# Sistema de cálculo y pago de salarios

## Introducción

El sistema de cálculo y pago de salarios consiste en un sistema de tiempo real de una empresa donde se planea llevar un registro detallado y preciso de los pagos que esta realiza a los trabajadores.

Actualmente el sistema solo reconoce dos tipos de trabajadores que son: por horas y por mes, siendo este último de dos tipos posibles, con comisión y sin comisión. En base a estos tres tipos básicos de trabajadores se calculara el pago del sueldo correspondiente.

Dado el caso, se diseño una arquitectura con los patrones necesarios para q sea sencilla, mantenible y entendible.

## Descripción del problema

A lo largo del diseño y desarrollo se presentaron una serie de problemas respecto a la arquitectura, ya que los cambios, nuevos registros y el mantenimiento serian difíciles de realizar.

Los principales problemas que se identificaron fueron:

* Código desordenado, sección lógica junto con la sección de control.
* Múltiples conexiones a la base de datos.
* Problemas al crear y reconocer los tipos de trabajadores existentes.
* Calcular el salario de diferentes formas, según cada tipo de trabajador.

## Diagrama de clases

El diagrama de clases inicial que se implemento es el siguiente:

De esta manera se mejoro y desarrollo un diagrama de clases que no presentara tantos problemas y sea sencillo de entender y mantener, finalizando en el siguiente resultado:

Siendo este diagrama se solucionaron los problemas encontrados. A continuación se demostrara como este diagrama soluciona dichos problemas.

## Problemas

### Problemas principales

* **Código desordenado, sección lógica junto con la sección de control**

Dado que el sistema es una aplicación con interfaces graficas y una serie de pasos a seguir para realizar operaciones, usamos el patrón arquitectónico MVC (Modelo – Vista – Controlador) con el cual se planea ordenar los fragmentos de código de tal forma que no existan clases que adquieran toda la responsabilidad.

* **Múltiples conexiones a la base de datos**

Al momento de instanciar la base de datos el código se repetía por todas las clases que solicitaban una conexión. Para evitar este problema se uso el patrón Singleton, a través del cual se declaro a la conexión y sus funciones como estáticas (static).

* **Problemas al crear y reconocer los tipos de trabajadores existentes**

Siendo que el trabajador podía ser de diferentes tipos, se paso a solucionar esta implementación con el patrón Factory Method

* **Calcular el salario de diferentes formas, según cada tipo de trabajador**

Siendo que el trabajador podía ser de diferentes tipos con una operación en común, *CalcularPago()*, pero con diferente funcionalidad, se paso a solucionar esta implementación con el patrón Strategy.

### Problemas secundarios

## Conclusiones

Los patrones tanto de diseño como arquitectónicos fueron de gran utilidad para la comprensión del diseño y funcionamiento del sistema, facilitando también el mantenimiento posteriormente.