UNIVERSIDAD
REY JUAN CARLOS

Grado en Tecnologías de las Telecomunicaciones

Curso Académico 2015/2016

Trabajo Fin de Grado

Análisis de proyectos FOSS

Autor : Joel Peralta Rodríguez

Tutor : Dr. Gregorio Robles

Proyecto Fin de Carrera

Análisis de proyectos FOSS

Autor : Joel Peralta Rodríguez

Tutor : Dr. Gregorio Robles Martínez

La defensa del presente Proyecto Fin de Carrera se realizó el día de junio de 2016, siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de junio de 2016

Dedicado a

mis padres, mi hermana y mi novia

II

Agradecimientos

Tras unos años que han pasado como una montaña rusa, con sus momentos altos y bajos, me gustaría acordarme de las personas que me han ayudado en todos esos momentos para llegar a este punto, en el cual, mediante este proyecto, se cierra una etapa muy bonita y estresante de mi vida, y se abre otra que será aún más bonita y seguramente, muchísimo más estresante.

En primer lugar querría agradecer a mi tutor en este proyecto, el profesor D. Gregorio Robles Martínez, su apoyo y sus consejos, no solo para la realización del trabajo de fin de grado, sino para la búsqueda de motivaciones y retos para mi futura carrera profesional.

Querría valorar y agradecer también el esfuerzo, tanto económico como mental, que supone tener a un *protoingeniero* en casa. Por este esfuerzo, mis padres y mi hermana merecen todo el reconocimiento por mi parte, porque sin ellos, yo no estaría donde estoy, porque su apoyo, ha sido esencial para no volverme loco y ellos conmigo, y porque ver el orgullo y el cariño que sienten por mi, hace que me esfuerce aún más cada día.

También querría agradecerle a mis abuelos, primas y primos, tías y tíos, a toda mi familia, las palabras de ánimo y la fe y orgullo que mostraron tener por mi, que en ocasiones llegó a superar la mía propia.

A mis compañeros de carrera de los que he aprendido mucho y a los que espero también haberles dejado un buen recuerdo. En especial a José con el que he estado estos cuatro años y con el que he compartido éxitos y derrotas.

Y además de mis compañeros de carrera, también a mis compañeros en el trabajo, que me trataron como uno más desde el principio y con los que he aprendido mucho en este corto periodo de tiempo.

Y por último, agradecerle a una persona muy especial que a pesar de la distancia, comparte conmigo todos mis sinsabores y festeja conmigo las victorias. Mi novia Carmen, que ha conseguido hacer de este estudiante, un futuro ingeniero muy feliz, con su apoyo constante y su

cariño sincero.

A todos ellos, y en especial mi familia y mi novia, gracias por hacer que este reto haya sido abordable. Muchas gracias.

Resumen

En este proyecto se pretende dar a conocer el ecosistema de proyectos de software libre y abierto, para mostrar los beneficios sociales y técnicos de su uso en la industria.

En un entorno en el que la tecnología interviene en muchos aspectos de nuestras vidas, la importancia de que esta tecnología se desarrolle por y para las personas es cada vez más importante.

Por tanto, gracias a la utilización de herramientas de análisis de ficheros y repositorios, como Scancode Toolkit, Github API y diferentes librerías Python, se realizarán análisis de proyectos de referencia en la comunidad *Open Source*, de los cuales se obtendrán datos legales, técnicos y sociales.

Esto nos permitirá conocer en profundidad las características y la importancia de este tipo de proyectos en el sector tecnológico y del desarrollo de software, además de su repercusión en la sociedad.

Summary

This project aims to be an introduction to the free and open source software ecosystem, showing the social and technical benefits of its use in the industry.

In an environment where technology is present in every aspect of our lives, the importance of this technology being developed by people and for people is more and more important.

Therefore, using file and repository analyzer tools like Scancode Toolkit, Github API and several Python libraries, we will analyze projects of high importance in the Open Source community, from which we will obtain legal, technical and social data.

This will allow us to know in deep the characteristics and the importance of this type of projects in the technological and software developing fields, as well as its repercussion in society.

Índice general

Índice de figuras

Capítulo 1

Introducción

En este proyecto se ha realizado un análisis del ecosistema de proyectos de software libre que podemos encontrar en la red, centrándose especialmente en la plataforma Github. Mediante este análisis se ha intentado dar a conocer las características de este entorno, además de determinar las peculiaridades que aportan los datos obtenidos a la formación y desarrollo del Free Open Source Software.

A continuación, para una mejor comprensión del proyecto, se explicarán el contexto del mismo, su estructura y las motivaciones personales para realizarlo.

1.1 Contexto

En un sector tan amplio como el de la programación, en el cual se pueden diseñar y desarrollar soluciones a problemas muy diversos, surge el planteamiento de este proyecto fin de grado para tratar de buscar las características comunes de diferentes proyectos de software libre y de sus desarrolladores.

Con la obtención de la información presente en estos proyectos, se pretende realizar una comparativa de los diferentes tipos de software libre que se están desarrollando en la actualidad. Además de sus características en cuanto a *copyrights* y licencias. De esta manera se podrán entender mejor las cualidades que ha de tener un determinado software para ser libre y abierto, y para poder progresar y obtener mayor relevancia en el mundo del Free Open Source Software.

Esta información legal, aunque secundaria, puede suponer un punto clave para un determinado proyecto, puesto que en la comunidad Free Open Source, un aspecto muy importante del

software debe ser su libertad de uso, acceso al código y distribución y modificación del mismo.

Este proceso de análisis se realiza en el contexto de una aplicación web, por tanto, el resultado del mismo podrá ser accesible a múltiples usuarios que busquen el tipo de proyecto que se está realizando en la actualidad, o a colaboradores activos de proyectos de software libre, o características de distintos proyectos, que les permitan tomar conciencia de la importancia de que el nuevo software que están desarrollando sea libre y accesible para todo el mundo.

1.2 Motivaciones personales

La principal motivación por la que se decidió realizar este proyecto es el proceso de análisis, filtrado y representación de grandes cantidades de datos. Además al tratarse de una aplicación web, engloba diferentes aspectos importantes del desarrollo de software, como son el desarrollo web, técnicas de Big Data y también el desarrollo de soluciones a problemas de índole académica.

Por otra parte, los diferentes tipos de tecnologías utilizadas para desarrollar el proyecto hacían que este fuera más interesante desde un punto de vista educativo y de desarrollo de las habilidades programáticas.

1.3 Estructura de la memoria

Para una mejor lectura, se desglosa a continuación la estructura de este documento:

- **1. Introducción.** Con el objetivo de enmarcar el proyecto en su contexto.
- **2. Objetivos.** Sección en la que se muestran los objetivos finales y parciales del proyecto.
- **3. Estado del arte.** Donde se muestran las tecnologías y metodologías utilizadas para elaborar las distintas partes del proyecto.
- **4. Diseño e implementación.** En esta sección se elabora una explicación del proceso realizado y el diseño previo a este proceso.
- **5. Resultados.** Para mostrar los resultados obtenidos tras la realización del proyecto.

- **6. Conclusiones.** Donde se enumeran las conclusiones obtenidas de los resultados de la sección anterior.
- **7. Apéndices.** Sección en la que se muestra cualquier información adicional para la adecuada comprensión de la memoria.
- **8. Bibliografía.** Lista de fuentes de información en las cuales se basa este documento.

Capítulo 2

Objetivos

2.1 Objetivo general

El objetivo principal del proyecto es dar una visión general de distintos proyectos de software libre que se desarrollan en la actualidad, para comprender mejor el ecosistema Open Source y ofrecer a los posibles usuarios una manera de encontrar proyectos populares en los que colaborar, usuarios activos que puedan unirse a nuevos proyectos y de observar las licencias, herramientas y lenguajes de programación más populares entre los usuarios.

2.2 Objetivos específicos

Para llevar a cabo el objetivo principal, se han cumplido una serie de objetivos secundarios que se describirán a continuación:

- Familiarizarse con la herramienta Scancode Toolkit que servirá para analizar proyectos desde la terminal de Unix, y posteriormente analizar la información devuelta por dicha herramienta.

- Inicializar el proyecto Django/Python que albergará la aplicación web en la que se engloba el análisis de proyectos.

- Adaptar la herramienta para ser usada como un módulo de Python y poder tener mayor versatilidad al analizar proyectos en distintos formatos.

- Desarrollar una API REST cuyos recursos servirán para analizar proyectos y devolver los resultados en un formato fácilmente representable en el navegador.
- Crear los modelos de la base de datos y almacenar los primeros proyectos FOSS que serán posteriormente representados.
- Montar la estructura del *framework* web AngularJS para realizar de manera correcta y ordenada la representación de los datos obtenidos de la API.
- Representar los datos obtenidos de la API REST mediante la librería nvd3, basada en d3.js, en diferentes gráficos.
- Corregir la estética de la página, además de distintos problemas de adaptación entre el back-end (Python) y el front-end (JavaScript).
- Analizar mayor número de proyectos para obtener mayor número de datos que ayuden a sacar conclusiones del entorno Open Source.

2.3 Planificación temporal

La planificación temporal del proyecto queda representada por el gráfico del apéndice B para una mejor comprensión de los plazos tomados para la consecución de los distintos objetivos.

Capítulo 3

Estado del arte

En esta sección se presentarán las tecnologías utilizadas en el desarrollo del proyecto, tanto en el análisis de datos, como en la representación de los mismos.

Para ello se realizará una descripción general de cada herramienta, y de las cualidades específicas por las cuales han sido utilizadas.

3.1 Free Open Source Software (FOSS)

Para entender el concepto de FOSS, se explican a continuación los requisitos del software para ser libre y abierto. Estos dos términos siguen las directrices de dos organizaciones, Free Software Foundation [4] y Open Source Initiative [10] respectivamente.

A continuación se especificarán las características de cada uno de ellos.

Para que el software se considere libre, ha de cumplir estas cuatro libertades:

- Libertad de uso del programa por cualquier razón.
- Libertad de acceso, estudio y modificación del código.
- Libertad de copia y distribución.
- Libertad de mejora y publicación de las mismas.

Para tener estas libertades, es necesario que el código fuente del software sea de libre acceso al público.

Por otro lado, la iniciativa Open Source se centra menos en las características éticas del software libre, y más en las técnicas y económicas. Sin embargo, son muy pocos los proyectos de software que siendo reconocidos por la Open Source Initiative, no sean considerados Free Software. Por esta razón se suele utilizar el término Free Open Source Software.

3.2 Python

Python es un lenguaje de programación *multiparadigma*, al permitir la programación orientada a objetos, declarativa y funcional. Es interpretado, por no requerir compilación, con *tipado* dinámico y *multiplataforma* [12].

Este lenguaje fue publicado por Guido van Rossum en 1991 en los Países Bajos como un proyecto de software libre. Actualmente se encuentra bajo una licencia Python Software Foundation License, compatible con la licencia general de GNU a partir de su versión 2.1.1.

Debido a su facilidad de aprendizaje y su uso sencillo, además de por su versatilidad, al permitir la combinación de programación declarativa con programación orientada a objetos, y el gran número de módulos utilizables en este lenguaje, Python se eligió como la opción adecuada como lenguaje de programación para este proyecto.

Actualmente existen dos versiones de Python, las 2.X y las 3.X. Para este proyecto, se utilizó la versión 2.7.6 de Python. Esto es debido a que, aunque las versiones 3.X se están desarrollando cada vez más, existen incompatibilidades entre estas y las versiones 1.9 del *framework* Django que se describirá a continuación.

3.2.1 Django

Django [3] es un *framework* web de alto nivel para el lenguaje de programación Python. Este *framework* permite el desarrollo rápido, limpio y estructurado de cualquier aplicación web sin importar su grado de complejidad. Django es, al igual que Python, de código abierto y cuenta con una licencia BSD.

Este *framework* estructura sus proyectos siguiendo el sistema Modelo-Vista-Controlador. Esto, sumado a su filosofía DRY (Don't Repeat Yourself), hacen que Django sea la mejor opción para entrar en contacto con el desarrollo de aplicaciones web de una manera estructurada, concisa y abordable para programadores de cualquier nivel.

Estas cualidades, además de la sencillez de interacción con bases de datos relacionales, supusieron un punto clave para la elección de Django como *framework* sobre el que desarrollar la aplicación web en la que se basa este proyecto.

Para la realización del mismo se ha utilizado la versión 1.9 de Django, por una mayor sencillez y una mejor *interoperabilidad* con las bases de datos relacionales, como puede ser MySQL. Esto supone como ya se ha comentado anteriormente, que existan restricciones de compatibilidad de está versión con las versiones 3.X de Python.

3.2.2 Scancode Toolkit

La herramienta Scancode Toolkit es un proyecto de software libre desarrollado en Python, que permite 'escanear' carpetas y ficheros para obtener, entre otros, las licencias y *copyrights* contenidos dentro de cada fichero.

Esta herramienta permite su uso desde la terminal de Linux, o en el caso de este proyecto, la utilización del código y una interfaz API contenida en el mismo para obtener la información requerida en cada caso.

Para poder utilizar Scancode Toolkit en el proyecto, se tuvo que adaptar el código para la obtención de la información legal del archivo directamente desde su contenido, en lugar de facilitar la localización del mismo, tal y como se realizaba por defecto en el código que se puede ver en la plataforma Github [9].

Esta herramienta además, cuenta con otras funcionalidades que no han sido utilizadas para realizar el análisis de los archivos en este proyecto. Estas funcionalidades son la obtención del lenguaje en el que están escritos, el número de lineas, etc.

3.2.3 GitPython

Esta librería Python sirve como interacción de alto nivel con repositorios git.

La interacción de esta librería con los repositorios abarca desde la funcionalidad de clonar un repositorio, a la obtención del '*blame*' de un fichero dentro del mismo. Y precisamente estas dos funcionalidades son las que se han utilizado en este proyecto.

La librería GitPython es de código abierto y está disponible para su uso y modificación en su repositorio de Github [5].

3.2.4 Pygments

Pygments es un paquete escrito en Python que permite resaltar las características sintácticas principales de un código fuente dado.

Esta librería nos sirve por tanto para poder encontrar patrones característicos de un determinado lenguaje dentro del código, y de esta forma, determinar en que lenguaje está programado un determinado archivo.

Esta diferenciación será usada en el análisis planteado en este proyecto, para determinar cuales son los lenguajes de programación más utilizados en los proyectos FOSS a analizar.

3.3 Base de datos

Para la realización de este proyecto se realizó una comparativa previa del tipo de base de datos que se utilizaría [13].

A la hora de escoger una base de datos adecuada para un proyecto, se pueden elegir entre dos tipos de bases de datos, relacional y no relacional.

Una base de datos relacional, normalmente con SQL como lenguaje base, es aquella que cumple con el modelo previamente fijado sobre ella. Este tipo de bases de datos permiten establecer relaciones entre los datos almacenados en distintas tablas.

Estas características suponen una serie de ventajas de uso:

- Mayor simplicidad de la base de datos
- Robustez.
- Flexibilidad y mejor *performance*.
- Escalabilidad.
- Mayor eficiencia al realizar peticiones sobre ella.

Por otro lado, las bases de datos no relacionales se caracterizan por no guardar relaciones entre sus distintas colecciones. Su modo laxo de almacenamiento permite guardar en sus colecciones cualquier tipo de dato o estructura sin seguir ningún modelo fijado previamente.

Dichas particularidades otorgan a las bases de datos no relacionales las siguientes ventajas:

- Sencillez en su diseño.
- Mayor escalabilidad horizontal.
- Mejores cualidades para usos de Big Data.
- Mayor rapidez en la obtención de los datos.

Observando las cualidades de ambas, se decidió usar una base de datos relacional. Ya que a pesar de su mejor funcionamiento en entornos de Big Data y su rapidez al devolver los datos almacenados, las bases de datos no relacionales tienen numerosos problemas a la hora de hacer

cálculos de datos agregados. Al tratarse nuestra aplicación web de un *dashboard* interactivo en el que se mostrarán datos sumados de los distintos proyectos almacenados, las bases de datos relacionales se adaptan mucho mejor al uso que se dará a los datos recogidos.

3.3.1 MySQL

MySQL [11] es un sistema de gestión de bases de datos relacionales *multiplataforma*, desarrollado en C y C++ bajo una licencia dual GPL/Licencia Comercial de Oracle Corporation y es considerada la base de datos relacional open source más popular del mundo.

La simplicidad de MySQL como gestor de bases de datos no relacionales supone una cualidad esencial para la gestión de páginas web con representación de datos dinámicos.

Además, MySQL soporta una gran cantidad de tipos diferentes para almacenar datos y al estar tan extendido, tiene múltiples APIs para interactuar con la base de datos desde diferentes lenguajes de programación.

Estas cualidades además de su escalabilidad y su facilidad de uso utilizando Django como *framework* web, son características clave para el uso de este gestor de bases de datos en este proyecto.

3.4 Front-end

En el código del cliente o front-end se han utilizado múltiples tecnologías para la visualización de los datos y la realización de una adecuada interfaz de usuario. Esas tecnologías se pasan a describir en las siguientes secciones.

3.4.1 JavaScript

JavaScript [8] es un lenguaje de programación interpretado, dialecto del estándar ECMAScript, y fue desarrollado por Brendan Eich, trabajador de Netscape en 1995. Este lenguaje es orientado a objetos, imperativo y con *tipado* débil y dinámico.

Este lenguaje es usado normalmente en el lado del cliente como parte del código utilizado por el navegador. Permitiendo la realización de páginas web dinámicas e interfaces de usuario.

JavaScript es el único lenguaje apoyado comúnmente por todos los navegadores modernos. Por esta razón, es el más utilizado en el desarrollo de aplicaciones web en el lado del cliente, y el que se utiliza en este proyecto.

AngularJS

AngularJS [1] es un *framework* web abierto de JavaScript, mantenido por Google y que permite la creación de SPA o Single Page Applications.

Angular permite un alto nivel de organización y dinamismo en el lado del cliente gracias a su principal característica, el two way data binding, cuyo funcionamiento permite que cualquier modificación realizada sobre la vista repercuta en el modelo y viceversa, tal y como se muestra en la figura 3.1. Esto lo han convertido en uno de los *frameworks* web más populares en la actualidad, al liberar al back-end de funciones de *renderizado* de *templates*.

Para la representación de los datos obtenidos del análisis, se utilizarán gráficos que se generarán mediante la modificación del DOM de la página web. Esto supondría una gran cantidad de código desestructurado si se utilizara JavaScript con jQuery como librería principal, pero

con la utilización de AngularJS, además de ordenar el código en una estructura MVC, se reduce drásticamente la cantidad del mismo.

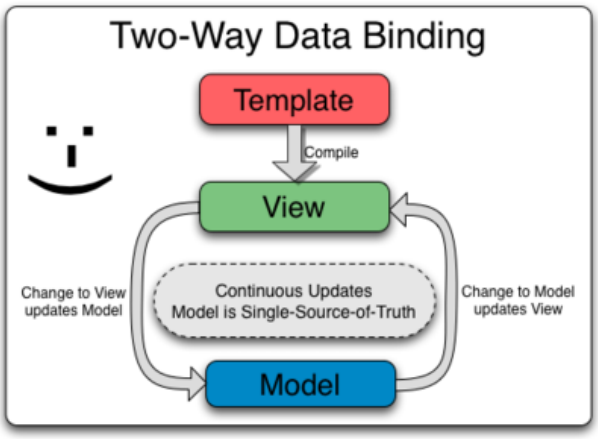

Figura 3.1: Funcionamiento del Two Way Data Binding

D3.js

La librería de JavaScript D3 [2], llamada así por las siglas Data-Driven Documents, permite producir, a partir de datos, gráficos dinámicos e interactivos mediante SVG.

Para este proyecto utilizaremos D3 con otra librería basada en d3.js, llamada nvd3.js, que permite crear de manera sencilla y uniforme gráficos de distintos tipos y para cualquier tipo de datos, que además se actualizan conforme estos datos vayan cambiando.

3.4.2 HTML5 y CSS3

Para la realización de este proyecto, juegan un papel importante tecnologías como HTML5 o CSS3, que permiten la creación de una interfaz de usuario llamativa y fácil para el correcto funcionamiento de la aplicación web.

HTML5

HTML5 [6] es un lenguaje de marcado que nos permite realizar plantillas en las que representar nuestra interfaz de usuario.

Además el DOM de la página, que es una representación en árbol de los distintos elementos de la plantilla HTML, nos permite modificar mediante distintas herramientas, como puede ser Angular, el aspecto y organización de nuestra interfaz.

CSS3

Como complemento a HTML, tenemos CSS, que es un lenguaje creado para definir la presentación y el estilo de documentos HTML o XML.

Mediante este lenguaje definiremos el estilo de nuestra interfaz de usuario, dotándola de las características deseadas para hacerla más atractiva al usuario.

3.4.3 Bootstrap

Bootstrap [14] es un conjunto de herramientas de código abierto, desarrollado y mantenido por Twitter, que permite hacer nuestra página HTML/CSS responsiva, es decir, Bootstrap permite que una página web se adapte al tamaño del navegador o el dispositivo en que está siendo mostrada.

Esta funcionalidad la otorga el sistema de celdas de Bootstrap que permite una mejor maquetación de nuestra página web.

En este proyecto se utiliza Bootstrap por cuestiones estéticas, pero además por seguir con la filosofía de organización y estructuración del código que se asume al utilizar AngularJS.

3.5 Control de versiones

Para un mejor seguimiento y control de los avances de este proyecto, se ha utilizado Git como herramienta de control de versiones.

Este software fue diseñado por Linus Torvalds en 2005 para permitirle, a él y a su equipo, un desarrollo más ágil y productivo.

Por este motivo y por la función de *backup* que estas versiones permiten, se ha utilizado este software para registrar cada hito conseguido en el proyecto.

Capítulo 4

Diseño e implementación

En esta sección se desarrollará el proceso seguido para realizar el diseño preliminar del proyecto y su implementación.

Primero se dará una visión general de la arquitectura de la aplicación, para posteriormente centrarse en las dos partes principales de la aplicación, el front-end y el back-end.

4.1 Arquitectura general

Este proyecto consiste en una aplicación web Django en la que se mostrarán los resultados de un análisis producido en el back-end de la aplicación.

El funcionamiento de la aplicación se muestra en la figura 4.1, en la que se puede resumir la estructura del proyecto.

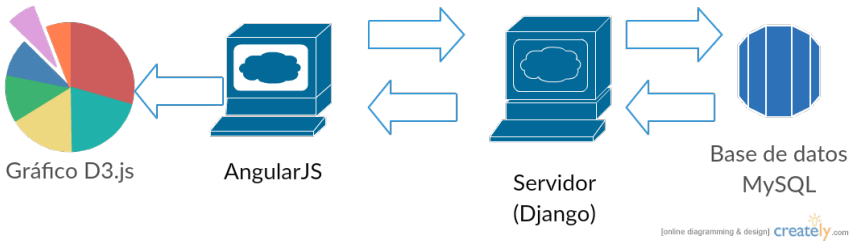

Figura 4.1: Estructura del proyecto

La aplicación constará de dos partes, el back-end y el front-end.

El back-end o lado del servidor ha sido desarrollado en Python mediante el *framework* web Django. En esta parte del código se ha desarrollado una API REST que devolverá, en función

de los recursos a los cuales se acceda, los resultados analizados en el momento de la petición o almacenados en la base de datos, del proyecto FOSS requerido por el usuario. Además permitirá la obtención de los datos globales de la aplicación, así como los correspondientes a un usuario y aquellos relacionados con una determinada información legal.

De los resultados devueltos por el servidor se servirá el código del navegador o front-end. Este código realizará funciones de visualización de los datos obtenidos. Mediante el *framework* web AngularJS se realizará una SPA, que mostrará dos secciones en las que visualizar los datos. Una en la que analizar un proyecto y otra en la que visualizar las estadísticas totales almacenadas en la base de datos, además de poder filtrarlas. Los datos se representarán mediante gráficos de la librería D3.

Para una mejor comprensión de la aplicación, se desarrollará en mayor profundidad el funcionamiento de la misma en los siguientes apartados.

4.2 Back-end

En este apartado se explicarán las características de la API REST desarrollada en el servidor Django.

Para implementar esta aplicación y esta API REST, se puso en funcionamiento primero un proyecto Django sobre el cual se desarrolló el back-end de la aplicación.

Desde este servidor Django se sirven los archivos correspondientes a la página principal y, en este caso, única de la aplicación, además del código correspondiente al funcionamiento de la página. También en este servidor se puede acceder a cuatro recursos diferentes correspondientes a la API REST ya mencionada.

Estos recursos son los siguientes:

- **/analyze**: Accediendo a este recurso y enviando en el cuerpo de la petición HTTP la *url* de un proyecto, se obtienen los datos correspondientes al análisis del proyecto al que corresponde la *url*. Entre estos datos se encuentran los ficheros que contiene y los contribuidores, licencias y *copyrights* presentes en ellos.

- **/all**: Mediante el acceso a este recurso se devolverán en la respuesta todos los datos contenidos en la base de datos que hagan referencia al total de lineas que ha desarrollado

cada contribuidor, los lenguajes utilizados y los *copyrights* y licencias presentes en los distintos ficheros almacenados.

- **/user**: Al realizar una petición HTTP sobre este recurso con el nombre de un usuario en su cuerpo, se devolverán los lenguajes utilizados por este usuario, los proyectos en los que ha participado y las licencias y *copyrights* que ha utilizado en sus ficheros.
- **/legal**: Al acceder a este recurso, se podrán enviar una licencia o un copyright por los cuales se quieran filtrar los ficheros almacenados en la base de datos, devolviendo en la respuesta todos los ficheros que contengan esta información legal.

4.2.1 Análisis de proyectos

A continuación se describirá el proceso de análisis de los distintos proyectos, así como la implementación de dicho análisis.

Proceso

El análisis se realizará al acceder al recurso /analyze, siempre y cuando no haya sido almacenado previamente en la base de datos, y la fecha en la que fue almacenado sea más reciente que la fecha de última modificación del proyecto.

Si se cumplen estas dos condiciones, en la vista correspondiente al recurso /analyze, se utilizará la librería GitPython para clonar el repositorio del proyecto a analizar, para posteriormente utilizar la herramienta Scancode Toolkit sobre la carpeta en la que este proyecto fue clonado.

Mediante esta herramienta finalmente se obtendrá la información legal de cada fichero, y mediante la librería GitPython y la API de Github se obtendrá otra información como pueden ser los contribuidores y las contribuciones de cada uno de ellos.

Implementación

Para implementar este proceso se hará uso de una función que con la herramienta Scancode Toolkit, recorrerá todos los archivos contenidos en la carpeta, para obtener de cada uno de ellos sus licencias y *copyrights*. Esta herramienta utiliza una lista de licencias y reglas que le sirven como patrón para comparar el contenido de cada fichero en busca de las licencias a las que correspondan dichas reglas.

Además de Scancode, se utilizará, como se especificó anteriormente, la API de Github, que devuelve un objeto JSON con la información requerida, en este caso, los contribuidores e información general del proyecto.

Por otro lado, también se utilizan GitPython, para obtener las lineas equivalentes al comando git *blame* de Git, que serán posteriormente *parseadas* para obtener el autor de cada una, y la biblioteca Pygments que a partir del contenido de cada fichero, determinará el lenguaje en que está programado en base a su sintaxis.

Otras consideraciones

Para el correcto funcionamiento de la aplicación, fueron necesarias ciertas modificaciones sobre la herramienta Scancode Toolkit:

- El funcionamiento por defecto de Scancode Toolkit carga la lista de reglas y licencias en cada análisis. En este caso, al tratarse de una aplicación web, esta funcionalidad no es viable, por lo tanto, se tuvo que modificar el funcionamiento de la herramienta para que la carga de licencias se realice nada más arrancar la aplicación web, almacenando después la lista en el servidor, y evitando así que se tengan que realizar cargas posteriores.

- Para facilitar el análisis completo de los proyectos se decidió leer el contenido de los distintos ficheros al comenzar el proceso de análisis, evitando así, las distintas lecturas que habría que realizar por defecto en el funcionamiento de las herramientas utilizadas. De esta forma, para evitar estas múltiple lecturas, se modificó la herramienta Scancode Toolkit para poder analizar el contenido de cada fichero directamente, en lugar de leerlo dentro del código de esta herramienta.

4.2.2 Procesado de estadísticas

En esta sección se describirán los procesos de obtención y envío de los datos estadísticos de los proyectos, además de su filtrado.

Tablas de la base de datos

La obtención de los datos de la base de datos de MySQL se realizará mediante el uso de los modelos que nos da la funcionalidad de Django. De esta manera, al acceder a los recursos /all,

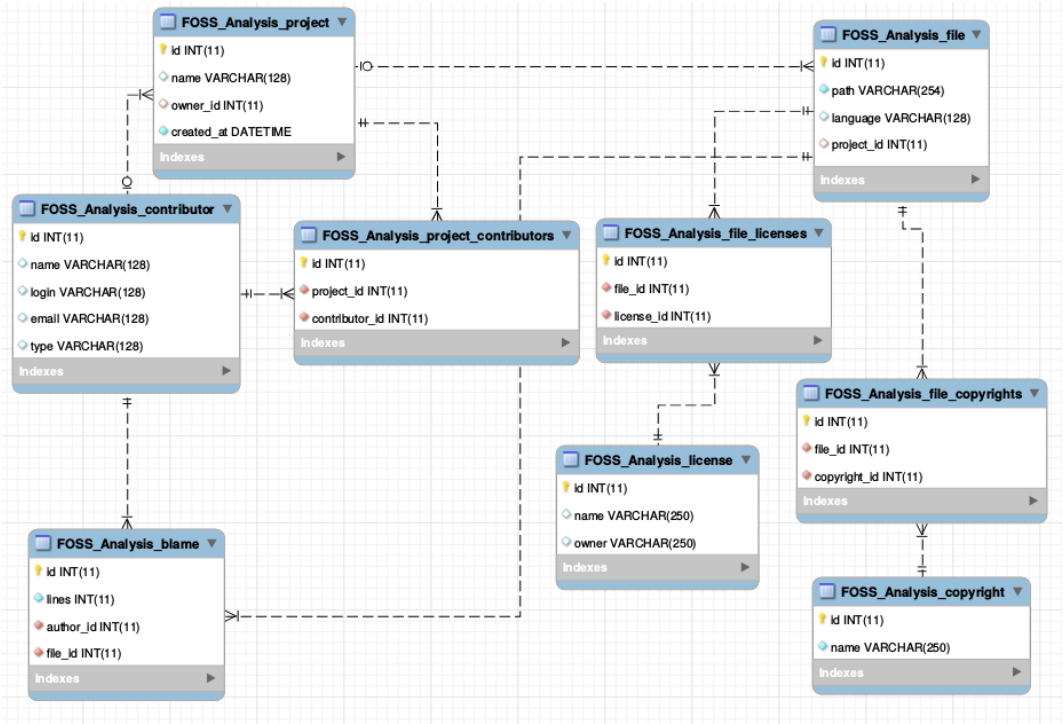

Figura 4.2: Distribución de la base de datos

/user o /legal, se realizarán las interacciones necesarias con la base de datos, gracias a la sencilla interfaz que nos proporcionan estos modelos.

Es necesario destacar que al tratarse de una base de datos relacional, se tendrá que acceder a las distintas tablas, a las cuales pertenecen los datos de estas relaciones, antes de elaborar la respuesta.

La disposición de las tablas presentes en la base de datos, además de las distintas relaciones entre ellas, se pueden observar en la figura 4.2.

En esta figura observamos las siguientes tablas de la base de datos:

- **Project**: Esta tabla contiene el nombre y la fecha de almacenamiento del proyecto, además del dueño y los contribuidores del mismo, mediante unas relaciones de *Foreign Key* y *Many to Many* respectivamente con la tabla de *Contributor*.

- ***Contributor***: Donde se almacena el nombre, login e email del contribuidor. Teniendo además el par login, email, como clave única de los contribuidores. De forma que no podrán almacenarse dos contribuidores cuyos campos de login y email sean ambos iguales.

- **File**: Para almacenar los ficheros contenidos en cada proyecto utilizaremos esta tabla, donde se almacenan el *path* absoluto donde ha sido analizado el fichero y el lenguaje en que este está programado. Además de esto, esta tabla estará relacionada con las tablas Project, Copyright y License para determinar a que proyecto pertenece el archivo y que *copyrights* y licencias están contenidas en él.

- ***Blame***: En la tabla *Blame* se almacenarán las contribuciones por cada fichero que ha realizado cada usuario. Así, se almacenarán las lineas de código que cada contribuidor ha realizado, guardando en el campo *author*, el id correspondiente con el contribuidor, y así mismo con el fichero en el campo file.

- **Copyright**: Donde se almacena el nombre del copyright y a quien pertenece.

- **License**: Donde se guardan los nombres de las distintas licencias.

Queries

Una vez descrito el modo en que están almacenados los datos en las distintas tablas de MySQL, procedemos a describir las interacciones que se han de realizar con la base de datos para poder obtener y enviar los distintos datos.

Estadísticas de proyecto Para la obtención de los datos globales de cada proyecto, accederemos al recurso /analyze en cuya vista se realizará una *query* sobre la tabla de proyectos en la que se traerán además, los datos relacionados con la tabla de contribuidores.

Tras realizar esta *query* se procesarán los datos para que sean *serializables* y para que tengan un formato JSON aceptable para el front-end programado en JavaScript.

Estadísticas totales En el caso de la obtención del total de los datos almacenados en la base de datos, se realizarán tres *queries* distintas sobre la tabla File, de las que se obtendrán los datos relacionados de las licencias y *copyrights* de todos los ficheros, además de los datos del lenguaje de cada uno de ellos. Por último, se hará una petición sobre la tabla *Blame* que devolverá el número de lineas totales realizadas por cada autor en los distintos ficheros.

Sobre estos datos se realizará una suma, mediante el método *count* de SQL, para cada valor diferente, gracias al método de la interfaz que aporta Django con el mismo nombre. Obteniendo

así el número total de lineas realizadas por cada autor y el número total de *copyrights*, licencias y lenguajes diferentes.

Por último se procesarán los datos para adaptarlos al formato JSON, y además al formato de los datos que tendrán que ser introducidos en los gráficos de D3.

Estadísticas de usuario Para obtener los datos concretos de un usuario, sobre la tabla *Blame* se realizarán tres peticiones para obtener el número total de ficheros programados en cada lenguaje, y que contienen cada tipo de copyright y licencia, filtrándolos por el usuario determinado. Además se harán dos *queries* en las que se obtendrán todos los nombres de proyectos en los que el usuario participa como dueño o como colaborador del mismo.

Este filtrado se realizará mediante el método where del lenguaje SQL o, en este caso, del método *filter* de la API de Django *models*.

Al igual que en el caso anterior, se realizará un procesado final antes de enviar la respuesta con los datos obtenidos filtrados y sumados.

Estadísticas legales Por último se obtendrán todos los ficheros filtrados por licencias o por *copyrights*.

En este caso se realizará un filtrado como en el caso anterior, pero con la diferencia de que, en este caso, se filtrará de manera no estricta, es decir, valdrá con introducir una palabra contenida en los *copyrights* o licencias del fichero, para que este sea devuelto en la respuesta.

4.3 Front-end

A continuación se pasará a explicar el funcionamiento y diseño de la parte del cliente de la aplicación.

Como ya se ha comentado antes, esta sección del proyecto está implementada con el *framework* AngularJS, el cual aporta una estructura al código que se pasará a explicar a continuación.

4.3.1 Estructura AngularJS

El *framework* Angular nos permite s permite dividir nuestro código por módulos, los cuales permitirán diferenciar de manera adecuada las distintas funcionalidades que pueda tener nuestra

web. Los módulos que forman esta aplicación son los siguientes:

- **Main**: Permite la selección de las distintas secciones de la página sin necesidad de recargar o acceder a una nueva *url*, convirtiendo así la página en una Single Page Application.
- **Analysis**: Accediendo a la API implementada en el servidor Django, obtiene los datos analizados del proyecto, y los almacena de modo que puedan ser usados por la plantilla HTML correspondiente a este módulo y por sus gráficos.
- **Statistics**: En este módulo al igual que en el de Analysis se obtendrán los datos de la API del servidor, y se mostrarán en tres plantillas diferentes ya sean datos totales, de usuario o legales.
- **Navbar**: Es el módulo que permite controlar el funcionamiento de la barra de navegación, permitiendo mostrar las secciones que contiene la página.
- **Chart**: En el módulo de Chart se crearán los gráficos que nos servirán para representar los datos de los módulos de Analysis y Statistics. Por tanto estos dos módulos tendrán una dependencia con el de Chart.

Todos estos módulos se inyectan en las dependencias del módulo principal de la aplicación, que será el módulo app.

4.3.2 Funcionamiento

En los distintos módulos se establecerán una serie de directivas Angular que permitirán el intercambio de los distintos parciales o plantillas HTML, gracias a la colocación del método ng-view en un elemento del DOM. Mediante este método, en este elemento se colocarán los contenidos de las plantillas referenciadas dentro de cada directiva.

Una vez mostrada la plantilla correspondiente a cada directiva, se permitirá mediante distintos formularios, la realización de peticiones de datos a la base de datos, tal y como se ha explicado anteriormente.

Tras obtener la respuesta a las peticiones, esta se almacenará en campos del elemento $scope de Angular. Permitiendo, mediante el Two Way Data Binding ya mencionado, la actualización y representación de los datos en cada plantilla.

Cabe mencionar, que en el caso de los gráficos de la biblioteca D3, es necesario actualizar el gráfico con los nuevos datos. Este proceso no es automático, por tanto se ha hecho uso del método $watch para añadir un manejador al evento de actualización de los datos. Así, cuando este evento se produzca, se llamará a una función que generará el gráfico de nuevo y lo actualizará con los nuevos datos.

4.4 Adaptación de la información

Debido a las distintas tecnologías utilizadas en el back-end y en el front-end, donde se utilizan Python y JavaScript respectivamente, se producen ciertas incompatibilidades entre los formatos de uno y otro extremo.

Estos conflictos no suponen un problema a la hora de realizar ambas partes por separado, pero es necesario atajarlos ya que entre uno y otro punto se produce un intercambio de información.

En este caso, al ser JavaScript, en el front-end, el consumidor de la información, será necesario que antes de su envío, la información sea procesada para darle un formato JSON. Este formato implica tener una colección de elementos con una clave y un valor asociado a ella.

Este formato permitirá una adecuada *serialización* desde el servidor al cliente para el correcto envío de la respuesta, y que en el front-end no se requiera ningún procesado especial de la respuesta, más que el de un elemento JSON propio de JavaScript y su posterior adaptación para su representación.

La adaptación de la información supone un elemento clave también para denominar a la API desarrollada como una API REST, puesto que esto conlleva una homogeneidad entre los recursos y el formato JSON permite dar esta homogeneidad a las respuestas de estos recursos.

Capítulo 5

Resultados

Una vez enumerado el proceso de análisis, se procederá a mostrar los resultados obtenidos del análisis de proyectos pertenecientes a repositorios Github de mayor o menor relevancia.

5.1 Interfaz de usuario

Primero se mostrará la interfaz de usuario de la aplicación en la que se representarán los datos que se enumerarán en las secciones siguientes.

En los dos primeros gráficos se mostrarán los lenguajes utilizados y los colaboradores más activos tal y como se puede ver en la figura 5.1.

En la segunda sección, mostrada en la figura 5.2 se representarán las licencias y *copyrights* presentes en el proyecto.

Y por último, en la tercera sección se verá una lista con el desglose de todos los ficheros con su datos individuales. Esta sección se muestra en la figura 5.3

Para representar unos resultados reales en esta interfaz, se analizarán distintos proyectos populares entre la comunidad de Free Open Source Software. Entre estos proyectos se han elegido los repositorios de herramientas como Docker, Ruby, Swift, React, Apache/Storm, Apache/Spark, Scala, Go, etc.

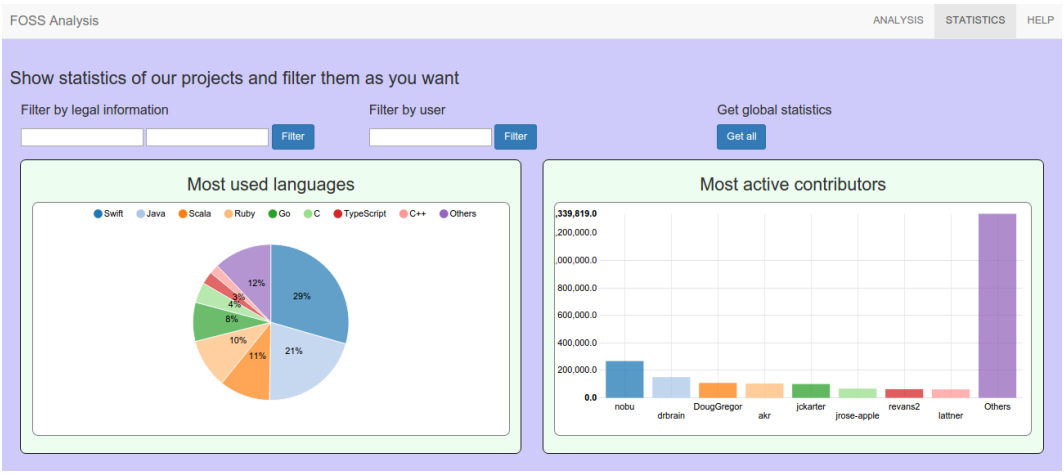

Figura 5.1: Gráficos de lenguajes y contribuidores

Figura 5.2: Gráficos de licencias y *copyrights*

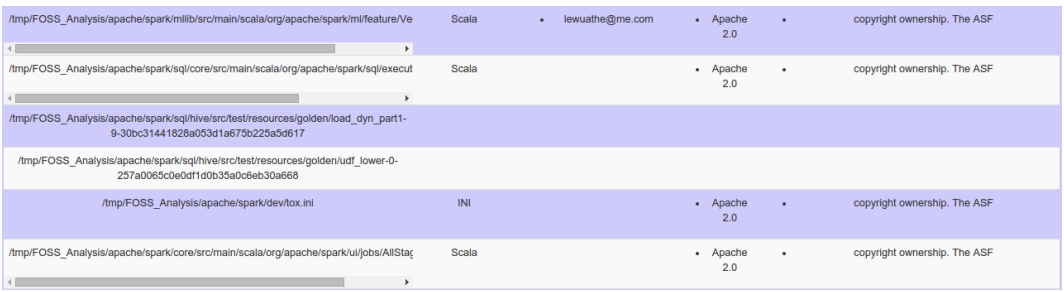

Figura 5.3: Lista de ficheros

5.2 Información general

En los distintos proyectos analizados se puede comprobar una serie de características comunes que varían muy poco de uno a otro. No obstante, la información obtenida de estas pequeñas diferencias nos sirve para analizar el entorno en que estos proyectos han sido realizados. En este apartado se describirán las propiedades generales que comparten todos estos proyectos.

Para analizar los repositorios se ha recurrido a la plataforma Github en la que se puede acceder a ellos. El hecho de que estén presentes de manera pública en una plataforma en la que el código es totalmente accesible, muestra las características Free Open Source de dichos proyectos.

Estas propiedades Free Open Source se aprecian también en la variedad de licencias utilizadas. Aunque no todos los proyectos utilicen las mismas licencias, la mayoría de ellas son permisivas y cumplen las cuatro libertades del software libre, de uso, de modificación, de distribución y de mejora.

Por otra parte, la mayoría de estos proyectos pertenecen a grandes empresas del sector tecnológico, que a pesar de ser lucrativas, han visto en el ecosistema Free Open Source el futuro del desarrollo software. Debido a la gran comunidad de desarrolladores presentes en este entorno, las empresas se benefician de la colaboración de programadores de todo el mundo para mejorar su software. De esta forma, observamos un número de contribuidores muy elevado, y por consiguiente, un crecimiento de las dimensiones del proyecto proporcional a este número.

Esta es otra característica fundamental de todos los proyectos FOSS si quieren ser fructíferos, la comunidad. Por un lado las empresas se benefician de la contribución de distintos colaboradores a los cuales incluso podrían reclutar, y por otro, los contribuidores desarrollan tanto su carrera como sus habilidades programáticas.

Por último, además de todas estas peculiaridades de los proyectos FOSS, se ha de destacar la reutilización y mejora de herramientas y proyectos a la hora de desarrollar un nuevo software. Todos los proyectos analizados, basan su funcionamiento en otro software, ya sean lenguajes de programación, herramientas web, herramientas para la distribución de aplicaciones, etc. Lo cual hace de la colaboración de la comunidad, un factor imprescindible para que un proyecto de software libre tenga éxito.

A continuación se enumerarán las cualidades específicas de diferentes proyectos analizados.

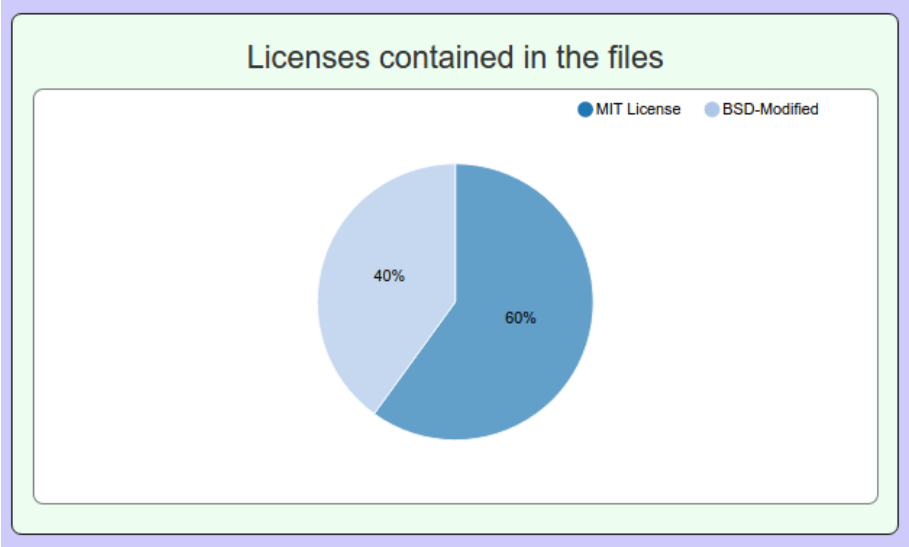

Figura 5.4: Licencias presentes en AngularJS

5.3 Información específica de los proyectos

En esta sección se enumerarán los resultados obtenidos para tres tipos diferentes de proyectos. Para herramientas de desarrollo software como son Angular y Docker, para lenguajes de programación como Ruby y Swift y por último, para herramientas de Big Data.

5.3.1 Herramientas para el desarrollo software

Docker y Angular son dos herramientas de software libre, cuyas funciones son de plataforma para aplicaciones distribuidas y *framework* web respectivamente.

Estas dos herramientas difieren en su funcionalidad, pero no tanto en sus características Free Open Source. Ambos proyectos usan en la práctica la licencia BSD-Modified mayoritariamente, como se muestra en las figuras 5.4 y 5.5. Ya que aunque AngularJS presenta un 60bajo la licencia MIT, al ser esta una licencia muy permisiva y ser compatible con otras licencias, en Angular se hace uso de la licencia BSD la cual es más restrictiva que la MIT License.

Los permisos que se otorgan a ambos proyectos mediante la licencia BSD-Modified son los siguientes:

- **Uso comercial**: Permiso para utilizar el código con propósitos comerciales y lucrativos.
- **Modificación**: Permiso para modificar y crear proyectos derivados del código original.

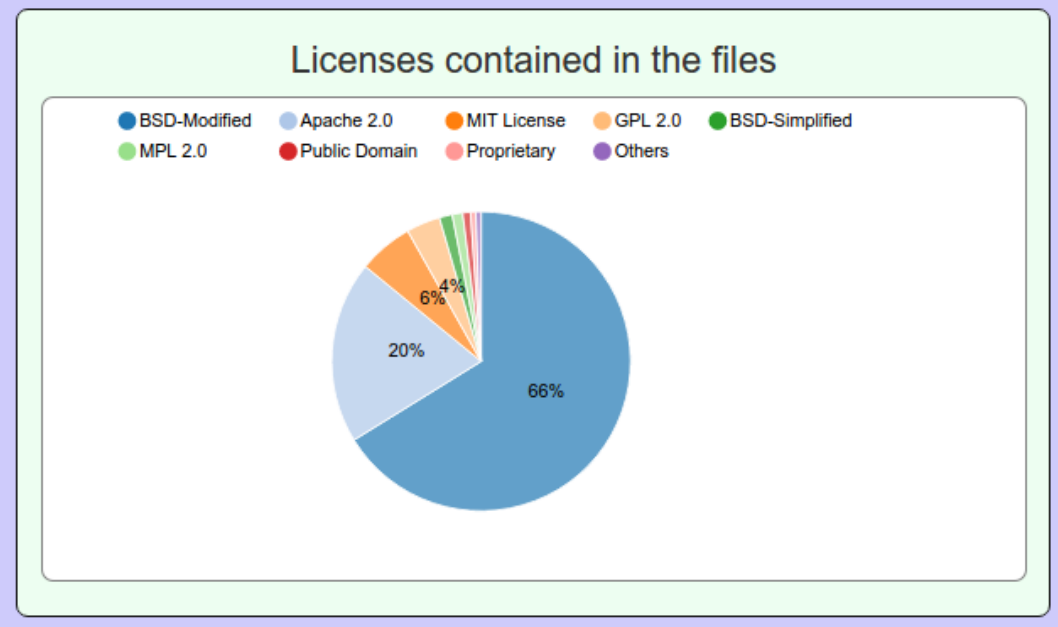

Figura 5.5: Licencias presentes en Docker

- **Distribución**: Permiso para distribuir el código original o cualquier derivado del mismo.
- **Compatibilidad de licencias**: La licencia BSD es compatible con cualquier otra licencia de software libre.

Además, la licencia MIT otorga permisos específicos de utilizar AngularJS para uso privado. Por otro lado, la licencia BSD-Modified restringe el uso del código de los siguientes modos:

- **Uso de *Trademarks***: No se permite usar el nombre de empresas o marcas registradas en el código para ratificar o beneficiar la popularidad de un trabajo derivado del original.
- **Responsabilización**: No se responsabilizará al autor del software por ningún perjuicio ocasionado.
- **Inclusión de copyright**: Se deberá conservar siempre en el código original y en su uso en cualquier código derivado, el copyright original del mismo.
- **Inclusión de licencia**: Se deberá conservar siempre el texto completo de la licencia BSD.

Estas restricciones salvo la del uso de *Trademarks* se aplican también a través de la licencia MIT, en el caso de Angular. Pero al usarse las licencias MIT y BSD conjuntamente, se aplican las restricciones de ambas.

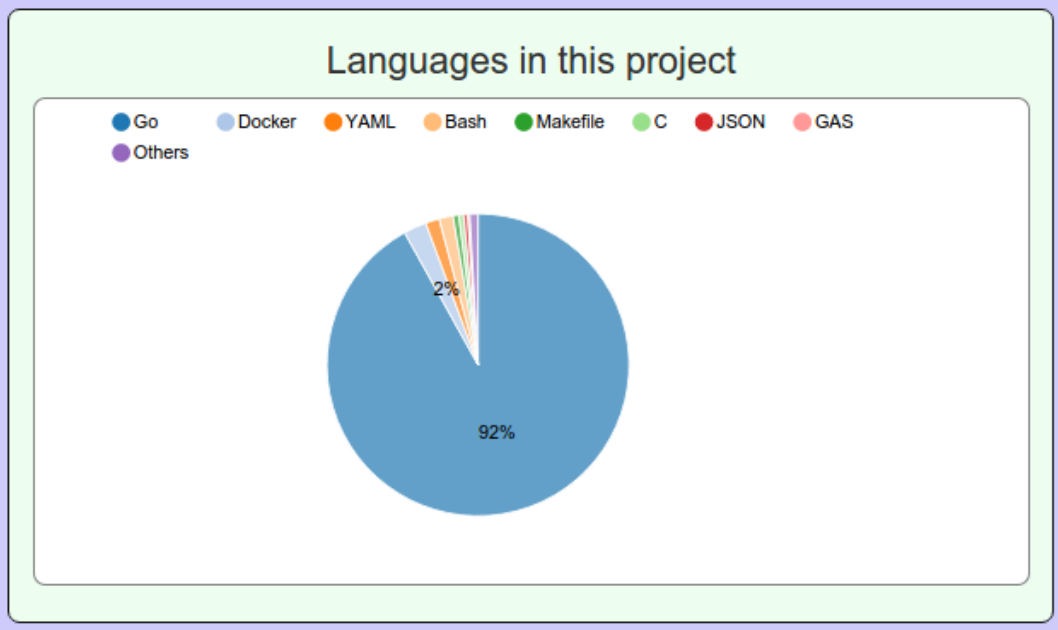

Figura 5.6: Lenguajes utilizados por Docker

Además de las licencias, cabe destacar el uso de determinados lenguajes en estas herramientas. Por un lado, Docker utiliza Go como lenguaje principal tal y como se muestra en la figura 5.6, este lenguaje es también Free Open Source, contando con una licencia BSD modificada, que le otorga a Google, su distribuidor, un derecho de patente sobre el lenguaje, pero sin que esto suponga un recorte en los derechos de utilización de Go por parte de Docker.

Por otro lado, en Angular los principales lenguajes de programación son TypeScript y Dart, que utilizan licencias Apache y BSD respectivamente, tal y como se ve en la figura 5.7. En este caso, los dos, al igual que en el caso de Docker, tienen licencias libres. Y Apache al igual que en el caso anterior, otorga derecho de patente a Microsoft sobre TypeScript, sin que esto suponga una pérdida de derechos por parte de Google sobre Angular.

5.3.2 Repositorios de lenguajes de programación

Además del análisis de herramientas concretas y de más alto nivel, se decidió analizar también proyectos correspondientes a lenguajes de programación para, además de ver sus características legales en cuanto a si son libres y abiertos, conocer los lenguajes y herramientas en los que se basan, a su vez, estos lenguajes.

Los lenguajes seleccionados, por ser relativamente modernos y con un nivel de abstracción

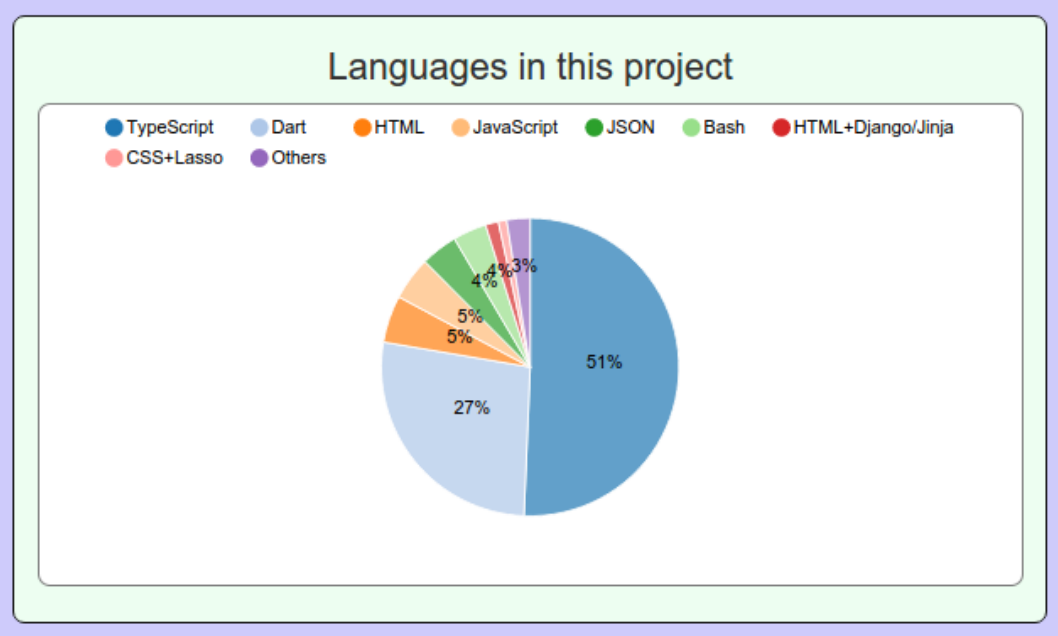

Figura 5.7: Lenguajes utilizados por AngularJS

superior a los que daban lenguajes como C o C++, son Ruby y Swift.

Ruby por su parte es un lenguaje diseñado y desarrollado por Yukihiro Matsumoto en 1995 y que tal y como se puede ver en la figura 5.8, consta de una serie de ficheros escritos en Ruby, que corresponden con librerías y funcionalidad específica del lenguaje. Pero una vez observamos el código del propio lenguaje, podemos comprobar que está programado en C. Esta situación se da en la mayoría de lenguajes modernos, que se basan en C, C++ y sus derivados, debido a la libertad de uso que estos proporcionan, permitiendo al desarrollador controlar todos los aspectos de su software de manera muy cercana al sistema operativo. Utilizando este lenguaje como base, se otorga al nuevo lenguaje, en este caso Ruby, un mayor nivel de abstracción que a pesar de restringir la libertad en su uso, proporciona una mayor simplicidad al mismo.

Por otro lado tenemos el caso de Swift. Swift es un lenguaje de programación desarrollado por Apple que fue software propietario hasta su versión 2.2 y que al igual que Ruby tiene una librería estándar programada en Swift y cuya base está escrita en C++.

Tal y como se puede observar en la figura 5.9, el 100otorga derechos de patente al desarrollador. En casos anteriores ya ha aparecido esta licencia. En aquellos casos, la licencia no afectaba a la herramienta que utilizaba ese lenguaje de programación porque se restringía únicamente a su uso, y en ningún momento modificaba el código de dicho lenguaje. En caso de

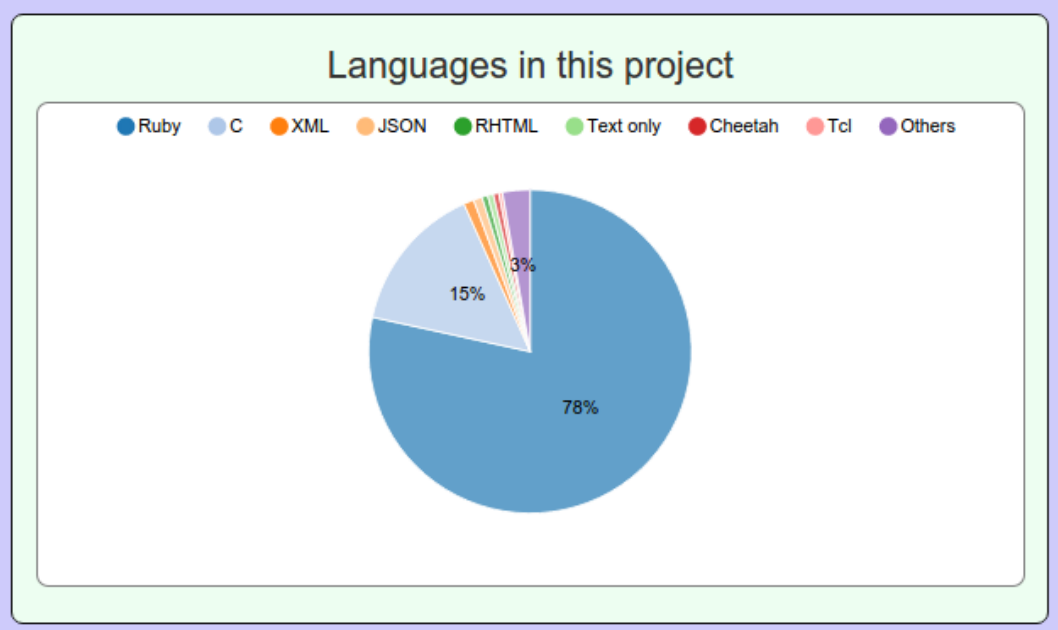

Figura 5.8: Lenguajes utilizados en el proyecto Ruby

que se quisiera modificar Swift, cualquier modificación realizada sobre el código supondría que el colaborador tiene que dar una licencia a todos los usuarios, de cualquier patente que pueda poseer sobre esa colaboración.

Esta restricción mitiga el problema de las patentes en el desarrollo de software libre, evitando que, si alguien intenta apropiarse de una patente mediante un trámite legal, sean revocados sus derechos de patente sobre todo el software o sobre las secciones de software en las cuales colaboró.

Este tipo de licencias están proliferando más últimamente para ayudar a grandes empresas como Apple o Google, a sentirse más cómodas al hacer que su software pertenezca a la comunidad Free Open Source.

5.3.3 Herramientas de Big Data

Por último, se mostrarán a continuación los resultados obtenidos del análisis de dos proyectos de software de mayor actualidad, los proyectos de Apache Spark y Apache Flink. Estos proyectos se engloban dentro del ámbito del desarrollo software llamado Big Data y se utilizan para el procesado de grandes cantidades de *streams* de datos, para aprendizaje máquina, etc.

Al ser Big Data un tema muy actual, estas herramientas contienen, además de por pertenecer

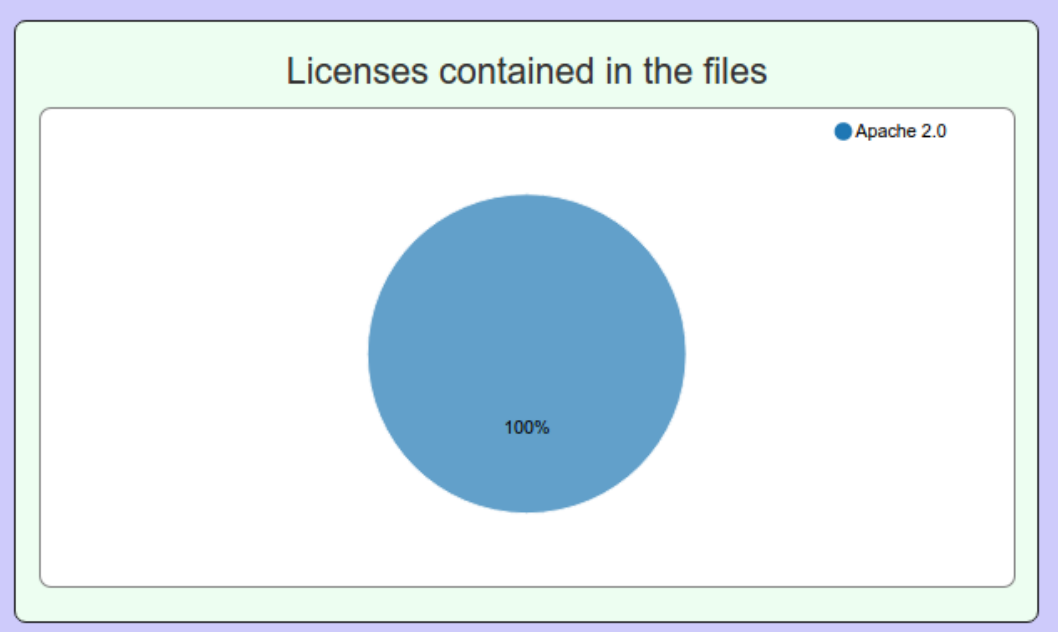

Figura 5.9: Licencias presentes en el proyecto Swift

a la Apache Software Foundation, licencias Apache 2.0 en la gran mayoría de su código. Esto aplica a estos proyectos el derecho de patente para todos sus colaboradores.

Por otro lado, si analizamos los lenguajes utilizados en estos dos proyectos, comprobamos que ambos contienen, en su mayoría, código escrito en Java y Scala, estos dos lenguajes son muy comunes en el análisis de grandes cantidades de datos, por su simplicidad y escalabilidad. Scala se ha convertido en uno de los lenguajes de programación principales para el desarrollo de herramientas de Big Data.

5.4 Lenguajes utilizados

En la sección anterior se han explicado los datos obtenidos para tres conjuntos diferentes de proyectos de software libre. A partir de ese análisis podemos estudiar el uso de diferentes tipos de lenguajes en función de las cualidades y objetivos del proyecto analizado.

En el caso de proyectos que implementan nuevos lenguajes de programación, se observa el uso generalizado de C, C++ y sus derivados, al ser estos lenguajes de programación de bajo nivel, que permiten implementar interfaces entre los procesos y el control de la memoria propios de lenguajes como C, para que el nuevo lenguaje alcance un mayor grado de abstracción.

Si observamos, por otra parte, proyectos relacionados con herramientas de más alto nivel de

abstracción, como *frameworks*, plataformas para la distribución de aplicaciones, herramientas de Big Data o *machine learning*, requerirán del uso de lenguajes de programación de alto nivel como pueden ser Ruby, Python, Go, Scala, JavaScript e incluso Java.

Además, en función de las necesidades de la herramienta como ya se ha mostrado, predominará el uso de un lenguaje u otro. De esta forma, para *frameworks* web en el front-end se suele utilizar JavaScript, para análisis de Big Data se suele utilizar Scala por su escalabilidad, y para proyectos de *machine learning* se suele utilizar Python entre otros por el gran número de librerías que presenta el lenguaje para este propósito.

A continuación se enumerarán las consecuencias que podemos extraer de los datos anteriormente recogidos.

5.5 Consecuencias extraídas

Tras la obtención de los datos anteriores, se pudo comprobar la importancia del software libre en el desarrollo de los nuevos proyectos de software, ya sea para empresas con ánimo de lucro o desarrolladores particulares. El mundo del desarrollo software se basa en la comunidad, en la reutilización de herramientas y en el intercambio de conocimientos para alcanzar los objetivos marcados en cada proyecto. Por tanto, se debería fomentar el uso de software basado en estos principios.

Al analizar los proyectos mencionados en la sección 5.3 se ha obtenido una mejor visión de las libertades que proporciona el software libre y los matices que existen dentro del mismo, porque aunque todo el software analizado sea libre, las diferentes licencias, otorgan distintas cualidades a cada proyecto.

La principal cualidad diferenciada entre las licencias es el uso del derecho de patente [7]. Este derecho, como ya se ha explicado anteriormente, impide al desarrollador apropiarse de un proyecto determinado mediante acciones legales, dando así una cualidad importante a los nuevos proyectos de software libre que harán que este sector se desarrolle aún más y que empresas de software propietario o que busquen lucrarse del software que producen, como pueden ser Apple y Google, inviertan y se involucren aún más en la comunidad de proyectos FOSS.

La presencia de estas compañías podrá influenciar a otras grandes empresas, y a futuras potencias del sector, a seguir colaborando y hacer que el software libre se generalice en el

mundo tecnológico, de lo que se podrán beneficiar usuarios, empresas y desarrolladores por igual.

Software propietario

En el caso de Apple, como ya se ha mencionado, se ha comenzado a trabajar en proyectos de software libre como Swift, que pretende sustituir a Objective-C. El principal problema de este lenguaje de programación es la falta de interacción con los desarrolladores que lo utilizan. Por esta razón Swift en sus últimas versiones decidió desarrollarse de manera abierta y libre, para que la interacción de los distintos contribuidores permitiera desarrollar el lenguaje y hacer que aumente en popularidad y facilidad de uso, y que además se puedan desarrollar aplicaciones con este lenguaje de manera más sencilla y óptima para el usuario.

Reutilización

Por último cabe mencionar que todos los proyectos actuales se caracterizan por su alto nivel de abstracción, y por tanto necesitan del uso de herramientas y lenguajes de programación de software libre desarrollados con anterioridad. Esta característica satisface la necesidad de reutilización del código libre y abierto, presente en estos proyectos, y promueve además la creación de nuevas herramientas que satisfagan nuevas necesidades.

Capítulo 6

Conclusiones

En esta sección se explicarán las determinaciones que se extraen de la finalización de este proyecto, valorando el cumplimiento de los objetivos puestos inicialmente y las lecciones aprendidas mediante su realización y el proceso de investigación.

6.1 Consecución de objetivos

Tras la finalización del proyecto, se puede determinar que todos los objetivos previamente fijados han sido realizados con éxito y que por tanto se ha alcanzado satisfactoriamente el objetivo principal de realizar el análisis de distintos proyectos FOSS para el posterior análisis y comprensión de los datos obtenidos como se ha realizado en la sección 5.5.

A continuación se enumerarán los resultados obtenidos para cada objetivo enumerado en el capítulo 2:

- Se consiguió utilizar la herramienta Scancode Toolkit para analizar distintos proyectos que sirvieran de ejemplo para los datos a almacenar en la base de datos.
- Se consiguió establecer la estructura de una aplicación Django con modelos que permitieran realizar la posterior interfaz con la base de datos.
- Se incluyó la herramienta Scancode Toolkit en el código presente dentro del proyecto Angular, para poder almacenar de manera más adecuada los datos obtenidos del análisis.
- Se desarrolló una API REST que interactúa con la herramienta Scancode Toolkit y la API de Github para devolver en formato JSON los datos correspondientes a cada proyecto.

- Se almacenaron distintos proyectos de importancia dentro del entono Open Source para que su análisis pudiera servir para comprender mejor su ecosistema.

- Se realizó la estructura Angular en el lado del cliente de la aplicación, incluyendo en ella, distintas directivas para permitir la representación de los datos mediante gráficos de la librería nvd3.js.

Es necesario mencionar también que aunque el funcionamiento del analizador de proyectos es adecuado, es necesario que se mejore su rendimiento en trabajos futuros. Se trató de analizar proyectos como el repositorio de Linux, pero por su tamaño, el analizador tardaba demasiado, lo que producía que en el momento de almacenar el proyecto en la base de datos, el servidor de MySQL se hubiera cerrado.

6.2 Aplicación de lo aprendido

Para poder enumerar los conocimientos obtenidos en la carrera y que se han puesto en práctica en este proyecto, se enumerará cada asignatura y los conocimientos aplicados que se aprendieron en dicha asignatura.

1. **Servicios y Aplicaciones Telemáticas**: El desarrollo de aplicaciones web mediante el *framework* web Django.

2. **Ingeniería de Sistemas de Información**: La realización y planteamiento de proyectos de software y la utilización y comprensión de las bases de datos.

3. **Desarrollo de Aplicaciones Telemáticas**: La implementación de interfaces de usuario con cierto nivel de complejidad, tanto visual como funcional.

4. **Sistemas Operativos**: Aplicación de programación concurrente y la comprensión de procesos de sistemas operativos.

5. **Ingeniería de Sistemas Telemáticos**: Aplicación de programación orientada a objetos.

6.3 Lecciones aprendidas

Además de lo estudiado en la carrera, este proyecto me ha permitido aprender acerca de los siguientes temas:

1. Plantear proyectos de ingeniería desde cero, y adaptar y documentar el proceso de desarrollo en función de los avances conseguidos.

2. Investigar y comparar las distintas herramientas presentes para realizar una determinada tarea, y escoger aquella que aporte los mayores beneficios al proyecto.

3. Analizar el funcionamiento del software para mejorar su implementación y el rendimiento que genera, además de la utilización de herramientas de *profiling* para determinar, mediante datos numéricos, la eficiencia del código.

4. Desarrollar todo el *stack* de una aplicación web, y estructurar adecuadamente el código tanto en el cliente como en el servidor.

5. Desarrollar software que permita atajar problemas de la vida real, analizando datos presentes en la red.

6. Conocer y obtener una visión más profunda de los proyectos de software libre además de concienciarse del entorno de comunidad que encontramos en estos proyectos.

En definitiva, este proyecto me ayudó a sentar las bases de un buen ingeniero, de manera que pude utilizar mis habilidades para resolver los problemas que fui encontrando durante el desarrollo de este proyecto.

6.4 Trabajos futuros

Aunque los objetivos planteados en este proyecto se han completado, ningún proyecto software llega a cumplir a la perfección las ideas preliminares que se tienen sobre él, al menos en su primera versión.

Este proyecto se centró en el análisis de proyectos concretos, que puedan ser un buen ejemplo de las características generales de los proyectos de software abierto y libre, sin embargo,

sería una buena mejora si se realizara un análisis masivo de un mayor número de proyectos y que obtengan un mayor número de datos diferentes.

Esta mejora supondría la utilización de metodologías de Big Data para el filtrado y la obtención de los datos determinantes para nuestro análisis, poblando la base de datos con información relevante para el análisis deseado.

Además, para mejorar el funcionamiento del análisis y la obtención de los datos se deberían aplicar técnicas de computación de alto rendimiento para que el procesado de proyectos de grandes dimensiones no suponga un tiempo de análisis demasiado alto.

6.5 Valoración personal

El hecho de que este proyecto se centrara en el software libre y la posibilidad de analizar proyectos importantes que se usan comúnmente a la hora de desarrollar cualquier proyecto software, incluyendo algunas de las que he utilizado en la carrera y en mi tiempo libre, supuso un aliciente para mí a la hora de escoger este proyecto.

Las ideas y libertades que representa el software libre, representan en mi opinión unos ideales que deberían promoverse y darse a conocer. Todo el mundo debería tener derecho a acceder a cualquier tipo de software, ya que nadie puede ser su dueño o inventor absoluto. En su gran mayoría, todos los proyectos de software se derivan de otros desarrollados en el pasado, y que permitieron sentar las bases de lo que se esta desarrollando ahora, así como los actuales proyectos sentarán las bases del software del futuro.

Por esta razón se debería tener una mentalidad global a la hora de desarrollar software, y esta es la finalidad intrínseca de este proyecto, mentalizar y mostrar a los posibles lectores de esta memoria, los beneficios que supone para todos los elementos del entorno de desarrollo de software, ya sean programadores, empresas o usuarios, el uso y promoción del Free Open Source Software.

Apéndice A

Manual de usuario

Para poder desarrollar este proyecto se tuvo que utilizar Django, que se puede instalar con el siguiente comando.

```
$ pip install -e django
```

Posteriormente, se inicializó un proyecto y aplicación Django escribiendo en la terminal, los siguientes comandos:

```
$ django-admin startproject mysite
$ python manage.py startapp myapp
```

Posteriormente para arrancar la aplicación y ver el resultado se utilizará el siguiente comando.

```
$ python manage.py runserver
```

Y además, para poder migrar los cambios realizados sobre los modelos a la base de datos, se aplicarán los siguientes comandos de manera consecutiva.

```
$ python manage.py makemigrations
$ python manage.py migrate
```

Además de Django, se utilizaron librerías python externas cuya instalación se realizó mediante los siguientes comandos:

```
$ pip install pygments
$ pip install gitpython
```

También, se tuvo que modificar el archivo settings.py del proyecto Django, para utilizar un tipo de base de datos MySQL, añadiendo esta línea.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        ...
    }
}
```

Por último, estos son los repositorios Github de la herramienta Scancode Toolkit y de este proyecto.

```
https://github.com/nexB/scancode-toolkit
https://github.com/jperaltar/FOSS_Analysis
```

Apéndice B

Planificación temporal

En esta tabla se muestra la planificación temporal general del proceso de realización del proyecto.

	Marzo 2016	Abril 2016	Mayo 2016	Junio 2016
1 – 7	Planteamiento del proyecto	Creación de las primeras vistas que conformarán la API de la aplicación		Finalización de la representación de datos mediante gráficos D3
8 – 14	Prueba de la herramienta Scancode Toolkit y análisis de los primeros proyectos.	Correción y optimización del análisis de proyectos, tanto en tiempo de procesado como en eficiencia	Estructuración del código del cliente mediante el framework AngularJS	Realización del análisis final de proyectos relevantes y desarrollo de la memoria y estracción de resultados
15 – 21	Estructuración del proyecto Django y creación de los modelos para la base de datos	Terminación de la API de la aplicación, añadiendo recursos para datos filtrados y específicos	Adaptación de formatos entre el servidor Python y el cliente JavaScript	
22 – 30/31	Adaptación de la herramienta Scancode Toolkit al funcionamiento de la aplicación Django	Inicio de la sección de estado del arte de la memoria	Inclusión de directivas con gráficos D3 para la representación final de los datos	

En general se siguieron los tiempos marcados aunque por diversos imprevistos, en ocasiones se tuvo que usar parte del tiempo especificado en cada semana para la realización de cada tarea, en la corrección o mejora de distintas secciones del proyecto.

Además, esta tabla incluye únicamente el periodo de tiempo en desde que se comenzó a desarrollar el proyecto hasta el momento de su finalización. El proceso inicial de planteamien-

to del problema y la elección del enfoque que se iba a tomar para definir el objetivo final y secundario del problema, tuvo lugar durante todo el año anterior a la iniciación, justo desde el momento en que se comenzó a plantear la realización de este proyecto, con el Profesor Gregorio Robles Martínez.

Bibliografía

[1] AngularJS. Web oficial de angularjs. https://angularjs.org/.

[2] D3.js. Web oficial de la librería d3.js. https://d3js.org/.

[3] Django. Web oficial de django project. https://www.djangoproject.com/.

[4] FreeSoftwareFoundation. Web oficial de la free software foundation. http://www.fsf.org/.

[5] GitPythonDevelopers. Repositorio en github de la librería gitpython. https://github.com/gitpython-developers/GitPython.

[6] HTML5Doctor. Web de html5 doctor. http://html5doctor.com/.

[7] G. Markham. Hacking for christ blog. http://blog.gerv.net/2013/02/an-introduction-to-modern-open-source-licence-patent-clauses/.

[8] MozillaDeveloperNetwork. Web oficial de mdn. https://developer.mozilla.org/es/docs/Web/JavaScript.

[9] nexB. Repositorio en github de la herramienta scancode toolkit. https://github.com/nexB/scancode-toolkit.

[10] OpenSourceInitiative. Web oficial de la open source initiative. https://opensource.org/.

[11] Oracle. Web oficial de mysql. https://www.mysql.com/.

[12] Python. Web oficial de python programming language. https://www.python.org/.

[13] J. Serra. Relational databases vs non-relational databases. http://www.jamesserra.com/archive/2015/08/relational-databases-vs-non-relational-databases/.

[14] TwitterBootstrap. Web oficial de bootstrap. http://getbootstrap.com/.