Documenacion para el video

Tabla “TaskToDo”

El Id es auto-increment ado

ROLEYDER-PC\SQLEXP...askToDo - dbo.Task

Column Name	Data Type	Allow Nulls
Id_Task	int	☐
Cod_Task	nchar(11)	☐
Descrip_Task	nvarchar(200)	☐
Date_StartTask	datetime	☐
Date_EndTask	datetime	☐
Status_Task	nchar(1)	☐
		☐

Función para generar el código de la tarea

```
USE [TaskToDo]
GO
/****** Object:  UserDefinedFunction [dbo].[fnGenCodTask]    Script Date: 4/14/2019 9:35:51 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER function [dbo].[fnGenCodTask] ()
returns nchar(6)
as
begin
declare @CodTask nchar(6)
 set @CodTask = (select max(Cod_Task) from Task)
 set @CodTask = 'TASK' + RIGHT('0'+LTRIM(RIGHT(ISNULL(@CodTask,'0'),2)+1),2)
return @CodTask
end
```

Procedimiento almacenado Generar para las acciones

```sql
USE [TaskToDo]
GO
/****** Object:  StoredProcedure [dbo].[spInsElimEditTask]    Script Date: 4/14/2019 9:38:21 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER proc [dbo].[spInsElimEditTask]
@prmCadxml nvarchar(max)
as
begin
declare @h int, @smsError varchar(500)
exec SP_XML_PREPAREDOCUMENT @h output, @prmCadxml
begin try
begin transaction
--Insert a new task
insert into Task(Cod_Task, Descrip_Task, Date_StartTask, Date_EndTask, Status_Task)
select dbo.fnGenCodTask(), ta.descriptask, ta.datestarttask, ta.dateendtask, ta.statustask
from OpenXml(@h, 'root/task',1)with(
idtask int,
descriptask nvarchar(200),
datestarttask datetime,
dateendtask datetime,
statustask nchar(1),

tipoedicion int
) ta where tipoedicion=1

--update a task


--update a task

update task
   set
   task.Descrip_Task = ta.descriptask,
   task.Date_StartTask = ta.datestarttask,
   task.Date_EndTask = ta.dateendtask,
   task.Status_Task = ta.statustask
   from OpenXml(@h, 'root/task',1)with(
   idtask int,
   descriptask nvarchar(200),
   datestarttask datetime,
   dateendtask datetime,
   statustask nchar(1),
   tipoedicion int
   ) ta inner join Task task on ta.idtask = task.Id_Task where ta.tipoedicion=2

   --delete a task
   update task
   set task.Status_Task='B'
   from OpenXml(@h, 'root/task',1)with(
   idtask int,
   tipoedicion int
   ) ta inner join Task tas on ta.idtask=tas.Id_Task where ta.tipoedicion=3

   if (@@TRANCOUNT > 0) COMMIT TRANSACTION
   end try
   begin catch
   if (@@TRANCOUNT >0) ROLLBACK TRANSACTION
   select @smsError = ERROR_MESSAGE()
   RAISERROR (@smsError,16,1)
   end catch
   end
```

Procedimiento almacenado para dar por hecha una tarea

```
USE [TaskToDo]
GO
/****** Object:  StoredProcedure [dbo].[spAnularTask]    Script Date: 4/14/2019 9:41:25 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER proc [dbo].[spAnularTask]
@prmId_Task int
as
begin
update Task set Status_Task ='B'
where Id_Task = @prmId_Task
end
```

Procedimiento almacenado para buscar una tarea por código

```
USE [TaskToDo]
GO
/****** Object:  StoredProcedure [dbo].[spBuscarTask]    Script Date: 4/14/2019 9:42:00 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER proc [dbo].[spBuscarTask]
@prmCodTask nchar(6)
as
begin
set nocount on
begin
select * from Task
where Cod_Task = @prmCodTask
end
set nocount off
end
```

Procedimiento almacenado para buscar una tarea por su respectivo id

```
USE [TaskToDo]
GO
/****** Object:  StoredProcedure [dbo].[spBuscartask2]    Script Date: 4/14/2019 9:43:24 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER proc [dbo].[spBuscartask2]
@prmidTask int
as

begin
set nocount on
 select Id_Task, Cod_Task, Descrip_Task, Date_StartTask, Date_EndTask, Status_Task from Task
where Id_Task = @prmidTask

set nocount off
end
```

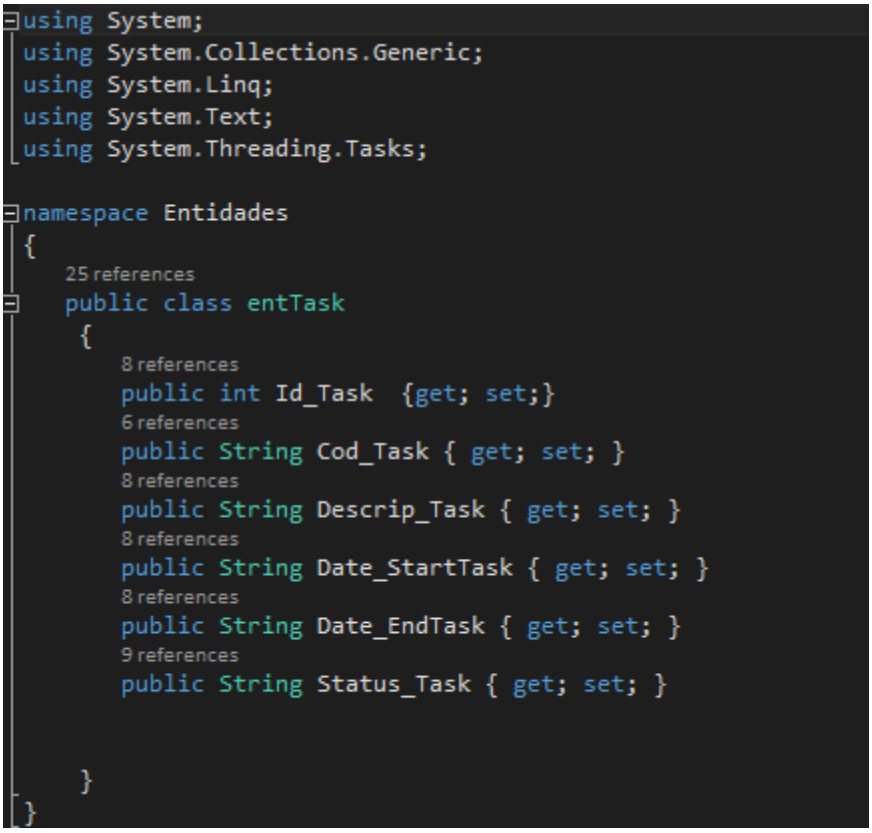
Procedimiento almacenado para listar las tareas

```
USE [TaskToDo]
GO
/****** Object:  StoredProcedure [dbo].[spListarTask]    Script Date: 4/14/2019 9:44:35 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER proc [dbo].[spListarTask]
as
begin
set nocount on
select Id_Task, Cod_Task, Descrip_Task, Date_StartTask, Date_EndTask, Status_Task  from Task
set nocount off
end
```

VISUAL STDUIO

CapaEntidades

Class: entTask



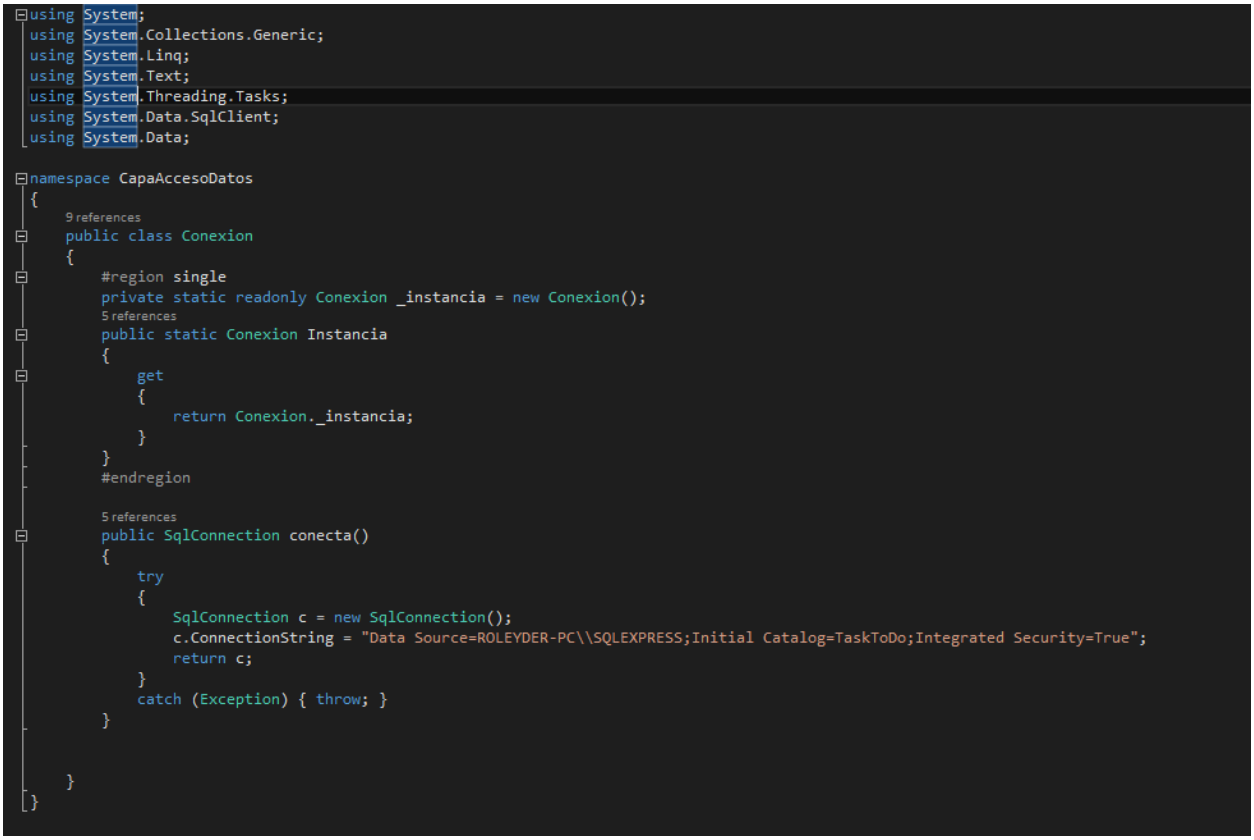
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Entidades
{
    25 references
    public class entTask
    {
        8 references
        public int Id_Task  {get; set;}
        6 references
        public String Cod_Task { get; set; }
        8 references
        public String Descrip_Task { get; set; }
        8 references
        public String Date_StartTask { get; set; }
        8 references
        public String Date_EndTask { get; set; }
        9 references
        public String Status_Task { get; set; }


    }
}
```

CapaAccesoDatos

Class: Conexion



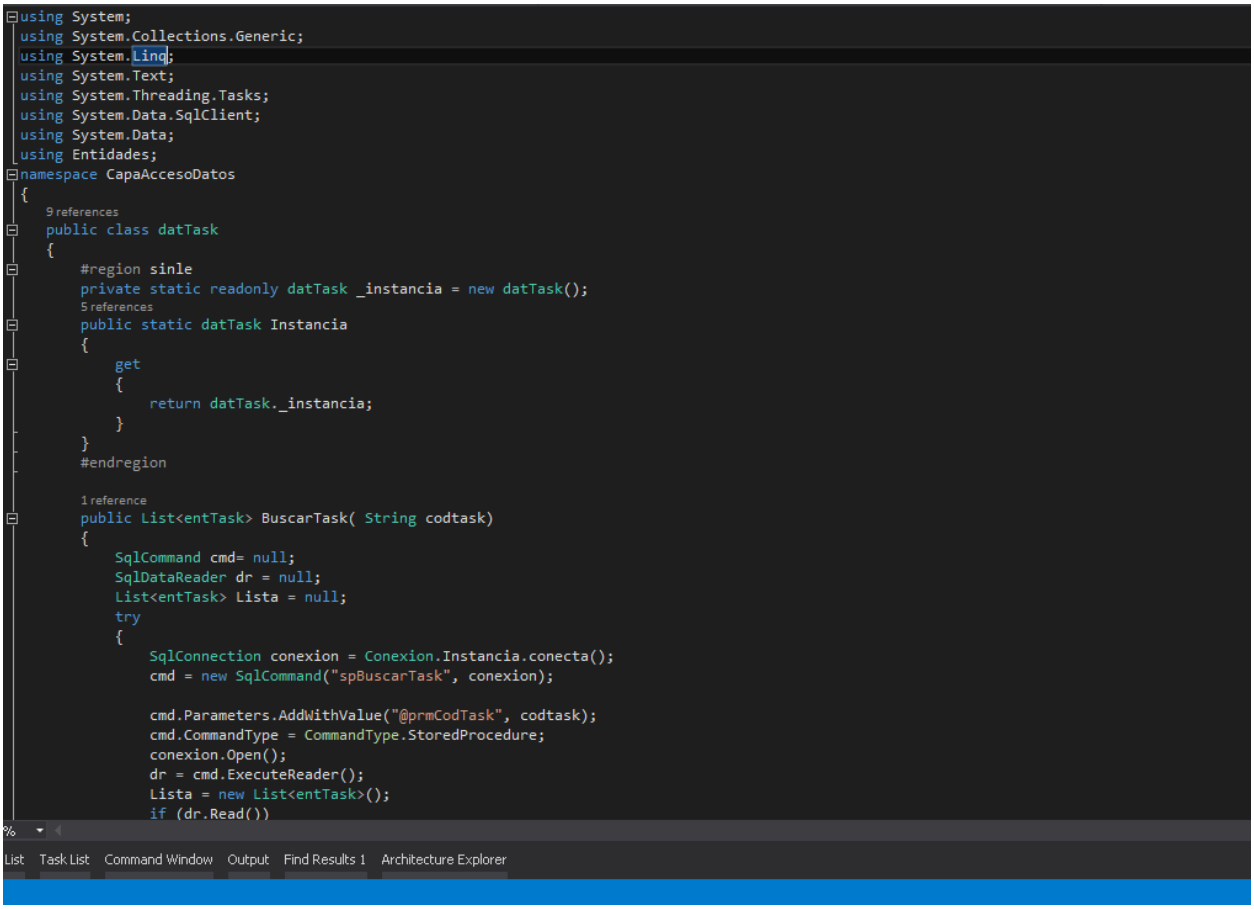
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.SqlClient;
using System.Data;

namespace CapaAccesoDatos
{
    9 references
    public class Conexion
    {
        #region single
        private static readonly Conexion _instancia = new Conexion();
        5 references
        public static Conexion Instancia
        {
            get
            {
                return Conexion._instancia;
            }
        }
        #endregion

        5 references
        public SqlConnection conecta()
        {
            try
            {
                SqlConnection c = new SqlConnection();
                c.ConnectionString = "Data Source=ROLEYDER-PC\\SQLEXPRESS;Initial Catalog=TaskToDo;Integrated Security=True";
                return c;
            }
            catch (Exception) { throw; }
        }


    }
}
```

Class: datTask



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.SqlClient;
using System.Data;
using Entidades;
namespace CapaAccesoDatos
{
    9 references
    public class datTask
    {
        #region sinle
        private static readonly datTask _instancia = new datTask();
        5 references
        public static datTask Instancia
        {
            get
            {
                return datTask._instancia;
            }
        }
        #endregion

        1 reference
        public List<entTask> BuscarTask( String codtask)
        {
            SqlCommand cmd= null;
            SqlDataReader dr = null;
            List<entTask> Lista = null;
            try
            {
                SqlConnection conexion = Conexion.Instancia.conecta();
                cmd = new SqlCommand("spBuscarTask", conexion);

                cmd.Parameters.AddWithValue("@prmCodTask", codtask);
                cmd.CommandType = CommandType.StoredProcedure;
                conexion.Open();
                dr = cmd.ExecuteReader();
                Lista = new List<entTask>();
                if (dr.Read())
```

```
public List<entTask> BuscarTask( String codtask)
{
    SqlCommand cmd= null;
    SqlDataReader dr = null;
    List<entTask> Lista = null;
    try
    {
        SqlConnection conexion = Conexion.Instancia.conecta();
        cmd = new SqlCommand("spBuscarTask", conexion);

        cmd.Parameters.AddWithValue("@prmCodTask", codtask);
        cmd.CommandType = CommandType.StoredProcedure;
        conexion.Open();
        dr = cmd.ExecuteReader();
        Lista = new List<entTask>();
        if (dr.Read())
        {
           entTask task = new entTask();
            task.Id_Task = Convert.ToInt32(dr["Id_Task"].ToString());
            task.Cod_Task =  dr["Cod_Task"].ToString();
            task.Descrip_Task = dr["Descrip_Task"].ToString();
            task.Date_StartTask = dr["Date_StartTask"].ToString();
            task.Date_EndTask= dr["Date_EndTask"].ToString();
            task.Status_Task = dr["Status_Task"].ToString();
            Lista.Add(task);
        }
    }

    catch(Exception) { throw;}
    finally {cmd.Connection.Close();}
    return Lista;
}
```

```
public List<entTask> listartask()
{
     SqlCommand cmd= null;
    SqlDataReader dr = null;
    List<entTask> Lista = null;
    try
    {
        SqlConnection conexion = Conexion.Instancia.conecta();
        cmd = new SqlCommand("spListarTask", conexion);

        cmd.CommandType = CommandType.StoredProcedure;
        conexion.Open();
        dr = cmd.ExecuteReader();
        Lista = new List<entTask>();
        while (dr.Read())
        {
            entTask task = new entTask();
            task.Id_Task = Convert.ToInt32(dr["Id_Task"].ToString());
            task.Cod_Task =  dr["Cod_Task"].ToString();
            task.Descrip_Task = dr["Descrip_Task"].ToString();
            task.Date_StartTask = dr["Date_StartTask"].ToString();
            task.Date_EndTask= dr["Date_EndTask"].ToString();
            task.Status_Task = dr["Status_Task"].ToString();
            Lista.Add(task);
        }
    }

    catch(Exception) { throw;}
    finally {cmd.Connection.Close();}
    return Lista;
}
```

```
public int taskrealizada(int idtask)
{
     SqlCommand cmd = null;
     var retorno = 0;
     try
     {
         SqlConnection cn = Conexion.Instancia.conecta();
         cmd = new SqlCommand("spAnularTask", cn);
         cmd.Parameters.AddWithValue("@prmId_Task", idtask);
         cmd.CommandType = CommandType.StoredProcedure;
         cn.Open();
         retorno = cmd.ExecuteNonQuery();
         return retorno;
     }
     catch (Exception) { throw; }
     finally { cmd.Connection.Close(); }
}
```

```
public int mantenimientotask(String cadxml)
{
    SqlCommand cmd = null;
    var resultado = 0;
    try
    {
        SqlConnection c = Conexion.Instancia.conecta();
        cmd = new SqlCommand("spInsElimEditTask", c);
        cmd.Parameters.AddWithValue("@prmCadxml", cadxml);
        cmd.CommandType = CommandType.StoredProcedure;
        c.Open();
        resultado = cmd.ExecuteNonQuery();
        return resultado;
    }
    catch (Exception) {throw;}
    finally {cmd.Connection.Close();}
}
```

```
        1 reference
        public entTask BuscarTask2(int idtask)
        {
            SqlCommand cmd = null;
            SqlDataReader dr = null;
            entTask task = null;

            try
            {
                SqlConnection cn = Conexion.Instancia.conecta();
               cmd = new SqlCommand("spBuscarTask2", cn);
               cmd.Parameters.AddWithValue("@prmidTask", idtask);
               cmd.CommandType = CommandType.StoredProcedure;
               cn.Open();
              dr = cmd.ExecuteReader();
             if(dr.Read())
             {
                  task = new entTask();
                  task.Id_Task = Convert.ToInt32(dr["Id_Task"]);
                task.Cod_Task = dr["Cod_Task"].ToString();
                task.Descrip_Task = dr["Descrip_Task"].ToString();
                task.Date_StartTask = dr["Date_StartTask"].ToString();
                task.Date_EndTask = dr["Date_EndTask"].ToString();
                task.Status_Task = dr["Status_Task"].ToString();
             }
            }
            catch (Exception) { throw; }
           finally {cmd.Connection.Close();}
            return task;
        }
    }
}
```

CapaNegocio

Class: negTask

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

namespace CapaNegocio
{
    9 references
    public class negTask
    {
        #region single
        private static readonly negTask _instancia = new negTask();
        5 references
        public static negTask Instancia
        {
            get { return negTask._instancia; }
        }
        #endregion
```

```
public int mantenimientotask(entTask t, int tipoedicion)
{
    try
    {
        String CadXml = "";
        CadXml += "<task ";
        CadXml += "idtask='" + t.Id_Task + "' ";
        CadXml += "descriptask='" + t.Descrip_Task + "' ";
        CadXml += "datestarttask='" + t.Date_StartTask + "' ";
        CadXml += "dateendtask='" + t.Date_EndTask + "' ";
        CadXml += "statustask='" + t.Status_Task + "' ";
        CadXml += "tipoedicion='" + tipoedicion + "' /> ";

        CadXml = "<root>" + CadXml + "</root>";
        int resultado = datTask.Instancia.mantenimientotask(CadXml);
        if (resultado <= 0) throw new ApplicationException("Error al cargar las tareas");
        return resultado;


    }
    catch (Exception) { throw;}

}
```

```
public int taskrealizada(int idtask)
{
    try
    {
        int retorno = datTask.Instancia.taskrealizada(idtask);
        if (retorno == 0) throw new ApplicationException("Error al anular la tarea");
        return retorno;
    }
    catch (Exception) { throw; }
}
```

```
public List<entTask> ListarTask()
{
    try
    {
        List<entTask> Lista = datTask.Instancia.listartask();
        if (Lista.Count <= 0) throw new ApplicationException("Lista de las tareas vacias");
        else if (Lista == null) throw new ApplicationException("Error al cargar las task");
        return Lista;
    }
    catch (Exception) { throw; }
}
```

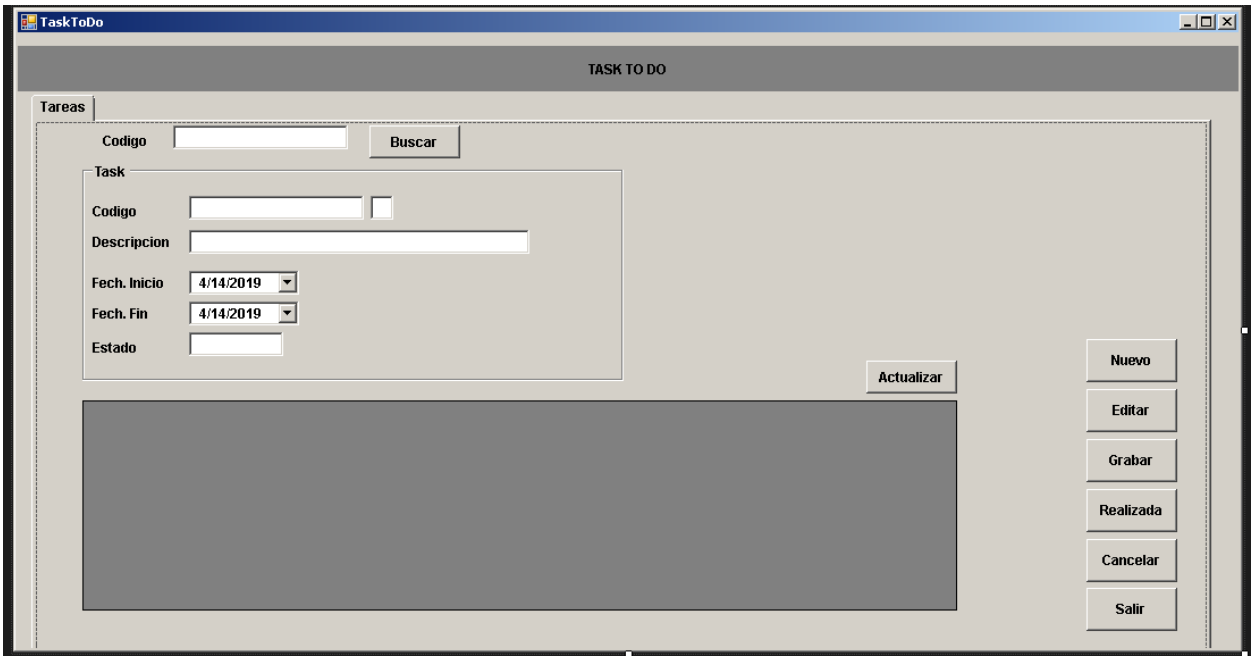
```
1 reference
public List<entTask> BuscarTask(string codtask)
{
    try
    {
        List<entTask> Lista = null;
        Lista = datTask.Instancia.BuscarTask( codtask);
        if (Lista == null) throw new ApplicationException("La tareano no existe");
        return Lista;
    }
    catch (Exception) { throw; }
}
```

```
1 reference
public entTask BuscarTask2(int idtask)
{
    try
    {
        entTask task = null;
        task = datTask.Instancia.BuscarTask2(idtask);
        if (task == null) throw new ApplicationException("No hay registros");
        return task;
    }
    catch (Exception) { throw;}



}
}
}
```

CapaPresentacion



Class: AccionEnControles

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace CapaPresentacion
{
    2 references
    public  class AccionBotones
     {
         1 reference
        public void limpiartxt(Control control)
        {
            try
            {
                foreach (Control txt in control.Controls)
                {
                    if (txt is TextBox)
                    {


                            ((TextBox)txt).Clear();

                    }
                    else if (txt is GroupBox)
                    {
                        foreach (Control txtgb in txt.Controls)
                        {
                            if (txtgb is TextBox)
                            {
                                ((TextBox)txtgb).Clear();
                            }
                        }
                    }
                }
            }
            catch (Exception) { throw; }
        }


}
```

}

```
4 references
public void bloqueardtp(Control control, Boolean estado)
{
    try
    {
        foreach (Control dtp in control.Controls)
        {
            if (dtp is GroupBox)
            {
                foreach (Control h in dtp.Controls)
                {
                    if(h is DateTimePicker)
                    {
                        ((DateTimePicker)h).Enabled = estado;
                    }
                }
            }
        }
    }
    catch(Exception) {throw;}
}
```

```
public void bloquartxt(Control control, Boolean estado)
{
    try
    {
        foreach (Control txt in control.Controls)
        {
            if (txt is TextBox)
            {
                ((TextBox)txt).Enabled = estado;
                if (txt.Name == "txtCodigoTask" || txt.Name == "txtIdTask" || txt.Name == "txtEstadoTask" )
                {
                    ((TextBox)txt).Enabled = false;
                }

                if (txt.Name == "txtBuscarTask")
                {
                    ((TextBox)txt).Enabled = true;
                }
            }
             else if (txt is GroupBox)
            {
                foreach (Control txtgb in txt.Controls)
                {
                    if (txtgb is TextBox)
                    {
                        ((TextBox)txtgb).Enabled = estado;
                        if (txtgb.Name == "txtCodigoTask" || txtgb.Name == "txtIdTask" || txtgb.Name == "txtEstadoTask")
                        {
                            ((TextBox)txtgb).Enabled = false;
                        }
                    }
                }
                 }
        }

    }
    catch (Exception) { throw; }
}

k List  Command Window  Output  Find Results 1  Architecture Explorer

Ln 94
```

FormTaskToDo

```
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.IO;
using System.Threading.Tasks;
using System.Windows.Forms;
using Entidades;
using CapaNegocio;

namespace CapaPresentacion
{
    3 references
    public partial class frmTask : Form
    {


        1 reference
        public frmTask()
        {
            InitializeComponent();
        }
        AccionBotones accion = new AccionBotones();

        6 references
        private void AccionBotones(Boolean nuevo, Boolean grabar ,Boolean realizar, Boolean cancelar, Boolean salir)
        {
            btnNuevo.Enabled = nuevo;
            btnGrabar.Enabled = grabar;
            btnRealizada.Enabled = realizar;
            btnCancelar.Enabled = cancelar;
            btnSalir.Enabled = salir;
        }
```

```
private void CrearGrid()
{
    try
    {
        dgvTask.Columns.Add("ColumnId", "Id");
        dgvTask.Columns.Add("ColumnNum", "#");
        dgvTask.Columns.Add("ColumnCodigo", "Codigo");
        dgvTask.Columns.Add("ColumnDescrip", "Descripcion");
        dgvTask.Columns.Add("ColumnFechInicio", "Fecha Inicio");
        dgvTask.Columns.Add("ColumnFechFin", "Fecha Fin");
        dgvTask.Columns.Add("ColumnEstado", "Estado");

        dgvTask.Columns[0].Visible = false;
        dgvTask.Columns[1].AutoSizeMode = DataGridViewAutoSizeColumnMode.AllCells;
        dgvTask.Columns[2].AutoSizeMode = DataGridViewAutoSizeColumnMode.AllCells;

        DataGridViewCellStyle css = new DataGridViewCellStyle();
        css.Alignment = DataGridViewContentAlignment.MiddleCenter;
        dgvTask.ColumnHeadersDefaultCellStyle = css;

        dgvTask.AllowUserToAddRows = false;
        dgvTask.MultiSelect = false;
        dgvTask.SelectionMode = DataGridViewSelectionMode.FullRowSelect;
    }
    catch (Exception) { throw; }
}
```

```
private void frmTask_Load(object sender, EventArgs e)
{

    CrearGrid();
    LlenarGrip();
    AccionBotones(true, false, false, true, false);
    accion.bloquartxt(this.tabPage1, false);
    accion.bloqueardtp(this.tabPage1, false);

}
```

```
3 references
private void LlenarGrip()
{
    try
    {
        int num = 0;
        dgvTask.Rows.Clear();
        List<entTask> Lista = negTask.Instancia.ListarTask();
        for (int i = 0; i < Lista.Count; i++)
        {


            num++;
            String[] fila = new string[] { Lista[i].Id_Task.ToString(), num.ToString(),
                Lista[i].Cod_Task, Lista[i].Descrip_Task, Lista[i].Date_StartTask.ToString(), Lista[i].Date_EndTask.ToString(),
                Lista[i].Status_Task };
            dgvTask.Rows.Add(fila);
        }
    }
    catch (Exception) { throw; }
}
```

```
private void btnActualizar_Click_2(object sender, EventArgs e)
{
    LlenarGrip();
}
```

```
private void btnGrabar_Click_2(object sender, EventArgs e)
{
    try
    {
        entTask t = new entTask();
        int tipoedicion = 1;
        if (txtIdTask.Text != "") { tipoedicion = 2; t.Id_Task = Convert.ToInt32(txtIdTask.Text); }

        t.Descrip_Task = txtDescriptTask.Text;
        t.Date_StartTask = Convert.ToString(dtpFechInicioTask.Value.ToString("yyyy/MM/dd"));
        t.Date_EndTask = Convert.ToString(dtpFechFinTask.Value.ToString("yyyy/MM/dd"));
        t.Status_Task = "A";
        int r = negTask.Instancia.mantenimientotask(t, tipoedicion);
        LlenarGrip();
        MessageBox.Show("Tarea insertada correctamente", "Mensaje", MessageBoxButtons.OK, MessageBoxIcon.Information);
       AccionBotones(true, false, false, true, false);
    accion.bloquartxt(this.tabPage1, false);
    }
    catch (Exception)
    {
        throw;
    }
}
```

```
private void dgvTask_CellClick_2(object sender, DataGridViewCellEventArgs e)
{
    try
    {
        int idtask = Convert.ToInt32(dgvTask.CurrentRow.Cells[0].Value);
        entTask task = null;
        task = negTask.Instancia.BuscarTask2(idtask);
        txtIdTask.Text = task.Id_Task.ToString();
        txtCodigoTask.Text = task.Cod_Task.ToString();
        txtDescriptTask.Text = task.Descrip_Task.ToString();
        dtpFechInicioTask.Value = Convert.ToDateTime(task.Date_StartTask);
        dtpFechFinTask.Value = Convert.ToDateTime(task.Date_EndTask);
        txtEstadoTask.Text = task.Status_Task.ToString();
        txtBuscarTask.Text = "";
        if (task.Status_Task == "B") txtEstadoTask.BackColor = Color.Red;
        else txtEstadoTask.BackColor = Color.Green;
        AccionBotones(true, false, false, true, true);
        accion.bloquartxt(this.tabPage1, false);
        accion.bloqueardtp(this.tabPage1, false);



    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

```
private void btnNuevo_Click_1(object sender, EventArgs e)
{
    accion.limpiartxt(this.tabPage1);
     accion.bloquartxt(this.tabPage1, true);
    AccionBotones(false, true, false, true, true);
    accion.bloqueardtp(this.tabPage1, true);
}

1 reference
private void btnCancelar_Click(object sender, EventArgs e)
{
    accion.bloquartxt(this.tabPage1, true);
    AccionBotones(true, false, true, false, true);
}

1 reference
private void txtBuscarTask_TextChanged(object sender, EventArgs e)
{

}

1 reference
private void btnEditar_Click(object sender, EventArgs e)
{
    try
    {
        accion.bloquartxt(this.tabPage1, true);
        accion.bloqueardtp(this.tabPage1, true);
        AccionBotones(false, true, true, true, true);
    }
    catch (Exception)
    {
        throw;
    }
```

}

```
1 reference
private void btnRealizada_Click(object sender, EventArgs e)
{
    try
    {
        //Almacenamos el id en una variable
        int idtask = Convert.ToInt32(dgvTask.CurrentRow.Cells[0].Value);
        DialogResult res = MessageBox.Show("¿Desea dar por realizada esta tarea?", "Aviso", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
        if (res==DialogResult.Yes)
        {
            int i = negTask.Instancia.taskrealizada(idtask);
            MessageBox.Show("La tarea ya fue dada como realizada", "bien", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }

    }
    catch (Exception) { throw; }
}
```

```
private void txtBuscarTask_KeyUp(object sender, KeyEventArgs e)
{
    try
    {
        dgvTask.Rows.Clear();
        int numero = 0;
        String codigo = txtBuscarTask.Text;
        List<entTask> Lista = negTask.Instancia.BuscarTask(codigo);
        for (int i =0; i < Lista.Count; i++)
        {
            numero++;
            String[] fila = new String[] { Lista[i].Id_Task.ToString(), numero.ToString(), Lista[i].Cod_Task, Lista[i].Descrip_Task,
                Lista[i].Date_StartTask.ToString(),
                Lista[i].Date_EndTask.ToString(), Lista[i].Status_Task };
            dgvTask.Rows.Add(fila);

        }

    }
    catch (Exception) { throw; }
}
```

```
1 reference
private void btnSalir_Click(object sender, EventArgs e)
{
    DialogResult resu = MessageBox.Show("¿Esta seguro de que quiere salir?", "pregunta", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
    if (resu==DialogResult.Yes)
    {
        Application.Exit();
    }
}
}
}
```

Fin!!!