Sistema de “MercaditoYa - Desarrollo de un sitio web para la gestión integral de inventario y pedidos del Minimarket Avan”
Documento de Estándares de Programación
Versión 4.0

Historia de Revisión

Ítem	Fecha	Versión	Descripción	Equipo
1	22/09/2025	1.0	Versión Inicial.	Emerson Castillo
2	23/09/2025	2.0	Descripción de las variables	Edinson Luna
3	23/09/2025	3.0	Descripción de las variables de Tipo Arreglo	Brandon Mena
4	23/09/2025	4.0	Se definieron los Controles	Manuel Dongo

Tabla de Contenidos

Estándares de Programación

1. OBJETIVO

Establecer normativas para la implementación del código fuente del proyecto, abarcando aspectos como variables, controles, clases, métodos y todos los elementos involucrados en el código.

Mejorar y homogeneizar, mediante las reglas propuestas, el estilo de programación de cada desarrollador.

- Los nombres de las variables se diseñarán de forma mnemotécnica, lo que permitirá identificar el tipo de dato asociado simplemente observando el nombre.

- Fomentar el uso de nombres sugestivos para las variables y funciones, de modo que su uso y propósito sean fácilmente comprensibles al examinar el nombre correspondiente.

- La elección de nombres para variables o funciones se llevará a cabo de manera automática y mecánica, siguiendo las reglas establecidas en nuestro estándar interno.

- Permitir el uso de herramientas automáticas de verificación de nomenclaturas.

Por consiguiente, se adoptarán estos patrones para lograr una comprensión clara del código y facilitar su mantenimiento.

2. DECLARACIÓN DE VARIABLES

Se propone que la declaración de las variables, se ajusten al motivo para la que se requieran. El mnemotécnico definido se establece tomando en consideración principalmente lo siguiente:

- Longitud máxima: 20 caracteres (ejemplo: nombreCompleto, repartidorId).
- Formato: camelCase (primera palabra en minúscula, las siguientes con inicial en mayúscula).
- Contexto: deben reflejar el dominio del negocio (ej.: productos, pedidos, usuarios)

A medida que aumenta el tamaño del proyecto, también aumenta la utilidad de reconocer rápidamente el alcance de las variables. Esto se consigue al escribir un prefijo de alcance de una letra delante del tipo de prefijo propio, sin aumentar demasiado la longitud del nombre de las variables.

Alcance	Prefijo	Ejemplo
Global	G	gblUsuarioActual
Nivel de la clase	*M*	mblEstadoPedido
Local del procedimiento / método	Ninguno	totalPedido
Público	P	pListaProductos
Privado	Pr	prIdSesion

- El tipo de dato al que pertenece la variable.

Por lo tanto la estructura de la variable es como sigue:

Estructura	Descripción de la Variable
LONGITUD. MAX.	□ 1 □□ 16 □
FORMATO	*Minúscula la primera parte y luego la segunda con Mayúsculas*
EJEMPLO	numCuenta

Siendo el nombre que identifica a la variable: **numCuenta**

2.1 Descripción de la Variable.

Nombre que se le asignará a la variable para que se le identifique y deberá de estar asociada al motivo para la cual se le declara.

Ejemplos:

idPedido: Identificador único del pedido.

estadoPedido: Estado del pedido (pendiente, confirmado, en camino, entregado).

fechaEntrega: Fecha programada de entrega.

esDelivery: Booleano que indica si es delivery o retiro en tienda.

repartidorId: Identificador del repartidor asignado.

productosCarrito: Lista de productos seleccionados en el carrito.

stockDisponible: Cantidad disponible en inventario.

2.2 Variables de Tipo Arreglo

En el caso de las definiciones de arreglos de elementos se declarará la variable con el prefijo de “lista”, el cual nos dará entender que se trata de una variable del tipo arreglo la cual contendrá de cero a mas datos, según el tamaño declarado.

Ejemplos:

listaProductos: Colección de productos disponibles.

listaPedidos: Pedidos registrados en el sistema.

listaCategorias: Categorías de productos.

listaUsuarios: Usuarios del sistema.

listaRepartidores: Repartidores disponibles.

3. Definición de Controles

Para poder determinar el nombre de un control dentro de cualquier aplicación de tipo visual, se procede a identificar el tipo al cual pertenece y la función que cumple dentro de la aplicación.

3.1 Tipo de datos

Tipo de variable	Mnemónico	Descripción
Byte	by	Entero de 8 bits sin signo.
Integer	in	Entero de 32 bits con signo.
Char	ch	Un carácter UNICODE de 16 bits
String	st	Cadena de caracteres
Date	dt	Formato de fecha/hora
Boolean	bl	Valor lógico: verdadero y falso
Float	fl	Coma flotantes, 11-12 dígitos significativos.
Double	db	Coma flotante, 64 bits (15-16 dígitos significativos)
Object	ob	Objeto genérico

3.2 Prefijo para el Control

El prefijo del control será determinado mediante tres caracteres que estarán conformados por las consonantes más representativas del control, es así, por ejemplo; el control Button, estará asociado al prefijo btn.

3.3 Nombre descriptivo del Control

Formado por la descripción de la función que lleva a cabo el control, esta debe ser descrita en forma específica y clara.

Tipo de control	Prefijo	Ejemplo
Label	lbl	lblNombre
TextBox	txt	txtApellido
Button	btn	btnLogin
RadioButton	rdo	rdoSeleccion
CheckBox	chk	chkRuta1
DropDownList	cmb	cmbDocumentos

3.4 Declaración de variables, atributos y objetos

1. Se debe declarar una variable por línea.

Título	Descripción
Sintaxis	[TipoVariable] [Nombre de la Variable]
Descripción	Todas las variables o atributo tendrán una longitud máxima de 30 caracteres. El nombre de la variable puede incluir más de un sustantivo los cuales se

	escribirán juntos. Si se tuvieran variables que puedan tomar nombres iguales, se le agregará un número asociado (si está dentro de un mismo método será correlativo).
Observaciones	En la declaración de variables o atributos no se deberá utilizar caracteres como: ● Letra Ñ o ñ. ● Caracteres especiales ¡, ^, #, $, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,]. ● Caracteres tildados: á, é, í, ó, ú.
Ejemplo	Public String nombre Indica una variable o atributo que guardará un nombre.

3.5 Declaración de clases

Título	Descripción
Sintaxis	[Tipo] Class [Nombre de Clase]
Descripción	El nombre de las clases tendrá una longitud máxima de 30 caracteres y las primeras letras de todas las palabras estarán en mayúsculas. Tipo se refiere a si la clase será: Private, Public o Protected.
Observaciones	En la declaración de clases no se deberá utilizar caracteres como: ● Letra Ñ o ñ. ● Caracteres especiales ¡, ^, #, $, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,]. ● Caracteres tildados: á, é, í, ó, ú.
Ejemplo	Private Class Empleado Indica una clase Empleado

3.6 Declaración de métodos

Título	Descripción
Sintaxis	nombreProcedim[(ListaParámetros)]
Descripción	El nombre del método constará hasta de 25 caracteres. La primera letra de la primera palabra del nombre será escrita en minúscula y las siguientes palabras empezarán con letra mayúscula.
Observaciones	En la declaración de métodos no se deberá utilizar caracteres como: ● Letra Ñ o ñ. ● Caracteres especiales ¡, ^, #, $, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,], _. ● Caracteres tildados: á, é, í, ó, ú.
Ejemplo	Protected calcularSueldo(String empleado) Indica un método calcularSueldo que recibe una variable por valor de tipo string al ámbito de la clase

3.7 Declaración de funciones

Título	Descripción
Sintaxis	[TipoDato] nombreFuncion[(ListaParámetros)]
Descripción	El nombre del objeto constará hasta de 25 caracteres, no es necesario colocar un nombre que indique la clase a la cual pertenece. La primera letra de la primera palabra del nombre será escrita en mayúsculas El tipo de dato de retorno se coloca al final y será obligatorio colocarlo.
Observaciones	En la declaración de objetos no se deberá utilizar caracteres como: ● Letra Ñ o ñ. ● Caracteres especiales ¡, ^, #, $, %, &, /, (,), ¿, ‘, +, -, *, {, }, [,], _. ● Caracteres tildados: á, é, í, ó, ú.
Ejemplo	Public int sumar(int A, int B) Indica una función que suma dos variables enteras

3.8 Control de versiones de código fuente

Cada modificación realizada será guardada de la forma:

Título	Descripción
Formato	[NOMBRE DOCUMENTO][_][FECHA][_][HORA] donde y la fecha estará en formato yyyymmdd y la hora en formato HHMM.
Descripción	Se generarán archivos con las siguientes extensiones:.zip o .rar. Por ejemplo: WSTENNIS_20070421_2056.zip

3.9 Controles ADO.NET

Objetos de ADO.NET Aunque hay miles de objetos disponibles como parte de .NET, es probable que se use ADO.NET como parte de las aplicaciones, por lo tanto algunos estándares para nombrar los objetos de ADO.NET más comunes. A continuación, se listan los prefijos que se utiliza:

Componente	Prefijo
DataSet	Ds
DataTable	Dt
DataView	Dv
DataRow	Drw
Connection*	Cnn
Command*	Cmd

DataAdapter*	Da
CommandBuilder*	Bld
DataReader*	Dr

Ejemplos: de declaración de los objetos ADO.net

- drEmps As New SqlDataReader()
- drCust As New SqlDataReader()
- dsEmps As DataSet
- dsCust As DataSet

4. Clases.

El nombre de las clases debe ser auto descriptivo de manera que no se requiera, en lo posible, entrar al código de la función para saber qué es lo que realiza.

El estándar para nombres de clases es usar iniciar con las siglas **cls,** la cual debe estar escrita en minúscula seguido del nombre que identifica la clase, la primera letra del nombre debe iniciar con mayúscula

- Ejemplos: clsCuenta, clsEmpleado, clsFactura

Nota:
- No se hará uso de los caracteres: Espacio en blanco " ", Caracter de subrayado "_".

5. Métodos, Procedimientos y Funciones definidos por el Usuario.

El nombre de las funciones y procedimientos debe ser auto descriptivo de manera que no se requiera, en lo posible, entrar al código de la función para saber qué es lo que realiza.
verbo-Sustantivo

El estándar para nombres de procedimiento es usar un Verbo que describa la acción realizada seguida por un sustantivo (objeto sobre el cual actúa). Se recomienda:

- Usar un nombre que represente una acción y un objeto. El nombre del procedimiento debe indicar qué hace el procedimiento a... o qué hace el procedimiento con....

- El verbo debe estar en infinitivo.

- Ser consistente en el orden de las palabras. Si se va a usar ***verboNombre***, siempre usar ***verboNombre***.

- Ser consistente en los verbos y sustantivos usados. Por ejemplo, si tiene un procedimiento ***asignarNombre***, en vez de ***colocarNombre***.

- Para la acción **modificar cuentas del cliente** se define:
  modificar Cuenta
  Verbo: modificar
  Sustantivo: Cuenta

Nota:
- No se hará uso de los caracteres: Espacio en blanco " ", Caracter de subrayado "_".
- La nomenclatura de argumentos o parámetros pasados a los procedimientos/funciones así como para valores devueltos por funciones sigue las mismas convenciones que la nomenclatura para variables.

6. Beneficios

- La documentación hace más legible un programa.
- Al documentar bien un programa desde un principio se evita que para cada modificación deba estudiarse profundamente el funcionamiento del programa, redescubriendo todo lo no documentado, con la ventaja adicional de que generalmente quién modifica el programa no es siempre quién lo escribió.

- Facilita la reutilización de módulos y rutinas desde cualquier otro programa o el mismo.
- Ayuda a determinar cuándo debe ser reescrito un código. Si existen problemas para explicar el código con un comentario, probablemente el código esté mal escrito.

7. Conclusiones

- Una buena programación e implementación legible, solo se logra usando y llevando de la mano un buen estándar o patrón de programación.
- Es muy importante que el programador tenga un conocimiento previo del estándar o en su defecto que lea el documento para prever diferencias.
- Al documentar se obtienen dos cosas fundamentales, un documento legible y segundo una buena base para los futuros desarrollos de mantenimiento del código.