**Ensayo** Ruddy A. Valdez 2021-1572

En C#, la elección entre dynamic y tipos estáticos es crucial para el diseño y el rendimiento de las aplicaciones. Ambos enfoques tienen sus propias ventajas y desventajas, por lo que es crucial comprender sus consecuencias para elegir la opción más adecuada según las circunstancias.

**Tipos estáticos.**

Los tipos estáticos en C# son aquellos cuyo tipo se conoce en tiempo de compilación. Esto incluye tipos como int, string, bool, y cualquier clase o estructura creada por el usuario. La gran ventaja de los tipos estáticos es la verificación de tipo en tiempo de compilación. Al conocer el tipo de datos con el que se está trabajando, el compilador puede verificar si el código es válido y detectar los errores antes de la ejecución.

Esto hace que el refactoring y la modificación del código sean más fáciles en la medida en que las inconsistencias son eliminadas tempranamente. Además, los tipos estáticos permiten más optimizaciones ya que el compilador puede generar un código más eficiente.

Los tipos estáticos también facilitan la auto complementación y la documentación en entornos de desarrollo, lo que hace que sea más fácil trabajar con código.

**Dynamic.**

El tipo Dynamic fue introducido en C# 4.0 y postula la postergación de la verificación de tipos hasta el tiempo de ejecución. Esto significa que el compilador no verifica los tipos hasta que el código se ejecute. Aunque dinámicamente, esto también supone un mayor riesgo.

La mayor ventaja del tipo Dynamic es que puede usarse en los escenarios de COM interop, reflexión y otros lugares donde los tipos pueden diferir. Además, es útil al integrarse con lenguajes de scripting, o cuando se trabaja con datos que pueden tener una estructura diferente, como JSON o XML, cuyo esquema es desconocido. Dicho esto, Dynamic es un enfoque que conlleva serias desventajas.

La falta de verificación de tipos durante el tiempo de compilación puede provocar errores de tiempo de ejecución más difíciles de rastrear y corregir. Dynamic también puede influir en la ejecución del programa, ya que se requieren recursos adicionales para resolver el tipo en tiempo de ejecución en comparación con el tiempo de compilación.

Dynamic es posible y conveniente para los siguientes objetivos:

**1. Interoperabilidad con COM.** Dynamic en C# puede usarse sin restricciones durante la comunicación con módulos COM cuando los objetos no se transmiten con estrictos relacionados.

**2.** **Soporte para APIs que usan reflexión.** Dynamic puede usarse en situaciones en las que la reflexión de las API se realiza a través de la manipulación de objetos con la reflexión.

**3. Datos con esquemas variables.** Se permite Dynamic en situaciones cuando una aplicación debe funcionar con datos con estructuras desconocidas o flexibles, por ejemplo, JSON o XML.

En resumen, aunque el uso de dynamic ofrece flexibilidad y es útil en escenarios específicos, debe emplearse con precaución debido a los riesgos asociados con la falta de verificación de tipos en tiempo de compilación y el posible impacto en el rendimiento. Para la mayoría de los casos, especialmente cuando el tipo de datos es conocido y fijo, los tipos estáticos son preferibles por su seguridad en tiempo de compilación y eficiencia en el rendimiento. El uso de dynamic debe reservarse para situaciones donde su flexibilidad es necesaria y justificada.