

**ESCUELA ESPECIALIZADA EN INGENIERIA ITCA FEPADE MEGATEC
REGIONAL LA UNIÓN**

ESCUELA DE INGENIERIA EN COMPUTACIÓN

TÉCNICO EN INGENIERÍA DE DESARROLLO DE SOFTWARE

MÓDULO DE DESARROLLO DE APLICACIONES DE ESCRITORIO

MANUAL DEL DESARROLLADOR:

“SISTEMA DE GESTIÓN DE PERSONAL”

PRESENTADO POR:

Vigil Gonzales, Henry Josué

Cienfuegos Martínez, Kelvin Eduardo

Águila Villatoro, Oscar Mauricio

Martínez Vásquez, Esaú Adalberto

LA UNIÓN, 12 DE NOVIEMBRE DE 2025

VISIÓN GENERAL DEL SISTEMA:	2
SOFTWARE UTILIZADO:	2
GUIA DE INICIO RÁPIDO:	2
ARQUITECTURA Y DISEÑO:	3
DISEÑO DE LA BASE DE DATOS:	5
DIAGRAMA DE CLASES.....	9
DIAGRAMA LÓGICO:	10
DICCIONARIO DE DATOS:	11
MODULO DE CALCULO DE PAGO:	12

VISIÓN GENERAL DEL SISTEMA:

La constructora ConstruGlobal enfrenta diferentes problemas para controlar la asistencia del personal de las obras, calcular correctamente los pagos y mantener un registro confiable de los trabajadores

activos e inactivos, actualmente estos controles se manejan de forma manual en hojas de cálculos, lo que suele generar errores frecuentes, duplicidad de registros y retraso en la gestión del personal.

SOFTWARE UTILIZADO:

Lenguaje: C# (.NET)

IDE: Visual Studio

Base de Datos: SQL Server

GUIA DE INICIO RÁPIDO:

Prerrequisitos de Software:

Instalar "Visual Studio 2022", "SQL Server Management Studio".

Instalar los siguientes nuggets: Microsoft database clean y system configuration.Configuration management

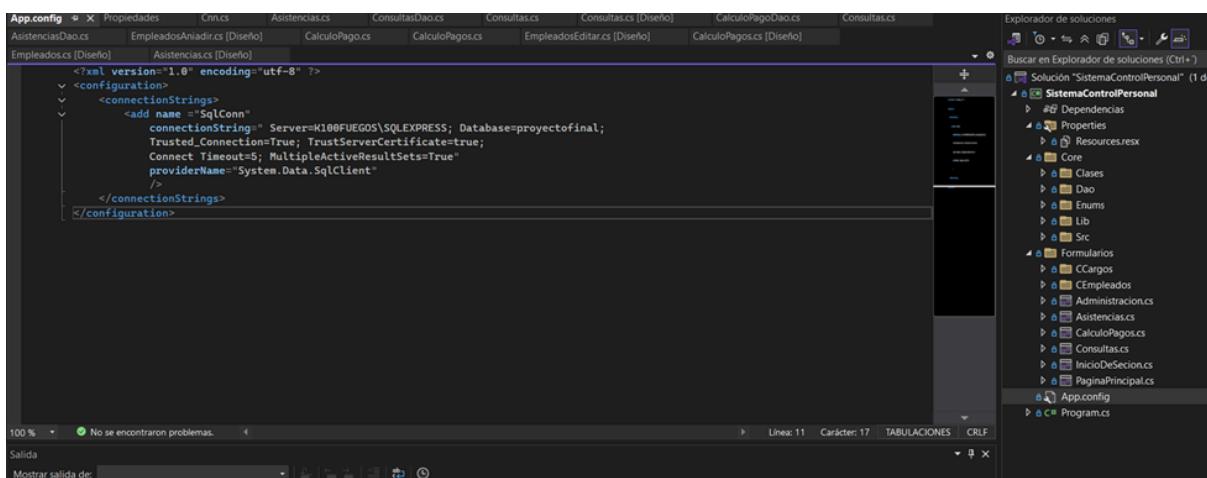
Obtención del Código Fuente:

Clona el repositorio:

```
git clone [https://github.com/k100fuegos/ProyectoFinal_DAE.git]
```

Rama principal: **main**.

Base de Datos: script de la base de datos (BDproyectoFinal.sql).



Configurar el archivo de App.Config: cambiar el nombre del servidor local K100FUEGOS\SQLEXPRESS al nombre del server local de su computadora

ARQUITECTURA Y DISEÑO:

¿Por qué se usó visual estudio para la creación de este sistema?

1. El Ecosistema .NET

Visual Studio es el hogar de C# y el framework .NET. C# es un lenguaje moderno, robusto y seguro, ideal para construir aplicaciones empresariales. Te permite crear una lógica de negocio clara y fácil de mantener para manejar empleados, planillas, asistencias, etc.

2. Herramientas de Interfaz de Usuario (UI)

Para un sistema de escritorio, Visual Studio te da acceso directo a Windows Forms (WinForms) o WPF. Puedes literalmente arrastrar y soltar botones, tablas y campos de texto para diseñar las ventanas del programa, lo que acelera el desarrollo visual enormemente.

3. El Depurador

El depurador de Visual Studio es el mejor del mercado. Te permite ejecutar tu código línea por línea, ver el valor de las variables en tiempo real y encontrar errores de una forma que es casi imposible en editores de texto más simples. Para un sistema que calcula pagos, esto es crucial.

Estructura del repositorio:

Esta conformado por 2 carpetas principales donde dividimos la documentación y codificación y un script para la creación de la base de datos

Documentación

Donde se encuentran los manuales de usuario y de desarrollador. También se encuentran los diagramas de clases, lógico y el diccionario de datos.

C#

SistemaControlPersonal Donde se encuentra alojado el sistema, dentro de esa carpeta encontrara la solución

.core

Donde se encuentran las clases, dao, enums, lib y src

.formularios

Donde se encuentran los formularios del sistema

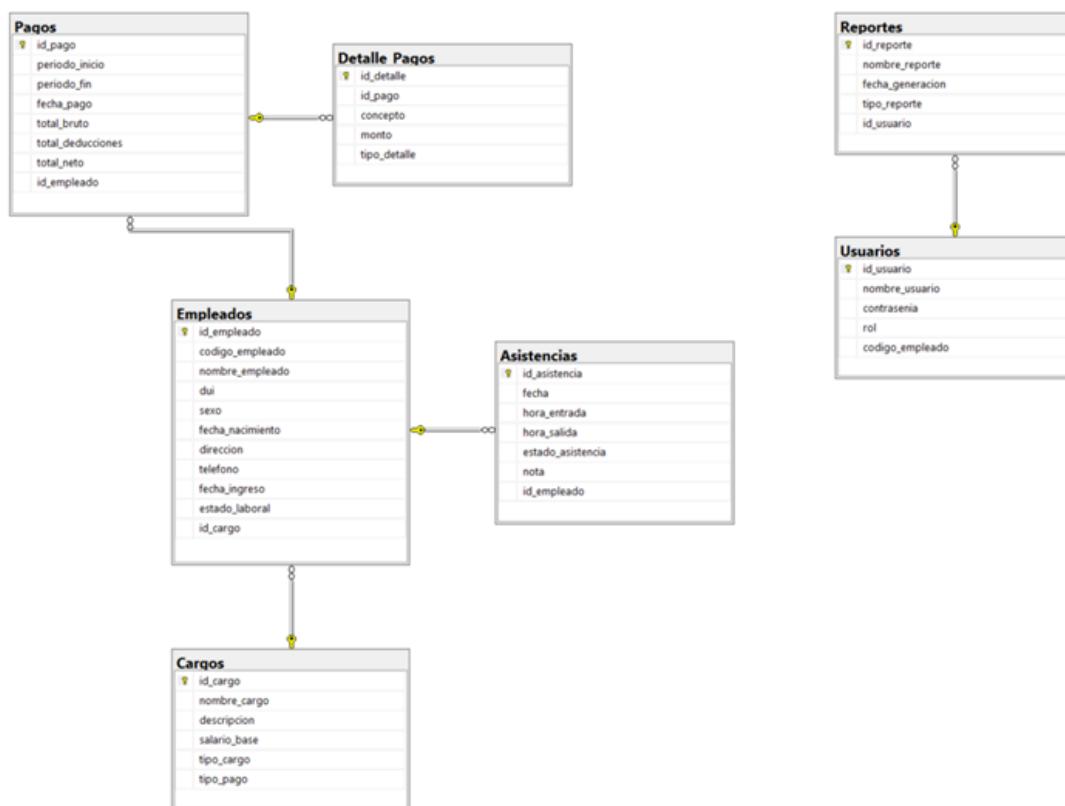
.properties

Donde se encuentra las propiedades y elementos del sistema

BDprojetoFinal.sql

El script de la base de datos usada para el sistema

DISEÑO DE LA BASE DE DATOS:



1. Tabla Empleados

Propósito:

Almacena los datos personales y laborales de cada empleado.

Campos clave:

id_empleado Clave primaria (PK).

codigo_empleado Identificador interno o visible al usuario.

nombre_empleado, dui, sexo, fecha_nacimiento, direccion, telefono Datos personales.

fecha_ingreso, estado_laboral Datos de control laboral.

id_cargo Clave foránea (FK) que conecta con la tabla Cargos.

Relaciones:

Un empleado pertenece a un cargo.

Un empleado puede tener muchas asistencias y muchos pagos.

2. Tabla Cargos

Propósito:

Define los distintos cargos o puestos dentro de la organización.

Campos clave:

id_cargo PK.

nombre_cargo, descripcion, salario_base, tipo_cargo, tipo_pago.

Relaciones:

Un cargo puede ser ocupado por muchos empleados.

3. Tabla Asistencias

Propósito:

Registra la asistencia diaria de los empleados.

Campos clave:

id_asistencia PK.

fecha, hora_entrada, hora_salida, estado_asistencia, nota.

id_empleado FK hacia Empleados.

Relaciones:

Un empleado puede tener muchos registros de asistencia.

Cada asistencia pertenece a un solo empleado.

4. Tabla Pagos

Propósito:

Guarda los pagos generados para cada empleado (sueldos, periodos, deducciones, etc.).

Campos clave:

id_pago PK.

periodo_inicio, periodo_fin, fecha_pago Definen el rango del pago.

total_bruto, total_deducciones, total_neto.

id_empleado FK hacia Empleados.

Relaciones:

Un pago pertenece a un empleado.

Un pago puede tener muchos detalles de pago (bonos, descuentos, etc.).

5. Tabla Detalle_Pagos

Propósito:

Desglosa los componentes de un pago (conceptos adicionales, descuentos, etc.).

Campos clave:

id_detalle PK.

id_pago FK hacia Pagos.

concepto, monto, tipo_detalle (ej. “bono”, “descuento”).

Relaciones:

Cada detalle pertenece a un pago.

Un pago puede tener múltiples detalles.

6. Tabla Usuarios

Propósito:

Gestiona el acceso al sistema (autenticación y roles).

Campos clave:

id_usuario PK.

nombre_usuario, contrasenia, rol.

codigo_empleado vincula el usuario con un empleado (no es una FK formal, pero funciona como enlace lógico).

Relaciones:

Un usuario puede generar varios reportes.

7. Tabla Reportes

Propósito:

Guarda información sobre los reportes generados por el sistema.

Campos clave:

id_reporte PK.

nombre_reporte, fecha_generacion, tipo_reporte.

id_usuario FK hacia Usuarios.

Relaciones:

Cada reporte es creado por un usuario.

Un usuario puede generar muchos reportes.

DIAGRAMA DE CLASES

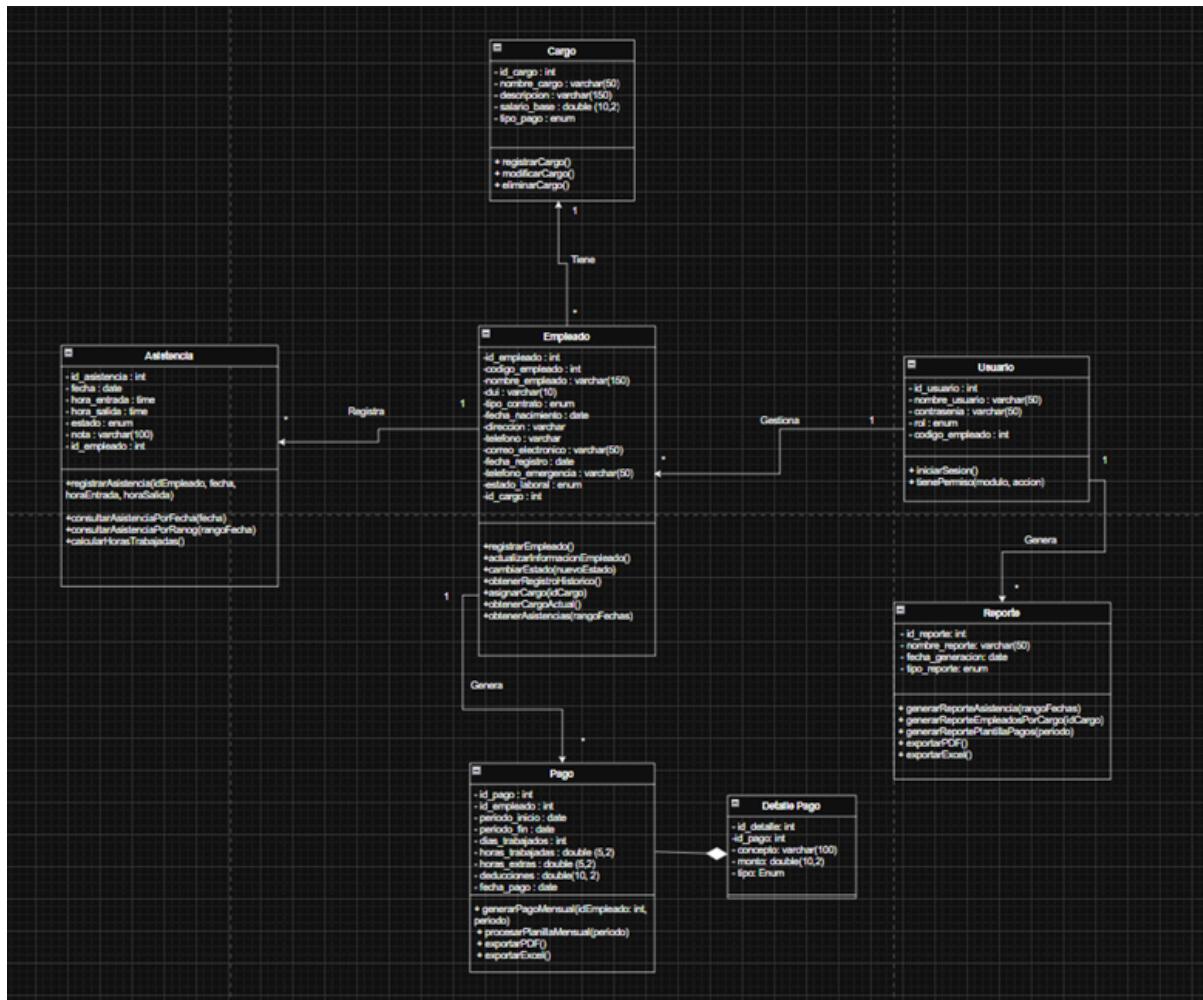
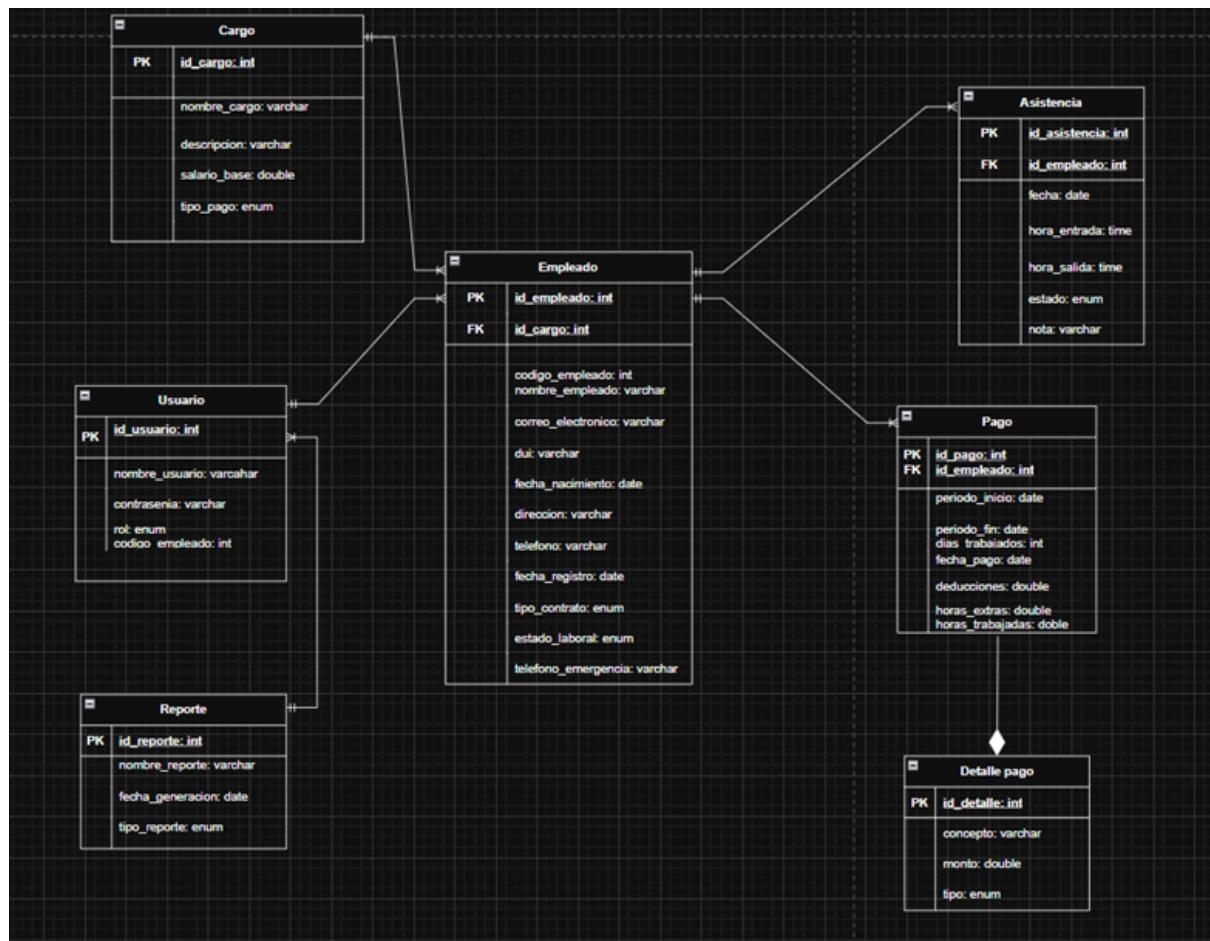


DIAGRAMA LÓGICO:



DICCIONARIO DE DATOS:

Entidad /	Campo	Tipo de dato	Tamaño /	Descripción
ASISTENCIA	id_asistencia	Entero		11 Identificador único del registro de asistencia (clave primaria).
	id_empleado	Entero		11 Identificador del empleado (clave foránea).
	fecha	Fecha	dd-mm-aaaa	Fecha del registro de asistencia.
	hora_entrada	Fecha	hh:mm	Hora en la que se registró la entrada
	hora_salida	Fecha	hh:mm	Hora en la que se registró la salida
	estado	Texto	25	Estado de asistencia: "Presente", "Ausente", "Falta Justificada" o "Tarde".
CARGO	nota	Texto	100	Observaciones o comentarios del día (opcional).
	id_cargo	Entero		11 Identificador único del cargo (clave primaria).
	nombre_cargo	Texto	50	Nombre del cargo (Ej. Ingeniero Residente, Albañil, etc.).
	descripcion	Texto	150	Breve descripción de las funciones del cargo.
	tipo_pago	Texto	15	Indica si el cargo se paga "Fijo" o "Por hora".
EMPLEADO	salario_base	Decimal	10,2	Monto base mensual o tarifa por hora del cargo.
	id_empleado	Entero		11 Identificador único del empleado (clave primaria).
	codigo_empleado	Entero		11 Identificador único del empleado.
	nombre_completo	Texto	150	Nombre completo del empleado.
	dui	Texto	10	Documento Único de Identidad (formato: 00000000-0).
	id_cargo	Entero		11 Identificador del cargo asignado (clave foránea).
	estado_laboral	Texto	10	Estado del empleado ("Activo" o "Inactivo").
	correo_electronico	Texto	50	Correo de contacto del empleado.
	telefono	Texto	50	Teléfono para contacto del empleado
PAGO_MEN	telefono_emergencia	Texto	50	Teléfono de emergencia del empleado
	fecha_registro	Fecha	dd-mm-aaaa	Fecha en que se registró al empleado.
	id_pago	Entero		11 Identificador del registro de pago (clave primaria).
	id_empleado	Entero		11 Empleado al que corresponde el pago (clave foránea).
	periodo_inicio	Texto	15	Mes y año de inicio del cálculo (Ej. "Octubre 2025").
	periodo_fin	Texto	15	Mes y año de fin del cálculo (Ej. "Octubre 2025").
	dias_trabajados	Entero	2	Total de días trabajados en el período.
USUARIO	horas_trabajadas	Decimal	5,2	Total de horas trabajadas (si aplica).
	horas_extras	Decimal	5,2	Total de horas extras trabajadas.
	deducciones	Decimal	10,2	Monto total descuentado por ausencias o retardos.
	pago_neto	Decimal	10,2	Monto final a pagar al empleado.
	horas_extras	Decimal	5,2	Número de horas extras trabajadas en el período.
	fecha_pago	Fecha	dd-mm-aaaa	Fecha en que se genera el pago.
	id_usuario	Entero		11 Identificador del usuario del sistema.
	nombre_usuario	Texto	50	Nombre del usuario.
	contrasenia	Texto	50	Contraseña cifrada para acceso.
	codigo_empleado	Entero		11 Llave foránea del Id del empleado

MODULO DE CALCULO DE PAGO:

CalculoPagos es un Form de WinForms cuyo objetivo principal es:

- Mostrar en un DataGridView (dgvCalculoPagos) una lista de empleados (modelo CalculoPagoEmpleado) traída vía CalculoPagosDao.
- Permitir búsqueda/filtrado en cliente sobre campos (código, nombre, cargo, tipo de pago).
- Permitir seleccionar una fila y, al pulsar btnCalcularPago, calcular el pago del empleado seleccionado y mostrar el detalle en lbxReporte.
- Soporta dos tipos de pago (detectados por cadenas): pago por hora y pago mensual.

Dependencias externas importantes

- CalculoPagosDao calculoDao método usado: GetAll(string) que devuelve List<CalculoPagoEmpleado>.
- CalculoPagoEmpleado debe exponer al menos: CódigoEmpleado, NombreEmpleado, Cargo, TipoPago, SalarioBase.

Inicialización y conexión de eventos

- En el constructor:
 - InitializeComponent() (Designer).
 - ConfiguracionGrid() configura columnas del dgvCalculoPagos.
 - Conector robusto para el TextBox de búsqueda: busca controles con nombre "txtBusqueda" o "textBuscador" y conecta su TextChanged a txtBusqueda_TextChanged. (Buena compatibilidad con distintos nombres.)
 - Conecta btnCalcularPago.Click a btnCalcularPago_Click si el control existe.
 - Llama a Cargar() para poblar el grid inicialmente.

ConfiguracionGrid()

- Desactiva AutoGenerateColumns, limpia columnas y añade manualmente 4 columnas: Código, Nombre, Cargo, Tipo Pago (vinculadas a DataPropertyName).
- Ajusta cada columna para que AutoSizeMode = Fill y reparte el ancho en partes iguales (usa FillWeight calculado por 100f / columnas).
- Se asegura de (re)conectar el evento CellFormatting actualmente el handler está vacío (comentado como punto de extensión).

btnCalcularPago_Click lógica de cálculo

1. Obtiene la fila actual: var fila = dgvCalculoPagos.CurrentRow; valida existencia.
2. var empleado = fila.DataBoundItem as CalculoPagoEmpleado; valida existencia.
3. Obtiene CultureInfo.CurrentCulture para parseo y formateo.
4. Normaliza tipo con Trim().ToLowerInvariant() (paso clave para detección).

5. Define constantes dentro del método:
 - LIMITE_MENSUAL = 176m; (se interpreta como horas mensuales estándar)
 - DIAS_MENSUALES = 30m;
 - VALOR_HORA_EXTRA_FIJO = 2m; (cada hora extra vale 2 dólares fijos)
6. Declara variables: horasTrabajadas, diasTrabajados, deducciones, salarioBase = empleado.SalarioBase, horasExtra, horasNormales, pagoNormal, pagoHorasExtra, pagoTotal.
7. Rama si tipo.Contains("hora"):
 - Requiere que txtHorasTrabajadas.Text y txtDiasTrajados.Text no estén vacíos. (Muestra Warning y retorna si faltan.)
 - Parsea txtHorasTrabajadas y txtDiasTrajados usando decimal.TryParse(..., NumberStyles.Number, culture, out ...).
 - Parsea txtDeducciones opcionalmente con NumberStyles.Currency | NumberStyles.Number.
 - Valida salarioBase > 0.
 - Calcula horasExtra = Max(0, horasTrabajadas - LIMITE_MENSUAL).
 - horasNormales = Max(0, horasTrabajadas - horasExtra) (equivale a min(horasTrabajadas, LIMITE_MENSUAL)).
 - pagoNormal = horasNormales * salarioBase;
 - pagoHorasExtra = horasExtra * VALOR_HORA_EXTRA_FIJO;
 - pagoTotal = pagoNormal + pagoHorasExtra - deducciones;
8. Rama si tipo.Contains("mes") || tipo.Contains("mensual") || tipo.Contains("monthly"):
 - Asigna horasTrabajadas = LIMITE_MENSUAL y diasTrabajados = DIAS_MENSUALES.
 - Parsea txtDeducciones si fue rellenado.
 - Valida salarioBase > 0.
 - SalarioBase ya es salario mensual → pagoNormal = salarioBase.
 - pagoHorasExtra = 0.
 - pagoTotal = pagoNormal - deducciones.
9. Si tipo no contiene "hora" ni "mes" devuelve mensaje de tipo no reconocido.
10. Redondea pagoNormal, pagoHorasExtra, pagoTotal a 2 decimales.
11. Llena lbxReporte con el detalle (nombre, código, cargo, tipo, salario base (formateado en cultura), horas, días, horas extra, pago normal, pago horas extra, deducciones, pago total).