**Plataforma de comercio electrónico aplicando inteligencia artificial para aumentar las ventas de celulares en la empresa MijoStore**

**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. OBJETIVOS 4**](#_heading=h.asxqpbktzgls)

[**2. DECLARACIÓN DE VARIABLES 5**](#_heading=h.r7i190edobnn)

[2.1 Descripción de la Variable. 6](#_heading=h.9kia0rcbseis)

[**3. Definición de Controles 6**](#_heading=h.acwa9qywrq59)

[3.1 Tipo de datos 6](#_heading=h.65z2963ztxza)

[3.2 Declaración de variables y atributos 6](#_heading=h.axot5bpxpj8m)

[3.3 Declaración de clases 7](#_heading=h.i7131yuqx1s8)

[3.4 Declaración de métodos 9](#_heading=h.y1ag4rcmkzj1)

[3.5 Control de versiones de código fuente 10](#_heading=h.cl42s5ekyyyq)

[**4. Clases. 11**](#_heading=h.44sinio)

[**5. Métodos y Funciones definidos 11**](#_heading=h.ehgwqavtdls3)

[**6. Beneficios 13**](#_heading=h.3ikyy2v8r484)

[**7. Conclusión 13**](#_heading=h.6it2m2muxp6a)

**Estándares de Programación**

# OBJETIVOS

* Establecer un conjunto integral de estándares de programación que abarque desde la estructura del código hasta las prácticas de documentación y control de versiones, con el fin de garantizar la consistencia y la calidad en todos los aspectos del desarrollo de software.
* Promover la modularidad y la reutilización de código a través de la definición de interfaces claras y la separación de preocupaciones, con el fin de facilitar el mantenimiento, la escalabilidad y la extensibilidad de los sistemas de software.
* Fomentar la adopción de prácticas de desarrollo ágil, como la entrega continua y la automatización de pruebas, para acelerar el ciclo de desarrollo, mejorar la calidad del software y adaptarse más fácilmente a los cambios en los requisitos del proyecto.
* Facilitar la colaboración y el intercambio de conocimientos entre los miembros del equipo de desarrollo mediante la creación de una documentación exhaustiva de los estándares de programación, la organización de sesiones de formación y la celebración regular de reuniones para revisar y actualizar los estándares según sea necesario.

# 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**

## Descripción de la Variable.

Es el nombre que se asignará a cada variable para identificarla claramente. Este nombre debe estar relacionado con la función o propósito para el cual se ha declarado la variable dentro de la base de datos o el sistema. Es importante que los nombres sean representativos y sigan una convención coherente para facilitar la comprensión y el mantenimiento del código.

Ejemplos:

idUsuario: Identificador único de cada usuario.

nombreProducto: Nombre del producto.

precioTotalPedido: Monto total de un pedido.

fechaPromocion: Fecha de inicio de una promoción.

# Definición de Controles

## Tipo de datos

| **Tipo de variable** | **Mnemónico** | **Descripción** |
| --- | --- | --- |
| Int | int | Almacena números enteros. |
| varchar | varchar | Almacena texto de longitud variable. |
| text | text | Almacena texto largo. |
| decimal | decimal | Almacena números con decimales. |
| date | date | Almacena fechas. |
| time | time | Almacena horas. |
| tinyint | tinyint | Almacena pequeños enteros, típicamente 0 o 1. |

## Declaración de variables y atributos

| **Título** | **Descripción** |
| --- | --- |
| **Sintaxis** | String Clave |
| **Descripción** | Variable que almacena la contraseña del usuario |
| **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 contraseña  Indica una variable que guardará la contraseña de una variable. |

## Declaración de clases

| **Título** | **Descripción** |
| --- | --- |
| **Sintaxis** | class Caracteristica |
| **Descripción** | La clase Caracteristica gestiona las especificaciones técnicas de un producto, como sistema operativo, RAM, cámara, batería, entre otros. Contiene atributos para almacenar estos detalles y probablemente se conecta a una base de datos para insertar, actualizar o recuperar las características asociadas a un producto específico. |
| **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** | Caracteristica |

| **Título** | **Descripción** |
| --- | --- |
| **Sintaxis** | class Categoria |
| **Descripción** | La clase Categoria parece estar diseñada para gestionar las categorías de productos |
| **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** | Categoria |

| **Título** | **Descripción** |
| --- | --- |
| **Sintaxis** | class Imagen |
| **Descripción** | La clase Imagen parece estar diseñada para gestionar las imágenes asociadas a productos dentro de un sistema |
| **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** | Imagen |

| **Título** | **Descripción** |
| --- | --- |
| **Sintaxis** | class Pedido |
| **Descripción** | La clase Pedido parece estar diseñada para gestionar los pedidos realizados por los usuarios en un sistema de ecommerce |
| **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** | Pedido |

| **Título** | **Descripción** |
| --- | --- |
| **Sintaxis** | class Producto |
| **Descripción** | La clase Producto gestiona la información de los productos en un ecommerce |
| **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** | Producto |

| **Título** | **Descripción** |
| --- | --- |
| **Sintaxis** | class Promocion |
| **Descripción** | La clase Promocion esta diseñada para gestionar promociones en un sistema de ecommerce |
| **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** | Promocion |

| **Título** | **Descripción** |
| --- | --- |
| **Sintaxis** | class TipoPromocion |
| **Descripción** | La clase TipoPromocion parece estar diseñada para gestionar los tipos de promociones disponibles en un sistema de ecommerce. |
| **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** | TipoPromocion |

| **Título** | **Descripción** |
| --- | --- |
| **Sintaxis** | class Usuario |
| **Descripción** | La clase Usuario parece estar diseñada para gestionar la información de los usuarios en un sistema |
| **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** | Usuario |

## Declaración de métodos

| **Título** | **Descripción** |
| --- | --- |
| **Sintaxis** | guardar() |
| **Descripción** | Este método inserta un nuevo registro de usuario en la base de datos. Utiliza una consulta INSERT para añadir los datos del usuario (como nombre, apellido, correo y clave) a la tabla usuario. Si la consulta se ejecuta correctamente, devuelve true, de lo contrario, false. |
| **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** | guardar() |

| **Título** | **Descripción** |
| --- | --- |
| **Sintaxis** | iniciarSesion() |
| **Descripción** | Este método se utiliza para verificar las credenciales de inicio de sesión de un usuario. Realiza una consulta en la base de datos para buscar un usuario con el correo proporcionado y luego verifica que la contraseña ingresada coincida con la almacenada en la base de datos usando password\_verify. Si ambos son correctos, devuelve el objeto del usuario, de lo contrario, devuelve false. |
| **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** | iniciarSesion() |

| **Título** | **Descripción** |
| --- | --- |
| **Sintaxis** | actualizar() |
| **Descripción** | Este método se utiliza para actualizar los datos de un usuario en la base de datos. Antes de realizar la actualización, verifica si el correo electrónico ya está en uso por otro usuario. Si el correo no está en uso, procede a actualizar los valores del nombre, apellido y correo, y si la imagen ha sido proporcionada, también se actualiza. Finalmente, retorna true si la actualización fue exitosa, o false si hubo algún error. |
| **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** | actualizar() |

| **Título** | **Descripción** |
| --- | --- |
| **Sintaxis** | eliminar() |
| **Descripción** | Este método elimina un usuario de la base de datos según su id. Ejecuta una consulta DELETE en la tabla usuario para eliminar el registro correspondiente. Si la operación se ejecuta correctamente, devuelve true, de lo contrario, false. |
| **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** | eliminar() |

## Control de versiones de código fuente

Cada modificación realizada será guardada de la forma:

| **Título** | **Descripción** |
| --- | --- |
| **Formato** | git commit -m “[Nombre del modulo] - [Descripción del avance]” |
| **Descripción** | Se generarán commits de acuerdo al avance del form con una breve descripción de las modificaciones. Por ejemplo: Login - Avance de interfaz |

# Clases.

En el desarrollo del sistema de ecommerce, se han diseñado diversas clases para gestionar de manera eficiente los diferentes aspectos del proceso de compra y administración. Estas clases representan las entidades principales del sistema y permiten organizar y manipular los datos de forma estructurada. Las clases utilizadas son las siguientes:

1. **Caracteristica**: Gestiona las características específicas de los productos, como especificaciones técnicas y detalles importantes.
2. **Categoria**: Organiza los productos en categorías para facilitar la búsqueda y navegación dentro del ecommerce.
3. **Imagen**: Maneja las imágenes asociadas a los productos, asegurando una presentación visual adecuada en la tienda.
4. **Pedido**: Registra los detalles de los pedidos realizados por los usuarios, incluyendo la información de envío y el estado del pedido.
5. **Producto**: Almacena la información esencial de los productos, como su nombre, descripción, precio, stock y otros atributos clave.
6. **Promocion**: Gestiona las promociones y descuentos aplicados a los productos en el ecommerce, mejorando la experiencia de compra para los clientes.
7. **TipoPromocion**: Define los distintos tipos de promociones disponibles en el sistema, permitiendo aplicar descuentos específicos según el tipo de promoción.
8. **Usuario**: Administra la información de los usuarios del sistema, permitiendo su autenticación, roles y gestión de perfiles.

# Métodos y Funciones definidos

class Caracteristica

obtenerCaracteristicasProducto($producto\_id)

guardar()

actualizar()

eliminar()

class Categoria

obtenerCategorias()

obtenerCategoria()

guardar()

eliminar()

obtenerCategoriaActual()

actualizar()

class Imagen

obtenerImagenesProducto()

guardar()

actualizar()

eliminar()

guardar\_imagen() (estática)

obtener\_imagen\_por\_id() (estática)

eliminar\_imagen() (estática)

actualizar\_imagen() (estática)

class Pedido

obtenerProductos()

guardar()

obtenerProductoActual()

guardarProductosPedido()

obtenerUltimoPedido()

obtenerTodosPorUsuario()

obtenerPedidoEspecifico()

obtenerUltimoPorUsuario()

obtenerProductosPorPedido()

obtenerUnidades()

actualizar()

class Producto

obtenerProductos()

guardar()

actualizar()

eliminar()

obtenerProductoActual()

obtenerAleatorio()

obtenerProductosPorCategoria()

obtenerProductosEnPromocion()

obtenerImagenes($producto\_id)

class Promocion

obtenerPromociones()

guardar()

actualizar()

desactivar()

obtenerPromocionesProducto($producto\_id)

eliminar()

obtenerPorId($id)

class TipoPromocion

getId()

setId($id)

getNombre()

setNombre($nombre)

obtenerTiposPromocion()

guardar()

actualizar()

eliminar()

class Usuario

guardar()

iniciarSesion()

actualizar()

# Beneficios

* Establece reglas y convenciones que garantizan que el código sea uniforme y fácilmente comprensible por todos los miembros del equipo.
* Los estándares promueven buenas prácticas de codificación, lo que puede llevar a la reducción de errores y a un código más robusto y mantenible.
* Al seguir estándares comunes, los desarrolladores pueden trabajar de manera más eficiente en equipo, entender el código de los demás y colaborar con mayor fluidez.
* Evitar debates innecesarios sobre la estructura y el estilo del código, permitiendo a los desarrolladores concentrarse en tareas más importantes.
* El código estandarizado es más fácil de integrar con otros sistemas y de escalar a medida que el proyecto crece.

# Conclusión

La documentación de estándares de programación es fundamental para asegurar la consistencia y calidad del código en un proyecto de desarrollo de software. Al establecer reglas y convenciones claras, promueve la colaboración efectiva, mejora la mantenibilidad del código y contribuye a la eficiencia y éxito del equipo de desarrollo. Es una inversión importante que puede proporcionar beneficios a largo plazo en términos de productividad y calidad del producto final.