Instalación del entorno de desarrollo

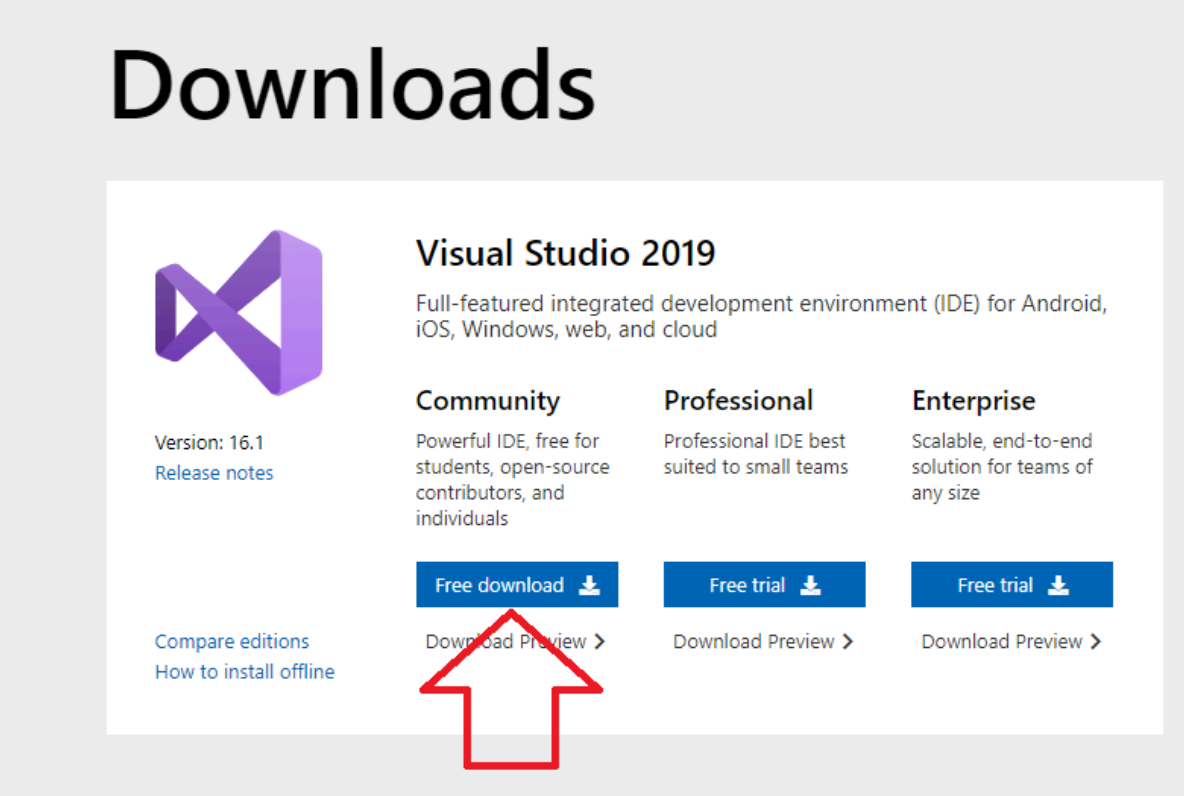
1 - Descargar e instalar Visual Studio

En esta primera entrada empezaremos por lo más básico pero a la vez más importante. consiste en instalar y poner en funcionamiento nuestro entorno de desarrollo.

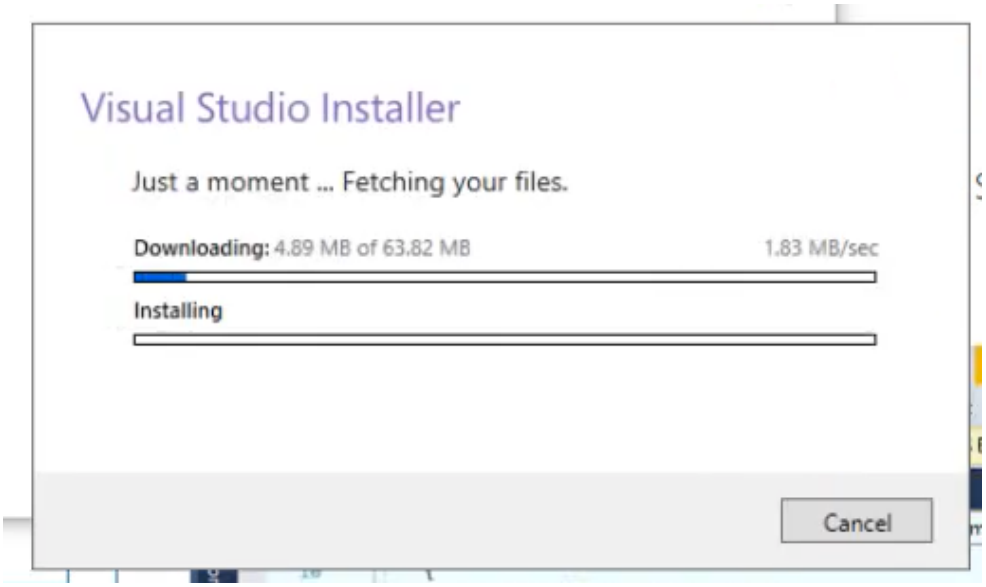
Un entorno de desarrollo es aquella aplicación que utilizamos para programar.

En concreto para programar en .Net utilizaremos visual studio, el cual lo podemos **descargar desde aquí en español de manera gratuita y oficial.**.

Una vez en el enlace, descargamos la versión comunidad.



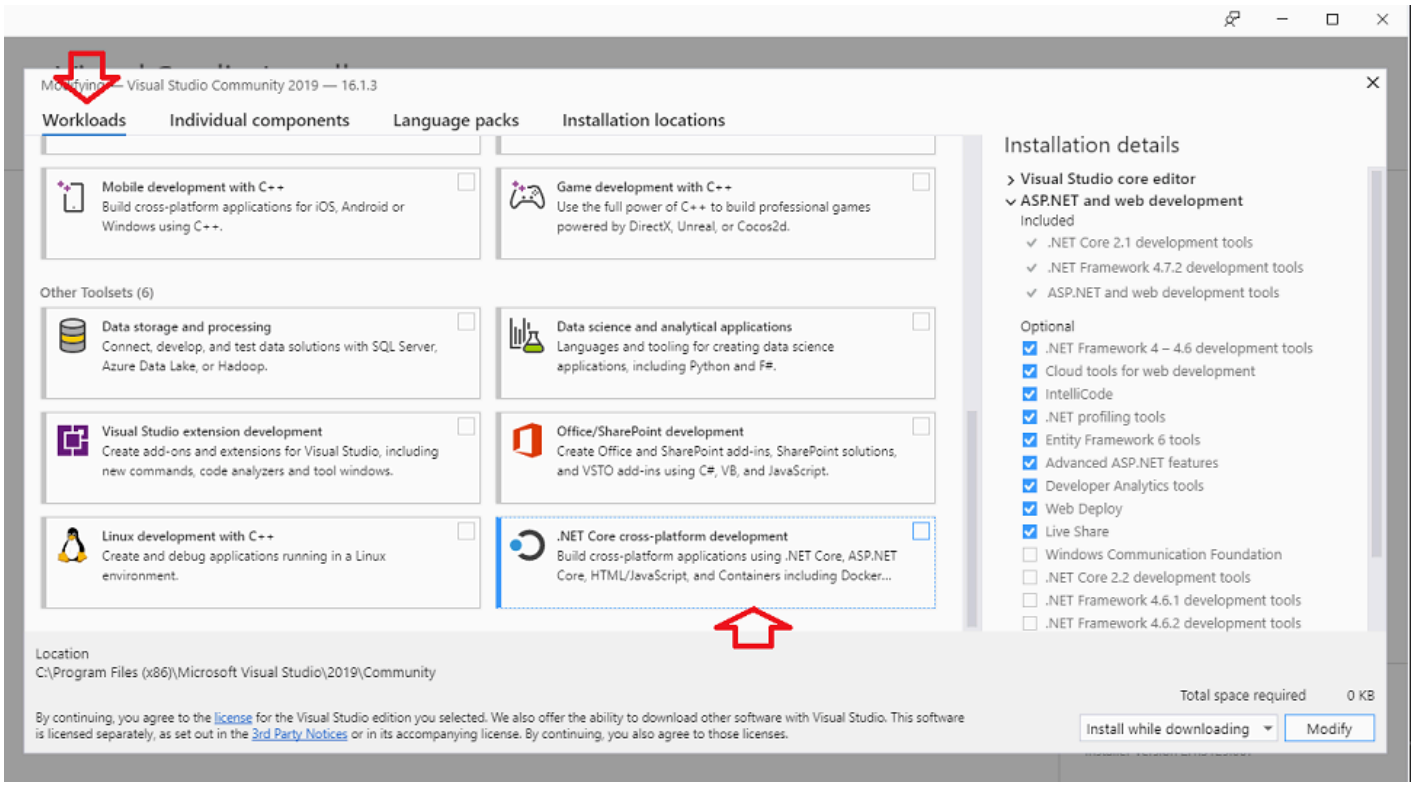
Lo que nos descarga es el launcher del instalador. por lo tanto, una vez descargado lo tenemos que abrir. y nos descargara el instalador



Una vez finalice la instalación, se nos abrirá automáticamente el instalador de visual studio. Si por ejemplo tuviésemos la versión de 2017 junto con la de 2019 se nos juntarían ambas en un solo instalador.

La ventana que veremos es como la siguiente, la cual contiene una serie de componentes que podemos añadir a nuestro visual studio.

estos componentes los podemos añadir tanto individualmente, como por paquetes. lo cual es mucho más cómodo.



Para nosotros solo es necesario uno de los paquetes, el que incluye ".Net Core." y pulsamos en instalar.

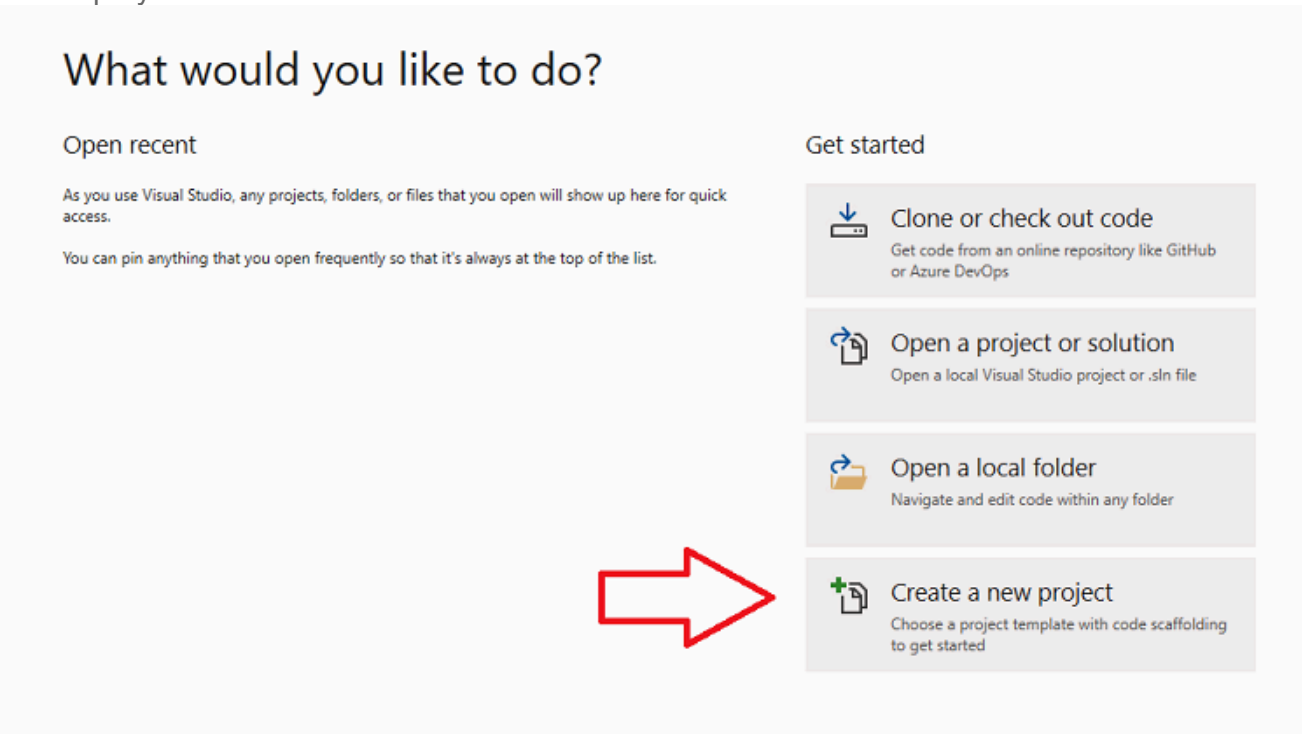
El tamaño de la descarga es más o menos grande, unos 7Gb por lo que puede tardar un rato. cuando finalice, veremos que en el mismo launcher tendremos un botón que pone

"Lanzar" al hacer click sobre el (o buscarlo en el menú de inicio) se nos abrirá visual studio 2019.

2 - Creación de una aplicación en Visual Studio

En la siguiente ventana tendremos varias opciones

- Clonar código (de un repositorio)
- Abrir un proyecto o solución
- Abrir una carpeta
- Crear un proyecto nuevo



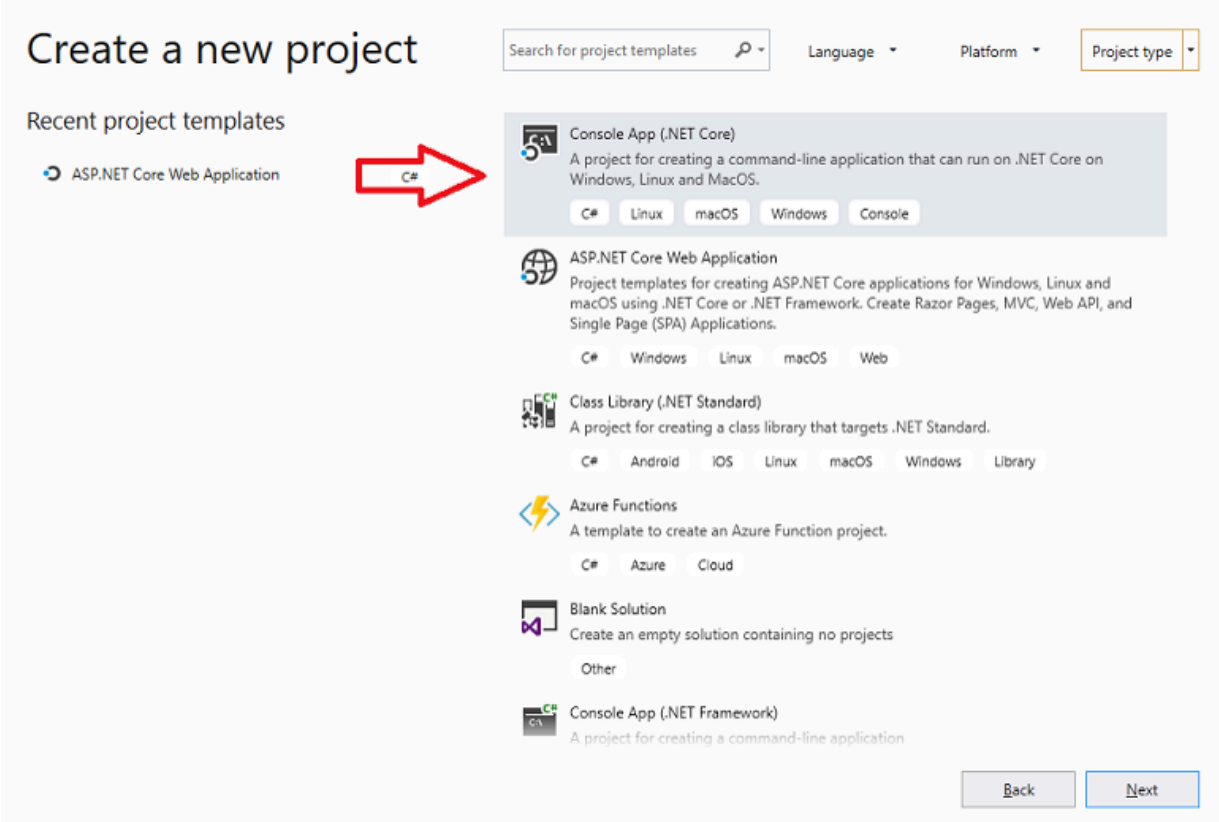
En nuestro caso la opción que nos interesa es la de crear un proyecto nuevo.

Al pulsar en ella se nos abrirá otro menú en el cual nos dejará introducir que tipo de proyecto queremos.

en esta venta podemos encontrar varios filtros

- Tipo de proyecto reciente utilizado
- Lenguaje del proyecto (en nuestro caso utilizaremos c#)
- Plataforma compatible con el tipo de proyecto
  - Ya que no es lo mismo programar para android que para IOs y todo ello lo podemos hacer desde visual studio
- Tipo de proyecto
  - Permite crear tanto aplicaciones móviles, como de escritorio, IoT, y muchas mas

Nosotros tendremos que indicar la primera opción. Aplicación de consola (.NET Core)



Pulsamos en siguiente y nos indicará que indiquemos un nombre para el proyecto. La carpeta donde se va a encontrar, yo personalmente recomiendo poner una carpeta en C:\ que se llame proyectos, y ahí todos los proyectos en los que se vaya a trabajar. y finalmente un nombre para la solución. Comúnmente el mismo que para el proyecto.

Y ya esta, esto nos creara nuestra primera aplicación de consola.

Por qué .NET Core

Hay varios motivos por los que recomiendo utilizar .NET Core. El principal y más importante es porque corre en todas las plataformas, tanto Windows, como iOS, como Linux, esta web mismo está programada en .NET Core y corriendo en un servidor Linux - prepararé un vídeo para mostrar los pasos de como hacerlo -

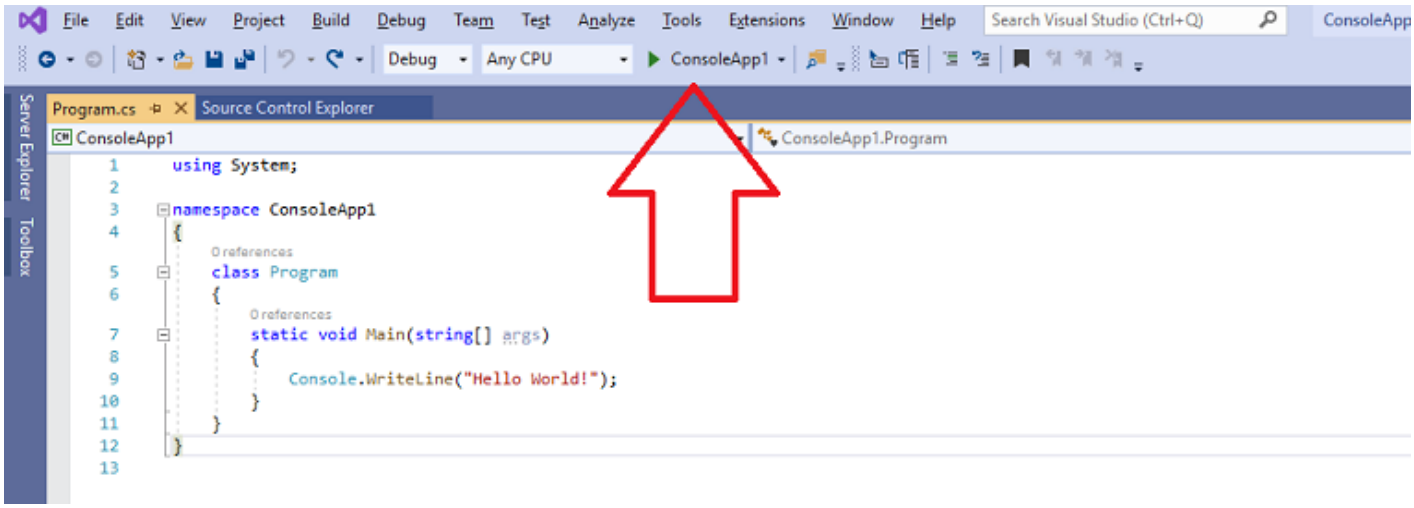
Otro punto muy importante, almenos para mi es el IDE o entorno de desarrollo, que es Visual Studio el cual no tiene rival alguno en ningún otro lenguaje de programación.

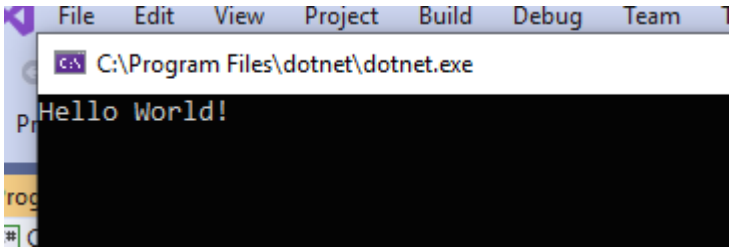
3 - Primera aplicación "Hello world"

Como podemos observar por defecto Visual Studio nos da un "hello world"

```
static void Main(string[] args)
{
    Console.WriteLine("Hello World!");
}
```

Si pulsamos en el botón de "Play" en nuestro visual studio nos arrancará la aplicación y nos imprimirá un "hello world"





Así de simple y sencillo es ejecutar nuestro primer "Hello world".

Antes de avanzar hay que indicar que cuando creamos un nuevo proyecto de consola siempre va a crearse una clase llamada Program y un método Main como los principales, estos se pueden cambiar, pero está estandarizado el no hacerlo.

4 - Estructura de un programa

Para crear una clase tenemos que :

pulsar botón secundario en el proyecto > añadir nuevo ítem > seleccionar clase (class)

Y nos creara algo muy similar a lo siguiente:

```
class Vehiculo{
}
```

Las clases contienen como elementos principales métodos y propiedades pero además pueden contener events, delegates, u otras clases dentro de otras, pero esto lo veremos más adelante.

4.1 - Propiedades

Las propiedades son atributos que contienen un valor a nivel de clase.

```
class Vehiculo{
    public decimal VelocidadMaxima { get; set; }
}
```

como vemos el ejemplo la creación de la propiedad continente:

- Modificador de acceso “Public”

- Tipo de dato “Decimal”
- Nombre de la propiedad “VelocidadMaxima”
- para dar valor un “set”
- para leer el valor un “get”
-

4.2 - Métodos

Los métodos son bloques de código los cuales tienen una finalidad que es realizar acciones.

El metodo mas comun en todas las clases es el constructor el cual se escribe de la siguiente manera:

```
class Vehiculo{
    public decimal VelocidadMaxima { get; set; }

    public Vehiculo(decimal velocidadMaxima){
        VelocidadMaxima = velocidadMaxima;
    }
}
```

Como vemos se llama igual que la clase. Además de esto los constructores se utilizan para dar un valor a las propiedades que contiene la clase.
Para dar un valor a las propiedades tenemos varias opciones:

- Asignar el valor dentro del constructor directamente
- Pasar el valor al constructor
- Utilizar otro método para asignar el valor
- Asignar el valor de forma directa una vez la clase esta instanciada

En nuestro ejemplo utilizaremos el de pasar al constructor el valor.

Para el ejemplo crearé otra variable que diga el consumo por kilómetro.

Esto nos permitirá crear otro método que nos devolverá el consumo total en cierta cantidad de kilómetros.

```
class Vehiculo{
    public decimal VelocidadMaxima { get; set; }
    public decimal ConsumoPorKilometro { get; set; }

    public Vehiculo(decimal velocidadMaxima, decimal consumoPorKilometro){
        VelocidadMaxima = velocidadMaxima;
        ConsumoPorKilometro = consumoPorKilometro;
    }

    public decimal ConsumoTotal(decimal kilometros){
        return ConsumoPorKilometro * kilometros;
    }
}
```

Como vemos en este caso es algo diferente ya que el método tiene después de la palabra “public” otro tipo de dato, el cual es el valor de retorno del método.

Importancia de los nombres de las variables

Es importante hacer un pequeño inciso aquí, ya que es un elemento que normalmente los programadores junior no caen en cuenta y es el nombre de las propiedades/variables y los métodos.

Es importante que tanto propiedades como métodos tengan nombres que se auto explican, y con esto que quiero decir, que si llamamos a una propiedad “VelocidadMaxima” todo el mundo cuando lee el código un tiempo después sabe que ahi en esa propiedad está almacenada la velocidad máxima. En cambio si llamamos a la variable “xmx” nadie va a entender que es lo que hace esa propiedad

4.3 - Comentarios

Los comentarios son bloques de texto añadidos a mano, y estos son añadidos para explicar qué es lo que pasa en el código, si alguna parte del mismo es muy compleja o liosa nos veremos en la obligación de poner algunos. pero como he indicado hace un momento, lo ideal sería que cada nombre de cada variable, propiedad y método nos indique claramente de lo que se trata.

Hay 3 formas de añadir comentarios:

- Comentarios de una línea utilizando “//”
- Bloque de comentario “/* texto */”
- Descripción de una función “\\\”
  - al pulsar 3 veces contrabarra si estamos en la parte superior de una función, el IDE nos generará automáticamente un bloque de comentarios el cual contiene:
    - <sumary> para explicar qué es lo que sucede en la función
    - <param> para cada uno de los parámetros de entrada
    - <returns> para indicar que es lo que devuelve

```
//comentario de línea

/*
    Comentario de bloque
*/

/// <summary>
/// Indica la cantidad de litros de gasolina gastados
/// </summary>
/// <param name="kilometros">Indicar kilometros realizados</param>
/// <returns></returns>
```

Como podemos observar, si tanto variables como métodos tienen un nombre adecuado, la utilización de comentarios no es necesaria dado que nos dan información que ya tenemos.

4.4 - Modificadores de acceso

Este apartado lo vamos a ver muy por encima ya que realizare otro video en el cual lo explicare todo con más detalle, simplemente enumerarlos y nombrar sus características.

Un modificador de acceso como su propio nombre indica, permite indicar la accesibilidad que tienen nuestros métodos o propiedades desde otros miembros o clases que los referencian.

Los principales son:

- public, sin restricciones
- internal, solo se puede acceder a ellos desde el mismo proyecto
- private, solo accesible desde la clase en la que se genera
- protected, desde la propia clase y desde las derivada.

por defecto en .NET el modificador de acceso es “internal”

4.5 -Declaración del namespace

Qué es el namespace o espacio de nombres?

el namespace es el lugar donde organizamos todas las clases y ficheros de nuestro proyecto. por detrás este usa un lenguaje XML el cual contiene qué clases están dentro de cada namespace y así permite usarla sin utilizar la directiva “using” la cual permite utilizar clases dentro de otros namespaces. (esto es muy utilizado en el día a día).

La utilización de namespace nos permite esquivar la ambigüedad con los nombres de las clases, ya que no puede haber dos clases con el mismo nombre en el mismo namespace, pero si en el mismo proyecto en diferentes namespaces.