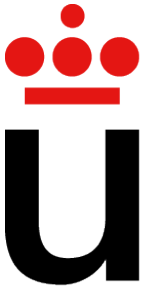

Universidad
Rey Juan Carlos

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE LA TELECOMUNICACIÓN

Curso Académico 2020/2021

Trabajo Fin de Grado

CREACIÓN AUTOMÁTICA DE PULL-REQUESTS A PARTIR DE RESULTADOS DE PYLINT

Autor : Raúl Cano Montero

Tutor : Dr. Gregorio Robles

Trabajo Fin de Grado

Creación Automática de Pull-Requests a partir de Resultados de Pylint

Autor : Raúl Cano Montero

Tutor : Dr. Gregorio Robles

La defensa del presente Proyecto Fin de Carrera se realizó el día de de 2021, siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de de 2021

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

En este capítulo se introduce el proyecto. Debería tener información general sobre el mismo, dando la información sobre el contexto en el que se ha desarrollado.

No te olvides de echarle un ojo a la página con los cinco errores de escritura más frecuentes[1].

Aconsejo a todo el mundo que mire y se inspire en memorias pasadas. Las memorias de los proyectos que he llevado yo están (casi) todas almacenadas en mi web del GSyC[2].

1.1. Sección

Esto es una sección, que es una estructura menor que un capítulo.

Por cierto, a veces me comentáis que no os compila por las tildes. Eso es un problema de codificación. Al guardar el archivo, guardad la codificación de "ISO-Latin-1" a "UTF-8" (o viceversa) y funcionará.

1.1.1. Estilo

Recomiendo leer los consejos prácticos sobre escribir documentos científicos en LaTeX de Diomidis Spinellis[3].

Lee sobre el uso de las comas[4]. Las comas en español no se ponen al tuntún. Y nunca,

[1] http://www.tallerdeescritores.com/errores-de-escritura-frecuentes

[2] https://gsyc.urjc.es/~grex/pfcs/

[3] https://github.com/dspinellis/latex-advice

[4] http://narrativabreve.com/2015/02/opiniones-de-un-corrector-de-estilo-11-recetas-par
html

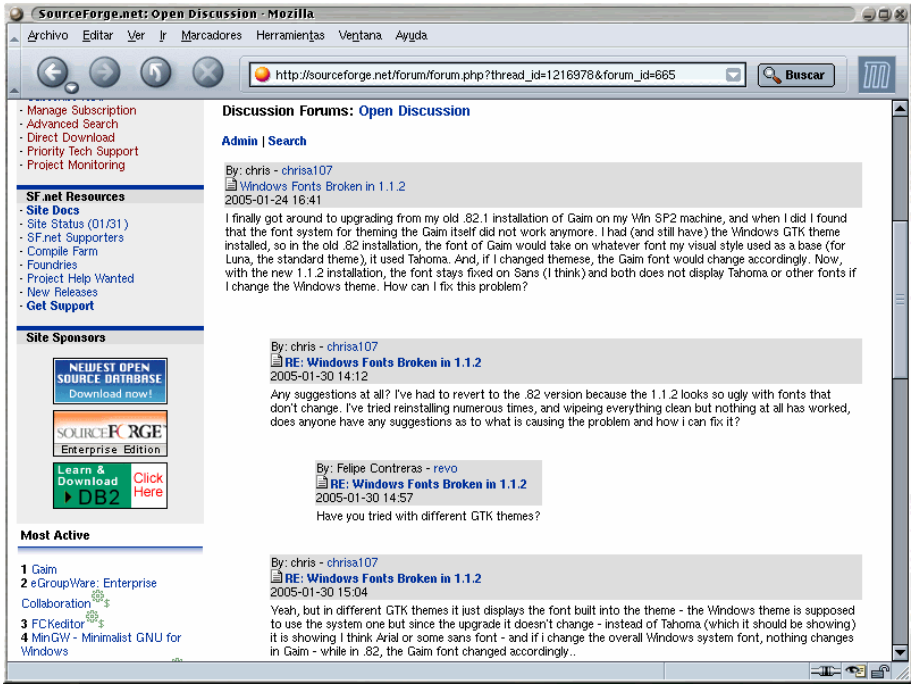

Figura 1.1: Página con enlaces a hilos

nunca entre el sujeto y el predicado (p.ej. en “Yo, hago el TFG” sobre la coma). La coma no debe separar el sujeto del predicado en una oración, pues se cortaría la secuencia natural del discurso. No se considera apropiado el uso de la llamada coma respiratoria o *coma criminal*. Solamente se suele escribir una coma para marcar el lugar que queda cuando omitimos el verbo de una oración, pero es un caso que se da de manera muy infrecuente al escribir un texto científico (p.ej. “El Real Madrid, campeón de Europa”).

A continuación, viene una figura, la Figura 1.1. Observarás que el texto dentro de la referencia es el identificador de la figura (que se corresponden con el “label” dentro de la misma). También habrás tomado nota de cómo se ponen las “comillas dobles” para que se muestren correctamente. Nota que hay unas comillas de inicio (“) y otras de cierre (”), y que son diferentes. Volviendo a las referencias, nota que al compilar, la primera vez se crea un diccionario con las referencias, y en la segunda compilación se “rellenan” estas referencias. Por eso hay que compilar dos veces tu memoria. Si no, no se crearán las referencias.

A continuación un bloque “verbatim”, que se utiliza para mostrar texto tal cual. Se puede utilizar para ofrecer el contenido de correos electrónicos, código, entre otras cosas.

```
From gaurav at gold-solutions.co.uk  Fri Jan 14 14:51:11 2005
From: gaurav at gold-solutions.co.uk (gaurav_gold)
```

```
Date: Fri Jan 14 19:25:51 2005
Subject: [Mailman-Users] mailman issues
Message-ID: <003c01c4fa40$1d99b4c0$94592252@gaurav7klgnyif>

Dear Sir/Madam,
How can people reply to the mailing list?  How do i turn off
this feature? How can i also enable a feature where if someone
replies the newsletter the email gets deleted?
Thanks

From msapiro at value.net  Fri Jan 14 19:48:51 2005
From: msapiro at value.net (Mark Sapiro)
Date: Fri Jan 14 19:49:04 2005
Subject: [Mailman-Users] mailman issues
In-Reply-To: <003c01c4fa40$1d99b4c0$94592252@gaurav7klgnyif>
Message-ID: <PC17302005011410485105780lb04d55@msapiro>

gaurav_gold wrote:
>How can people reply to the mailing list?  How do i turn off
this feature? How can i also enable a feature where if someone
replies the newsletter the email gets deleted?

See the FAQ
>Mailman FAQ: http://www.python.org/cgi-bin/faqw-mm.py
article 3.11
```

1.2. Estructura de la memoria

En esta sección se debería introducir la esctura de la memoria.

Así:

- En el primer capítulo se hace una intro al proyecto.
- En el capítulo 2 (ojo, otra referencia automática) 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

El objetivo de este Trabajo de Fin de Grado es realizar el análisis, mediante la herramienta Pylint, la corrección del código y la creación de Pull-Requests sobre proyectos desarrollados en Python y almacenados en GitHub, para que éstos cumplan con las normas recogidas en la guía de estilo de Python denominada PEP8

2.2. Objetivos específicos

Para poder cumplir con el objetivo general se han tenido en cuenta los siguientes objetivos específicos:

- **Trabajar con proyectos reales.** Utilizar la aplicación para analizar y corregir proyectos reales.
- **Aplicación web.** Desarrollar como una aplicación web para hacer mejor y más sencilla la experiencia de usuario.
- **Accesibilidad desde internet.** Hacer que la aplicación esté en funcionamiento continuamente y sea accesible desde cualquier ubicación.
- **Compatibilidad con el servidor GitLab de la ETSIT.** Hacer que la aplicación sea compatible con el GitLab de la ETSIT para permitir que los alumnos de la escuela puedan

analizar y corregir sus proyectos.

- **Análisis de aceptación de las Pull-Requests.** Evaluar la aceptación de los cambios realizados en los proyectos mediante el análisis del estado de las Pull-Requests realizadas.

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

3.1. Python

Python[1] es un lenguaje de programación Open Source escrito por Guido van Rossum entre finales de los años 80 y principio de los 90. El objetivo de la creación de Python era tener un lenguaje de programación con una sintaxis sencilla, como ABC, añadiendo la posibilidad de realizar llamadas al sistema con compatibilidad con distintos sistemas operativos.[2]

Es un lenguaje de programación interpretado, interactivo y orientado a objetos. Al ser un lenguaje interpretado se reduce el tiempo entre la escritura del código y la ejecución del mismo, ya que no hay que compilarlo cada vez que se realice una modificación del código. Sin embargo, ésto implica que sea necesaria la instalación de un intérprete en la máquina que va a ejecutar el código y aumenta el tiempo de la ejecución respecto a los lenguajes compilados.

Python incorpora módulos, excepciones, tipado dinámico, tipos de datos dinámicos de muy alto nivel y clases. La librería estándar de Python le permite cubrir distintas áreas, como el procesamiento de cadenas de texto, protocolos de red, ingeniería del software e interacción con el sistema operativo.

Además de la programación orientada a objetos, soporta múltiples paradigmas de programación como la programación procedimental y la programación funcional y también es compatible con diversos sistemas operativos, incluyendo Windows y múltiples variantes de Unix, como Linux y macOS.

Actualmente Python es uno de los lenguajes de programación más utilizados.[4]

[1] `https://www.python.org/`

3.2. PEP 8

Python Enhancement Proposal 8, más conocida como PEP 8[2], es una guía de estilo para código escrito en Python que contiene recomendaciones y convenciones dirigidas a mejorar la consistencia y legibilidad del código. Algunas de las convenciones indicadas por PEP 8 son las siguientes:

- **Tabulación**: Se debe tabular utilizando 4 espacios por cada nivel de tabulación.
- **Caracteres por línea**: Cada línea de código debe tener, como máximo, una longitud de 79 caracteres.
- **Imports**: Los módulos adicionales deben ser importados al inicio de cada fichero y sólo debe haber uno por línea.
- **Espacios antes y después de operadores**: Antes y después de cada operador debe haber un único espacio.
- **Nomenclatura**: Cada elemento tiene definidas unas normas distintas al realizar su declaración.

PEP 8 también nos indica que las distintas normas pueden ser ignoradas en determinados casos como, por ejemplo:

- Cuando se añade código a una librería ya existente que sigue un estilo distinto se debe respetar dicho estilo para mantener la consistencia dentro de dicha librería.
- En caso de que el código necesite funcionar en versiones antiguas de Python y modificarlo suponga una incompatibilidad.
- Cuando modificar el código dificulte la lectura y comprensión del mismo.

3.3. Pylint

Pylint[3] es una herramienta de análisis de código escrito en Python que comprueba si hay incumplimiento de la guía de estilo PEP 8. Es un analizador de código estático, lo que facilita

[2] `https://www.python.org/dev/peps/pep-0008/`
[3] `https://www.pylint.org/`

el análisis del código ya que se realiza sin tener que ejecutarlo.

La salida de Pylint muestra para cada error detectado en qué fichero, línea y columna se ha encontrado y, al final de la salida, una calificación del código analizado en función de los errores que se han detectado. El formato de esta salida y las opciones del análisis, como los errores que deben buscarse o los ficheros que deben analizarse, son configurables mediante argumentos introducidos al ejecutar la herramienta o mediante un fichero de configuración llamado pylintrc.

3.4. Django

Django[4] es un framework de desarrollo de aplicaciones web de código abierto escrito en Python. Django está basado en una variación de la arquitectura MVC (Model-View-Controller) a la que denominan MVT (Model-View-Template):

- **Model**: Datos con los que el sistema trabaja. Se definen en el fichero models.py.
- **View**: Describe qué datos se presentan mediante los ficheros views.py y urls.py.
- **Template**: Definido por las plantillas HTML y CSS que describen cómo se presentan los datos. .

3.5. PostgreSQL

PostgreSQL[5] es un sistema de gestión de bases de datos relacionales de código abierto basado en POSTGRES. PostgreSQL es compatible con gran parte del estándar SQL, añade algunas características propias, como el control de concurrencias multiversión o la compatibilidad con consultas complejas, y permite al usuario añadir nuevos tipos de datos, funciones u operadores.

3.6. HTML

HTML[6] (Hypertext Markup Language) es el lenguaje de marcado estándar para la creación de páginas web. Este estándar está a cargo del World Wide Web Consortium (W3C) y es com-

[4] `https://www.djangoproject.com/`
[5] `https://www.postgresql.org/`
[6] `https://www.w3.org/standards/webdesign/htmlcss`

patible con todos los navegadores web. Con HTML se define la estructura de las páginas web mediante la utilización de distintas etiquetas.

3.7. CSS

CSS[7] (Cascading Style Sheets) es un lenguaje de diseño web utilizado para definir la apariencia y el estilo visual de una página web. Al igual que HTML, es un estándar a cargo del World Wide Web Consortium (W3C). CSS establece cómo se muestran los elementos declarados en un documento HTML definiendo propiedades como los colores, fuentes y diseño. Para ello CSS hace uso de reglas, formadas por selectores, que indican sobre qué elementos HTML se aplican, y declaraciones, que definen y modifican las propiedades. CSS es independiente de HTML y se puede llevar a cabo desde un fichero separado, lo que facilita el mantenimiento y la reutilización de las hojas de estilo.

3.8. Bootstrap

Bootstrap[8] es un framework de código abierto dirigido al desarrollo front-end de páginas web. Es una biblioteca que contiene herramientas HTML, CSS y JavaScript que facilitan el diseño y ayudan a realizar modificaciones de manera más sencilla en el estilo visual. Bootstrap contiene plantillas predefinidas con diferentes estilos de botones, formularios, desplegables o pestañas, entre otros componentes. Una de las características más importantes de Bootstrap es que también aporta el diseño responsive, que adapta automáticamente la presentación de la página web a diferentes tipos de dipositivos teniendo en cuenta el tamaño de pantalla y la resolución, lo que facilita la visualización y aumenta la compatibilidad de las páginas web.

3.9. JavaScript

JavaScript[9] es un lenguaje de programación interpretado, orientado a objetos, imperativo, multiparadigma, de tipado débil y dinámico y basado en prototipos. TIene una sintaxis inspirada

[7] `https://www.w3.org/standards/webdesign/htmlcss`
[8] `https://getbootstrap.com/`
[9] `https://developer.mozilla.org/es/docs/Web/JavaScript`

en la de Java y C++, con sentencias que funcionan igual que en esos lenguajes. Su uso principal es el scripting en páginas web dirigido a establecer el comportamiento de las páginas web al suceder un evento determinado. La ejecución de estos scripts se realiza en el lado del cliente, en el navegador web, sin tener acceso al lado del servidor. El uso de JavaScript está ampliamente extendida debido a que es compatible con cualquier navegador y sistema operativo.

3.10. jQuery

jQuery[10] es una librería de JavaScript de código abierto que simplifica la interacción entre JavaScript y HTML. jQuery facilita la manipulación del HTML, el manejo de eventos, el desarrollo de animaciones y el uso de AJAX y es compatible con cualquier navegador web y sistema operativo actual.

3.11. Chart.js

Chart.js[11] es una librería de JavaScript de código abierto dirigida a la representación y visualización de datos. Permite generar gráficos de distintos tipos, como barras, líneas, circulares o de dispersión, entre otros, y personalizar cómo se muestran dichos gráficos. Actualmente es una de las librerías de visualización de datos en JavaScript más populares en GitHub.

3.12. JSON

JSON[12] (JavaScript Object Notation) es un formato de intercambio de datos fácil de leer y escribir para humanos y fácil de generar e interpretar para las máquinas. Como su nombre indica, está basado en la sintaxis de objetos de JavaScript, teniendo una estructura formada por pares de nombre/valor. JSON es independiente de cualquier lenguaje y, por ello, es compatible con la gran mayoría de los lenguajes de programación actuales.

[10] `https://jquery.com/`
[11] `https://www.chartjs.org/`
[12] `https://www.json.org`

3.13. API-REST

Una API (Application Programming Interface) es un conjunto de instrucciones y protocolos ofrecidos por determinadas aplicaciones para dar acceso a su propia información o funciones. La API actúa como intermediario entre la propia aplicación y el solicitante y devuelve a éste último una respuesta con el resultado de la operación realizada o con la información solicitada.

Una API-REST es un tipo de API en el que el acceso está basado en arquitectura REST (Representational State Transfer), por lo que las llamadas a dicha API se llevan a cabo mediante peticiones HTTP y el uso de objetos XML o JSON.

3.14. Git

Git[13] es un software de código abierto de control de versiones diseñado por Linus Torvald para facilitar la gestión y el trabajo en equipo en proyectos de desarrollo de software. Para un proyecto almacenado en un repositorio, Git permite que cada integrante del equipo tenga una versión en local del proyecto y vaya subiendo sus modificaciones al repositorio.

El funcionamiento de Git se basa en la creación, uso y mezcla de ramas dentro del proyecto, existiendo una rama principal, conocida como master, y creándose ramas secundarias para realizar las modificaciones en el código. Una vez se realicen las modificaciones, la rama secundaria se mezcla con la rama master y cuando otros miembros del equipo actualicen su rama master descargarán los cambios realizados sin que esto afecte a su trabajo.

3.15. GitHub

GitHub[14] es un servicio web, propiedad de Microsoft, que almacena repositorios de proyectos de desarrollo software que utilizan el sistema de control de versiones Git.

[13] `https://git-scm.com/`
[14] `https://github.com/`

3.16. GitLab

GitLab[15] es un servicio web de código abierto que almacena repositorios de proyectos de desarrollo software que utilizan el sistema de control de versiones Git.

3.17. Heroku

Heroku[16]

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 [1]. 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 Scholar[17]. 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
          and Gonz\'alez-Barahona, Jes\'us M.},
  booktitle={ProSim'05},
```

[15] https://about.gitlab.com/

[16]

[17] 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.

```
  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 [1], 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 [3]!

También existe la posibilidad de poner notas al pie de página, por ejemplo, una para indicarte que visite la página del GSyC[18].

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 3.17 (siento ser pesado, pero nota cómo he puesto la referencia).

[18] `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.

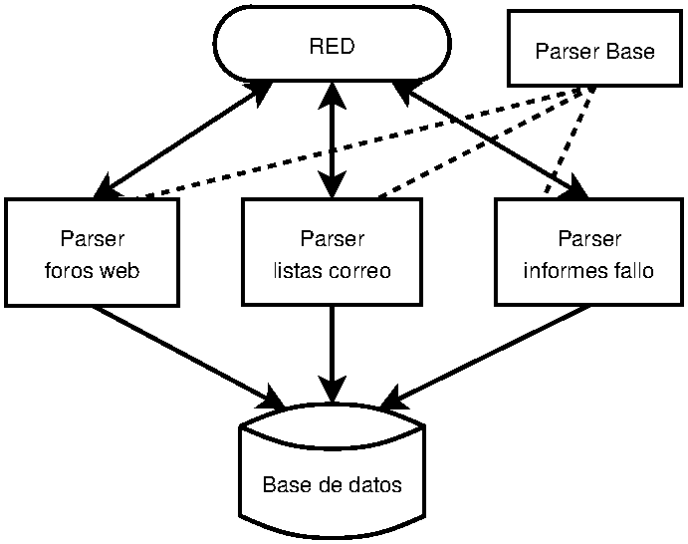

Figura 4.1: Estructura del parser básico.

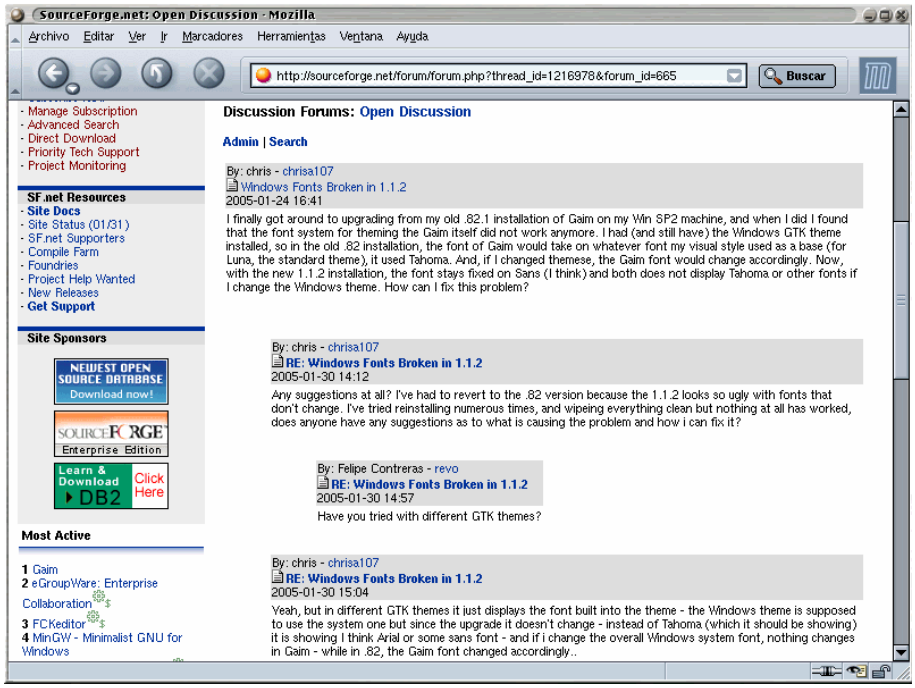

Figura 4.2: Página con enlaces a hilos

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

[2] Python Software Foundation. *Python Documentation: General Python FAQ*, s.f. `https://docs.python.org/3/faq/general.html`.

[3] G. Robles, J. J. Merelo, and J. M. González-Barahona. Self-organized development in libre software: a model based on the stigmergy concept. In *ProSim'05*, 2005.

[4] Stack Exchange, inc. *Stack Overflow Developer Survey 2020*, 2020. `https://insights.stackoverflow.com/survey/2020`.