

UNIVERSIDAD MARIANO GALVEZ DE GUATEMALA  
FACULTAD DE INGENIERIA EN SISTEMAS DE INFORMACION Y CIENCIAS DE  
LA COMPUTACION  
CENTRO UNIVERSITARIO CAMPUS JUTIAPA

**FACULTAD:**

INGENIERIA EN SISTEMAS

**CURSO:**

PROGRAMACION I

**CATEDRATICO:**

ING RUDIN AYALA

**ALUMNO:**

ANTONY EZEQUIEL PINEDA PINEDA

**CARNE:**

0905-24-11747

## LLM Utilizado: ChatGPT (GPT-4)

### Prompt utilizado:

*"En C#, tengo una clase Chofer que valida el tipo de licencia en el constructor según la edad. Sin embargo, si el Tipo Licencia cambia en tiempo de ejecución, no hay un método que lo valide. ¿Cómo puedo asegurarme de que cualquier modificación de TipoLicencia sea coherente con la edad? ¿Cuál es la mejor práctica para hacerlo?"*

---

### ◆ Respuesta y Análisis de ChatGPT

ChatGPT sugirió varias estrategias para abordar el problema:

- 1) **Validación dentro del set de la propiedad TipoLicencia**
  - a. Implementar validaciones dentro del set para que cualquier cambio en TipoLicencia sea verificado antes de ser aplicado.
  - b. Lanzar excepciones si la combinación de edad y licencia es inválida.
- 2) **Validación al cambiar Edad**
  - a. Si la Edad cambia en tiempo de ejecución, asegurarse de que la licencia sigue siendo válida.
  - b. Se recomienda que al modificar Edad, se revalide TipoLicencia.
- 3) **Encapsulación de la lógica en un método privado**
  - a. Para mantener un código más limpio y reutilizable, la validación se coloca en un método aparte (ValidarLicencia).
- 4) **Alternativas avanzadas (para aplicaciones más grandes)**
  - a. Implementar IValidableObject si se trabaja con frameworks como ASP.NET Core.
  - b. Usar eventos (PropertyChanged) si se requiere un comportamiento reactivo.

### La mejor manera de Hacerlo:

La mejor practica recomendada es encapsular la validación en el set de la propiedad y usar un método privado, ya que evita valores inválidos en tiempo de ejecución, mantiene la lógica encapsulada y también reutilizable, y ultimo pero no menos importante, maneja de un manera correcta cambios tanto en Edad como en TipoLicencia

## ◆ Implementación Recomendada por ChatGPT

csharp

📄 Copiar

✎ Editar

```
public class Chofer
{
    public string Nombre { get; set; }
    private int _edad;
    private string _tipoLicencia;

    public int Edad
    {
        get => _edad;
        set
        {
            _edad = value;
            ValidarLicencia(_tipoLicencia, _edad);
        }
    }

    public string TipoLicencia
    {
        get => _tipoLicencia;
        set
        {
            ValidarLicencia(value, _edad);
            _tipoLicencia = value.ToUpper();
        }
    }

    public Chofer(string nombre, int edad, string tipoLicencia)
    {
        Nombre = nombre;
        Edad = edad;
        TipoLicencia = tipoLicencia;
    }

    private void ValidarLicencia(string tipoLicencia, int edad)
    {
        tipoLicencia = tipoLicencia.ToUpper();
        if ((tipoLicencia == "A" && edad >= 21) ||
            (tipoLicencia == "B" && edad >= 18) ||
            (tipoLicencia == "C" && edad >= 18))
        {
            // Licencia válida
        }
        else
        {
            throw new ArgumentException($"Licencia {tipoLicencia} no válida para la edad {edad}");
        }
    }
}
```

**LLM Utilizado:** Gemini (Modelo de Lenguaje Grande de Google)

**Prompt utilizado:**

"En C#, tengo una clase Chofer que valida el tipo de licencia en el constructor según la edad. Sin embargo, quiero que si la propiedad TipoLicencia cambia en el transcurso del programa, se valide automáticamente en función de la edad del chofer. Actualmente, la validación solo ocurre cuando se inicializa el objeto. ¿Cuál es la mejor manera de implementar esta validación para que cada vez que TipoLicencia cambie, se verifique si es válido en función de la Edad? ¿Cuál sería la mejor práctica para hacerlo?"

**Respuesta y Análisis de Gemini:**

Gemini analizó tu solicitud y proporcionó las siguientes estrategias para asegurar la validación de `TipoLicencia` en función de la `Edad` en tiempo de ejecución:

- **Validación dentro del 'set' de la propiedad `TipoLicencia`:**
  - Implementar la lógica de validación directamente dentro del 'set' de la propiedad `TipoLicencia`.
  - Esto garantiza que cualquier cambio en `TipoLicencia` se verifique antes de ser aplicado.
  - Lanzar excepciones (`ArgumentException`) si la combinación de edad y licencia es inválida.
- **Validación al cambiar la propiedad `Edad`:**
  - Asegurar que al modificar la propiedad `Edad`, la propiedad `TipoLicencia` se revalide para mantener la coherencia.
  - Se recomienda que al modificarse la edad, se vuelva a validar que la licencia siga siendo válida.
- **Encapsulación de la lógica de validación en un método privado:**
  - Extraer la lógica de validación a un método privado (`ValidarTipoLicencia`) para mejorar la legibilidad y la reutilización del código.
- **Uso de eventos:**
  - Utilizar eventos para notificar cuando la propiedad `TipoLicencia` cambia, y realizar la validación en el manejador del evento.

**La mejor manera de Hacerlo:**

Para asegurar que la propiedad `TipoLicencia` de tu clase `Chofer` siempre sea válida en función de la `Edad`, la mejor práctica es encapsular la lógica de validación dentro del set de la propiedad `TipoLicencia`. Esto implica crear un método privado, como `ValidarTipoLicencia`, que contenga las reglas de validación y llamarlo desde el set cada vez que se intente modificar `TipoLicencia`. Además, es crucial revalidar `TipoLicencia` si la `Edad` del chofer cambia en tiempo de ejecución. Este enfoque garantiza que los datos de la clase permanezcan coherentes y evita estados inválidos.

## Implementación recomendada por Gemini

C#



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

    public Chofer(int edad, string tipoLicencia)
    {
        _edad = edad;
        _tipoLicencia = ValidarTipoLicencia(tipoLicencia);
    }

    public string TipoLicencia
    {
        get { return _tipoLicencia; }
        set { _tipoLicencia = ValidarTipoLicencia(value); }
    }

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

    private string ValidarTipoLicencia(string tipoLicencia)
    {
        if (_edad < 18 && tipoLicencia != "A")
        {
            throw new ArgumentException("Un chofer menor de 18 años solo puede tener tipo A");
        }
        // Agrega aquí otras validaciones según tus reglas de negocio
        return tipoLicencia;
    }
}
```