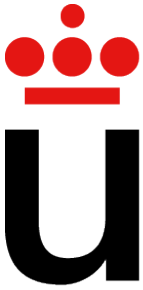

Universidad
Rey Juan Carlos

GRADO EN INGENIERÍA EN SISTEMAS AUDIOVISUALES Y MULTIMEDIA

Curso Académico 2022/2023

Trabajo Fin de Grado

EVALUACIÓN SISTEMÁTICA DE PROYECTOS DE SOFTWARE LIBRE

Autor : Iván Miguel Molinero

Tutor : Dr. Gregorio Robles Martínez

Trabajo Fin de Grado/Máster

Evaluación Sistemática de Proyectos de Software Libre

Autor : Iván Miguel Molinero

Tutor : Dr. Gregorio Robles Martínez

La defensa del presente Proyecto Fin de Carrera se realizó el día de de 202X, siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de de 202X

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

Aquí viene un resumen del proyecto. Ha de constar de tres o cuatro párrafos, donde se presente de manera clara y concisa de qué va el proyecto. Han de quedar respondidas las siguientes preguntas:

- ¿De qué va este proyecto? ¿Cuál es su objetivo principal?
- ¿Cómo se ha realizado? ¿Qué tecnologías están involucradas?
- ¿En qué contexto se ha realizado el proyecto? ¿Es un proyecto dentro de un marco general?

Lo mejor es escribir el resumen al final.

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

Actualmente se comparten muchos proyectos de software libre en Internet en general y en Github [2][1] en particular. Algunos ejemplos de proyectos que se comparten son paquetes de Python que ayudan a desarrolladores a no tener que reinventar la rueda", es decir, que si necesitas hacer un proyecto que lea URLs en uno de sus pasos (por ejemplo) y alguien ya ha hecho un paquete para ello, debes usar ese paquete para poder avanzar en tu proyecto de una forma más rápida y eficaz.

Un proyecto de software libre es un desarrollo que es público; los demás desarrolladores pueden verlo y mejorarlo a través de Pull Request y también descargarlo gratuitamente para su uso por eso muchos usuarios lo prefieren a proyectos de código cerrado.

Ahora bien, en Github se comparten infinidad de proyectos de software libre y no siempre es fácil distinguir entre los que son buenos y los que no lo son tanto: ¿cuál es más seguro? o ¿cuál es más eficiente? Para resolver estas y más preguntas es que, gracias al paquete PyGithub[2], he desarrollado esta aplicación web que permite al usuario introducir el nombre de un repositorio y analizarlo según los parámetros que más le convengan.

El proyecto[5] nace como una extensión de OpenBRR[3] que es una forma de analizar proyectos en función a los siguientes 6 parámetros: funcionalidad, calidad, soporte, comunidad, adopción y usabilidad. A estos parámetros se les asigna un peso y a partir de ahí, da una calificación final sobre 5 del proyecto. A parte de OpenBRR existen otras formas de analizar

[1] `https://github.com/`

[2] `https://github.com/pygithub/pygithub`

[3] `https://link.springer.com/chapter/10.1007/978-3-642-13244-5_18`

proyectos como OSMM, OpenBQR o MOSST entre muchos otros pero me decanté por esta opción al ser la que más se centraba en analizar los proyectos de software libre.

Esta herramienta ayudará a docentes y desarrolladores a distinguir entre los miles de repositorios de Github cuáles son los que merecen la pena agilizando mucho la búsqueda. Pongamos un ejemplo práctico para que se entienda mejor: un desarrollador busca un paquete que realice una función específica para su proyecto pero si tiene que buscar entre las miles de opciones cuál es la mejor, tardaría un tiempo excesivo ya que tendría que analizar cada código por si mismo (con todo lo que ello supone como entenderlo si no está bien comentado), buscar vulnerabilidades, descargarlo, probarlo , etc. En cambio, gracias a este proyecto podrá buscar su mejor opción configurando el análisis según le convenga pudiendo poner el foco solo en seguridad, en seguridad y funcionalidad o en los 6 apartados que tiene OpenBRR.

1.1. Estructura de la memoria

- Capítulo 1: se explica brevemente el origen del proyecto, sus bases y funcionamiento .
- Capítulo 2: se muestran los objetivos del proyecto.
- A continuación se presenta el estado del arte en el capítulo 3.
- ...

Capítulo 2

Objetivos

2.1. Objetivo general

Mi trabajo de fin de grado consiste en crear una aplicación web que analice repositorios de Github para darles una calificación sobre 5 basándose en los parámetros de OpenBRR.

2.2. Objetivos específicos

A la hora de realizar este proyecto, he abordado los siguientes objetivos específicos:

- Estudiar, aprender y utilizar Django[1].
- Estudiar qué es OpenBRR y saber cómo analizar un código para poder calificar cada uno de sus parámetros.
- Analizar el paquete PyGithub en busca de las funciones que necesitaba para analizar repositorios.
- Rellenar y enviar formularios con Django.
- Manipular los datos enviados por el usuario y enviar los resultados.
- Probar el proyecto tanto con repositorios vacíos, como con repositorios teóricamente buenos para comprobar si los resultados eran coherentes.

[1] `https://www.djangoproject.com/`

- Solucionar errores como que el usuario introduzca un repositorio que no exista.
- Introducir mejoras tales como permitir al usuario introducir una dirección de correo electrónico para que le lleguen los resultados ahí también.
- Probar aún más repositorios haciendo que la aplicación se centrase en diferentes parámetros para comprobar que hacía un correcto análisis.

2.3. Planificación temporal

El proyecto se empezó a pensar a finales de Febrero de 2022 pero no fue hasta el verano de ese año que se empezó a trabajar en serio en él ya que anteriormente tenía que compaginarlo con las prácticas externas y no disponía del tiempo necesario. De Marzo a Junio de 2022 dedicaba las mañanas de los sábados y domingos a investigar qué era OpenBRR, en qué consistía y cómo podía usarlo para este proyecto. Asimismo mi tutor me recomendó usar el paquete Perceval[2] de Python para analizar los repositorios de GitHub pero tras investigar por mi cuenta di con el paquete PyGithub el cual analizaba los repositorios de una forma más rápida y cómoda para el desarrollador ya que únicamente con introducir el nombre del repositorio es capaz de sacar todos los datos necesarios.

Durante el verano, pasaba las mañanas de lunes a viernes trabajando en el TFG y conseguí entender el funcionamiento completo de PyGithub así como analizar todos los datos que me devolvía y relacionarlos con los 6 parámetros de OpenBRR intentando que cada dato perteneciera al parámetro que más le correspondiera.

En el mes de Septiembre decidí implementar Django en la aplicación para ser capaz de leer, manipular y crear páginas web a través de las peticiones del usuario. No tenía conocimientos sobre esta tecnología así que usé los tutoriales de DjangoGirls[3][1] para aprender a utilizarla dedicando 3 tardes de lunes a viernes y las mañanas de los sábados y domingos.

Finalmente, de Octubre de 2022 a Enero de 2023, empecé con el desarrollo de la aplicación web dedicando el mismo reparto de tiempo que para aprender Django ya que tenía clases de la universidad por las mañanas salvo en las vacaciones de Navidad que le dediqué todas las mañanas sin excepción. Durante el desarrollo surgieron varias dudas como qué datos asignar a

[2] `https://perceval.readthedocs.io/en/latest/perceval/git.html`
[3] `https://djangogirls.org/es/`

qué parámetros y, sobretodo, qué peso asignarle a cada dato pero tras varias reuniones con mi tutor decidimos que el usuario pudiera manejar los pesos a su antojo para así poder centrarse en analizar los repositorios de la forma que más le convenga. A finales de Diciembre comencé a hacer pruebas con repositorios profesionales como el propio de PyGithub o los de myTeachingURJC[4] (los cuales son utilizados por los profesores de la universidad para compartir contenido de sus asignaturas) y con repositorios de peor calidad como mis primeros repositorios en busca de evaluaciones coherentes y algún error. Para principios de Enero la aplicación ya era funcional pero no tenía ningún tipo de estilo así que dediqué el resto del tiempo a personalizar su aspecto, así como a corregir y controlar errores que surgían durante las diferentes pruebas que realizaba. Por tanto, dediqué las mañanas de finales de Enero y todo Febrero para la realización de la memoria con el objetivo de poder presentar el proyecto en Marzo.

[4] `https://github.com/myTeachingURJC`

Capítulo 3

Estado del arte

Para la realización de este proyecto se ha utilizado Python, PyGithub, Github, Django, HTML, CSS, Javascript y OpenBRR.

3.1. Python

Python[4] es un lenguaje de programación interpretado que le da importancia a la legibilidad del código teniendo una sintaxis muy limpia. Está orientado a objetos y permite ser un buen lenguaje para aprender a programar ya que tiene una curva de aprendizaje muy suave qracias a su sintaxis muy cercana al lenguaje humano. En un lenguaje de código abierto, lo que le permite estar en constante mejora por parte de los usuarios y ser adaptado para diferentes usos como el Machine Learning.

Python fue creado por Guido van Rossum en 1991 mientras trabajaba en el sistema operativo Amoeba[1] teniendo como objetivo controlar excepciones y manejar interfaces de dicho sistema operativo. A finales del año 2000 nació Python 2.0 el cual soportaba Unicode por primera vez y empezó a ser desarrollado por la comunidad bajo la mirada de Guido. Finalmente, en 2008 se lanzó Python 3.0 siendo la versión más completa de Python pero incompatible con las anteriores por lo que muchas características se implementaron también en Python 2.6.

La filosofía de Python se basa en los siguientes puntos para hacer la vida más fácil al desarrollador:

- Hermoso es mejor que feo.

[1] `https://www.cs.vu.nl/pub/amoeba/`

- Explícito es mejor que implícito.
- Simple es mejor que complejo.
- Plano es mejor que anidado.
- Disperso es mejor que denso.
- El código legible cuenta.
- Casos especiales no son lo suficientemente especiales para romper las reglas.
- Casi siempre lo práctico vence a lo formal
- Los errores no deben pasar nunca desapercibidos, a menos que se especifique este comportamiento.
- Ante una ambigüedad, descarte la tentación a adivinar.
- Debe haber una, y preferentemente una sola, manera obvia de lograr. algo, aunque esta generalmente no está clara a primera vista a menos que seas un genio.
- Ahora es mejor que nunca, aunque en muchas ocasiones nunca es mejor que ahora mismo.
- Si la implementación es difícil de explicar, entonces es una mala idea.
- Si la implementación es fácil de explicar, entonces pudiera ser una buena idea.
- Los espacios de nombre son una buena idea, hagamos más de eso.

Para finalizar, aquí podemos ver un ejemplo de un fragmento de código en Python correspondiente a un desarrollo que hice por mi cuenta[2] :

```
class MiCuchillo(Sprite):
  def __init__(self, image, dir, x, y):
      super().__init__(image)
      self.position = (x, y)
      self.direccion = dir
      self.cshape = AARectShape(self.position, self.width/5, self.height/5)
```

[2] https://github.com/ivanmiguelmolinero/Python2D

```
def update(self, dt):
    coor_cuchillo = manejador_scroll.world_to_screen(self.x, self.y)[0]
    if coor_cuchillo > 1280 or coor_cuchillo < 0:
        self.kill()
    if self.direccion == 'd':
        self.position += Vector2(20, 0)
    if self.direccion == 'i':
        self.position -= Vector2(20, 0)
    self.cshape.center = Vector2(self.position[0], self.position[1])
```

3.2. PyGithub

PyGithub[3] se trata de la librería utilizada en la aplicación para analizar los datos del repositorio. Adicionalmente, también permite administrar recursos de Github como repositorios, perfiles de usuario y organizaciones. Es una librería de código abierto lo que permite que esté en constante mejora y desarrollo por parte de la comunidad.

Algunas funciones de interés son:

3.2.1. getrepository()

Esta función permite obtener todos los datos del repositorio únicamente pasándole el nombre del repositorio. Ejemplo de uso sacado directamente de este proyecto:

```
# Obtenemos el repositorio introducido por el usuario
  repo = get_repository(request.GET['text'])
```

3.2.2. getwiki()

Esta función es un ejemplo de los datos que nos permite analizar PyGithub. En concreto, “getwiki()´´ nos devuelve si el repositorio en cuestión tiene wiki o no. Otros ejemplos de funciones de este tipo son “repo.homepage´´, “repo.subscriberscount´´, “repo.getcommits().totalCount´´...

Capítulo 4

Diseño e implementación

Aquí viene todo lo que has hecho tú (tecnológicamente). Puedes entrar hasta el detalle. Es la parte más importante de la memoria, porque describe lo que has hecho tú. Eso sí, normalmente aconsejo no poner código, sino diagramas.

4.1. Arquitectura general

Si tu proyecto es un software, siempre es bueno poner la arquitectura (que es cómo se estructura tu programa a "vista de pájaro").

Por ejemplo, puedes verlo en la figura 4.1. LaTeX pone las figuras donde mejor cuadran. Y eso quiere decir que quizás no lo haga donde lo hemos puesto...Eso no es malo. A veces queda un poco raro, pero es la filosofía de LaTeX: tú al contenido, que yo me encargo de la maquetación.

Recuerda que toda figura que añadas a tu memoria debe ser explicada. Sí, aunque te parezca evidente lo que se ve en la figura 4.1, la figura en sí solamente es un apoyo a tu texto. Así que explica lo que se ve en la figura, haciendo referencia a la misma tal y como ves aquí. Por ejemplo: En la figura 4.1 se puede ver que la estructura del *parser* básico, que consta de seis componentes diferentes: los datos se obtienen de la red, y según el tipo de dato, se pasará a un *parser* específico y bla, bla, bla...

Si utilizas una base de datos, no te olvides de incluir también un diagrama de entidad-relación.

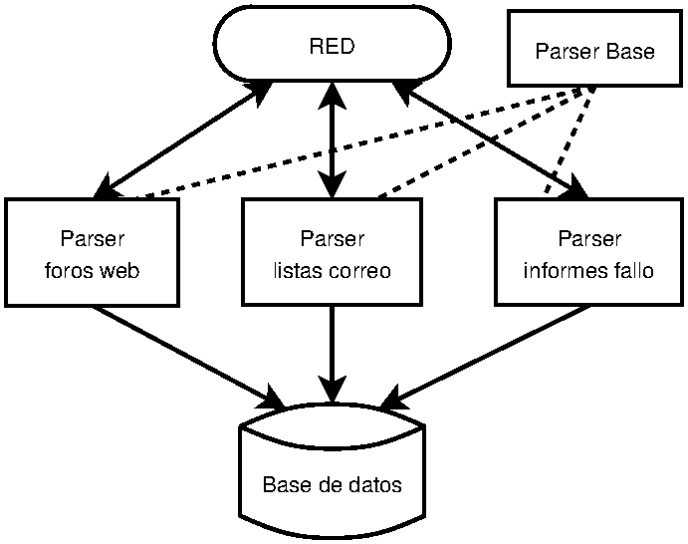

Figura 4.1: Estructura del parser básico.

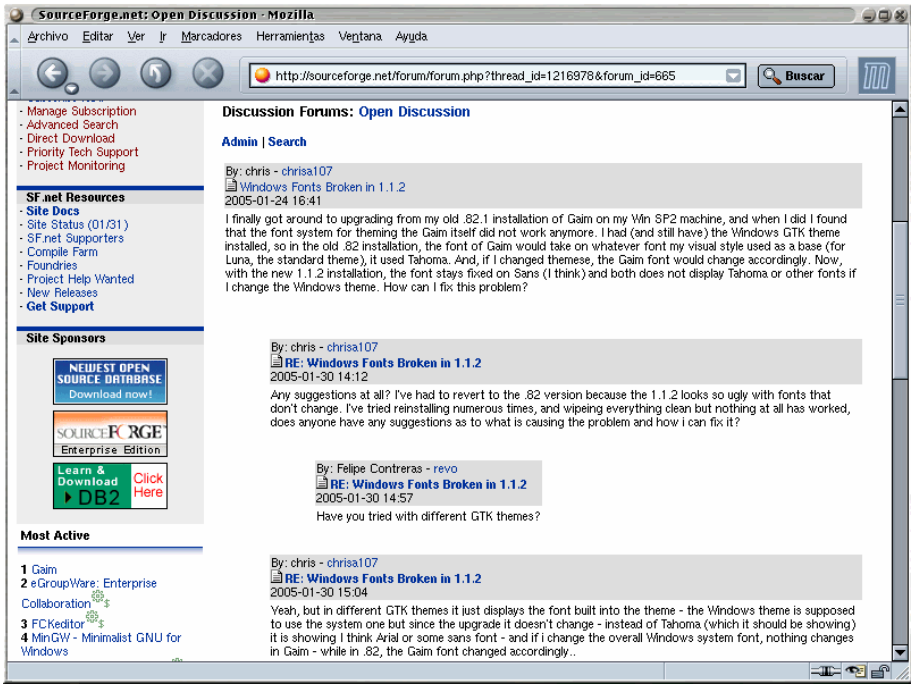

Figura 4.2: Página con enlaces a hilos

Capítulo 5

Experimentos y validación

Este capítulo se introdujo como requisito en 2019. Describe los experimentos y casos de test que tuviste que implementar para validar tus resultados. Incluye también los resultados de validación que permiten afirmar que tus resultados son correctos.

Capítulo 6

Resultados

En este capítulo se incluyen los resultados de tu trabajo fin de grado.

Si es una herramienta de análisis lo que has realizado, aquí puedes poner ejemplos de haberla utilizado para que se vea su utilidad.

Capítulo 7

Conclusiones

7.1. 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.

Y si has llegado hasta aquí, siempre es bueno pasarle el corrector ortográfico, que las erratas quedan fatal en la memoria final. Para eso, en Linux tenemos aspell, que se ejecuta de la siguiente manera desde la línea de *shell*:

```
aspell --lang=es_ES -c memoria.tex
```

7.2. 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

7.3. Lecciones aprendidas

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

1. Aquí viene uno.
2. Aquí viene otro.

7.4. Trabajos futuros

Ningún proyecto ni 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.

Apéndice A

Manual de usuario

Esto es un apéndice. Si has creado una aplicación, siempre viene bien tener un manual de usuario. Pues ponlo aquí.

Bibliografía

[1] Djangogirls.
https://djangogirls.org/es/.

[2] Github.
https://docs.github.com/es.

[3] Pygithub.
https://github.com/pygithub/pygithub.

[4] Python.
https://web.archive.org/web/20200224120525/https://luca-d3.com/es/data-speaks/diccionario-tecnologico/python-lenguaje.

[5] Repositorio de mi trabajo de fin de grado.
https://github.com/ivanmiguelmolinero/TFG.