Temas utilizados

Herencia

Archivo: Operario

```
namespace Trabajador
{
    10 referencias
    public class Operario : Empleado
    {
        2 referencias
        public Operario(string nombre, string apellido, int id) : base(nombre,apellido,id)
```

Archivo: Supervisor

```
namespace Trabajador
{
    public class Supervisor : Empleado
    {
        public Supervisor(string nombre, string apellido, int id) : base(nombre, apellido, id)
```

Propiedades

Archivo: Empleado

```
2 referencias
public string Nombre { get => nombre; set => nombre = value; }
2 referencias
public string Apellido { get => apellido; set => apellido = value; }
2 referencias
public int Id { get => id; set => id = value; }
```

Colecciones

Formulario modal

Archivo: FormCrear y FormProcesando

```
formProcesando.Show();
```

```
timer.Stop();
FormAprobacion formAprobacion = new FormAprobacion();


this.Close();
formAprobacion.Show();
```

Clase estáticas

Archivo: Listas

```
public static class Listas
{
    private static List<Operario> iniciosOperario = new List<Operario>();
    private static List<Supervisor> iniciosSupervisor = new List<Supervisor>();

    4 referencias
    public static List<Operario> IniciosOperario { get => iniciosOperario; set => iniciosOperario = value; }
    3 referencias
    public static List<Supervisor> IniciosSupervisor { get => iniciosSupervisor; set => iniciosSupervisor = value; }
}
```

Polimorfismo

Archivo: Empleado, Supervisor , Operario

```
5 referencias
public abstract bool Ingreso<T>(string nombre, string apellido, int id, List<T>list);
```

```
3 referencias | 1/1 pasando
public override bool Ingreso<T>(string nombre, string apellido, int id, List<T>lista)
{

    if (typeof(T) == typeof(Supervisor))
    {
        // Convierte la lista a List<Operario> si el tipo genérico es Operario
        List<Supervisor> listOperarios = lista as List<Supervisor>;

        foreach (Supervisor operario in listOperarios)
        {
            if (operario.Nombre == nombre && operario.Apellido == apellido && operario.Id == id)
            {
                return true;
            }
        }
    }
    return false;
}
```

Excepciones

Archivo: inico, CrudDao, Archivo

```
private void dia_Click(object sender, EventArgs e)
{
    string path = @"C:\Users\nicol\Desktop\Avila.Daniel.Parcial\Info\Configuracion.json";
    Config colores = new Config();
    try
    {
        if (File.Exists(path))
        {
            Archivos<Config> colorJs = new Archivos<Config>();
            Config coloresJson = colorJs.Leer_JSON<Config>();
            fondo = ColorTranslator.FromHtml(coloresJson.ColorClaro);
            cambiarColor(this);
        }
        else
        {
            Archivos<Config> colorJs = new Archivos<Config>();
            colorJs.EscribirJson<Config>(colores);
        }
    }
    catch (Exception ex)
    {
        Archivos<string>.error(DateTime.Now, MethodBase.GetCurrentMethod().DeclaringType.Name, MethodBase.GetCurrentMethod().Name, e

    }
```

Archivos y Serializacion

Archivo: Archivo

```
3 referencias
public void EscribirJson<T>(T objeto)
{
    string archivo = $"{path}/Configuracion.json";

    if (!Directory.Exists(path))
    {
        Directory.CreateDirectory(path);
    }
    JsonSerializerOptions options = new JsonSerializerOptions();
    options.WriteIndented = true;

    string jsonString = JsonSerializer.Serialize(objeto, options);

    File.WriteAllText(archivo, jsonString);
}

3 referencias
public  T Leer_JSON<T>()
{
    string archivo = $"{path}/Configuracion.json";
    string jsonString = File.ReadAllText(archivo);
    T objeto = JsonSerializer.Deserialize<T>(jsonString);

    return objeto;
}
```

Generics

Archivos: Empleado, Supervisor, Operaio, Archivos

```
6 referencias | 1/2 pasando
public abstract bool Ingreso<T>(string nombre, string apellido, int id, List<T>list);
```

Interfaces

Archivos: IArchivos

```
namespace Fabrica
{
    1 referencia
    internal interface Iarchios<T>
    {
        2 referencias
        bool CrearTxt(string dato);
        1 referencia
        void crearXml(T objeto);
        1 referencia
        void EscribirJson<T>(T objeto);
        1 referencia
        T Leer_JSON<T>();


    }
}
```

Test unitario

Archivo: UnitTest1

```
{
    [TestMethod]
    | 0 referencias
    public void Actualizar_DeberiaActualizarOperario_SipervisorEnBD()
    {
        // Arrange
        int valorAleatorio = new Random().Next(0, 2);
        string nombre = (valorAleatorio == 0) ? "testOperador" : "testSupevisor";
        string apellido = (valorAleatorio == 0) ? "testOperador" : "testSupevisor";
        int id = 0; // Reemplaza con un ID existente en tu base de datos.
        string trabajos = (valorAleatorio == 0) ? "OPERADOR" : "SUPERVISOR";
        string trabajo = trabajos;
        // Act
        CrudDAO.Actualizar(nombre, apellido, id, trabajo);

    }

    [TestMethod]
    | 0 referencias
    public void IngresoOperario_PuedeIniciaSecion_DeberiaRetornarTrue()
    {
        Operario operario = new Operario("","",0);
        //Arrange
        bool expected = true;
        //Act
        bool actual = operario.Ingreso("testOperador", "testOperador", 0, CrudDAO.LeerOperarios()) ;
        //Assert
        Assert.AreEqual(expected, actual);
    }
```

Base de datos

Archivo: CrudDao

```
namespace Trabajador
{
    16 referencias
    public static class CrudDAO
    {
        static string connectionString;
        static SqlCommand command;
        static SqlConnection connection;
        const string NOMBRE_COLUMNA = "NOMBRE";
        const string APELLIDO_COLUMNA = "APELLIDO";
        const string ID_COLUMNA = "ID";


        0 referencias
        static CrudDAO()...

        2 referencias
        public static void Guardar(string nombre, string apellido,string trabajo)...

        3 referencias | 1/1 pasando
        public static void Actualizar(string nombre,string apellido, int id,string trabajo)...

        4 referencias | 1/1 pasando
        public static List<Operario> LeerOperarios()
        {
            List<Operario> operarios = new List<Operario>();
```

Delegados

Archivo: inicio, Material

```
    4 referencias
    private void cambiarColor(Form instancia)
    {
        instancia.BackColor = fondo;

    }
```

```
InitializeComponent();
var materialesOrdenados = Inventario.Stock.Select(item => new { Componente = item.Key, Cantidad = item.Value }).ToList();
materialesOrdenados.Sort((material1, material2) => material1.Cantidad - material2.Cantidad);
dataGridView1.DataSource = materialesOrdenados;
this.cambiarColor = cambiarColor;
```

Eventos

Archivo: inico

```
private void Inicio_Load(object sender, EventArgs e)
{
    btnSupervisor.Click += btnSupervisor_Click;
    btnOperario.Click += btnOperario_Click;
}
```