**Ejercicio #6: Nomenclatura**

**Variables, Tipos y Constantes:**

1. **Definición:**

Las variables son “contenedores” que almacenan datos que pueden cambiar durante la ejecución de un programa. Se utilizan para guardar y modificar valores de forma dinámica durante la ejecución de un programa.

Las constantes son “contenedores” que almacenan datos cuyo valor no puede modificarse una vez que se asigna inicialmente. Las constantes se utilizan para almacenar valores que no van a cambiar durante la ejecución de un programa.

Los tipos son Valores…

1. **Concepto:**

La declaración de variables y constantes debe hacerse de forma clara y explicita, con el fin de mantener un código legible (Clean Code), seguro y de fácil comprensión para cuando se requiera realizar modificaciones al código.

1. **Estructura:**
2. Debe ser escrito siempre en minúscula.
3. Si la declaración hace referencia a un nombre compuesto (Ej. Dato Sesión), se unifican por un “\_” quedando así: “dato\_sesion”.
4. Son escritas de forma singular.
5. Si se va a declarar un tipo Array se puede hacer en singular.

**Funciones**

1. **Definición**:

Una función es un bloque de código diseñado para realizar una tarea especifica. Se ejecuta cuando es llamada desde “otra sección” del programa. Estas pueden o no recibir o devolver un valor. Dentro del bloque de la función se escriben los pasos que se requieran para que realice la tarea requerida.

Las funciones se crean con el fin de organizar el código de manera eficiente, escalable y reutilizable (El ideal es evitar repeticiones).

1. **Concepto:**

La declaración de funciones debe hacerse de forma clara y explicita, con el fin de mantener un código legible (Clean Code), seguro y de fácil comprensión para cuando se requiera realizar modificaciones al código.

Se realiza con nombres descriptivos que indiquen claramente qué hace la función.

* 1. Implementar verbos en los métodos para indicar su acción.
  2. Siempre especificar el retorno del método. (Así no devuelva un valor se debe marcar la función de que no retorna nada).
  3. Se deben crear métodos o funciones de longitud y complejidad corta con el fin de no caer en código “Espaguetti”.
  4. Se debe definir cuál será su forma encapsulamiento:
     1. Public
     2. Private
     3. Protected

1. **Estructura:**
2. La mayoría de las funciones o métodos hace referencia a:
   1. Post (Crear)
   2. Delete (Borrar)
   3. Get (Tomar)
   4. Update (Actualizar)
   5. Print (Imprimir)
3. Debe ser en mayúscula su inicial.
4. Se pueden combinar con el objetivo de la función, es decir si la función va a obtener una lista, una tabla, actualizar un registro, obtener un objeto.
5. Ejemplo:
   1. PostUsuario
   2. DeleteUsuario
   3. GetUsuario
   4. GetListUsuarios (Va en plural si se requiere una lista)
   5. GetTableUsuarios.

**Etiquetas**

* Definir un encabezado ayuda a mantener el código organizado y accesible, facilitando la gestión y mantenimiento a largo plazo.
* La estructura del encabezado propuesta la es la siguiente:

Texto

Descripción generada automáticamente con confianza media

**Modelo de capas**

* Teniendo en cuenta que los proyectos se dividen en 2 partes, el FrontEnd y el Backend para cada una de estas se define la siguiente estructura de código:

1. Para el Front es la siguiente:

Texto

Descripción generada automáticamente

1. Para el Back la siguiente:

Texto

Descripción generada automáticamente

1. **Descripción funciones:**

* Al momento de crear cada una de las funciones, es importante dejar una breve descripción de la acción que realiza la misma con el fin de que cualquier persona que tenga acceso a esta, pueda entenderla.

Texto

Descripción generada automáticamente

1. **Comentarios y anotaciones especificas**:

* Para adquirir y aplicar buenas prácticas al generar código, es fundamental implementar comentarios breves y específicos dentro de las líneas de código. Estas anotaciones ayudan tanto al autor original como a cualquier otra persona que acceda al código, facilitando la comprensión de la lógica, funcionalidad y propósito de este.

Texto

Descripción generada automáticamente

1. Logs

* Al momento de realizar la lógica del backend, es de suma importancia el tener considerado un registro de log, este permitirá el monitoreo, la depuración y la auditoria de la aplicación.
* Dentro del registro de logs existen niveles de información que ayudan a definir dentro del seguimiento del funcionamiento del código, dichos niveles se dividen de la siguiente forma:
  1. **Debug:** Información detallada, típicamente de interés solo cuando se está depurando.
  2. **Info:** Confirmaciones de que las cosas están funcionando como se espera.
  3. **Warn:** Indicación de que algo inesperado ocurrió, o indicativo de algún problema en el futuro cercano.
  4. **Error:** Debido a un problema mas grave, la funcionalidad ha sido afectada.
  5. **Fatal:** Errores críticos causando que el programa cese algunas funciones o se detenga completamente.

1. **Proteger respuestas del backend**.

Teniendo siempre presente el principio de crear código implementando las mejores e idóneas prácticas de programación y estándares de seguridad para el consumo de EndPoint. Se requiere siempre que cada respuesta que se genere a partir de un endpoint, esta se encuentre encriptada, y para poder consumir el servicio siempre sea solicitado el token suministrado.