Sistema De Gestión De Nómina En Java (POO)

Dairo Enrique Montiel Martínez

José Armando Navarro

José David Pérez Molina

Análisis Y Desarrollo De Software

Ficha: 3066035

SENA

CTPGA

Félix Sánchez Ramos

05/09/2025

1. INTRODUCCIÓN

En este informe se expone el Sistema de Gestión de Nómina desarrollado en Java, una aplicación de escritorio cuyo propósito es automatizar y simplificar el cálculo y la administración de pagos salariales.

El sistema incorpora una interfaz gráfica de usuario (GUI) desarrollada en Swing, que permite registrar datos de empleados y generar recibos de pago de manera rápida y precisa. El alcance contempla tanto a empleados de tiempo completo como parcial, aplicando las deducciones y aportes establecidos por la legislación laboral colombiana actual.

A nivel técnico, el proyecto integra conceptos de Programación Orientada a Objetos (POO) mediante la aplicación de abstracción, herencia y polimorfismo para modelar empleados, deducciones y aportes. Asimismo, implementa el patrón de diseño Modelo–Vista–Controlador (MVC), lo que asegura organización, escalabilidad y mantenibilidad del sistema.

Este proyecto no solo busca entregar una herramienta funcional, sino también fortalecer competencias en diseño de sistemas, investigación, desarrollo de interfaces, modularización del código y buenas prácticas de programación.

2. OBJETIVOS

El objetivo de este proyecto es desarrollar un sistema de gestión de nómina, que sea escalable y fácil de usar, automatizando el cálculo de salarios, deducciones y aportes legales, a la vez que ofrece una interfaz para la administración de empleados y la generación de reportes.

Para alcanzar este propósito, se plantearon como metas:

Implementar algoritmos que automaticen los cálculos salariales teniendo en cuenta el tipo de contrato, las horas trabajadas y las deducciones legales vigentes.

Diferenciar la gestión de empleados de tiempo completo y parcial mediante módulos específicos que faciliten su administración.

Generar recibos de pago detallados en formato TXT, con información clara de devengados, deducciones y aportes.

Diseñar una interfaz gráfica que optimice la experiencia del usuario.

Aplicar principios de Programación Orientada a Objetos (POO) y el patrón Modelo–Vista–Controlador (MVC) para asegurar modularidad, escalabilidad y mantenibilidad del sistema.

3. ARQUITECTURA Y ESTRUCTURA DEL PROYECTO

El proyecto está estructurado en cinco paquetes principales:

Paquete Principal (Raíz)

```
src/
└── App.java              # Punto de entrada de la aplicación
```

App.java: Clase principal que inicializa el sistema, configura el Look and Feel y lanza la interfaz gráfica en el Event Dispatch Thread (Manejar y procesar los eventos de la interfaz gráfica).

Modelo

Contiene las entidades y la lógica de negocio:

```
src/modelo/
├── Empleado.java          # Clase abstracta base
├── EmpleadosCompletos.java  # Empleados tiempo completo
└── EmpleadoParcial.java     # Empleados tiempo parcial
```

Descripción de Clases:

Empleado.java: Clase abstracta base con atributos generales (nombre, estado civil, horas trabajadas) y el método abstracto calcularSalarioBruto().

EmpleadoCompleto.java: Especialización para empleados de tiempo completo, con salario base ($1,423,500 COP) , horas extras (125%), deducciones y aportes patronales.

EmpleadoParcial.java: Especialización para empleados de tiempo parcial, con cálculo proporcional según horas trabajadas y deducciones recomendadas.

Vista

Encargada de la interfaz gráfica de usuario (GUI) y la presentación de datos:

```
src/vista/
├── MainFrame.java            # Ventana principal
├── MenuPrincipalPanel.java      # Dashboard lateral
├── PanelRegistroEmpleado.java   # Formulario de registro
├── PanelEmpleadosCompleto.java  # Vista empleados fijos
└── PanelEmpleadosParcial.java   # Vista empleados parciales
```

Componentes de la Interfaz:

MainFrame.java: Ventana principal con BorderLayout y CardLayout.

MenuPrincipalPanel.java: Panel de navegación lateral fijo.

PanelRegistroEmpleado.java: Formulario de registro de datos.

PanelEmpleadosCompleto.java: Tabla y gestión de empleados de tiempo completo.

PanelEmpleadosParcial.java: Tabla y gestión de empleados de tiempo parcial.

Servicios

```
src/servicios/
├── Nomina.java          # Gestión de lista de empleados
└── Exportar.java        # Generación de archivos TXT
```

Servicios del Sistema:

Nomina.java: Administración central de empleados, registro en memoria y cálculos.

Exportar.java: Generación de archivos TXT con recibos de pago y creación automática de directorio "nominas/" (en caso de no existir).

Recursos

```
src/recursos/
└── logo.png             # Logo de la aplicación
```

Principios de diseño aplicados

1.Programación Orientada a Objetos: encapsulación, herencia y polimorfismo.

2.Single Responsibility Principle: cada clase cumple una función específica.

3.Open/Closed Principle: el sistema es extensible sin necesidad de modificar la base.

Gestión de Interfaces

1.BorderLayout Principal: estructura la ventana con un panel lateral fijo y área de contenido central.

2.CardLayout: permite navegar fluidamente entre paneles sin recarga.

PROCESO DE FUNCIONAMIENTO

Captura de datos: nombre, contrato, estado civil y horas trabajadas.

Validación: verificación de campos obligatorios, tipos de datos y valores positivos.

Procesamiento: cálculo automático de salario bruto, deducciones y aportes.

Visualización: presentación en tablas por tipo de empleado.

Exportación: generación de recibos de pago en formato TXT.

4. CÁLCULOS Y REGLAS DE NEGOCIO

Empleados de Tiempo Completo

Si el empleado trabaja **192 horas o menos**:

$$Salario \ = \frac{1{,}423{,}500}{192} * \ horas\ trabajadas$$

Si el empleado trabaja **más de 192 horas**:

$$Salario = 1{,}423{,}500 + ((horas\ trabajadas - 192) \times valor\ hora\ extra)$$

Donde:

valor hora extra = (1,423,500 ÷ 192) × 1.25

Deducciones del empleado: Salud (4%), Pensión (4%), Fondo de Solidaridad 1%.

Aportes del empleador: Salud (8.5%), Pensión (12%), ARL (0.522%), Parafiscales (9%).

Empleados de Tiempo Parcial:

Salario Bruto

$$Salario = horas\_trabajadas \times 6{,}189$$

Deducciones recomendadas: Salud (4%), Pensión (4%), Fondo de Solidaridad (1% si aplica). Como trabajadores independientes, estas deducciones son voluntarias y deben ser gestionadas por el empleado.

5. GESTIÓN DE ERRORES Y ALMACENAMIENTO

Gestión de errores:
El sistema implementa validación de entradas para evitar datos vacíos o incorrectos, control de excepciones mediante bloques try-catch, y notificaciones al usuario utilizando la clase JOptionPane de Swing para mostrar mensajes descriptivos y preventivos.

Almacenamiento: datos en memoria RAM (no persiste entre sesiones).

6. VENTAJAS TÉCNICAS Y FUNCIONALES

Flexibilidad para distintos contratos y horas variables.

Escalabilidad y preparación para base de datos.

Interfaz intuitiva con navegación clara.

Código modular y bien documentado.

Cumplimiento de la legislación laboral colombiana.

7. LIMITACIONES

Limitaciones actuales

Almacenamiento temporal en memoria.

Exportación solo en formato TXT.

Falta de implementación para otros tipos de contrato (solo se contemplan empleados de tiempo completo y parcial).

8. CONCLUSIONES Y REFLEXIÓN DE APRENDIZAJE

El desarrollo del **Sistema de Gestión de Nómina en Java** permitió alcanzar los objetivos iniciales: automatizar cálculos salariales, aplicar deducciones legales, generar reportes y garantizar un diseño modular y escalable.

Más allá del logro técnico, este proyecto representó una experiencia de **aprendizaje integral** en varios aspectos:

Diseño de sistemas: estructuración en capas y modularidad.

Investigación aplicada: análisis de normativas laborales en Colombia.

Desarrollo de interfaces: diseño de GUIs intuitivas en Swing.

Aplicación de POO y MVC: uso práctico de abstracción, herencia y polimorfismo.

Gestión del proyecto: documentación, validación y pruebas para asegurar calidad.

Aunque la versión actual tiene limitaciones, la arquitectura implementada facilita la evolución hacia sistemas más robustos con persistencia y nuevas funcionalidades.

En conclusión, este proyecto no solo brindó una solución técnica funcional, sino también un **ejercicio formativo clave**, en el cual se aprendió a investigar, diseñar, implementar y perfeccionar un sistema realista, integrando teoría y práctica en el proceso de desarrollo de software.