Universidad
Rey Juan Carlos

ESCUELA DE INGENIERÍA DE FUENLABRADA

GRADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

EVOLUCIÓN DE LA PARTICIPACIÓN VOLUNTARIA DE PROYECTOS DE SOFTWARE LIBRE: EVIDENCIAS DE DEBIAN

Autor : Pablo Cabeza Portalo

Tutor : Dr. Gregorio Robles

Curso académico 2023/2024

Trabajo Fin de Grado

Evolución de la Participación Voluntaria en Proyectos de Software Libre: Evidencia de Debian

Autor : Pablo Cabeza Portalo

Tutor : Dr. Gregorio Robles

La defensa del presente Proyecto Fin de Carrera se realizó el día de de 2024, siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de de 2024

Dedicado a

mi familia, mis amigos y a mi pareja.

II

Agradecimientos

Este es el fin de una de las etapas más importantes de mi vida y por ello quiero agradecer a varias personas que me han acompañado en este proceso. Primero quiero agradecer a mi madre Antonia todo el apoyo que me ha dado en cada una de mis decisiones, tanto personales como académicas. Eres quien me impulsa a lograr mis sueños. Segundo a mi padre Manuel que siempre ha sido mi ejemplo a seguir. Me ha aportado valores fundamentales como el trabajo y el esfuerzo que han sido imprescindibles para abordar esta carrera. Gracias a los dos por darme la vida. También agradecer a mi tutor, Gregorio, por darme este proyecto, tener paciencia y ayudarme en todo cuanto pudo para desarrollarlo. Por último, quiero agradecer a esa persona tan especial que conocí en esta universidad y que me dio la confianza y la fuerza necesaria para lograr mis metas. Paula, gracias por ser mi compañera de viaje y por aparecer en el momento que mas lo necesitaba. Hemos afrontado este desafío juntos y sin ti no hubiera sido igual.

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. Contexto 2

1.1.1. Proyecto Debian 2

1.1.2. Versiones Debian 2

1.2. Estructura de la memoria 4

2. Objetivos **7**

2.1. Objetivo general 7

2.2. Objetivos específicos 7

2.3. Planificación temporal 8

3. Estado del arte **11**

3.1. Python 3.12.0 11

3.2. My sql connector 12

3.3. Matplotlib 13

3.4. Pycharm 13

3.5. SQL 14

3.6. MySQL WorkBench 15

4. Diseño e implementación **17**

4.1. Arquitectura general 17

5. Experimentos y validación **19**

6. Resultados **21**

7. Conclusiones **23**

7.1. Consecución de objetivos 23

7.2. Aplicación de lo aprendido 23

7.3. Lecciones aprendidas 24

7.4. Trabajos futuros 24

A. Manual de usuario **25**

Bibliografía **27**

Índice de figuras

2.1. Diagrama de Gantt 9

3.1. Lenguajes de programación más populares del mundo 12
3.2. Ejemplos gráficas Matplotlib 13
3.3. Pycharm 14
3.4. SQL Workbench 15

4.1. Fases del tratamiento de datos 18

Capítulo 1

Introducción

Vivimos en un mundo totalmente digitalizado en el que la presencia de ordenadores está a la orden del día. Con esto no nos referimos únicamente a ordenadores de escritorio. A diario interactuamos con una amplia gama de dispositivos compuestos por ordenadores camuflados. Desde electrodomésticos inteligentes pasando por coches modernos hasta en tarjetas de crédito. Estos tienen incorporados ordenadores pequeños pero potentes que realizan una serie de tareas para el beneficio y la mejora de la vida humana.

En la mayoría de estos ordenadores de uso cotidiano, como portátiles o móviles, se aloja un Sistema Operativo. Un Sistema Operativo es el "intermediario" entre el usuario y el hardware del ordenador a partir de software. Gestiona los recursos del hardware proporcionando una interfaz al usuario. De esta forma pueden interactuar con dicho ordenador. Algunos de los sistemas operativos más usados son: iOS, Android, macOS, Microsoft Windows o Linux.

Linux es un sistema operativo de código abierto el cual es gratuito para cualquier usuario que quiera adoptarlo en su computadora. Este consta de muchas distribuciones. Las distribuciones son versiones del Sistema Operativo de Linux desarrolladas por diferentes individuos, equipos o empresas para mejorar la experiencia de los usuarios.

Una de estas distribuciones es "Debian" y es sobre la que tratará este estudio. Debian es un Sistema Operativo que trabaja con el Kernel (núcleo) de Linux y ha ido aportando distintas versiones desde 1993. Treinta años después nos preguntamos ciertas cosas como, ¿Los individuos que trabajaban en las primeras versiones siguen actualizando Debian? ¿Se trabaja individualmente o por equipos? ¿Cuántos paquetes sacan en cada versión? ¿Qué ocurre con ellos? Todo esto lo veremos a continuación.

1.1. Contexto

1.1.1. Proyecto Debian

El Proyecto Debian está formado por un grupo de voluntarios a nivel mundial que trabajan para producir una distribución del Sistema Operativo Linux basada en ”software libre”.

Con el término ”software libre”no nos referimos a su coste. Este va enfocado a la ”libertad real”dentro del software, es decir, ”software de código abierto”. Esto significa que cualquier usuario puede acceder al código fuente para estudiarlo, revisarlo, modificarlo o distribuirlo sin restricción alguna.

Debian es la distribución de Linux más relevante sin fines comerciales. En su comienzo, fue la única abierta a la participación de diferentes usuarios que quisieran aportar al proyecto con su trabajo.

Con el tiempo fue asentando un gran conjunto de directrices y procedimientos para el empaquetamiento y distribución de software. Esto les sirvió para poder alcanzar los estándares de calidad requeridos y con ello asegurar su buen funcionamiento.

1.1.2. Versiones Debian

Debian está conformado por varias versiones desde 1993 las cuales explicaremos a continuación:

- **Versiones 0.x (1993 - 1995):** estas versiones fueron las primeras y mas rudimentarias pero dieron lugar a la creación de Debian gracias a su creador **Ian Murdock**.
  - **Debian 0.01 hasta 0.90.**
  - **Debian 0.91:** disponía de un sencillo sistema de empaquetamiento que permitía instalar y desinstalar paquetes.
  - **Debian 0.93R5:** se asignaron responsabilidades de cada paquete a cada uno de los desarrolladores. Se comenzo a usar el administrador de paquetes **dpkg** para la instalación de paquetes después de la instalación del sistema. base.
- **Versiones 1.x (1996 - 1997): Bruce Perens** fue designado como líder del proyecto después de que Ian lo designara.

  - **Debian 1.0:** esta versión nunca fue publicada debido a una confusión al distribuir una versión en desarrollo con el nombre equivocado de Debian 1.0 que daría problemas en ejecución.
  - **Debian 1.1 Buzz:** es la primera versión de Debian con un nombre en clave sacado de las peliculas de **"Toy Story"**.
  - **Debian 1.2 Rex:** esta versión estaba completamente en formato **ELF** y usaba el núcleo (kernel) Linux 2.0.

    El formato **ELF** (Executable and Linkable Format) es un estándar. Se usa en sistemas operativos tipo **UNIX** (como Linux). Sirve para organizar y manejar archivos ejecutables, bibliotecas compartidas y otros objetos binarios.
  - **Debian 1.3 Bo**.

- **Versiones 2.x (1998 - 2000): Ian Jackson** pasó a ser el líder del proyecto.
  - **Debian 2.0 Hamm:** fue la primera versión multiplataforma de Debian. Agregó soporte para arquitecturas de la serie **Motorola 68000**.
  - **Debian 2.2 Potato:** agregó soporte para las arquitecturas PowerPC y ARM (CPU's de arquitectura RISC creadas por diferentes empresas).

- **Versiones 3.x (2002 -2005):**
  - **Debian 3.0 Woody:** se agregaron más arquitecturas a esta versión y fue la primera en usar **software criptográfico**. Este se usa para codificar información y mantener la transferencia segura de datos.
  - **Debian 3.1 Sarge:** incluye un nuevo instalador llamado **debian-installer**. Contiene detección automática de hardware, instalación sin supervisión y está traducido a más de treinta idiomas.

- **Debian 4.0 Etch (2007):**se añadieron mejoras como un instalador gráfico o la verificación criptográfica de los paquetes descargados entre otras.

- **Debian 5.0 Lenny (2009):**añadió la arquitectura **ARM EABI** para dar soporte a los nuevos procesadores **ARM**.

- **Debian 6.0 Squeeze:** con esta versión fue la primera vez que una distribución de Linux se extendía para permitir también el uso de un núcleo no Linux.
- **Debian 7.0 Wheezy (2011):**se introdujo el soporte de **multiarquitectura**. Esto permitía que los usuarios instalaran en una misma máquina paquetes de múltiples arquitecturas.
- **Debian 8 Jessie (2013):**trajo importantes mejoras de seguridad, como un nuevo kernel que solucionaba varias vulnerabilidades (como **ataques de enlace simbólico**).
- **Debian 9 Stretch (2015):**se introdujeron paquetes para la depuración a través de un repositorio nuevo en el archivo. Facilitaría el proceso de depuración y solución de problemas relacionados con esos paquetes.
- **Debian 10 Buster (2019):** incluyó por primera vez un marco de control de acceso obligatorio. Restringe las acciones que pueden realizar los programas, limitando su acceso a ciertos recursos del sistema, como archivos, directorios, redes, etc.
- **Debian 11 Bullseye (2021):** introduce un nuevo paquete, ipp-usb, que utiliza el protocolo IPP-over-USB, independiente del fabricante y soportado por muchas impresoras actuales. Esto permite que un dispositivo USB sea tratado como un dispositivo de red.

1.2. Estructura de la memoria

A continuación se exponen los capítulos en los que se organiza esta memoria y los puntos clave tratados en cada uno de ellos:

- **Capítulo 1: Introducción.** Se explica el contexto de Debian y las diferentes versiones distribuidas a lo largo de su historia.
- **Capítulo 2: Objetivos.** Se especifica cuales son los objetivos parciales para lograr el objetivo general.
- **Capítulo 3: Estado del arte.** Se muestran y explican las diferentes tecnologías usadas para la realización de dicho proyecto.
- **Capítulo 4: Diseño e implementación.** Se muestran las diferentes etapas que se han seguido en este proyecto a detalle.

- **Capítulo 5: Experimentos y validación.** Se indica el proceso seguido para alcanzar los diferentes objetivos usando diferentes tecnologías.

- **Capítulo 6: Resultados.** Se muestran los diferentes resultados de estos experimentos. También se comentan los diferentes patrones o tendencias procedentes del análisis de dichos resultados.

- **Capítulo 7: Conclusiones.** Se observan los resultados obtenidos y se comparan con lo que se esperaba obtener. Se realizan una serie de deducciones tras el tratamiento y el análisis de los datos. Se aplican los conocimientos adquiridos en la carrera y se plantean nuevas lineas de investigación.

Capítulo 2

Objetivos

2.1. Objetivo general

Este proyecto de fin de grado se basa en el análisis de los lanzamientos, paquetes y mantenedores presentes en la evolución de la distribución Debian a lo largo de su historia.

Se busca conocer, de cada lanzamiento (**release**), la evolución en el número de personas y equipos que lo mantienen, cuantos de ellos siguen en releases posteriores y que ocurre con los paquetes de un mantenedor si este abandona el proyecto. Se ha podido llevar acabo gracias al estudio de diferentes paquetes de datos aportados por Debian[1] en su página oficial. De esta forma obtenemos las diferencias y similitudes necesarias para la comparación de los releases de forma que podamos comprender su evolución.

2.2. Objetivos específicos

Para poder llevar a cabo dicho objetivo se requiere:

- **Diseño del diagrama entidad relación.** Comprender los diferentes campos que conforman un paquete y crear un diagrama para el posterior diseño de la base de datos que los alojará. .
- **Creación de la base de datos.** Crear una base de datos con un buen diseño (basada en el diagrama anterior) para poder lanzar **Query´s** con las que obtener la información que se

[1] `https://www.debian.org/releases/`

requiere.

- **Formulación de Query´s.** Analizar la información que queremos extraer de la base de datos y crear las llamadas necesarias para obtenerla.
- **Creación de tablas y gráficas.** Con la información obtenida se crean diferentes tablas y gráficas para ayudar a la comprensión de los datos de forma visual.

2.3. Planificación temporal

En el Diagrama de la Figura 2.1 se visualizan las diferentes tareas realizadas junto con la organización en días de las mismas.

Primero se marcaron los objetivos junto con las tecnologías a usar en este proyecto. Más tarde se realizó un análisis de los diferentes paquetes para poder crear el diagrama, el diseño de la base de datos y sus tablas correspondientes. Realizamos el parseo de las diferentes datos del paquete para poder insertarlos correctamente en la base de datos. La inserción de los datos fue lo más problemático en este proyecto. Cada release consta de muchos paquetes en los cuales hay que parsear, extraer e insertar todos los datos. Este estudio se realiza sobre 11 releases. Al tratarse de tanto volumen de datos, el tiempo de ejecución fue elevado. Al comprobar las bases de datos y hallar errores, se insertaban de nuevo los datos por lo que esto fue lo más complejo del proyecto.

Finalmente se realizaron una serie de llamadas sql (Query´s) con las que extraer la información necesaria para el proyecto junto con sus gráficas para obtener dicha información de forma visual.

Con ello pudimos obtener conclusiones sobre el estudio de estos datos y comenzar con la redacción de la memoria.

Dr. Gregorio Robles me propuso dicho proyecto en noviembre de 2023 y el tiempo desempeñado en él ha sido de 4h diarias reflejadas en los días laborales, de lunes a viernes. En los meses de abril y mayo se intensifico mi desempeño a 6h diarias para desarrollar la memoria con la mayor dedicación posible.

Nombre de actividad	Fecha inicio	Duración en días	Fecha fin
Instalación y aprendizaje de las tecnologías a usar	23-nov	3	26-nov
Analisis de paquetes y creación de diagrama entidad-relación	27-nov	7	04-dic
Creación de las bases de datos junto a sus tablas sql	05-dic	2	07-dic
Parseo de la información de todos los paquetes	08-dic	35	12-ene
Inserción de los paquetes en las bases de datos	14-dic	62	14-feb
Formulación de querys sql	15-feb	413	03-abr
Creación de tablas y gráficas	07-mar	27	03-abr
Redacción de la memoria	04-abr		

Figura 2.1: Diagrama de Gantt

Capítulo 3

Estado del arte

3.1. Python 3.12.0

Python [2] es un lenguaje de programación orientado a objetos de alto nivel con una sintaxis fácil de interpretar y leer. Tiene un amplio uso en computación científica, desarrollo web y automatización.

Peter Norvig, director de investigación de Google afirma que ”Python ha sido una parte importante de Google desde el principio, y permanece así a medida que el sistema crece y evoluciona”.

Al ser un lenguaje popular tiene una mayor selección de bibliotecas, lo que ahorra a un desarrollador cantidades increíbles de tiempo y esfuerzo. También tiene más tutoriales y documentación. Esto aumenta las probabilidades de encontrar soluciones a los problemas.

Una curiosidad del lenguaje es que la empresa de mayor emprendimiento de la ïnteligencia artificial” **Open AI**[1] está diseñada con python en gran parte y sus bibliotecas son públicas para su uso.

En la figura 3.1 podemos observar que Python es el lenguaje más usado del mundo según sus últimos datos en 2019 [5].

[1] `https://platform.openai.com/docs/libraries/python-library`

Figura 3.1: Lenguajes de programación más populares del mundo

3.2. My sql connector

Esta es la librería más importante de Python para poder desarrollar este proyecto. MySQL Connector/Python [6] permite a los programas de Python acceder a las bases de datos MySQL, utilizando una API que cumple con la Especificación de API de Base de Datos de Python. Sus funciones más destacadas e importantes son:

1. **Conexión a la base de datos:** con la función **mysql.connector.connect().**
2. **Ejecución de consultas SQL:** con la función **cursor.execute().**
3. **Recuperación de resultados:** con la función **cursor.fetchall()**.
4. **Inserción, actualización y eliminación de datos:** con consultas SQL como INSERT, UPDATE y DELETE, ejecutadas con la función **cursor.execute().**
5. **Gestión de errores:** se capturan y manejan errores en el código Python usando excepciones.
6. **Desconexión de la base de datos:** con la función **connection.close().**

3.3. Matplotlib

Matplotlib es [3] es una librería de **Python open source**. John Hunter, neurobiólogo, fue su desarrollador inicial en 2002. Su objetivo era visualizar las señales eléctricas del cerebro de personas epilépticas. Por ello intentó replicar las diferentes funcionalidades de MATLAB (gráficas) con Python.

Matplotlib ha sido mejorado a lo largo del tiempo por numerosos contribuidores de la comunidad open source. Se usa para crear gráficas y diagramas de gran calidad que aportan la visualización de los datos de forma detallada.

Es posible crear trazados, histogramas, diagramas de barras y cualquier tipo de gráfica como en la Figura 3.2 con unas lineas de código.

Esta librería es particularmente útil para las personas que trabajan con Python o NumPy.

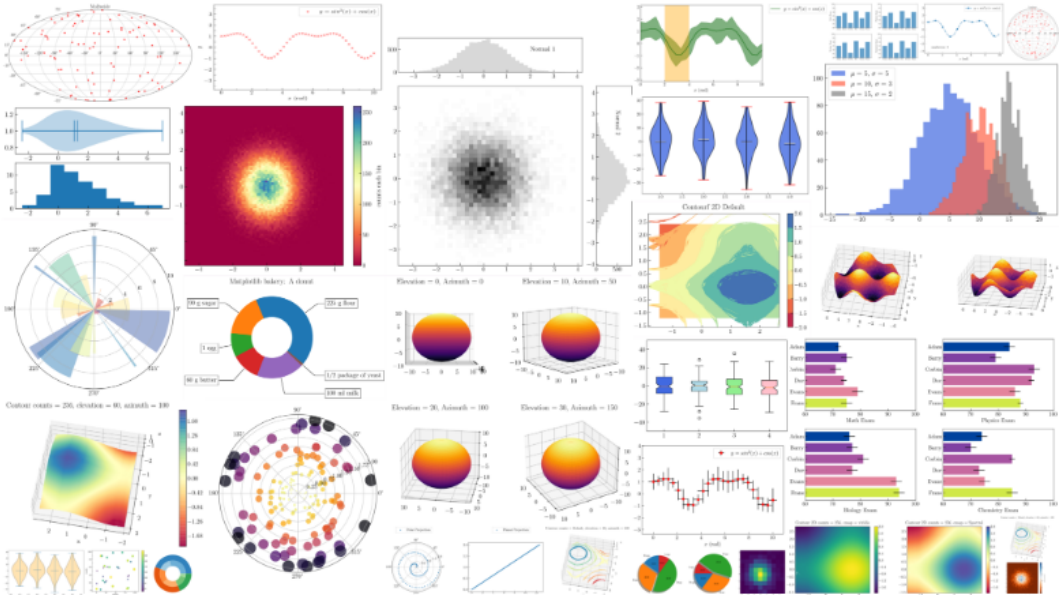

Figura 3.2: Ejemplos gráficas Matplotlib

3.4. Pycharm

PyCharm es el **IDE** [4] más popular para Python hasta la fecha. Esta plataforma híbrida se utiliza habitualmente para el desarrollo de aplicaciones en Python por grandes empresas como Twitter, Facebook, Amazon y Pinterest.

Un **Integrated Development environment (IDE)** o **Entorno de Desarrollo Integrado (EDI)** es un conjunto de herramientas necesarias para desarrollar software. Incluye un editor y un compilador.

Pycharm es compatible con Windows, Linux y macOS. Además contiene módulos y paquetes que ayudan a los desarrolladores a programar software con Python más rápido y con menos esfuerzo y se puede personalizar para responder a las necesidades específicas de un proyecto.

Figura 3.3: Pycharm

3.5. SQL

Es un lenguaje de programación que **almacena y procesa información en una base de datos relacional**.

Una base de datos **relacional** almacena información en forma de tabla, con filas y columnas que representan diferentes atributos de datos junto con sus relaciones. Con ello se puede **almacenar, actualizar, eliminar, buscar y recuperar** información de la base de datos.

Es un lenguaje de consulta popular que se usa con frecuencia en todos los tipos de aplicaciones debido a su alta integración con el resto de lenguajes tales como **java, python, C#, PHP, etc**.

3.6. MySQL WorkBench

Es una herramienta visual ideal para **modelar, diseñar y administrar bases de datos MySQL** junto con el uso de código MySQL.

Se trata de una herramienta gráfica que fue creada por la compañía Oracle. Es un programa de cliente que, a través de un entorno de desarrollo integrado, facilita la creación, consulta y administración de bases de datos [1].

Este software tiene múltiples funcionalidades como:

- **Modelado de datos**: diseñar, modelar, gestionar y generar bases de datos de forma visual.
- **Ingeniería inversa**: recopilar información o datos a partir de un producto determinado para saber qué elementos lo componen.
- **Migrar bases de datos**: migrar desde Microsoft SQL Server, Microsoft Access, Sybase ASE y otros sistemas de gestión de bases de datos a MySQL.

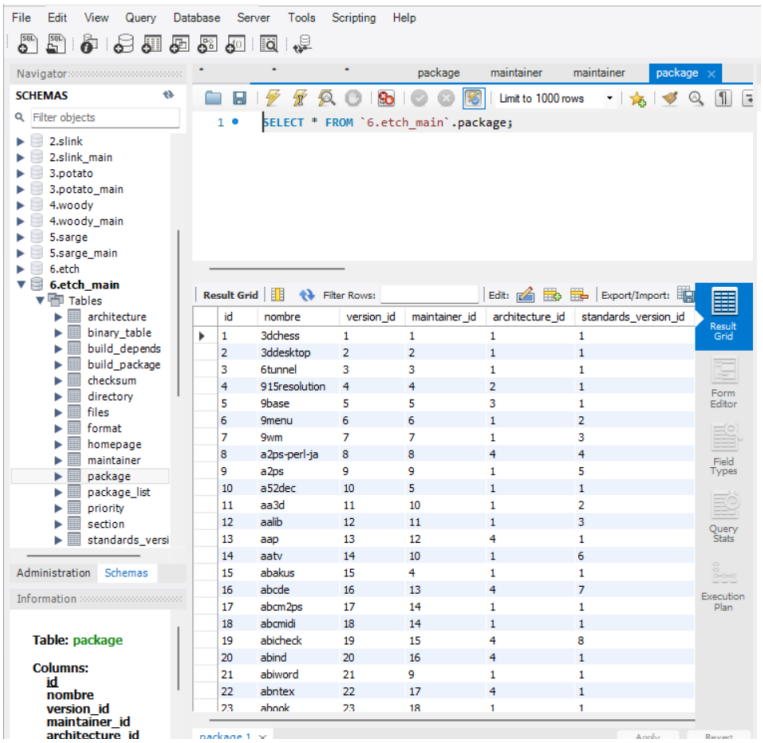

Figura 3.4: SQL Workbench

Capítulo 4

Diseño e implementación

4.1. Arquitectura general

Para poder completar este proyecto debemos analizar una gran cantidad de datos. Debian consta de 11 releases (lanzamientos) en los cuales se hallan un gran volumen de paquetes que va aumentando exponencialmente en releases posteriores.

Por ello, se necesita seguir una secuencia específica representada en la **Figura 4.1** para descargar, limpiar, ordenar y extraer estos datos de forma efectiva.

Primero debemos descargar los diferentes releases de Debian desde su página oficial [**?**]. Con ello ya podremos analizar los diferentes paquetes que contienen y así poder diseñar el diagrama entidad - relación correspondiente.

Una vez tenemos el diseño completo, podremos comenzar con la creación de las diferentes BBDD (bases de datos) tal que cada release sea una BBDD con sus correspondientes tablas sql.

En este punto creamos un script cuya funcionalidad será parsear los diferentes paquetes de cada release con el fin de extraer la información necesaria.

Una vez extraída dicha información creamos otro script que, conectándose a la BBDD correspondiente, introduzca los datos parseados en la BBDD para su posterior análisis.

Se crean una serie de Query´s o llamadas sql para poder obtener la información que necesitamos y así responder a los intereses del proyecto.

Por último, creamos una serie de gráficas y tablas para ayudar a la visualización de los resultados obtenidos en este estudio.

Figura 4.1: Fases del tratamiento de datos

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] G. Castillo. Mysql workbench: Qué es, descarga, instalación y uso, 2023.

[2] D. R. Center. ¿qué es python?, 2022.

[3] Datascientest. Matplotlib: todo lo que tienes que saber sobre la librería python de dataviz, 2022.

[4] Datascientest. Pycharm : Todo sobre el ide de python más popular, 2022.

[5] G. Moreno. Los lenguajes de programación más usados del mundo, 2019.

[6] M. SQL. Chapter 1 introduction to mysql connector/python, 2024.