HOPARK
Un espacio para ti.

12.16.2020

# Proyecto Hopark

## Plan de Configuración

Hopark
Ingeniería de Software

Integrantes:

- Abdón Calderón.
- Roberto Goyes.
- Esteban Vera.

# 1. Propósito

El propósito de este documento es definir de una manera clara y concisa como se van a manejar los estándares y procedimientos utilizados para la gestión de la configuración de los documentos elaborados por Hopark para la ejecución de este proyecto.

# 2. Estandarización de Codificación

## 2.1. Creación de Librería de Clases

Toda creación de librería tendrá el siguiente formato la inicial del nombre del proyecto seguido del punto (.) y un nombre que denote a la librería de clases.

*Ejemplo:*

H.DatosUsuario.

## 2.2. Creación de Clases

Los nombres de las clases deben empezar con mayúsculas.

Para nombres compuestos la primera letra de cada palabra empieza con mayúsculas.

*Ejemplo:*

Estilo.css

BuscarParqueo.py

## 2.3. Organización de la Clase

***2.3.1.*** Encabezado de la Clase.
***2.3.2.*** Sentencias import.
***2.3.3.*** Declaración de Clase.
**2.3.4.** Constantes.
**2.3.5.** Declaración de variables.
- Variables de Instancias.

**2.3.6.** Métodos.
**2.3.7.** Sentencia.
**2.3.8.** Comentarios

### Descripción de cada parte de la Organización de Clases.

### 2.3.1. Encabezado de la Clase.

El contenido que tendrá el encabezado de la Clase será:

- Nombre de la Clase: como se llamará el archivo que se está desarrollando.
- Autor: Persona que implementa la Clase.
- Versión: El formato es **V1**.
- Librería de Ubicación: según lo planteado en el punto anterior de creación de librerías
- Descripción: Explicación de cómo funciona una determinada clase que se está implementando.

Ejemplo:

```
/*-------------------------------------------------------------------------------------------------------------
*               Nombre de Clase:      CrearUsuario.py
*               @author               Abdon Calderon L.
*               @version              V 1
*               Librería de ubicación: H. CrearUsuario
                Descripción:          Breve explicación del funcionamiento de esta
                                      Clase de cómo esta su implementación
-------------------------------------------------------------------------------------------------------------*/
```

### 2.3.2 Sentencias Import

La sentencia import nos indica las determinadas librerías que va a utilizar en la clase que se está implementando. El formato es el siguiente primero se coloca la palabra reserva **import,** y luego el nombre de la librería que se desea importar.

Ejemplo:

```
import H.DatosUsuario
```

### 2.3.3 Declaración de Clase

Define la estructura que tendrá la clase. El formato es el siguiente; la palabra clave: “def”, y por ultimo el nombre de la clase.

Ejemplo:

- def BuscarUsuario()

### 2.3.4 Constante

En Python, las constantes son usualmente declaradas y asignadas en un módulo. Aquí, el módulo significa un nuevo archivo que contiene variables, funciones, etc; el cual es importada en el archivo principal. Dentro del módulo, las constantes son escritas en letras MAYÚSCULAS y separadas las palabras con el carácter underscore _.

Ejemplo:

```
IP_DB_SERVER = "127.0.0.1"

PORT_DB_SERVER = 3307

USER_DB_SERVER = "root"

PASSWORD_DB_SERVER = "123456"

DB_NAME = "nomina"
```

### 2.3.5 Declaración de Variables.

Cada variable debe tener un nombre único llamado identificador. Eso es muy de ayuda pensar las variables como contenedores que contienen data el cual puede ser cambiado después a través de técnicas de programación.

Ejemplo.

```
Var xxxx

String xxx

Int xxx
```

Nota: Los nombres de las variables deben ser significativos de acuerdo a la funcionalidad que tiene la variable.

**Variable de Instancia.**

Las variables de instancia deben tener el mismo nombre de la clase, pero deben ir en minúscula, en el caso de ser nombre compuesto los subnombres irán seguido de subguión (_).

Ejemplo:

```
Usuario_usuario
```

### 2.3.6 Métodos.

Definir un método es bastante simple, sólo tenemos que añadirlo en la clase y luego llamarlo desde el objeto con los paréntesis, como si de una función se tratase:

Ejemplo:

```
class Galleta:
    chocolate = False

    def saludar():
        print("Hola, soy una galleta muy sabrosa")

Galleta.saludar()
```

### 2.3.7 Sentencias

- **Sentencia Sato**

**if/else:** La sintaxis es la siguiente.

```
if galleta.chocolate:
    print("La galleta tiene chocolate")
else:
    print("La galleta no tiene chocolate")
```

**For:** la sentencia es la siguiente:

```
for i in [0, 1, 2]:
    print("Hola ", end="")
print()
```

**While:** la sentencia es la siguiente.

```
while i <= 50:
    print(i)
    i = 3 * i + 1
```

### 2.3.8 Comentarios

Una norma de programación es que cada método debe tener su comentario; describiendo cual es la función del método como que parámetros recibe y que parámetros retorna.

- **Comentarios de Línea.**
  Solo se inicializa con dos //  y seguido del texto que desea comentar.

  Ejemplo:

  ```
  //descripción de una variable
  ```

- **Comentarios de Bloques.**
  Se inicializa con /* seguido del texto que desea como comentario y se cierra con */. Este tipo de comentario puede tener cualquier cantidad de líneas que desee.

  Ejemplo:

  ```
  /*descripción  en varias
    Líneas
  */
  ```