**Título del Trabajo**  
  
Jarvic Balanta Santacruz  
Universidad Cooperativa de Colombia  
Facultad de Ingeniería  
Patrones de Diseño Orientado a Objetos

# Resumen

Esta actividad explora la implementación de los patrones **Singleton** y **Factory Method** en C#. Se analizaron casos prácticos donde estos patrones facilitan la gestión de recursos compartidos y la creación flexible de objetos. Se abordaron escenarios como la gestión de registros en un sistema bancario y la fabricación de vehículos con diferentes características.

# Introducción

Los patrones de diseño proporcionan soluciones reutilizables para problemas comunes en el desarrollo de software. En este ejercicio, se estudiaron dos patrones creacionales clave: **Singleton**, que garantiza una única instancia de una clase, y **Factory Method**, que permite la creación de objetos sin especificar su clase concreta. Se implementaron ejemplos prácticos en C# para comprender su aplicación y beneficios.

# Resultados

Repositorio:

* **Escenario:** Una aplicación bancaria necesita registrar eventos de usuario (inicio de sesión, transacciones, errores). Para evitar múltiples instancias del Logger y asegurar un control centralizado, se implementa el patrón Singleton.

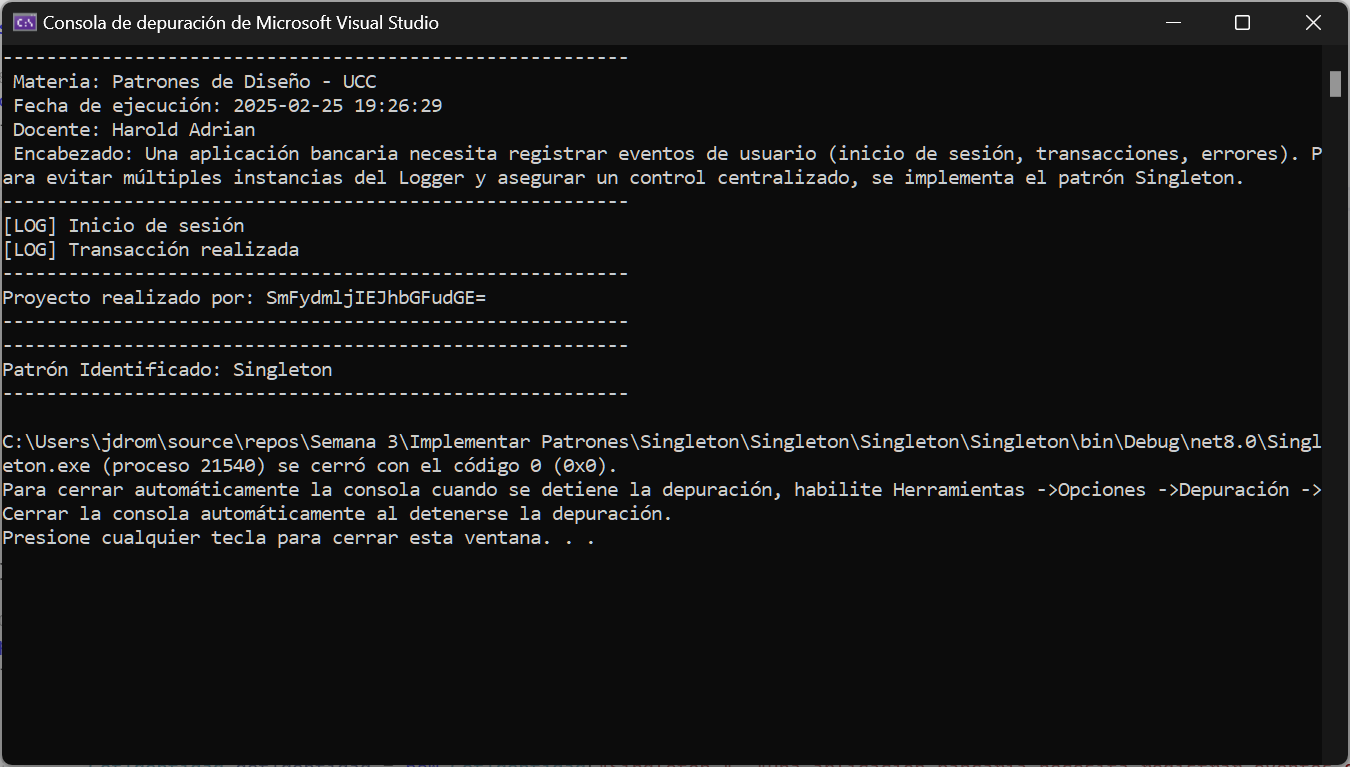


Ilustración 1.Caso de Uso: Logger en Singleton.

* **Escenario:** Una empresa automotriz fabrica autos eléctricos y a gasolina. Se requiere una fábrica que genere instancias sin especificar directamente la clase concreta.

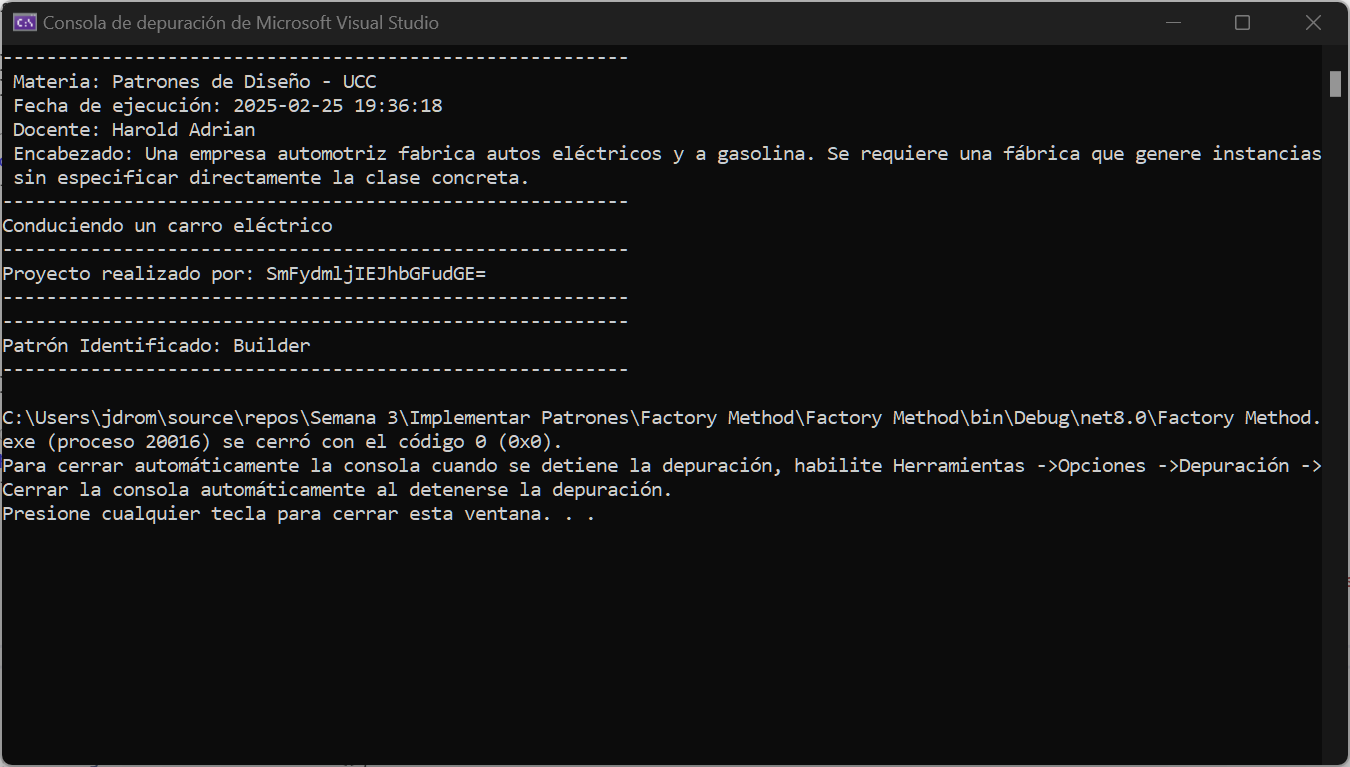


Ilustración 2. b) Caso de Uso: Creación de Vehículos.

* **Historia de usuario:** Un sistema de gestión de pedidos necesita que la conexión a la base de datos sea única para evitar conflictos y mejorar el rendimiento.

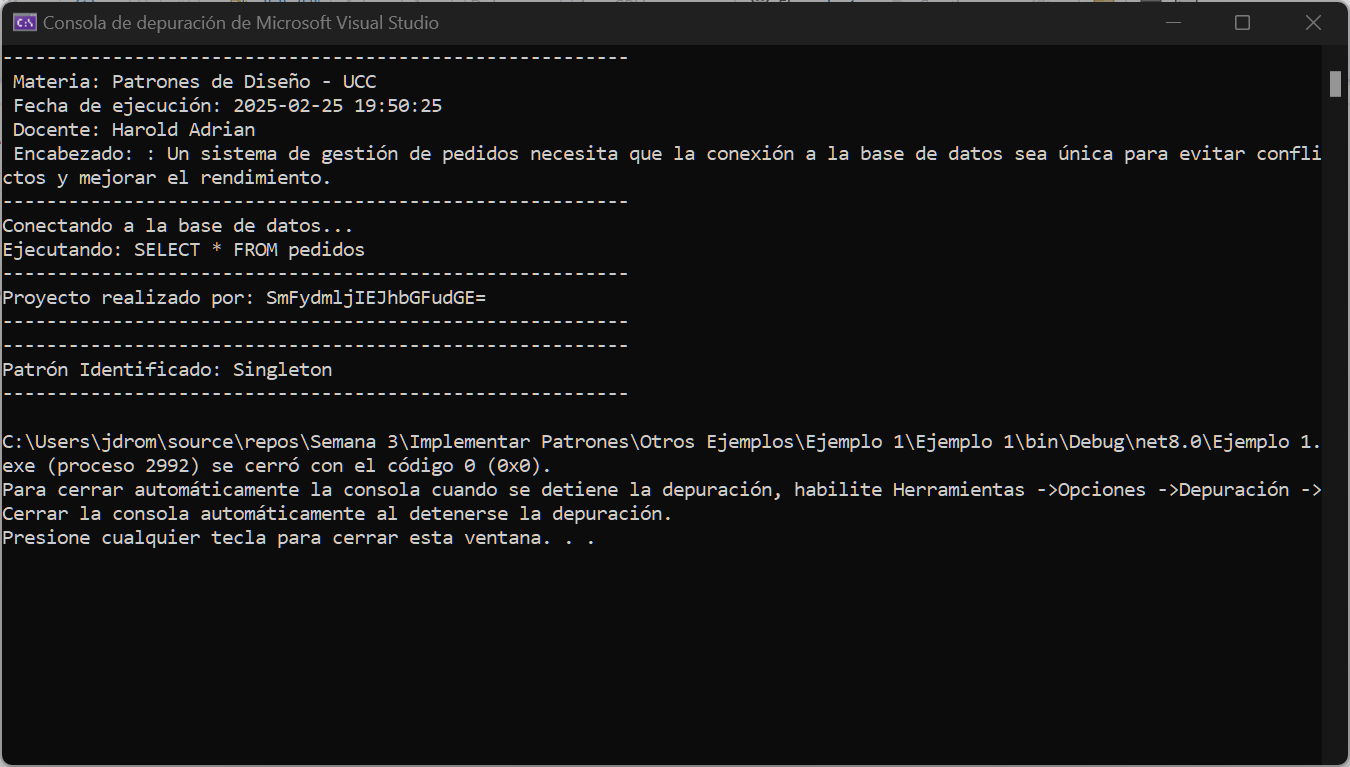


Ilustración 3.: Implementar Singleton en un servicio de base de datos.

* **Historia de usuario:** Un software de diseño gráfico necesita una fábrica que genere distintos tipos de figuras sin definir explícitamente sus clases.

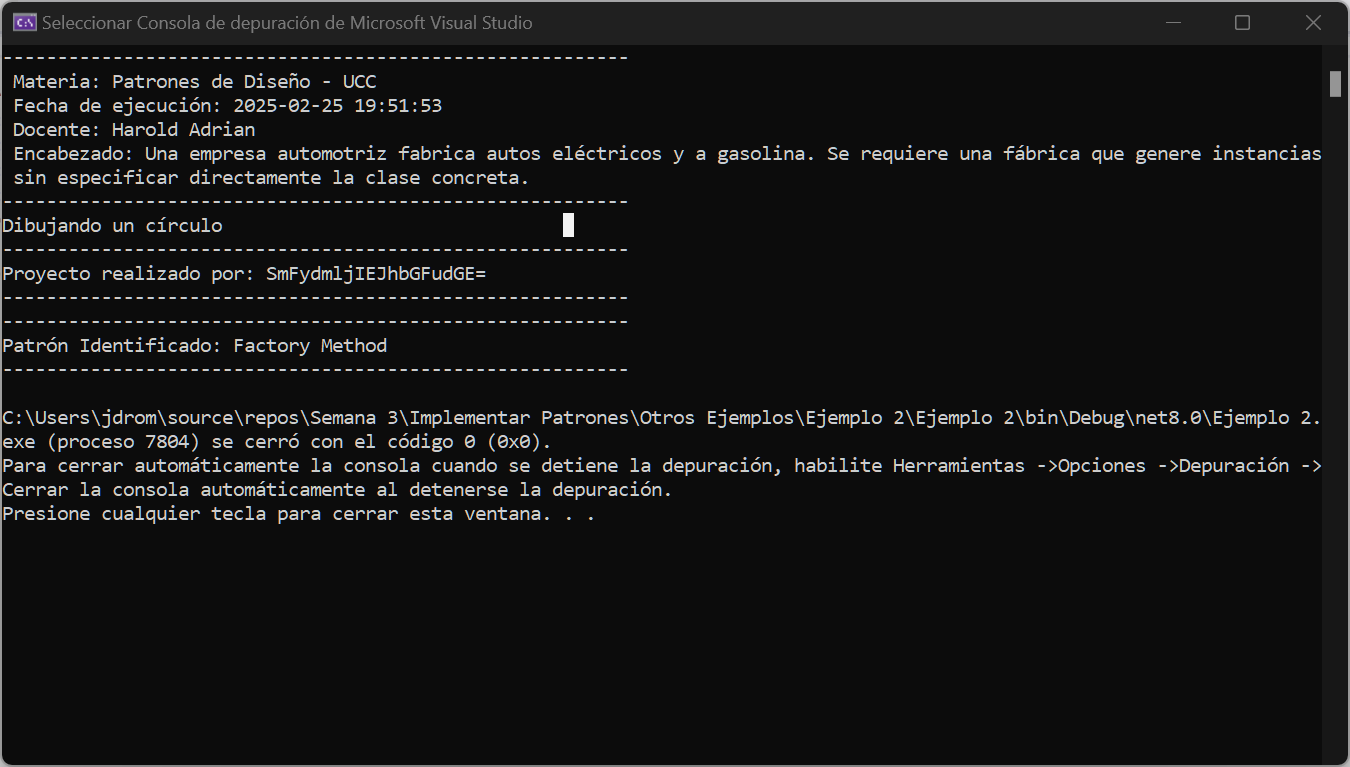


Ilustración 4.Crear una fábrica de figuras geométricas.

# Conclusiones

La implementación de **Singleton** permite asegurar un control centralizado de recursos compartidos, como la gestión de conexiones a bases de datos o el manejo de logs. Sin embargo, su uso en aplicaciones altamente concurrentes debe manejarse con precaución para evitar bloqueos y problemas de rendimiento.

Por otro lado, el **Factory Method** proporciona flexibilidad en la creación de objetos, reduciendo el acoplamiento y mejorando la extensibilidad del código. Su implementación en la fabricación de vehículos evidenció cómo este patrón facilita la gestión de múltiples tipos de instancias sin necesidad de modificar el código existente.

En conclusión, ambos patrones son herramientas esenciales en el desarrollo de software escalable y bien estructurado. Su correcta aplicación permite optimizar la creación y gestión de objetos dentro de una arquitectura orientada a objetos.

# Referencias

Bolaños Rodríguez, H. A. (2025). *Curso Patrones de Diseño OO - Semana 3.1*. Universidad Cooperativa de Colombia.

.