**Aplicación para la Generación Automatizada de un Diccionario de Datos**

**Versión 2.0**

**Historia de Revisión**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Historial de revisiones** | | | | |
| **Ítem** | **Fecha** | **Versión** | **Descripción** | **Equipo** |
| 1 | 11/06/2025 | 2.0 | Versión Final. | 3 |

**Tabla de Contenidos**

[**1. Objetivo 4**](#_a94q5hnris6w)

[**2. Declaración de variables 4**](#_1fob9te)

[2.1 Descripción de la variable. 6](#_q7sfkbet4f8c)

[2.2 Variables de Tipo Arreglo 6](#_dawiagdjj7za)

[**3. Definición de Controles 6**](#_1phbb8blsytq)

[3.1 Tipo de datos 7](#_xtbn9du9x73j)

[3.2 Prefijo para el Control 7](#_ooi4snwrhyzo)

[3.3 Nombre descriptivo del Control 7](#_on2pn8qeuv67)

[3.4 Declaración de variables, atributos y objetos 8](#_cihm6e2u430m)

[3.5 Declaración de clases 9](#_gi5ey1e7f0ct)

[3.6 Declaración de métodos 9](#_lichjgm73g3w)

[3.7 Declaración de funciones 9](#_lnxbz9)

[3.8 Control de versiones de código fuente 10](#_4hb2f38efbj5)

[3.9 Controles ADO.NET 10](#_wmk86328anc7)

[**4. Clases. 11**](#_44sinio)

[**5. Métodos, Procedimientos y Funciones definidos por el Usuario. 12**](#_u4ny90qz1ytj)

[**6. Beneficios 13**](#_p245e967dc9n)

[**7. Conclusiones 13**](#_sfvjmo2pjjqa)

**Estándares de Programación**

# Objetivo

El objetivo de definir estos estándares es establecer una guía clara y consistente para la escritura del código fuente en el proyecto Generador Automatizado de Diccionarios de Datos, asegurando uniformidad, legibilidad, facilidad de mantenimiento y reducción de errores a lo largo del ciclo de vida del software.

Se busca que todos los desarrolladores sigan un mismo estilo de codificación, de forma que el código sea comprensible no solo por quien lo escribió, sino también por cualquier miembro del equipo que deba revisarlo, depurarlo o ampliarlo.

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:

* La longitud debe ser lo más recomendable posible. No debe ser tan grande de tal forma que el programador tenga la facilidad de manejo sobre la variable y ni tan corta que no pueda describirse claramente. Para el caso establecemos una longitud máxima de variable de 16 caracteres.
* Alcance de la variable

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** |
| --- | --- | --- |
| Nivel de la clase | *M* | clsNUsuarios  clsEEncargado |
| Local del procedimiento / método | Mtd | MtdRegistrarUsuario  MtdBuscarEncargadoID |
| Público | P | pidUsuario |
| Privado | Pr | prCorreo |

* 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 | FeCreacion |

## 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:** ID\_foto, ID\_usuario, ID\_encargado

## 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 más datos, según el tamaño declarado.

**Ejemplos:** listaPublicaciones

# 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.

## 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) |
| Tinyint | tiny | Almacena valores que están dentro de un rango muy limitado. |

## 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.

## 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** |
| jLabel1  jLabel2  jLabel3 | lbl | lblUsuario  lblContraseña  lblRegistrame |
| jTextBox1  jTextbox2 | txt | txtUsurio  txtContraseña |
| jButton | btn | btnEntrar |
| jRadioButton1  jRadioButton2  jRadioButton3 | rbtn | rbtnUsuario  rbtnEncargado  rbtnAdministrador |

|  |  |  |
| --- | --- | --- |
| **Tipo de control** | **Prefijo** | **Ejemplo** |
| jLabel1  jLabel2  jLabel3  jLabel4  jLabel5  jLabel6  jLabel7 | lbl | lblLogo  lblInicio  lblComprte un proble  lblMensajes  lblNotificaciones  lblUsuario  lbltituloPublicacion |
| jTextField1  jTextField2 | txt | txtPublicacion  txtComentario |
| jPanel1  jPanel2 | rbtn | pnMenuPrincipal  pnMain |

|  |  |  |
| --- | --- | --- |
| **Tipo de control** | **Prefijo** | **Ejemplo** |
| jLabel1  jLabel2  jLabel3  jLabel4  jLabel5  jLabel6 | lbl | lblPublicaciones  lblReacciones  lblCantPublicaciones  lblCantReacciones  lblCorreo  lblUsuario |
| jButton1  jButton2 | btn | btnEditarPerfil  btnCerrarSesion |

## 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 ubicación  Indica una variable o atributo que guardará la ubicación de la foto. |

## 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** | Public Class Usuario  Indica una clase Usuario |

## 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** | Public Boolean MtdRegistrarUsuario(clsEUsuarios objEU)  Indica un método MtdRegistrarUsuario recibe el objEU |

**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 Boolean MtdModificarEncargado(clsEEncargado objEE)  Indica una función que modifica los datos de un encargado. |

## 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 |

## 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 |
| **DataRow** | Drw |
| **Connection\*** | Cnn |
| **Command\*** | Cmd |
| **CommandBuilder\*** | Bld |

Ejemplos: de declaración de los objetos ADO.net

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

# Clases.

El nombre de las clases debe ser autodescriptivo 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 de la primera palabra en mayúscula de la carpeta, y luego el nombre que identifica a la clase, la primera letra del nombre también debe iniciar con mayúscula

Las clases que se utilizaron el el codigo fueron:

clsNEncargados

clsNFoto

Conexion

clsNAdministradores

clsNBaneo

clsNComentarios

clsNAdministradores

clsNHistorialBusqueda

clsNPublicaciones

clsNUsuarios

clsEEncargados

clsEFoto

clsEAdministradores

clsEBaneo

clsEComentarios

clsEAdministradores

clsEHistorialBusqueda

clsEPublicaciones

clsEUsuarios

# Métodos, Procedimientos y Funciones definidos por el Usuario.

El nombre de las funciones y procedimientos debe ser autodescriptivo 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 Encargado**

Verbo: modificar

Sustantivo: Encargado

**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.

# Beneficios

* Documentar un programa desde el principio mejora su legibilidad y facilita su mantenimiento.
* Evita la necesidad de estudiar profundamente el funcionamiento cada vez que se realiza una modificación, lo que es especialmente útil cuando el que modifica no fue el autor original.
* Facilita la reutilización de módulos y rutinas en otros programas.
* Ayuda a identificar cuándo es necesario reescribir código: si es difícil explicar una sección con un comentario, es probable que el código esté mal escrito.

# Conclusiones

En la implementación de sistemas como el Generador Automatizado de Diccionarios de Datos, la estandarización en la codificación representa un pilar fundamental para garantizar un desarrollo ordenado, comprensible y sostenible en el tiempo.

Adoptar un conjunto de estándares de programación no solo mejora la legibilidad del código, sino que también facilita la colaboración entre múltiples desarrolladores, minimiza los errores lógicos y acelera los procesos de depuración y mejora continua. Al seguir un patrón común, el equipo de desarrollo puede comunicarse a través del código de forma más clara y efectiva, además se resalta la importancia de la documentación técnica, tanto del sistema como del propio código fuente. Esta no solo sirve como referencia inmediata para los programadores, sino que también se convierte en un activo clave para futuras integraciones, mantenimiento o escalamiento del software.