

Recomendaciones de Mejores Prácticas de Codificación para lenguaje C#

**Bogotá D.C., 10 de Septiembre de 2018**

Información del Documento

|  |  |
| --- | --- |
| **Nombre del Proyecto:** | Documentación de la Arquitectura Empresarial del Instituto Colombiano Agropecuario –ICA– |
| **Preparado por:** | Randolfo Gerardo Patiño De La Cruz | **Versión del Documento No:** | 0.5 |
| **Título:** | **Recomendaciones de Mejores Prácticas de Codificación para Lenguaje: C#** | **Fecha de Versión del Documento:** | 2018-09-10 |
| **Revisado por:** |  | **Fecha de Revisión** | 2018-09-20 |

Registro de Cambios y Actualizaciones del Documento

| **Versión N°.** | **Fecha de Versión** | **Responsable del cambio** | **Descripción del Cambio** |
| --- | --- | --- | --- |
| 0.1 | 2018-02-06 | Randolfo Gerardo Patiño  | Versión inicial del documento |
| 0.2 | 2018-03-11 | Danny Ríos Tolosa | Eliminación de la sección “Equivalencias entre C# y .NET” |
| 0.3 | 2018-03-12 | Randolfo Gerardo Patiño  | Ajustes generales, para solicitud de aprobación. |
| 0.4 | 2018-05-29 | Randolfo Gerardo Patiño  | Ajustes al documento, por cambio de logo-símbolos institucionales |
| 0.5 | 2018-09-10 | Randolfo Gerardo Patiño  | Actualización documental. |
|  | *[AAAA/MM/DD]* |  |  |

TABLA DE CONTENIDO

Pág.

[1. Introducción 5](#_Toc508567825)

[2. Convenciones del documento 5](#_Toc508567826)

[3. Terminología y Definiciones 6](#_Toc508567827)

[4. Resumen de estándares y codificación 7](#_Toc508567828)

[4.1. Convenciones de Nombre 8](#_Toc508567829)

[4.2. Palabras reservadas 9](#_Toc508567830)

[4.3. Estilos de Codificación 9](#_Toc508567831)

[4.4. Uso del Lenguaje 10](#_Toc508567832)

[5. Convenciones de Nombres 11](#_Toc508567833)

[5.1. Recomendaciones Generales 11](#_Toc508567834)

[5.2. Uso de nombres y sintaxis 13](#_Toc508567835)

[6. Estilo de Codificación 15](#_Toc508567836)

[6.1. Formato 15](#_Toc508567837)

[REFERENCIA BIBLIOGRÁFICA 16](#_Toc508567838)

LISTA DE TABLAS

Pág.

[Tabla 1. URL de las fuentes de información de referencia 5](#_Toc508567839)

[Tabla 2. Convenciones de Color y Énfasis 5](#_Toc508567840)

[Tabla 3. Convenciones de Palabras clave 6](#_Toc508567841)

[Tabla 4. Terminología de: *Modificador de Acceso* 6](#_Toc508567842)

[Tabla 5. Terminología de: *Camel Case* 6](#_Toc508567843)

[Tabla 6. Terminología de: *Pascal Case* 7](#_Toc508567844)

[Tabla 7. Terminología de: *Common Type System (CTS)* 7](#_Toc508567845)

[Tabla 8. Terminología de: *Identificador* 7](#_Toc508567846)

[Tabla 9. Terminología de: *Número Mágico* 7](#_Toc508567847)

[Tabla 11. Convenciones de Nombres 9](#_Toc508567848)

[Tabla 12. Palabras reservadas 9](#_Toc508567849)

[Tabla 13. Estilos de Codificación 10](#_Toc508567850)

[Tabla 14. Uso del Lenguaje 10](#_Toc508567851)

[Tabla 13. Utilización de nombres y sintaxis 14](#_Toc508567852)

**RECOMENDACIONES DE MEJORES PRÁCTICAS DE CODIFICACIÓN PARA LENGUAJE C#**

# Introducción

En el presente documento se describen reglas y recomendaciones para el desarrollo de aplicaciones, usando los lenguajes de la plataforma Microsoft .Net: C#, y su objetivo principal es definir guías para enmarcar la consistencia de estilo y de formato para ayudar a los desarrolladores a evitar errores comunes. Así mismo establecer estándares de programación.

También se incluyen lineamientos y recomendaciones para el desarrollo de aplicaciones escalables y en capas.

De forma específica se cubren convenciones de: Nombres, estilos de codificación, uso del lenguaje, mejores prácticas para aplicaciones, etc. Todos estos temas son cubiertos detallando reglas y recomendaciones las cuales son aplicables para C#. Las fuentes de información utilizadas para el desarrollo de este documento son las siguientes:

|  |  |
| --- | --- |
| **Tema Fuente de la Información** | **URL** |
| Design Guidelines for ClassLibrary Developers | http://msdn2.microsoft.com/en-us/library/czefa0ke(VS.71).aspx |
| Guidelines and Best Practices | http://msdn2.microsoft.com/en-us/library/ms184412.aspx |
| Best Practices | http://msdn2.microsoft.com/en-us/library/ms184411(VS.80).aspx |

Tabla 1. URL de las fuentes de información de referencia

*Fuente: Estándares y Consideraciones Desarrollo .NET.*

# Convenciones del documento

Se refieren estándares para asegurar la claridad al impartir las reglas y guías. Se enuncian ciertas convenciones particulares, que son usadas para resaltar o enfatizar:

| ***Color y Énfasis*** |
| --- |
| **Azul** | Texto coloreado de azul indica una palabra clave de C# ode .NET |
| **Negrilla** | Texto con énfasis adicional para resaltarlo |

Tabla 2. Convenciones de Color y Énfasis

*Fuente: Estándares y Consideraciones Desarrollo .NET.*

| ***Palabras Clave*** |
| --- |
| **Siempre** | Enfatiza que la regla debe ser cumplir |
| **Nunca** | Enfatiza que la acción nunca se debe de ejecutar |
| **Evitar**  | Enfatiza que la acción debe de ser prevenida, pero algunas excepciones pueden existir |
| **Tratar** | Enfatiza que la regla debe de cumplirse siempre y cuando sea posible y sea adecuado |
| **Ejemplo** | Precede el texto usado para ilustrar la regla o recomendación |
| **Razón** | Explica el propósito detrás de la regla o recomendación |

Tabla 3. Convenciones de Palabras clave

*Fuente: Estándares y Consideraciones Desarrollo .NET.*

# Terminología y Definiciones

Se utilizará la siguiente terminología:

| ***Modificador de Acceso*** |
| --- |
|  | Las palabras clave de C# o de .NET que declaran el nivel de accesibilidad de código que se tiene para las clases, métodos y atributos.**Ejemplo**: [C#]private, public, internal, protected, staticAunque los modificadores de acceso varían, las clases y la mayoría de sus miembros usan por defecto private o Private. Excepciones para esto son las interfaces y enumerados los cuales por defecto son públicos.En la sección de breve resumen se muestra una tabla de equivalencias de modificadores de acceso entre .NET y C#. |

Tabla 4. Terminología de: *Modificador de Acceso*

*Fuente: Estándares y Consideraciones Desarrollo .NET.*

| ***Camel Case*** |
| --- |
|  | Forma de escribir una palabra donde su primera letra está en minúsculas, y la primera letra de las subsiguientes palabras en mayúsculas.**Ejemplo**:nombreVendedor, nombreEntidad, nombreProyectoEntidad |

Tabla 5. Terminología de: *Camel Case*

*Fuente: Estándares y Consideraciones Desarrollo .NET.*

| ***Pascal Case*** |
| --- |
|  | Forma de escribir una palabra donde la primera letra está en mayúscula, y la primera letra de las subsiguientes palabras igualmente en mayúsculas.**Ejemplo**:NombreVendedor, NombreMenu, NombreEstadoProyecto |

Tabla 6. Terminología de: *Pascal Case*

*Fuente: Estándares y Consideraciones Desarrollo .NET.*

| ***Common Type System (CTS)*** |
| --- |
|  | El sistema común de tipos (CTS) del .NET Framework define como los tipos son declarados, usados y manejados. Todos los tipos nativos de C# y de .NET están basados sobre el CTS para asegurar el soporte de la integración de los diferentes lenguajes del .NET Framework. |

Tabla 7. Terminología de: *Common Type System (CTS)*

*Fuente: Estándares y Consideraciones Desarrollo .NET.*

| ***Identificador*** |
| --- |
|  | El desarrollador define un nombre único para declarar un objeto o instancia de alguno. |

Tabla 8. Terminología de: *Identificador*

*Fuente: Estándares y Consideraciones Desarrollo .NET.*

| ***Número Mágico*** |
| --- |
|  | Cualquier número usado con una expresión (o inicialización de alguna variable) que no contiene en si un significado obvio o conocido. Este usualmente excluye los enteros 0 y 1 y cualquier otra equivalente numérico que se evalúe como cero. |

Tabla 9. Terminología de: *Número Mágico*

*Fuente: Estándares y Consideraciones Desarrollo .NET.*

# Resumen de estándares y codificación

Esta sección contiene cuadros y tablas describiendo a alto nivel el resumen de las recomendaciones y estándares de nombres, estilos de codificación y uso del lenguaje.

Igualmente se presenta una tabla de equivalencias entre los lenguajes .NET y C# para tener en cuenta cada una de las reglas

## Convenciones de Nombre

“c” = camelCase

“P” = PascalCase

“\_” = Prefijo con raya inferior

“X” = No aplicable

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Identificador** | **Public** | **Protected** | **Internal Friend** | **Private** | **Notas** |
| Archivo Proyecto | P | X | X | X | Coincidir con el Ensamblado y con el Espacio de Nombre (Namespace) |
| Archivo Fuente | P | X | X | X | Coincidir con la clase que contiene |
| Otros Archivos | P | X | X | X | Aplicar cuanto sea posible |
| Espacio de Nombre (namespace) | P | X | X | X | Coincidir parcialmente el Proyecto/Ensamblado |
| Clase o Estructura | P | P | P | P | Añadir sufijo de subclase |
| Interfase | P | P | P | P | Añadir prefijo de capital |
| Clase Genérica | P | P | P | P | Usar T o K como identificadores de tipo |
| Método | P | P | P | P | Usar un verbo o pareja verbo-objeto |
| Propiedad | P | P | P | P | No usar prefijos con Get o Set |
| Campo | P | P | P | [C#]: \_c[.NET]: c | Solo usar campos privados.**Nunca notación Húngara** |
| Constante | P | P | P | [C#]: \_c[.NET]: c | Solo usar campos privados |
| Campo Estático | P | P | P | [C#]: \_c[.NET]: c |  |
| Enumerador | P | P | P | P |  |
| Delegado | P | P | P | P |  |
| Evento  | P | P | P | P |  |
| Parámetro | X | X | X | c |  |
| Variables internas | X | X | X | c | Evitar usar nombres de un solo carácter o de enumerados |

Tabla 11. Convenciones de Nombres

*Fuente: Estándares y Consideraciones Desarrollo .NET.*

## Palabras reservadas

Las palabras reservadas en un lenguaje son utilizadas por el compilador con fines específicos. Las cuales no se deben utilizar al momento de definir los nombres de los elementos del programa tal como variables y procedimientos.

Las siguientes palabras no son reservadas, pero de igual manera no deben ser utilizadas para el nombramiento de variables, procedimientos y otros elementos

|  |
| --- |
| **Palabras Reservadas** |
| Aggregate | Distinct | sFalse | Preserve | Unicode |
| Ansi | Equals | IsTrue | Skip | Until |
| Assembly | Explicit | Join | Skip While | Where |
| Auto | From | Key (Visual Basic) | Strict | # ExternalSource |
| Binary | Group By | Mid | Take | # Region |
| Compare | Group Join | Off | Take While |  |
| Custom | Into | Order By | Text |  |

Tabla 12. Palabras reservadas

*Fuente: Estándares y Consideraciones Desarrollo .NET.*

## Estilos de Codificación

| **Código** | **Estilo** |
| --- | --- |
| Archivos Fuente | Un espacio de nombre por archivo y una clase por archivo |
| Llaves "{" | Una por línea. Siempre usar llaves cuando sea opcional |
| Indentación | Usar tabs de tamaño de 4 espacios en blanco |
| Comentarios | [C#]: Usar:// o /// pero no /\* … \*/ y no utilizar marcos de asteriscos. Utilizar /// para comentarios XML de las funciones y métodos. [.NET]: Usar el **‘** para todo tipo de comentario. |
| Variables | Una variable por declaración |

Tabla 13. Estilos de Codificación

*Fuente: Estándares y Consideraciones Desarrollo .NET.*

## Uso del Lenguaje

|  |  |
| --- | --- |
| **Código** | **Estilo** |
| Tipos Nativos de Datos | * Usar datos nativos de datos de C# y .NET versus los tipos del CTS del .NET.
* [C#] Usar int **NO** System.Int32
* [.NET] Usar Integer **NO** System.Int32
 |
| Propiedades | * Evitar utilizar prefijos Get o Set
 |
| MyBase/ base | * Usar solamente en constructores o en sobre escritura.
 |
| Condicionales | * Evitar evaluar condiciones booleanas contra true o false.
* No utilizar asignaciones en la condición.
* Evitar usar invocaciones a métodos en las condiciones.
 |
| Excepciones | * Usar throwe; cuando se necesite re-lanzar excepciones.
* Capturar solo cuando se vaya a manejar la excepción.
* Usar validaciones para evitar excepciones.
* Heredar siempre de la clase Exception..
 |
| Eventos | * Siempre chequear nulos antes de invocarlos.
 |
| Dispose() y Close() | * Siempre invocarlos si están implementados en la clase, declares cuando sean necesarios.
 |
| Finalizadores | * Evitarlos.

[C#] Usar los destructores de c#.Nunca cree el método Finalize() en C#.* [.NET] Implementar el método Finalize().
 |

Tabla 14. Uso del Lenguaje

*Fuente: Estándares y Consideraciones Desarrollo .NET.*

# Convenciones de Nombres

En esta sección se concentra en el trabajo de nombrar proyectos, archivos de código fuente, e identificadores incluyendo campos, variables, propiedades, métodos, parámetros, clases, interfases y espacios de nombres (Namespaces).

## Recomendaciones Generales

1. Siempre usar nombres con estilo de escritura CamelCase o PascalCase.
2. Evitar nombres totalmente en MAYÚSCULAS o en minúsculas. Nombres deuna sola palabra serán totalmente en minúsculas si se usa con el estilo de escritura CamelCase.
3. Nunca cree espacios de nombres (namespaces), clases, métodos, propiedades, campos o parámetros que varían solamente por su capitalización. (Mayúsculas)
4. Nunca se deben utilizar nombres que comiencen con caracteres numéricos.
5. Siempre se deben seleccionar nombres específicos y con significado en sí mismos.
6. Variables y propiedades deben describir la entidad que representan no el tipo o tamaño.
7. **Nunca usar notación Húngara**.

**Ejemplo:** strNombre o chrTipoRol

1. Evitar el uso de abreviaturas a menos que el nombre completo sea excesivo (25 caracteres).
2. Evitar el uso de underscore “\_” y guión “-“ en el nombramiento de los métodos.
3. Evitar abreviaturas que su longitud sea mayor a 5 caracteres.
4. Cualquier abreviatura debe ser totalmente conocida y aceptada por el equipo de desarrollo.
5. Usar mayúsculas en caso de abreviaciones de 2 letras, y estilo de escritura PascalCase para abreviaturas más largas.
6. Nunca usar palabras reservadas para nombres.
7. Evitar conflictos de nombres con los namespaces o tipos existentes en el Framework .NET.
8. Evitar añadir redundancia en prefijos o sufijos de los identificadores.

**Ejemplo:**

[C#] :

/// Mal!

public enum TiposRolEnum { … }

public class CPoliticasInternas { … }

public struct CoordenadaStruct { … }

[.NET] :

/// Mal!

Public Enum TiposRolEnum (TiposRol)

…

End Enum

Public Class CPoliticasInternas

…

End Class

Public Structure CoordenadaStruct

…

End Class

1. Nunca incluir el nombre de la clase a los nombres de las propiedades.

**Ejemplo**: Cliente.Nombre **NO** Cliente.NombreCliente

1. Tratar de utilizar los siguientes prefijos en las variables y propiedades booleanas, “Puede”, “Es” o “Tiene”.
2. Evitar la utilización de Boxing y UnBoxing, debido a que son procesos lentos

**Ejemplo:**

Dim obj As Object = NewPoint(1, 1)

Dim valor As Int32 = Convert.ToInt32(obj)

1. En el nombramiento de métodos de una clase, se deben mantener nombres consistentes para métodos con operaciones similares, por ejemplo: Definir un método GuardarArchivo si tiene un método CargarArchivo, definir un método Cerrar si tiene un método Abrir.
2. Los métodos no deben exceder en más de 50 líneas ejecutables, es preferible dividir los métodos en varios procedimientos. Esto debido a que el compilador es más eficiente compilando métodos menores a 32 bytes de código.
3. En el caso de tener métodos que tengan más de 5 parámetros, es preferible utilizar estructuras o una clase.
4. Se debe evitar el código quemado o Hard-Code, mediante la utilización de un archivo de parámetros (De tipo property o XML)
5. Cuando se defina el namespace raíz para un proyecto, utilice el nombre de la compañía, seguido por el nombre del proyecto y opcionalmente la característica de diseño, par evitar que existan 2 namespaces publicados con el mismo nombre, con estilo de escritura PascalCase.

**Ejemplo**: Empresa.SistemaOnline o

Empresa.SistemaFacturacion o

DNP.Datos.Formulación o

DNP.Logica.Ejecución o

DNP.Vista.Administración o

## Uso de nombres y sintaxis

| **Identificador** | **Convención de nombramiento** |
| --- | --- |
| Archivo de Proyecto | * Utilizar estilo de Escritura Pascal
* Siempre coincidir el nombre del ensamblado y el espacio de nombres raíz.

**Ejemplo:**Empresa.SistemaOnline.csproj -> Empresa.SistemaOnline.dll -> namespace: Empresa.SistemaOnline |
| **Archivo Fuente** | * Utilizar estilo de Escritura Pascal
* Siempre coincidir el nombre de la clase y el nombre del archivo.
* Evitar incluir más de una clase, enumerador (global) o delegado (global) por archivo.
* Usar un nombre descriptivo para el archivo que contenga múltiples clases, enumeradores o delegados.

**Ejemplo**: [C#] MiClase.cs => public class MiClase[.NET] MiClase.vb => Public Class MiClase |
| Archivo de Recurso | * Tratar de utilizar estilo de escritura Pascal.
* Utilizar un nombre que describa la información que contenga el archivo.
 |
| Namespaces | * Estilo de Escritura Pascal
* Tratar de coincidir el nombre ensamblado/proyecto.

**Ejemplo**:[C#] namespace Empresa.SistemaWeb[.NET] Namespace Empresa.SistemaWeb |
| Clase o Estructura | * Utilizar estilo de Escritura Pascal
* Usar un sustantivo o frase de sustantivos para el nombre dela clase.
* Agregar un sufijo apropiado cuando se nombre sub clases.

**Ejemplo**:[C#] private classMiClase{ … }internal classCoordenadaAttribute: Attribute{ … }public class ClienteCollection : CollectionBase{ … }public class ClienteEventArgs : EventArgs{ … }private struct PropiedadesAplicacion{ … }[.NET]Private Class MiClase…End ClassFriend Class CoordenadaAttribute Inherits Attribute …End ClassPublic Class ClienteCollection Inherits CollectionBase …End ClassPublic Class ClienteEventArgs Inherits EventArgs …End ClassPrivate Structure PropiedadesAplicacion …End Structure |
| Interfase |  |
| Método |  |
| Propiedad |  |
| Campo (Público, Protegido o interno) |  |
| Campo (Privado) |  |
| Constante o campo estático |  |
| Enumerador |  |
| Delegado o Evento |  |
| Variable (Interna) |  |
| Parámetro |  |

Tabla 13. Utilización de nombres y sintaxis

*Fuente: Estándares y Consideraciones Desarrollo .NET.*

# Estilo de Codificación

Se describe el camino preferido para implementar código fuente en C# y .NET, para así crear un código legible, claro y consistente que sea fácil de comprender y mantener.

## Formato

1. Nunca declarar más de un espacio de nombres por archivo.
2. Evitar colocar múltiples clases en un mismo archivo.
3. [C#] Siempre colocar llaves (“{“ y “}”) en líneas separadas.
4. [C#] Siempre usar llaves (“{“ y “}”) en sentencias condicionales.
5. Siempre usar Tab e indentación de tamaño 4.
6. Declarar cada variable independientemente – nunca en la misma sentencia.
7. Colocar la sentencia “using” o “Imports” en la parte superior del archivo. El grupo de los espacios de nombres de .NET colocarle por encima de los espacios de nombres particulares y todos en ordenados alfabéticamente.
8. Agrupar la implementación interna de las clases por el tipo de miembros enel siguiente orden:
9. Campos
10. Propiedades
11. Constructores y Destructores
12. Métodos
13. Enumeradores, Estructuras o Clases internas
14. Ordenar las declaraciones a partir de los modificadores de acceso y visibilidad:
15. Privados
16. Internos
17. Protegidos
18. Público
19. Utilizar #región o #Region " ” para agrupar los tipos de miembros de la clase.
20. Añadir el nombre de la carpeta al espacio de nombres para archivos en el interior de estos.
21. Recursivamente indentar todos lo bloques de código en medio de llaves.
22. [C#] Evitar declarar múltiples atributos en una misma línea. En lugar de ello colocar cada atributo en una sentencia separada.

# REFERENCIA BIBLIOGRÁFICA

* Estándares y Consideraciones de desarrollo para proyectos sobre plataformas Microsoft.NET. Por: Eduardo Rivera Alva. Versión 0.8. Tomado el 06 de febrero de 2018. [*https://es.scribd.com/document/209660/Estandares-y-Consideraciones-de-Desarrollo-para-Microsoft-NET*](https://es.scribd.com/document/209660/Estandares-y-Consideraciones-de-Desarrollo-para-Microsoft-NET).
* Design Guidelines for Class Library Developers, [*http://msdn2.microsoft.com/es-*](http://msdn2.microsoft.com/es-)[*es/library/ czefa0ke(VS.71).aspx*](http://msdn2.microsoft.com/es-es/library/czefa0ke%28VS.71%29.aspx).
* Guidelines and Best Practices. [*http://msdn2.microsoft.com/es-es/library/ms184412.aspx*](http://msdn2.microsoft.com/es-es/library/ms184412.aspx).
* Best Practices. [*http://msdn2.microsoft.com/es-es/library/ms184411(VS.80).aspx*](http://msdn2.microsoft.com/es-es/library/ms184411%28VS.80%29.aspx).
* Instant SQL Formatter. [*http://www.wangz.net/cgi-bin/pp/gsqlparser/sqlpp/sqlformat.tpl*](http://www.wangz.net/cgi-bin/pp/gsqlparser/sqlpp/sqlformat.tpl).
* Buenas Prácticas de Codificación, Estándar de codificación y documentación de codigo fuente para aplicativos informáticos, Guia rápida del estándar de codificación y documentación de código fuente para aplicativos informáticos. [*http://larebeca/GestióndeTIC/LineamientosTécnicos/SistemasdeInformación/Lineamient*](http://larebeca/Gesti%C3%B3ndeTIC/LineamientosT%C3%A9cnicos/SistemasdeInformaci%C3%B3n/Lineamient)[*osTécnicosSI/tabid/724/Default.aspx*](http://larebeca/Gesti%C3%B3ndeTIC/LineamientosT%C3%A9cnicos/SistemasdeInformaci%C3%B3n/LineamientosT%C3%A9cnicosSI/tabid/724/Default.aspx).