

GVA

Grupo 2

Semana 2

Estándar de Implementación

Versión 1.0

Historia de revisiones

Autor	Descripción/Cambios desde versión anterior	Versión	Fecha
Martin Tambucho	Creación del documento.	1.0	30/08/2014
Christopher Quincke	Revisión del documento.	1.0	31/08/2014

Contenido

1. Convenciones Generales	3
2. Convenciones Específicas	3
2.0. Variables	3
2.1 Constantes	3
2.2 Indentación	4
3. Otros	
3.0. Convenciones de los comentarios	4
3.1 Convenciones del lenguaje	5
3.2 Convenciones de nomenclatura de nombres	5

1. Convenciones Generales

- Se usa la configuración predeterminada del editor de código (sangría automática, sangrías de cuatro caracteres, tabuladores guardados como espacios).
- El código será escrito en inglés, para facilitar los get, set y evitar tildes y "ñ".
- Utilizar nombres descriptivos y los más claros posibles.
- En general no utilizar abreviaciones ya que hacen que el código sea menos legible.
- Se escribe solamente una declaración por línea.
- Se escribe únicamente una instrucción por línea.
- Si la sangría no se aplica automáticamente a las líneas de continuación, se aplica una tabulación.
- Se agrega al menos una línea en blanco entre las definiciones de método y las definiciones de propiedad.

2. Convenciones Específicas

2.0. Variables

- Todas las variables deben estar en inglés.
- De ser posible las variables se inicializan en la misma línea que es declarada.
- Variables privadas comienzan con "_" y siguen con minúscula.

Ejemplo:

```
private string _example;
```

- Variables públicas comienzan con minúscula.

Ejemplo:

```
public string example;
```

2.1. Constantes

- Igual que las variables.

2.2. Indentación

- Se usan paréntesis para dotar a las cláusulas de un formato de expresión.
Ejemplo:

```
if ((var1 > var2) && (var1 > var3))
{
    // Realizar tal cosa.
}
```

- Dentro de lo posible una línea de código no debe superar los 80 caracteres.

3. Otros

3.0. Convenciones de los comentarios

- Los comentarios se realizan con "//".
- El comentario se sitúa en una línea independiente, no al final de una línea de código.
- El texto del comentario comienza con una letra mayúscula.
- El texto del comentario finaliza con un punto.
- Entre el delimitador del comentario (//) y el texto del comentario se inserta un espacio.
- No se crean bloques de asteriscos con formato alrededor de los comentarios.
Ejemplo:

```
// Esto es un ejemplo que comienza en este renglón y continúa
// en este otro.
```

- Todas las funciones deben tener su encabezado definiendo que hace la misma, qué y cuáles parámetros recibe y que retorna.
Ejemplo:

```
/// <summary>
/// Acá se dice que hace la función.
/// </summary>
/// <param name="param1">Acá se especifica el param1</param>
/// <param name="param2"> Acá se especifica el param2</param>
/// <returns>Acá se indica que retorna la función</returns>
```

```
public int ExampleFunction(int param1, int param2)
{
    // Código.
}
```

(La estructura de este tipo de encabezado se genera solo poniendo "//" en el renglón previo a la definición de la función).

3.1. Convenciones de lenguaje

- El operador + se usa para concatenar cadenas cortas (de String).
- En general, conviene usar int en lugar de tipos sin signo.
- Siempre se usa una instrucción try-catch para controlar la mayor parte de las excepciones.
- Para evitar excepciones y mejorar el rendimiento omitiendo comparaciones innecesarias, se usa && en lugar de & y || en lugar de | al realizar comparaciones.
- Se usan nombres descriptivos para las variables y funciones.

3.2. Convenciones de nomenclatura de nombres

- Variables privadas comienzan con "_" y siguen con minúscula.
`private string _example;`
- Variables públicas comienzan con minúscula.
`public string example;`
- En variables con más de una palabra se escriben todas juntas comenzando en mayúscula todas las palabras menos la primera.
`public string wordsExampleOne;`
`private string _wordsExampleTwo;`
- Nombres de funciones Camel (primera letra de cada palabra en mayúscula y palabras unidas).