# Correcciones primera tutoría

Entidades compratidas

[KnownType(typeof(Empresa))]

[KnownType(typeof(Empleado))]

[DataContract]

public abstract class Usuario

{

private string \_logueo;

private string \_contrasenia;

private string \_nombreCompleto;

[DataMember]

public string Logueo

{

get { return \_logueo; }

set

{

if (value.Trim().Length > 12 || value.Trim().Length <= 0)

throw new Exception("Usuario debe tener 12 caracteres");

else

\_logueo = value;

}

}

[DataMember]

public string Contrasena

{

get { return \_contrasenia; }

set {

Regex \_expresion = new Regex("[a-zA-ZñÑ]{3}[0-9]{2}[|°¬¡!#$%&/=()¿?'\_´{},;.:`+\*~^<>@]");

if (!\_expresion.IsMatch(value))

{

throw new Exception("El formato de la contraseña no es válido.");

}

else

{

\_contrasenia = value;

}

}

}

[DataMember]

public string NombreCompleto

{

get { return \_nombreCompleto; }

set

{

if ((value.Trim().Length > 50) || (value.Trim().Length <= 0))

throw new Exception("Nombre cliente inválido");

else

\_nombreCompleto = value;

}

}

public Usuario()

{ }

public Usuario(string logueo, string contrasenia, string nombreCompleto)

{

Logueo = logueo;

Contrasena = contrasenia;

NombreCompleto = nombreCompleto;

}

}

[DataContract]

public class Empresa : Usuario

{

private string \_telefono;

private string \_direccion;

private string \_email;

[DataMember]

public string Telefono

{

get { return \_telefono; }

set {

if (value.Trim().Length == 9)

\_telefono = value;

else

throw new Exception("Telefono inválido");

}

}

[DataMember]

public string Direccion

{

get { return \_direccion; }

set

{

if (value.Trim().Length > 0 && value.Trim().Length <= 50)

\_direccion = value;

else

throw new Exception("Direccion inválida");

}

}

[DataMember]

public string Email

{

get { return \_email; }

set {

if (Regex.IsMatch(value, @"^[^@\s]+@[^@\s]+\.[^@\s]+$", RegexOptions.IgnoreCase))

\_email = value;

else

throw new Exception("Email inválido");

}

}

public Empresa()

{ }

public Empresa(string logueo, string contrasenia, string nombreCompleto,

string telefono, string direccion, string email)

:base(logueo, contrasenia, nombreCompleto)

{

Telefono = telefono;

Direccion = direccion;

Email = email;

}

}

[DataContract]

public class Empleado : Usuario

{

private string \_horaInicio;

private string \_horaFin;

[DataMember]

public string HoraInicio

{

get { return \_horaInicio; }

set

{

Regex \_expresion = new Regex("[0-2][0-9][:][0-5][0-9][:][0-5][0-9]");

if (!\_expresion.IsMatch(value))

throw new Exception("Formato hora inicio inválido.");

else

\_horaInicio = value;

}

}

[DataMember]

public string HoraFin

{

get { return \_horaFin; }

set

{

if (!new Regex("[0-2][0-9][:][0-5][0-9][:][0-5][0-9]").IsMatch(value))

throw new Exception("Formato hora fin inválido.");

else

\_horaFin = value;

}

}

public Empleado()

{

}

public Empleado(string logueo, string contrasenia, string nombreCompleto,

string horaInicio, string horaFin)

:base(logueo, contrasenia, nombreCompleto)

{

HoraInicio = horaInicio;

HoraFin = horaFin;

}

}

[DataContract]

public class Paquete

{

private int \_codigo;

private string \_tipo;

private string \_descripcion;

private double \_peso;

private Empresa \_empresaOrigen;

[DataMember]

public int Codigo

{

get { return \_codigo; }

set {

if (value >= 1)

\_codigo = value;

else

throw new Exception("Codigo paquete inválido");

}

}

[DataMember]

public string Tipo

{

get { return \_tipo; }

set {

string[] arr = { "fragil", "comun", "bulto"};

if (arr.All(value.Contains))

\_tipo = value;

else

throw new Exception("Tipo inválido");

}

}

[DataMember]

public string Descripcion

{

get { return \_descripcion; }

set {

if (value.Trim().Length > 0 && value.Trim().Length <= 100)

\_descripcion = value;

else

throw new Exception("Descripcion inválida");

}

}

[DataMember]

public double Peso

{

get { return \_peso; }

set {

if (value > 0)

\_peso = value;

else

throw new Exception("Peso inválido");

}

}

[DataMember]

public Empresa EmpresaOrigen

{

get { return \_empresaOrigen; }

set {

if (value != null)

\_empresaOrigen = value;

else

throw new Exception("Empleado inválido para paquete");

}

}

public Paquete()

{

}

public Paquete(int codigo, string tipo, string descripcion, double peso, Empresa empresaOrigen)

{

Codigo = codigo;

Tipo = tipo;

Descripcion = descripcion;

Peso = peso;

EmpresaOrigen = empresaOrigen;

}

}

[DataContract]

public class Solicitud

{

private int \_numero;

private DateTime \_fechaEntrega;

private string \_nombreDestinatario;

private string \_direccionDestinatario;

private string \_estado;

private Empleado \_empleado;

private List<Paquete> \_paquetesSolicitud = new List<Paquete>();

[DataMember]

public int Numero

{

get { return \_numero; }

set {

if (value > 0)

\_numero = value;

else

throw new Exception("Numero solicitud inválido");

}

}

[DataMember]

public DateTime FechaEntrega

{

get { return \_fechaEntrega; }

set {

if (value != null && value > DateTime.Today)

\_fechaEntrega = value;

else

throw new Exception("Fecha de entrega inválida");

}

}

[DataMember]

public string NombreDestinatario

{

get { return \_nombreDestinatario; }

set {

if (value.Trim().Length > 0 && value.Trim().Length <= 50)

\_nombreDestinatario = value;

else

throw new Exception("Direccion inválida");

}

}

[DataMember]

public string DireccionDestinatario

{

get { return \_direccionDestinatario; }

set {

if (value.Trim().Length > 0 && value.Trim().Length <= 50)

\_direccionDestinatario = value;

else

throw new Exception("Direccion inválida");

}

}

[DataMember]

public string Estado

{

get { return \_estado; }

set {

string[] arr = { "en deposito", "en camino", "entregado" };

if (arr.All(value.Contains))

\_estado = value;

else

throw new Exception("Estado inválido");

}

}

[DataMember]

public Empleado Empleado

{

get { return \_empleado; }

set {

if (value != null)

\_empleado = value;

else

throw new Exception("Empleado inválido");

}

}

[DataMember]

public List<Paquete> PaquetesSolicitud

{

get { return \_paquetesSolicitud; }

set {

if (value != null && value.Count > 0)

\_paquetesSolicitud = value;

else

throw new Exception("Solicitud sin paquetes");

}

}

public Solicitud()

{

}

public Solicitud(int numero, DateTime fechaEntrega, string nombreDestinatario,

string direccionDestinatario, string estado, Empleado empleado, List<Paquete> paquetesSolicitud)

{

Numero = numero;

FechaEntrega = fechaEntrega;

NombreDestinatario = nombreDestinatario;

DireccionDestinatario = direccionDestinatario;

Estado = estado;

Empleado = empleado;

PaquetesSolicitud = paquetesSolicitud;

}

}

Definición de wcf

[ServiceContract]

public interface IService1

{

#region LogicaUsuario

[OperationContract]

void AltaUsuario(Usuario usuario, Empleado usLog);

[OperationContract]

void BajaUsuario(Usuario usuario, Empleado usLog);

[OperationContract]

Usuario BuscarUsuario(string logueo, Empleado usLog);

[OperationContract]

Usuario LogueoUsuario(string logueo, string contraseana);

[OperationContract]

void ModificarUsuario(Usuario usuario, Empleado usLog);

[OperationContract]

List<Empresa> ListarEmpresas(Empleado usuarioLogueado);

#endregion

#region LogicaPaquete

[OperationContract]

void AltaPaquete(Paquete paquete, Empleado usLog);

[OperationContract]

Paquete BuscarPaquete(int codigo, Empleado usLog);

[OperationContract]

List<Paquete> ListadoPaquetesSinSolicitud(Empleado usLog);

#endregion

#region LogicaSolicitud

[OperationContract]

void AltaSolicitud(Solicitud solicitud, Empleado usLog);

[OperationContract]

void ModificarEstadoSolicitud(Solicitud solicitud, Empleado usLog);

[OperationContract]

string listadoSolicitudesEnCamino(Usuario usLog);

[OperationContract]

List<Solicitud> listadoSolicitudesEmpresa(Empresa usLog);

#endregion

}

public class Service1 : IService1

{

#region LogicaUsuario

void IService1.AltaUsuario(Usuario usuario, Empleado usLog)

{

FabricaLogica.GetLogicaUsuario().AltaUsuario(usuario, usLog);

}

void IService1.BajaUsuario(Usuario usuario, Empleado usLog)

{

FabricaLogica.GetLogicaUsuario().BajaUsuario(usuario, usLog);

}

Usuario IService1.BuscarUsuario(string logueo, Empleado usLog)

{

return FabricaLogica.GetLogicaUsuario().BuscarUsuario(logueo, usLog);

}

Usuario IService1.LogueoUsuario(string logueo, string contraseana)

{

return FabricaLogica.GetLogicaUsuario().LogueoUsuario(logueo, contraseana);

}

void IService1.ModificarUsuario(Usuario usuario, Empleado usLog)

{

FabricaLogica.GetLogicaUsuario().ModificarUsuario(usuario, usLog);

}

List<Empresa> IService1.ListarEmpresas(Empleado usuarioLogueado)

{

return FabricaLogica.GetLogicaUsuario().ListarEmpresas(usuarioLogueado);

}

#endregion

#region LogicaPaquete

void IService1.AltaPaquete(Paquete paquete, Empleado usLog)

{

FabricaLogica.GetLogicaPaquete().AltaPaquete(paquete, usLog);

}

Paquete IService1.BuscarPaquete(int codigo, Empleado usLog)

{

return FabricaLogica.GetLogicaPaquete().BuscarPaquete(codigo, usLog);

}

List<Paquete> IService1.ListadoPaquetesSinSolicitud(Empleado usLog)

{

return FabricaLogica.GetLogicaPaquete().ListadoPaquetesSinSolicitud(usLog);

}

#endregion

#region LogicaSolicitud

void IService1.AltaSolicitud(Solicitud solicitud, Empleado usLog)

{

FabricaLogica.GetLogicaSolicitud().AltaSolicitud(solicitud, usLog);

}

void IService1.ModificarEstadoSolicitud(Solicitud solicitud, Empleado usLog)

{

FabricaLogica.GetLogicaSolicitud().ModificarEstadoSolicitud(solicitud, usLog);

}

string IService1.listadoSolicitudesEnCamino(Usuario usLog)

{

return FabricaLogica.GetLogicaSolicitud().listadoSolicitudesEnCamino(usLog);

}

List<Solicitud> IService1.listadoSolicitudesEmpresa(Empresa usLog)

{

return FabricaLogica.GetLogicaSolicitud().listadoSolicitudesEmpresa(usLog);

}

#endregion

}

Conexión.cs

namespace Persistencia

{

internal class Conexion

{

internal static string Cnn(EntidadesCompartidas.Usuario u = null)

{

if (u == null)

return "Data Source =.; Initial Catalog = ProyectoSegundo2020; Integrated Security = true";

else

return "Data Source =.; Initial Catalog = ProyectoSegundo2020; User=" + u.Logueo + "; Password='" + u.Contrasena + "'";

}

}

}

Logueo usuario

public Empleado LoguearEmpleado(string logueo, string contrasena)

{

SqlConnection conexion = null;

SqlDataReader drEmpleado = null;

Empleado empleado = null;

try

{

conexion = new SqlConnection(Conexion.Cnn(null));

SqlCommand cmdBuscarEmpresa = new SqlCommand("LogueoEmpleado", conexion);

cmdBuscarEmpresa.CommandType = CommandType.StoredProcedure;

cmdBuscarEmpresa.Parameters.AddWithValue("@logueo", logueo);

cmdBuscarEmpresa.Parameters.AddWithValue("@contrasena", contrasena);

conexion.Open();

drEmpleado = cmdBuscarEmpresa.ExecuteReader();

if (drEmpleado.Read())

{

empleado = new Empleado((string)drEmpleado["logueo"], (string)drEmpleado["contrasena"], (string)drEmpleado["nombreCompleto"], Convert.ToString((TimeSpan)drEmpleado["horaInicio"]), Convert.ToString((TimeSpan)drEmpleado["horaFin"]));

}

return empleado;

}

catch (Exception ex)

{

throw ex;

}

finally

{

if (drEmpleado != null)

{

drEmpleado.Close();

}

if (conexion != null)

{

conexion.Close();

}

}

}

public Empresa LoguearEmpresa(string logueo, string contrasena)

{

SqlConnection conexion = null;

SqlDataReader drEmpresa = null;

Empresa empresa = null;

try

{

conexion = new SqlConnection(Conexion.Cnn(null));

SqlCommand cmdBuscarEpresa = new SqlCommand("LogueoEmpresa", conexion);

cmdBuscarEpresa.CommandType = CommandType.StoredProcedure;

cmdBuscarEpresa.Parameters.AddWithValue("@logueo", logueo);

cmdBuscarEpresa.Parameters.AddWithValue("@contrasena", contrasena);

conexion.Open();

drEmpresa = cmdBuscarEpresa.ExecuteReader();

if (drEmpresa.Read())

{

empresa = new Empresa((string)drEmpresa["logueo"], (string)drEmpresa["contrasena"], (string)drEmpresa["nombreCompleto"], (string)drEmpresa["telefono"], (string)drEmpresa["direccion"], (string)drEmpresa["email"]);

}

return empresa;

}

catch (Exception ex)

{

throw ex;

}

finally

{

if (drEmpresa != null)

{

drEmpresa.Close();

}

if (conexion != null)

{

conexion.Close();

}

}

}

CREATE PROCEDURE LogueoEmpleado

@logueo VARCHAR(12),

@contrasena varchar(6)

AS

BEGIN

SELECT a.\*, b.horaInicio, b.horaFin

FROM Usuario a INNER JOIN Empleado b

ON a.logueo = b.logueo

WHERE a.logueo = @logueo AND a.contrasena = @contrasena AND a.activo = 1

END

CREATE PROCEDURE LogueoEmpresa

@logueo VARCHAR(12),

@contrasena varchar(6)

AS

BEGIN

SELECT a.\*, b.direccion, b.email, b.telefono

FROM Usuario a INNER JOIN Empresa b

ON a.logueo = b.logueo

WHERE a.logueo = @logueo AND a.contrasena = @contrasena AND a.activo = 1

END

Búsqueda de usuario internal (ejemplo empresa)

internal Empresa interBuscarEmpresa(string logueo)

{

SqlConnection conexion = null;

SqlDataReader drEmpresa = null;

Empresa empresa = null;

try

{

conexion = new SqlConnection(Conexion.Cnn(null));

SqlCommand cmdBuscarEpresa = new SqlCommand("interBuscarEmpresa", conexion);

cmdBuscarEpresa.CommandType = CommandType.StoredProcedure;

cmdBuscarEpresa.Parameters.AddWithValue("@logueo", logueo);

conexion.Open();

drEmpresa = cmdBuscarEpresa.ExecuteReader();

if (drEmpresa.Read())

{

empresa = new Empresa((string)drEmpresa["logueo"], (string)drEmpresa["contrasena"], (string)drEmpresa["nombreCompleto"], (string)drEmpresa["telefono"], (string)drEmpresa["direccion"], (string)drEmpresa["email"]);

}

return empresa;

}

catch (Exception ex)

{

throw ex;

}

finally

{

if (drEmpresa != null)

{

drEmpresa.Close();

}

if (conexion != null)

{

conexion.Close();

}

}

}

CREATE PROCEDURE interBuscarEmpresa

@logueo VARCHAR(12)

AS

BEGIN

SELECT Usuario.\*, Empresa.telefono, Empresa.direccion, Empresa.email

FROM Usuario INNER JOIN Empresa

ON Usuario.logueo = Empresa.logueo

WHERE Usuario.logueo = @logueo

END

Alta paquete solicitud

internal void AltaPaqueteSolicitud(SqlTransaction trn, Solicitud solicitud, Paquete paquete)

{

try

{

SqlCommand cmdAltaPaquete = new SqlCommand("AltaPaqueteSolicitud", trn.Connection);

cmdAltaPaquete.CommandType = CommandType.StoredProcedure;

cmdAltaPaquete.Parameters.AddWithValue("@codigoPaquete", paquete.Codigo);

cmdAltaPaquete.Parameters.AddWithValue("@numeroSolicitud", solicitud.Numero);

SqlParameter valorRetorno = new SqlParameter("@valorRetorno", SqlDbType.Int);

valorRetorno.Direction = ParameterDirection.ReturnValue;

cmdAltaPaquete.Parameters.Add(valorRetorno);

cmdAltaPaquete.Transaction = trn;

cmdAltaPaquete.ExecuteNonQuery();

switch ((int)valorRetorno.Value)

{

case -1:

throw new Exception("No existe solicitud.");

case -2:

throw new Exception("No existe paquete.");

case -3:

throw new Exception("Paquete ya existente en solicitud.");

case -4:

throw new Exception("Error al insertar paqueteSolicitud.");

}

}

catch (Exception ex)

{

trn.Rollback();

throw ex;

}

}

CREATE PROCEDURE AltaPaqueteSolicitud

@numeroSolicitud INT,

@codigoPaquete INT

AS

BEGIN

IF NOT EXISTS (SELECT \* FROM Solicitud WHERE numero = @numeroSolicitud)

BEGIN

RETURN -1

END

IF NOT EXISTS (SELECT \* FROM Paquete WHERE codigo = @codigoPaquete)

BEGIN

RETURN -2

END

IF EXISTS (SELECT \* FROM PaquetesSolicitud

WHERE numeroSolicitud = @numeroSolicitud AND codigoPaquete = @codigoPaquete)

BEGIN

RETURN -3

END

INSERT INTO PaquetesSolicitud

VALUES (@numeroSolicitud, @codigoPaquete)

IF (@@ERROR <> 0)

BEGIN

RETURN -4

END

END

Listado paquetes solicitud

internal List<Paquete> listadoPaquetesSolicitud(int solicitud)

{

Paquete paquete = null;

Empresa empresa = null;

List<Paquete> listaPaquetes = new List<Paquete>();

SqlConnection conexion = null;

SqlDataReader drPaquete = null;

try

{

conexion = new SqlConnection(Conexion.Cnn(null));

SqlCommand cmdListadoSolicitues = new SqlCommand("ListarPaquetesSolicitud", conexion);

cmdListadoSolicitues.CommandType = CommandType.StoredProcedure;

cmdListadoSolicitues.Parameters.AddWithValue("@numeroSolicitud", solicitud);

conexion.Open();

drPaquete = cmdListadoSolicitues.ExecuteReader();

while (drPaquete.Read())

{

empresa = PersistenciaEmpresa.getInstancia().interBuscarEmpresa((string)drPaquete["empresa"]);

paquete = new Paquete((int)drPaquete["codigo"], (string)drPaquete["tipo"], (string)drPaquete["descripcion"],

(double)drPaquete["peso"], empresa);

listaPaquetes.Add(paquete);

}

return listaPaquetes;

}

catch (Exception ex)

{

throw ex;

}

}

CREATE PROCEDURE ListarPaquetesSolicitud

@solicitud int

AS

BEGIN

SELECT \*

FROM Paquete

WHERE codigo in (

SELECT codigoPaquete

FROM PaquetesSolicitud

WHERE numeroSolicitud = @solicitud)

END

Listado paquetes sin solicitud

public List<Paquete> ListadoPaquetesSinSolicitud(Empleado usLog)

{

Paquete paquete = null;

Empresa empresa = null;

List<Paquete> listaPaquetes = new List<Paquete>();

SqlConnection conexion = null;

SqlDataReader drPaquete = null;

try

{

conexion = new SqlConnection(Conexion.Cnn(usLog));

SqlCommand cmdListadoSolicitues = new SqlCommand("ListarPaquetesSinSolicitud", conexion);

cmdListadoSolicitues.CommandType = CommandType.StoredProcedure;

conexion.Open();

drPaquete = cmdListadoSolicitues.ExecuteReader();

while (drPaquete.Read())

{

empresa = PersistenciaEmpresa.getInstancia().interBuscarEmpresa((string)drPaquete["empresa"]);

paquete = new Paquete((int)drPaquete["codigo"], (string)drPaquete["tipo"], (string)drPaquete["descripcion"],

(double)drPaquete["peso"], empresa);

listaPaquetes.Add(paquete);

}

return listaPaquetes;

}

catch (Exception ex)

{

throw ex;

}

}

CREATE PROCEDURE ListarPaquetesSinSolicitud

AS

BEGIN

SELECT \*

FROM Paquete

WHERE codigo not in (SELECT codigoPaquete FROM PaquetesSolicitud)

END

PersistenciaSolicitud.cs

internal class PersistenciaSolicitud : IPersistenciaSolicitud

{

private static PersistenciaSolicitud \_instancia = null;

private PersistenciaSolicitud() { }

public static PersistenciaSolicitud getInstancia()

{

if (\_instancia == null)

{

\_instancia = new PersistenciaSolicitud();

}

return \_instancia;

}

public void AltaSolicitud(Solicitud solicitud, Empleado usLog)

{

SqlConnection conexion = null;

SqlTransaction trn = null;

try

{

conexion = new SqlConnection(Conexion.Cnn(usLog));

SqlCommand cmdAltaSolicitud = new SqlCommand("AltaSolicitud", conexion);

cmdAltaSolicitud.CommandType = CommandType.StoredProcedure;

cmdAltaSolicitud.Parameters.AddWithValue("@fechaEntrega", solicitud.FechaEntrega);

cmdAltaSolicitud.Parameters.AddWithValue("@nombreDestinatario", solicitud.NombreDestinatario);

cmdAltaSolicitud.Parameters.AddWithValue("@direccionDestinatario", solicitud.DireccionDestinatario);

cmdAltaSolicitud.Parameters.AddWithValue("@estado", solicitud.Estado);

cmdAltaSolicitud.Parameters.AddWithValue("@empleado", solicitud.Empleado.Logueo);

SqlParameter valorRetorno = new SqlParameter("@valorRetorno", SqlDbType.Int);

valorRetorno.Direction = ParameterDirection.ReturnValue;

cmdAltaSolicitud.Parameters.Add(valorRetorno);

SqlParameter idSolicitud = new SqlParameter("@@IDENTITY", SqlDbType.Int);

idSolicitud.Direction = ParameterDirection.ReturnValue;

cmdAltaSolicitud.Parameters.Add(idSolicitud);

conexion.Open();

trn = conexion.BeginTransaction();

cmdAltaSolicitud.Transaction = trn;

cmdAltaSolicitud.ExecuteNonQuery();

solicitud.Numero = Convert.ToInt32(idSolicitud.Value);

switch ((int)valorRetorno.Value)

{

case -1:

throw new Exception("El empleado no esta activo.");

case -2:

throw new Exception("Error al insertar solicitud.");

}

foreach (Paquete paquete in solicitud.PaquetesSolicitud)

{

PersistenciaPaquete.getInstancia().AltaPaqueteSolicitud(trn, solicitud, paquete);

}

trn.Commit();

}

catch (Exception ex)

{

trn.Rollback();

throw ex;

}

finally

{

if (conexion != null)

{

conexion.Close();

}

}

}

public void ModificarEstadoSolicitud(Solicitud solicitud, Empleado usLog)

{

SqlConnection conexion = null;

try

{

conexion = new SqlConnection(Conexion.Cnn(usLog));

SqlCommand cmdModificarSolicitud = new SqlCommand("ModificarEstadoSolicitud", conexion);

cmdModificarSolicitud.CommandType = CommandType.StoredProcedure;

cmdModificarSolicitud.Parameters.AddWithValue("@numero", solicitud.Numero);

cmdModificarSolicitud.Parameters.AddWithValue("@empleado", solicitud.Empleado.Logueo);

SqlParameter valorRetorno = new SqlParameter("@valorRetorno", SqlDbType.Int);

valorRetorno.Direction = ParameterDirection.ReturnValue;

cmdModificarSolicitud.Parameters.Add(valorRetorno);

conexion.Open();

cmdModificarSolicitud.ExecuteNonQuery();

switch ((int)valorRetorno.Value)

{

case -1:

throw new Exception("El empleado no exíste o no está activo.");

case -2:

throw new Exception("Solicitud inexistente.");

case -3:

throw new Exception("Error al modificar estado.");

}

}

catch (Exception ex)

{

throw ex;

}

finally

{

if (conexion != null)

{

conexion.Close();

}

}

}

public List<Solicitud> listadoSolicitudesEnCamino(Usuario usLog)

{

SqlConnection conexion = null;

SqlDataReader drSolicitud = null;

Solicitud solicitud = null;

List<Solicitud> listaSolicitudes = new List<Solicitud>();

List<Paquete> listaPaquetes = new List<Paquete>();

Empleado empleado = null;

try

{

conexion = new SqlConnection(Conexion.Cnn(usLog));

SqlCommand cmdListadoSolicitues = new SqlCommand("listadoSolicitudesEnCamino", conexion);

cmdListadoSolicitues.CommandType = CommandType.StoredProcedure;

conexion.Open();

drSolicitud = cmdListadoSolicitues.ExecuteReader();

while (drSolicitud.Read())

{

listaPaquetes = PersistenciaPaquete.getInstancia().listadoPaquetesSolicitud((int)drSolicitud["numero"]);

empleado = PersistenciaEmpleado.getInstancia().interBuscarEmpleado((string)drSolicitud["empleado"]);

solicitud = new Solicitud((int)drSolicitud["numero"], (DateTime)drSolicitud["fechaEntrega"], (string)drSolicitud["nombreDestinatario"],

(string)drSolicitud["direccionDestinatario"], (string)drSolicitud["estado"], empleado, listaPaquetes);

listaSolicitudes.Add(solicitud);

}

return listaSolicitudes;

}

catch (Exception ex)

{

throw ex;

}

finally

{

if (conexion != null)

{

conexion.Close();

}

}

}

public List<Solicitud> listadoSolicitudesEmpresa(Empresa usLog)

{

SqlConnection conexion = null;

SqlDataReader drSolicitud = null;

Solicitud solicitud = null;

List<Solicitud> listaSolicitudes = new List<Solicitud>();

List<Paquete> listaPaquetes = new List<Paquete>();

Empleado empleado = null;

try

{

conexion = new SqlConnection(Conexion.Cnn(usLog));

SqlCommand cmdListadoSolicitues = new SqlCommand("ListadoSolicitudesEmpresa", conexion);

cmdListadoSolicitues.CommandType = CommandType.StoredProcedure;

conexion.Open();

drSolicitud = cmdListadoSolicitues.ExecuteReader();

while (drSolicitud.Read())

{

listaPaquetes = PersistenciaPaquete.getInstancia().listadoPaquetesSolicitud((int)drSolicitud["numero"]);

empleado = PersistenciaEmpleado.getInstancia().interBuscarEmpleado((string)drSolicitud["empleado"]);

solicitud = new Solicitud((int)drSolicitud["numero"], (DateTime)drSolicitud["fechaEntrega"], (string)drSolicitud["nombreDestinatario"],

(string)drSolicitud["direccionDestinatario"], (string)drSolicitud["estado"], empleado, listaPaquetes);

listaSolicitudes.Add(solicitud);

}

return listaSolicitudes;

}

catch (Exception ex)

{

throw ex;

}

finally

{

if (conexion != null)

{

conexion.Close();

}

}

}

}

--------------------------------------------SP Solicitud------------------------------------------

CREATE PROCEDURE AltaSolicitud

@fechaEntrega DATETIME,

@nombreDestinatario VARCHAR(50),

@direccionDestinatario VARCHAR(50),

@empleado VARCHAR(50)

AS

BEGIN

IF NOT EXISTS (SELECT \*

FROM Usuario a

inner join Empleado b on a.logueo = b.logueo

WHERE a.logueo = @empleado AND a.activo = 1 ) -- empleado activo?

BEGIN

RETURN -1

END

INSERT INTO Solicitud (fechaEntrega, nombreDestinatario, direccionDestinatario, empleado)

VALUES (@fechaEntrega, @nombreDestinatario, @direccionDestinatario, @empleado)

IF (@@ERROR <> 0)

BEGIN

RETURN -2

END

RETURN @@IDENTITY

END

GO

CREATE PROCEDURE listadoSolicitudesEnCamino

AS

BEGIN

SELECT \*

FROM Solicitud

WHERE estado = 'en camino'

END

GO

CREATE PROCEDURE ListadoSolicitudesEmpresa

@empresa varchar(12)

AS

BEGIN

SELECT a.\*

FROM Solicitud a

INNER JOIN PaquetesSolicitud b on a.numero = b.numeroSolicitud

INNER JOIN Paquete c on b.codigoPaquete = c.codigo

WHERE c.empresa = @empresa

END

GO

CREATE PROCEDURE ModificarEstadoSolicitud

@numero INT,

@empleado VARCHAR(50)

AS

BEGIN

IF NOT EXISTS (SELECT \*

FROM Usuario a

inner join Empleado b on a.logueo = b.logueo

WHERE a.logueo = @empleado AND a.activo = 1 )

BEGIN

RETURN -1

END

IF NOT EXISTS (SELECT \* FROM Solicitud WHERE numero = @numero)

BEGIN

RETURN -2

END

-- seria con todos los datos de la solicitud por parametro,

-- Y consulto el estado a travez de allí

-- seria con este ejemplo y solo el numero de solicitud como parametro?

UPDATE Solicitud

SET estado = CASE

WHEN estado = 'en deposito' then 'en camino'

WHEN estado = 'en camino' then 'entregado'

END,

empleado = @empleado -- Modifico el empleado? y los demas datos de la solicitud?

WHERE numero = @numero

IF (@@ERROR <> 0)

BEGIN

RETURN -3

END

END

Creación de roles y asignación de permisos.

CREATE ROLE db\_rol\_empleado

CREATE ROLE db\_rol\_empresa

CREATE ROLE db\_rol\_publico

USE master

GO

CREATE LOGIN [IIS APPPOOL\DefaultAppPool] FROM WINDOWS WITH DEFAULT\_DATABASE = ProyectoSegundo2020

GO

USE ProyectoSegundo2020

CREATE USER [IIS APPPOOL\DefaultAppPool] FOR LOGIN [IIS APPPOOL\DefaultAppPool]

GO

EXEC sp\_addrolemember [db\_rol\_publico], [IIS APPPOOL\DefaultAppPool]

Luego de la creación de sp

--publico

GRANT EXECUTE ON dbo.LogueoEmpleado TO [db\_rol\_publico]

GRANT EXECUTE ON dbo.LogueoEmpresa TO [db\_rol\_publico]

GRANT EXECUTE ON dbo.listadoSolicitudesEnCamino TO [db\_rol\_publico]

GRANT EXECUTE ON dbo.interBuscarEmpleado TO [db\_rol\_publico]

GRANT EXECUTE ON dbo.interBuscarEmpresa TO [db\_rol\_publico]

GRANT EXECUTE ON dbo.listadoPaquetesSolicitud TO [db\_rol\_publico]

--empleado

GRANT EXECUTE ON dbo.AltaEmpleado TO [db\_rol\_empleado]

GRANT EXECUTE ON dbo.BajaEmpleado TO [db\_rol\_empleado]

GRANT EXECUTE ON dbo.ModificarEmpleado TO [db\_rol\_empleado]

GRANT EXECUTE ON dbo.AltaEmpresa TO [db\_rol\_empleado]

GRANT EXECUTE ON dbo.BajaEmpresa TO [db\_rol\_empleado]

GRANT EXECUTE ON dbo.ModificarEmpresa TO [db\_rol\_empleado]

GRANT EXECUTE ON dbo.AltaPaquete TO [db\_rol\_empleado]

GRANT EXECUTE ON dbo.AltaSolicitud TO [db\_rol\_empleado]

GRANT EXECUTE ON dbo.AltaPaqueteSolicitud TO [db\_rol\_empleado]

GRANT EXECUTE ON dbo.ModificarEstadoSolicitud TO [db\_rol\_empleado]

GRANT EXECUTE ON dbo.ListarPaquetesLibres TO [db\_rol\_empleado]

GRANT EXECUTE ON dbo.BuscarEmpleado TO [db\_rol\_empleado]

GRANT EXECUTE ON dbo.BuscarEmpresa TO [db\_rol\_empleado]

GRANT EXECUTE ON dbo.BuscarPaquete TO [db\_rol\_empleado]

GRANT EXECUTE ON dbo.listarEmpresas TO [db\_rol\_empleado]

--empresa

GRANT EXECUTE ON dbo.ListadoSolicitudesEmpresa TO [db\_rol\_empresa]