

**Sistema de Implementación de un sistema
de API rest de cursos para la facultad de Epis de
la UPT
Documento de Estándares de Programación
Versión 1.0**

SISTEMA Implementación de un sistema de API rest de cursos para la facultad de Epis de la UPT	Versión: 1.0
Documento de Estándares de Programación	

Historia de Revisión

Historial de revisiones				
Ítem	Fecha	Versión	Descripción	Equipo
1	27/11/2022	1.0	Versión Final.	Helbert Condori Loayza, Diego Aranda Reyes, Erick Mamani Lima y Soledad Maltrain Yáñez

Tabla de Contenidos

1.	OBJETIVO	4
2.	DECLARACION DE VARIABLES	5
2.1	Descripción de la Variable.	5
2.2	Variables de Tipo Arreglo	6
3.	Definición de Controles	7
3.1	Tipo de datos	7
3.2	Prefijo para el Control	7
3.3	Nombre descriptivo del Control	7
3.4	Declaración de variables, atributos y objetos	7
3.5	Declaración de clases	8
3.6	Declaración de métodos	8
3.7	Declaración de funciones	9
3.8	Control de versiones de código fuente	9
4.	Clases.	10
5.	Métodos, Procedimientos y Funciones definidos por el Usuario.	10
6.	Beneficios	11
7.	Conclusiones	11

SISTEMA Implementación de un sistema de API rest de cursos para la facultad de Epis de la UPT	Versión: 1.0
Documento de Estándares de Programación	

Estándares de Programación

1. OBJETIVO

Reglamentar la forma en que se implementará el código fuente del proyecto, pasando, por las variables, controles, clases, métodos, ficheros, archivos y todo aquello que esté implicado en el código,

Mejorar y uniformizar a través de las reglas que se proponen, el estilo de programación que tiene cada programador.

- 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.

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

2. DECLARACION 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	Ninguno	NombreProfesor
Nivel de la clase	<i>Ninguno</i>	APIProfesorViews
Local del procedimiento / método	Ninguno	get
Público	Ninguno	datos
Privado	Ninguno	CodigoProfesor

- 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	<i>Minúscula la primera parte y luego la segunda con Mayúsculas</i>
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: idCuenta, tipoEstado, instalacion

SISTEMA Implementación de un sistema de API rest de cursos para la facultad de Epis de la UPT	Versión: 1.0
Documento de Estándares de Programación	

2.2 Variables de Tipo Lista

En el caso de las definiciones de Lista de elementos se declarará la variable con el prefijo de las 3-4 primeras letras abreviadas, tomamos como ejemplo a “list”, 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: list

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	Descripción
Byte	Entero de 8 bits sin signo.
Integer	Entero de 32 bits con signo.
Char	Un carácter UNICODE de 16 bits
String	Cadena de caracteres
Date	Formato de fecha/hora
Boolean	Valor lógico: verdadero y falso
Float	Coma flotante, 11-12 dígitos significativos.
Double	Coma flotante, 64 bits (15-16 dígitos significativos)
Object	Objeto genérico

3.2 Prefijo para el Control

No se harán uso de controles

3.3 Nombre descriptivo del Control

No se harán uso de controles

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: <ul style="list-style-type: none"> • 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.

SISTEMA Implementación de un sistema de API rest de cursos para la facultad de Epis de la UPT	Versión: 1.0
Documento de Estándares de Programación	

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: <ul style="list-style-type: none"> • Letra Ñ o ñ. • Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ', +, -, *, {, }, [,], _. • Caracteres tildados: á, é, í, ó, ú.
Ejemplo	Private Class APIProfesorViews Indica una clase Profesor

3.6 Declaración de métodos

Título	Descripción
Sintaxis	[Tipo] [nombredelmetodo] [(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: <ul style="list-style-type: none"> • Letra Ñ o ñ. • Caracteres especiales ¡, ^, #, \$, %, &, /, (,), ¿, ', +, -, *, {, }, [,], _. • Caracteres tildados: á, é, í, ó, ú.
Ejemplo	def get(self , request,CodigoProfesor=0): Indica un método get que recibe 3 variables, una self, otra request y otra CodigoProfesor

SISTEMA Implementación de un sistema de API rest de cursos para la facultad de Epis de la UPT	Versión 1.0
Documento de Estándares de Programación	

3.7 Declaración de funciones

No se hicieron uso de funciones

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 yyymmdd y la hora en formato HHMM.
Descripción	Se generarán archivos con las siguientes extensiones:.zip o .rar. Por ejemplo: WSTENNIS_20070421_2056.zip

SISTEMA Implementación de un sistema de API rest de cursos para la facultad de Epis de la UPT	Versión: 1.0
Documento de Estándares de Programación	

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.

Además, debe indicar a que parte de la capa pertenecen como en el caso de ser la capa de datos descentralizada se agrega al final la palabra **tbNOMBRE** si es una clase del orm de django mientras que en las demás se deja como **APINOMBREViews**.

- Ejemplos: **APIProfesorViews**, **APICursosViews**, **TbProfesor**

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.

SISTEMA Implementación de un sistema de API rest de cursos para la facultad de Epis de la UPT	Versión 1.0
Documento de Estándares de Programación	

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.