

Debugging Demons



Universidad Tecnológica Nacional. Facultad Regional San Rafael.

Integrantes

Agustín Perez Pesce	Mario Cristian Sánchez
Braian Troncoso	Santiago Mendoza
Franco Sebastián Genre	Sebastián Galván
Mariano Farias	

Trabajo Práctico N° 3 – Base de Datos

 $\pmb{\Omega}$ - Añadir etiquetas a la hoja columnas.html y enviar captura.

```
lack dagger File Edit Selection View Go \cdots lack \leftarrow \Rightarrow
                                                                                                                                                                                    ፱ columnas.html × | ᢖ columnas.css

∃ columnas.css

                                                                                                                                                                                                                                 <!DOCTYPE html>
                                                                                                                                                                                                                                 <html lang="en">
  وړ
                                                                                                                                                                                                                                                         <meta charset="UTF-8">
                                                                                                                                                                                                                                                            <meta name="viewport" content="width=device-width, initial-scale=1.0">
                                                                                                                                                                                                                                                           <title>Página Web</title>
B
 (8)
                                  > OUTLINE
                        \otimes 0 \triangle 0 \begin{tabular}{lll} \textcircled{M} & 0 & \begin{tabular}{lll} \textcircled{O} & \begin{tabular}{lll} \begi
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            ^ Ф)) /с ESP 20:10
15/9/2023

∠ Escribe aquí para buscar.
```



Debugging Demons



Universidad Tecnológica Nacional. Facultad Regional San Rafael.

```
∠ Página Web

                                                                                               □ □ □ □ □ -
	imes File Edit Selection View Go \cdots \leftarrow 	o
     EXPLORER
                        columnas.html

∃ columnas.css ×

                                                                                                            ∨ PÁGINA... [t] ET ひ 🗗
                         margin: 0px;
                                  padding: 0px;
وړ
                                  list-style: none;
                                  text-decoration: none;
                                  font-family: sans-serif;
₹
留
                               .contenedor {
                                  margin: auto;
(1)
                                  width: 100%;
                                  height: auto;
margin: auto;
                                  width: 100%;
                                  height: auto;
(2)
    > OUTLINE
    > TIMELINE
   ⊗ 0 ♠ 0 ♥ 0 🖯 Connect 🕏 Live Share 🔾 tabnine starter
                                                                 へ (39) / ESP 20:11
15/9/2023
   Escribe aguí para buscar.
```

 Ω Realizar un ejemplo de las dos propiedades SOLID vistas en clase.

Principio de Responsabilidad Única (SRP):

El Principio de Responsabilidad Única establece que una clase debe tener una sola razón para cambiar. Esto significa que una clase debe tener una única responsabilidad o función en el sistema, y cualquier cambio en esa responsabilidad debe conducir a cambios en esa clase y en ninguna otra.

Ejemplo:

Supongamos que estamos desarrollando un sistema de gestión de empleados. En lugar de crear una única clase gigante que maneje todas las funcionalidades relacionadas con los empleados, aplicamos el SRP dividiendo las responsabilidades en diferentes clases:

```
Empleado
                                                                                               CalculadorDeSalario
    def __init__(self, nombre, salario):
                                                                · nombre: String
        self.nombre = nombre
                                                                                            + calcular_salario(): float

    salario: float

        self.salario = salario
class CalculadoraDeSalario:
                                                                                               GeneradorDeReportes
                                                                                             + generar_reporte(): String
    @staticmethod
    def calcular_salario(empleado):
        # Cálculos complejos para determinar el salario
        return empleado.salario
class Generador
    @staticmethod
    def generar_reporte(empleado):
        # Generar un informe basado en los datos del empleado
        return f"Nombre: {empleado.nombre}, Salario: {empleado.salario}"
```



Debugging Demons



Universidad Tecnológica Nacional. Facultad Regional San Rafael.

En este ejemplo, tenemos tres clases diferentes, cada una con una única responsabilidad: la clase Empleado almacena los datos del empleado, la clase CalculadoraDeSalario calcula el salario y la clase GeneradorDeReportes genera informes. Esto sigue el SRP, ya que cada clase tiene una razón para cambiar única.

Principio Abierto/Cerrado (OCP):

El Principio Abierto/Cerrado establece que las entidades (clases, módulos, funciones, etc.) deben estar abiertas para extenderse, pero cerradas para modificarse. Esto significa que debemos poder agregar nuevas funcionalidades o extensiones a un sistema sin modificar el código existente.

Ejemplo:

Supongamos que tenemos un sistema de procesamiento de pagos con un módulo de procesamiento de pagos existente. Aplicamos el OCP de la siguiente manera:

```
from abc import ABC, abstractmethod
                                                                             ProcesadorDePago
                                                                           procesar_pago()
   @abstractmethod
   def procesar_pago(self, monto):
                                                             Procesador De Pago Tarjeta
                                                                                       ProcesadorDePagoPaypal
                                                            + procesar_pago()
                                                                                       + procesar_pago()
          esadorDePagoTarjeta(ProcesadorDePago):
   def procesar_pago(self, monto):
        # Lógica para procesar pagos con tarjeta de crédito
        print(f"Procesando pago con tarjeta: ${monto}")
                          Pal(ProcesadorDePago):
   def procesar_pago(self, monto):
        # Lógica para procesar pagos con PayPal
        print(f"Procesando pago con PayPal: ${monto}")
```

En este ejemplo, tenemos una clase abstracta ProcesadorDePago que define un método abstracto procesar_pago. Luego, creamos dos clases concretas (ProcesadorDePagoTarjeta y ProcesadorDePagoPayPal) que extienden la funcionalidad sin modificar la clase base. Esto sigue el OCP, ya que podemos agregar más procesadores de pago en el futuro sin modificar el código existente.