



ESTANDARES DE DESARROLLO EN C# .Net  
Versión 1.0

T.I.C.

	<b>Estándares de Desarrollo para Microsoft .NET C#</b>		
	<b>Dirección:</b> Tecnología de la Información		
	<b>Versión:</b> 1.0		<b>Mayo 2019</b>


### REVISIONES

Fecha	Versión	Descripción	Autor
2/Mayo/2019	1.0	Versión original	Samuel Arone

	<b>Estándares de Desarrollo para Microsoft .NET C#</b>	
	<b>Dirección:</b> Tecnología de la Información	
	<b>Versión:</b> 1.0	<b>Mayo 2019</b>

## INDICE

<b>OBJETIVOS</b>	<b>4</b>
<b>1. ITEMS CONSIDERADOS PARA ESTANDARIZACION EN C#</b>	<b>4</b>
1.1 Convenciones y Estándares de Nombres	4
1.2 Declaración de variables y propiedades de clase	5
1.3 Definición de tipos de en las variables	6
1.4 Clases	7
1.5 Interfaces	7
<b>2. BACKEND</b>	<b>8</b>
2.1 Areas	8
2.2 Helper generico de mantenimiento	8
2.3 Acciones del Controlador	9
2.4 Filtro de Seguridad	10
2.5 Tipos de Métodos para las acciones	10
2.6 Variables de Session	11
2.7 Controlador	11
2.8 Modelo	11
2.9 Envió de Correo	11
<b>3. FRONT END</b>	<b>11</b>
3.1 Hojas de Estilo en Cascada	12
3.2 Framework Bootstrap CSS	12
3.3 Diseño de Inspinia	14

	<b>Estándares de Desarrollo para Microsoft .NET C#</b>	
	<b>Dirección:</b> Tecnología de la Información	
	<b>Versión:</b> 1.0	<b>Mayo 2019</b>

## ESTANDARES EN MS .NET C#

### OBJETIVOS

Implementar el uso de estándares con las mejores prácticas establecidas por los desarrolladores de la plataforma Microsoft .Net "C#" de la organización.

### 1. ITEMS CONSIDERADOS PARA ESTANDARIZACION EN C#


#### 1.1 Convenciones y Estándares de Nombres

<b>Comentarios</b>	Listado convenciones para todos los tipos de Objetos en C#.
<b>Razón</b>	Para facilitar la lectura de código en Visual Studio.

Object Name	Notation	Length	Plural	Prefix	Suffix	Abbreviation	Char Mask
Class name	PascalCase	128	No	No	Yes	No	[A-z][0-9]
Constructor name	PascalCase	128	No	No	Yes	No	[A-z][0-9]
Method name	PascalCase	128	<b>Yes</b>	No	No	No	[A-z][0-9]
Method arguments	<b>camelCase</b>	128	<b>Yes</b>	No	No	<b>Yes</b>	[A-z][0-9]
Local variables	<b>camelCase</b>	50	<b>Yes</b>	No	No	<b>Yes</b>	[A-z][0-9]
Constants name	PascalCase	50	No	No	No	No	[A-z][0-9]
Field name	<b>camelCase</b>	50	<b>Yes</b>	No	No	<b>Yes</b>	[A-z][0-9]
Properties name	PascalCase	50	<b>Yes</b>	No	No	<b>Yes</b>	[A-z][0-9]
Delegate name	PascalCase	128	No	No	Yes	<b>Yes</b>	[A-z]
Enum type name	PascalCase	128	<b>Yes</b>	No	No	No	[A-z]

	<b>Notación PascalCase</b>
<b>Comentarios</b>	El primer carácter de todas las palabras se escribe en Mayúsculas y los otros caracteres en minúsculas.

```
public class Mensaje
{
    public string Texto {get; set;}
    public void AgregarMensaje()
    {
        //...
    }
}
```

	Estándares de Desarrollo para Microsoft .NET C#	
	Dirección: Tecnología de la Información	
	Versión: 1.0	Mayo 2019

```
public Mensaje(){
    //..
}
}
```

	Notación CamelCase
Comentarios	El primer carácter de todas las palabras se escribe en Mayúsculas y los otros caracteres en minúsculas.


```
string codigoUsuario = "sarone";
Usuario oUsuario = new Usuario();
List<Categorias> listaCategorias = new List<Categorias>();
```

## 1.2 Declaración de variables y propiedades de clase

Comentarios	Se deben de declarar todas las propiedades y variables privadas de una clase en su parte superior.
Razón	Es una práctica que evita tener que estar buscando la definición de dichas propiedades y variables.

```
public class Usuario
{
    // Propiedades
    public string Codigo {get; set;}
    public string Correo {get; set;}
    public bool EsHabilitado {get; set;}
    public int Edad {get; set;}

    // Constructor
    Usuario()
    {
        // ...
    }
}
```

	<b>Estándares de Desarrollo para Microsoft .NET C#</b>	
	<b>Dirección:</b> Tecnología de la Información	
	<b>Versión:</b> 1.0	<b>Mayo 2019</b>

### 1.3 Definición de tipos de en las variables

<b>Comentarios</b>	Se debe usar los nombres de tipos predefinidos “primitivos” en lugar de los nombres de los tipos de sistema como Int16, Int32, String, Boolean, etc.
<b>Razón</b>	Los tipos primitivos tienen mejor performance y es mucho más natural de leer.


```
// Correcto
string codigo;
int correo;
bool esHabilitado;

// Evitar
String codigo;
Int32 correo;
Boolean esHabilitado;
```

<b>Comentarios</b>	Se debe usar el tipo implícito “var”. Excepto para los tipos primitivos (string, int, etc)
<b>Razón</b>	Elimina ruido, sobre todo cuando se trata de tipos genéricos complejos. El tipo es fácilmente detectable con los tooltips de Visual Studio.

```
// var
var file = File.Create(path);
var customers = new Dictionary();

// Primitivos
int index =100;
string timeSheet;
bool isCompleted;
```

	<b>Estándares de Desarrollo para Microsoft .NET C#</b>	
	<b>Dirección:</b> Tecnología de la Información	
	<b>Versión:</b> 1.0	<b>Mayo 2019</b>

## 1.4 Clases

<b>Comentarios</b>	Utilizar Substantivos para nombrar las clases.
--------------------	--

```
public class Empleado
{
}

public class Categoria
{
}

public class Hilo
{
}
```


## 1.5 Interfaces

<b>Comentarios</b>	Utilizar el prefijo "I" y nombres o adjetivos para nombrarlas.
<b>Razón</b>	Es consistente con .NET Framework y fáciles de recordar.

```
public class IHelper
{
}

public class ICotizacion
{
}

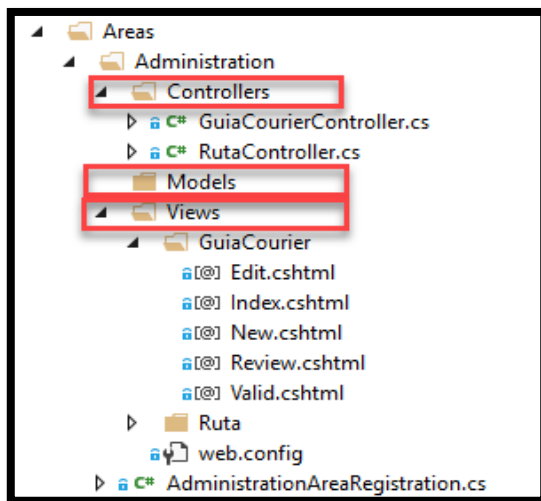
public class IHilo
{
}
```

	<b>Estándares de Desarrollo para Microsoft .NET C#</b>	
	<b>Dirección:</b> Tecnología de la Información	
	<b>Versión:</b> 1.0	<b>Mayo 2019</b>

## 2 BACKEND

### 2.1 Areas

<b>Comentarios</b>	Utilizar el modelo MVC con áreas es una forma de organizar nuestras aplicaciones, cada área tendrá su conjunto de controladores, modelos y vistas, de forma que tenemos una estructura de archivos ordenados, un ejemplo de ellos es tener áreas separadas por modulo.
<b>Ejemplo</b>	Administración, Contabilidad, Comercial, Desarrollo Textil.



### 2.2 Helper generico de mantenimiento

<b>Comentarios</b>	Utilizar la clase genérica "bLMantenimiento" para cualquier operación o transacción a la base de datos.
--------------------	---


```

public string GetData_Mensajes()
{
    bLMantenimiento oMantenimiento = new bLMantenimiento();
    string parametroJSON = _.Get("par");
    string data = oMantenimiento.get_Data("getMensaje", parametroJSON, false, Util.ERP);
    return data;
}

public string SaveData_Mensaje()
{

```



	Estándares de Desarrollo para Microsoft .NET C#	
	Dirección: Tecnología de la Información	
	Versión: 1.0	Mayo 2019

```

    bLMantenimiento oMantenimiento = new bLMantenimiento();
    string parametroJSON = _.Post("par");
    int id = oMantenimiento.save_Rows_Out("usp_Insert_Mensaje", parametroJSON, Util.ERP);
    string data=string.Empty;
    string mensaje = _.Mensaje("new", id > 0, data, id);
    return mensaje;
}

```

## 2.3 Acciones del Controlador

Comentarios	Se debe crear acciones del controlador con prefijos de nombres establecidos.
Ejemplo	GetData_[Titulo] GetExcel_[Titulo] GetPdf_[Titulo]

```

// Obtener data de tipo "string"
public string GetData_Empleado(){
}


public string GetData_Auditorias(){
}

public string GetData_PackingListxId(){
}

// Obtener data de tipo "FileContentResult" para archivos de tipo Excel, PDF, Texto, Imagen, etc.
public FileContentResult GetExcel_Empleado(){
}

public FileContentResult GetPdf_PackingList(){
}

```

	Estándares de Desarrollo para Microsoft .NET C#	
	Dirección: Tecnología de la Información	
	Versión: 1.0	Mayo 2019

## 2.4 Filtro de Seguridad

Comentarios	Utilizar el filtro de seguridad [AccessSecurity] para todas las acciones del controlador, esto ayuda a tener una seguridad para cada página (vista), solo podrán acceder a estos recursos los usuarios con los permisos autorizados

```
[AccessSecurity]
public ActionResult PackingListAdd()
{
    return View();
}
```

## 2.5 Tipos de Métodos para las acciones

Comentarios	Agregar el tipo de método para cada acción del controlador [HttpGet] y [HttpPost]
-------------	---

HttpGet	Utilizar cuando se obtiene un dato o alguna información de la BD o descarga de un archivo del servidor.
---------	---

```
[HttpGet]
public string GetData_Mensajes()
{
    bLMantenimiento oMantenimiento = new bLMantenimiento();
    string parametroJSON = _.Get("par");
    string data = oMantenimiento.get_Data("getMensaje", parametroJSON, false, Util.ERP);
    return data;
}
```

HttpPost	Utilizar cuando se envíe un formulario al controlador para insertar, editar en la Base de datos o subir un archivo al servidor
----------	--

```
[HttpPost]
public string SaveData_Mensaje()
{
    bLMantenimiento oMantenimiento = new bLMantenimiento();
    string parametroJSON = _.Post("par");
    int id = oMantenimiento.save_Rows_Out("usp_Insert_Mensaje", parametroJSON, Util.ERP);
    string data=string.Empty;
    string mensaje = _.Mensaje("new", id > 0, data, id);
    return mensaje;
}
```

}

## 2.6 Variables de Session

Comentarios	Tratar de no usar variables de session, si se llega a utilizar solo se debe almacenar poca información ya que consume mucha memoria del servidor dependiendo del número de usuarios.
-------------	--

## 2.7 Controlador

Comentarios	El Controlador solo será una capa de pasarela de lo posible no utilizar reglas de negocio. Utilizar pocas líneas en esta capa, como máximo 20 líneas para que sea legible el código.
-------------	---

## 2.8 Modelo

Comentarios	El Modelo solo será utilizado si las líneas de la acción del controlador supera las 20 líneas de código o va a tener lógica de negocio, como por ejemplo creación de un documento PDF, Excel y otros.
-------------	---

## 2.9 Envío de Correo

Comentarios	Para enviar correo se utilizará el método "sendMailBandeja" de la clase "blMail" esta implementa de manera automática todo el flujo de envió.
-------------	---


```

beMailSQL obeMail = new beMailSQL();
obeMail.body = "Se adjunta el reporte tecnico";
obeMail.codigo_usuario = "erp";
obeMail.copia = "sarone@wts.com.pe";
obeMail.correo_usuario = "erp@wts.com.pe";
obeMail.to_address = "laboratorio@wts.com.pe";
obeMail.subject = "WTS Laboratory - Final Report";
obeMail.file_attachments = name_file;

blMail oblMail = new blMail();
oblMail.sendMailBandeja(obeMail);

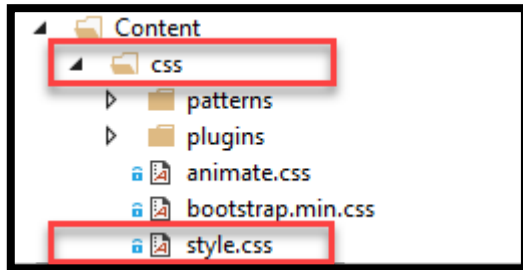
```

## 3. FRONT END

	<b>Estándares de Desarrollo para Microsoft .NET C#</b>	
	<b>Dirección:</b> Tecnología de la Información	
	<b>Versión:</b> 1.0	<b>Mayo 2019</b>

### 3.1 Hojas de Estilo en Cascada

<b>Comentarios</b>	Utilizar hojas de estilo css para controlar el look and feel de las páginas, no incrustar estilos directamente en el código HTML
--------------------	--



### 3.2 Framework Bootstrap CSS

<b>Comentarios</b>	Utilizar el framework bootstrap para facilitar el proceso de maquetación de interfaces y obtener una experiencia agradable y atractiva del usuario
--------------------	--

```

<!-- Encabezado -->
<div class="row wrapper white-bg page-heading">
  <div id="content_header" class="ibox float-e-margins">
    <div class="col-sm-6">
      <h2 class="font-bold">Sticker Orvis</h2>
    </div>
    <div class="col-sm-6 text-right position_right">
      <button id="btn_New" type="button" class="btn btn-default">
        <span class="fa fa-plus"></span>
        Nuevo
      </button>
    </div>
  </div>
</div>
<!-- Detalle -->
<div class="row wrapper-content gray-bg animated">
  <div id="content_form" class="ibox">
    <div class="ibox-title">
      <div class="form-horizontal row ">
        <div class="col-sm-6">
          <h3 id="_title" class="text-navy bold">Lista Sticker</h3>
        </div>
      </div>
    </div>
    <div id="div_Details" class="ibox-content">
      <div class="table-responsive">
        <input type="text" class="form-control input-sm m-b-xs" id="filter"

```

```
placeholder="Search">
<table class="footable table table-stripped table-bordered "
data-page-size="10" data-filter=#filter id="tbl_CartonLabel">
<thead>
<tr>
<th class='col-sm-1' data-sort-ignore="true"></th>
<th class='col-sm-1' data-sort-ignore="true">Sticker</th>
<th class='col-sm-1' data-sort-ignore="true"># Registro</th>
<th class='col-sm-1' data-sort-ignore="true">PO</th>
<th class='col-sm-1' data-sort-ignore="true">Usuario</th>
<th class='col-sm-1' data-sort-ignore="true">Fecha</th>
</tr>
</thead>
<tbody></tbody>
<tfoot>
<tr>
<td colspan="7">
<ul class="pagination pull-right"></ul>
</td>
</tr>
</tfoot>
</table>
</div>
</div>
</div>
</div>
```

### 3.3 Diseño de Inspinia

<b>Comentarios</b>	Utilizar los diseños de <b>Inspinia</b> como plantilla principal creada específicamente para el <b>framework bootstrap</b> , en ello se encuentra modelos y componentes para reutilizar en el ERP
--------------------	---

