Propiedades en C#

Introducción

En C#, **las propiedades** son una forma de exponer campos de una clase de manera controlada. Permiten encapsular la lógica de acceso (`get`) y de asignación (`set`) de valores, de manera que se mantiene la flexibilidad de un método y la simplicidad de un campo.

Las **propiedades indexadas** (o *indexers*) permiten a los objetos comportarse como si fueran colecciones, accediendo a sus elementos mediante índices como en los arrays o listas.

Propiedades simples

Propiedad básica

Una propiedad básica tiene un **getter** y un **setter**.

```
class Persona{
    private string _nombre;

    public string Nombre    {
        get { return _nombre; }
        set { _nombre = value; }
    }
}
```

Uso:

```
var p = new Persona();
p.Nombre = "Ana";   // setter
Console.WriteLine(p.Nombre); // getter → "Ana"
```

Propiedades automáticas

Cuando no necesitamos lógica especial, C# permite **propiedades automáticas** que crean un campo de respaldo (*backing field*) de manera implícita:

```
class Persona{
    public int Edad { get; set; }
}
```

Uso:

```
var p = new Persona();
p.Edad = 20;
Console.WriteLine(p.Edad); // 20
```

Propiedades con lógica de validación

Podemos controlar los valores asignados en `set`.

```
class Usuario{
    private string _nombre;

    public string Nombre {
        get => _nombre;
        set => _nombre = (value.Length > 10) ? value.Substring(0, 10) :
value;
    }
}
```

Uso:

```
var p = new Persona();
p.Edad = 20;
Console.WriteLine(p.Edad); // 20
```

Propiedades de sólo lectura o sólo escritura

- Solo lectura → solo `get`.
- Solo escritura → solo `set` (menos común).

```
class Circulo{
    public double Radio { get; set; }
    public double Area {
        get { return Math.PI * Radio * Radio; }
    }
}
```

Uso:

```
var c = new Circulo { Radio = 5 };
Console.WriteLine(c.Area); // 78.5398...
```

Propiedades Indexadas (Indexers)

Un **indexer** permite que una clase se acceda como si fuera un array o diccionario.

```
class Agenda{
    private string[] _contactos = new string[10];

    public string this[int i] {
        get { return _contactos[i]; }
        set { _contactos[i] = value; }
    }
}
```

Uso:

```
var a = new Agenda();
a[0] = "Ana";
Console.WriteLine(a[0]); // Ana
```

Indexadores con clave personalizada

Podemos definir indexadores con otros tipos (ejemplo: `string`).

```
class DiccionarioSimple{
    private Dictionary<string, string> _datos = new();

    public string this[string clave] {
        get => _datos.ContainsKey(clave) ? _datos[clave] : "No
encontrado";
        set => _datos[clave] = value;
    }
}
```

Uso:

```
var d = new DiccionarioSimple();
d["coche"] = "car";
Console.WriteLine(d["coche"]); // car
Console.WriteLine(d["avión"]); // No encontrado
```