República Bolivariana de Venezuela

Ministerio del Poder Popular para la Educación

Universitaria Universidad Politécnica Territorial de Caracas

“Mariscal de Sucre”

PNF Informática

Sección 701201 – Trayecto I – Trimestre III

Algoritmo y Programación

Informe - Proyecto 3 Sistema Avanzado de Gestión de Personal y Proyectos

Profesor:

José Gregorio Ortiz

Alumna:

Sougleimy Aparicio V- 25213728

Caracas, Noviembre del 2025

Índice

Caracas, Noviembre del 2025

Introducción

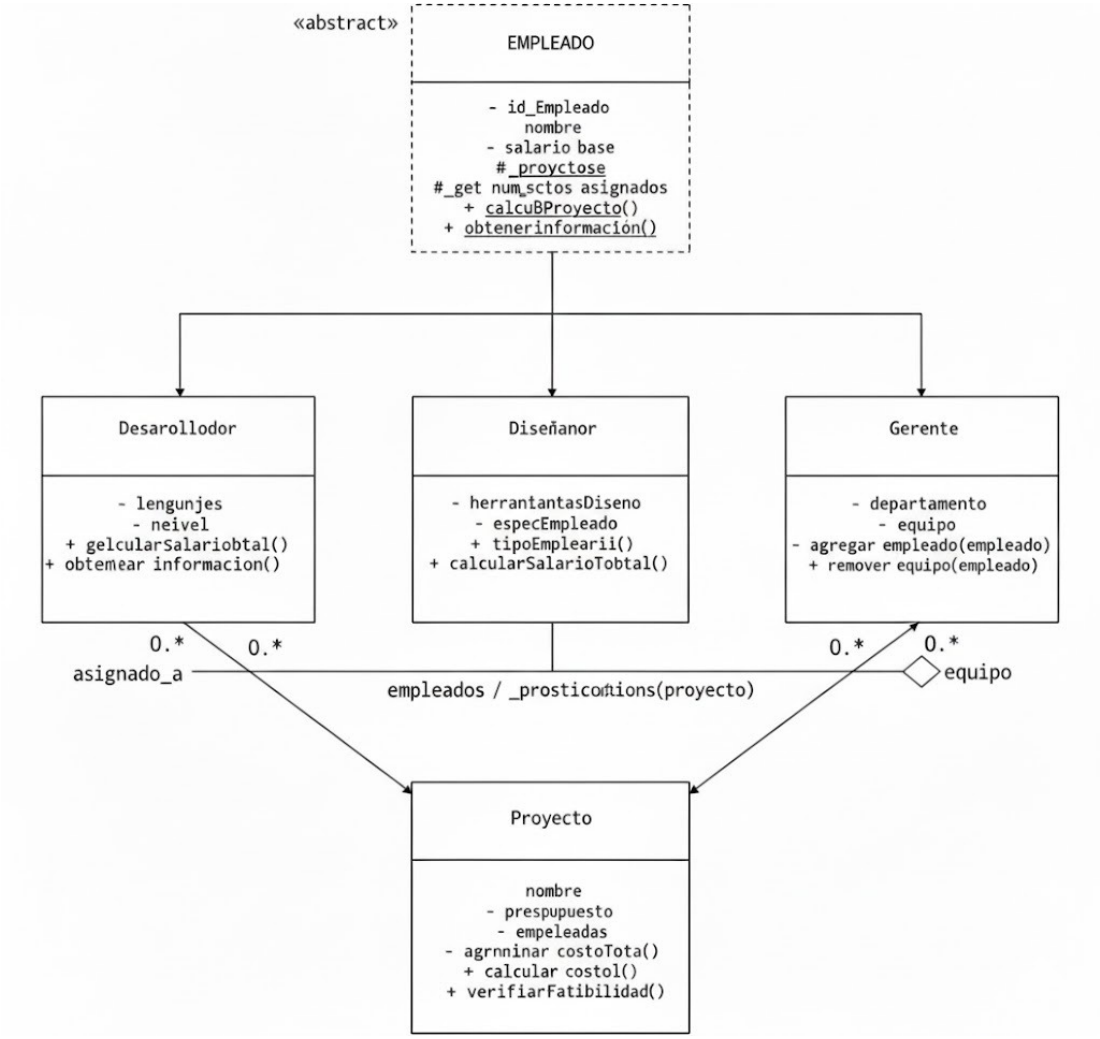
El informe a continuación trata de un proyecto presentado es un Sistema de Gestión de Personal y Proyectos desarrollado en el lenguaje Python, aplicando las indicaciones del profesor enviado en un enunciado y los conocimientos previamente adquiridos sobre la Programación Orientada a Objetos (POO) en clases. La finalidad principal de este trabajo fue cumplir con el enunciado: crear para una empresa de tecnología un sistema orientado a objetos para gestionar su personal y los proyectos en los que participan. Para lograr esto, se utilizaron y aplicaron los principios fundamentales de la POO, incluyendo Herencia, Composición, Encapsulamiento, y Polimorfismo.

También podemos decir que se implementó un caso base de prueba para asegurar que el sistema fuera funcional y cumpliera con los requerimientos. Este caso base verificó la capacidad del sistema para: Crear diferentes tipos de empleados, como Desarrollador, Diseñador y Gerente; Calcular el salario total de cada empleado según reglas específicas, lo que demuestra la aplicación del Polimorfismo; Asignar empleados a proyectos, respetando el límite de participación establecido para cada tipo de rol y finalmente evaluar la viabilidad financiera de los proyectos.

Caracas, Noviembre del 2025

Diagrama de clases

El diagrama conceptual de clases muestra la jerarquía de Herencia donde Desarrollador, Diseñador y Gerente derivan de la clase base abstracta Empleado. Además, se observa la relación de Composición donde la clase Proyecto contiene una lista de objetos Empleado.



Caracas, Noviembre del 2025

Implementación de clases

Clase base: Empleados

Esta clase es la principal y sirvió de guía para crear todas las demás la cual implemente con una Clase Abstracta (abc.ABC), sirviendo como contrato para todas las especializaciones de personal. Se implementa un Encapsulamiento mediante el uso de atributos privados (con el prefijo __) para __nombre, __id_empleado, y __salario_base, controlando el acceso a través de métodos getter.

Ya que se solicitó un Id único para cada empleado se agrega una variable global _Unico_ID que se incrementa en cada interacción para garantizar que cada empleado posea un identificador único. El método calcular_salario() es abstracto (@abc.abstractmethod), lo cual obliga a las clases hijas a definir su propia lógica de cálculo salarial.

```python
class Empleado(abc.ABC):

    def __init__(self, nombre: str, salario_base: float):
        global _Unico_ID
        self.__nombre: str = nombre
        self.__id_empleado: int = _Unico_ID
        _Unico_ID = _Unico_ID + 1
        self.__salario_base: float = salario_base
        self._proyectos_asignados: List['Proyecto'] = []

    def nombre(self) -> str:
        return self.__nombre
    def id_empleado(self) -> int:
        return self.__id_empleado
    def salario_base(self) -> float:
        return self.__salario_base
    def get_num_proyectos(self) -> int:
        return len(self._proyectos_asignados)

    @abc.abstractmethod
    def calcular_salario(self) -> float:
        pass
```

Clases: Desarrollador, Diseñador y Gerente

Las clases Desarrollador, Diseñador y Gerente aplican la Herencia al extender la clase Empleado y demuestran la Especialización de la funcionalidad al sobrescribir el método calcular_salario():

Desarrollador: Su método calcular_salario() calcula un bono fijo que depende de su nivel: $200.0 para "Junior", $500.0 para "SemiSenior", y $1000.0 para "Senior".

```
        self.nivel: str = nivel

    def calcular_salario(self) -> float:
        bono: float = 0.0
        if self.nivel == "Junior":
            bono = 200.0
        elif self.nivel == "SemiSenior":
            bono = 500.0
        elif self.nivel == "Senior":
            bono = 1000.0
        return self.salario_base() + bono
```

Diseñador: El método calcular_salario() aplica una lógica de bono basada en las herramientas que domina: $300.0 por usar "Figma", $200.0 por usar "Photoshop" o "Illustrator" (solo si no usa Figma), y $400.0 si usa tres o más herramientas.

```
usa_tres_o_mas: bool = len(self.herramientas) >= 3

if usa_figma:
    bono = bono + 300.0

if not usa_figma and usa_photoshop_illustrator:
    bono = bono + 200.0

if usa_tres_o_mas:
    bono = bono + 400.0

return self.salario_base() + bono
```

Gerente: El método calcular_salario() calcula un bono del 15% (0.15) sobre la suma total de los salarios calculados de los empleados listados en su atributo equipo. Este cálculo es un ejemplo clave de comportamiento adaptativo, ya que el Gerente llama al método calcular_salario() de cada miembro, sin importar su tipo específico, para obtener el costo total de su equipo.

```
def calcular_salario(self) -> float:
    total_salarios_equipo: float = 0.0
    for e in self.equipo:
        total_salarios_equipo = total_salarios_equipo + e.calcular_salario()

    bono_equipo: float = total_salarios_equipo * 0.15
    return self.salario_base() + bono_equipo

def mostrar_informacion(self) -> None:
    super().mostrar_informacion()
    print("  Departamento: " + self.departamento + ", Empleados a cargo: " + str(len(self.equipo)))
    print("  Bono por Equipo: $" + "{:,.2f}".format(self.calcular_salario() - self.salario_base()))
```

Clase: Proyecto

La clase Proyecto aplica el principio de Composición al mantener la lista de objetos Empleado asignados al proyecto (self.empleados).

```
def __init__(self, nombre: str, presupuesto: float):
    self.nombre: str = nombre
    self.presupuesto: float = presupuesto
    self.empleados: List[Empleado] = []
```

El método agregar_empleado() verifica las reglas de negocio, incluyendo:

Un Desarrollador tiene un límite de 3 proyectos activos, y un Diseñador un límite de 2.

Se imprime un mensaje de error si se intenta asignar a un Gerente como miembro de un proyecto.

```
if tipo_empleado == 'Desarrollador':
    limite_proyectos = 3
elif tipo_empleado == 'Diseñador':
    limite_proyectos = 2
elif tipo_empleado == 'Gerente':
    print("--- Error de Asignacion: " + empleado.nombre() + " (G
    return
else:
    limite_proyectos = 3

if empleado.get_num_proyectos() >= limite_proyectos:
    print("--- Error de Limite: " + empleado.nombre() + " (ID: "
    return

if empleado in self.empleados:
    print("--- Error de Duplicado: " + empleado.nombre() + " ya
    return

self.empleados.append(empleado)
empleado._proyectos_asignados.append(self)
print("   " + empleado.nombre() + " ha sido asignado al proyecto
```

Y el método viabilidad() evalúa si el costo_total() del personal es menor o igual al 70% del presupuesto del proyecto.

```python
def costo_total(self) -> float:
    total: float = 0.0
    for e in self.empleados:
        total = total + e.calcular_salario()
    return total

def viabilidad(self) -> bool:
    costo: float = self.costo_total()
    viable: bool = costo <= self.presupuesto * 0.7
    print("--- Viabilidad de Proyecto '" + self.no
    print("Presupuesto: $" + "{:,.2f}".format(self
    print("Viable: " + str(viable))
    return viable
```

Caso Prueba y Resultados

Para probar el sistema se realizó el caso prueba y se añadió al código para ver cómo se comportaba:

- 1 gerente con 2 desarrolladores y 1 diseñador en su equipo.
- 2 proyectos con presupuestos definidos.
- Asigna empleados a los proyectos respetando sus límites.
- Muestra la viabilidad de cada proyecto.
- Intenta asignar un cuarto proyecto a un desarrollador y captura el error.

Dando así los siguiente resulta en el cuadro adjunto e imagenes

Prueba Realizada	Descripción de la Prueba	Resultado Esperado	Resultado de la Ejecución
Cálculo de Salario del Gerente	El salario del Gerente (Diana Gerente, ID 4) se calcula al 15% del total de salarios de su equipo (dev_senior, dev_junior, designer).	El salario base del Gerente ($5000.0) debe incrementarse con el bono calculado sobre el equipo.	Correcto. El bono por equipo se calcula y se refleja en el salario final del gerente.
Restricción de Asignación de Gerente	Se intenta asignar al Gerente (Diana Gerente) como miembro de proyecto_a.	Debe imprimir un error de asignación y detener la operación.	Correcto. Se imprime el mensaje: --- Error de Asignacion: Diana Gerente (Gerente) no puede ser asignado a un proyecto como miembro....
Prueba de Límite de Proyectos	El dev_senior ya está en 3 proyectos (proyecto_a, proyecto_b, dev_senior_p3). Se intenta asignarle dev_senior_p4 (proyecto 4).	El intento en dev_senior_p4 debe fallar, imprimiendo el mensaje de límite excedido.	Correcto. Se imprime el mensaje: --- Error de Limite: Andres Senior (ID: 1) ya esta en su límite de 3 proyectos activos..
Análisis de Viabilidad	Se evalúa la viabilidad de proyecto_a y proyecto_b tras la asignación de personal.	Se evalúa si el costo total del personal no supera el 70% del presupuesto.	Correcto. Los proyectos muestran el costo estimado, el límite de 70% y el estado de viabilidad.

```
PS C:\Users\USER> & C:/Users/USER/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/USER/Pictures/proyecto/Proyecto OOP Sougleimy Aparicio.py"

           SISTEMA AVANZADO DE GESTIÓN DE PERSONAL Y PROYECTOS

  Andres Senior ha sido agregado al equipo del Gerente Diana Gerente
  Diego Junior ha sido agregado al equipo del Gerente Diana Gerente
  Alucard Designer ha sido agregado al equipo del Gerente Diana Gerente

  ESTADO INICIAL DE EMPLEADOS (0 PROYECTOS)
--- Empleado: Diana Gerente (ID: 4) ---
Salario Calculado: $6,710.00
Proyectos Asignados: 0
  Departamento: Desarrollo de Producto, Empleados a cargo: 3
  Bono por Equipo: $1,710.00
--- Empleado: Andres Senior (ID: 1) ---
Salario Calculado: $5,000.00
Proyectos Asignados: 0
  Nivel: Senior, Lenguajes: Python, JS, SQL
--- Empleado: Diego Junior (ID: 2) ---
Salario Calculado: $2,700.00
Proyectos Asignados: 0
  Nivel: Junior, Lenguajes: Java, C#
--- Empleado: Alucard Designer (ID: 3) ---
Salario Calculado: $3,700.00
Proyectos Asignados: 0
  Especialidad: UI, Herramientas: Figma, Photoshop, Illustrator
================================================

--- ASIGNACIÓN DE PROYECTOS ---
  Andres Senior ha sido asignado al proyecto 'Alfa'.
  Diego Junior ha sido asignado al proyecto 'Alfa'.
  Alucard Designer ha sido asignado al proyecto 'Alfa'.
  Andres Senior ha sido asignado al proyecto 'Omega'.
  Diego Junior ha sido asignado al proyecto 'Omega'.
--- Error de Asignacion: Diana Gerente (Gerente) no puede ser asignado a un proyecto como miembro. Solo como responsable.
```

```
  ESTADO FINAL DE EMPLEADOS (PROYECTOS ASIGNADOS)
--- Empleado: Diana Gerente (ID: 4) ---
Salario Calculado: $6,710.00
Proyectos Asignados: 0
  Departamento: Desarrollo de Producto, Empleados a cargo: 3
  Bono por Equipo: $1,710.00
--- Empleado: Andres Senior (ID: 1) ---
Salario Calculado: $5,000.00
Proyectos Asignados: 2
  Nivel: Senior, Lenguajes: Python, JS, SQL
--- Empleado: Diego Junior (ID: 2) ---
Salario Calculado: $2,700.00
Proyectos Asignados: 2
  Nivel: Junior, Lenguajes: Java, C#
--- Empleado: Alucard Designer (ID: 3) ---
Salario Calculado: $3,700.00
Proyectos Asignados: 1
  Especialidad: UI, Herramientas: Figma, Photoshop, Illustrator
================================================

--- PRUEBA DE LÍMITES DE PROYECTOS ---
  Andres Senior ha sido asignado al proyecto 'Proyecto 3'.
--- Error de Limite: Andres Senior (ID: 1) ya esta en su limite de 3 proyectos activos.

--- ANÁLISIS DE VIABILIDAD ---
--- Viabilidad de Proyecto 'Alfa' ---
Presupuesto: $50,000.00 | Costo Estimado: $11,400.00 | Limite (70%): $35,000.00
Viable: True
--- Viabilidad de Proyecto 'Omega' ---
Presupuesto: $20,000.00 | Costo Estimado: $7,700.00 | Limite (70%): $14,000.00
Viable: True
PS C:\Users\USER>
```

Caracas, Noviembre del 2025

Conclusión

En este proyecto se cumplio el objetivo principal al aplicar contodos los conceptos fundamentales de la Programación Orientada a Objetos, incluyendo la Herencia, la Especialización de la funcionalidad mediante sobrescritura de métodos, el Encapsulamiento y la Composición. Las pruebas incluidas en el código verifican que el sistema opera bajo las reglas de negocio establecidas sobre bonificaciones, límites de asignación y viabilidad financiera.

Caracas, Noviembre del 2025