

**UNIVERSIDAD PRIVADA DE TACNA**

**FACULTAD DE INGENIERÍA**

**Escuela Profesional de Ingeniería de Sistemas**

**Estándares de Programación**

**Proyecto *“SISTEMA DE GESTIÓN HOTELERA”***

Curso: *Programación III*

Docente: Ing Elard Ricardo Rodriguez Marca

Integrantes:

***Nina Vargas, Luigui Augusto 2019065166***

***Chambe Torres, Edgard Reynaldo 2019064917***

***Condori Vargas, Tomas Yoel 2018000487***

***Rojas Bedregal ,Brian Erick***

**Tacna – Perú**

**2023**

**Proyecto *“SISTEMA DE GESTIÓN HOTELERA”***

**Documento de Estándares de Programación**

**Versión 1.0**

**Historia de Revisión**

| **Historial de revisiones** | | | | | |
| --- | --- | --- | --- | --- | --- |
| **Ítem** | **Fecha** | **Versión** | **Descripción** | **Equipo** | |
| 1 |  | 1.0 | Versión Final. |  | |

**Tabla de Contenidos**

1. OBJETIVO 4

2. DECLARACIÓN DE VARIABLES 5

2.1 Descripción de la Variable. 5

2.2 Variables de Tipo Arreglo 5

3. Definición de Controles 6

3.1 Tipo de datos 6

3.2 Prefijo para el Control 6

3.3 Nombre descriptivo del Control 6

3.4 Declaración de variables, atributos y objetos 6

3.5 Declaración de clases 7

3.6 Declaración de métodos 7

3.7 Declaración de funciones 8

3.8 Control de versiones de código fuente 8

3.9 Controles ADO.NET 8

4. Clases. 10

5. Métodos, Procedimientos y Funciones definidos por el Usuario. 10

6. Beneficios 10

7. Conclusiones 11

**Estándares de Programación**

1. **OBJETIVO**

En este documento reglamentaremos la forma en que se implementará el código fuente del proyecto “Sistema de Gestión Hotelera”, pasando, por las variables, controles, clases, métodos,formularios de windows conectado a la base de datos utilizado y todo aquello que esté implicado en el código.

Todo ello con el objetivo de mejorar y uniformizar a través de las reglas que se proponen:

* Los nombres de variables serán mnemotécnicos con lo que se podrá saber el tipo de dato de cada variable con sólo ver el nombre de la variable.
* Los nombres de variables serán sugestivos, de tal forma que se podrá saber el uso y finalidad de dicha variable o función fácilmente con solo ver el nombre de la variable.
* La decisión de poner un nombre a una variable o función será mecánica y automática, puesto que seguirá las reglas definidas por nuestro estándar.
* Permite el uso de herramientas automáticas de verificación de nomenclaturas.
* El uso de distintas herramientas del cuadro para la interfaz gráfica.

Por tanto, se seguirán dichos patrones para un entendimiento legible del código y para facilitar el mantenimiento del mismo.

1. **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** |
| --- | --- | --- |
| Global | G | gstrNombreUsuario |
| Nivel de la clase | *M* | mblnProgresoDelCálculo |
| Local del procedimiento / método | Ninguno | dblVelocidad |
| Público | P | pCantidadUsuario |
| Privado | Pr | prCantidadVenta |

* 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**

* 1. **Descripción de las Variables utilizadas en el Proyecto.**

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.

**Variables del Cargo :**

private int id

private String nombre

**Variables del Cliente:**

private String dni

private String nombre

private String apellido

private String telefono

**Variables de Habitaciones :**

private int id;

private int fkdnicli;

private String numero;

private String tipo;

private double precio;

private String estado;

private byte[] foto;

**Variables del Reseva :**

private int id;

private String cliente\_dni;

private int habitacion\_id;

private String fechareserva ;

private String estado;

private String fecha\_entrada;

private String fecha\_salida;

**Variables del Usuario:**

private String dni;

private String clave;

private String nombre;

private String apellido;

private String email;

private String telefono;

private String estado;

private int fk\_cargo**;**

**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.

* 1. **Tipo de datos**

| **Tipo de variable** | **Mnemónico** | **Descripción** |
| --- | --- | --- |
| Integer | int | Entero de 32 bits con signo. |
| String | st | Cadena de caracteres |
| Float | fl | Coma flotantes, 11-12 dígitos significativos. |
| Double | db | Coma flotante, 64 bits (15-16 dígitos significativos) |
| Object | ob | Objeto genérico |

* 1. **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.

| Tipo de Control | Prefijo |
| --- | --- |
| Label | lbl |
| TextBox | txt |
| Button | btn |
| jtable | TABLA[NOMBRE] |

* 1. **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** | **Descripción** |
| --- | --- | --- |
| TextBox | txt | txtCodbuscar |
| TextBox | txt | txtCodigo |
| TextBox | txt | txtDescripcion |
| TextBox | txt | txtapellido |
| TextBox | txt | txtdni |
| TextBox | txt | txtnombre |
| TextBox | txt | txttelefono |
| TextBox | txt | txtdni |
| TextBox | txt | txtnombre |
| TextBox | txt | txttelefono |
| TextBox | txt | txtdnicli |
| TextBox | txt | txtnrohab |
| TextBox | txt | txtprecio |
| TextBox | txt | txtClave |
| TextBox | txt | txtUsuario |
| TextBox | txt | txtclave |
| TextBox | txt | txtdni |
| TextBox | txt | txtemail |
| TextBox | txt | txtnombre |
| TextBox | txt | txttelefono |
| Button | btn | btnCancelar |
| Button | btn | btnGuardar |
| Button | btn | btnEstadoEmpleado |
| Button | btn | btnBuscar |
| Button | btn | btnSalir |
| Button | btn | btnCargo |
| Button | btn | btnCliente |
| Button | btn | btnHabitaciones |
| Button | btn | btnSalir |
| Button | btn | btnagregarrim |
| Button | btn | btnguardar |
| Button | btn | btnlimpiar |
| Button | btn | btnSalir |
| Button | btn | btnSalirl |
| Button | btn | btnalquilar |
| Button | btn | btnmantenimiento |
| Button | btn | btnCancelar |
| Button | btn | btnListar |
| Button | btn | btnNuevo |
| Button | btn | btnSalir |
| JComboBox | cmb | cmbtipohab |
| JComboBox | cmb | cmbcargo |
| JTable | jt | TABLACLIENTE |
| JTable | jt | TABLALISTA |

* 1. **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 atributos 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. |

* 1. **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 |

| **Título** | **Descripción** |
| --- | --- |
| **Sintaxis** | internal class Conexion |
| **Descripción** | El nombre de las clases tendrá una longitud máxima de 30 caracteres y las primeras letras de todas las palabras estarán en mayúsculas. |
| **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** | En esta clase se realizará la conexión a la base de datos SQL. |

| **Título** | **Descripción** |
| --- | --- |
| **Sintaxis** | **- clsECargo**  **- clsECliente**  **- clsEHabitaciones**  **- clsEReservas**  **- clsEUsuarios** |
| **Descripción** | El nombre de las clases tendrá una longitud máxima de 30 caracteres y las primeras letras de todas las palabras estarán en mayúsculas. |
| **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. |
| **Descripción General** | Todas las clases estarán contenidas en la Capa ‘’.Datos’’ |

| **Título** | **Descripción** |
| --- | --- |
| **Sintaxis** | **RenderTabla**  **clsNCargo**  **clsNCliente**  **clsNHabitaciones**  **clsNReservas**  **clsNUsuarios** |
| **Descripción** | El nombre de las clases tendrá una longitud máxima de 30 caracteres y las primeras letras de todas las palabras estarán en mayúsculas. |
| **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. |
| **Descripción General** | Todas las clases estarán contenidas en la Capa ‘’.Entidad’’,las cuales tendrán los tipos de variables que se usaran junto a la base de datos. |

* 1. **Declaración de funciones**

| **Título** | **Descripción** |
| --- | --- |
| **Sintaxis** | * **public void inicio()** * **public void Limpiar()** * **private void MostrarTabla()** |
| **Descripción** | El nombre del objeto consta 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. |
| **Descripción General** | Todas las funciones estarán contenidas en dentro del formulario “FrmUsuarios”, los cuales al ser ejecutadas permitirán su funcionamiento. |

| **Título** | **Descripción** |
| --- | --- |
| **Sintaxis** | * private void MostrarTabla() |
| **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. |
| **Descripción General** | Todas las funciones estarán contenidas en dentro del formulario “FrmPrincipal”, los cuales al ser ejecutadas permitirán su funcionamiento. |

| **Título** | **Descripción** |
| --- | --- |
| **Sintaxis** | * private void MtdLimp() |
| **Descripción** | El nombre del objeto consta 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. |
| **Descripción General** | Todas las funciones estarán contenidas en dentro del formulario “FrmListaHab”, los cuales al ser ejecutadas permitirán su funcionamiento. |

| **Título** | **Descripción** |
| --- | --- |
| **Sintaxis** | * public void bloqueo() * public void accion(int n,JButton a) * private void actualizo\_estado() |
| **Descripción** | El nombre del objeto consta 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. |
| **Descripción General** | Todas las funciones estarán contenidas en dentro del formulario “FrmHabitaciones”, los cuales al ser ejecutadas permitirán su funcionamiento. |

* 1. **Control de versiones de código fuente**

Cada modificación realizada será guardada de la forma:

| **Título** | **Descripción** |
| --- | --- |
| **Formato** | [Reserva][ \_ ][FECHA][ \_ ][HORA] donde y la fecha estará en formato yyyymmdd y la hora en formato HHMM. |
| **Descripción** | Se generarán archivos con las siguientes extensiones: PDF O EXCEL.Por ejemplo: Reserva |

* 1. **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
* System.Data.SqlClient
* System.Data
* System.Threading.Tasks

1. **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 del nombre que identifica la clase, la primera letra del nombre debe iniciar con mayúscula.

Nombre de las Clases utilizadas en el Proyecto:

* **clsECargo**
* **clsECliente**
* **clsEHabitaciones**
* **clsEReservas**
* **clsEUsuarios**
* **RenderTabla**
* **clsNCargo**
* **clsNCliente**
* **clsNHabitaciones**
* **clsNReservas**
* **clsNUsuarios**

**Nota:**

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

1. **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 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.

1. **Beneficios**

* La documentación hará más legible nuestro programa.
* 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.
* Se han implementado tipos de variables en las cuales se almacenará la información que estará conectada a la base de datos SQL.
* El uso de clases para guardar los datos y secuencia de objetos del mismo tipo almacenados consecutivamente en la memoria.
* Uso de condicionales que nos permitirán establecer una serie de condiciones al interior de nuestro programa, que nos ayudará a determinar qué acciones llevará cabo dadas ciertas condiciones.

1. **Conclusiones**

* Una buena programación e implementación legible, sólo se logra usando y llevando de la mano un buen estándar o patrón de programación.
* Al culminar con la realización del proyecto para el beneficio de los registros en la empresa, de acuerdo con lo establecido, se procedió a realizar la programación en lenguaje Java.
* Se implementó todo lo realizado en el curso de Programación III en este proyecto.