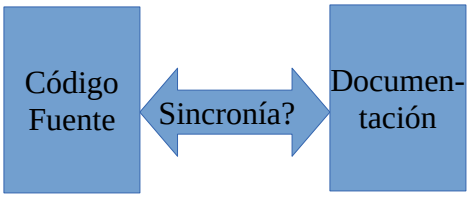
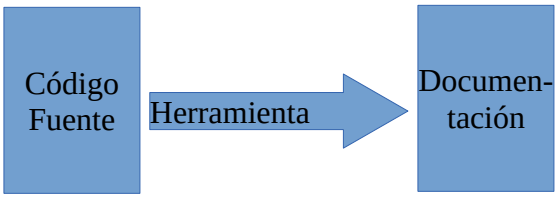
Documentación en C#

1 Introducción

En un proyecto, se generan muchos tipos de documentación: análisis, diseño, codificación y pruebas. En este texto, nos centraremos en la generación de documentación durante la fase de codificación, es decir, la documentación que describe el diseño (a bajo nivel) empleado en el propio código de la aplicación.



El gran problema de la documentación es la problemática de mantener en sincronía el propio código de la aplicación y la documentación en sí misma. Por esto mismo, dado que resulta ser imposible solventar esta falta de sincronía, la documentación se escribe en el código, y se genera en el formato deseado a partir del código fuente, utilizando una herramienta que interpreta y extrae dicha documentación.



2 C# XML Doc

El tipo de documentación que emplea C# no es la más extendida, **JavaDoc**, sino una versión propia basada en HTML. Cada parte de documentación se incluye en un nodo, haciendo posible no solo extraerla posteriormente, sino que se pueda incluso utilizar desde la propia herramienta.

Propósito	Etiqueta JavaDoc	C# XML Doc
Resumen del objetivo	*brief*	*summary*
Descripción del parámetro	*param*	*param name = “”*
Descripción del retorno	*return*	*returns*
Referencia cruzada	*see*	*see cref=””*

La documentación XML de C# se sitúa siempre justo encima del elemento a documentar, bien sea una clase, método, propiedad o atributo. Para distinguir un comentario con documentación de un comentario “normal”, es necesario comenzarlos con “///”.

Nótese que se deben documentar siempre todos los elementos de la aplicación. Si bien a un cliente, en buena lógica, sólo le interesa la documentación sobre clases y miembros públicos, para la creación de documentación interna es buena idea escribirla toda. Posteriormente se podrá elegir entre generar toda la documentación (para uso interno) o solo aquellas entidades públicas (para clientes).

La norma a seguir es documentar sucintamente lo obvio, y profusamente aquel comportamiento que por cualquier razón resulte inesperado.

Ejemplo

```
/// <summary>
/// Representa coordenadas cartesianas bidimensionales.
/// </summary>
class Punto {
	/// <summary>
	/// Crea nuevos puntos, a partir de sus dos coordenadas.
	/// <param name="x">La coordenada horizontal.</param>
	/// <param name="y">La coordenada vertical.</param>
	/// </summary>
	public Punto(int x, int y)
	{
		this.X = x;
		this.Y = y;
	}

	/// <summary>
	/// Obtiene o modifica la coordenada horizontal, como int.
	/// </summary>
	public int X {
		get; set;
	}

	/// <summary>
	/// Obtiene o modifica la coordenada vertical, como int.
	/// </summary>
	public int Y {
		get; set;
	}

	/// <summary>
	/// Obtiene la distancia a un segundo punto.
	/// </summary>
	/// <param name="b">El segundo <see cref="Punto"/>.</param>
	/// <returns>La distancia entre este punto y b, como int.</returns>
	public int GetDistancia(Punto b)
	{
		int toret = Math.Pow( this.X – b.X, 2 );

		toret += Math.Pow( this.Y – b.Y, 2 );
		toret = Math.Sqrt( toret );

		return toret;
	}
}
```