

## Prompt Utilizado

*Estoy trabajando en una clase Chofer donde, en el constructor, se define el tipo de licencia y se valida en función de la edad. Sin embargo, si durante la ejecución del programa se cambia la propiedad TipoLicencia, la validación con la edad no se vuelve a ejecutar. En C#, ¿cuál es la mejor práctica para implementar una validación que asegure la coherencia de la licencia al modificarla dinámicamente?*

## Respuestas de los Modelos de Lenguaje

### **RESPUESTA DEL MODELO GEMINI-2.5-PRO-EXP-03-25:**

El modelo sugiere implementar la lógica de validación dentro del setter de la propiedad TipoLicencia. De esta manera, cada vez que se asigne un nuevo valor a la propiedad, la validación se ejecutará automáticamente.

### **Respuesta completa:**

¡Excelente pregunta! Es un escenario común donde el estado inicial es válido, pero las modificaciones posteriores pueden romper la coherencia del objeto.

La mejor práctica en C# para manejar esto es **\*\*implementar la lógica de validación dentro del `setter` de la propiedad `TipoLicencia`\*\***. De esta manera, cada vez que se intente asignar un nuevo valor a la propiedad, la validación se ejecutará automáticamente.

Aquí te muestro cómo implementarlo:

```
```csharp
using System;

// 1. Define los tipos de licencia (mejor usar un enum)
public enum TipoLicencia
{
    Ninguna,
    Particular, // Ejemplo: Edad mínima 18
    Profesional // Ejemplo: Edad mínima 21
}

// 2. Define la clase Chofer
public class Chofer
{
    private int _edad;
    private TipoLicencia _tipoLicencia; // Campo privado para almacenar el valor

    // Propiedad pública para la Edad
    public int Edad
    {
        get { return _edad; }
        set
        {
            if (value < 0) // Validación básica de edad
            {

```

```

        throw new ArgumentOutOfRangeException(nameof(Edad), "La edad no puede
ser negativa.");
    }
    // Validar si el *nuevo* valor de edad es compatible con la licencia *actual*
    ValidarCoherenciaEdadLicencia(value, _tipoLicencia);
    _edad = value;
}
}
// Propiedad pública para el Tipo de Licencia
public TipoLicencia TipoLicencia
{
    get { return _tipoLicencia; }
    set
    {
        // Validar si el *nuevo* valor de licencia es compatible con la edad *actual*
        ValidarCoherenciaEdadLicencia(_edad, value);
    }
}

```

***RESPUESTA DEL MODELO CLAUDE-3-7-SONNET-20250219:***

Este modelo también recomienda el uso de propiedades con getters y setters personalizados para validar cada vez que se cambie el valor. Además, sugiere implementar un método privado `EsLicenciaValidaParaEdad` que contenga la lógica de validación específica para cada tipo de licencia y edad.

**Respuesta completa:**

Para implementar una validación que asegure la coherencia entre el tipo de licencia y la edad cuando se modifica la propiedad dinámicamente en C#, lo mejor es usar propiedades con getters y setters personalizados. Así puedes validar cada vez que se cambie el valor.

```

```csharp
public class Chofer
{
    private int _edad;
    private string _tipoLicencia;

    public Chofer(int edad, string tipoLicencia)
    {
        Edad = edad;
        TipoLicencia = tipoLicencia; // Aquí se validará automáticamente
    }

    public int Edad
    {
        get => _edad;
        set
        {
            _edad = value;
        }
    }
}

```

```

        ValidarLicenciaConEdad(); // Validar licencia cuando cambia la edad
    }
}

public string TipoLicencia
{
    get => _tipoLicencia;
    set
    {
        if (EsLicenciaValidaParaEdad(value, _edad))
        {
            _tipoLicencia = value;
        }
        else
        {
            throw new ArgumentException($"El tipo de licencia { value } no es válido para la
edad { _edad }");
        }
    }
}

private bool EsLicenciaValidaParaEdad(string tipoLicencia, int edad)
{
    // Implementa tu lógica de validación aquí
    switch (tipoLicencia)
    {
        case "A": return edad >= 16;
        case "B": return edad >= 18;
        case "C": return edad >= 21;
        default: return false;
    }
}

private void ValidarLicenciaConEdad()
{
    // Si ya hay una licencia asignada, verificar que siga siendo válida
    if (_tipoLicencia != null && !EsLicenciaValidaParaEdad(_tipoLicencia, _edad))
    {
        // Puedes lanzar una excepción o manejar de otra forma
        throw new InvalidOperationException($"La licencia { _tipoLicencia } ya no es
válida para la edad actual { _edad }");
    }
}
}
}

```

## **Conclusión**

Ambos modelos coinciden en que la mejor práctica para asegurar la coherencia entre el tipo de licencia y la edad en una clase Chofer es implementar la validación dentro del setter de la propiedad TipoLicencia. Esta estrategia garantiza que cualquier modificación de la propiedad sea validada en tiempo real, manteniendo la integridad de los datos. Además, encapsular la lógica de validación en métodos privados, como `EsLicenciaValidaParaEdad`, mejora la claridad y mantenibilidad del código.