LLMs consultados:

1.Copilott

2. Perplexity AI (versión actualizada a marzo de 2025).

Prompts utilizados:

1. Prompt para Microsoft Copilot:

Estoy realizando un proyecto en C# denominado `Chofer`. En el constructor de la clase, se verifica la coherencia entre la propiedad `TipoLicencia` y la propiedad `Edad`. Las licencias exigen un mínimo de edad (23 años para la categoría 'A', 21 para la categoría 'B', 18 para la categoría 'C' o 'M'). No obstante, no dispongo de un procedimiento que verifique si la propiedad `TipoLicencia` se altera tras la creación del objeto. ¿Cómo puedo asegurar que cualquier modificación respete las normas empresariales? ¿Cuál es la técnica más efectiva para llevarlo a cabo?

2. Prompt para Perplexity:

Estoy creando una clase denominada `Chofer` en C#. En un principio, en el constructor de la clase, confirmé que la propiedad `TipoLicencia` concuerde con la edad del conductor. Las licencias exigen una edad mínima, tales como 23 años para la licencia de tipo 'A', 21 años para la de tipo 'B' y 18 años para las de tipo 'C' o 'M'. No obstante, noté que si durante la ejecución del programa se altera la propiedad `TipoLicencia`, no existe una validación que lo confirme. ¿Cómo puedo establecer esta validación durante el tiempo de ejecución y qué práctica es la más adecuada para gestionarlo? ¿Podrías incorporar ejemplos concretos y potenciales optimizaciones en el diseño?

Soluciones y la mejor forma de hacerlo:

<u>Solución propuesta por Microsoft Copilot:</u>

Microsoft Copilot recomendó implementar un setter personalizado que directamente valida el nuevo valor asignado. Este enfoque asegura que cualquier cambio cumpla con las reglas de negocio. Ejemplo:

Codigo:

```
private string _tipoLicencia;
 public string TipoLicencia
 {
     get { return _tipoLicencia; }
     set
     {
         if (!EsEdadAdecuada(Edad, value)) // Validamos
cuando se asigna un nuevo valor
         {
             throw new Exception($"La edad {Edad} no es
adecuada para la licencia tipo {value}");
         }
         _tipoLicencia = value;
     }
 }
```

Solución propuesta por Perplexity:

Esta solución utiliza un método explícito, junto con enumeraciones y excepciones personalizadas para mejorar la escalabilidad y claridad.

Codigo.

```
  public void CambiarLicencia(string nuevaLicencia)
  {
      if (!EsEdadAdecuada(Edad, nuevaLicencia))
      {
          throw new Exception("La edad no es adecuada
para la Licencia seleccionada");
      }
      TipoLicencia = nuevaLicencia;
  }
```

¿Cuál es la mejor forma de hacerlo?

La propuesta de Perplexity AI se presenta más directa y adecuada para aplicaciones escalables debido a su:

1. Modularidad: Utiliza métodos claros y controlados para manejar cambios en la licencia.
2. Enumeraciones: Mejora la claridad y la validación de valores válidos para .
3. Excepciones personalizadas: Ofrecen contexto específico para errores, lo que facilita el manejo de excepciones en proyectos más complejos.

Sin embargo, la solución de Microsoft Copilot es perfectamente válida para proyectos más pequeños y simples, ya que es más directa y fácil de implementar.