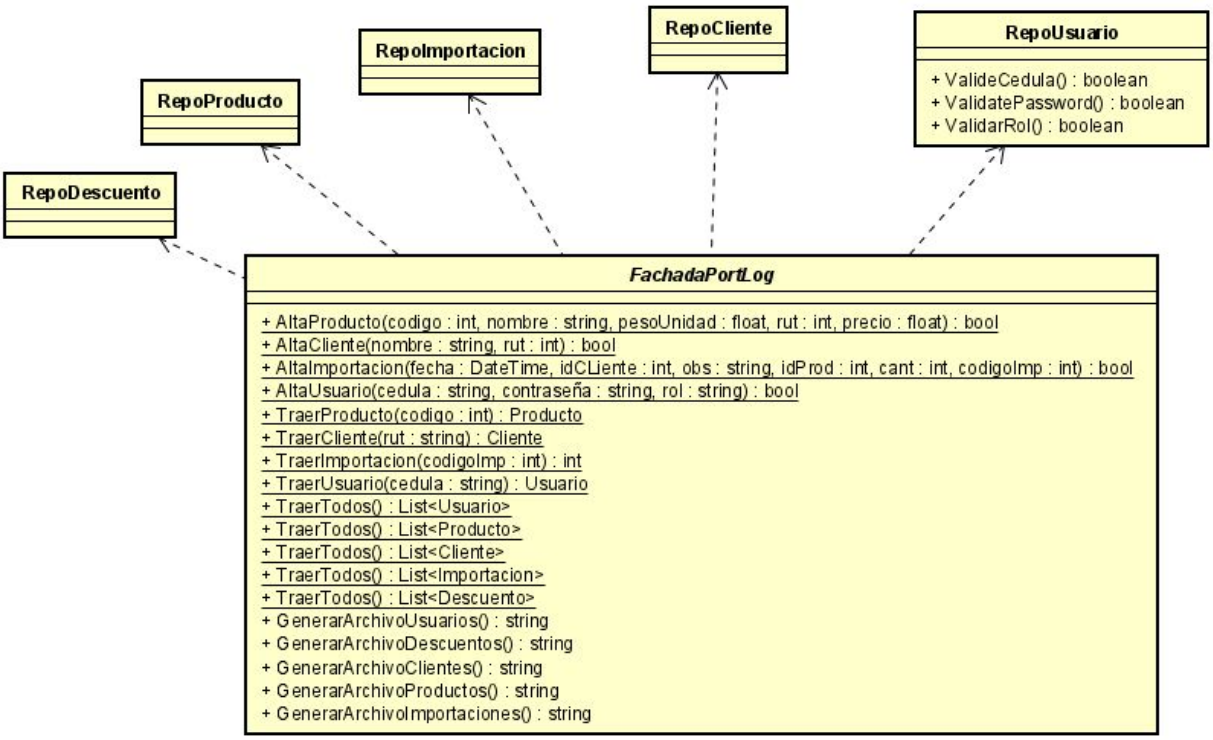
La comunicación con la Base de Datos se corta durante la llamada. Se muestra un error.

Pos Condiciones

La nueva vista muestra un mensaje de éxito y los archivos son generados en su directorio correspondiente.

Diagrama con las clases de Dominio





Codigo fuente de las clases implementadas

//CLASE CLIENTE

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Dominio
{
   public class Cliente
   {
      private string rut;
      private string nombre;
      private DateTime antiguedadFecha;
      public DateTime AntiguedadFecha
      {
         get { return antiguedadFecha;}
         set { antiguedadFecha = value;}
      }

      public string Nombre
```

```
        {
            get { return nombre; }
            set { nombre = value; }
        }
        public string Rut
        {
            get { return rut; }
            set { rut = value; }
        }

        public int CantidadDias() {
            int i = 0;
            TimeSpan iT = DateTime.Today - antiguedadFecha;
            i = iT.Days;
            return i;
        }
    }
}
```

//CLASE DESCUENTO

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Dominio
{
        public class Descuento
        {
        public int AntiguedadAplicable { get; set; }
        public decimal DescuentoAplicable { get; set; }

        public decimal ComisionDiaria { get; set; }

        }
}
```

//CLASE IMPORTACION

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Dominio
{
        public class Importacion
        {
        private DateTime fechaIngreso;
        private DateTime fechaSalidaPrevista;
        private Producto producto;
        private int cantidad;
        private int codigoImp;
        public decimal PrecioPorUnidad { get; set; }

        public DateTime FechaSalida
        {
        get { return fechaSalidaPrevista; }
        set { fechaSalidaPrevista = value; }
        }
```

```
public int CodigoImp
{
get { return codigoImp; }
set { codigoImp= value; }
}

public int Cantidad
{
get { return cantidad; }
set { cantidad = value; }
}


public Producto Producto
{
get { return producto; }
set { producto = value; }
}


public DateTime FechaIngreso
{
get { return fechaIngreso; }
set { fechaIngreso = value; }
}

public decimal CalcularCosto() {
int cantidadDias = 0;
cantidadDias = ((TimeSpan)(fechaSalidaPrevista - fechaIngreso)).Days;
decimal costo = Cantidad * PrecioPorUnidad * cantidadDias;
return costo;
}

public bool EsCalculable() {
bool esCalculable = false;
if (fechaIngreso < DateTime.Now && fechaSalidaPrevista > DateTime.Now) {
        esCalculable = true;
}
return esCalculable;
}


}
```

```
}
```

//CLASE PRODUCTO

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Dominio
{
        public class Producto
        {
        private int codigo;
        private string nombre;
        private Cliente cliente;

        public float PesoUnidad { get; set; }

        public Cliente Cliente
        {
        get { return cliente; }
        set { cliente = value; }
        }
```

```
        public string Nombre
        {
        get { return nombre; }
        set { nombre = value; }
        }

        public int Codigo
        {
        get { return codigo; }
        set { codigo = value; }
        }

        }
}
```

//CLASE USUARIO

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Text.RegularExpressions;

namespace Dominio
{
        public class Usuario
        {
        private int cedula;
        private string contrasenia;
        private string rol;

        public string Rol
        {
        get { return rol; }
        set { rol = value; }
        }


        public string Contrasenia
```

```
{
get { return contrasenia; }
set { contrasenia = value; }
}

public int Cedula
{
get { return cedula; }
set { cedula = value; }
}

public static bool ValidateCedula(string cedula)
{
var input = cedula;
if (string.IsNullOrWhiteSpace(input))
{
return false;
}
var hasMiniMaxChars = new Regex(@".{6,15}");
if (!hasMiniMaxChars.IsMatch(input))
{
        return false;
}
else
{
        return true;
}
}

public static bool ValidatePassword(string password)
{
var input = password;
if (string.IsNullOrWhiteSpace(input))
{
        return false;
}
var hasNumber = new Regex(@"[0-9]+");
var hasUpperChar = new Regex(@"[A-Z]+");
var hasMiniMaxChars = new Regex(@".{6,15}");
var hasLowerChar = new Regex(@"[a-z]+");

if (!hasLowerChar.IsMatch(input))
{
```

```
            return false;
        }
        else if (!hasUpperChar.IsMatch(input))
        {
            return false;
        }
        else if (!hasMiniMaxChars.IsMatch(input))
        {
            return false;
        }
        else if (!hasNumber.IsMatch(input))
        {
            return false;
        }
        else
        {
            return true;
        }
        }

        public static bool ValidarRol(string rolAsignado) {
        bool esValido = false;
        if (rolAsignado == "deposito" || rolAsignado == "admin")
        {
            esValido = true;
        }
        return esValido;
        }

        }
}
```

//CLASE FACHADA

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Dominio;

namespace Repositorios
{
        public static class FachadaDistribuidora
        {
        public static bool AltaUsuario(int cedula, string contrasenia,string rol)
        {
        bool ret = false;
        Usuario usu = new Usuario()
        {
                Cedula= cedula,
                Contrasenia = contrasenia,
                Rol = rol,

        };
        RepoUsuario repoUsu= new RepoUsuario();
        ret = repoUsu.Alta(usu);
        return ret;
```

```
}

public static Usuario BuscarUsuarioPorCi(string ci)
{

Usuario usuarioEncontrado = new Usuario();
RepoUsuario repoUsu = new RepoUsuario();
usuarioEncontrado = repoUsu.BuscarPorId(Convert.ToInt32(ci));
return usuarioEncontrado;
}

public static bool AltaCliente(string rut, string nombre, DateTime antiguedadFecha)
{
bool ret = false;
Cliente clie = new Cliente()
{
        Rut = rut,
        Nombre = nombre,
        AntiguedadFecha = antiguedadFecha

};
RepoCliente repoCli = new RepoCliente();
ret = repoCli.Alta(clie);
return ret;
}

public static List<Cliente> TraerClientes() {
List<Cliente> clientes = new List<Cliente>();
RepoCliente repoCli = new RepoCliente();
clientes = repoCli.TraerTodo();
return clientes;
}

public static Cliente TraerCliente(string rut) {
Cliente cliente = new Cliente();
RepoCliente repoCli = new RepoCliente();
cliente = repoCli.BuscarPorId(Int32.Parse(rut));
return cliente;
}

public static Cliente TraerClientePorRut(string rut)
{
Cliente cliente = new Cliente();
```

```
RepoCliente repoCli = new RepoCliente();
cliente = repoCli.BuscarPorRut(rut);
return cliente;
}

public static bool AltaProducto(string nombre, Cliente cliente, float pesoUnidad) {

bool ret = false;
Producto prod = new Producto()
{
        Nombre = nombre,
        Cliente = cliente,
        PesoUnidad = pesoUnidad

};
RepoProducto repoProd = new RepoProducto();
ret = repoProd.Alta(prod);
return ret;
}

public static Producto BuscarProductoPorId(string id)
{

Producto productoEncontrado = new Producto();
RepoProducto repoProd = new RepoProducto();
productoEncontrado = repoProd.BuscarPorId(Convert.ToInt32(id));
return productoEncontrado;
}

public static List<Importacion> TraerImportaciones() {
List<Importacion> importaciones = new List<Importacion>();
RepoImportacion RepoImp = new RepoImportacion();
importaciones = RepoImp.TraerTodo();
return importaciones;
}

public static string GenerarArchivoCliente() {
RepoCliente repoCli = new RepoCliente();
string devolucion = repoCli.GenerarArchivo();
return devolucion;
}

public static string GenerarArchivoDescuento()
```

```
        {
        RepoDescuento repoDesc = new RepoDescuento();
        string devolucion = repoDesc.GenerarArchivo();
        return devolucion;
        }

        public static string GenerarArchivoImportacion()
        {
        RepoImportacion repoImpo= new RepoImportacion();
        string devolucion = repoImpo.GenerarArchivo();
        return devolucion;
        }

        public static string GenerarArchivoProducto()
        {
        RepoProducto repoProd= new RepoProducto();
        string devolucion = repoProd.GenerarArchivo();
        return devolucion;
        }

        public static string GenerarArchivoUsuario()
        {
        RepoUsuario repoUsu = new RepoUsuario();
        string devolucion = repoUsu.GenerarArchivo();
        return devolucion;
        }
        }
}
```

//CLASE IREPOSITORIO

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Repositorios
{
        interface IRepositorio<T>
        {
        bool Alta(T obj);
        bool Baja(int id);
        bool Modificacion(T obj);
        List<T> TraerTodo();
        T BuscarPorId(int id);
        string GenerarArchivo();
        }
}
```

//CLASE REPOCLIENTE

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Dominio;
using System.Data;
using System.Data.SqlClient;
using System.Text.RegularExpressions;
using System.IO;

namespace Repositorios
{
        public class RepoCliente : IRepositorio<Cliente>
        {
        public bool Alta(Cliente obj)
        {
                bool ret = false;

                //CAMBIAR XXXX POR LO QUE CORRESPONDA!!!!!!!
                string strCon = "Data Source=(local)\\SQLEXPRESS; Initial Catalog=PortLog5;
Integrated Security=SSPI;";
                SqlConnection con = new SqlConnection(strCon);
```

```
                string sql = "insert into Clientes(rut, nombreCli, antiguedadFecha) values(@rut,
@nombreCli, @antiguedadFecha);";
                SqlCommand com = new SqlCommand(sql, con);

                com.Parameters.AddWithValue("@rut", obj.Rut);
                com.Parameters.AddWithValue("@nombreCli", obj.Nombre);
                com.Parameters.AddWithValue("@antiguedadFecha", obj.AntiguedadFecha);

        try
                {
                con.Open();
                int afectadas = com.ExecuteNonQuery();
                con.Close();

                ret = afectadas == 1;
                }
                catch
                {
                throw;
                }
                finally
                {
                if (con.State == ConnectionState.Open) con.Close();
                }

                return ret;

        }

        public bool Baja(int id)
        {
        throw new NotImplementedException();
        }

        public Cliente BuscarPorId(int rut)
        {
        Cliente cli = null;
        string rutI = rut.ToString();
        //CAMBIAR XXXX POR LO QUE CORRESPONDA!!!!!!!
        string strCon = "Data Source=(local)\\SQLEXPRESS; Initial Catalog=PortLog5;
Integrated Security=SSPI;";
        SqlConnection con = new SqlConnection(strCon);
```

```
        string sql = "select * from Clientes where rut=@rut;";
        SqlCommand com = new SqlCommand(sql, con);

        com.Parameters.AddWithValue("@rut", rutI);

        try
        {
                con.Open();
                SqlDataReader reader = com.ExecuteReader();

                if (reader.Read())
                {
                cli = new Cliente
                {
                Rut = reader.GetString(0),
                Nombre = reader.GetString(1),
                AntiguedadFecha = reader.GetDateTime(2)
                };
                }

                con.Close();
        }
        catch
        {
                throw;
        }
        finally
        {
                if (con.State == ConnectionState.Open) con.Close();
        }

        return cli;
        }

        public string GenerarArchivo()
        {
        string devolucion = "";
        //CAMBIAR XXXX POR LO QUE CORRESPONDA!!!!!!!
        string strCon = "Data Source=(local)\\SQLEXPRESS; Initial Catalog=PortLog5;
Integrated Security=SSPI;";
        SqlConnection con = new SqlConnection(strCon);

        string sql = "select * from Clientes;";
```

```
        SqlCommand com = new SqlCommand(sql, con);

        try
        {


                con.Open();
                SqlDataReader reader = com.ExecuteReader();

                while (reader.Read())
                {
                devolucion += reader.GetString(0) + "#" + reader.GetString(1) + "#" +
reader.GetDateTime(2).ToString() + "\n" ;

                }

                con.Close();

        }
        catch
        {
                throw;
        }
        finally
        {
                if (con.State == ConnectionState.Open) con.Close();
        }

        return devolucion;
        }

        public bool Modificacion(Cliente obj)
        {
        throw new NotImplementedException();
        }

        public List<Cliente> TraerTodo()
        {
        List<Cliente> clientes = new List<Cliente>();

        //CAMBIAR XXXX POR LO QUE CORRESPONDA!!!!!!!
        string strCon = "Data Source=(local)\\SQLEXPRESS; Initial Catalog=PortLog5;
Integrated Security=SSPI;";
```

```
        SqlConnection con = new SqlConnection(strCon);

        string sql = "select * from Clientes";
        SqlCommand com = new SqlCommand(sql, con);

        try
        {
                con.Open();
                SqlDataReader reader = com.ExecuteReader();

                while (reader.Read())
                {
                Cliente cli = new Cliente
                {
                Rut = reader.GetString(0),
                Nombre = reader.GetString(1),
                AntiguedadFecha = reader.GetDateTime(2)
                };

                clientes.Add(cli);
                }

                con.Close();
        }
        catch
        {
                throw;
        }
        finally
        {
                if (con.State == ConnectionState.Open) con.Close();
        }

        return clientes;
        }

        public Cliente BuscarPorRut(string rut)
        {
        Cliente cli = null;

        //CAMBIAR XXXX POR LO QUE CORRESPONDA!!!!!!!
        string strCon = "Data Source=(local)\\SQLEXPRESS; Initial Catalog=PortLog5;
Integrated Security=SSPI;";
```

```
        SqlConnection con = new SqlConnection(strCon);

        string sql = "select * from Clientes where rut=@rut;";
        SqlCommand com = new SqlCommand(sql, con);

        com.Parameters.AddWithValue("@rut", rut);

        try
        {
                con.Open();
                SqlDataReader reader = com.ExecuteReader();

                if (reader.Read())
                {
                cli = new Cliente
                {
                Rut = reader.GetString(0),
                Nombre = reader.GetString(1),
                AntiguedadFecha = reader.GetDateTime(2)
                };
                }

                con.Close();
        }
        catch
        {
                throw;
        }
        finally
        {
                if (con.State == ConnectionState.Open) con.Close();
        }

        return cli;
        }

        }
}
```

//CLASE REPODESCUENTO

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Dominio;
using System.Data;
using System.Data.SqlClient;
using System.IO;

namespace Repositorios
{
        public class RepoDescuento : IRepositorio<Descuento>
        {
        public bool Alta(Descuento obj)
        {
        throw new NotImplementedException();
        }

        public bool Baja(int id)
        {
        throw new NotImplementedException();
        }
```

```
        public Descuento BuscarPorId(int id)
        {
        throw new NotImplementedException();
        }

        public string GenerarArchivo()
        {
        string devolucion = "";
        //CAMBIAR XXXX POR LO QUE CORRESPONDA!!!!!!!
        string strCon = "Data Source=(local)\\SQLEXPRESS; Initial Catalog=PortLog5;
Integrated Security=SSPI;";
        SqlConnection con = new SqlConnection(strCon);

        string sql = "select * from Descuentos;";
        SqlCommand com = new SqlCommand(sql, con);

        try
        {


                con.Open();
                SqlDataReader reader = com.ExecuteReader();

                while (reader.Read())
                {
                devolucion += reader.GetInt32(0).ToString() + "#" +
reader.GetDecimal(1).ToString() + "#" + reader.GetDecimal(2) + "\n";

                }

                con.Close();

        }
        catch
        {
                throw;
        }
        finally
        {
                if (con.State == ConnectionState.Open) con.Close();
        }

        return devolucion;
```

```
        }

        public bool Modificacion(Descuento obj)
        {
        throw new NotImplementedException();
        }

        public List<Descuento> TraerTodo()
        {
        List<Descuento> descuentos = new List<Descuento>();

        //CAMBIAR XXXX POR LO QUE CORRESPONDA!!!!!!!
        string strCon = "Data Source=(local)\\SQLEXPRESS; Initial Catalog=PortLog5;
Integrated Security=SSPI;";
        SqlConnection con = new SqlConnection(strCon);

        string sql = "select * from Descuentos";
        SqlCommand com = new SqlCommand(sql, con);

        try
        {
                con.Open();
                SqlDataReader reader = com.ExecuteReader();

                while (reader.Read())
                {
                Descuento desc = new Descuento
                {
                AntiguedadAplicable = reader.GetInt32(0),
                DescuentoAplicable = reader.GetDecimal(1),
                ComisionDiaria = reader.GetDecimal(2)
                };

                descuentos.Add(desc);
                }

                con.Close();
        }
        catch
        {
                throw;
        }
        finally
```

```
        {
                if (con.State == ConnectionState.Open) con.Close();
        }

        return descuentos;
        }
        }
}
```

//CLASE REPOIMPORTACION

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Dominio;
using System.Data;
using System.Data.SqlClient;
using System.Text.RegularExpressions;
using System.IO;

namespace Repositorios
{
        public class RepoImportacion : IRepositorio<Importacion>
        {
        public bool Alta(Importacion obj)
        {
        bool ret = false;

        //CAMBIAR XXXX POR LO QUE CORRESPONDA!!!!!!!
        string strCon = "Data Source=(local)\\SQLEXPRESS; Initial Catalog=PortLog5;
Integrated Security=SSPI;";
        SqlConnection con = new SqlConnection(strCon);
```

```
        string sql = "insert into Importaciones (cantidad, fechaIngreso, fechaSalidaPrevista,
IdProd, precioPorUnidad) values(@cant, @fechaI, @fechaS, @idProd, @precioPorUnidad);";
        SqlCommand com = new SqlCommand(sql, con);

        com.Parameters.AddWithValue("@cant", obj.Cantidad);
        com.Parameters.AddWithValue("@fechaI", obj.FechaIngreso);
        com.Parameters.AddWithValue("@fechaS", obj.FechaSalida);
        com.Parameters.AddWithValue("@idProd", obj.Producto.Codigo);
        com.Parameters.AddWithValue("@precioPorUnidad", obj.PrecioPorUnidad);


        try
        {
                con.Open();
                int afectadas = com.ExecuteNonQuery();
                con.Close();

                ret = afectadas == 1;
        }
        catch
        {
                throw;
        }
        finally
        {
                if (con.State == ConnectionState.Open) con.Close();
        }

        return ret;
        }

        public bool Baja(int id)
        {
        throw new NotImplementedException();
        }

        public Importacion BuscarPorId(int id)
        {
        throw new NotImplementedException();
        }

        public string GenerarArchivo()
        {
```

```
        string devolucion = "";
        //CAMBIAR XXXX POR LO QUE CORRESPONDA!!!!!!!
        string strCon = "Data Source=(local)\\SQLEXPRESS; Initial Catalog=PortLog5;
Integrated Security=SSPI;";
        SqlConnection con = new SqlConnection(strCon);

        string sql = "select * from Importaciones;";
        SqlCommand com = new SqlCommand(sql, con);

        try
        {

                con.Open();
                SqlDataReader reader = com.ExecuteReader();

                while (reader.Read())
                {
                devolucion+= reader.GetInt32(0).ToString() + "#" + reader.GetInt32(1).ToString()
+ "#" + reader.GetDecimal(2).ToString() + "#" + reader.GetDateTime(3).ToString() + "#" +
reader.GetDateTime(4).ToString() + "#" + reader.GetInt32(5).ToString() + "\n";

                }

                con.Close();

        }
        catch
        {
                throw;
        }
        finally
        {
                if (con.State == ConnectionState.Open) con.Close();
        }

        return devolucion;
        }

        public bool Modificacion(Importacion obj)
        {
        throw new NotImplementedException();
        }
```

```
        public List<Importacion> TraerTodo()
        {
        List<Importacion> importaciones = new List<Importacion>();

        //CAMBIAR XXXX POR LO QUE CORRESPONDA!!!!!!!
        string strCon = "Data Source=(local)\\SQLEXPRESS; Initial Catalog=PortLog5;
Integrated Security=SSPI;";
        SqlConnection con = new SqlConnection(strCon);

        string sql = "select * from Importaciones;";
        SqlCommand com = new SqlCommand(sql, con);

        try
        {
                con.Open();
                SqlDataReader reader = com.ExecuteReader();

                while (reader.Read())
                {

                Importacion importacion = new Importacion
                {
                CodigoImp = reader.GetInt32(0),
                Cantidad = reader.GetInt32(1),
                PrecioPorUnidad= reader.GetDecimal(2),
                FechaIngreso = reader.GetDateTime(3),
                FechaSalida = reader.GetDateTime(4),
                Producto =
FachadaDistribuidora.BuscarProductoPorId((Convert.ToString(reader.GetInt32(5))))

                };

                importaciones.Add(importacion);

                }
                con.Close();
        }
        catch
        {
                throw;
        }
        finally
        {
```

```
                if (con.State == ConnectionState.Open) con.Close();
        }

        return importaciones;
        }


        }
}
```

//CLASE REPOPRODUCTO

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Dominio;
using System.Data;
using System.Data.SqlClient;
using Repositorios;
using System.Globalization;
using System.IO;

namespace Repositorios
{
        public class RepoProducto : IRepositorio<Producto>
        {
        public bool Alta(Producto obj)
        {

                bool ret = false;

                //CAMBIAR XXXX POR LO QUE CORRESPONDA!!!!!!!
                string strCon = "Data Source=(local)\\SQLEXPRESS; Initial Catalog=PortLog5;
Integrated Security=SSPI;";
                SqlConnection con = new SqlConnection(strCon);
```

```
            string sql = "insert into Productos(nombreProd, pesoUnidad, rut) values(@nom,
@pesoU, @rut);";
            SqlCommand com = new SqlCommand(sql, con);

            com.Parameters.AddWithValue("@nom", obj.Nombre);
            com.Parameters.AddWithValue("@pesoU", obj.PesoUnidad);
            com.Parameters.AddWithValue("@rut", obj.Cliente.Rut);

            try
            {
            con.Open();
            int afectadas = com.ExecuteNonQuery();
            con.Close();

            ret = afectadas == 1;
            }
            catch
            {
            throw;
            }
            finally
            {
            if (con.State == ConnectionState.Open) con.Close();
            }

            return ret;

        }

        public bool Baja(int id)
        {
        throw new NotImplementedException();
        }

        public Producto BuscarPorId(int id)
        {
        Producto productoExistente = null;

        //CAMBIAR XXXX POR LO QUE CORRESPONDA!!!!!!!
        string strCon = "Data Source=(local)\\SQLEXPRESS; Initial Catalog=PortLog5;
Integrated Security=SSPI;";
        SqlConnection con = new SqlConnection(strCon);
```

```
        string sql = "select * from Productos where IdProd=@id;";
        SqlCommand com = new SqlCommand(sql, con);

        com.Parameters.AddWithValue("@id", id);

        try
        {
                con.Open();
                SqlDataReader reader = com.ExecuteReader();

                if (reader.Read())
                {
                productoExistente = new Producto
                {
                Codigo = reader.GetInt32(0),
                Nombre = reader.GetString(1),
                PesoUnidad = (float)reader.GetDouble(2),
                Cliente = FachadaDistribuidora.TraerClientePorRut(reader.GetString(3))
                };
                }
                con.Close();
        }
        catch
        {
                throw;
        }
        finally
        {
                if (con.State == ConnectionState.Open) con.Close();
        }
        return productoExistente;
        }

        public string GenerarArchivo()
        {
        string devolucion = "";
        //CAMBIAR XXXX POR LO QUE CORRESPONDA!!!!!!!
        string strCon = "Data Source=(local)\\SQLEXPRESS; Initial Catalog=PortLog5;
Integrated Security=SSPI;";
        SqlConnection con = new SqlConnection(strCon);

        string sql = "select * from Productos;";
```

```
        SqlCommand com = new SqlCommand(sql, con);

        try
        {

                con.Open();
                SqlDataReader reader = com.ExecuteReader();

                while (reader.Read())
                {
                devolucion+= reader.GetInt32(0).ToString() + "#" + reader.GetString(1) + "#" +
reader.GetDouble(2).ToString() + "#" + reader.GetString(3) + "\n";

                }
                con.Close();

        }
        catch
        {
                throw;
        }
        finally
        {
                if (con.State == ConnectionState.Open) con.Close();
        }

        return devolucion;
        }

        public bool Modificacion(Producto obj)
        {
        throw new NotImplementedException();
        }

        public List<Producto> TraerTodo()
        {
        List<Producto> productos = new List<Producto>();

        //CAMBIAR XXXX POR LO QUE CORRESPONDA!!!!!!!
        string strCon = "Data Source=(local)\\SQLEXPRESS; Initial Catalog=PortLog5;
Integrated Security=SSPI;";
        SqlConnection con = new SqlConnection(strCon);
```

```
        string sql = "select * from Productos;";
        SqlCommand com = new SqlCommand(sql, con);

        try
        {
                con.Open();
                SqlDataReader reader = com.ExecuteReader();

                while (reader.Read())
                {

                Producto producto = new Producto
                {
                Codigo = reader.GetInt32(0),
                Nombre = reader.GetString(1),
                PesoUnidad = (float)reader.GetDouble(2),
                Cliente = FachadaDistribuidora.TraerClientePorRut(reader.GetString(3))
                };

                productos.Add(producto);

                }
                con.Close();
        }
        catch
        {
                throw;
        }
        finally
        {
                if (con.State == ConnectionState.Open) con.Close();
        }

        return productos;
        }
        }
}
```

//CLASE REPOUSUARIO

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Dominio;
using System.Data;
using System.Data.SqlClient;
using System.Text.RegularExpressions;
using System.IO;

namespace Repositorios
{
        public class RepoUsuario : IRepositorio<Usuario>
        {
        public bool Alta(Usuario obj)
        {
        {
                bool ret = false;

                if (ValidarRol(obj.Rol) && ValidateCedula(Convert.ToString(obj.Cedula)) &&
ValidatePassword(obj.Contrasenia)) {

                //CAMBIAR XXXX POR LO QUE CORRESPONDA!!!!!!!
```

```
            string strCon = "Data Source=(local)\\SQLEXPRESS; Initial Catalog=PortLog5;
Integrated Security=SSPI;";
            SqlConnection con = new SqlConnection(strCon);

            string sql = "insert into Usuarios(cedula, contraseña, rol) values(@cedula, @pw,
@rol);";
            SqlCommand com = new SqlCommand(sql, con);

            com.Parameters.AddWithValue("@cedula", obj.Cedula);
            com.Parameters.AddWithValue("@pw", obj.Contrasenia);
            com.Parameters.AddWithValue("@rol", obj.Rol);

            try
            {
            con.Open();
            int afectadas = com.ExecuteNonQuery();
            con.Close();

            ret = afectadas == 1;
            }
            catch
            {
            throw;
            }
            finally
            {
            if (con.State == ConnectionState.Open) con.Close();
            }

            }

            return ret;
        }

        }

        public bool Baja(int id)
        {
        throw new NotImplementedException();
        }

        public Usuario BuscarPorId(int cedulaEnviada)
        {
```

```
        Usuario usuarioExistente = null;

        //CAMBIAR XXXX POR LO QUE CORRESPONDA!!!!!!!
        string strCon = "Data Source=(local)\\SQLEXPRESS; Initial Catalog=PortLog5;
Integrated Security=SSPI;";
        SqlConnection con = new SqlConnection(strCon);

        string sql = "select * from Usuarios where cedula=@ci;";
        SqlCommand com = new SqlCommand(sql, con);

        com.Parameters.AddWithValue("@ci", cedulaEnviada);

        try
        {
                con.Open();
                SqlDataReader reader = com.ExecuteReader();

                if (reader.Read())
                {
                usuarioExistente = new Usuario
                {
                Cedula = reader.GetInt32(0),
                Contrasenia = reader.GetString(1),
                Rol = reader.GetString(2)
                };
                }
                con.Close();
        }
        catch
        {
                throw;
        }
        finally
        {
                if (con.State == ConnectionState.Open) con.Close();
        }
        return usuarioExistente;
        }

        public bool Modificacion(Usuario obj)
        {
        throw new NotImplementedException();
        }
```

```
public List<Usuario> TraerTodo()
{
throw new NotImplementedException();
}

private static bool ValidateCedula(string cedula)
{
var input = cedula;
if (string.IsNullOrWhiteSpace(input))
{
        return false;
}
var hasMiniMaxChars = new Regex(@".{6,15}");
if (!hasMiniMaxChars.IsMatch(input))
{
        return false;
}
else
{
        return true;
}
}

private static bool ValidatePassword(string password)
{
var input = password;
if (string.IsNullOrWhiteSpace(input))
{
        return false;
}
var hasNumber = new Regex(@"[0-9]+");
var hasUpperChar = new Regex(@"[A-Z]+");
var hasMiniMaxChars = new Regex(@".{6,15}");
var hasLowerChar = new Regex(@"[a-z]+");

if (!hasLowerChar.IsMatch(input))
{
        return false;
}
else if (!hasUpperChar.IsMatch(input))
{
        return false;
```

```
        }
        else if (!hasMiniMaxChars.IsMatch(input))
        {
                return false;
        }
        else if (!hasNumber.IsMatch(input))
        {
                return false;
        }
        else
        {
                return true;
        }
        }

        private static bool ValidarRol(string rolAsignado)
        {
        bool esValido = false;
        if (rolAsignado == "deposito" || rolAsignado == "admin")
        {
                esValido = true;
        }
        return esValido;
        }

        public string GenerarArchivo()
        {
        string devolucion = "";
        //CAMBIAR XXXX POR LO QUE CORRESPONDA!!!!!!!
        string strCon = "Data Source=(local)\\SQLEXPRESS; Initial Catalog=PortLog5;
Integrated Security=SSPI;";
        SqlConnection con = new SqlConnection(strCon);
        string sql = "select * from Usuarios;";
        SqlCommand com = new SqlCommand(sql, con);

        try
        {
                        con.Open();
                SqlDataReader reader = com.ExecuteReader();

                while (reader.Read())
                {
```

```
                devolucion += reader.GetInt32(0).ToString() + "#" + reader.GetString(1) + "#" +
reader.GetString(2)+ "\n";
                }
                con.Close();
                }
        catch
        {
                throw;
        }
        finally
        {
                if (con.State == ConnectionState.Open) con.Close();
        }
        return devolucion;
        }
        }
}
```

Script para creacion de tablas

USUARIOS (**cedula**, contraseña, rol)
CLIENTES (**rut**, nombreCli, antiguedadFecha)
PRODUCTOS (**IdProd**, nombreProd, pesoUnidad, *rut*)
IMPORTACIONES (**codigoImp**, cantidad, fechaIngreso, fechaSalidaPrevista, precioPorUnidad, *IdProd*)
DESCUENTO (antiguedadAplicable, descuentoAplicable, comisionDiaria)

Script

```
CREATE DATABASE PortLog5;
USE [PortLog5]
CREATE TABLE Usuarios (
        cedula int NOT NULL,
        contraseña nvarchar(50) NOT NULL,
        rol nvarchar(50) NOT NULL,
        PRIMARY KEY (cedula)
);

CREATE TABLE Clientes (
        rut nvarchar(50) NOT NULL,
        nombreCli nvarchar(50) NOT NULL,
```

```
	antiguedadFecha DATETIME,
	PRIMARY KEY (rut)
);

CREATE TABLE Productos (
	IdProd int IDENTITY(1,1) NOT NULL,
	nombreProd nvarchar(50) NOT NULL,
	pesoUnidad float,
	PRIMARY KEY (IdProd),
	rut nvarchar(50) FOREIGN KEY REFERENCES Clientes(rut)
);

CREATE TABLE Importaciones (
	codigoImp int IDENTITY(1,1) NOT NULL,
	cantidad int NOT NULL,
precioPorUnidad decimal,
	fechaIngreso DATETIME,
	fechaSalidaPrevista DATETIME,
	PRIMARY KEY (codigoImp),
	IdProd int FOREIGN KEY REFERENCES Productos(IdProd)
);

CREATE TABLE Descuentos (
	antiguedadDiasAplicable int,
	descuentoAplicable decimal,
	comisionDiaria decimal
);



/*INSERTS*/
/*Tabla Usuarios*/
INSERT INTO Usuarios VALUES
(1, 'uno', 'admin'),
(473192081, 'Aminta1983', 'admin'),
(482498581, 'Valium1996', 'admin'),
(473192082, 'Aminta1983', 'deposito'),
(482498582, 'Valium1996', 'deposito');

/*Tabla Clientes:*/
insert into Clientes values
('123488889902', 'Tata Consultancy', '14-JUN-2019'),
('123477778565','Atos SA','01-JUL-2018'),
```

```
('144321514562','ANTEL','07-FEB-2020'),
('155555552351','Papaleras Hnos', '21-MAR-2017'),
('110003095823','Farmashop','25-SEP-2016');

/*Tabla Productos:*/
insert into Productos values
('Indica Weed Critical',0.20, '123488889902'),
('Metales',70.5, '123488889902'),
('Maderas', 230,'144321514562'),
('Craft Beer',300,'123477778565'),
('Peras',11,'155555552351'),
('Mouses',30,'110003095823'),
('Teclado', 11,'123488889902'),
('Parlantes',38,'123477778565'),
('Herramientas de Lu', 554,'123477778565'),
('MultiJuice Potion',200,'110003095823');

/*Tabla Importaciones*/
INSERT INTO Importaciones VALUES
(15,10, '14-JUN-2020', '20-JUN-2020', 1),
(20,200, '01-JAN-2020', '20-JAN-2020', 2),
(350,70, '01-JAN-2020', '20-JUN-2020', 3),
(50,100, '11-FEB-2020', '30-JUL-2020', 4),
(50,100, '21-MAR-2020', '15-SEP-2020', 5);

/*Tabla Descuentos*/
INSERT INTO Descuentos VALUES (365, 5, 3);
```