UNIVERSIDAD
REY JUAN CARLOS

INGENIERIA DE TELECOMUNICACIONES

Curso Académico 2014/2015

Proyecto Fin de Carrera

DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN WEB PARA EL ANÁLISIS DE PROYECTOS DE SOFTWARE LIBRE

Autor : Almudena Mateos López

Tutor : Dr. Gregorio Robles

Proyecto Fin de Carrera

DISEÑO E IMPLEMENTACIÓN DE UNA APLICACION WEB PARA EL ANÁLISIS DE PROYECTOS DE SOFTWARE LIBRE

Autor : Almudena Mateos López

Tutor : Dr. Gregorio Robles Martínez

La defensa del presente Proyecto Fin de Carrera se realizó el día de de 2015, siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de de 2015

Dedicado a

mis padres y hermana

"Nuestra recompensa se encuentra en el esfuerzo y no en el resultado.

Un esfuerzo total es una victoria completa."

Mahatma Gandhi

II

Agradecimientos

Y por fin llego el día que tanto tiempo he estado esperando. Hoy se acaba la etapa más dura y bonita que he vivido hasta ahora. Han sido muchos los años de sufrimiento, de horas en la biblioteca, de lagrimas.. pero todo tiene su recompensa, por fin acabe la carrera! Lo mejor de este largo camino es la gente con la que lo he compartido, compañeros, profesores y grandes amigos, aquellos que te llegan al corazón y se quedan para siempre. A ellos les debo que el haber llegado hasta aquí haya sido más llevadero.

A mi tutor, Gregrorio, gracias por implicarte tanto con los alumnos en todas tus asignaturas y vivir esta profesión con tanto entusiasmo.

A Carlitos, por tu gran amistad y apoyo, eres una gran persona; a mi rubio, Larraco, por tu eterna sonrisa y tu gran corazón, no cambies nunca; a Victor, por todos esos buenos y malos momentos y esas largas conversaciones con Carlos resolviendo el mundo mientras hacíamos descansos, las echaré de menos; a Alberto, Joel, Raquel, Sergio, Victor, Celia, mis tronquetas y toda la gente que me dejo. Gracias por todo, tenéis un huequito en mi corazón para siempre.

A Fran, gracias por tu apoyo incondicional. Ya lo sabes todo.

Y por supuesto a mi familia, que siempre esta ahí. Abuela, por fin acabé, gracias por todas las velas y tu cariño, ya sabes cuanto te quiero. Papá, fuiste el que más me apoyo cuando decidí tomar este camino, gracias por confiar en mi. Tata, mi alma gemela, gracias por calmarme cuando lo he necesitado. Pero sobre todo, tanto este proyecto como mi título, se lo quiero dedicar a mi madre. Mamá, se podría decir que la mitad de mi carrera te la debo a ti, por todo el apoyo, sufrimiento, por cada lagrima y esforzarte tanto por nosotras, te lo debo todo. GRACIAS.

Resumen

Este Proyecto Fin de Carrera consiste en el diseño e implementación de una aplicación web capaz de analizar proyectos de Software Libre alojados en la plataforma de desarrollo colaborativo GitHub. Se pretende crear una aplicación que analice los repositorios de los proyectos que hay en dicha plataforma y sea capaz de mostrar a través de una interfaz intuitiva métricas e indicadores que detallen la evolución de cada proyecto, respondiendo así a preguntas lógicas en el desarrollo de cualquier software.

A pesar de que el Software Libre está en pleno crecimiento y cada día es más usado por parte de particulares y empresas que apuestan por esta filosofía y quieren ahorrar costes, el análisis de este tipo de Software es casi desconocido. Por tanto, crear una herramienta capaz de ofrecer estadísticas y adquirir el conocimiento necesario para saber interpretarlas es de gran importancia.

Entendemos que conocer la comunidad de desarrolladores que hay detrás de cada aplicación es vital a la hora de evaluar el futuro de dicho software y su ciclo de vida, por lo que en este proyecto nos hemos centrado más en realizar métricas e indicadores que analicen el comportamiento de dichos contribuidores a lo largo de todo el desarrollo del Software. De esta manera podremos conocer datos tan importantes como el estado actual del proyecto, de que manera está contribuyendo su comunidad o por cuantos miembros está formada.

La aplicación que hemos creado en este Proyecto Fin de Carrera utiliza los repositiorios que hay alojados en GitHub, una de las plataformas más utilizadas por los programadores para almacenar los cambios que se realizan durante el desarrollo de un Software y poder compartirlos

con terceros desarrolladores de una manera muy cómoda. Esta plataforma ofrece la posibilidad de que cualquier persona pueda acceder a estos repositorios y evaluar así su información.

Para la construcción de nuestro proyecto hemos extraído dicha información, a través de la herramienta CVSAnalY, y la hemos almacenado en el servidor de base de datos de Software Libre más popular, MySQL. Ayudándonos por el lenguaje de base de datos SQL, se ha realizado una fase de minería de datos para poder mostrar a través de una página web el resultado de las estadísticas obtenidas.

Para procesar todos los datos y realizar la lógica de la aplicación nos hemos apoyado en el framework Django y en Python, el lenguaje de programación principal de este proyecto. Finalmente, en la creación de la página web hemos utilizado numerosas tecnologías que facilitan la construcción de aplicaciones web dinámicas e intuitivas, tales como HTML, CSS, AJAX, JavaScript, Bootstrap y HighCharts.

Summary

This Final Degree Project involves the design and implementation of a web application able to analyze Free Software projects placed in the collaborative development platform GitHub. The aim is to create an application which analyzes the repositories of the projects contained in this platform and to be capable of displaying metrics and indicators that detail the progress of each project through an intuitive interface and answering logical questions in the development of any software.

Despite the fact that free software is growing every day and it is more used by individuals and companies that bet on this philosophy and want to save costs, the analysis of this type of software is almost unknown. Therefore to create a tool capable of providing statistics and acquiring the necessary knowledge to know how to interpret knowledge is of major importance.

We understand that knowing the developer community behind each application is vital when assessing the future of this software and its life cycle, consequently, in this project we have focused mainly on making metrics and indicators to analyze the behavior of these contributors throughout the whole development of software. In this way we can find important information such as the current status of the project, in what way it is helping your community or how many members is it formed by.

The application we have created in this Final Degree Project uses repositories which are hosted in GitHub, one of the most commonly used platforms by programmers to store the changes made during the development of software and share them with third-party developers in a very easy way. This platform offers the possibility for anyone to access these repositories and

assess its information.

For the elaboration of our project we have extracted the information through the CVSAnalY tool, and stored it in the most popular Free Software database server, MySQL. By using the database language, SQL, a data mining phase has been carried out in order to show the results of the statistics obtained through a web site.

To process all the data and apply the logic of the application we have taken advantage of the framework Django and Python, the main programming language of this project. Finally, in the website creation we have used many technologies that facilitate the construction of dynamic and intuitive web applications, such as HTML, CSS, AJAX, JavaScript, and Highcharts Bootstrap.

Índice general

Índice de figuras

Capítulo 1

Introducción

En este primer capítulo haremos una pequeña introducción al proyecto. Contaremos qué nos ha motivado a hacerlo y el marco histórico en el que se ha realizado, centrándonos en su utilidad en la vida real. Además explicaremos como se ha estructurado la memoria para que el lector pueda seguirla de una manera más clara.

1.1. Contexto

Cuando nos enfrentamos a un proyecto de ámbito privado, siempre hay una o varias personas que lo dirigen, se encargan de su seguimiento y de que todo vaya en el sentido que se pretende. Las personas que conforman el grupo de trabajo de dicho proyecto suelen conocerse, normalmente trabajan para la misma empresa, se reúnen y discuten entre ellas cada fase.

Estos proyectos suelen tener fecha de entrega y un equipo entero de personas disponible durante varias horas al día y durante varios días o meses para su cumplimiento. Cada persona tiene su rol definido y sus horas de trabajo establecidas para sacar el proyecto adelante. Además cuando dicho proyecto sale a la luz, hay un soporte físico y humano que lo sostenta, encargando de la fase de producción, las posibles actualizaciones y que analiza el trabajo elaborado en la etapa de desarrollo para tener un *feedback* en proyectos futuros.

La mejor técnica para evaluar un proyecto es la reflexión colectiva de las personas que han formado parte de él y el debate riguroso en grupo. Se fijan aspectos que determinan el cumplimiento de los objetivos iniciales y la calidad del producto final. Para esta evaluación se han ido recogiendo datos o indicadores a lo largo de todo el proceso que miden, por ejemplo, la eficacia o eficiencia del proyecto.

Pero, ¿qué pasa con los proyectos de Sofware Libre? Durante la carrera, hemos contado con profesores defensores del Software Libre, aquel que puede ser copiado, modificado, estudiado y distribuido por cualquier persona. Amparada por las licencias que la protegen de su privatización, como la GNU (General Public License), esta filosofía se encuentra en un momento de auge. En los últimos años, cada vez son más los desarrolladores que distribuyen sus programas, y el código que las forman, de manera libre y abierta. ¿Qué ventajas tiene respecto al software propietario?

- Al proporcionar acceso al código cualquiera puede modificarlo y adaptarlo a sus necesidades.
- Ciclo de vida actualizado: este tipo de aplicaciones se mantienen gracias a las comunidades de usuarios en todo el mundo que aportan mejoras, corrigen los posibles errores o *bugs* que puedan aparecer y publican nuevas versiones de este.
- Esta comunidad de usuarios también realiza traducciones sobre las aplicaciones: cualquier persona capacitada puede traducir y adaptar el software a cualquier lengua.
- A diferencia del software propietario, las licencias de software libre se pueden instalar en todas las máquinas que se quiera, las veces que se necesite.
- Independencia del proveedor: no está supeditado a las condiciones de mercado del proveedor. Si este desapareciera, cualquier desarrollador puede continuar dando soporte a la aplicación.
- Mayor seguridad y privacidad de los datos: al tener acceso al código, cualquiera puede ver como se tratan y almacenan los datos. Esto permite que tanto hackers como empresas de seguridad auditen el software disminuyendo el riesgo de puertas traseras en las que se pueda introducir código malicioso como Troyanos o captador de teclas.

- Y, por supuesto, el bajo coste de las aplicaciones, que en su mayoría es gratuito.

A pesar de todas estas ventajas, hoy en día el software libre no es cien por cien conocido, muchas personas lo confunden con el software gratuito. Se han realizado numerosas estadísticas respecto a este tema y la tendencia cada año es que cada vez más personas estén a favor de él. Aun así todavía hay mucha gente que no confía plenamente en sus resultados.

Según el estudio "Valoración del software libre en la sociedad 2014" en el que se realizan encuestas sobre el uso, conocimiento, confianza, preferencias y cambio del Software Libre en 16 países de habla hispana, solo un 42 % de las personas encuestadas fueron capaces de reconocer la respuesta correcta cuando se les preguntaba sobre qué es el software libre, frente a un 34,8 % del 2013 en la misma encuesta. Como se puede ver, el 35 % de los encuestados conocen la definición de Software Libre pero piensan que es gratuito.

Figura 1.1: ¿Qué es el Software Libre? Valoración del software libre en la sociedad 2014

Cuando se trata de confianza, a pesar de que mucha gente no sepa exactamente su definición, las estadísticas aumentan a favor del Software Libre habiendo un 80 % de encuestados a favor de esta filosofía, frente a un 20 % en contra. La mayoría de personas que lo han usado confían plenamente en él, argumentado por el número de actualizaciones que tiene y la comunidad de desarrolladores que hay detrás de él.

Figura 1.2: ¿El Software Libre te da confianza? Valoración del software libre en la sociedad 2014

Frente a la pregunta de si usas Software Libre, la gran mayoría afirman que sí, solo un 30 % dice no usarlo. Esta respuesta no sería del todo válida ya que no cuenta a las personas que lo usan sin saberlo, por ejemplo, cualquier usuario de Moxila Firefox que no sepa qué es un navegador de Software Libre.

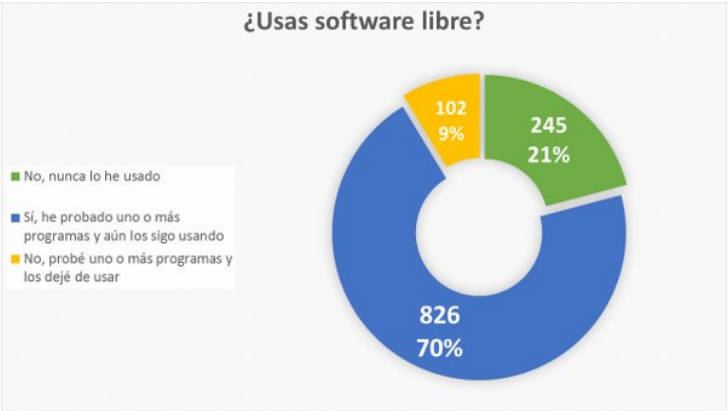

Figura 1.3: ¿Usas Software Libre? Valoración del software libre en la sociedad 2014

Si atendemos a nivel mundial, según las estadísticas realizadas por RedHat, empresa creadora de algunas distribuciones Linux como Red Hat enterprise Linux o Fedora, el Software Libre es ampliamente usado, siendo los países más desarrollados aquellos que más oportunidades le dan.

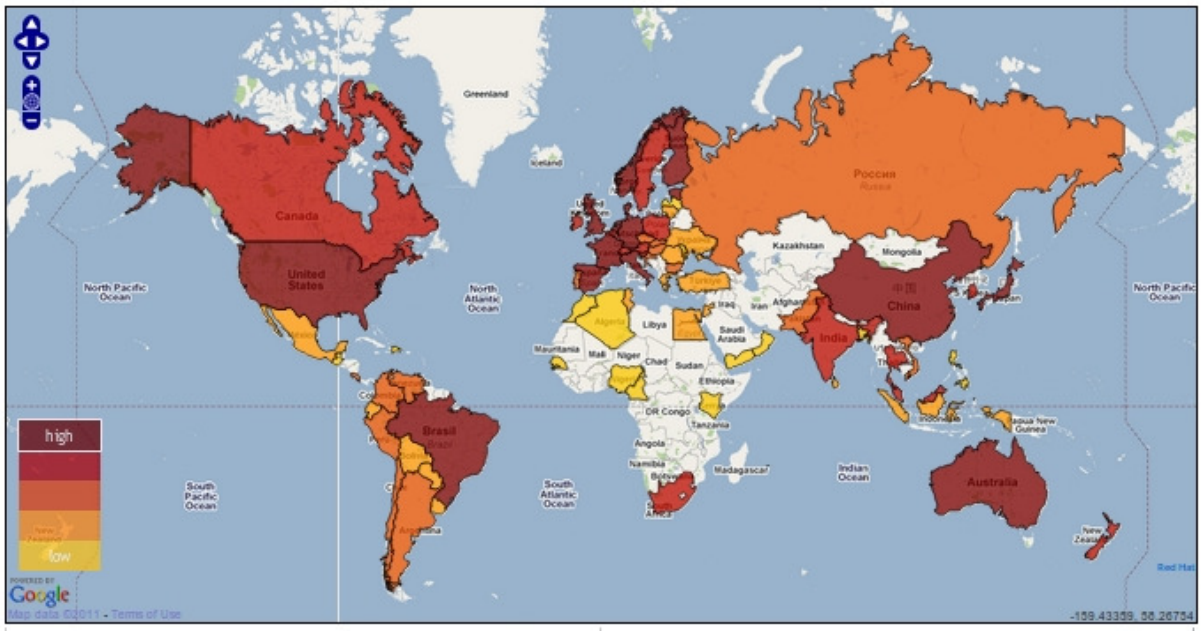

Figura 1.4: Mapa mundial sobre el uso de Software Libre

Con todo esto, y con las estadísticas en la mano, podemos llegar a la conclusión de que el Software Libre está en pleno crecimiento. Cada vez son más los usuarios y empresas que lo utilizan a diario y confían plenamente en él.

1.2. Motivación

A lo largo de mi corta carrera profesional me he dado cuenta de la importancia de analizar cualquier proyecto realizado en una empresa. Por mi experiencia he podido observar la cantidad de recursos, tanto de hardware y software como de personas, dedicados a ello. Y no solo en los proyectos, cientos y miles de KPIs e indicadores se miden diariamente en todas las áreas de una empresa con el fin de medir y mejorar cualquier parte relacionada. El elaboramiento de Dashboard y reportes y su posterior estudio es parte del día a día de muchas personas, entre ellas de mi.

Y me pregunto, ¿quién se encarga de realizar dicho análisis en proyectos de Software Libre?

Cuando se trata de proyectos pequeños es fácil, pero, ¿los proyectos que involucran a cientos o miles de personas, como Linux, Git o JQuery? Es cierto que en estos proyectos siempre hay un grupo de personas más involucradas y que llevan el peso de la mayor parte de su desarrollo, pero ¿como medir la aportación del resto de sus desarrolladores?. Normalmente son personas completamente desconocidas entre ellas y dispersas geográficamente que dedican parte de su tiempo libre en aportar *su granito de arena* para crear actualizaciones o nuevas funcionalidades, en corregir posibles *bugs*, o mejorar el código ya escrito. ¿Quién mide el tiempo y la aportación de dichas personas?, ¿cuanto valor han aportado al proyecto?, ¿es factible contar con personas que solo dedican parte de su tiempo libre y no están "obligados por contrato" a continuar con el trabajo empezado?

Estas son algunas de las preguntas que me han llevado a realizar mi proyecto: una APLICACIÓN WEB PARA EL ANÁLISIS DE PROYECTOS DE SOFTWARE LIBRE.

1.3. Estructura

En este apartado vamos a explicar la estructura de esta memoria para facilitar su comprensión al lector.

- El primer capítulo, en el cual nos encontramos ahora, consta de 3 partes en las que se ayuda al lector a ponerse en situación. Se narra el contexto en el que está desarrollado el proyecto así como los intereses que nos ha llevado a hacerlo. Por último se hace un pequeño resumen de la estructura de la memoria.

- El segundo capítulo, OBJETIVOS detalla las metas que se quieren lograr con este proyecto y que fueron marcados al comienzo del desarrollo de la aplicación y durante su producción.

- El capítulo ESTADO DEL ARTE, describe cada una de las tecnologías que toman parte en el proyecto. Se pretende dar una visión general de las funcionalidades de cada una de ellas para posteriormente explicar como contribuyen en este proyecto.

- Posteriormente, en el capítulo de DISEÑO E IMPLEMENTACION, detallaremos como se ha desarrollado la aplicación desde un principio. Consta de cuatro apartados:

  - Arquitectura general: explicaremos cual es el diseño que sigue la aplicación. Como podremos ver después, se trata de una aplicación cliente-servidor; describiremos su arquitectura y las partes que la forman.
  - Fases del proyecto: aquí expondremos las etapas que hemos seguido para la construcción del software, desde el momento en que decidimos que aplicación queríamos desarrollar.
  - Diseño e implementación del servidor: es uno de los puntos más importantes. Describe toda la lógica de la aplicación realizada en la parte del servidor, así como la estructura de la base de datos utilizada.
  - Diseño e implementación del cliente: En este punto hablaremos de la interfaz que se muestra al usuario, la parte de presentación de la aplicación, y como está implementada.

- En el apartado de RESULTADOS haremos dos casos de estudio con diferentes proyectos y analizaremos los resultados obtenidos en ellos.

- CONCLUSIONES: Echaremos la vista hacia atrás, valorando si hemos logrado los objetivos marcados al principio.

Capítulo 2

Objetivos

2.1. Objetivo general

El objetivo, por tanto, es diseñar e implementar una aplicación web que permita analizar cualquier proyecto de software libre alojado en la plataforma de desarrollo colaborativo GitHub y que responda a ciertas preguntas que se pueden formular al finalizar un proyecto de estas características, tales como:

- ¿Cuales son las perspectivas de futuro del proyecto? ¿Nos garantiza un ciclo de vida actualizado?
- ¿Tiene una comunidad de desarrolladores férrea y estable?
- ¿Cuántas personas han formado parte del desarrollo del proyecto? ¿Cuál es su contribución?
- ¿Quienes son las personas que más han aportado al proyecto? ¿Cuál es su peso? ¿Continúan aportando mejoras y nuevas versiones?
- Si estas personas dejaran de colaborar, ¿ofrece el proyecto confianza de continuidad?
- ¿Cuales son los intervalos de tiempo más productivos?

- ¿Hay dispersión geográfica entre los desarrolladores?
- etc.

2.2. Objetivos especificos

Para conseguir dichos objetivos, nos hemos planteado otros más específicos. Pequeños hitos de trabajo para construir paso a paso nuestra aplicación.

- Crear una base de datos y almacenar los repositorios de los proyectos a analizar.
- Diseñar e implementar un servidor capaz de elaborar la lógica de la aplicación.
- Diseñar e implementar una interfaz clara y sencilla por la cuál mostrar a los usuarios los datos obtenidos.
- Implementar métricas y gráficos dinámicos que muestren las estadísticas de evolución del proyecto.

Para conseguir estos objetivos hemos tenido que documentarnos y aprender el lenguaje de base de datos así como el funcionamiento de cada una de las tecnologías usadas para tales fines.

Capítulo 3

Estado del arte

A continuación vamos a detallar cada una de las tecnologías que se han usado en este proyecto.

3.1. Django

> "Django es un entorno de desarrollo web escrito en Python que fomenta el desarrollo rápido y el diseño limpio y pragmático."[1]

En otras palabras, es un framework de alto nivel, una plataforma de desarrollo que facilita la creación de páginas web.

Se basa en un patrón de diseño MCV "Modelo-Vista-Controlador". Este patrón, además de separar claramente los datos de la lógica en una aplicación, está pensado para reutilizar código y facilitar las tareas de desarrollo. Esta estructura también facilita el poder hacer cambios o el mantenimiento de la aplicación de una forma simple sin afectar a las demás partes. La arquitectura de este patrón es simple:

[1] `http://django.es/`

- Modelo: representa la información del sistema, los datos que queremos mostrar y es el encargado de gestionar los accesos y modificaciones a dichos datos. Es el acceso a la base de datos. Trabaja con la Vista facilitando los datos a mostrar y con el Controlador para realizar las modificaciones, añadir, eliminar o consultar los datos pertinentes según los privilegios que se hayan otorgado en cada momento.

- Controlador: Es la lógica de negocio. Contiene los métodos necesarios para acceder al Modelo y pasarle los datos a la Vista según la entrada que reciba. En Django el controlador se llama *View*.

- Vista: Suele ser la interfaz de usuario. Presenta los datos obtenidos por el Modelo en el formato definido. Django utiliza un sistema de plantillas (Templates) para mostrar dicha información.

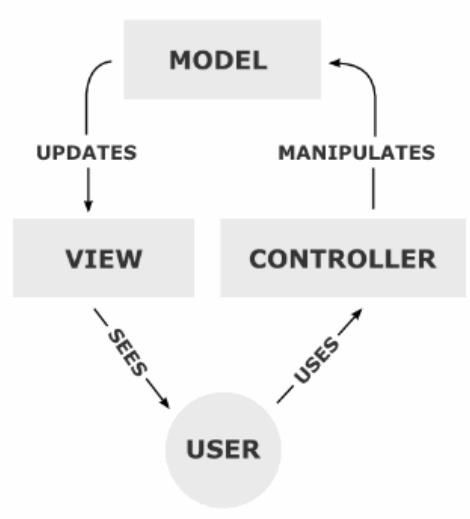

Figura 3.1: Diagrama MVC

A pesar de seguir muy fielmente este patrón, los creadores de Django no se quieren ligar a ningún modelo en particular; por ello en lo que en un patrón MVC sería el Controlador, en Django son las vistas (View), y lo que serían las Vistas en un MVC en Django son plantillas (Templates). Esto hace que sea también conocido como un framework MTV, modelo-template-vista.

Esta estructura, además de simplificar y organizar el desarrollo web, sigue el principio DRY (Don't repeat yourself) que pone énfasis en el re-uso del código evitando la duplicidad, lo que aporta claridad y limpieza a la hora de programar una aplicación.

Centrándonos un poco más en Django nos encontramos con las siguientes características, algunas de ellas solo existentes en este framework, lo que lo hace una de las mejores opciones a la hora de desarrollar páginas web:

- Python: Es el corazón de Django, y lo que realmente le hace potente. Al utilizar este lenguaje de alto nivel, Django hereda todas sus características y permite desarrollar aplicaciones muy rápidas y potentes.
- Módulo admin: Django es el único framework con un módulo de Administrador preinstalado por defecto, permitiendo gestionar nuestra aplicación sin necesidad de escribir más código. Este módulo permite administrar los usuarios, contraseñas e incluso sincronizar las bases de datos para manejarlas desde él mediante una interfaz sencilla.
- Contiene un mapeador de URLs basado en expresiones regulares. Es un método que hace corresponder peticiones URLs con el código que manejan dichas peticiones.
- Sistema de plantillas con heredabilidad que separa la presentación de los datos. Normalmente se usan para generar HTML aunque se puede usar cualquier lenguaje de texto.
- Api robusta de base de datos.

Con todo esto podemos llegar a la conclusión de que Django es un framework de alto nivel, que promueve un diseño simple y limpio, legible y transparente facilitando el desarrollo de aplicaciones web rápidas y complejas.

3.2. Python

Es el lenguaje usado por Django y el usado en este proyecto. Es un lenguaje interpretado de alto nivel, multiparadigma y de código abierto. Veamos que significa esto:

- Al ser un lenguaje interpretado no necesita compilación, las instrucciones se ejecutan directamente. Al ejecutar el código fuente Python se encarga de convertir este código en bytecodes que después traduce en un lenguaje entendible por la máquina. Las grandes ventajas de esto es que lo hace portable con independencia de la plataforma, facilita la depuración del código y el tipado es dinámico, las variables pueden tomar valores de distinto tipo.

- Es un lenguaje con una sintaxis muy simple y fácil de manejar, lo que favorece su aprendizaje. Esto se debe a que el pseudo-código de Python está escrito en un nivel de abstracción más alto que el de la máquina, sin necesidad de ir a un bajo nivel, de manejar registros, ni direcciones de memoria o subrutinas; es como si estuvieras escribiendo en inglés.

- Python es multiparadigma, ya que soporta varios estilos de programación: la programación orientada a objetos, la imperativa y la funcional.

- Posee una licencia de código abierto, *"Python Software Foundation License"*. Es de libre distribución y se puede leer su código fuente, incluso hacerle cambios o actualizaciones. Esta idea de compartir el código hace que los programas sean de mayor calidad que los de software propietario.

3.3. Git y GitHub

Git es un sistema de control de versiones distribuido que permite llevar un seguimiento de los cambios que se han producido en un programa desde el comienzo de su desarrollo. Fue diseñado por Linus Torvalds, creador del famoso sistema operativo Linux, para tener un control de versiones sobre este. Es Software Libre, distribuido bajo los términos de *Lincencia Pública General GNU v2*. Pero veamos la importancia de Git y porqué se ha hecho tan popular.

Cuando estamos creando un programa siempre es bueno realizar copias de seguridad sobre las distintas versiones que vamos realizando para no perder los datos o para no perder el estado anterior. Al final acabamos con muchos archivos renombrados de diferente manera, lo cual, ni es práctico, ni eficiente. A esto le tenemos que sumar que cabe la posibilidad de que

no seamos el único programador de un proyecto; normalmente solemos colaborar con varios y trabajar en equipo cuando desarrollamos un software. Esto puede dar problemas en los flujos de trabajo: machacar el trabajo de otro, no tener constancia de cambios producidos, etc. Necesitamos una herramienta para administrar el desarrollo de un software. Para solucionar todos estas limitaciones surge Git.

Usar un sistema de control de versiones como Git nos permite conservar y restaurar versiones antiguas de nuestros ficheros, mezclar cambios de diferentes versiones y comparar las modificaciones realizadas entre dos versiones diferentes. Además permite llevar un seguimiento detallado sobre cualquier cambio, las personas que participaron, fechas de actualización del software y otros datos muy valiosos que se pueden aportar al proyecto. Se trata de un sistema multiplataforma, capaz de operar en los sistemas operativos más usados: Linux, Windows, Mac, incluso Solaris. Además Git es un sistema distribuido; en este tipo de sistemas cada colaborador trabaja con una copia en local de todo el repositorio y puede realizar commits (enviar cambios al sistema de control) desde cualquier sitio sin necesidad de estar conectado a la red, lo que te da autonomía en el trabajo. Esta es la gran ventaja de Git, permite trabajar en cualquier momento y lugar manteniendo actualizado el sistema de control de versiones y sin la necesidad de estar conectado al servidor central.

Otras características importantes de Git son:

- A diferencia de otros sistemas de control de versiones, en los que se almacenan únicamente las diferencias respecto al fichero original, Git guarda un estado actual del archivo con todos sus cambios en caso de que los haya, y en caso de no haberlos, guarda una referencia al archivo original, optimizando los recursos.
- Todos los archivos se guardan por una suma de comprobación o checksum, SHA-1, lo que hace casi imposible que un fichero se corrompa o se pierda.
- Se pueden publicar todas las modificaciones mediante HTTP, FTP, rsync o SSH.

Todo ello hace el trabajo de los desarrolladores mucho más fácil y organizado, lo que ha hecho que Git se convierta en el sistema de control de versiones preferido por la mayoría de

programadores.

GitHub es un servidor para alojar los repositorios a través de Git y, en parte, ha sido culpable de su éxito. Es un sitio web donde compartir el código de una aplicación. A pesar de que Git es un sistema distribuido y no necesita de servidores ya que podemos guardar los cambios en local, si estamos desarrollando una aplicación en equipo, GitHub nos facilita mucho el trabajo a la hora de compartir todos los avances. Esto ha hecho que muchos programas de Código Abierto usen Github como repositorio y gracias a ello tengan mayores contribuciones. Otra de sus grandes ventajas es que podemos visualizar el código desde su navegador de una manera clara y con la sintaxis resaltada correspondiente del lenguaje que estemos usando, de manera que podemos consultar cualquier archivo sin necesidad de descargarlo, incluso podemos visualizar versiones anteriores de él.

Además GitHub ofrece una serie de características que van más allá de compartir el código, algunas de las cuales son: una wiki y una página web para cada proyecto, un visor de ramas para poder observar todo el progreso, una herramienta para escribir anotaciones sobre el código. Esto le hace ser una excelente herramienta para el trabajo de un desarrollador.

Otro servicio que ofrece GitHub es el poder colaborar con el software de terceras personas. Para ello puedes clonar un repositorio ajeno a tu cuenta mediante un Fork, realizar las modificaciones o aportaciones que consideres oportunas y finalmente enviar una solicitud de pull a la persona dueña de dicho repositorio para que valore tu contribución y añadirla al código original sin que se pierdan los commits realizados con tu usuario.

No obstante, aunque está pensada para compartir conocimiento y favorecer el Código Abierto, GitHub es un proyecto comercial, a diferencia de Git. Esta aplicación, que es gratuita para proyectos públicos, ofrece la opción de alojar hasta 5 proyectos de manera privada mediante un pago mensual de 7 dólares, lo cual resulta muy barato.

3.4. CVSAnalY

CVSAnalY es una herramienta que permite extraer la información de un repositorio, la procesa y la almacena en una base de datos. En dicha información se encuentra los autores/commiters que han participado en un proyecto, sus emails, el número de commits y de ficheros que han realizado, fechas, comentarios realizados en cada uno de ellos, etc. Además, al estar almacenado en una base de datos se puede extraer fácilmente las relaciones entre todos estos datos.

Para el funcionamiento de CVSAnalY necesitamos las siguientes dependencias, algunas de las cuales explicaremos en esta memoria:

- RepositoryHandler
- Git
- Python MysqlDB
- cvs

Debemos hacer una mención especial a sus orígenes. CVSAnalY es un proyecto de software libre creado en 2006 por Libresoft[2]. Libresoft es un grupo de investigación creado en la Universidad Rey Juan Carlos cuyo propósito es la investigación de software libre. Las personas que han contribuido en el desarrollo de CVSAnalY son miembros de esta universidad:

- Gregorio Robles. Tutor de este proyecto.
- Alvaro Navarro.
- Jesus M. Gonzalez Barahona.
- Israel Herraiz
- Juan Jose Amor

[2] `http://www.libresoft.es/`

- Martin Michimayr
- Alvaro del Castillo
- Santiago Dueñas

3.5. MySQL

Es un servidor de bases de datos SQL (Structured Query Language), un sistema de administración de base de datos relacional, multihilo y multiservidor. Se puede distribuir en dos versiones:

- Bajo la GNU GPL (General Public License), de software libre.
- MySQL AB, para uso privado. Requiere la compra de una licencia y suele usarse en empresas.

Al ser un sistema relacional guarda los datos en diferentes tablas en vez de usar una sola. Es el modelo más usado para administrar datos dinámicamente. Se basa en establecer relaciones que pueden considerarse como tuplas (cada fila sería un registro o tupla y las columnas los campos). Y al tratarse de un sistema multiusuario y multihilo puede ser utilizado por varias personas simultáneamente y procesar las peticiones también de forma simultánea.

SQL es un lenguaje para interactuar con base de datos que nos permite agregar o eliminar campos, realizar consultas o modificar tablas o registros. Con MySQL podremos realizar todas estas acciones desde el terminal de MySQL o desde el propio Python. Administrar el servidor de MySQL es muy fácil. Las instrucciones más comunes para manejarlo desde la shell (y las usadas en este proyecto) son muy intuitivas:

- show databases; Muestra todas las bases de datos que han sido creadas previamente y están almacenadas en el servidor.

- use *n*ombre; Indica a MySQL que vamos a usar esa base de datos.
- create *n*ombre; Crea una base de datos con el *nombre* indicado.

3.6. HTML

HTML (HiperText Markup Language) es el lenguaje de marcado para escribir páginas web creado por Tim Berners-Lee en 1980. Es un estándar de la W3C (World Wide Web Consortium), y como tal, está pensado para que todos los navegadores interpreten un mismo fichero de la misma forma.

Es un lenguaje estructurado, está definido por una estructura simple y un código, llamado código html, formado por etiquetas o marcas que determinan la forma en la que debe aparecer el texto en la web. Esto hace que sea un lenguaje muy fácil de leer y entender pero puede aumentar mucho el código de la página web. Se basa en la referenciación para cualquier elemento que no sea texto: imágenes, vídeos, etc. Para añadir cualquiera de dichos elementos se hace una referencia a la ubicación de estos en vez de copiarlos en el código, y es el navegador el encargado de interpretarlo para visualizarlo en la página web. También se puede incluir un tipo de código o programa llamado script que aportan diferentes maneras de comportamiento en las web.

3.7. CSS

CSS u hojas de estilo en cascada, es un lenguaje usado para definir el estilo de una página HTML. Las hojas de estilo se crearon para compensar la carencia de HTML respecto a la presentación, en el cual cada elemento tiene su propio estilo sin tener armonía con el resto.

La idea es separar el contenido de una página de su presentación. Con CSS podemos tener un sitio web con varias páginas y una sola hoja de estilo a la que se referencien que defina el estilo de todas ellas, de esta manera podremos cambiar cualquier aspecto: color de fondo, tipo de letra, estilo de imagenes o tablas... y modificaremos todas las páginas web por igual. No

obstante, el estilo de una página también se puede definir dentro del propio HTML, bien en la cabecera del documento o en cada elemento particular con la etiqueta *style*.

3.8. Bootstrap y Bootsnipp

Bootstrap es el framework de Twitter, de software libre, para el diseño de aplicaciones web. Es una herramienta en auge muy útil para los desarrolladores a la hora de diseñar aplicaciones web de una forma elegante. Cada vez son más las páginas que incorporan esta tecnología, ejemplo de ello es, por supuesto, Twitter.

Una de sus características principales es que permite crear interfaces simples e intuitivas a través de CSS y HTML con un diseño adaptativo o Responsive Design: la página web se adapta al tamaño del dispositivo por el que se está mostrando: ordenador, tablet, móvil... Otra de sus grandes ventajas es que incorpora extensiones de JavaScript opcionales para animar el sitio web. Pero sobre todo, y lo que he ha hecho tan popular, es el ahorro de tiempo que supone no partir de cero en el diseño de una aplicación y así poder dedicar más tiempo al código.

Bootsnipp es un repositorio de elementos o snippets para complementar el diseño de una aplicación web realizada con Bootstrap a través de HTML, CSS y JavaScript.

3.9. JavaScript

Desarrollado por Netscape Communication, JavaScript, también conocido como JS, es el lenguaje de scripting más conocido y usado de todos. Es un lenguaje interpretado, de tipado dinámico, orientado a objetos y multiplataforma. Como todo lenguaje de scripting se caracteriza por ser limitado y seguro, no se puede escribir aplicaciones enteras solo con él, sirve para pequeños programas o scripts que realicen instrucciones muy concretas.

Aunque puede ser usado en la parte del servidor de una aplicación, normalmente se utiliza en el lado del cliente mejorando la interfaz con el usuario haciendo que las aplicaciones respondan

a eventos, como clicks de ratón, formularios, etc. y construir así páginas web dinámicas. En la parte del cliente el encargado de interpretar código JS integrado en páginas web son los navegadores, lo que alivia el trabajo del servidor. Esto hace que el código sea visible y pueda ser leído por cualquiera. Todos los navegadores modernos son capaces de interpretarlo.

3.10. HighCharts

Es una librería de JavaScript que permite la creación de gráficos interactivos. Una de sus grandes ventajas es que es compatible con la mayoría de navegadores, incluso iPad y iPhone.

Esta herramienta ofrece gráficos de todas las formas posibles: lineas, columnas, área, circular, en 3D... o combinaciones de estos. Genera gráficos interactivos en los que aparece el valor de cada punto solo con poner el ratón en él. Además te permite descargarte dichos gráficos directamente desde la aplicación web en formato PNG, PDF, JPG o SGV en función de tus necesidades, lo que le hace una herramienta muy útil en usos comerciales; tanto es así que es usada por 61 de las 100 mayores empresas del mundo como son Facebook, Visa, Nokia, Ericsson, etc..

3.11. Ajax

Acrónimo de Asynchronous JavaScript and XML, AJAX es una de las técnica de desarrollo web más populares que existen para crear aplicaciones cliente/servidor dinámicas e interactivas. Al igual que JavaScript, AJAX es interpretado en el lado del cliente por el navegador, manteniendo una comunicación asíncrona con el servidor en segundo plano, lo que alivia el trabajo de este.

Esta técnica nos permite crear aplicaciones web dinámicas y muy rápidas ya que es capaz de realizar cambios en la página web sin necesidad de recargarlas ni realizar peticiones al servidor, ya que los datos han sido adquiridos previamente y cargados en segundo plano para ser

visualizados como respuesta a un evento por parte del usuario.

AJAX en realidad no es una tecnología para el desarrollo web, sino una expresión para englobar el uso de ciertas tecnologías trabajando juntas. Estas tecnologías son:

- HTML y CSS (explicado previamente)
- DOM (Document Object Model): es una API para representar documentos HTML y XML y manipularlos de forma rápida y eficiente.
- XML (eXtensive Markup Language): es un lenguaje de marcas usado para transferir datos con el servidor y almacenarlos de forma legible.
- XMLHttpRequest: es una interfaz creada para intercambiar datos con el servidor web mediante peticiones HTTP y HTTPS.

Capítulo 4

Diseño e implementación

En este capítulo explicaremos como se ha desarrollado el proyecto y las fases por las que ha pasado. Diferenciaremos claramente las partes de las que consta y haremos un análisis más detallado de cada una de ellas. A su vez, detallaremos su funcionamiento basándonos en diagramas que faciliten el entendimiento de su estructura y las tecnologías usadas en cada fase.

4.1. Arquitectura general

La aplicación que hemos creado se basa en un modelo cliente-servidor. En este tipo de arquitectura las tareas se reparten entre dos partes claramente diferenciadas: el cliente, encargado de realizar las peticiones al programa; y el servidor, el cual es el responsable de atender dichas peticiones y devolver la respuesta oportuna al cliente. Podemos diferenciar otra tercera parte no menos importante, el servidor de datos o base de datos, que almacena y proporciona los datos que el servidor tiene que procesar en cada petición.

En la interacción con la base de datos se puede observar otra vez la estructura cliente-servidor: el servidor previamente mencionado actúa como cliente de la base de datos, siendo esta en este caso el servidor. En la siguiente figura se muestra un esquema de la interacción de cada componente de una manera clara.

Ante todo se busca la separación lógica de cada una de las partes. No obstante, aunque existe esta separación, no es necesaria la separación física, tanto cliente como servidor pueden estar alojados en la misma máquina.

Figura 4.1: Arquitectura cliente-servidor

4.1.1. Servidor

El servidor es donde reside toda la lógica de la aplicación y el programa en sí. En esta fase del proyecto hemos usado el servidor de desarrollo que viene por defecto en Django. Es un servidor ligero, muy cómodo en la fase de desarrollo de una aplicación. Su gran desventaja es que solo pueden manejar una petición fiable a la vez y no ha pasado por auditorias de seguridad de ningún tipo.

Por defecto al lanzar este servidor escucha solo conexiones locales, pero se puede configurar para que escuche en otro puerto o en una dirección IP especifica (útil si quieres que otro desarrollador vea tu aplicación) con el siguiente comando:

python manage.py runserver 8080

En este servidor se aloja el fichero views.py, que es el que contiene toda la lógica de la aplicación. Básicamente toma una petición vía web, la procesa y devuelve la respuesta en código HTML. Para saber que información es la que se está pidiendo realiza un mapeo del patrón de la url, basado en expresiones regulares, y la función que debe ser llamada por este. Esto

viene definido en el archivo urls.py que se crea en Django automáticamente al comenzar cada proyecto.

4.1.2. Cliente

El cliente es donde se encuentra el usuario final que accede a la aplicación a través de un navegador o cliente web (Chrome, Firefox..). En nuestra aplicación el cliente se encarga de procesar las peticiones del usuario y mandarlas al servidor. Posteriormente recibe la respuesta en forma de página html, la procesa y la visualiza por el navegador.

En la mayoría de vistas de nuestra aplicación van incluidos códigos JavaScript y Ajax capaces de proporcionar páginas web dinámicas y en algunas ocasiones para realizar cambios en la aplicación sin necesidad de recargar de nuevo la página, sin realizar más peticiones al servidor, y proporcionar así mayor velocidad. Es función del cliente el interpretar estos códigos, lo que alivia el trabajo al servidor.

4.1.3. Servidor de datos

El tercer módulo a tener en cuenta es la base de datos. Está alojada en la parte del servidor, el cual actúa como cliente de esta, y se encarga de almacenar los datos de la aplicación, de proporcionarselos al servidor web cuando sea necesario y de asegurar la integridad de estos. Para este proyecto usamos la base de datos MySQL. Es una base de datos relacional, las tablas se relacionan entre ellas a través de los distintos campos.

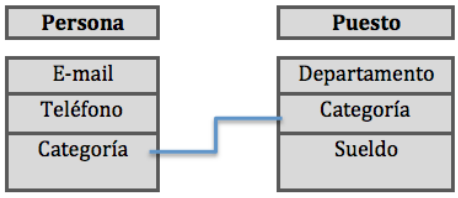

Figura 4.2: Base de datos relacional

Lo primero de todo después de instalar MySQL y haber iniciado el servidor es acceder a él para administrarlo. Para conectarte basta con el comando *mysql -u user -p* e introducir posteriormente la contraseña que se seleccionó en la fase de instalación para el usuario administrador *root*. Una vez dentro existen diferentes comandos en función de tus necesidades, como crear la base (o bases) de datos que se requieran.

Una vez creada la base de datos, y aunque Django proporciona una API para interactuar con ellas, al haber elegido MySQL como servidor debemos instalar el conector MySQLdb. Finalmente, podremos acceder a la base de datos a través de Python mediante consultas SQL.

4.2. Fases del proyecto

Como todo proyecto, hemos pasado por diferentes fases de desarrollo:

- Documentación: Fue la primera de las fases. Cuando ya tenía claro el tipo de proyecto que iba a hacer y qué tecnologías iba a tomar como punto de partida me documente sobre ello. Algunas ya las conocía por las diferentes asignaturas de la carrera: Django, Python, HTML... Pero con otras no había trabajado nunca. Busqué documentación sobre el funcionamiento de cada una de ellas y las funcionalidades que me aportaban. Uno de los trabajos que más me costó fue la base de datos, ya que es un tema que no se ve mucho en la carrera y nunca había trabajado con MySQL, por lo que casi tuve que empezar de cero con esta tecnología.

- Instalación: Fue la etapa en la que tuve más problemas. En un principio quería trabajar sobre Macintosh, ya que es el sistema operativo que uso a diario y en el cual me siento más cómoda. Pero a medida que instalaba cada programa empecé a tener problemas con las versiones y no terminaba de cuajar el trabajo realizado. Para solucionarlo obtuve un portátil del departamento de Gsyc de la URJC con el sistema operativo Linux y comencé a instalarme todo de nuevo.

- Implementación: Cuando ya tuve todo instalado empecé por crear la aplicación en Djan-

go. Posteriormente busqué un proyecto de software libre en GitHub sobre el cuál comenzar a realizar mi aplicación. Tuve muchas opciones a elegir pero, finalmente, me decanté por Gedit, que es el editor de texto, de software libre, sobre el cuál he programado esta aplicación, además al ser un proyecto grande iba a obtener mucha información de él. Después cloné el repositorio de Gedit a un directorio local, creé una base de datos vacía en MySQL y con CVSAnalY traspasé el proyecto clonado a mi base de datos. El resto fue empezar a programar la aplicación.

A continuación, para entender un poco más la implementación y diseño de la aplicación se muestran dos diagramas de flujo.

El primero es el flujo de proceso establecido en este Proyecto Fin de Carrera desde que queremos analizar un software hasta que se muestran las estadísticas por el cliente web.

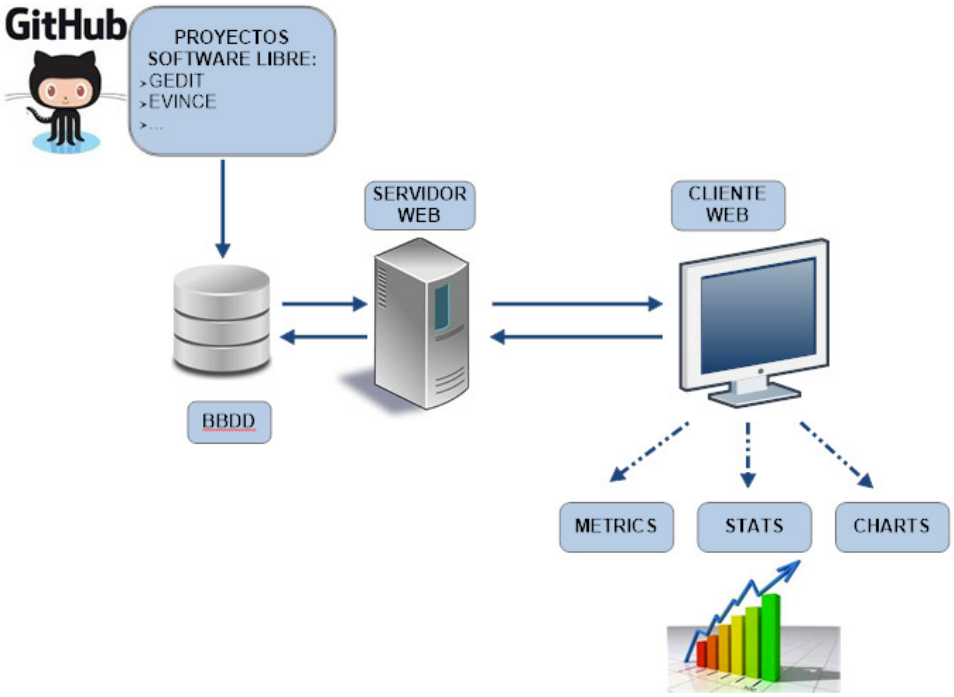

Figura 4.3: Diagrama de flujo del proceso de análisis de un proyecto de software libre

El siguiente diagrama muestra las tecnologías usadas en cada fase.

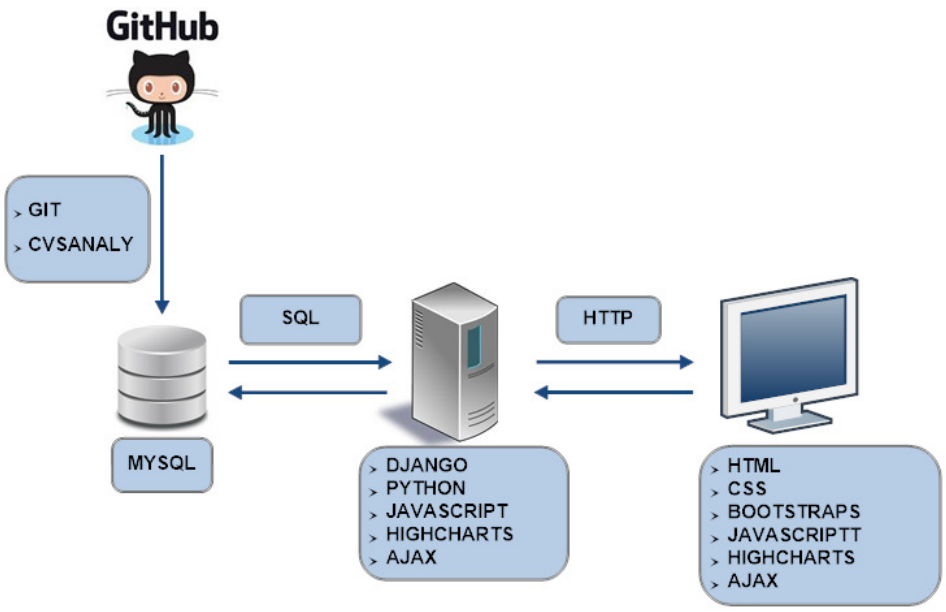

Figura 4.4: Diagrama de flujo de las tecnologías usadas en cada componente

4.3. Diseño e implementación del servidor

Como hemos mencionado anteriormente, la implementación del servidor la hemos desarrollado en Python mediante el framework Django y usando como servidor de datos MySQL. Empezaremos explicando algunas funcionalidades del servidor, las cuales están programadas en el archivo *views.py*, que contiene toda la lógica de negocio. Posteriormente explicaremos como están organizadas las tablas de nuestra base de datos y la manera en la que hemos accedido a ellas.

4.3.1. Funcionalidades del servidor

La principal tarea del servidor es procesar la información que requiere el usuario, dicha información se extrae de la base de datos mediante consultas en SQL. Para realizar estas consultas

debemos conocer muy bien la estructura de la base de datos. Como explicaremos en el siguiente apartado, hemos utilizado la herramienta *phpMyAdmin* con este propósito.

Una de las consultas que el usuario puede realizar es la lista de personas que han contribuido en el proyecto y su correo electrónico. Además para cada una de ellas se obtiene un historico por meses a lo largo de todo el proyecto y una lista de las contribuciones que han realizado y la fecha en la que lo hicieron. Aunque no parece gran cosa, es interesante extraer esta información, con ella podremos saber el número total de contribuyentes, la participación de cada uno de ellos en el proyecto y qué han aportado e incluso contactar con ellos si fuera necesario. Además, si el proyecto todavía sigue en desarrollo, sabremos cuantos de esos autores siguen colaborando en él y cuantos abandonaron.

La elaboración de estadísticas es otra de las funcionalidades del servidor. Podremos conocer datos tan importantes como la territorialidad de un proyecto, conocida también por *knowledge concentration*, esto es el porcentaje de ficheros que conforman el proyecto que han sido elaborados y actualizados por una sola persona, de tal manera que si dicha persona abandona el proyecto esa parte podría quedar sin nadie que la pudiese continuar. También aporta el dato de quién es la persona que más a colaborado con el proyecto y cuanta ha sido su contribución globalmente, así como un TOP 10 de los desarrolladores que más han participado.

Toda esta información es importante, pero cuando queremos analizar en profundidad un proyecto necesitamos ver con claridad una evolución histórica, algo que nos aporte conocimiento de como ha ido creciendo la aplicación a lo largo de los meses o años. Si necesitamos tomar medidas o simplemente analizar la productividad de los desarrolladores a lo largo del tiempo necesitaremos mostrar esta información a través de alguna herramienta que lo visualice de una manera clara. Para ello nos hemos valido de los gráficos. Son la herramienta más usada para realizar un seguimiento detallado en cualquier tipo de negocio o proyecto; nos permite ver de forma muy rápida, casi con un vistazo, los datos que queremos analizar.

Para poder visualizar las estadísticas de esta forma hemos tenido que procesar los datos extraídos de la base de datos, y modificarlos para que adquieran un formato de claves-valor en vez del formato que utiliza Python.

Finalmente todos estos datos han sido importados a HTML para poder visualizarlos en el navegador a través de plantillas o *templates* proporcionadas por Django. Este punto es un trabajo que se realiza a medias con el cliente: por un lado el servidor se encarga de elaborar dichas plantillas y generar el código JavaScript necesario tanto para los gráficos, como para ciertos aspectos dinámicos que hemos querido aportar en las páginas y para las partes que incluyen AJAX; y por el otro el cliente capaz de interpretar todo esto.

4.3.2. Implementación del servidor de datos

Para este proyecto hemos importado a nuestra base de datos los repositorios de GitHub a través de CVSAnaly. Veamos los pasos llevados a cabo detalladamente para conseguir este propósito:

- Lo primero de todo es clonar el repositorio desde GitHub, plataforma donde están alojados, a nuestro servidor usando la herramienta Git. En GitHub, se dispone de varios métodos para realizar este paso: mediante HTTPS, SSH o Subversion. En este caso hemos usado la URL HTTPS proporcionada con estos propósitos y el comando *"git clone url"* a través de nuestro terminal Linux.

- Posteriormente creamos una base de datos vacía en nuestro servidor de base de datos MySQL: *"create database nombre"*

- Y finalmente extraemos del repositorio clonado los datos para almacenarlos en la base de datos a través de la herramienta CVSAnalY.

Es importante conocer la estructura de tablas creadas por CVSAnalY para realizar la minería de datos y poder sacarles rendimiento. En la Figura 4.4 se muestra el diagrama de relación de las tablas creadas por CVSAnalY y los campos que contienen.

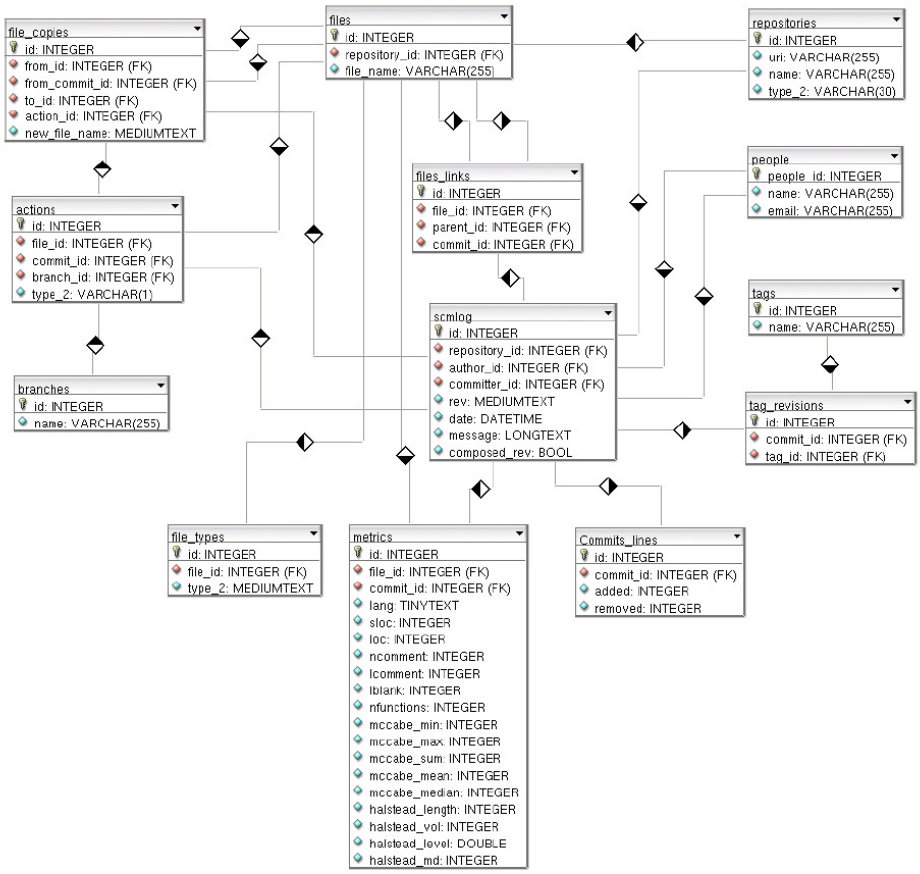

Figura 4.5: Diagrama relacional de CVSAnaly

La tabla más importante para sacar las estadísticas y métricas de este Proyecto Fin de Carrera es la de scmlog, que contiene los campos sobre el autor y commiter de cada commit, la fecha en la que se realizó, y el comentario que el autor hizo sobre dicho commit. Además utilizamos también las tablas de people para extraer información de las personas que han participado en el proyecto, reporsitories y actions para conocer los cambios en cada fichero.

Una herramienta muy útil para conocer la estructura de la base de datos y poder manejarla es phpMyAdmin. Nos facilita una visión clara sobre las tablas que forman cada base de datos.

phpMyAdmin es una herramienta escrita en PhP diseñada para administrar bases de datos en MySQL mediante una interfaz web. Es capaz de crear o eliminar bases de datos, modificarlas, añadir o eliminar tablas, administrar privilegios o ejecutar cualquier consulta en SQL. A parte, puede importar datos desde CVS y SQL o exportarlos a varios formatos como CVS, XML,

PDF, etc. Calcula estadísticas sobre cuantas consultas se han realizado al servidor o el tráfico obtenido incluso mostrarlas por gráfico.

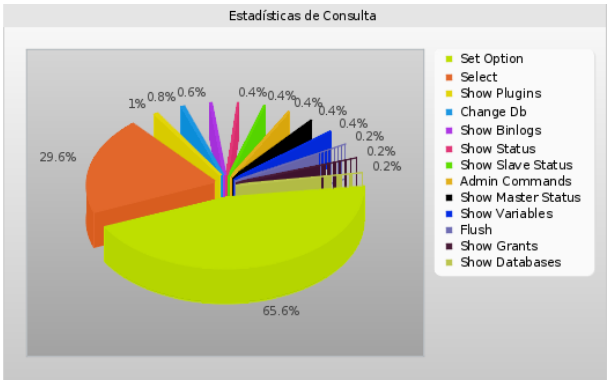

Figura 4.6: Estadísticas de consulta de las bases de datos usadas en el proyecto

Además ofrece una interfaz sencilla para mostrar las tablas de la base de datos que seleccionemos pudiendo acceder a cada una de ellas con solo un click y ver su contenido y estructura. Como podemos apreciar en las Figuras 4.6 y 4.7 en las que se muestran un listado de las tablas de la base de datos que hemos creado para Gedit y la estructura de una esas las tablas respectivamente , desde la interfaz de phpMyAdmin podremos modificar cualquiera de ellas.

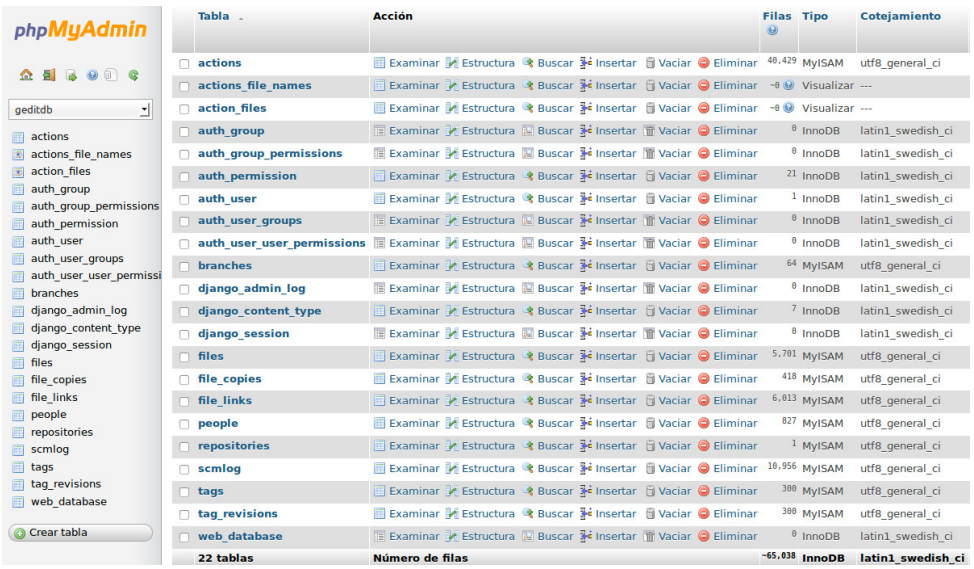

Figura 4.7: Tablas de la base de datos creada para alojar los repositorios del proyecto Gedit

	id	parent_id	file_id	commit_id	file_path
Editar Editar en línea Copiar Borrar	1	-1	1	1	gedit
Editar Editar en línea Copiar Borrar	2	1	2	1	gedit/commands.c
Editar Editar en línea Copiar Borrar	3	1	3	1	gedit/gE_about.c
Editar Editar en línea Copiar Borrar	4	1	4	1	gedit/gE_document.c
Editar Editar en línea Copiar Borrar	5	1	5	1	gedit/gE_files.c
Editar Editar en línea Copiar Borrar	6	1	6	1	gedit/gE_init.c
Editar Editar en línea Copiar Borrar	7	1	7	1	gedit/gE_prefs.c
Editar Editar en línea Copiar Borrar	8	1	8	1	gedit/gedit.c
Editar Editar en línea Copiar Borrar	9	1	9	1	gedit/menus.c
Editar Editar en línea Copiar Borrar	10	-1	10	1	src
Editar Editar en línea Copiar Borrar	11	10	11	1	src/commands.c
Editar Editar en línea Copiar Borrar	12	10	12	1	src/gE_about.c
Editar Editar en línea Copiar Borrar	13	10	13	1	src/gE_document.c
Editar Editar en línea Copiar Borrar	14	10	14	1	src/gE_files.c
Editar Editar en línea Copiar Borrar	15	10	15	1	src/gE_init.c
Editar Editar en línea Copiar Borrar	16	10	16	1	src/gE_prefs.c
Editar Editar en línea Copiar Borrar	17	10	17	1	src/gedit.c
Editar Editar en línea Copiar Borrar	18	10	18	1	src/menus.c
Editar Editar en línea Copiar Borrar	19	-1	19	3	AUTHORS
Editar Editar en línea Copiar Borrar	20	-1	20	3	COPYING
Editar Editar en línea Copiar Borrar	21	-1	21	3	INSTALL
Editar Editar en línea Copiar Borrar	22	-1	22	3	KNOWNBUGS
Editar Editar en línea Copiar Borrar	23	-1	23	3	NEWS
Editar Editar en línea Copiar Borrar	24	-1	24	3	README

Figura 4.8: Ejemplo del contenido de una tabla del proyecto Gedit

Para este Proyecto Fin de Carrera, además de las bases de datos exportadas de los proyectos clonados de GitHub, hemos necesitado crear una nuestra, con una relación de todos los proyectos que hemos alojado en MySQL y de los que podemos obtener estadísticas.

4.4. Diseño e implementación del cliente

La función principal del Cliente es la presentación de los datos obtenidos. Es el encargado de interpretar los documentos HTML y los códigos JavaScript y AJAX para mostrarlos por el navegador. Igualmente se encarga de enviar las peticiones del usuario al servidor y recibir su respuesta.

Para realizar las páginas que van a ser finalmente mostradas nos hemos basado en un sistema de plantillas o *Templates* que soporta Django. La finalidad de estas es separar la lógica de la aplicación de su presentación, para que un cambio en cualquiera de ellas no afecte a la otra parte. Estas plantillas son capaces de generar cualquier tipo de formato basado en texto, aunque

normalmente se usan para crear páginas en HTML, y es como las hemos usado en este proyecto. Desde el fichero *views.py* le pasamos los valores que queramos mostrar en contexto, es decir, en forma de variable y valor asociado en cada caso, usando la función *render()*.

Hoy en día es casi necesario desarrollar aplicaciones adaptadas a multiples dispositivos: PCs, móviles, tablets... Gracias a Bootstrap hemos conseguido diseñar una aplicación que se adapte al tamaño de cualquier pantalla, es decir, una aplicación con diseño responsivo. Como vemos en la siguiente figura, nuestro cliente web se adapta perfectamente a todos los tamaños.

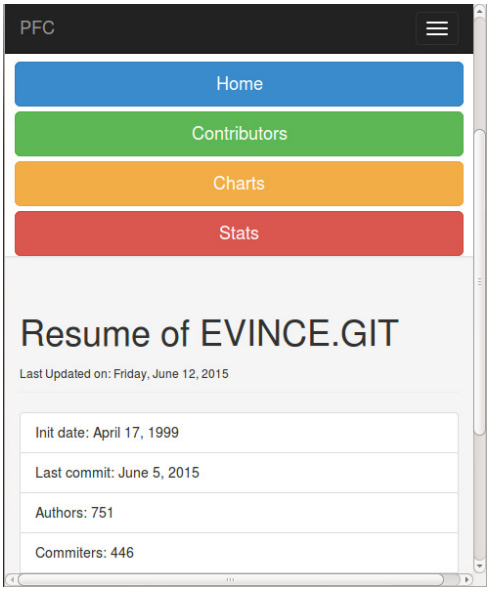

Figura 4.9: Diseño responsivo

Finalmente tendremos una serie de plantillas en código HTML, que junto con CSS y el framework Bootstrap y bootsnipp dan el aspecto que queremos mostrar. Como explicaremos más adelante, hemos incorporado pequeños scripts escritos en JavaScript y AJAX para crear páginas web dinámicas y rápidas.

La primera de las páginas que nos encontramos al acceder a la aplicación es la página principal en la que aparece un formulario para indicar que proyecto de software libre queremos analizar. En este formulario debemos poner el nombre de la base de datos donde se aloja nuestro proyecto. Justo debajo se muestra un listado de los proyectos disponibles. Este campo es obligatorio y deberá ser válido para acceder a la base de datos en cuestión, de no ser así se mostrará un mensaje por pantalla y tendremos la opción de introducir de nuevo el nombre.

Además, a la izquierda de la página aparece un mensaje de bienvenida con una breve descripción de la aplicación y un enlace al resumen escrito en esta memoria.

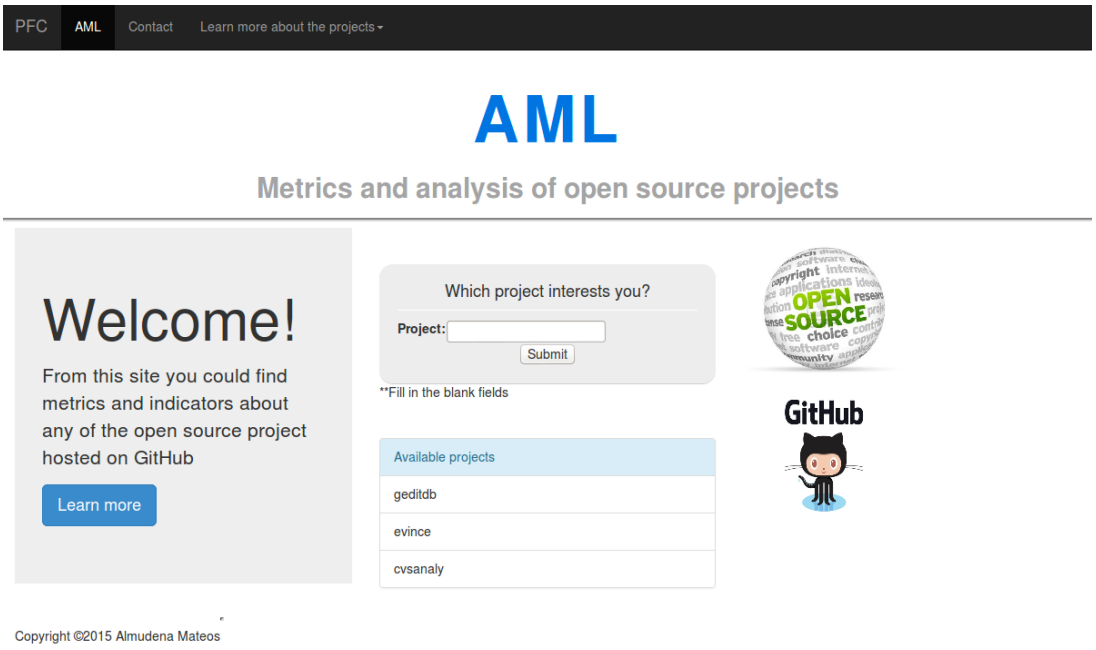

Figura 4.10: Página principal

Una vez hemos accedido, se muestra una página de inicio con los datos más generales del proyecto. A su vez, contaremos con un menú lateral para acceder a las diferentes funcionalidades que provee la aplicación.

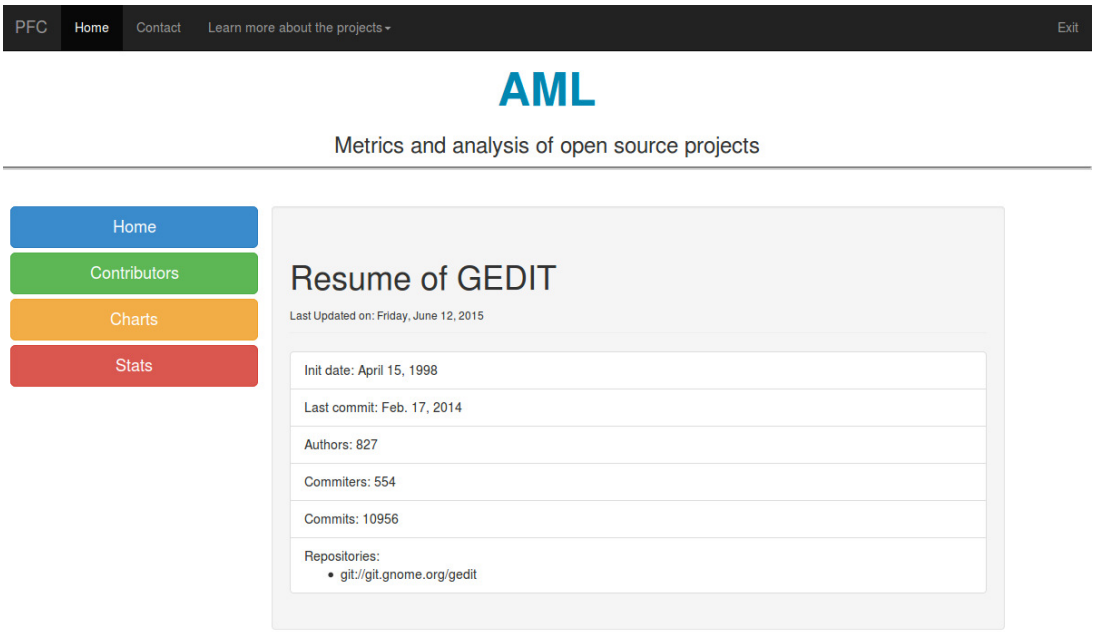

Figura 4.11: Página Home

En esta página encontraremos información tal y como la fecha de inicio y fin del proyecto, el número de autores y commiters que han participado en él, el número total de commits realizados y la lista de repositorios que tiene.

Si vamos navegando por el menú situado a la izquierda de cada página nos encontramos con las siguientes opciones:

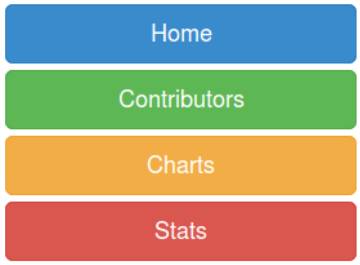

Figura 4.12: Menú lateral

- Contributors: Aquí encontraremos una lista de cada uno de las personas que han participado en el desarrollo del proyecto y su dirección de correo electrónico por si quisiéramos contactar con ellas. Si hacemos click en cualquiera de ellas aparecerá una gráfica sobre la evolución del desarrollador en cuestión en este proyecto durante los años que duro y un desplegable implementado con AJAX con los commits que han realizado con la fecha en que los hicieron y el comentario que dejaron en cada uno de ellos.

- Charts: Nos muestra un listado de los gráficos disponibles. Podremos acceder a cada gráfico haciendo click en su nombre sin necesidad de que se recargue la página de nuevo, gracias a que está implementado con AJAX. En una primera instancia hemos desarrollado 4 gráficos dinámicos, cada uno de ellos nos muestra el valor en cada punto solo con poner el ratón encima de el. Además tenemos la opción de descargarlo en formato PNG, JPEG, PDF, SVG o mandar directamente a la impresora. Para realizar esto hemos usado la librería de HighCharts e incrustado un pequeño script escrito en JavaScript en el código HTML. Este script recoge los datos que le pasamos desde el fichero *views.py*: los datos en forma de clave-valor, así como el eje de abscisas.

```
<script type="text/javascript">
var chart;
var chart2;
jQuery(document).ready(function() {
        chart = new Highcharts.Chart({
                chart: {{chart|safe}},
                title: {{title|safe}},
                subtitle: {
                        text: '',
                        x: -20
                },
                xAxis: {{xAxis|safe}},
                yAxis: {{yAxis|safe}},
                tooltip: {
                        formatter: function() {
                        return ''+ this.series.name +''+this.x +': '+ this.y;
                        }
                },
                legend: {
                        layout: 'vertical',
                        align: 'right',
                        verticalAlign: 'top',
                        x: -10,
                        y: 100,
                        borderWidth: 0
                },
                series: {{series|safe}}
        });

});
```

Figura 4.13: Código JavaScript en HTML para el procesamiento de los gráficos

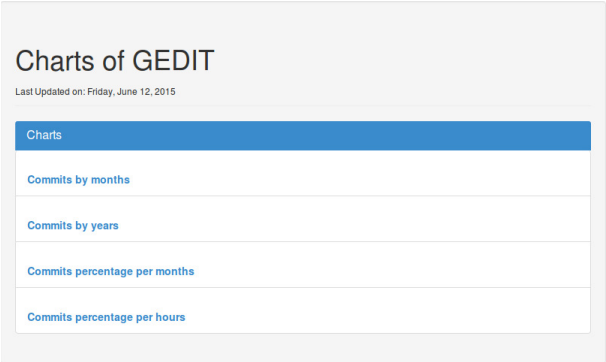

Figura 4.14: Listado de los gráficos disponibles en la aplicación

Si accedemos a ellos veremos que se han implementado gráficos de dos tipos: lineales y por columnas:

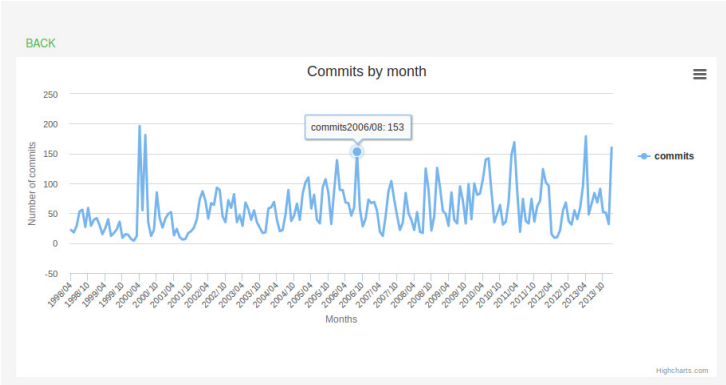

Figura 4.15: Gráfico lineal

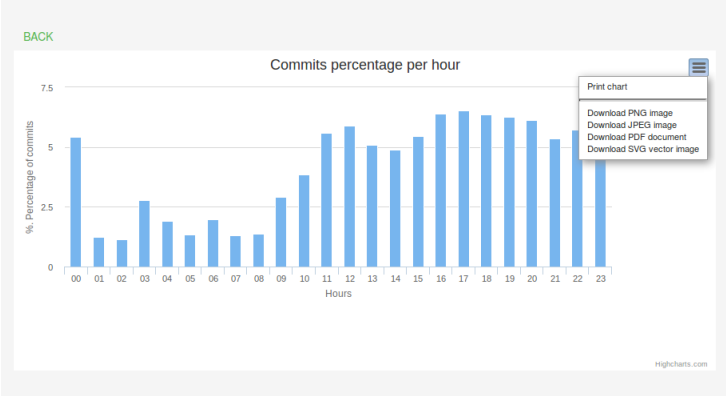

Figura 4.16: Gráfico en columnas

Vamos a explicar que información nos aporta cada uno de estos gráficos. Luego en el apartado de resultados veremos como podemos interpretar cada una de ellas y las conclusiones que podemos sacar.

- Nº de commits por meses: Muestra una evolución de los commits realizados por mes durante los años que ha durado el proyecto.
- Nº de commits por año: Es la misma representación que la gráfica anterior pero agrupando los commits por cada año del proyecto.
- Porcentaje de commits por meses: Representa el porcentaje de commits realizados cada mes del año.
- Porcentaje de commits por horas: Representa el porcentaje de commits realizados cada hora del día.

- Stats: En esta pestaña se muestran parte de las estadísticas del proyecto a analizar. Veremos datos como la territorialidad del proyecto, el desarrollador que más commits ha aportado al proyecto y cual es su peso en todo el proyecto, el porcentaje de autores que son commiters y el porcentaje de commits realizados por los autores del código. En el Capítulo de Resultados explicaremos con detalle que significan cada uno de estos datos. Además se muestra una tabla con el TOP 10 de los commiters que más han contribuido, .

Desde todas las páginas se muestra un menú en la parte superior a través del cual podremos acceder a la página de contacto donde se listan los participantes de este proyecto con sus datos de contacto y enlaces a las principales redes sociales del autor. Además desde cada una de las páginas podremos salir del proyecto en el que estemos y volver a la página principal mediante el enlace *Exit* situado en la parte superior derecha de cada una de ellas.

Figura 4.17: Página de contactos

Capítulo 5

Resultados

En este capítulo vamos a analizar los resultados obtenidos para entender el significado de cada métrica y gráfico que mostramos. Para ello realizaremos dos casos de estudio de dos proyectos de Software Libre: Gedit y eVince. Compararemos ambos proyectos examinando los datos y gráficos obtenidos y llegaremos a una conclusión sobre ellos.

5.1. Casos de estudio

Lo primero que debemos saber son las fechas de cada proyecto y su duración, de esta manera podremos saber si el proyecto todavía esta en fase de desarrollo o si ya ha finalizado. Además debemos conocer también el volumen de cada proyecto y el número de desarrolladores que han participado en cada uno de ellos. En la página de inicio de nuestra aplicación podremos encontrar estos datos para cada proyecto.

Como se puede ver en las siguientes imágenes para cada uno de los proyectos mencionados anteriormente:

- Gedit: Este proyecto comenzó en el año 1998 y la última vez que se actualizó fue en Febrero del 2014. Su duración total es de 16 años por lo que deducimos que se trata de un

proyecto de grandes dimensiones. En él colaboraron 827 autores y se realizaron un total de 10.956 commits.

- eVince: Por el contrario, eVince, a pesar de que lleva los mismos años de duración (1999-2015), esta todavía en vigor, el ultimo commit del que tenemos constancia ha sido realizado apenas unos días. 751 autores han formado parte de él con un total de 7298 commits.

Si comparamos estos datos deducimos que eVince es un proyecto que tiene una evolución más lenta en su desarrollo y con una comunidad de desarrolladores menor que en Gedit, pero que todavía se encuentra en funcionamiento. Esto nos asegura un programa actualizado y en continuo desarrollo, lo que nos garantiza soporte y fiabilidad. Esto no implica que Gedit sea poco fiable, estamos hablando de un proyecto consolidado y uno de los editores de texto por excelencia de GNU/Linux, pero lleva más de un año sin actualizarse.

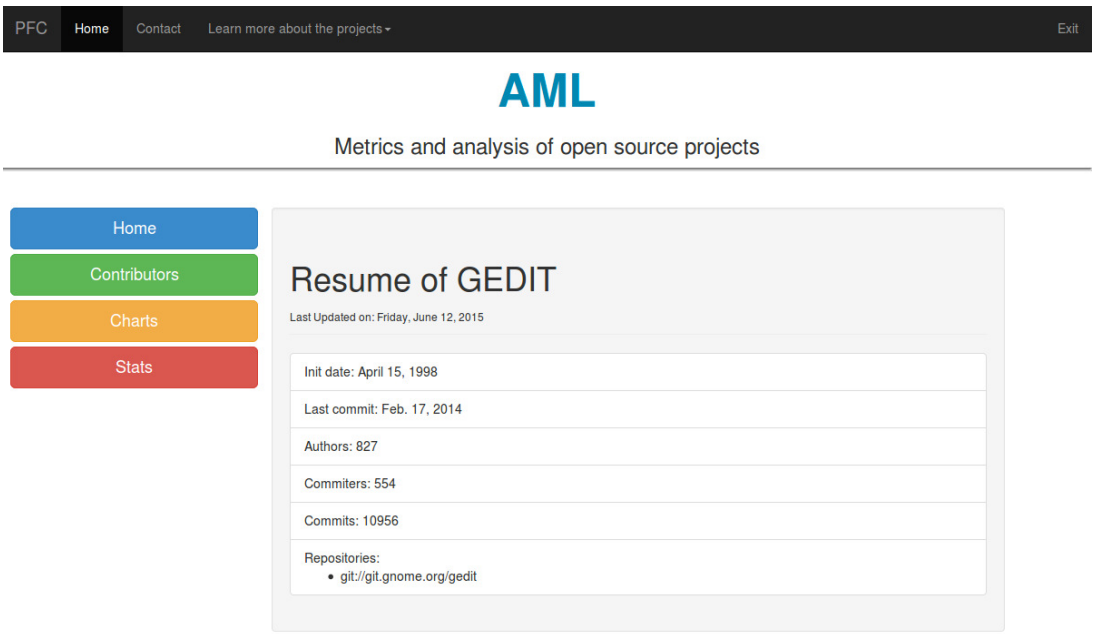

Figura 5.1: Página home Gedit

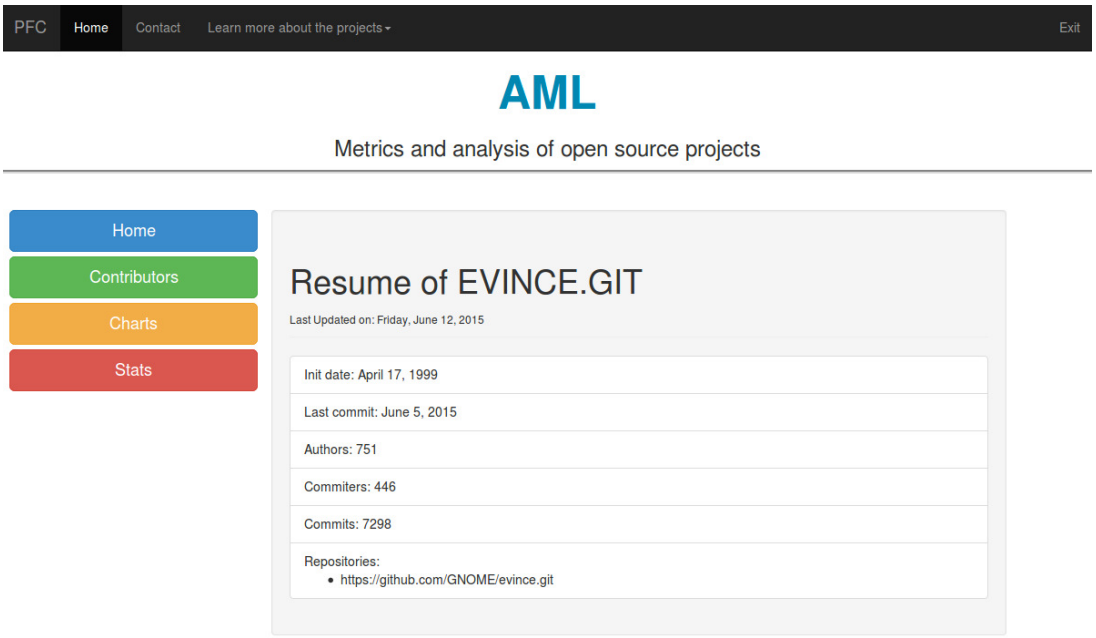

Figura 5.2: Página home eVince

Si atendemos a las estadísticas del proyecto, en el apartado “Stats” de la aplicación, podremos conocer datos de gran interés.

La territorialidad es un dato muy importante en un proyecto de Software Libre, nos indica cuantos ficheros del proyecto han sido elaborados por una sola persona. Con este dato podemos intuir el futuro del proyecto, si tuviese una territorialidad muy alta, por encima de un 80 %, significaría que este dependería de una o pocas personas, que si dejaran de desarrollar la aplicación podría conllevar problemas que irían apareciendo a medio plazo, como la falta de soporte, actualizaciones, incompatibilidades de versiones con navegadores o sistemas operativos nuevos...

En nuestro caso, para ambos proyectos, encontramos una territorialidad en torno al 50 %, algo superior en eVince con un 57,76 %. Con esto vemos que, aunque ambos tienen este dato dentro de un rango positivo, es algo superior el de eVince, y por tanto peor.

Otro dato interesante es el porcentaje de autores que son a la vez commiters. Para entender esto vamos a definir cada termino:

- Autores: aquellas personas que han desarrollado código.
- Commiters: son los miembros del proyecto que aplican los cambios.

Para el caso de Gedit un 66,99 % de los autores son a su vez commiters y han contribuido con un 87,57 % al total de commits del proyecto. Veamos que significa esto: normalmente los autores que no aplican los cambios son personas ajenas a los repositiorios que contribuyen a mejorar el proyecto añadiendo nuevas funcionalidades o corrigiendo errores: clonan el repositorio en local, realizan su contribución y hacen un pull para que la persona dueña de ese repositorio apruebe la modificación. Un 12,43 % de los commits totales del proyecto han sido realizados de esta manera.

Sin embargo, en eVince, solo un 59 % de las personas que han contribuido al proyecto son commiters, y se han encargado del 81,75 % de este. Aquí ha aumentado el porcentaje de personas que han contribuido sin ser dueñas de los repositorios a un 41 % y el porcentaje del proyecto hecho por ellas a un 18,25 %.

A parte de estos datos, en este punto podremos conocer también quien es el autor que más a contribuido en el proyecto y cuanto es el porcentaje de su participación en el proyecto, así sabremos si el grueso de un proyecto ha sido desarrollado por una persona o por un equipo de trabajo, independientemente del resto de personas que colaboren en actualizarlo o mejorar errores.

Por ejemplo en el caso de Gedit su mayor desarrollador se llama Paolo Borelli, que aplicó un total de 1085 commits lo que supone un 9,9 % del proyecto. A este le siguen muy de cerca los desarrolladores que ocupan el segundo y tercer puesto, con un total de 926 y 735 commits respectivamente.

Para eVince, su mayor contribuidor es Carlos García Campos, con un total de 995 commits que suman un 13,63 % del proyecto, más alejado del segundo y tercer puesto. El trabajo en este

proyecto ha estado menos repartido que para Gedit.

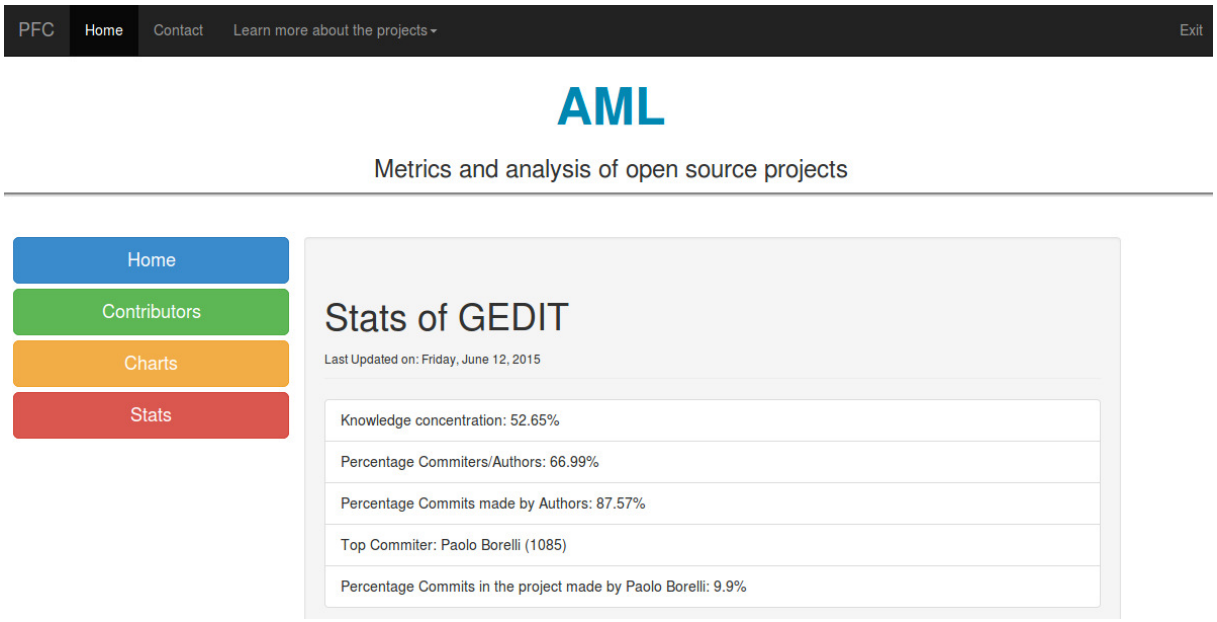

Figura 5.3: Estadísticas Gedit

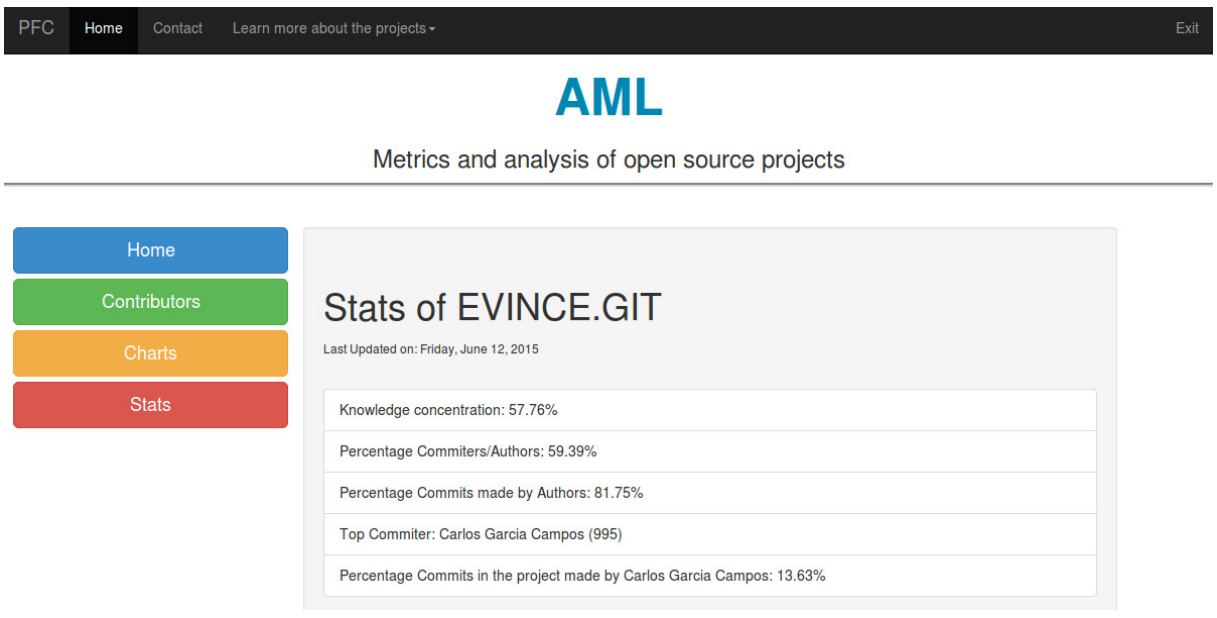

Figura 5.4: Estadísticas eVince

Most active contributors

Author name	Commits
Paolo Borelli	1085
Ignacio Casal Quinteiro	926
Paolo Maggi	735
Paolo Borelli	407
Jesse van den Kieboom	238
gedit	236
Kjartan Maraas	195
Jim Campbell	150
Sbastien Wilmet	145
Jose Maria Celorio	145

Figura 5.5: TOP 10 Gedit

Most active contributors

Author name	Commits
Carlos Garcia Campos	995
Carlos Garcia Campos	483
Nickolay V. Shmyrev	426
Marco Pesenti Gritti	353
Christian Persch	261
Christian Persch	99
Jonathan Blandford	97
Hib Eris	97
Daniel Mustieles	91
Martin Kretzschmar	91

Figura 5.6: TOP 10 eVince

Continuamos ahora analizando los gráficos mostrados en el apartado de “Charts” y compararemos el resultado obtenido en cada uno de ellos:

- Commits by months:

  Este gráfico nos muestra una evolución del proyecto por meses desde sus inicios hasta la fecha del ultimo commit realizado. Con él podremos comprobar cuales han sido los periodos de tiempo de mayor y menor desarrollo del proyecto.

  En el gráfico de Gedit vemos que los dos primeros años fueron los que menos commits se realizaron, podemos intuir que esta es la etapa en la que se desarrollo la primera versión

del programa por los miembros del proyecto. Después, en el primer semestre del 2000, se produjo el mayor volumen de trabajo y, en los años sucesivos los picos de trabajo se concentran también en este periodo del año, esto nos lleva a pensar que han lanzado actualizaciones cada año a principios del segundo semestre, hasta el 2014, año en el que data su última versión estable. Durante estos años se hicieron una media de 75 commits por mes.

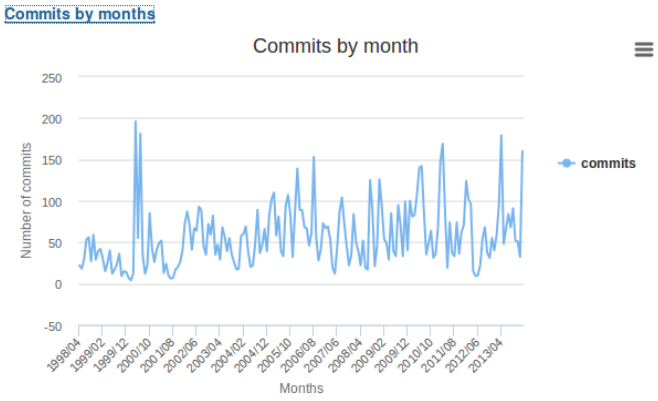

Figura 5.7: Gráfico: Commits by month Gedit

Para el caso de eVince, y como comentamos anteriormente, su evolución es más lenta, durante casi los 6 primeros años, el mes que más commits tiene registrados fue Agosto del 1999 con un total de 40 commits, el resto de los meses apenas hubo cambios. A finales del 2004 la evolución cambia drasticamente entrando en los meses de más volumen de trabajo, hasta mediados del 2005. Si indagamos en su historia, esto coincide con la fecha en la que se lanzo la versión 2.12 de GNOME, la cual incluía eVince por primera vez.

En fechas posteriores se produce una evolución más o menos lineal con una media de unos 60 commits por mes, siendo los más picos negativos en los meses de Noviembre y Diciembre, lo que se puede interpretar como actualizaciones o nuevas versiones lanzadas posteriormente.

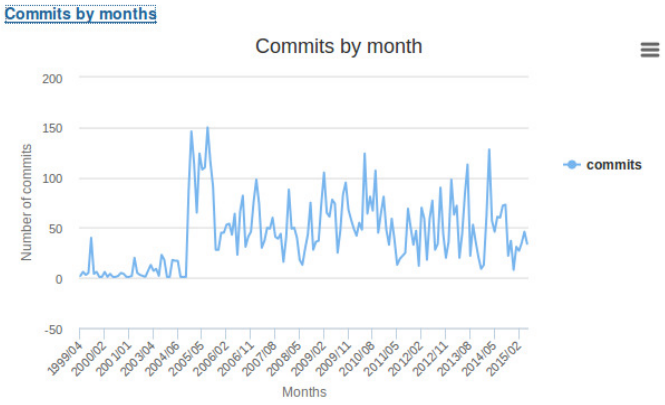

Figura 5.8: Gráfico: Commits by month eVince

- Commits by years:

  Este gráfico representa también la evolución del proyecto desde un nivel más alto de detalle, por años.

  Si nos fijamos en el gráfico de Gedit vemos que, aunque no es un incremento lineal, el número de contribuciones ha aumentado considerablemente desde sus inicios hasta la última modificación de la que tenemos constancia.

  Si contrarrestamos los datos obtenidos, por ejemplo, en Abril del 2011 Gedit se tradujo al Checo, Español, Japones y Eslovaco y saco una versión disponible con OpenBSD (SO Unix), entre otras actualizaciones, lo que da sentido a que el 2010 sea el año que más contribuciones hubo según nuestras estadísticas.

Figura 5.9: Gráfico: Commits by years Gedit

En el caso de eVince, con este gráfico se puede ver claramente que los primeros años no hubo apenas contribuciones, hasta el año 2005, que, como hemos comentado anteriormente, eVince fue incluido en GNOME. A partir de dicho año las aportaciones han sido muy lineales, destacando los años 2009 y 2010. Estos años fueron numerosas las actualizaciones del programa: se mejoró el soporte de impresión, se añadió la opción de un modo de colores invertido, se actualizó la presentación y la capacidad para anotar archivos en PDF... entre otras muchas mejoras.

Figura 5.10: Gráfico: Commits by year eVince

- Percentage commits per months:

Una métrica interesante de analizar son los meses en los cuales ha habido más contribuciones al proyecto. Conocer este dato nos ayuda a saber cuando hay más actualizaciones o mejoras del programa.

En el caso de Gedit, si atendemos al histórico de las versiones que han lanzado, desde el 2011 han publicado nuevas versiones los meses de Marzo y Septiembre. A priori esto no es lo que se muestra en el gráfico, pero si lo analizamos detenidamente vemos que justo dos meses antes de cada nueva versión se producen los meses de más contribuciones.

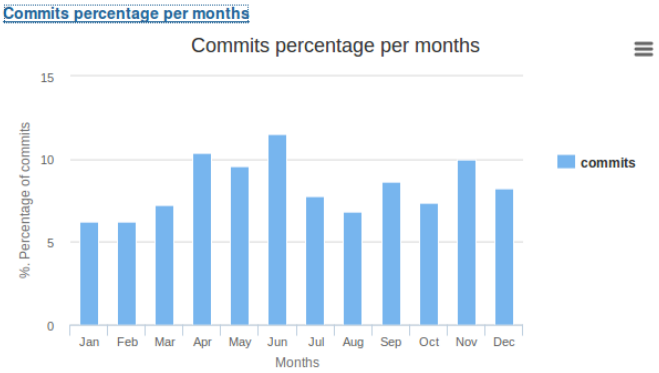

Figura 5.11: Gráfico: Commits percentage per months Gedit

Esto tiene su lógica, si nos paramos a pensar, cuando queremos lanzar una nueva versión trabajamos durante cierto tiempo en los cambios que queremos hacer, y posteriormente debe haber una fase para implementarlos y depurarlos antes de sacar dichas actualizaciones al mercado.

Para el caso de eVince no podemos aplicar esta deducción, prácticamente tiene la misma contribución todo el año, a excepción de los tres primeros meses donde hay un descenso claro del porcentaje de contribuciones.

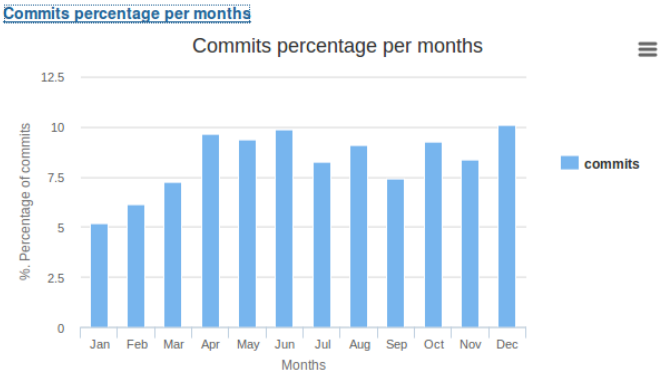

Figura 5.12: Gráfico: Commits percentage per months eVince

- Percentage commits per hours:

  En todo proyecto es muy importante conocer las horas del día en que se rinde más, para lograr una mejor planificación. En el caso de software libre, es interesante conocer este

dato ya que muchas de las contribuciones son por parte de gente ajena al proyecto y que colabora con él fuera de su horario laboral.

Esto lo podemos corroborar en el siguiente gráfico, donde el volumen más grande de commits se produce por la tarde, a partir de las 16:00h, llegando incluso hasta el horario nocturno.

También nos aporta información sobre el origen de los desarrolladores del programa y la dispersión geográfica de estos. Si hubiesen colaborado programadores de todos los continentes por igual no notaríamos apenas diferencia en los commits realizados por hora debido a las distintas franjas horarias que hay, por ejemplo, entre América, Europa y Asia.

En Gedit, puesto que los desarrolladores principales son Europeos: Paolo Maggi (Italia), Paolo Borelli (Italia), Steve Frécinaux (Bélgica), etc.. y hay una franja horaria claramente definida en el gráfico, podemos suponer que la mayor parte del proyecto se ha realizado desde Europa.

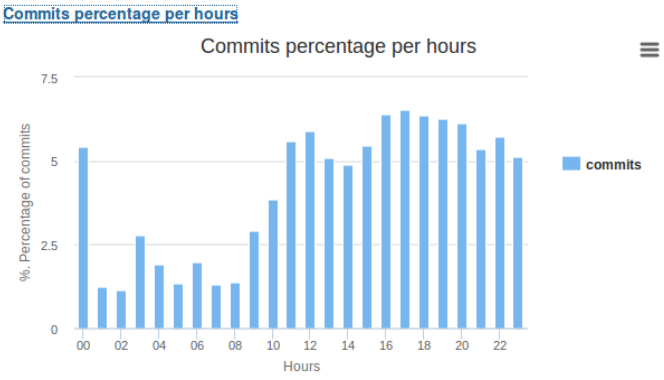

Figura 5.13: Gráfico: Commits percentage per hours Gedit

En eVince ocurre lo mismo, la franja de horas en las que se ha desarrollado más es a partir del medio día y se concentra en horario vespertino. Si además buscamos la nacionalidad de sus mayores contribuidores vemos que son también Europeos: Carlos García Campos (España), Nickolay V. Shmyrev (Rusia), etc. por lo que deducimos también que el proyecto se ha desarrollado en su mayoría en Europa.

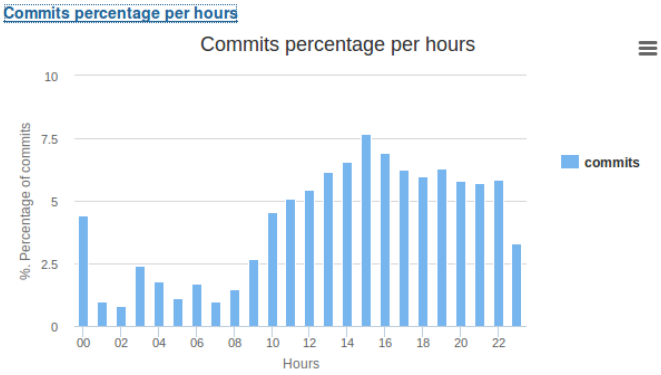

Figura 5.14: Gráfico: Commits percentage per hours eVince

Capítulo 6

Conclusiones

6.1. Consecución de objetivos

En esta sección echaremos la vista hacia atrás, hacia los objetivos que nos marcamos al comienzo de este Proyecto Fin de Carrera para ver si los hemos cumplido satisfactoriamente. Para ello contestaremos a las preguntas formuladas en el Capítulo 2.

Lo primero es analizar si hemos logrado los objetivos específicos fijados para construir la aplicación. Según hemos visto en el apartado de "Implementación y diseño", Capítulo 4, hemos conseguido crear una aplicación web capaz de mostrar al usuario las métricas y estadísticas oportunas elaboradas a partir de los datos obtenidos de los proyectos alojados en la plataforma de desarrollo colaborativo GitHub. Para ello hemos creado una base de datos donde almacenar los proyectos, hemos implementado toda la lógica necesaria capaz de elaborar dichas estadísticas y, finalmente, mostrarlas al usuario a través de una interfaz intuitiva. Por lo que podemos concluir que los objetivos específicos marcados al comienzo del proyecto se han conseguido.

Pero el objetivo final no es simplemente eso, sino que la aplicación desarrollada sea capaz de contestar a ciertas preguntas que nos podemos plantear sobre cualquier proyecto y profundizar en así él.

En muchas ocasiones para conocer un proyecto lo más importante es conocer su comunidad de desarrolladores. Las personas que han participado en él dicen mucho sobre la calidad de dicho proyecto. Por eso esta aplicación se ha centrado más en analizar a la gente que contribuye en el desarrollo de un software que al software en sí.

Al principio de esta memoria nos planteamos ciertas preguntas que creemos que son de interés a la hora de analizar un proyecto de Software Libre. En este Proyecto Fin de Carrera hemos intentado crear una aplicación capaz de responderlas.

Una de las cuestiones más importantes a la hora de evaluar un software es si saber si tiene un ciclo de vida actualizado y si sus contribuidores siguen desarrollando en aportar nuevas funcionalidades y mejoras a este. Analizando los datos obtenidos por nuestra aplicación podemos ver para cada proyecto si está formado por una comunidad de desarrolladores férrea y que continúa formando parte de él. Si atendemos a la fecha en la que se realizaron los últimos commits e interpretando los gráficos sobre el número de commits, podemos ver la evolución que lleva el proyecto en los últimos meses o años y sacar conclusiones sobre su actividad reciente, respondiendo así a la pregunta de si el proyecto sigue vivo y actualizado. Además el número de contribuidores nos da una idea de la solidez del proyecto.

Debemos conocer también como y cuanto ha contribuido cada persona. Aquí nos encontramos el problema de distinguir a los que son autores de los que son commiters y tienen permisos de modificar el código. Haciendo clara esta distinción sabremos si cada persona ha colaborado desde un punto externo del proyecto o forma parte de los miembros que lo gestionan. De esta manera hemos calculado el porcentaje de contribuidores que son miembros del proyecto y cuanto ha sido su aportación frente a la aportación de los colaboradores externos.

Así mismo, la aplicación nos permite seguir la evolución en número de commits para cada persona desde que empezó a contribuir en el proyecto, así como un listado de las aportaciones que ha realizado durante toda su participación. Esto nos ayuda a la hora de evaluar el trabajo de cada desarrollador, si continua formando parte de la comunidad y de qué manera ha contribuido con el proyecto.

La aplicación también nos permite conocer las personas que más han colaborado y su peso

en el proyecto; podremos llevar un seguimiento más detallado sobre dichas personas y saber si aún siguen involucradas con el software en cuestión, que, junto con el dato de territorialidad del proyecto, nos llevan a conocer la estabilidad y la confianza que trasmite el proyecto.

Si queremos conocer los tiempos de productividad, la aplicación nos ofrece los datos necesarios medidos tanto en los meses del año como en horas del día. Además analizando los datos mostrados en el gráfico de horas, nos podremos hacer una idea de la dispersión geográfica de los desarrolladores. Este dato solo se puede concretar en términos de franjas horarias, no podremos hacer la distinción por paises, y es meramente una aproximación.

Con todo ello, una vez finalizado este Proyecto Fin de Carrera y si intentamos responder a las preguntas formuladas al comienzo, llegamos a la conclusión de que hemos logrado nuestro objetivo: diseñar e implementar una aplicación web que permita analizar y conocer la comunidad de cualquier proyecto de software libre alojado en GitHub.

6.2. Aplicación de lo aprendido

Para realizar este Proyecto Fin de Carrera hemos utilizado numerosas tecnologías y conocimientos. Alguna de ellas eran totalmente nuevas para mi: mysql, sql, bootstrap, highcharts... y he tenido que documentarme y aprender su funcionamiento para ponerlas en práctica en este proyecto.

Sin embargo, una de las cosas que más me ha gustado de esta última etapa de la carrera ha sido usar el conocimiento adquirido a lo largo de todos estos años en numerosas asignaturas y aunandolo en mi PFC. Además, tal y como está diseñada mi aplicación he puesto en práctica las lecciones aprendidas en las tres asignaturas con las que más he disfrutado y que definen la estructura y funcionalidad de mi aplicación: Sistemas Telemáticos II, SAT y Proyectos.

1. Sistemas Telemáticos II: A pesar de que el objetivo principal de esta asignatura es aprender a programar en Java, se hizo especial hincapié en la estructura cliente-servidor y en el traspaso de información entre estos (peticiones, protocolos...). A pesar de que en este

proyecto no hemos definido el protocolo de comunicación, ya que se basa en peticiones HTTP, partíamos de un conocimiento previo sobre la arquitectura de este tipo de aplicaciones.

2. SAT: Esta es sin duda la asignatura más relacionada con el diseño de mi aplicación. Todas las tecnologías base utilizadas en este proyecto las había aprendido previamente en esta asignatura: Python, Django, AJAX (una parte al menos), HTML, CSS..

3. Proyectos: En esta asignatura aprendimos los conceptos básicos para la gestión de proyectos así como las herramientas necesarias para analizarlos tanto cuantitativa como cualitativamente: Métricas, KPIs, Indicadores... Nos enseñaron la importancia y la repercusión que tenía en el mundo real tener un feedback sobre cualquier proyecto realizado. Se puede decir que la lógica de esta aplicación se basa en esta asignatura.

4. Mención especial requieren todas las asignaturas del GSyC, por enseñarnos la cultura del Software Libre.

6.3. Lecciones aprendidas

Como he mencionado anteriormente, a pesar de que ya partía con conocimiento de muchas de las tecnologías empleadas, he aprendido otras muchas que considero son de gran utilidad:

1. MySQL: Este es el servidor de base de datos de Software Libre más usado de todos. Gracias a este proyecto he aprendido esta tecnología que aporta gran valor de cara a mi futuro profesional.

2. SQL: Es sin duda la tecnología que más me ha gustado aprender. Para mi el lenguaje de base de datos es fundamental en el conocimiento de cualquier ingeniero y una gran carencia en el plan de estudios de nuestra carrera. Creo que me va a aportar muchos beneficios el haber adquirido este conocimiento de cara a mi carrera laboral.

3. JavaSript: Este lenguaje de scripting es hoy en día el más usado y muy útil para construir páginas web dinámicas.

4. Bootstrap: No conocía la existencia de este framework hasta que comencé el Proyecto Fin de Carrera. Es de gran utilidad para diseñar la interfaz de usuario en páginas web y junto con Bootsnipp hacen realmente maravillas en la presentación. Es, quizás, la tecnología que más me ha impresionado.

5. HighCharts: Esta librería ofrece una gran cantidad de gráficos dinámicos en múltiples formatos que aportan un valor añadido a cualquier aplicación de estas características.

6.4. Trabajos futuros

Toda aplicación requiere actualizaciones y versiones que implementen nuevas funcionalidades. El objetivo principal de este proyecto está cumplido, hemos creado una aplicación capaz de analizar proyectos de Software Libre y que muestre a través de un navegador ciertas métricas. Pero la minería de datos es un campo inmenso, siempre se pueden descubrir más patrones y sacar más estadísticas a partir de ellos.

Otra de las cosas que pueden mejorar esta aplicación, es que sea capaz de generar por sí misma reports o dashboards de un proyecto y los visualize por pantalla generando versiones imprimibles o incluso que se mánden automáticamente a un correo dado.

Además se podría configurar un servidor más potente; en mi caso me decantaría por Apache con mod_python (un plugin de Apache) sobre una máquina Linux como servidor web, ya que ofrece una configuración robusta.

Bibliografía

[1] VALORACIÓN DEL SOFTWARE LIBRE EN LA SOCIEDAD *http://www.portalprogramas.com/software-libre/informe*

[2] REDHAT *http://www.redhat.com/en*

[3] SOFTWARE LIBRE *http://www.gnu.org/philosophy/free-sw.es.html*

[4] GITHUB *https://github.com/*

[5] GIT *https://git-scm.com/*

[6] DJANGO, *https://www.djangoproject.com*,

[7] PYTHON *https://www.python.org*

[8] MYSQL *https://www.mysql.com/*

[9] SQL *http://www.w3schools.com/sql/*

[10] AJAX *http://jquery.com/download/*

[11] JAVASCRIPT *http://librosweb.es/libro/javascript/*

[12] CVSANALY (Última consulta, Mayo 2015)
https://github.com/MetricsGrimoire/CVSAnalY
http://herraiz.org/papers/english/

[13] GEDIT (Última consulta, Mayo 2015)
https://github.com/GNOME/gedit
https://en.wikipedia.org/wiki/Gedit

[14] EVINCE (Última consulta, Mayo 2015)
https://github.com/GNOME/evince
http://freecode.com/projects/evince/releases

[15] BOOTSTRAP *http://getbootstrap.com/*

[16] BOOTSNIPP *http://bootsnipp.com/*

[17] HIGHCHARTS *http://www.highcharts.com/*