



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*

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

1. Introducción	1
1.1. Estructura de la memoria	2
2. Objetivos	3
2.1. Objetivo general	3
2.2. Objetivos específicos	3
2.3. Planificación temporal	4
3. Estado del arte	5
3.1. Sección 1	6
4. Diseño e implementación	7
4.1. Arquitectura general	7
5. Experimentos y validación	9
6. Resultados	11
7. Conclusiones	13
7.1. Consecución de objetivos	13
7.2. Aplicación de lo aprendido	13
7.3. Lecciones aprendidas	14
7.4. Trabajos futuros	14
A. Manual de usuario	15
Bibliografía	17

Índice de figuras

4.1. Estructura del parser básico.	8
4.2. Página con enlaces a hilos	8

Capítulo 1

Introducción

Actualmente se comparten muchos proyectos de software libre en Internet en general y en Github [1]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 PyGithub², 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 nace como una extensión de OpenBRR³ 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 proyectos como

¹<https://github.com/>

²<https://github.com/pygithub/pygithub>

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

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 Django¹.
- 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 centrara en diferentes parámetros para comprobar que hacía un correcto análisis.

2.3. Planificación temporal

A mí me gusta que aquí pongáis una descripción de lo que os ha llevado realizar el trabajo. Hay gente que añade un diagrama de GANTT. Lo importante es que quede claro cuánto tiempo llevas (tiempo natural, p.ej., 6 meses) y a qué nivel de esfuerzo (p.ej., principalmente los fines de semana).

Capítulo 3

Estado del arte

Descripción de las tecnologías que utilizas en tu trabajo. Con dos o tres párrafos por cada tecnología, vale. Se supone que aquí viene todo lo que no has hecho tú.

Puedes citar libros, como el de Bonabeau et al., sobre procesos estigmérgicos [?]. Me encantan los procesos estigmérgicos. Deberías leer más sobre ellos. Pero quizás no ahora, que tenemos que terminar la memoria para sacarnos por fin el título. Nota que el ~ añade un espacio en blanco, pero no deja que exista un salto de línea. Imprescindible ponerlo para las citas.

Citar es importantísimo en textos científico-técnicos. Porque no partimos de cero. Es más, partir de cero es de tontos; lo suyo es aprovecharse de lo ya existente para construir encima y hacer cosas más sofisticadas. ¿Dónde puedo encontrar textos científicos que referenciar? Un buen sitio es Google Scholar¹. Por ejemplo, si buscas por “stigmergy libre software” para encontrar trabajo sobre software libre y el concepto de *estigmergia* (¿te he comentado que me gusta el concepto de estigmergia ya?), encontrarás un artículo que escribí hace tiempo cuyo título es “Self-organized development in libre software: a model based on the stigmergy concept”. Si pulsas sobre las comillas dobles (entre la estrella y el “citado por ...”, justo debajo del extracto del resumen del artículo, te saldrá una ventana emergente con cómo citar. Abajo a la derecha, aparece un enlace BibTeX. Púlsalo y encontrarás la referencia en formato BibTeX, tal que así:

```
@inproceedings{robles2005self,  
  title={Self-organized development in libre software:  
    a model based on the stigmergy concept},  
  author={Robles, Gregorio and Merelo, Juan Juli\`an
```

¹<http://scholar.google.com>

Uno	2	3
Cuatro	5	6
Siete	8	9

Cuadro 3.1: Ejemplo de tabla. Aquí viene una pequeña descripción (el *caption*) del contenido de la tabla. Si la tabla no es autoexplicativa, siempre viene bien aclararla aquí.

```

and Gonz\'alez-Barahona, Jes\'us M.},
booktitle={ProSim'05},
year={2005}
}

```

Copia el texto en BibTeX y pégalo en el fichero `memoria.bib`, que es donde están las referencias bibliográficas. Para incluir la referencia en el texto de la memoria, deberás citarlo, como hemos hecho antes con, lo que pasa es que en vez de el identificador de la cita anterior (`bonabeau:swarm`), tendrás que poner el nuevo (`robles2005self`). Compila el fichero `memoria.tex` (`pdflatex memoria.tex`), añade la bibliografía (`bibtex memoria.aux`) y vuelve a compilar `memoria.tex` (`pdflatex memoria.tex`)...y *voilà* ¡tenemos una nueva cita

También existe la posibilidad de poner notas al pie de página, por ejemplo, una para indicarte que visite la página del GSyc².

3.1. Sección 1

Hemos hablado de cómo incluir figuras. Pero no hemos dicho nada de tablas. A mí me gustan las tablas. Mucho. Aquí un ejemplo de tabla, la Tabla 3.1 (siento ser pesado, pero nota cómo he puesto la referencia).

²<http://gsyc.es>

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.

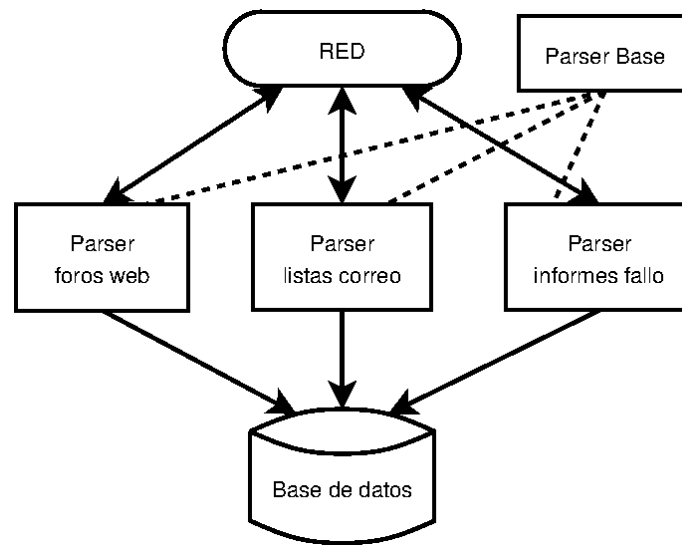


Figura 4.1: Estructura del parser básico.

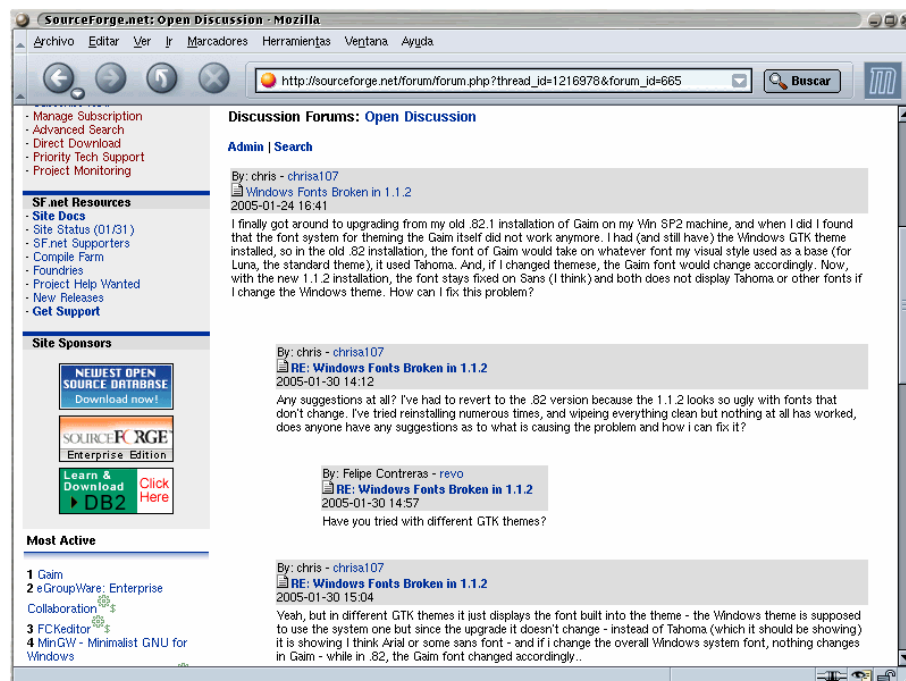


Figura 4.2: Página con enlaces a hilos

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] Github.

<https://docs.github.com/es>.