ENTREGA PLAN MEJORAMIENTO

Realizar Diseño orientado a objetos en lenguaje de programación .NET

Presentado por:

Dagnober Castañeda cubillos

Aprendiz ficha 2184587

Presentado a:

Jesús Ropero Barbosa

Instructor Teleinformática

SERVICIO NACIONAL DE APRENDIZAJE SENA

REGIONAL DISTRITO CAPITAL

CENTRO GESTION DE MERCADOS, LOGISTICA Y TI

ANÁLISIS Y DESARROLLO DE SISTEMAS DE INFORMACION

- PROGRAM.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using PlanM.Modelo;


namespace PlanM
{
    class Program
    {

        static List<Empleados> Empleados = new List<Empleados>();
        static Validaciones Validar = new Validaciones();

        static void Main(string[] args)
        {
            int menu;
            string aux;
            bool entradaValida = false;

            do
            {
                Console.SetCursorPosition(45, 8);
Console.WriteLine("Bienvenidos....");

                Console.SetCursorPosition(45, 10); Console.WriteLine("1.) Agregar
Empleados");
                Console.SetCursorPosition(45, 11); Console.WriteLine("2.) Buscar
Empleados");
                Console.SetCursorPosition(45, 12); Console.WriteLine("3.) Listar
Empleados");

                Console.SetCursorPosition(45, 14); Console.WriteLine("0.) Salir...
");

                do
                {
                    Console.SetCursorPosition(47, 18); Console.Write("╔");
                    Console.SetCursorPosition(52, 18); Console.Write("╗");
                    Console.SetCursorPosition(47, 20); Console.Write("╚");
                    Console.SetCursorPosition(52, 20); Console.Write("╝");
                    Console.SetCursorPosition(40, 16); Console.WriteLine("Escoja una
opcion");
                    Console.SetCursorPosition(50, 19); aux = Console.ReadLine();
                    if (!Validar.Vacio(aux))
                        if (Validar.TipoNumero(aux))
                            entradaValida = true;
                } while (!entradaValida);

                menu = Convert.ToInt32(aux);

                switch (menu)
                {
                    case 1:
                        AgregarEmpleado();
                        break;
```

```
                    case 2:
                        BuscarEmpleado();
                        break;
                    case 3:
                        ListarEmpleados();
                        break;
                    case 0:
                        Console.WriteLine("Gracias y hasta luego !...");
                        break;
                    default:
                        Console.WriteLine("Opcion invalida");
                        break;
                }
            } while (menu > 0);
        }

        static void AgregarEmpleado()
        {

            Console.Clear();
            var Db = new planmejoContext();
            string Nombre, Cedula, Salario, DiasVacaciones;
            int Cedula1, Salario1, DiasVacaciones1, divi, totalPagar;

            bool cedVal = false;
            bool nomVal = false;
            bool salVal = false;
            bool dvVal = false;


            Console.Clear();
            Console.SetCursorPosition(40, 5);
Console.WriteLine("_____________________________________");
            Console.SetCursorPosition(40, 6); Console.WriteLine("         Ingrese
datos          ");
            Console.SetCursorPosition(40, 7);
Console.WriteLine("_____________________________________");


            do
            {

                Console.SetCursorPosition(20, 10); Console.WriteLine("Digite cedula
del nuevo empleado: ");
                Console.SetCursorPosition(60, 10); Cedula = Console.ReadLine();
                if (!Validar.Vacio(Cedula))
                    if (Validar.TipoNumero(Cedula))
                        cedVal = true;
            } while (!cedVal);
            Cedula1 = Convert.ToInt32(Cedula);

            do
            {

                Console.SetCursorPosition(20, 11); Console.WriteLine("Digite el
nombre del empleado: ");
                Console.SetCursorPosition(60, 11); Nombre = Console.ReadLine();
                if (!Validar.Vacio(Nombre))
```

```
                    if (Validar.TipoTexto(Nombre))
                        nomVal = true;
            } while (!nomVal);

            do
            {

                Console.SetCursorPosition(20, 12); Console.WriteLine("Digite sueldo
del empleado: ");
                Console.SetCursorPosition(60, 12); Salario = Console.ReadLine();
                if (!Validar.Vacio(Salario))
                    if (Validar.TipoNumero(Salario))
                        salVal = true;
            } while (!salVal);
            Salario1 = Convert.ToInt32(Salario);

            do
            {

                Console.SetCursorPosition(20, 13); Console.WriteLine("Digite dias de
vacaciones del empleado : ");
                Console.SetCursorPosition(60, 13); DiasVacaciones =
Console.ReadLine();
                if (!Validar.Vacio(DiasVacaciones))
                    if (Validar.TipoNumero(DiasVacaciones))
                        dvVal = true;
            } while (!dvVal);
            DiasVacaciones1 = Convert.ToInt32(DiasVacaciones);

            divi = Salario1 / 30;
            totalPagar = divi * DiasVacaciones1;


            Empleados AUX = new Empleados();
            AUX.Cedula = (uint)Convert.ToInt32(Cedula);
            AUX.Nombre = Nombre;
            AUX.Salario = (int)Convert.ToInt32(Salario);
            AUX.DiasTrabajados = (int)Convert.ToInt32(DiasVacaciones);
            AUX.VacacionesPagar = (int)Convert.ToInt32(totalPagar);

            Db.Empleados.Add(AUX);
            Empleados.Add(AUX);
            Db.SaveChanges();

            Console.Clear();
        }

        static void BuscarEmpleado()
        {
            Console.Clear();
            var db = new planmejoContext();
            var empleados = db.Empleados.ToList();
            string cedula;
            bool cedVal = false;

            do
            {
                Console.Clear();
```

```
                Console.SetCursorPosition(35, 5); Console.WriteLine("BUSCAR UN
EMPLEADO...");
                Console.SetCursorPosition(35, 10); Console.WriteLine("Digite la
cedula a buscar: ");
                Console.SetCursorPosition(40, 15); cedula = (Console.ReadLine());
                if (!Validar.Vacio(cedula))
                    if (Validar.TipoNumero(cedula))
                        cedVal = true;
            } while (!cedVal);

            if (Existe(Convert.ToInt32(cedula)))
            {

                Console.SetCursorPosition(38, 5); Console.WriteLine("Empleado
encontrado...");

                Empleados myEmpleado = ObtenerDatos(Convert.ToInt32(cedula));

                Console.SetCursorPosition(38, 8); Console.WriteLine("Cedula: " +
myEmpleado.Cedula);
                Console.SetCursorPosition(38, 9); Console.WriteLine("Nombre: " +
myEmpleado.Nombre);
                Console.SetCursorPosition(38, 10); Console.WriteLine("Salario: " +
myEmpleado.Salario);
                Console.SetCursorPosition(38, 11); Console.WriteLine("Dias de
vacaciones:" + myEmpleado.DiasTrabajados);
                Console.SetCursorPosition(38, 12); Console.WriteLine("Vacaciones a
Pagar: " + myEmpleado.VacacionesPagar);

                Console.SetCursorPosition(30, 20); Console.WriteLine("Presione una
tecla para continnuar");
                Console.ReadKey();

            }
            else
            {
                Console.WriteLine("El empleado no existe... ");
                Console.ReadKey();
            }
            Console.WriteLine("Presiona una tecla para continuar...");
            Console.ReadKey();
            Console.Clear();
        }

        static void ListarEmpleados()
        {
            Console.Clear();
            var db = new planmejoContext();
            var empleados = db.Empleados.ToList();
            int y = 10;
            Console.SetCursorPosition(2, 5); Console.WriteLine("...Listar
Empleados...");

            Console.SetCursorPosition(2, y); Console.WriteLine("Cedula: ");
            Console.SetCursorPosition(15, y); Console.WriteLine("Nombre: ");
            Console.SetCursorPosition(35, y); Console.WriteLine("Salarios: ");
            Console.SetCursorPosition(45, y); Console.WriteLine("Dias: ");
```

```
            Console.SetCursorPosition(55, y); Console.WriteLine("Total pago
vacaciones: ");

            foreach (var myEmpleado in empleados)

            {
                y++;
                Console.SetCursorPosition(2, y);
Console.WriteLine(myEmpleado.Cedula);
                Console.SetCursorPosition(15, y);
Console.WriteLine(myEmpleado.Nombre);
                Console.SetCursorPosition(35, y);
Console.WriteLine(myEmpleado.Salario);
                Console.SetCursorPosition(45, y);
Console.WriteLine(myEmpleado.DiasTrabajados);
                Console.SetCursorPosition(55, y);
Console.WriteLine(myEmpleado.VacacionesPagar);

            }

            Console.WriteLine("\n");

            Console.SetCursorPosition(10, 20); Console.WriteLine("Presione una tecla
para volver al menu principal");
            Console.ReadKey();
            Console.Clear();


        }

        static bool Existe(int cedula)
        {
            Console.Clear();
            var db = new planmejoContext();
            var empleados = db.Empleados.ToList();
            bool aux = false;
            foreach (var myEmpleado in empleados)
            {
                if (myEmpleado.Cedula == cedula)
                    aux = true;
            }
            return aux;
        }

        static Empleados ObtenerDatos(int cedula)
        {
            var db = new planmejoContext();
            var empleados = db.Empleados.ToList();
            foreach (Empleados ObjetoEmpleado in empleados)
            {
                if (ObjetoEmpleado.Cedula == cedula)
                    return ObjetoEmpleado;
            }
            return null;
        }

    }
}
```

- VALIDACIONES.CS

```
using System;

using System.Text.RegularExpressions;

namespace PlanM
{
    class Validaciones
    {
        public bool Vacio(string texto)
        {
            if (texto.Equals(""))
            {
                Console.WriteLine(" .... Deje de dar ENTER ....");
                return true;
            }
            else
                return false;
        }

        public bool TipoNumero(string texto)
        {
            Regex regla = new Regex("[0-9]{1,9}(\\.[0-9]{0,2})?$");

            if (regla.IsMatch(texto))
                return true;
            else
            {
                Console.WriteLine(".....No Ingrese LETRAS !!!! .....");
                return false;
            }

        }


        public bool TipoTexto(string texto)
        {
            Regex regla = new Regex("^[a-zA-Z ]*$");

            if (regla.IsMatch(texto))
                return true;
            else
            {
                Console.WriteLine(" ... No ingrese NUMEROS !!!! ...");
                return false;
            }
        }



    }
      }
```

- EMPLEADOS.CS

```
    using System;
using System.Collections.Generic;

namespace PlanM.Modelo
{
    public partial class Empleados
    {
        public uint Cedula { get; set; }
        public string Nombre { get; set; }
        public int? Salario { get; set; }
        public int? DiasTrabajados { get; set; }
        public int? VacacionesPagar { get; set; }
    }
}
```

- PLANMEJOCONTSXT.CS

```
    using System;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata;

namespace PlanM.Modelo
{
    public partial class planmejoContext : DbContext
    {
        public virtual DbSet<Empleados> Empleados { get; set; }

        protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
        {
            if (!optionsBuilder.IsConfigured)
            {
#warning To protect potentially sensitive information in your connection string, you
should move it out of source code. See http://go.microsoft.com/fwlink/?LinkId=723263
for guidance on storing connection strings.
                optionsBuilder.UseMySql("Server=localhost; userid=root; password=;
Database=planmejo");
            }
        }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<Empleados>(entity =>
            {
                entity.HasKey(e => e.Cedula);

                entity.ToTable("empleados");

                entity.Property(e => e.Cedula).HasColumnName("cedula");

                entity.Property(e => e.DiasTrabajados)
                    .HasColumnName("diasTrabajados")
                    .HasColumnType("int(11)");

                entity.Property(e => e.Nombre)
```

```
                    .HasColumnName("nombre")
                    .HasMaxLength(45);

                entity.Property(e => e.Salario)
                    .HasColumnName("salario")
                    .HasColumnType("int(11)");

                entity.Property(e => e.VacacionesPagar)
                    .HasColumnName("vacacionesPagar")
                    .HasColumnType("int(11)");
            });
        }
    }
}
```

- DIFICULTADES

En el desarrollo del programa encontramos diversas dificultades:

- ✓ Conexión base de datos
- ✓ Operación
- ✓ Posicionamiento menu

- DESARROLLO DEL PROGRAMA

Para el desarrollo del programa fue importante tener en cuenta conceptos como:

- Librerías: Se uso para la verificación diferentes estructuras de lenguaje

- Static void: Se uso para la creación de los bloques

- Do: Se uso para que el desarrollo del ciclo

- If – Else: Se uso para la validación los cálculos de cada proceso

- While: Se uso para cerrar el ciclo

- Console.WriteLine(); Se uso para mostrar el mensaje en pantalla

- Console.Clear(); Se uso para limpiar el mensaje que no queríamos mostrar en pantalla

- Console.ReadLine(); Se uso para avanzar en el programa en un salto de línea ingresando los caracteres a solicitar

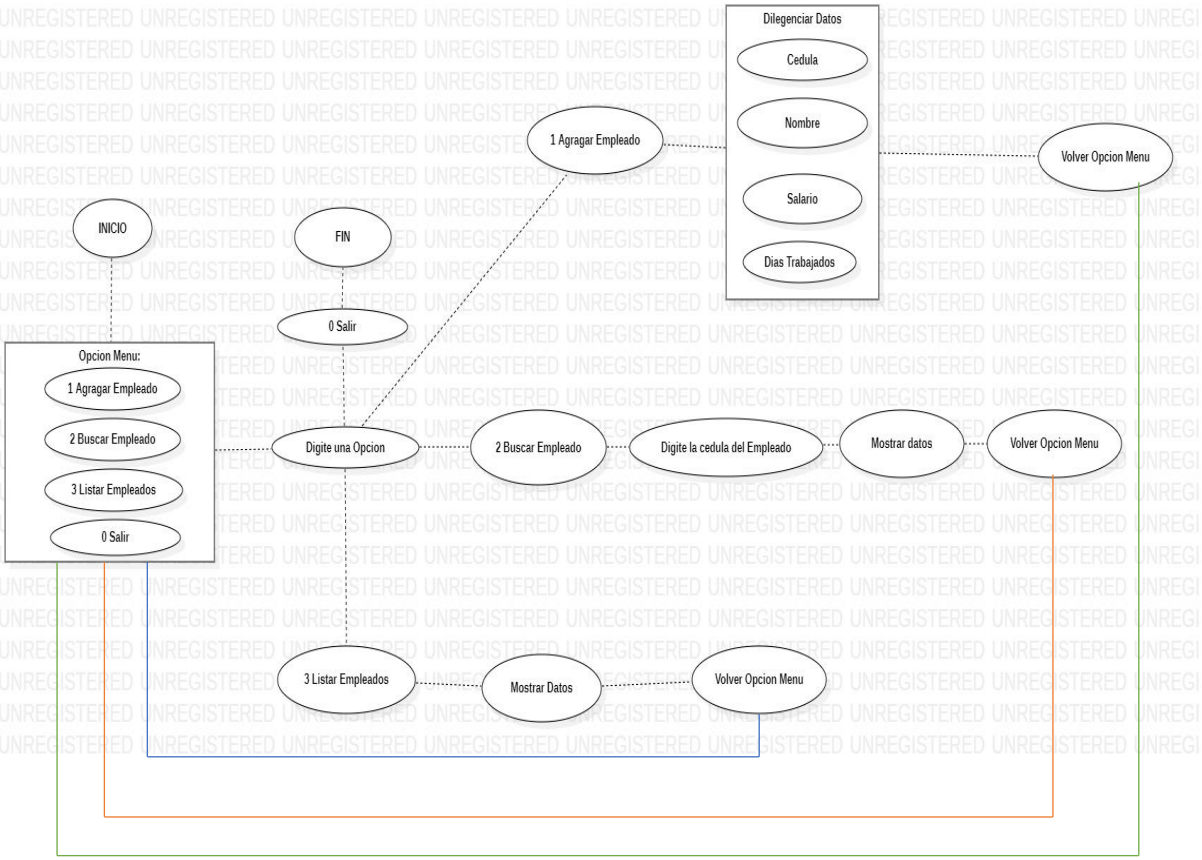
- Console.ReadKey(); Se uso para la continuidad de los ciclos

- Case: Se uso para dar el funcionamiento al menú solicitado según las opciones brindadas en el primer “Do” del programa

- Console.SetCursorPosition: Se uso para darle posición a las variables en la Pantalla

Para la solución del ejercicio iniciamos con la creación de un proyecto llamado PlanM, en el que realizamos varios procesos utilizando los conceptos anteriormente mencionados.

DIAGRAMA DE FLUJO.



Nota: diagrama echo en StarUML línea de colores en Word

Link repositorio GitHub: https://github.com/Dagnober/PlanM

link video explicación aplicación en Meet: https://drive.google.com/file/d/1MHayZ1mbbQexmtVzwts2IFRp90e73hVG/view