# Sistema de Gestión de Empleados

Imagina que has sido contratado por una empresa de software para desarrollar un sistema de gestión de empleados. Este sistema permitirá registrar y gestionar la información de los empleados de la empresa.

**Objetivos**

- Practicar los conceptos de Programación Orientada a Objetos (POO) en C#, como clases, objetos, encapsulación, y métodos.
- Reforzar el uso de un sistema de control de versiones
- Generar el diagrama de clases del programa

**Requisitos del Sistema**

1. **Clase Empleado**:
    - **Propiedades**:
        - Id (Guid)
        - Nombre (string)
        - Apellido (string)
        - NumeroDeIdentificacion (string)
        - Edad (byte)
        - Posicion (string)
        - Salario (double)
    - **Métodos**:
        - CalcularBonificacion: Este método calculará una bonificación del 10% sobre el salario del empleado.(método privado)
        - MostrarInformacion: Este método imprimirá en la consola la información del empleado (nombre completo, edad, posición y salario pero con la bonificacion) (método publico).

2. **Clase Empresa**:
    - **Propiedades**:
        - Nombre (string)
        - Direccion (string)
        - ListaEmpleados (List<Empleado>)
    - **Métodos**:
        - AgregarEmpleado: Este método permitirá agregar un empleado a la lista de empleados.
        - EliminarEmpleado: Este método permitirá eliminar un empleado de la lista utilizando su nombre y apellido.
        - MostrarTodosLosEmpleados: Este método imprimirá en la consola la información de todos los empleados en la lista.
        - ActualizarEmpleado: Este método permitirá actualizar un empleado de la lista utilizando su número de identificacion.
        - BuscarEmpleado: Este método permitirá buscar un empleado por numero de documento
        - MostrarEmpleadosPorCargo: Este método me permitirá buscar a los empleabos pero deacuerdo a su cargo, ejemplo me debe mostrar a todos los “supervisores”

# Parte 2

Continuando con el desarrollo del sistema de gestión, ahora se necesita agregar la funcionalidad para gestionar también la información de los clientes de la empresa. Dado que los clientes y empleados comparten algunos atributos y el método MostrarInformacion, esta segunda parte te permitirá practica los conceptos de herencia y polimorfismo en C#.

**Objetivo**

El objetivo de esta parte del ejercicio es implementar la clase Cliente, utilizando herencia y polimorfismo para compartir atributos y métodos comunes con la clase Empleado.

**Requisitos del Sistema**

1. **Clase Base Persona**:
    - **Propiedades**:
        - Nombre (string)
        - Apellido (string)
        - Edad (int)
    - **Métodos**:
        - MostrarInformacion: Este método imprimirá en la consola la información de la persona (nombre completo y edad).
2. **Clase Empleado** (hereda de Persona):
    - **Propiedades**:
        - Id (Guid)
        - NumeroDeIdentificacion (string)
        - Posicion (string)
        - Salario (decimal)
    - **Métodos**:
        - CalcularBonificacion: Este método calculará una bonificación del 10% sobre el salario del empleado.
        - Sobrescribe el método MostrarInformacion para incluir la posición y el salario.
3. **Clase Cliente** (hereda de Persona):
    - **Propiedades**:
        - Email (string)

- Telefono (string)

- **Métodos**:
  - Sobrescribe el método MostrarInformacion para incluir el email y el teléfono.

4. **Clase Empresa** (actualizada):
   - **Propiedades**:
     - ListaEmpleados (List<Empleado>)
     - ListaClientes (List<Cliente>)
   - **Métodos**:
     - AgregarCliente: Este método permitirá agregar un cliente a la lista de clientes.
     - EliminarCliente: Este método permitirá eliminar un cliente de la lista utilizando su nombre y apellido.
     - MostrarTodosLosClientes: Este método imprimirá en la consola la información de todos los clientes en la lista.

# Parte 3

En esta parte del ejercicio, se requiere realizar modificaciones adicionales para mejorar la estructura y seguridad del código. Se debe alterar la clase Persona para que sea abstracta y cambiar el encapsulamiento de sus propiedades a protected. Además, se debe crear una clase estática que administre la información y facilite la creación de empleados y clientes, y también proporcionar una interfaz visual en consola.

**Objetivo**

El objetivo de esta parte del ejercicio es implementar la clase Persona como abstracta, cambiar el encapsulamiento de sus propiedades, utilizar una clase estática con métodos estáticos para la administración de empleados y clientes, y agregar métodos para mostrar interfaces visuales en la consola.

**Requisitos del Sistema**

**Clase Abstracta Persona**:

1. **Propiedades (cambiadas a protected y agregamos el Id)**:
    - Id (Guid)
    - Nombre (string)
    - Apellido (string)
    - Edad (int)
2. **Métodos**:
    - Método abstracto MostrarInformacion.

**Clase Empresa**:

**Métodos**:

- EliminarEmpleado: Este método permitirá eliminar un empleado de la lista utilizando su número de identificacion.

**Clase Estática Administracion**:

1. Métodos Estáticos:
    - CrearEmpleado: Este método pedirá los datos de un empleado y creará una instancia de Empleado, después retornara dicha instancia.
    - CrearCliente: Este método pedirá los datos de un cliente y creará una instancia de Cliente, después retornara dicha instancia.
2. Métodos para mostrar interfaces visuales en la consola:
    - MostrarTitulo(string titulo)
    - MostrarPie(string pie)
    - MostrarSeparador()