Universidad Carlos III

Curso Desarrollo de Software 2021-22

Práctica

Curso 2021-22

Normativa de Código

Fecha: **11/02/22**

GRUPO: **80** EQUIPO: **06**

Alumnos: **Jorge Álvarez Rodríguez / Alba Marco Ugarte**

ÍNDICE

1. Estándar para la organización de archivos

1.1 Leyenda de derechos de autor

Al principio de nuestros archivos, habrá una leyenda donde se especificará: el título del archivo, versión, descripción del contenido del archivo, autor , fecha de creación del archivo y fecha de la última modificación.

A continuación, se muestra un ejemplo de código en el que se aplica esta normativa para poder determinar los derechos de autor que tendrá el archivo en cuestión.

```
8   """
9
10  Titulo: Normativa de Codigo Python
11  Version: Python 3.8
12  Descripcion: Ejemplo de normativa de código para EG2
13  Autor: Alba Marco Ugarte (100451139@alumnos.uc3m.es)
14         Jorge Alvarez Rodriguez (100451189@alumnos.uc3m.es)
15  Fecha_creacion: 08/02/2022
16  Ultima_modificacion: 11/02/2022
17
18  """
```

1.2 Gestión de Ficheros

Cada uno de los ficheros fuente y ficheros de interés deberán ser gestionados y almacenados en forma de carpetas para mantener una organización clara y sencilla. En caso de tratarse de un fichero ejecutable, éste deberá ser almacenado en una carpeta llamada “exe”; si se tratase de un fichero relacionado con una biblioteca de funciones, deberá ser guardado en una carpeta que reciba el nombre de “library” y en caso de tratarse de un fichero fuente relacionado con el tipo de letra que puede ser empleado, se gestionará dentro de una carpeta llamada “typefont”.

1.3 Ficheros de clases

Todas las clases que vayan a ser usadas en el código tendrán su propio fichero y se nombrará usando el estilo CamelCase. Por ejemplo, *VaccineManager.py*.
La cabecera de los ficheros que contengan una clase deberán contener la siguiente información: Nombre de la clase, descripción de la misma y precauciones para evitar futuros problemas.

A continuación, se expone un ejemplo sobre la cabecera.

```
"""

Nombre: NormativaSoftware
Descripción: Se expone la normativa de ccódigo que será necesaria implementar a la hora
             de realizar cualquier tipo de programación.
Precauciones:Si alguno de los ficheros es modificado, cada uno de los integrantes encargados del
             desarrollo del código, deberán ser avisados para evitar posibles confusiones.

"""
```

2. Estándar para Nombres y Variables

2.1 Clases y miembros de clases

Para la declaración de clases emplearemos siempre el tipo de fuente UpperCamelCase, o también conocido como PascalCase. Éste se caracteriza por tener la primera letra de cada una de las palabras en mayúsculas. Por ejemplo, *VaccineManager*.

Por otro lado, para las variables, nombre de funciones, métodos, atributos y argumentos, haremos uso del CamelCase. En cuanto a las constantes, se hará uso de la UpperSnake. Ésta se caracteriza por tener todas las letras que forman la palabra en mayúsculas, y en caso de ser más de una palabra, se encontrarán separadas por una barra baja. Por ejemplo, *VACCINE_DATE.*

Todos los nombres de variables y atributos estarán limitados a dos palabras como máximo. A excepción de los métodos y clases, que serán tres.

2.2 Visibilidad

Todas las variables de las clases tendrán que ser declaradas en forma de propiedades. El uso de propiedades nos permitirá exponer en una clase una manera pública de establecer y obtener valores. También permitirá ocultar el código de implementación o verificación.

Por otro lado, cada una de las declaraciones deberá corresponder con una línea de código única y diferente del resto. Además, las declaraciones e inicializaciones consecutivas deberán estar completamente alineadas en forma de tabla, para poder mantener una estructura clara, sencilla y ordenada.

Todas las variables globales deberán ser inicializadas al comienzo del documento para evitar posibles confusiones futuras. En caso de tratarse de una variable necesaria para alguno de

los métodos o funciones establecidas, deberán ser declaradas seguidas del propio nombre de éstos.

3. Estándar para métodos

3.1 Estructura y declaración de métodos

Los métodos deberán seguir la siguiente estructura establecida en caso de emplear Python:

- Habrá un comentario inicial que incluya el título del método en mayúsculas para poder diferenciarlo del resto. Seguido a éste, deberá haber un comentario a modo de explicación que especifique qué es lo que realiza. Así se dejará claro cuál es el objetivo del mismo.
- La correcta indentación o sangrado será indispensable para que el programa pueda detectar correctamente cada una de las líneas de código programadas. De esta forma, se evitarán posibles errores. El indentado en Python además de permitir que el código sea más legible, nos ahorra tener que hacer uso de las llaves .
- En caso de que el método contenga más de cuatro argumentos, los siguientes se escribirán en la siguiente línea. A continuación, se muestra un ejemplo:

```
9    def metodoPrueba (self, nombre, fecha, anio,
10                     color, numero):
```

- No se definirá un tamaño máximo para los métodos.
- En el cuerpo de cada método, deberá existir una separación entre cada uno de los componentes que lo conforman, para que a la vista sea más clara y sencilla su comprensión.

4. Estándar para el tratamiento de excepciones

La detección de errores y excepciones serán llevadas a cabo directamente por los propios componentes del método. En caso de ser introducido un argumento no válido, el propio componente detectará el inconveniente y mostrará un mensaje de error.

5. Reglas modificadas en pylintrc

Aproximadamente 17 reglas han sido modificadas en el pylint.rc en base a nuestra normativa de código, y son las siguientes:

1) Las normativas relacionadas con el estilo del tipo de fuente han sido modificadas a sus respectivos tipos. En el caso de los argumentos, atributos, funciones, métodos y variables se ha establecido el tipo de fuente camelCase. Ejemplo:

```
- # Naming style matching correct argument names.
argument-naming-style= camelCase
```

2) Para cada uno de los tipos de fuente que han sido modificados, hemos tenido que establecer una expresión regular que correspondiese con ella. A continuación, se muestra un ejemplo de la expresión regular correspondiente a una clase (PascalCase):

```
- # Naming style matching correct class names.
class-naming-style=PascalCase
class-rgx=^[A-Z]{1}[a-z]{0,20}[A-Z]{0,1}[a-z]{0,20}\Z
```

En el caso de la clase, forzamos mediante la expresión regular a que empiece por una mayúscula cada una de las palabras que conforman el nombre de ésta. También hemos establecido una expresión regular más concreta para aquellos atributos (camelCase) que son de tipo privado. Estos son aquellos que pueden comenzar por un "_". Ejemplo de nuestra regla:

```
- attr-rgx= ^[_]{0,2}[a-z]{1,20}[A-Z]{0,1}[a-z]{0,20}\Z
```

3) Hemos modificado algunas de las palabras prohibidas (bad-names) que no podría permitir nuestro código a la hora de su diseño. A continuación, se muestran las palabras que hemos definido como "no permitidas":

```
-bad-names=kacch,
    darwin,
    escorial,
    rodri,
    fluvia,
    racist
```

4) También hemos tomado la decisión de cambiar el número máximo de caracteres para una única línea. Cien caracteres nos parecían pocos, en caso de que un usuario quisiese desarrollar un comentario mucho más extenso. Es por ello que hemos establecido doscientos caracteres como límite. La regla es la siguiente:

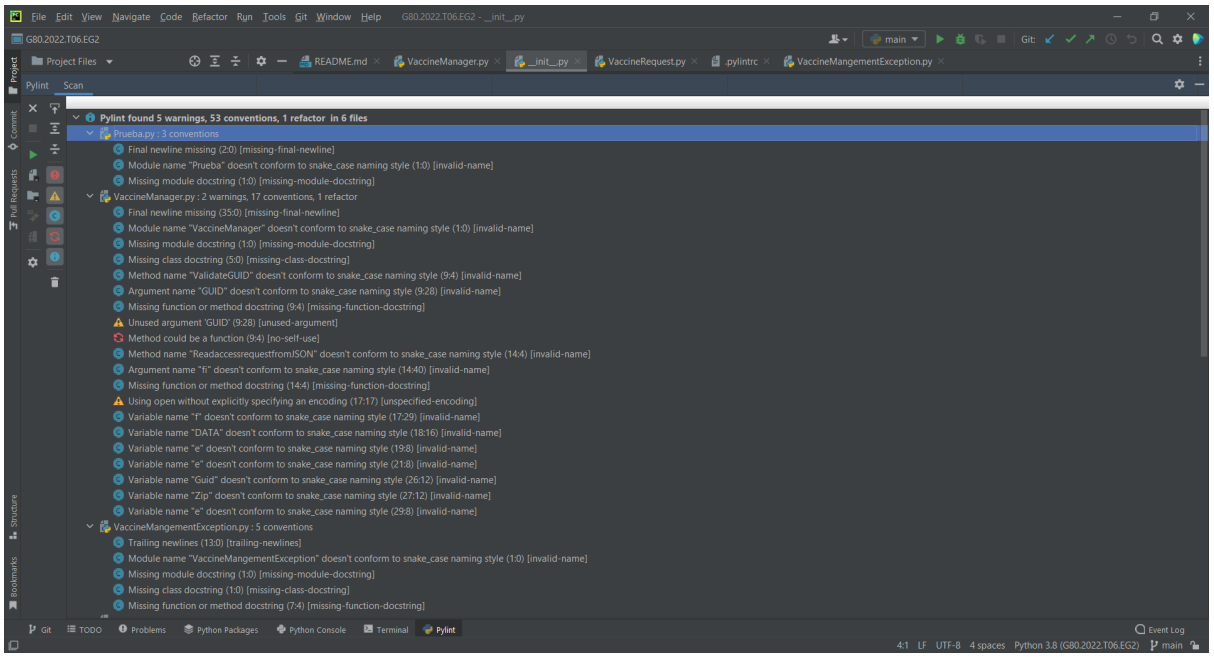
```
-# Maximum number of characters on a single line.
max-line-length=200
```

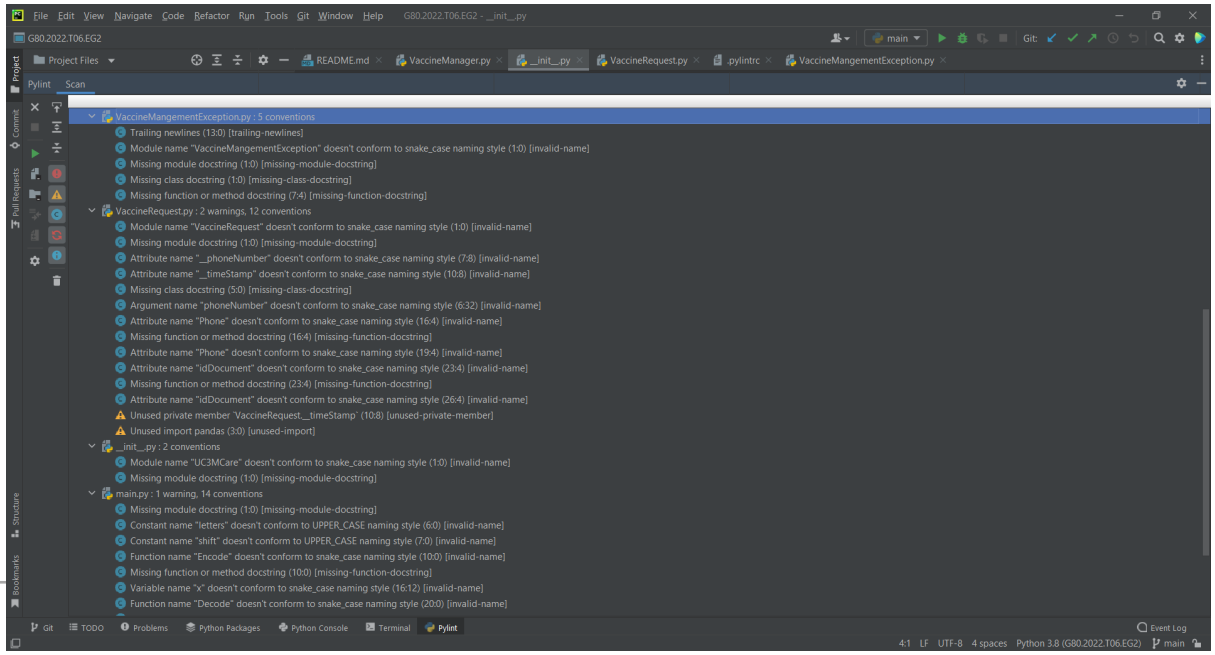
6. Directorio raíz del proyecto

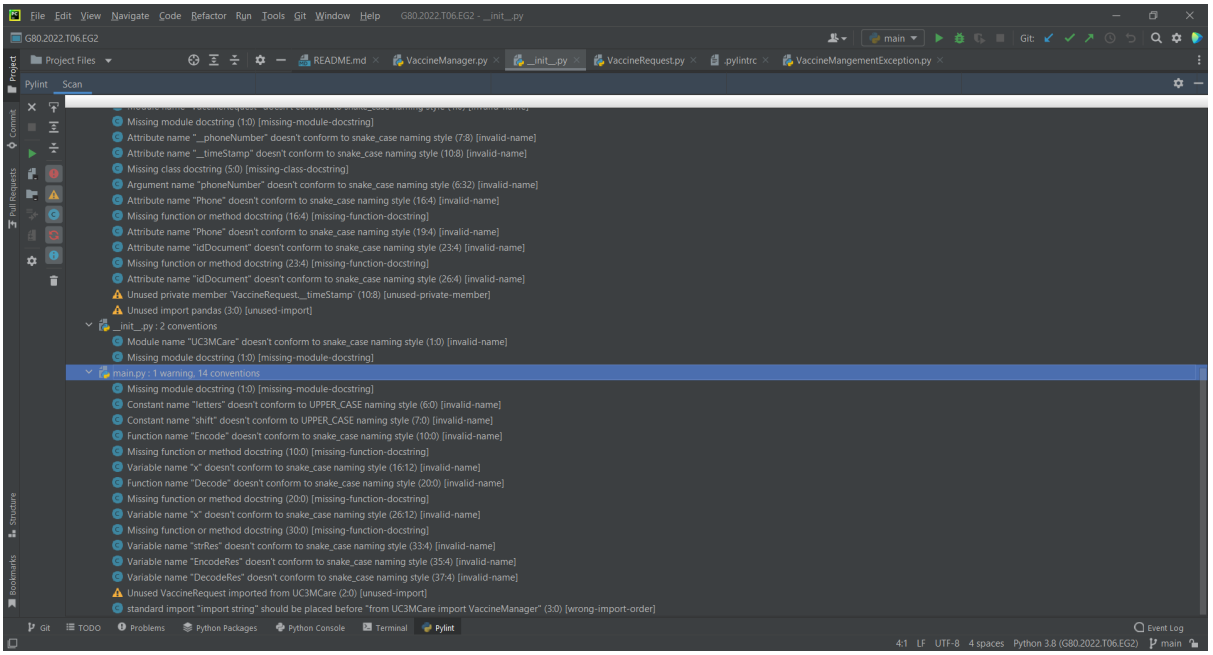
El directorio raíz correspondiente a nuestro proyecto es el siguiente: https://github.com/100451139/G80.2022.T06.EG2.git

7. Capturas de pantalla de los errores de pylintrc

A continuación, se muestran las capturas de pantalla relacionadas con los errores y warnings y una última captura en la que se muestra que todos los errores ya han sido solucionados y no se detecta ninguno más.






File Edit View Navigate Code Refactor Run Tools Git Window Help
G80.2022.T06.EG2 - __init__.py
G80.2022.T06.EG2
main
Git:
Project Files
README.md
VaccineManager.py
__init__.py
VaccineRequest.py
.pylintrc
VaccineMangementException.py
Pylint
Scan
Missing module docstring (1:0) [missing-module-docstring]
Attribute name "__phoneNumber" doesn't conform to snake_case naming style (7:8) [invalid-name]
Attribute name "__timeStamp" doesn't conform to snake_case naming style (10:8) [invalid-name]
Missing class docstring (5:0) [missing-class-docstring]
Argument name "phoneNumber" doesn't conform to snake_case naming style (6:32) [invalid-name]
Attribute name "Phone" doesn't conform to snake_case naming style (16:4) [invalid-name]
Missing function or method docstring (16:4) [missing-function-docstring]
Attribute name "Phone" doesn't conform to snake_case naming style (19:4) [invalid-name]
Attribute name "idDocument" doesn't conform to snake_case naming style (23:4) [invalid-name]
Missing function or method docstring (23:4) [missing-function-docstring]
Attribute name "idDocument" doesn't conform to snake_case naming style (26:4) [invalid-name]
Unused private member `VaccineRequest__timeStamp` (10:8) [unused-private-member]
Unused import pandas (3:0) [unused-import]
__init__.py : 2 conventions
Module name "UC3MCare" doesn't conform to snake_case naming style (1:0) [invalid-name]
Missing module docstring (1:0) [missing-module-docstring]
main.py : 1 warning, 14 conventions
Missing module docstring (1:0) [missing-module-docstring]
Constant name "letters" doesn't conform to UPPER_CASE naming style (6:0) [invalid-name]
Constant name "shift" doesn't conform to UPPER_CASE naming style (7:0) [invalid-name]
Function name "Encode" doesn't conform to snake_case naming style (10:0) [invalid-name]
Missing function or method docstring (10:0) [missing-function-docstring]
Variable name "x" doesn't conform to snake_case naming style (16:12) [invalid-name]
Function name "Decode" doesn't conform to snake_case naming style (20:0) [invalid-name]
Missing function or method docstring (20:0) [missing-function-docstring]
Variable name "x" doesn't conform to snake_case naming style (26:12) [invalid-name]
Missing function or method docstring (30:0) [missing-function-docstring]
Variable name "strRes" doesn't conform to snake_case naming style (33:4) [invalid-name]
Variable name "EncodeRes" doesn't conform to snake_case naming style (35:4) [invalid-name]
Variable name "DecodeRes" doesn't conform to snake_case naming style (37:4) [invalid-name]
Unused VaccineRequest imported from UC3MCare (2:0) [unused-import]
standard import "import string" should be placed before "from UC3MCare import VaccineManager" (3:0) [wrong-import-order]
Git
TODO
Problems
Python Packages
Python Console
Terminal
Pylint
Event Log
4:1 LF UTF-8 4 spaces Python 3.8 (G80.2022.T06.EG2)
main