UNIVERSIDAD
REY JUAN CARLOS

TITULACIÓN EN MAYÚSCULAS

Curso Académico 2020/2021

Trabajo Fin de Carrera/Grado/Máster

TÍTULO DEL TRABAJO EN MAYÚSCULAS

Autor : Nombre del Alumno

Tutor : Dr. Gregorio Robles

Proyecto Fin de Carrera

FIXME: Título

Autor : FIXME

Tutor : Dr. Gregorio Robles Martínez

La defensa del presente Proyecto Fin de Carrera se realizó el día de de 20XX, siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de de 20XX

Dedicado a

mi familia / mi abuelo / mi abuela

II

Agradecimientos

Aquí vienen los agradecimientos... Aunque está bien acordarse de la pareja, no hay que olvidarse de dar las gracias a tu madre, que aunque a veces no lo parezca disfrutará tanto de tus logros como tú... Además, la pareja quizás no sea para siempre, pero tu madre sí.

Resumen

Este proyecto pretende realizar un estudio sobre la mala interpretación, o el mal uso, de una de las reglas de estilo que contiene el pep8[1]. La regla en cuestión es la siguiente: "**Máxima longitud de las líneas:** Limita todas las líneas a un máximo de 79 caracteres". Esta regla ha producido un pequeño debate en la comunidad Python, consistente en saber si se está dejando un peor código, un código más ilegible, debido a una mala práctica de los desarrolladores a la hora de "forzar" su código a que cumpla dicha regla del pep8. En el estudio realizado en este proyecto intentamos detectar las líneas de código que han sufrido esta mala práctica por parte de los desarrolladores, la cual consiste en cambiar el nombre de variables, dándoles un nuevo nombre más corto, para cuadrar la longitud de la línea a 79 caracteres. Esto produce que el nuevo nombre de la variable carezca de sentido alguno, lo que hace que el código sea totalmente ilegible.
Si esto ocurre nos encontramos ante un problema, ya que la regla de estilo de máxima longitud está produciendo un efecto contrario al que pretendía.

Para realizar este estudio se ha implementado el lado del servidor con Node.js (utilizando Express) y lenguaje javascript. Con la idea de dejar preparado un servidor por si en un futuro se quiere envolver esta idea en una aplicación web, sólo tener que realizar la parte del cliente en javascript y conectarlo de forma fácil al servidor con Express.

[1] `http://https://pythonwiki.wikispaces.com/pep8`

Summary

Here comes a translation of the “Resumen” into English. Please, double check it for correct grammar and spelling. As it is the translation of the “Resumen”, which is supposed to be written at the end, this as well should be filled out just before submitting.

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1. Motivación

Cuando terminas de estudiar y por fin empiezas a dedicarte profesionalmente al desarrollo de software, rápidamente te das cuenta de una cosa, la cual nunca, o casi nunca, habías pensado en tu vida de estudiante. En tu trabajo pasas más tiempo leyendo código de otra persona que tu propio código. Esto hace que repentinamente, empiecen a cobrar sentido las frases que tanto te repetían tus profesores sobre estructurar bien tu código o elegir nombres "inteligentes" para las variables o funciones.

Coincidiendo con el inicio de mi etapa profesional, cuando verdaderamente aprendo la importancia de escribir un código de calidad, y no sólo "algo que funcione", mi tutor, Gregorio me habla sobre un vídeo[1] en el que Raymond Hettinger expone buenas prácticas para escribir un código inteligente y de calidad. En este vídeo se plantea, lo que para mi, es una interesante cuestión, la cual, podemos resumir en la siguiente pregunta: ¿Podemos dejar un peor código al realizar un cambio para cumplir una regla del pep8?.

Dicha cuestión es la que vamos a estudiar en este TFG, sin embargo, nos centraremos en un caso muy concreto. La regla en cuestión es la siguiente: "Máxima longitud de las líneas: Limita todas las líneas a un máximo de 79 caracteres" y el caso que nos preocupa es que el desarrollador se encuentre ante una línea de código mayor a 80 caracteres y decida cambiarla haciendo más corto el nombre de alguna variable para cuadrar el tamaño a 79 caracteres, de forma que el nombre de la variable, después del cambio, carezca de ningún sentido.

[1] `https://www.youtube.com/watch?v=wf-BqAjZb8M`

Este hecho me parece digno de estudio, por varios motivos, uno de ellos, indudablemente es el debate que puede crear en la comunidad Python sobre si un código que cumple el pep8 puede ser de peor calidad que uno que no lo cumpla, debido a esta cuestión.
Pero siendo sincero, el motivo por el cual decidí realizar el proyecto sobre este tema fue que, con mi experiencia como estudiante, no pude evitar imaginar el siguiente pensamiento por parte de cualquiera de nosotros: “He terminado mi práctica de python, voy a pasar a mi código un filtro que me he descargado del pep8 para entregar un buen código, lo más importante es que pase este filtro, por tanto, si me salta algún error haré cualquier cambio rápido y sin pensar mucho sólo para forzar a que el filtro pase correctamente”.

Creo firmemente que como estudiantes no somos capaces de asimilar la importancia de escribir un código de calidad, y de las horas y horas que ahorraremos a nuestros compañeros de trabajo, o a nosotros mismo cuando retomemos nuestro propio código de hace tiempo. Son cosas simples, pero es importante que desde el principio de nuestra formación se cojan estas buenas costumbres.

Capítulo 2

Objetivos

2.1. Objetivo general

labelsec:objetivo-general

2.2. Objetivos específicos

labelsec:objetivos-especificos

2.3. Planificación temporal

labelsec:planificacion-temporal

Capítulo 3

Estado del arte

Descripción de las tecnologías que utilizas en tu trabajo. Con dos o tres párrafos por cada tecnología, vale.

Puedes citar libros, como el de Bonabeau et al. sobre procesos estigmérgicos [1].

También existe la posibilidad de poner notas al pie de página, por ejemplo, una para indicarte que visite la página de LibreSoft[1].

3.1. Sección 1

[1] `http://www.libresoft.es`

Capítulo 4

Diseño e implementación

4.1. Arquitectura general

figura 4.1.

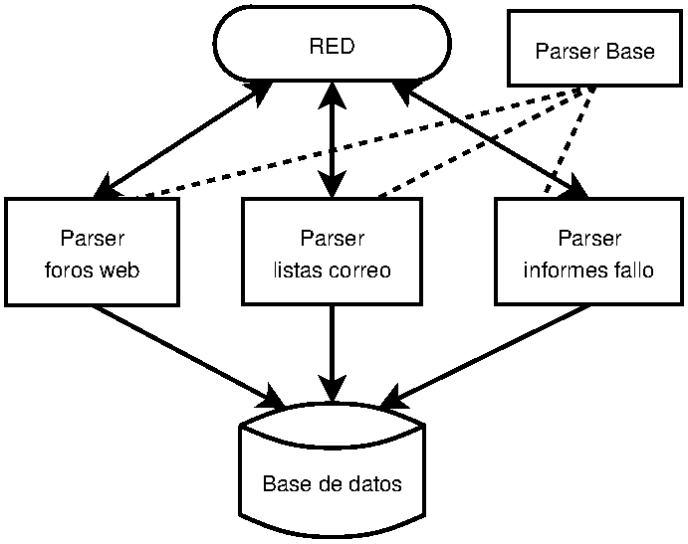

Figura 4.1: Estructura del parser básico

Capítulo 5

Resultados

Capítulo 6

Conclusiones

6.1. Consecución de objetivos

Esta sección es la sección espejo de las dos primeras del capítulo de objetivos, donde se planteaba el objetivo general y se elaboraban los específicos.

Es aquí donde hay que debatir qué se ha conseguido y qué no. Cuando algo no se ha conseguido, se ha de justificar, en términos de qué problemas se han encontrado y qué medidas se han tomado para mitigar esos problemas.

6.2. Aplicación de lo aprendido

Aquí viene lo que has aprendido durante el Grado/Máster y que has aplicado en el TFG/TFM. Una buena idea es poner las asignaturas más relacionadas y comentar en un párrafo los conocimientos y habilidades puestos en práctica.

1. a
2. b

6.3. Lecciones aprendidas

Aquí viene lo que has aprendido en el Trabajo Fin de Grado/Máster.

1. a

2. b

6.4. Trabajos futuros

Ningún software se termina, así que aquí vienen ideas y funcionalidades que estaría bien tener implementadas en el futuro.

Es un apartado que sirve para dar ideas de cara a futuros TFGs/TFMs.

6.5. Valoración personal

Finalmente (y de manera opcional), hay gente que se anima a dar su punto de vista sobre el proyecto, lo que ha aprendido, lo que le gustaría haber aprendido, las tecnologías utilizadas y demás.

Apéndice A

Manual de usuario

Bibliografía

[1] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Articial Systems*. Oxford University Press, Inc., 1999.