



GRADO EN INGENIERÍA DE SISTEMAS DE
TELECOMUNICACIÓN

Curso Académico 2014/2015

Trabajo Fin de Grado

SISTEMA DE SEGUIMIENTO DE PARÁMETROS
DE DESARROLLO DE PROYECTOS SOFTWARE

Autor : Javier Sanjuán Orgaz

Tutor : Dr. Jesús María González Barahona

*Dedicado a
mi familia*

Agradecimientos

Como dijo el gran Pitágoras, “La felicidad consiste en saber unir el final con el principio”, y escribiendo estas líneas pongo final a una etapa y empiezo una nueva experiencia en mi vida, donde esta singladura que acabo de atravesar, espero que sea la primera piedra de un gran futuro prometedor.

Sinceramente, nunca pensé que mi trabajo fin de grado fuera una aplicación, y sobre todo que fuera de programación, aun recuerdo con añoranza cuando en primero de carrera cursé mi primera asignatura de programación la cual me sonaba a “chino”, recuerdo el revuelo entre mis compañeros tras la primera clase, donde veíamos imposible entender todo aquello, pero poco a poco según iba pasando el tiempo, te dabas cuentas que ibas comprendiendo la programación, entendías el funcionamiento, y poco a poco empezaba a engancharme, y suponía una gran satisfacción cuando veías funcionar tu mini aplicación la cual te hacía sentir el hombre más feliz del mundo.

Pero fue realmente fue en tercero, cuando apareció en mi camino, una asignatura llamada Servicios y Aplicaciones en Redes de Ordenadores, cuyo profesor era Jesús María González Barahona, mi tutor. El fué el primero que empezó a enseñarme el gran mundo de las aplicaciones web, relacionando todos los conceptos teóricos, con el gran mundo de la Web, nunca olvidaré la entrega final de dicha asignatura, suponía mi primera aplicación Web. Además quiero agradecerle su ayuda en el proyecto con sus constantes ideas y su sabiduría transmitida. En estos últimos meses, muchas personas me han ayudado. Empezando por mis dos compañeros de universidad Jesús Alonso y Quan Zhou, que desde el primer semestre que nos conocimos hemos trabajado juntos en todo, formando un gran equipo. También me gustaría agradecer a los profesores, que de una forma o de otra han dejado una huella en mí, con sus conocimientos y de los cuales me llevo un gran recuerdo de una bonita aunque dura etapa en la universidad. Por supuesto, quiero agradecer a mi familia el apoyo que me ha dado durante estos años, ellos siempre

me han ayudado y animado en los momentos más difíciles y creo que esta etapa a la cual estoy poniendo punto final con estas líneas, ellos tienen en este logro su parte de reconocimiento.

Por último, agradecerle a ti, al lector, que gastes parte de tu tiempo en compartir contigo este proyecto el cual me ha supuesto un gran esfuerzo.

Resumen

El objeto de este proyecto es construir un dashboard para la visualización de datos, para la elaboración se han utilizado ficheros JSON, los cuales ya venían dados, mientras el grueso de nuestro trabajo ha consistido en el tratamiento de los ficheros y la visualización de dichos ficheros mediante bibliotecas de representación gráfica.

Estos datos se han producido analizando repositorios de desarrollo software con las herramientas de MetricsGrimoire, y se han postprocesado, analizándolos esos datos con Grimoire-Lib. La aplicación muestra los datos de los documentos JSON de forma detallada e interactiva, además permite al usuario poder navegar y escarbar en estos datos.

La aplicación esta basada en tecnología HTML5, con la premisa de conseguir una Single Page Applications (SPA), el deseo de utilizar este patrón es, porque en la actualidad existe una mayor expectativa por parte de los usuarios de que las aplicaciones web se mimeticen con las aplicaciones tradicionales de escritorio, ofreciendo su inmediatez, velocidad y fluidez en el uso. Este dashboard consigue este objetivo con la utilización HTML5, CSS3.

A la hora de desarrollar la aplicación, pese a que no es para el público general, no hay que olvidar que estamos en el siglo XXI por lo que es importante que la aplicación sea recursiva y se pueda visionar en cualquier dispositivo, esto se ha conseguido mediante la tecnología, Bootstrap.

Summary

Here comes a translation of the “Resumen” into English. Please, double check it for correct grammar and spelling.

Índice general

1. Introducción	1
1.1. Descripción del problema	2
1.2. Objetivo Principal	3
1.3. Requisitos	4
1.4. Modelo de desarrollo	5
2. Estado de Arte y Contexto	9
2.1. Software libre	9
2.1.1. Ingeniería del Software Libre	9
2.2. Cloud Computing	12
2.3. Openstack	13
2.4. Métricas de Grimoire	14
2.5. VizGrimoire	15
2.6. GitHub	15
2.7. Bitergia	17
2.8. OpenHub	18
3. Tecnologías Utilizadas	21
3.1. Bibliotecas de Representación Gráfica	21
3.1.1. HighChart.js	21
3.1.2. Chart.js	22
3.2. JavaScript	22
3.3. HTTP	24
3.4. HTML5	25

3.5. CSS	26
3.6. BootStrap	28
3.7. jQuery	29
3.8. jQuery Ui	30
4. Proceso de Desarrollo	31
4.1. Iteración 0. Estudio previo	32
4.1.1. Bibliotecas Manejadas	32
4.1.2. Ejercicios JavaScript	33
4.2. Iteración 1	33
4.2.1. Obteniendo JSON	33
4.2.2. Eventos Trigger	35
4.2.3. Parseo JSON	37
4.3. Iteración 2	38
4.3.1. Entendiendo HighChart	38
4.3.2. Introducción de Eventos	44
4.3.3. Implementación	46
4.4. Iteración 3: Diseño BootStrap	48
4.4.1. Gráficos Charts	48
4.4.2. Tablas de Ordenación	52
4.4.3. Tabs	53
4.4.4. Fichero Properties	53
4.4.5. Opción de Volver Atrás	54
4.5. Iteración 4: Deploy en un Servidor	56
4.5.1. Configuración CutFTP	57
5. Detalles del Producto Final	59
5.1. Arquitectura general	59
5.2. Descripción funcional completa del Dashboard	62
6. Conclusiones	69
6.1. Lecciones aprendidas	70

<i>ÍNDICE GENERAL</i>	XI
6.2. Conocimientos aplicados	71
6.3. Trabajos futuros	72
Bibliografía	73

Índice de figuras

2.1. Apariencia de OpenStack	14
2.2. Estadísticas ofrecidas por GitHub	16
2.3. Apariencia de Bitergia	17
2.4. Apariencia de Open Hub	18
2.5. Dashboard de OpenHub	19
3.1. Charts ofrecidos por la biblioteca Chart.js	22
4.1. Esquema de la función cargar fichero	37
4.2. Ejemplo Representación HighCharts	41
4.3. Antes de Disparar el Evento	47
4.4. Después de activar el Evento	47
4.5. Obtención datos de un usuario específico	48
4.6. Chart empleado para las gráficas	49
4.7. Popup de información	50
4.8. Dejar visible solo la cabecera del chart	51
4.9. Ampliar Chart a pantalla completa	51
4.10. Tablas empleadas para la ordenación	52
4.11. Ejemplo de Pestañas utilizadas en la app	53
5.1. Estructura de la Obtención del Fichero	60
5.2. Arquitectura General	61
5.3. Loading Principal	62
5.4. Pestaña General	63
5.5. Ampliar Informacion Icono +	63

5.6. Pestaña Business	64
5.7. Evento Activado Business	64
5.8. Pestaña Demographic	65
5.9. Evento Activado Demographic	65
5.10. Evento Activado Demographic Usuario	66
5.11. Pestaña Source Code Management	66
5.12. Evento activado en la Tabla	67
5.13. Pestaña Abstract	68
5.14. Pestaña Company Information	68

Capítulo 1

Introducción

Los proyectos de desarrollo software son complejos y son difíciles de entender ya que hace falta muchos datos, este proyecto trata de aproximarse a este problema, mediante la implementación de un dashboard, que obtiene los datos de repositorios y a partir de aquí se encarga de visualizarlo, creando una interfaz muy potente para el usuario, con el objetivo de que el usuario se abstraiga de los datos y navegue por la aplicación comodamente.

Los datos se obtienen de ficheros JSON, estos datos son producidos mediante MetricsGrimoire (esta parte del proceso ya viene dada). Con estos datos se construye una interfaz, que esta interconectada, con el objetivo de que el usuario pueda escarbar en los datos y buscar la información que realmente le interesa.

Cuando hablamos de escarbar en los datos, nos referimos a ganar profundidad, esto quiere decir que empezando en un conjunto de muestras muy amplio, ir navegando por las gráficas mediante los eventos hasta llegar a una única muestra.

Todo esto se ha conseguido mediante código JavaScript para la parte de procesado y análisis, mediante que las gráficas se han plasmado en la interfaz con la utilización de HTML, CSS y BootStrap.

Pero para tí, lector puede surgirte la pregunta de ¿Por qué? o ¿Cuál es la motivación de hacer esto? En las siguientes líneas espero poder hacerte entender mi motivación y entusiasmo por el mismo.

Actualmente los avances tecnológicos, permiten a la sociedad evolucionar a un ritmo muy elevado, hoy en día la sociedad está completamente conectada a la tecnología y estamos abiertos a cualquier cambio tecnológico por muy radical que sea, estamos en constante adaptación con

el mundo de las aplicaciones.

Pero detrás de todas estas aplicaciones tecnológicas, hay una serie de empresas o personas que trabajan muy duro, por el desarrollo de estas, y por encima de las personas están las empresas, que aunque esta sea una parte desconocida para el usuario final, la empresa se mueve por rentabilidad y la rentabilidad se convierte en dinero, para conseguir este objetivo se necesita tener una estadística de los factores que intervienen en un proyecto y realizar un seguimiento a una serie de parámetros, con el objetivo de obtener sus propias estadísticas como, rentabilidad de un proyecto, cambios realizados en eso proyecto, equipo de recursos humanos utilizados, la producción.

Este proyecto esta enfocado, a largo plazo, con el objetivo de conseguir una herramienta potente en un futuro, en la que puedan trabajar futuros compañeros mejorando e incluyendo nuevas funcionalidad, por el cual es muy importante conocer y entender que es el software libre y cuales son las bases principales para conseguir nuestro propósito.

1.1. Descripción del problema

El objeto de este proyecto es tratar de solucionar un problema que cada día tiene mayor importancia, como todos sabéis los proyectos en la actualidad son bastante complejos debido a la gran cantidad de datos que manejan así como a la cantidad de personas que están trabajando en el mismo. Con el fin de solucionar este problema, se elabora dicho proyecto, el cual se encarga de la creación de un dashboard, a partir de una serie de ficheros JSON ya dados, construyendo una interfaz de usuario rica y que dispone de un gran potencial de búsqueda de datos.

¿Pero porque esta aplicación? En la actualidad, estamos una sociedad en constante lucha por ver que empresa produce mayor productividad, cual tiene mayores ganancias. Pero en el mundo del software existen otros parámetros no menos importantes, que pueden proporcionar bastante información, como cambios en un repositorios, las líneas que se han añadido y un largo etc. Para evaluar todos estos parámetros, es importante tener una herramienta que nos abstraiga de consultar controladores de versión, herramientas como Sonar para gestion de código... Más que nada porque muchos de estos datos serán utilizados por gente que no es del mundo de la informática y posiblemente no sabrá utilizar ninguna de las herramientas anteriormente citadas.

Con este panorama actual cabe destacar que este proyecto puede considerarse ligado a uno

más grande que pretende potenciar una rama de la ingeniería del software: la ingeniería del software libre.

Ésta pretende aprovechar la existencia de una ingente cantidad de información accesible derivada de las formas de desarrollo abiertas (código fuente, comunicaciones entre desarrolladores, etc.) que llevan a cabo los proyectos del software libre, de manera que puedan ser cuantificados, medidos y estudiados. Los resultados y su consiguiente análisis ayudarán enormemente en la comprensión de los fenómenos asociados a la generación del software libre, al tiempo que facilitarán la toma de decisiones a partir de la experiencia adquirida.

Con el fin de crear unas herramientas esenciales que permitan disponer del análisis de la mayor cantidad de proyectos posibles, el Grupo de Sistemas y Comunicaciones de la universidad (GSyC) está interesado en este tipo de herramientas, en este contexto se están desarrollando dashboards y otras herramientas (como por ejemplo las que produce la información que consume el presente dashboard), en colaboración con otras organizaciones, como la empresa Bitergia.

1.2. Objetivo Principal

El objetivo principal de este proyecto es la construcción de una aplicación que funcione en un navegador web, y que permita visualizar los datos de desarrollo de software obtenidos a partir de los repositorios de desarrollo de un proyecto.

Dentro de ese gran objetivo principal existen otros subobjetivos:

- **Procesado de Ficheros:** El dashboard tiene que ser capaz de obtener y manejar ficheros JSON, que son ofrecidos por fuentes fidedignas, las cuales han producido esos ficheros mediante MetricsGrimoire.
- **Visualización de los Datos:** Tenemos que ser capaces de representar gráficamente los datos obtenidos de los ficheros, construyendo gráficas que permitan una visualización clara. Para ello se ha utilizado HighChart.js
- **Eventos:** Otro de los objetivos importantes del proyecto, los eventos consiste en programar acciones que ocurran cuando el usuario realice alguna acción, en nuestro caso será lo que nos permita ganar profundidad y navegar entre gráficas.

- **Interfaz de Usuario:** Contruir una interfaz de usuario agradable y sencilla de utilizar, para que el cliente pueda navegar y sentirse agusto, el objetivo es que el usuario no tenga que aprender a utilizarla, si no que sea intuitiva.
- **Abstraer al usuario:** El usuario final tiene que estar al margen del código o de rellenar parámetros, ya que posiblemente la herramienta no esté pensada para gente de ingeniería, si no que esta pensada para economistas, jefes de proyectos los cuales buscan una herramienta de análisis de datos sencilla y con una curva de aprendizaje mínima.

Hay que cumplir con todos estos objetivos anteriormente citados pero sin olvidar que nuestra aplicación debe ser dinámica y fluida adaptándonos a la demanda del usuario, ya que hoy en día el usuario pide que las transacciones y cambios de páginas sean prácticamente inmediatos, por eso la aplicación se basa en un Single Pages Application, con esto conseguimos que toda la lógica se realice en la parte del cliente, por lo que reducimos tráfico con el servidor y el servidor queda como un servicio para obtener los datos.

1.3. Requisitos

El proyecto deberá cumplir ciertos requisitos básicos:

- **Definir una interfaz simple e intuitiva para el usuario final:** Esto quiere decir que el usuario tiene que tener una vista completa pero sencilla, es decir muy intuitiva que le permita navegar con facilidad, sin necesidad de tener una curva de aprendizaje excesivamente pronunciada
- **Proceso automatizado al máximo:** El usuario tiene que estar al margen de como se hacen las cosas, para conseguir este requisito tenemos que ponernos en su lugar, ya que al consumidor solo le interesan los datos que pueda ofrecer la aplicación.
- **Fácil puesta en marcha y actualizaciones inapreciables para el usuario:** Las futuras actualizaciones tienen que ser transparentes, es decir cuando el desarrollador quiera actualizar la aplicación, no debe tener la necesidad de que el usuario realice ningún proceso.

- Tiempo de extracción de datos será crítico, es decir tiene que ser inapreciable para el usuario: El tiempo de manejo de los datos tiene que ser prácticamente inapreciable, por lo que necesitamos tener un código bien estructurado y con el mínimo coste computacional.
- Compatibilidad con Navegadores: Si queremos tener éxito con nuestra aplicación necesitamos que sea compatible con todos los navegadores del mercado, ya que si limitamos los navegadores, reducimos los usuarios a los que podemos llegar.
- Gran volumen de datos: Necesitamos que funcione con sistemas informáticos basados en la acumulación a gran escala de datos.

1.4. Modelo de desarrollo

Para el desarrollo de cualquier proyecto de software se realizan una serie de tareas entre la idea inicial y el producto final. Ese desarrollo sigue una determinada metodología modelo de desarrollo, el cual establece el orden en el que se llevan a cabo las tareas en el proyecto y nos provee de requisitos de entrada y salida para cada una de las actividades.

El desarrollo de este proyecto se ha basado en un modelo de adaptación de SCRUM.

SCRUM es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.

En nuestra adaptación se realizan entregas parciales y regulares de una implementación del producto final, priorizadas según las necesidades de cada entrega. Es por ello, que SCRUM está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales.

SCRUM también se utiliza para resolver situaciones en que no se está entregando al cliente lo que necesita, cuando las entregas se alargan demasiado, los costes se disparan o la calidad no es aceptable, cuando se necesita capacidad de reacción ante la competencia, cuando la moral de los equipos es baja y la rotación alta, cuando es necesario identificar y solucionar ineficiencias sistemáticamente o cuando se quiere trabajar utilizando un proceso especializado en el desarro-

llo de producto.

En SCRUM un proyecto se ejecuta en bloques temporales cortos y fijos (iteraciones de un mes natural y hasta de dos semanas, si así se necesita). Cada iteración tiene que proporcionar un resultado completo, un incremento de producto final que sea susceptible de ser entregado con el mínimo esfuerzo al cliente cuando lo solicite.

El proceso parte de la lista de objetivos/requisitos priorizada del producto, que actúa como plan del proyecto. En esta lista el cliente (en el ámbito del proyecto sería el tutor) prioriza los objetivos balanceando el valor que le aportan respecto a su coste y quedan repartidos en iteraciones y entregas. De manera regular el cliente puede maximizar la utilidad de lo que se desarrolla y el retorno de inversión mediante la replanificación de objetivos del producto, que realiza durante la iteración con vista a las siguientes iteraciones.

La creación de prototipos se ajusta perfectamente a las características de este proyecto, permite ir ajustandonos a situaciones reales, e ir comprobando su correcto funcionamiento. Estos prototipos han sido desarrollados, de forma incremental basándonos en el anterior (siempre que éste superase las pruebas), incorporando nuevas funcionalidades. Esto garantiza que las funciones básicas resulten extensamente probadas.

El conjunto de actividades en que se divide el modelo de desarrollo escogido hubo de ser adaptado a la dimensión del proyecto, dando lugar a las siguientes fases:

- **Comunicación con el Tutor.** El tutor especifica los objetivos del ciclo de la iteración. Se corresponde con una reunión donde el tutor del proyecto pone en conocimiento del autor del mismo los requisitos de cada ciclo. Unos requisitos funcionales y estructurales que, aunque generales, siempre precisos, dado que no tratamos con un cliente convencional, y estamos en un entorno universitario.
- **Planificación.** Se definen recursos, tiempo y otros parámetros relacionados con la presente iteración. De nuevo proporcionadas por el tutor a tenor del resultado de evaluar el prototipo correspondiente a la iteración previa.
- **Construcción y adaptación.** El sistema es desarrollado. Engloba tareas como el diseño y la implementación, ampliamente comentadas en el próximo capítulo.
- **Pruebas.** Se confirma que los requisitos se cumplen satisfactoriamente. Se remite al tutor tanto el prototipo como los resultados de las pruebas aplicadas sobre éste.

Cabe destacar que aunque los distintos analizadores de fuentes de información se presenten como desarrollados en paralelo, este hecho no se corresponde estrictamente con la realidad. Sin embargo, al haber evolucionado de modo similar, se prefiere no complicar innecesariamente la exposición.

Capítulo 2

Estado de Arte y Contexto

2.1. Software libre

Todo programa que sea considerado software libre debe ofrecer una serie de libertades. Se resumen en:

Libertad de usar el programa con cualquier fin, sin necesidad de comunicarlo a los desarrolladores;

Libertad de estudiar el código fuente del programa y modificarlo adaptándolo a nuestras necesidades, sin necesidad de hacer públicas las modificaciones;

Libertad de distribuir copias, tanto binarios como código fuente, modificadas o no, gratis o cobrando por su distribución;

Libertad de modificar el programa y publicar las mejoras para beneficio de la comunidad.

2.1.1. Ingeniería del Software Libre

El enfoque sistemático y cuantificable que propone la ingeniería del software siempre tuvo como barreras las propias de las formas en que el software ha sido desarrollado, publicado y distribuido. Aspectos como el formato binario, oscurantismo en modelo de negocio y limitaciones comerciales han impedido validar resultados por parte de equipos independientes.

El reciente auge del software libre aporta novedades a esta ingeniería del software. La implantación de Internet junto con las licencias que fomentan la colaboración en el desarrollo del software, han favorecido a que además del código fuente, se disponga de repositorios de versio-

nes donde observar la evolución del software o listas de correo que reflejan las comunicaciones durante el desarrollo. De estas fuentes puede obtenerse gran cantidad de datos de valor, incluso de forma automatizada.

Varios factores son los aportados a la ingeniería del software tradicional desde ingeniería del software libre:

- Visión temporal incorporada al análisis: necesaria ya que el proceso de creación cambia y su evolución analizada de forma continua proporciona información muy interesante (lenguajes más usados, evolución de colaboradores de un proyecto) destinada a servir de ayuda en la toma de decisiones.
- Análisis a gran escala: dada la inexistencia de impedimentos para ampliar el análisis al conjunto global de los proyectos de software libre gracias a la disponibilidad de la información generada durante su desarrollo. La ingeniería del software libre hace posible evaluar un proyecto dentro de entornos globales y de menor envergadura, ofreciendo información desde distintos puntos de vista, lo cual beneficia en la mencionada toma de decisiones.

En cierto modo, la ingeniería del software libre plantea cuantificar unos parámetros que nos permitan pronosticar con precisión costes, recursos y plazos.

En la actualidad el software libre carece de estos métodos, aunque la disponibilidad del código fuente y la información generada durante su desarrollo constituye un enorme potencial para que cambie esta situación. La ingeniería del software pretende también aplicar las cualidades de la ingeniería del software en el desarrollo de proyectos de software libre, de modo que se garantice a los desarrolladores la forma de generar software de calidad siguiendo los paradigmas adecuados. La ingeniería del software pretende aportar resultados objetivos y contrastables acerca de la evolución del software y desterrar así apreciaciones que algunos dan por ciertas. A corto plazo, la ingeniería del software libre tiene por objetivo realizar un análisis completo del desarrollo del software libre permitiendo indagar en los procesos que están involucrados, así como una adaptación de modelos de previsión de costes del estilo de COCOMO en el software propietario. Puede considerarse que el software libre funciona gracias a una “mano negra”, que hace que el software se genere mágicamente. Es por ello que la ingeniería del software libre busca comprender los elementos e interacciones que engloba esta laguna de conocimiento

denominada “mano negra”.

Se hace imprescindible un análisis de los datos relacionados con el software libre para poder alcanzar los objetivos, previamente descritos, que la ingeniería del software libre se propone. Debería procurarse que las herramientas empleadas para ello están disponibles para que grupos independientes puedan verificar los resultados. En este proceso de análisis pueden diferenciarse dos fases. En la primera etapa, un grupo de utilidades independientes entre se recogen datos cuantificables del código fuente y otros flujos de información y almacenan los resultados en un formato intermedio, que serán analizados en la siguiente fase. Lo ideal es que esta fase se realice de forma automática. La segunda fase, no tan madura como la anterior, integra programas que toman como entrada los parámetros almacenados en el formato intermedio y se dedican a su análisis, procesado e interpretación. Se han propuesto varios modos de analizar los resultados, de entre las que destacamos: herramientas de análisis de clústers, que a partir de porciones reducidas de datos, agrupan los interrelacionados con el objetivo de categorizarlos; herramientas de análisis estadístico, que simplifican el procesado de grandes cantidades de datos y permiten mostrar gráficamente los resultados; una interfaz web, cuyo fin es, además de proporcionar acceso a los resultados de este gran proyecto, la aplicación de los programas que generan esta interfaz a otros proyectos de software libre, de modo que surja una realimentación del proyecto global.

En cuanto a las fuentes a analizar, la que alberga mayor información en potencia es el código fuente. De él pueden extraerse parámetros como tamaño, número de líneas lógicas o físicas, número de desarrolladores, lenguaje de programación, etc. Uno de los estudios pioneros en este campo se encarga de calcular el número de líneas físicas de código de proyectos de software libre y aplicar el modelo COCOMO para obtener conclusiones en torno al coste, tiempo y recursos empleados en el desarrollo del software.

Otras fuentes de interés son aquellas donde se produce intercambio de información entre desarrolladores, como listas de correo o canales de IRC. De estos últimos aún no se han definido claramente los parámetros a buscar, mientras que de las listas interesa recuperar de cada mensaje del archivo: el nombre y dirección del autor, la fecha, e incluso podría cuantificarse la longitud del mensaje.

Existe otro tipo de fuentes de información compuesto por una serie de herramientas que

sincronizan el trabajo de los distintos desarrolladores de un software. Las más comunes son los sistemas de control de versiones, de los cuales obtener conclusiones acerca de la participación de cada desarrollador; y los sistemas de gestión de errores.

Una modalidad más de recopilación, quizás aplicable en un futuro, es la relacionada con la información personal de los desarrolladores, que a día de hoy no suele facilitarse, y que ayudaría a conocer con mayor profundidad la comunidad del software libre. Además, si se dispusiera de los datos laborales de estos desarrolladores —como proyectos en los que colaboran y horas empleadas— podría establecerse una previsión de costes y recursos para futuros proyectos de software libre.

2.2. Cloud Computing

El término cloud computing es un término bastante amplio y en algunas ocasiones confuso, pero está de moda porque suena bien y ahora mismo cualquier producto nuevo que salga al mercado suele llevar la coletilla "... cloud", se trate o no de un producto propiamente de cloud computing. Es por ello por lo que hay que explicar cuales son las características esenciales del cloud computing según la NIST:

- Servicio disponible de forma automática y a demanda: Un usuario puede comenzar a utilizar un recurso de cloud (almacenamiento, una instancia, etc.) sin ninguna intervención por parte de un operador de la empresa que presta el servicio.
- Acceso a través de la red: Los recursos están disponibles a través de la red (Internet o otro tipo de red pública o privada), mediante mecanismos estandarizados que permitan el uso de clientes diversos, desde teléfonos a grandes ordenadores.
- Los recursos se agrupan en pools en un modelo multi-tenancy. Los recursos son compartidos en general por múltiples clientes, que pueden disponer de ellos a demanda y a los que se les debe garantizar aislamiento y seguridad, a pesar de estar haciendo uso de recursos compartidos.
- Elasticidad: Es un concepto nuevo relacionado con el cloud y que lleva al extremo el concepto de escalabilidad, ya que los recursos del usuario del cloud pueden crecer y decrecer de forma rápida en función de sus necesidades.

Los servicios que se ofrezcan a los usuarios que cumplan estos requisitos se pueden considerar propiamente cloud computing y puesto que son requisitos bastante amplios, normalmente se clasifican en función del tipo de servicio que ofrezcan en tres capas: SaaS, PaaS e IaaS.

- **Software as a Service (SaaS):** Aplicación completa ofrecida como servicio en la nube. Es el ejemplo más conocido y usado de cloud computing en el que cualquier usuario hace uso de un determinado software de una empresa a través de Internet, como por ejemplo los servicios de Google, microsoft Office 365 y un larguísimo etcétera.
- **Platform as a Service (PaaS):** Aplicación completa para el desarrollo y despliegue de software. Los desarrolladores de software pueden optar por utilizar una plataforma en la nube para sus desarrollos, facilitándoles mucho la tareas de pruebas y despliegue. Algunas de las opciones más conocidas de PaaS son Google App Engine, Windows Azure, Red Hat OpenShift o Heroku. Obviamente esta capa de cloud es sólo para desarrolladores o empresas que se dedican al desarrollo, pudiendo ser obviada por el resto de usuarios.
- **Infrastructure as a Service (IaaS):** Principalmente almacenamiento y capacidades de cómputo (máquinas virtuales) ofrecidos como servicio en la nube. Los administradores de sistemas de todo el mundo pueden hoy en día plantearse montar un servicio en una máquina física, una máquina virtual o una instancia en el cloud, ofreciendo esta última unas opciones muy interesantes sobre todo para despliegues de demanda variable. Los servicios de cloud IaaS más conocidos son los de Amazon Web Services, RackSpace Cloud, Joyent o Windows Azure.

2.3. Openstack

OpenStack es una plataforma cloud computing de software libre que en tan solo tres años de desarrollo se ha convertido en una de las principales opciones para implementar un cloud de IaaS público o privado.

OpenStack es una colección de tecnologías Open Source que proporcionan un software para el despliegue escalable de un cloud computing. OpenStack proporciona Infraestructura como Servicio o IaaS (Infrastructure as a Service) y es un proyecto que se inició en el año 2010 por la empresa Rackspace Cloud y por la agencia espacial norteamericana, NASA. Actualmente más

de 150 empresas se han unido al proyecto, entre las que se encuentran empresas tan importantes como AMD, Intel, Canonical, SUSE Linux, Red Hat, IBM, Dell, HP, Cisco, etc. OpenStack es software libre bajo los términos de la licencia Apache.

Actualmente OpenStack desarrolla dos proyectos relacionados: OpenStack Compute, que proporciona recursos computacionales a través de máquinas virtuales y gestión de la red, y OpenStack Object Storage, que proporciona un servicio de almacenamiento de objetos redundante y escalable. Muy relacionado con el proyecto OpenStack Compute, existen otros proyectos complementarios como Keystone o Glance.

OpenStack puede ser utilizado por cualquiera organización que busque desplegar un cloud de gran escala tanto para uso privado como público. OpenStack es un proyecto interesante casi para cualquier tipo de organización: pequeñas y medianas empresas, administración, grandes corporaciones, proveedores de servicio, empresas de valor añadido, centros de cálculo y un largo etcétera.

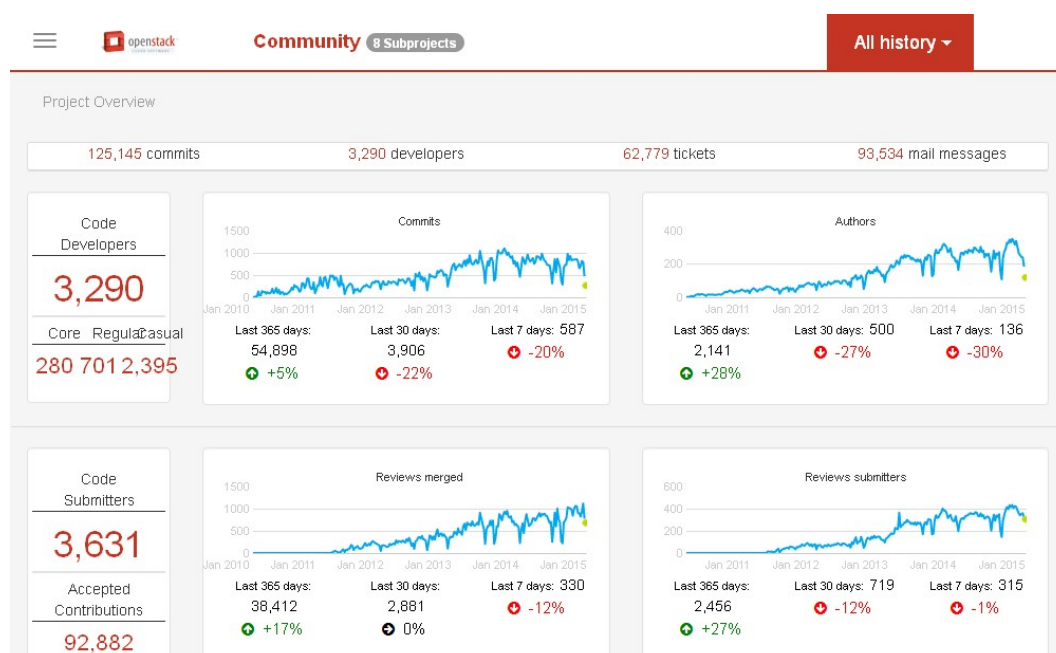


Figura 2.1: Apariencia de OpenStack

2.4. Métricas de Grimoire

Metrics Grimoire es un conjunto de herramientas para obtener datos de los repositorios relacionados con el desarrollo de software : gestión de código fuente (también conocido como

control de versiones) , sistemas de seguimiento de problemas (también conocido como errores de informes) , listas de correo , etc. Los datos y metadatos sobre el desarrollo de software procesos se recuperan de los repositorios (información sobre las confirmaciones, gestión de entradas , la comunicación en listas de correo , etc.) , y luego se organizaron y se almacenan en bases de datos SQL que luego pueden ser extraídos por los patrones de actividad específicos.

Las herramientas MetricsGrimoire apoyan muchos tipos de depósitos , incluidas las previstas en GitHub (git y seguimiento de problemas GitHub).

MetricsGrimoire ya se ha utilizado para analizar muchos proyectos diferentes y, junto con herramientas de visualización como VizGrimoire es posible conseguir cuadros de mando o informes , como los proporcionados por Bitergia.

2.5. VizGrimoire

VizGrimoire es un conjunto de herramientas cuyo objetivo es analizar y visualizar datos sobre el desarrollo de software . Actualmente , se centra en los datos producidos por las herramientas MetricsGrimoire (CVSanalY , Bicho y MailingListStats) .

VizGrimoire es promovido por Bitergia , la empresa que presta servicios de análisis development software, pero se trata de un proyecto abierto a la comunidad.

La siguiente es una lista de la actual gama de instrumentos con una breve descripción . Para obtener más información, visite cada uno de los repositorios de herramientas, para que se mas fácil tener una vision de la arquitectura vizGrimoire.

- VizGrimoireR: Analiza los datos en las bases de datos producidas por herramientas MetricsGrimoire desde los repositorios de desarrollo de software (Sistemas de gestión de código fuente , los sistemas de seguimiento de problemas , listas de correo , etc.) .
- VizGrimoireJS: Desarrollado en un entorno de JavaScript para crear cuadros de mando e informes interactivos y visualizaciones de datos producidos por vizGrimoireJS.

2.6. GitHub

GitHub es una plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones Git. Además de eso, puedes contribuir a mejorar el

software de los demás. Para poder alcanzar esta meta, GitHub provee de funcionalidades para hacer un fork y solicitar pulls. Realizar un fork es simplemente clonar un repositorio ajeno (genera una copia en tu cuenta), para eliminar algún bug o modificar cosas de él. Una vez realizadas tus modificaciones puedes enviar un pull al dueño del proyecto. Éste podrá analizar los cambios que has realizado fácilmente, y si considera interesante tu contribución, adjuntarlo con el repositorio original.

En la actualidad, GitHub es mucho más que un servicio de alojamiento de código. Además de éste, se ofrecen varias herramientas útiles para el trabajo en equipo. Entre ellas, caben destacar:

- Una wiki para el mantenimiento de las distintas versiones de las páginas.
- Un sistema de seguimiento de problemas que permiten a los miembros de tu equipo detallar un problema con tu software o una sugerencia que deseen hacer.
- Una herramienta de revisión de código, donde se pueden añadir anotaciones en cualquier punto de un fichero y debatir sobre determinados cambios realizados en un commit específico.
- Un visor de ramas donde se pueden comparar los progresos realizados en las distintas ramas de nuestro repositorio.

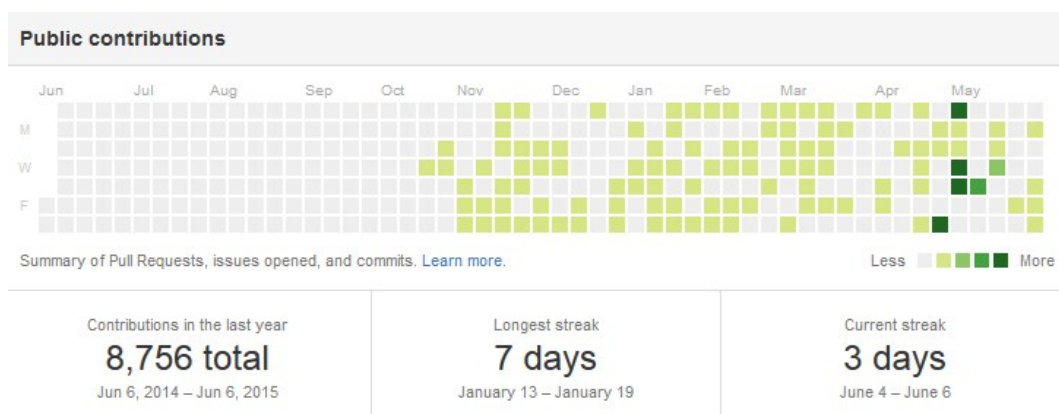


Figura 2.2: Estadísticas ofrecidas por GitHub

2.7. Bitergia

Bitergia, toma como base de su negocio los datos ofrecidos por las herramientas de desarrollo, ha producido una serie de herramientas que permiten llevar a cabo la extracción y posterior análisis de dicha información. El objetivo es que los clientes puedan tomar decisiones basadas en datos objetivos y no en percepciones subjetivas del proyecto.

La empresa Bitergia está comercializando los servicios basados en Grimoire, como explicaremos en la siguiente sección de este capítulo, pero que una primera definición, Grimoire son las métricas.



Figura 2.3: Apariencia de Bitergia

Bitergia, una compañía apoyada por el Vivero de Empresas del Parque Científico de la Universidad Carlos III de Madrid (UC3M), y formada por miembros del grupo de LibreSoft de la Universidad Rey Juan Carlos, que investiga cómo innovar en el análisis cuantitativo en profundidad de proyectos de software libre, en los que todo el proceso de desarrollo se realiza en herramientas que transmiten información accesible a cualquier persona interesada.

2.8. OpenHub

Open Hub es un sitio web que nos proporciona información muy útil sobre cualquier proyecto de código abierto. El sitio proporciona estadísticas acerca de la longevidad de los proyectos, sus licencias y las cifras de software (como líneas de código fuente y las estadísticas de los Commit, etc). También podemos encontrar en que lenguaje fue escrito, proyectos similares, y una infinidad de información extra.

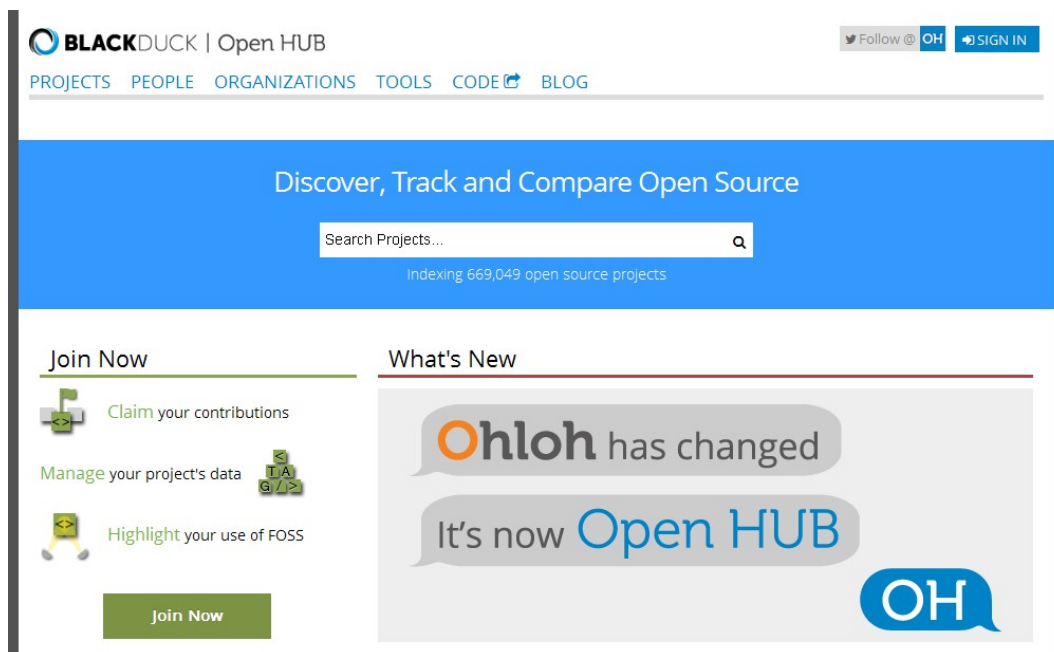


Figura 2.4: Apariencia de Open Hub

A continuación se muestra una captura de pantalla del dashboard que es ofrecido por Open-Hub

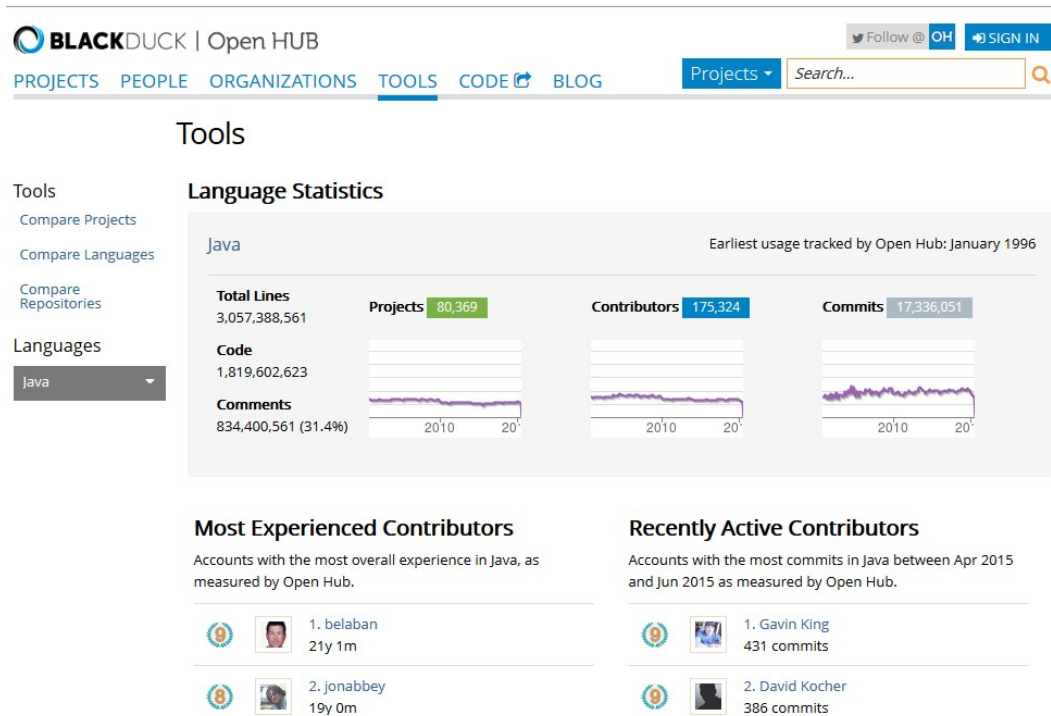


Figura 2.5: Dashboard de OpenHub

Capítulo 3

Tecnologías Utilizadas

Antes de abordar este proyecto, se necesita conocer que fuentes de estudio vamos a utilizar así como conocer los lenguajes de programación que vamos a emplear.

3.1. Bibliotecas de Representación Gráfica

En la actualidad hay numerosas bibliotecas en javascript que permiten la representación de datos, pero a la hora de elegir una hay que tener muy claro que nos permite y cuales son las ventajas frente a las otras que disponemos. En una primera fase estuve trabajando con: Chart.js, Jslate, Dashing, Dc, Freeboard. Pero finalmente me decante por utilizar HighChart

3.1.1. HighChart.js

Es una biblioteca, desarrollada en javascript, la cual permite una gran variedad de representaciones gráficas. Bajo mi punto de vista destacaría, la posibilidad de disparar eventos, esto permite que podamos concatenar gráficas sin la necesidad de ir navegando a través de un menú, además que nos permite profundizar en las gráficas.

Además también permite representaciones en 3D e incluso charts que se van modificando con el paso del tiempo. Como nota cabe comentar que esta biblioteca también tiene representaciones para mapas y stocks.

Con respecto al paso de parámetros, es una biblioteca la cual es muy específica y poco flexible ya que cada representación tiene unos parámetros específicos, que la hacen que sea

muy estricta. Como ventaja podemos destacar que la estructura de los parámetros hace que sea muy sencilla identificarlo y permite modificar los valores rápidamente, de cara al desarrollador es muy amena de trabajar con dicha biblioteca.

También destacaría que tiene una documentación muy bien estructurada y que permite al programador una rápida identificación de los problemas.

3.1.2. Chart.js

Charts.js es una librería con la que puedes crear gráficos simples, limpios y atractivos, es usada tanto por diseñadores y desarrolladores. Entre sus características podemos mencionar que posee 6 tipos de gráficos no tiene dependencias y es super ligero. Utiliza el elemento canvas de HTML5 y está disponible en todos los navegadores modernos, incluso brinda soporte a Internet Explorer 7 y 8. Chart.js tiene diseño adaptativo, es modular e interactivo. Para más información puedes revisar la documentación completa en su sitio. Es un proyecto Open Source, puedes ver su código fuente en su repositorio de GitHub.

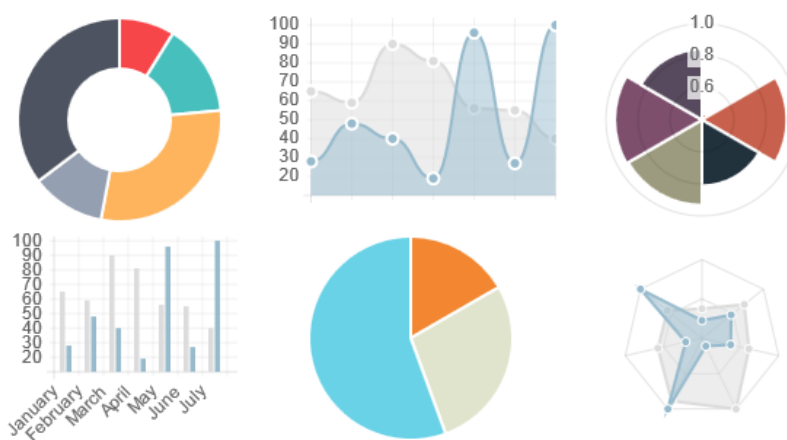


Figura 3.1: Charts ofrecidos por la biblioteca Chart.js

3.2. JavaScript

El lenguaje JavaScript es un lenguaje de programación que surgió por la necesidad de ampliar las posibilidades del HTML. En efecto, al poco tiempo de que las páginas web aparecieran, se hizo patente que se necesitaba algo más que las limitadas prestaciones del lenguaje básico,

ya que el HTML solamente provee de elementos que actúan exclusivamente sobre el texto y su estilo, pero no permite, como ejemplo sencillo, ni siquiera abrir una nueva ventana o emitir un mensaje de aviso. La temprana aparición de este lenguaje, es posiblemente la causa de que se haya convertido en un estándar soportado por todos los navegadores actuales, a diferencia de otros, que solo funcionan en los navegadores de sus firmas creadoras.

Además es muy importante no confundir Java con JavaScript ya que son cosas totalmente distintas y que no tiene nada que ver porque, Java es un lenguaje compilado, es decir, que una vez escrito el programa, y a partir de su código fuente, mediante la compilación se genera un fichero ejecutable para una determinada plataforma (Unix, Windows, etc.) que será completamente autónomo. Es un lenguaje de propósito general, infinitamente más potente que JavaScript, con el que se han escrito infinidad de aplicaciones muy conocidas, entre ellas los sistemas de telefonía móvil.

JavaScript es un lenguaje interpretado línea a línea por el navegador, mientras se carga la página, que solamente es capaz de realizar las acciones programadas en el entorno de esa página HTML donde reside. Sólo es posible utilizarlo con otro programa que sea capaz de interpretarlo, como los navegadores web.

Este es un lenguaje orientado a objetos, es decir que la mayoría de las instrucciones que se emplean en los programas, en realidad son llamadas a propiedades y métodos de objetos del navegador, y en algunos casos del propio lenguaje. En Java, en cambio, no hay nada que no esté en un objeto.

¿Pero que ventajas y desventajas tiene utilizar este tipo de lenguaje de scripting?...Intentemos resumir algunas:

- El lenguaje de scripting es seguro y fiable porque está en claro y hay que interpretarlo, por lo que puede ser filtrado; para el mismo Javascript, la seguridad es casi total y sólo en su primera versión el CIAC (Computer Incident Advisory Committee) señaló problemas de leve entidad, entre ellos la lectura de la caché y de los sitios visitados, de la dirección e-mail y de los file presentes en el disco. Sin embargo, estos fallos se corrigieron ya en las versiones de Netscape sucesivas a la 2.0;
- Los script tienen capacidades limitadas, por razones de seguridad, por lo cual no es posible hacer todo con Javascript, sino que es necesario usarlo conjuntamente con otros len-

guajes evolucionados, posiblemente más seguros, como Java. Dicha limitación es aún más evidente si queremos operar en el hardware del ordenador, como, por ejemplo, la fijación en automático de la resolución vídeo o la impresión de un documento;

- Un problema importante es la seguridad y es que como todo sabeis el código es visible y puede ser leído por cualquiera, incluso si está protegido con las leyes del copyright.
- El código Javascript se ejecuta en el cliente por lo que el servidor no es solicitado más de lo debido; un script ejecutado en el servidor, podría conducir a varios problemas de capacidad y gestión.
- El código del script debe descargarse completamente antes de poderse ejecutar y ésta es la otra cara de la moneda de lo que hemos dicho anteriormente: si los datos de un script son muy elevados, el tiempo que tardará en descargarse será muy elevado, aunque esta hoy en día es prácticamente inapreciable debido a los anchos de banda disponibles.

3.3. HTTP

El Protocolo de Transferencia de Hipertexto (Hypertext Transfer Protocol) es un sencillo protocolo cliente-servidor que articula los intercambios de información entre los clientes Web y los servidores HTTP. La especificación completa del protocolo HTTP 1/0 está recogida en el RFC 1945.

Desde el punto de vista de las comunicaciones, está soportado sobre los servicios de conexión TCP/IP, y funciona de la misma forma que el resto de los servicios comunes de los entornos UNIX: un proceso servidor escucha en un puerto de comunicaciones TCP (por defecto, el 80), y espera las solicitudes de conexión de los clientes Web. Una vez que se establece la conexión, el protocolo TCP se encarga de mantener la comunicación y garantizar un intercambio de datos libre de errores.

HTTP se basa en sencillas operaciones de solicitud/respuesta. Un cliente establece una conexión con un servidor y envía un mensaje con los datos de la solicitud. El servidor responde con un mensaje similar, que contiene el estado de la operación y su posible resultado. Todas las operaciones pueden adjuntar un objeto o recurso sobre el que actúan; cada objeto Web (documento HTML, fichero multimedia o aplicación CGI) es conocido por su URL.

Para profundizar más en el funcionamiento de HTTP, veremos primero un caso particular de una transacción HTTP; en los siguientes apartados se analizarán las diferentes partes de este proceso.

Cada vez que un cliente realiza una petición a un servidor, se ejecutan los siguientes pasos:

- Un usuario accede a una URL, seleccionando un enlace de un documento HTML o introduciéndola directamente en el campo Location del cliente Web.
- El cliente Web descodifica la URL, separando sus diferentes partes. Así identifica el protocolo de acceso, la dirección DNS o IP del servidor, el posible puerto opcional (el valor por defecto es 80) y el objeto requerido del servidor.
- Se abre una conexión TCP/IP con el servidor, llamando al puerto TCP correspondiente. Se realiza la petición. Para ello, se envía el comando necesario (GET, POST, HEAD), la dirección del objeto requerido (el contenido de la URL que sigue a la dirección del servidor), la versión del protocolo HTTP empleada (casi siempre HTTP/1.0) y un conjunto variable de información, que incluye datos sobre las capacidades del browser, datos opcionales para el servidor.
- El servidor devuelve la respuesta al cliente. Consiste en un código de estado y el tipo de dato MIME de la información de retorno, seguido de la propia información.
- Se cierra la conexión TCP.

3.4. HTML5

Son las siglas de HyperText Markup Language, hace referencia al lenguaje de marcado para la elaboración de páginas web. Es un estándar que sirve de referencia para la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web, como texto, imágenes, videos, entre otros. Es un estándar a cargo de la W3C, organización dedicada a la estandarización de casi todas las tecnologías ligadas a la web, sobre todo en lo referente a su escritura e interpretación.

El número 5 hace referencia a la quinta revisión del lenguaje, esta es la primera vez que HTML y XHTML se han desarrollado en paralelo.

- **Sintaxis.-** La sintaxis de HTML 5 es compatible con HTML 4 y XHTML 1. Con HTML 5 se puede usar sintaxis de XML. Con HTML 5 se puede usar sintaxis de MathML, que es un lenguaje usando para describir notaciones matemáticas, permitiendo la integración de fórmulas matemáticas a Internet.
- **Nuevos elementos y nuevos atributos.-** Además de agregar elementos nuevos al lenguaje, a algunos elementos existentes se le agregan atributos, y otros elementos cambian. Para información detallada de estas diferencias, puedes consultar la página oficial de WC3 que describe las diferencias en lenguaje entre HTML 4 y HTML 5.
- **Modelo del contenido.-** HTML 5 busca reducir la confusión en el concepto de bloques existente entre HTML 4 y CSS. Con este fin, define nuevas categorías y define qué elementos son parte de cada una. Para mayor información, puedes consultar la sección de cambios al modelo del contenido, descrito en la página oficial de WC3.
- **APIs.-** HTML5 agregó nuevas APIs (un API son las siglas en inglés de Application Programming Interface, que es una biblioteca de utilerías que puede ser utilizada por algún software, en este caso aplicaciones de Internet), además de que extendió, cambió o hizo obsoletas algunas de las ya existentes software

3.5. CSS

El lenguaje CSS son las siglas de Cascading Style Sheets, en español Hojas de estilo en Cascada. Es una tecnología que nos permite crear páginas web de una manera más exacta. Gracias al CSS somos mucho más dueños de los resultados finales de la página, pudiendo hacer muchas cosas que no se podía hacer utilizando solamente HTML.

La filosofía de CSS se basa en intentar separar lo que es la estructura del documento HTML de su presentación. Por decirlo de alguna manera: la página web sería lo que hay debajo (el contenido) y CSS sería un cristal de color que hace que el contenido se vea de una forma u otra. Usando esta filosofía, resulta muy fácil cambiarle el aspecto a una página web: basta con cambiar el cristal que tiene delante.

Como HTML, CSS también ha tenido diferentes versiones a lo largo del tiempo, en la tabla adjunta se pueden encontrar los años de publicación

Versión	Año de Lanzamiento
CSS 1	Publicada en 1996.
CSS 2	Publicada en 1998.
CSS 2.1	Publicada en 2004.
CSS 3	Publicada en 2011.
CSS 4	Se Espera la publicación en 2019.

Las ventajas principales en esta nueva versión CSS3 son la inclusión de nuevas propiedades especialmente en cuanto al aspecto gráfico. La actualización ha incluido los tan reclamados bordes redondeados, textos con sombras, la capacidad de asignar múltiples fondos, un mejor manejo de tablas incluyendo el estilo zebra, multi-columnas, etc. El modelo conserva muchas de las actuales propiedades y trabajar con nuevos selectores.

Si desglosamos un poquito más cada campo obtenemos lo siguiente:

- Capacidades visuales avanzadas: CSS 3 contiene varias mejoras en cuanto a interfaz gráfica, posicionamiento y tamaño de los objetos, usando condiciones de alineación para cada uno. El objetivo es que sea más sencillo posicionar los controles dentro de la página y que cuenten con otras características como desplazamiento.
- Hojas de Estilo Aural: Pretende utilizar las opciones de ciertos dispositivos con capacidades de reproducción de sonido. El módulo de audio podría agregar sonidos de fondo o efectos de transición que se activarían mediante determinado evento. Otras propiedades permitan controlar la posición del sonido que se está reproduciendo, etc.
- Bordes y Fondos: Las nuevas capacidades de CSS 3 permitirían entre otras cosas usar imágenes para los bordes, redondear y/o agregar sombras. Posicionamiento de elementos en pantalla: se podrá controlar de mejor manera los objetos y su dirección (horizontal o vertical). Además, se quiere incluir el módulo de paginación para crear pies de página, referencias cruzadas y construir cabeceras para títulos de secciones. También se desea introducir una nueva propiedad para dividir secciones en columnas.

- Fuentes: Nuevas funciones sobre todo encaminadas a brindar un mejor soporte a múltiples lenguajes. Se pretende incluir el @font-face para utilizar fuentes externas. Otros cambios sustanciales serían un mejor modelo para trabajar CSS con DOM incluyendo la posibilidad de cambiar valores a las propiedades, uso de NAMESPACES como XML y la inclusión de un mejor soporte para manejo de expresiones matemáticas.

3.6. BootStrap

Bootstrap es un framework Front End muy utilizado y extendido entre diseñadores y desarrolladores. Fue creado por dos empleados de Twitter como un sistema de software libre. Existe una comunidad muy activa que amplía constantemente las capacidades de este framework. Es un framework completo, es decir, que tiene la rejilla de estilos CSS, componentes (html+css) y añade además una capa de JAVA para elementos de interacción. Bootstrap 3 se construyó basado y pensado para la web móvil “mobile-first approach” o enfoque móvil de primera, donde se prioriza la web moderna, la web que se puede ver correctamente en cualquier dispositivo, desde pantallas pequeñas con poca resolución hasta dispositivos con resoluciones grandes y pantallas de gran tamaño.

Pero como toda tecnología, siempre tiene sus ventajas y sus desventajas, las cuales el usuario debe conocer para poder desarrollar correctamente. Algunas de las mas importantes son:

Ventajas

- Utiliza componentes y servicios creados por la comunidad web, tales como: HTML5 shim, Normalize.css, OOCSS, jQuery UI, LESS y GitHub.
- Es un conjunto de buenas prácticas que perduran en el tiempo.
- El famoso Grid system, que por defecto incluye 12 columnas fijas o fluidas, dependiendo de si tu diseño será responsivo o no.
- El uso de LESS, que es una ampliación a las famosas hojas de estilo CSS, pero a diferencia de éstas, funciona como un lenguaje de programación, permitiendo el uso de variables, funciones, operaciones aritméticas, entre otras, para acelerar y enriquecer los estilos en un sitio web.

- OOCSS, css orientado a objetos, que está organizado por módulos independientes y re-utilizables en todo el proyecto.
- Hay una enorme comunidad que soporta este desarrollo y cuenta con implementaciones externas como WordPress, Drupal, SASS o jQuery UI.
- Herramienta sencilla y ágil para construir sitios web e interfaces. Una vez se entiende y dominas su funcionamiento es muy fácil, hacer efectos y diseñar interfaces que te ahorran realmente mucho tiempo de trabajo.
- Incluye numerosos themes que son editables y personalizables que permiten crear una apariencia muy visible.

Desventajas

- Es necesario adaptarse a su forma de trabajo, si bien su curva de aprendizaje es liviana, deberás comprender y familiarizarte con su estructura y nomenclatura.
- Debes adaptar tu diseño a un grid de 12 columnas, que se modifican según el dispositivo.
- Bootstrap por defecto te trae anchos, márgenes y altos de línea, y realizar cambios específicos es por decir, un poco tedioso.
- Es complicado, cambiar de versión si has realizado modificaciones profundas sobre el core.
- Si necesitas añadir componentes que no existen, debes hacerlos tú mismo en CSS y cuidar de que mantenga coherencia con tu diseño y cuidando el responsive.

Estas son las ventajas y desventajas mas importantes , y bajo mi punta las mas importante que un desarrollador tener que conocer a la hora de implementar una aplicación

3.7. jQuery

jQuery es una biblioteca gratuita de Javascript, cuyo objetivo principal es simplificar las tareas de creación de páginas web responsivas, acordes a lo estipulado en el Web 2.0, la cual

funciona en todos los navegadores modernos. Por otro lado, se dice que jQuery ayuda a que nos concentremos de gran manera en el diseño del sitio, al abstraer por completo todas las características específicas de cada uno de los navegadores. Otra de las grandes ventajas de jQuery es que se enfoca en simplificar los scripts y en acceder/modificar el contenido de una página web.

Ventajas de jQuery con respecto a otras alternativas

Es importante comentar que jQuery no es el único framework que existe en el mercado. Existen varias soluciones similares que también funcionan muy bien, que básicamente nos sirven para hacer lo mismo. Como es normal, cada uno de los frameworks tiene sus ventajas e inconvenientes, pero jQuery es un producto con una aceptación por parte de los programadores muy buena y un grado de penetración en el mercado muy amplio, lo que hace suponer que es una de las mejores opciones.

Además, es un producto serio, estable, bien documentado y con un gran equipo de desarrolladores a cargo de la mejora y actualización del framework. Otra cosa muy interesante es la dilatada comunidad de creadores de plugins o componentes, lo que hace fácil encontrar soluciones ya creadas en jQuery para implementar asuntos como interfaces de usuario, galerías, votaciones, efectos diversos, etc.

Uno de los competidores de jQuery, es Mootools, que también posee ventajas similares. Os dejo el enlace al Manual de Mootools, que también puede ser interesante, porque seguramente lo tengamos explicado con mayor detalle que jQuery.

3.8. jQuery Ui

Es una biblioteca que nos permite introducir mayor funcionalidad a nuestra página web. Dicha biblioteca es jquery ui, es una biblioteca de componentes para el framework jQuery que le añaden un conjunto de plugins, widgets y efectos visuales para la creación de aplicaciones web. Entre otras cosas nos aplican a botones, pestañas, calendarios, campos de texto con auto-completado y barras de deslizamiento.

Capítulo 4

Proceso de Desarrollo

En éste capítulo, se procede a explicar las diferentes etapas de desarrollo, para que se pueda entender toda la evolución del proyecto.

Dentro de las fases de trabajo, se puede dividir en las siguientes secciones:

- **Iteración 0.** Fase inicial que consiste en el estudio de la estructura de los diferentes documentos a *parsear*, así como las herramientas que disponemos para elaborarlo. En realidad se trata de un paso anterior al desarrollo y, por tanto, no implica un avance en el proceso ni desemboca en un prototipo, pero la vital importancia de esta labor nos obliga a dedicarle un lugar destacado en la evolución del proyecto.
- **Iteración 1.** El objetivo de esta etapa es la creación del sistema de *Obtención de Fichero* básico que establecerá la base para cualquier otro fichero, es decir, esta fase bien planteada, en un futuro no debería llevarnos a modificar dicha iteración.
- **Iteración 2.** Una segunda fase que pretende extender la funcionalidad de *HighChart*, para poder conseguir nuestro objetivo final de un dashboard con eventos, con dichos eventos conseguimos que el usuario escarbe y navegue a través de las gráficas.
- **Iteración 3.** En esta fase esta orientada al aspecto visual donde introduciremos a nuestro dashboard un aspecto mucho más elegante para conseguir llamar la atención del usuario, con la premisa de crear una interfaz sencilla y fácil.
- **Iteración 4.** Una última fase dedica a la instalación de la aplicación en un servidor real, para que pueda estar accesible a cualquier persona que disponga de conexión a internet.

4.1. Iteración 0. Estudio previo

En esta primera iteración, nos encargamos de estudiar y realizar las primeras pruebas con las bibliotecas que se nos han propuesto, el objetivo de esta fase es conocer nuestras herramientas de trabajo, decidir una herramienta la cual nos permita manejar mejor los datos, para conseguir nuestro objetivo.

4.1.1. Bibliotecas Manejadas

Como se comentó en el apartado anterior, he manejado y utilizado varias bibliotecas hasta que he encontrado la que más se adapta a mis necesidades. Durante este período de prueba se han realizado numerosas pruebas, bien de representación, parámetros, eventos... Las características que buscábamos para nuestro proyecto son las siguientes:

- **Parámetros:** Es muy importante que la biblioteca tenga un fácil acceso a los parámetros para poder modificarlos adecuándonos a nuestras necesidades. Cuando hablamos de parámetros de un gráfico, nos referimos a: etiquetas, leyendas, eventos...
- **Paso de Datos:** Sin duda el más importante ya que una biblioteca poco flexible puede complicar bastante el trabajo, debido a que vamos a tener que parsear los datos antes de realizar cualquier representación, para adaptarlos al formato que admite la biblioteca.
- **Eventos:** Dada la necesidad de este proyecto, necesitamos una biblioteca que nos permita disparar eventos, ya que en nuestro dashboard necesitamos que se realicen acciones cuando pinchamos en las gráficas.
- **Documentación:** Es muy importante tener una buena documentación y API, que permita al usuario tener acceso rápido y preciso a lo que ofrece la biblioteca, el objetivo de la documentación es suavizar la curva de aprendizaje.

A continuación, adjunto una serie de gráficas donde se pueden ver las diferentes bibliotecas que he utilizado, finalmente esa biblioteca que se ha elegido es HighChart.

4.1.2. Ejercicios JavaScript

Debido a que era un lenguaje nuevo para mí, he tenido que realizar un esfuerzo extra, en este esfuerzo incluye leer documentación, libros propuestos por el tutor y realizar unos ejercicios previos para poder conocer la sintaxis del lenguaje.

Siguiendo el consejo de mi tutor, el me recomendó utilizar una página para realizar una serie de ejercicios, dicha página es CodeAcademy, es recurso online donde puedes realizar ejercicios por capítulos incrementando la dificultad y con la ventaja de comprobar si lo has hecho bien automáticamente, ya que dispone de un corrector.

Además para futuros alumnos que quieran iniciarse en este lenguaje, les animo y les digo que es un lenguaje bastante fácil y de comprensión rápida donde se pueden ver resultados muy buenos en poco tiempo, solo con un poco de esfuerzo y una buena base de programación en cualquier otro lenguaje, permiten desarrollar códigos los cuales implementan bastante funcionalidad.

4.2. Iteración 1

Tras la exploración de las bibliotecas, el siguiente paso que se nos plantea es ¿Cómo obtenemos los datos?, Esta carga de datos debe hacerse de forma rápida y dinámica, el objetivo de esta iteración es conseguir los ficheros sin bloquear el resto de la aplicación.

4.2.1. Obteniendo JSON

En nuestra aplicación, va a tener mucha importancia, como obtenemos el fichero, ya que al cargar numerosos ficheros json, necesitamos que sean llamadas no bloqueantes, es decir que sean asíncronas, para poder dar un aspecto dinámico a nuestro dashboard. Esto quiere decir que el código no tiene que ser secuencial, podrá salir antes unas gráficas que otras sin importar el orden de las llamadas en el código. La primera función que se pensó en utilizar para implementar esto fue la siguiente:

```
\$.ajax({  
  url: 'MyArray.json',  
  async: false,  
  dataType: 'json',  
  success: function (response) {
```

```

        //do stuff with response.
    }
})

```

El inconveniente de esta función es el parámetro `async`, como podemos ver esta a `false`, esto lo que hace es bloquear todo el código, ya que esta guardando la sincronización por lo tanto hasta que no obtengamos un fichero completo no va a continuar con el otro. Esto empezó a notarse cuando cargamos cuatro o más ficheros donde la aplicación se quedaba en stand-by esperando a que terminara, de cara al usuario final ofrece una impresión de que no funciona correctamente.

Investigando un poco y comentando el problema con el tutor, decidimos decantarnos por las llamadas asíncronas. Al ser asíncronas se ejecutan en paralelo, por lo que necesitamos una herramienta que nos avise cuando termine, esto se denomina eventos, que lo comentaremos en la siguiente sección.

Por lo tanto la función resultante para la carga de fichero y que utilizaremos en todas las cargas será la siguiente, comentar que tiene incorporados eventos, pero ahora lo importante es entender el funcionamiento de la parte de obtención del fichero.

Función:

```

function cargarfichero(nombrejson,eventtrigger){
    \$.getJSON("json/"+nombrejson).success(function(data) {
        //actualizo mi estado a terminado
        \$("#*").trigger(eventtrigger,[eventtrigger,data])
    }).error(function(){
        //En caso de error levanto el evento de gráfica mal pintada
        alert("No se ha podido cargar el fichero: "+nombrejson)
    });
}

```

En esta función tenemos dentro una llamada a `getJSON`, esto es una función de jQuery que lo que esta realizando internamente es lo siguiente:

```

\$.ajax({
    dataType: "json",
    url: url,
    data: data,
    success: success
});

```

Como se puede apreciar el parecido con la primera función que utilizamos es muy alto, aunque si nos fijamos en los parámetros que tenemos son los mismos menos el parámetro `async`

que aquí no aparece porque por defecto esta puesto a true, esto quiere decir que esta función es asíncrona por defecto.

El motivo de usar `getJSON` en vez de poner la primera función con `async` igual `true`, es por reglas de calidad de código y porque ya que es una función integrada no hay necesidad de repetir código. Para terminar bien de entender nuestra función adjunto una tabla con los parámetros que necesitamos para que funcione correctamente nuestra función: `cargaFichero`.

Parámetros	Descripción
<code>nombreJson</code>	Este parámetro es requerido, y es el nombre del fichero json.
<code>eventTrigger</code>	Nombre del evento que vamos a disparar cuando haya terminado la carga

4.2.2. Eventos Trigger

Como comentábamos anteriormente, necesitamos recuperar el hilo que esta en paralelo encargado de la carga de fichero, para ello disponemos de los eventos, que tienen como objetivo avisar cuando terminan la tarea asignada, levantado un flag, el cual estan escuchando los manejadores. Esto permite recuperar el hilo de ejecución principal.

En jQuery existen dos formas de llamar a eventos, `trigger` y `triggerHandler`, pero entre ellos dos existen algunas diferencias.

- La función `triggerHandler()` no ejecuta el evento predeterminado del elemento (por ejemplo en el caso de un formulario, `triggerHandler` no enviaría el formulario).
- Mientras `trigger()` se ejecutará en todos los elementos devueltos por el objeto jQuery, `triggerHandler()` solo afecta al primer elemento encontrado.
- Los eventos creados con `triggerHandler()` no se propagan por la jerarquía DOM; si no son manejados por el elemento destinatario directamente, no hacen nada.
- A diferencia de `trigger()`, `triggerHandler()` no devolverá un objeto jQuery si no el valor que haya devuelto el último elemento que lo ha hecho ejecutar. Si ningún controlador lo desencadena, devuelve `undefined`.

En nuestra aplicación, los eventos son implementados de la siguiente manera, hay que distinguir entre dos cosas, la primera de ella que es cuando se crea el evento que esto se hace mediante:

```
\$("#id").trigger(eventtrigger,[eventtrigger,data])
```

Esta sentencia lo que hace es crear el evento, a esta función se le pasan dos parámetros:

Parámetros	Descripción
eventtrigger	Este es el nombre que le damos al evento, es importante que sea único e unívoco.
id	Identificador del elemento del DOM donde actuará el manejador.
data	Este data son los datos que nos devuelve la llamada a getJson

La segunda fase es cuando se activa el evento que esto se ejecuta en código de la siguiente manera:

```
\$("#nombreDiv").on("eventName",function(){
```

Los parámetros que necesitamos en esta línea son los siguientes:

Parámetros	Descripción
nombreDiv	Nombre del div o elemento donde tendrá efectos el evento
eventName	Nombre del evento que va a activar
function	(Opcional) Nombre de la función a ejecutar

Tras terminar esta sección de los eventos, tenemos completado el proceso por el cual obtenemos los ficheros, para resumirlo en una imagen esquema sería:

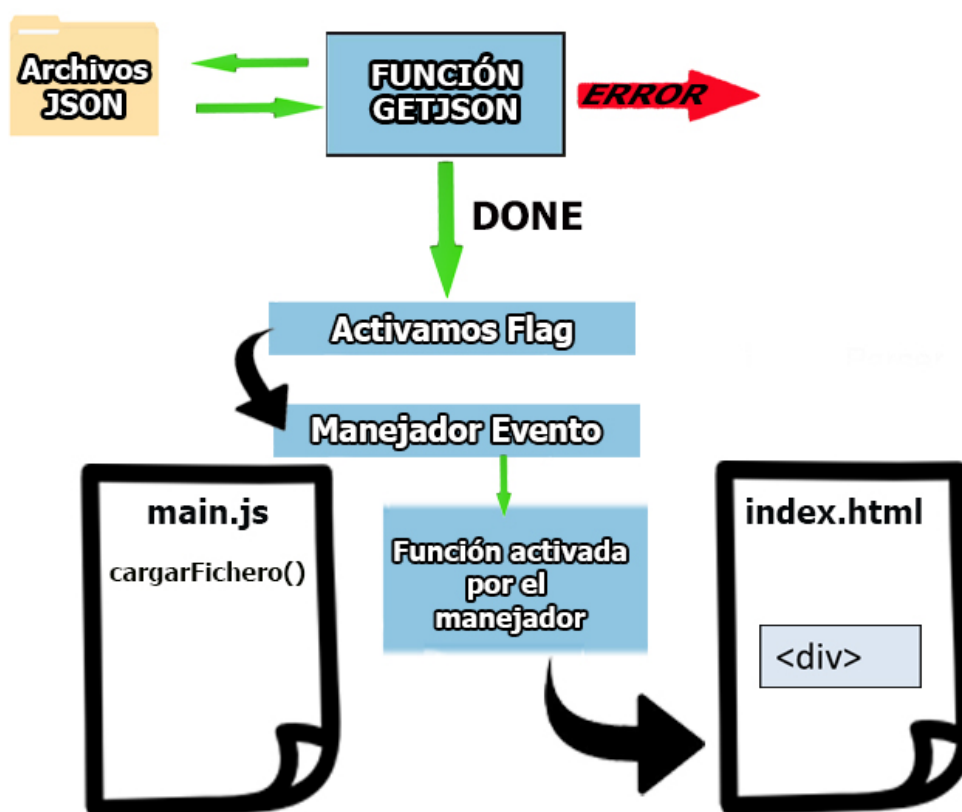


Figura 4.1: Esquema de la función cargar fichero

La siguiente pregunta que nos planteamos una vez obtenido los datos del fichero es ¿Qué hacemos con los datos?

4.2.3. Parseo JSON

Continuando con el proceso general y para responder a la pregunta que planteábamos en la anterior sección, ¿Qué hacemos con los datos?. Con los datos en nuestro poder estamos preparados para pasar a la fase de procesado con el objetivo de manejarlos a nuestro antojo.

Esta parte de la aplicación es muy importante, aunque a priori no tenga una gran repercusión, si está mal implementada podemos tener una aplicación lenta y con una gran necesidad de recursos para relaizar las acciones, con esto me refiero a que la forma en la que vamos a obtener los datos, va a jugar un papel muy importante de cara a la velocidad de procesado, y de cara al desarrollador evitará muchas líneas de código.

Introduciéndonos más en el propio código es muy importante que tengamos funciones muy genéricas, con esto quiero decir funciones que por ejemplo no le importe el tipo de fichero o

la métrica del interior, si no que tu le puedas pedir un parámetro y te lo pueda retornar sin importarle si es un fichero evolutionary, static o cualquier otro tipo.

En nuestro caso disponemos de varias funciones, de obtención de datos, el motivo de esto es que dependiendo el tipo de gráfica que utilicemos necesitamos servir, HighChart requerirá los parámetros de una forma o de otra.

Para entender el funcionamiento de esta parte es recomendable consultar el código fuente, que se encuentra disponible en GitHub.

4.3. Iteración 2

Se podría decir que ya estamos listos para representar cualquier fichero de los que vamos a trabajar, pero no podemos olvidarnos de entender el funcionamiento de la biblioteca que vamos a utilizar. HighChart es una biblioteca muy compleja, que puede llevar al desarrollador a una curva de aprendizaje muy pronunciada, ya que dispone de bastante configuración, en esta sección vamos a tratar de explicar como funciona la biblioteca de visualización empleada.

4.3.1. Entendiendo HighChart

HighChart es una biblioteca muy potente para la representación de gráficas, como todas las bibliotecas que poseen mucha funcionalidad necesitan una fase de aprendizaje, con respecto a esto hay que mencionar que la fase de aprendizaje en esta biblioteca es muy lineal, ya que en función del tiempo que le dediques, así son los resultados, es decir para usuarios que estén pensando en utilizar bibliotecas de visualización es muy recomendable ya que se pueden obtener resultados de forma muy rápida, además unos resultados más que aceptables.

Incluir la Biblioteca

Para tener disponible la biblioteca en primer lugar necesitamos descargárnosla de la página oficial.

En segundo lugar en nuestro fichero html principal en la parte de las cabeceras necesitamos incluir lo siguiente:

```
<!-- Librería Highcharts -->  
<script type='text/javascript' src="Highcharts-4.0.4%281%29/js/highcharts.js"></script>
```

```

<!-- Librería Highcharts 3D -->
    <script type='text/javascript' src="Highcharts-4.0.4%281%29/js/highcharts-3d.js"></script>
<!-- Librería Exporting -->
    <script type='text/javascript' src="Highcharts-4.0.4%281%29/js/modules/exporting.js"></script>
<!-- Librería Data -->
    <script type='text/javascript' src="Highcharts-4.0.4%281%29/js/modules/data.js"></script>
<!-- Librería drilldown-->
    <script type='text/javascript' src="Highcharts-4.0.4%281%29/js/modules/drilldown.js"></script>
<!-- Librería Highcharts More -->
    <script type='text/javascript' src="Highcharts-4.0.4%281%29/js/highcharts-more.js"></script>

```

Como podemos observar se insertan mediante la etiqueta *script* ya que es una biblioteca basada en funciones javascript, HighChart dispone de diferentes módulos, cada uno esta dedicado a una funcionalidad (3D,Exportación..)

Una vez tenemos incluido esto en nuestro index.html ya podemos utilizar la biblioteca.

Primera Representación

Lo primero antes de empezar a escribir código, es fundamental comprobar que la biblioteca que hemos incluido está funcionando correctamente, para ello podemos utilizar el siguiente ejemplo sencillo:

```

\$(function () {
    $('#container').highcharts({
        title: {
            text: 'Monthly Average Temperature',
            x: -20 //center
        },
        subtitle: {
            text: 'Source: WorldClimate.com',
            x: -20
        },
        xAxis: {
            categories: ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun',
                'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']
        },
        yAxis: {
            title: {
                text: 'Temperature (°C)'
            },
            plotLines: [{
                value: 0,
                width: 1,
                color: '#808080'
            }

```

```

    ]]
  },
  tooltip: {
    valueSuffix: '°C'
  },
  legend: {
    layout: 'vertical',
    align: 'right',
    verticalAlign: 'middle',
    borderWidth: 0
  },
  series: [{
    name: 'Tokyo',
    data: [7.0, 6.9, 9.5, 14.5, 18.2, 21.5, 25.2, 26.5, 23.3, 18.3, 13.9, 9.6]
  }, {
    name: 'New York',
    data: [-0.2, 0.8, 5.7, 11.3, 17.0, 22.0, 24.8, 24.1, 20.1, 14.1, 8.6, 2.5]
  }, {
    name: 'Berlin',
    data: [-0.9, 0.6, 3.5, 8.4, 13.5, 17.0, 18.6, 17.9, 14.3, 9.0, 3.9, 1.0]
  }, {
    name: 'London',
    data: [3.9, 4.2, 5.7, 8.5, 11.9, 15.2, 17.0, 16.6, 14.2, 10.3, 6.6, 4.8]
  }]
});
});

```

El resultado que debemos obtener es el siguiente:

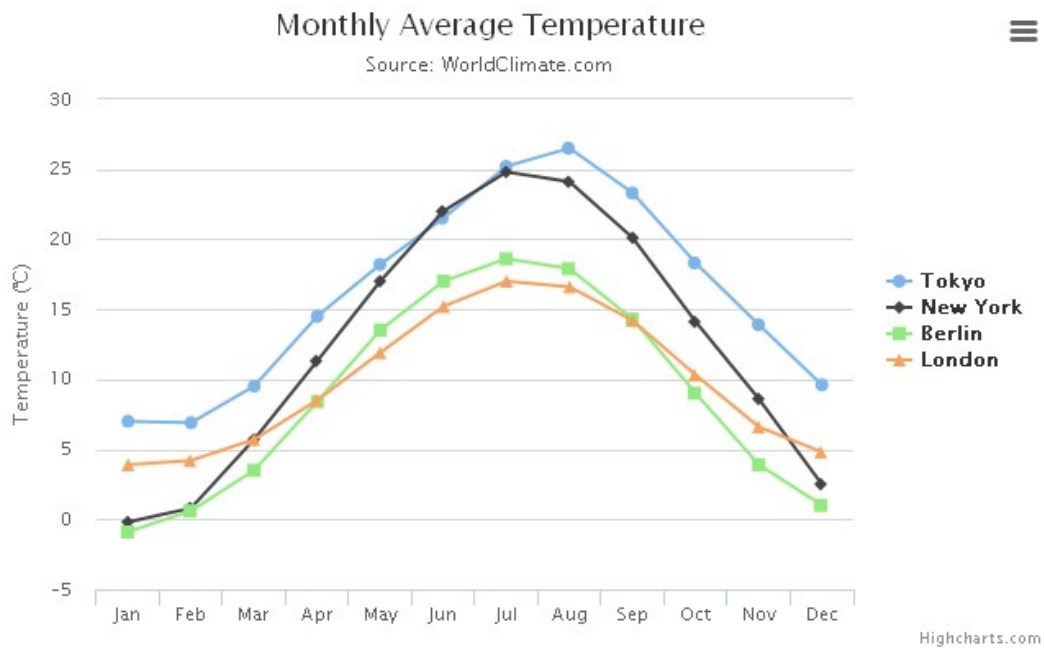


Figura 4.2: Ejemplo Representación HighCharts

Si obtenemos la representación superior es que todo ha ido bien, y la biblioteca esta funcionando sin problemas.

Si nos fijamos en la función superior de representación los datos estan metidos directamente en el propio código, y eso no es eficiente porque si no necesitaríamos todas esas líneas de código para cada representación. Otra vez siguiendo las reglas de código, queremos una función a la cual la pasemos el div de destino y los datos y sea capaz de representar el gráfico.

Esto se consigue con la otra forma que posee la biblioteca para introducir datos, que es mediante una variable options:

```
var options = {
  chart: {
    events: {
      click: function (event) {

      }
    },
    zoomType: 'x',
    panning: true,
    panKey: 'shift',
    renderTo: graficaId
  },
  title: {
```

```

        text: arrayDatos[2],
        x: -20
    },
    xAxis: {
        type: 'datetime',
    },
    plotOptions: {
        series: {
            pointStart: Date.UTC(arrayDatos[5], arrayDatos[4], 0),
            pointInterval: (24*3600*1000)*6 // one day
        }
    },
    yAxis: {
        title: {
            text: ''
        },
        min: 0,
        plotLines: [{
            value: 0,
            width: 1,
            color: '#808080'
        }]
    },
    legend: {
        enabled: false,
        layout: 'vertical',
        align: 'right',
        verticalAlign: 'middle',
        borderWidth: 0
    },
    series: [{
        name: arrayDatos[6],
        data: arrayDatos[0]
    }]
};

```

Con esta variable tenemos cargada toda la información lista para pasársela a HighChart y representar los datos. Este es un ejemplo de variable que es utilizada en el proyecto, a lo largo del proyecto esta variable se irá configurando en función del gráfico que se quiera representar ya que unos gráficos permiten unas configuraciones u otras.

Si os preguntáis cual es el objetivo de tener esta variable, aunque supongo que la mayoría de los lectores lo habréis deducido, permitirme que explique que dicha variable se puede introducir en una funcion por ejemplo con la siguiente cabecera:

```
function pintarGeneral (arrayDatos, graficaId){
```

Esto nos permite llamar todas las veces a dicha función y solo tenerla en código una vez, siempre que le pasemos correctamente los datos en el arrayDatos y un identificador de un div del DOM, ya seremos capaz de tener lista nuestra variable para representar la gráfica. Dicho arrayDatos se optiene de la sección anterior concretamente es la salida de las funciones parseadoras.

Por último solo nos falta instanciar nuestro elemento, para ello tendremos que llamar a nuestro constructor, esto se hace mediante:

```
var chart = new Highcharts.Chart(options);
```

Que tiene los siguientes parámetros:

Parámetros	Descripción
options	Variable que contiene los datos de estilo de la gráfica (Requerido).

Una vez instanciado el objeto ya tendríamos visible nuestra gráfica en el div que le hemos indicado, por lo que ya hemos sido capaces de de crear nuestro gráfico.

Pero además permitirme que explique que hay otra forma de introducir datos en la variable options, sin necesidad de volver a la función original, es decir, por ejemplo imaginaros que hemos llamado a la función anterior de pintarGeneral y nos ha devuelto la variable options, pero por ejemplo queremos añadir un evento a esa gráfica, tenemos dos opciones o nos configuramos otra función que por defecto devuelva una varibale options, o la mas rápida y que mejor se adapta a las reglas de código limpio es utilizar la siguiente propiedad de la biblioteca:

```
Highcharts.setOptions({  
  plotOptions: {  
    series: {  
      cursor: 'pointer',  
    }  
  }  
})
```

Con esto ya tendríamos colocado en nuestra varibale options una nueva configuración, concretamente el cambio introducido sería que al estar encima de la serie se nos cambiaría el cursor de la flecha tradicional por la manita, muy útil este cambio para avisar al usuario de que hay una opción contenida.

Dentro de todas las propiedades que permite HighChart, tiene muchas opciones que se pueden configurar en la gráfica, tampoco es mi objetivo explicar la biblioteca entera ya que sería casi imposible por que posee múltiples opciones, invito a todos los lectores que pasen por la página y en concreto por la API ya que esta muy detallada y con muchos ejemplos de uso, además que permite editarlos online.

Una primera prueba juntado todas las secciones anteriores obtendríamos un dashBoard con una apariencia muy simple, pero que por detrás está corriendo bastante lógica ya que tenemos un fichero .html el cual tiene un código JavaScript que se encarga de la obtención de los datos mediante `getJSON`, devolviéndonos los datos en una variable, que una vez que hemos terminado de realizar completamente la carga, levantamos un trigger para avisar "He Terminado", y volver a la ejecución principal y continuar con el parseo para la posterior representación.

4.3.2. Introducción de Eventos

Una vez tenemos nuestras gráficas representadas, para ganar mayor funcionalidad, vamos a introducir eventos, ya que la biblioteca que hemos seleccionado permite disparar eventos al realizar algún cambio en la gráfica.

Los eventos nos van a permitir que podamos realizar acciones cuando pinchemos o clicamos en alguna parte de la web, concretamente los eventos de las gráficas van a servir para que podamos obtener información de esta gráfica, o bien del punto donde estamos pinchando o que nos muestra otras gráficas relacionadas para ampliar información, si lo miramos desde otro punto de vista los eventos nos permiten ganar profundidad en los datos, es decir nos permite ir escarbando en los datos con el objetivo de conocer más información.

Como comente anteriormente la mejor forma de introducir eventos es mediante la función que dispone HighChart de `SetOptions`, pero nuevamente para introducir eventos necesitamos conocer que significada cada parámetro y saber lo que queremos hacer, para conseguir realmente lo que estamos buscando:

En esta biblioteca podemos poner varios tipos de eventos, además que dichos eventos pueden ser colocados en áreas, regiones... En nuestro caso nos centraremos en los eventos que se ponen en las series, ya que es donde el usuario va a querer obtener información. Dentro de las series disponemos de los siguientes tipos de eventos:

- **AfterAnimate:** Los eventos se activan después de que la serie ha terminado su animación inicial, o en caso de que la animación está desactivada, se ejecutará inmediatamente cuando se muestre la serie.
- **CheckboxClick:** Se activa cuando se hace clic en la casilla junto al nombre de la serie en la leyenda. Se puede pasar un parámetro a la función, y para acceder al estado de la checkboxes se accede mediante `event.checked`.
- **Click:** Se activa cuando se hace clic en la serie. Un parámetro, se pasa a la función. Este parámetro contiene información de eventos común basado en jQuery o Mootools dependiendo de la biblioteca base que se este utilizando. Además, `event.point` mantiene un puntero al punto de la gráfica más cercano.
- **Hide:** Se activa cuando la serie se oculta tras el tiempo de generación de la gráfica, ya sea haciendo clic en el elemento de leyenda o llamando a la función `.hide()`
- **LegendItemClick:** Se activa cuando se hace clic en el elemento de leyenda que pertenece a la serie. Un parámetro es pasado a la función. Este evento es muy útil cuando queremos eliminar series al clickar en la leyenda, además permite mostrar un mensaje de confirmación
- **MouseOut:** Se activa cuando el ratón sale fuera de la serie, es decir no estamos encima de ella. En el parámetro que se pasa a la función del evento contiene información. Además dispone de la opción `stickyTracking`, que hace que se dispare el evento si el cursor se sale fuera del recuadro de la gráfica.
- **MouseOver:** Se activa cuando el ratón pasa por encima de la serie. Proporciona un parámetro con información útil para la función.
- **Show:** Se activa cuando se muestra la serie después del tiempo de generación del gráfico, ya sea haciendo clic en el elemento de leyenda o llamando a la función `.show()`.

Como comente anteriormente cuando queremos activar un evento en una determinada gráfica puesto que tenemos las funciones muy generalizadas no nos interesa crearnos una función nueva si no que utilizamos las facilidades que ofrece HighChart.

```

Highcharts.setOptions({
  plotOptions: {
    series: {
      events: {
        checkboxClick: function (event) {
          alert("Evento CheckboxClick")
        },
        afterAnimate: function (event) {
          alert("Evento AfterAnimate")
        },
        click: function (event) {
          alert("Evento Click")
        },
        hide: function (event) {
          alert("Evento Hide")
        },
        legendItemClick: function (event) {
          alert("Evento LegendItemClick")
        },
        mouseOut: function (event) {
          alert("Evento MouseOut")
        },
        mouseOver: function (event) {
          alert("Evento MouseOver")
        },
        show: function (event) {
          alert("Evento Show")
        }
      }
    }
  }
})

```

Como se puede observar, en el código superior habría activado todos los eventos sobre la variable options que dispongamos. De cara a la aplicación, lo más útil es que se trabaje sólo con eventos de click, para poder garantizar el control del usuario sobre la aplicación.

4.3.3. Implementación

El resultado de esto, es el que mostramos a continuación en las siguientes dos imágenes, en la primera de ellas aparece nuestro dashboard con una única gráfica de demografía y al hacer click en un período nos muestra los usuarios que estan dentro de ese rango.

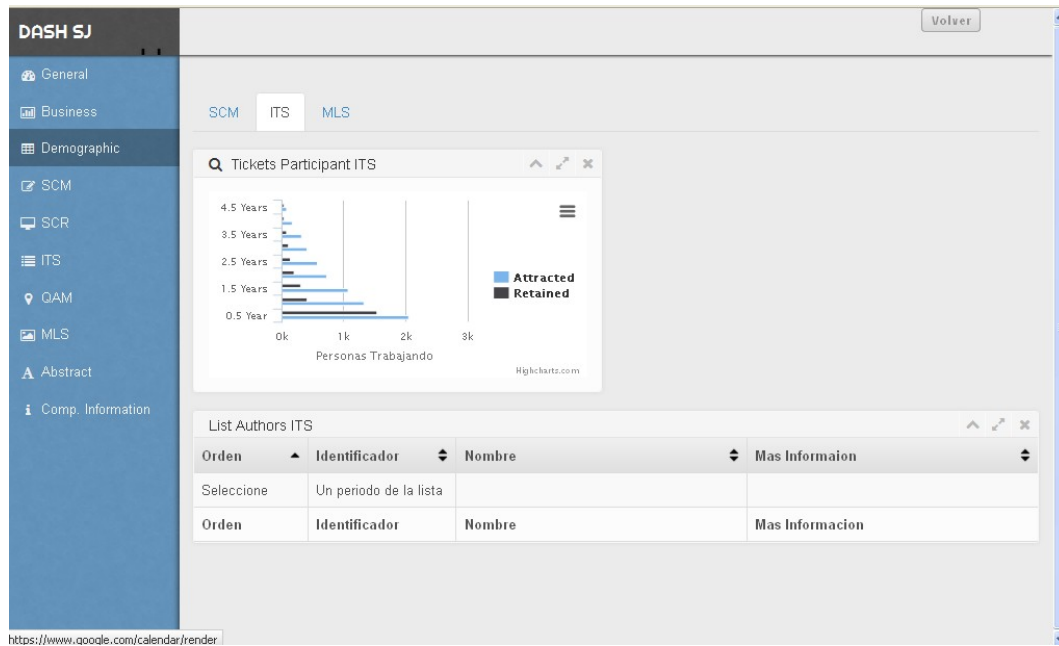


Figura 4.3: Antes de Disparar el Evento

Y este es el resultado del evento activado, es decir se ha rellenado una tabla con las personas del período seleccionado:

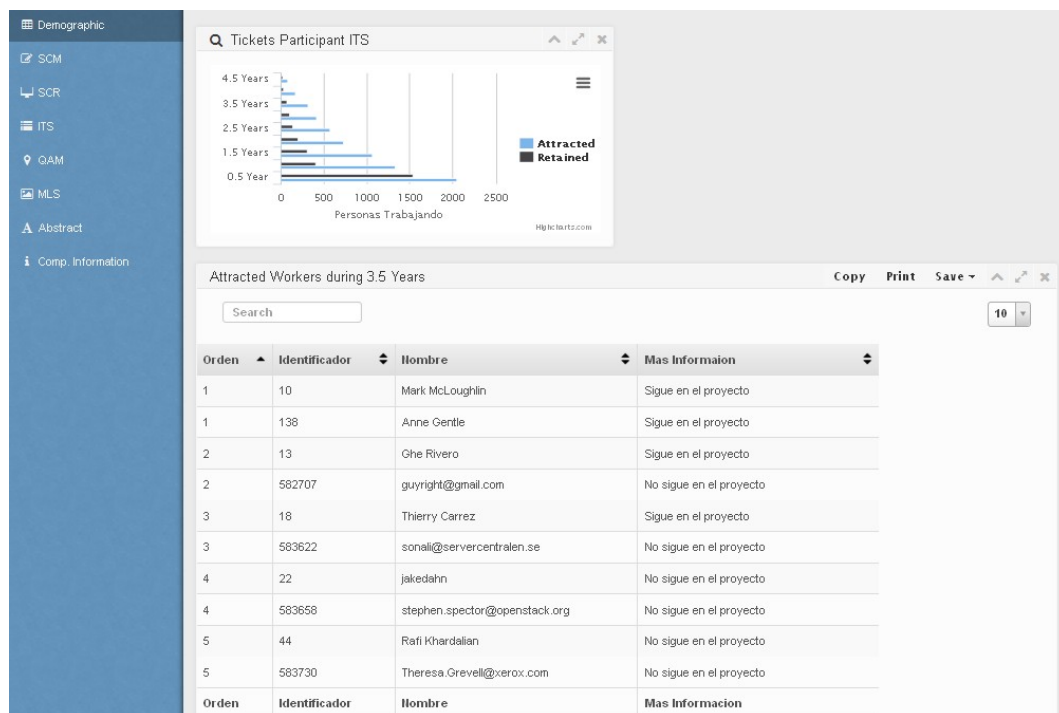


Figura 4.4: Después de activar el Evento

Además aprovecho para poner un ejemplo profundidad que comentaba anteriormente, si volvemos hacer click en uno de los usuarios y ese usuario dispone de fichero propio de estadística, mostraríamos la información de ese usuario. Con esto lo que hemos conseguido es empezar en una gráfica muy genérica en la que solo había números y de ahí hemos pasado a unas tablas con personas, disparando un nuevo evento para un usuario seleccionado hemos llegado a obtener su información específica. A continuación se puede ver una imagen del resultado final de el proceso de escarbado en lo datos.



Figura 4.5: Obtención datos de un usuario específico

4.4. Iteración 3: Diseño BootStrap

En esta iteración nos centramos en introducir bootstrap en nuestro proyecto, para ganar una mejoría en la interfaz de usuario a la vez que obtenemos un dashboard fácil e intuitivo, y adaptable a cualquier dispositivo.

Para incluir bootstrap no basta con incluir una plantilla y ya esta, si no que tenemos que entender el funcionamiento de bootstrap, examinar el código de ejemplo es fundamental y guardar los estilos es una regla básica, ya que si modificamos el dom a nuestro antojo perderemos la responsividad.

Dentro de esta iteración podemos dividir el planteamiento en diferente secciones donde se abordaran cada uno de los elementos introducidos:

4.4.1. Gráficos Charts

Utilizamos para introducir nuestros gráficos un chart proporcionado por BootStrap, pero configurándolo para que tenga un aspecto personalizado, el resultado es el siguiente:



Figura 4.6: Chart empleado para las gráficas

Dichos charts tiene las siguientes características:

- Botón información: Dispara un popup en la que podemos mostrar información de la gráfica, por ejemplo una pequeña explicación de la gráfica. Este popup carga el contenido del mensaje de un fichero `config.properties`

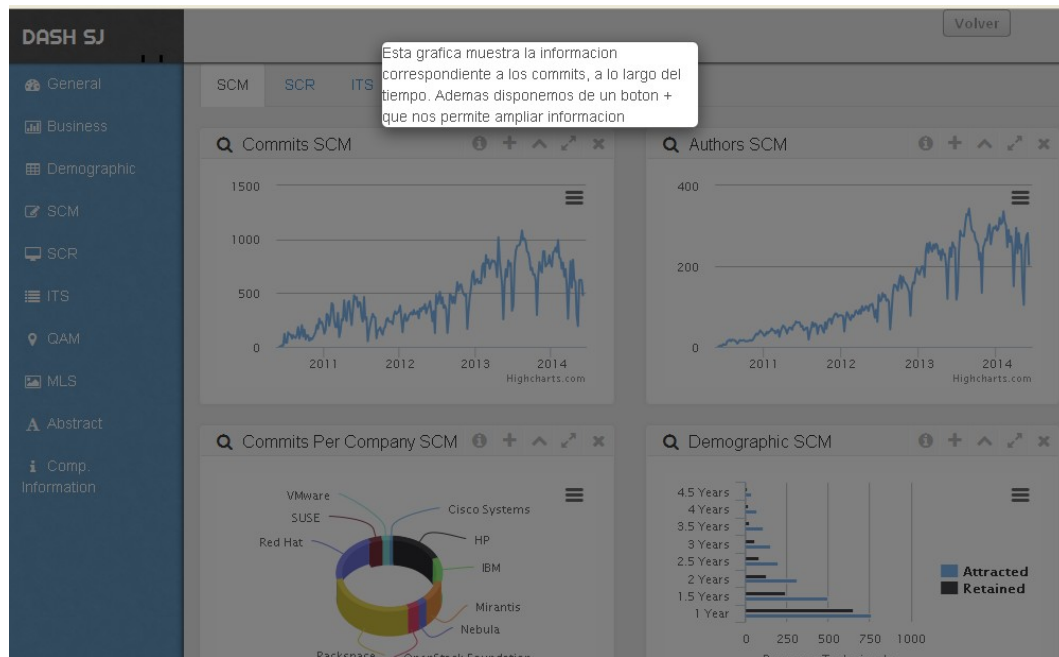


Figura 4.7: Popup de información

- **Botón +:** Este botón nos muestra más información sobre esta gráfica llevándonos a otras gráficas relacionadas con esta misma, con esto permitimos que el usuario navegue a través de las gráficas sin la necesidad de utilizar el menú general.
- **Botón ^:** Esto nos permite ocultar la gráfica y dejar visible solo la cabecera por si queremos reducir el espacio y dejar visible otras gráficas inferiores, esto nos permite evitar estar haciendo scroll todo el rato, para ver las gráficas inferiores.

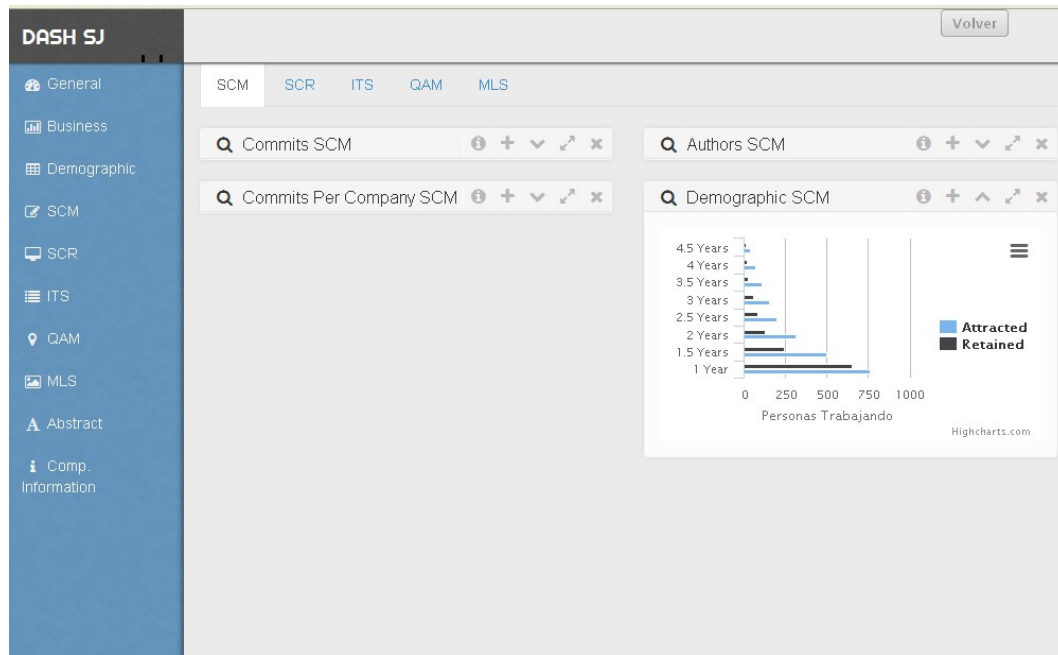


Figura 4.8: Dejar visible solo la cabecera del chart

- Botón \uparrow : Esto nos aumenta el div y nos lo saca mucho más grande en primer plano. Esta opción no está disponible para highchart ya que no tiene la opción de redimensionamiento en caliente.

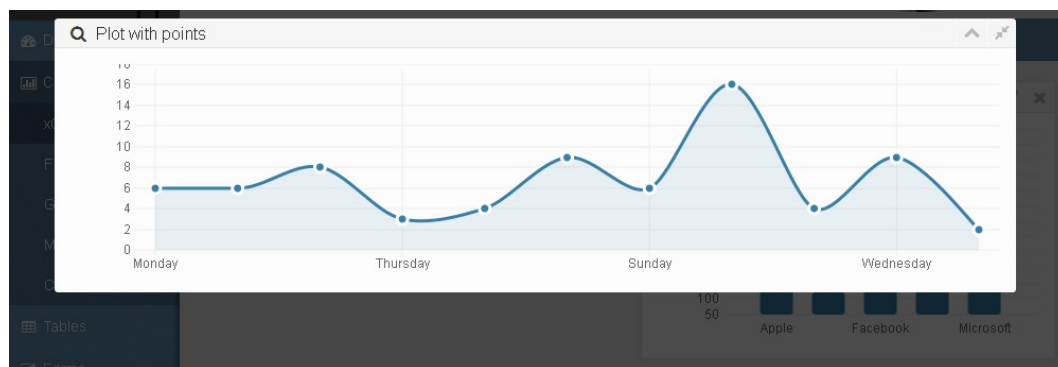
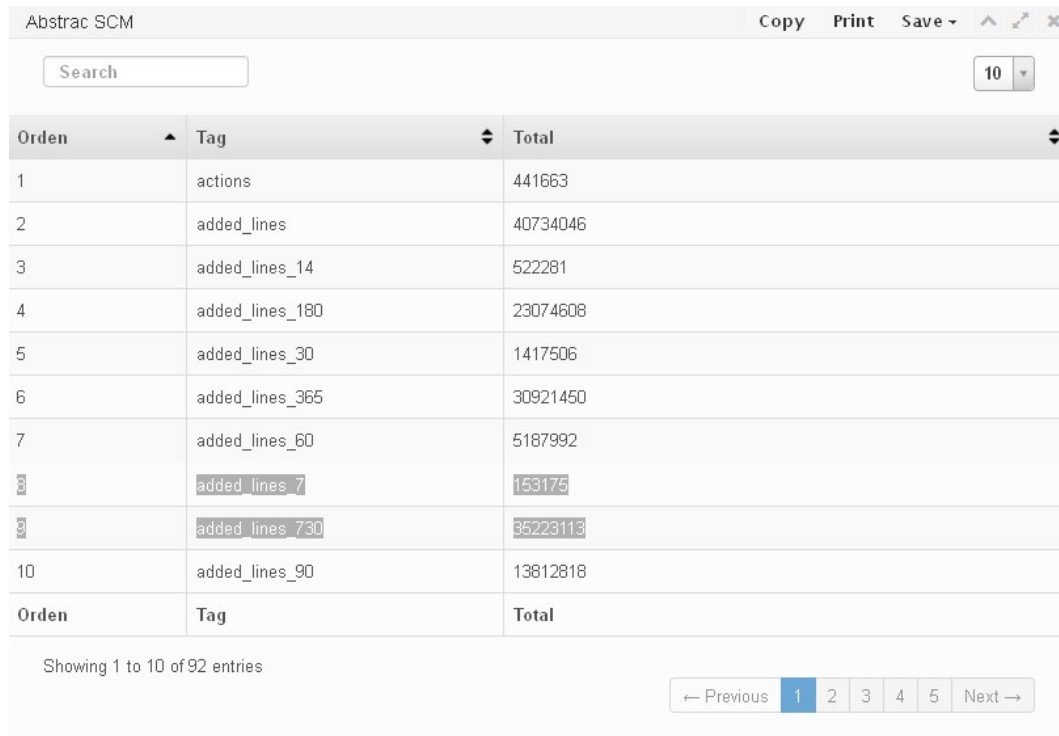


Figura 4.9: Ampliar Chart a pantalla completa

- Botón x: Elimina esta gráfica de la pestaña en la estamos: Se podrá volver a ver el chart borrado una vez que entremos en la pestaña otra vez.

4.4.2. Tablas de Ordenación

En nuestro dashboard hemos incluido tablas para mostrar resultados, dichas tablas se han desarrollado con las siguientes características:



Orden	Tag	Total
1	actions	441663
2	added_lines	40734046
3	added_lines_14	522281
4	added_lines_180	23074608
5	added_lines_30	1417506
6	added_lines_365	30921450
7	added_lines_60	5187992
8	added_lines_7	153175
9	added_lines_730	35223113
10	added_lines_90	13812818

Showing 1 to 10 of 92 entries

← Previous 1 2 3 4 5 Next →

Figura 4.10: Tablas empleadas para la ordenación

Dicha gráfica tiene las siguientes características:

- Dispone de un botón de impresión, donde nos muestra todas las entradas de la tabla, a pantalla completa para que la podamos imprimir.
- Botón de Save: Nos permite guardar exportar las tablas a diferentes formatos, en función de lo que desee el usuario. Los formatos disponibles son: CSV, EXCEL, PDF.
- Incorpora un buscador el cual busca por cualquier columna, con la ventaja de que el buscador es instantáneo, esto implica que el usuario ve en el acto el resultado del filtrado.
- Las columnas tienen opción de ordenar por columnas de mayor a menor y viceversa.
- Incluye paginación para que la tabla se adapte a nuestras necesidades y sea más fácil manejarla al buscar la información, además incluye la elección del número de elementos

que queremos mostrar.

- También tienen incorporados eventos para sacar información sobre la fila que clickamos.

4.4.3. Tabs

En ciertas partes del dashboard, se han incluido pestañas, con el objetivo de que el usuario tenga mejor acceso y más ordenada la información que desea consultar.

Esta opción es posible verla en las pestañas donde se muestra información de varios tipos, por ejemplo, General, Demographic, Business.

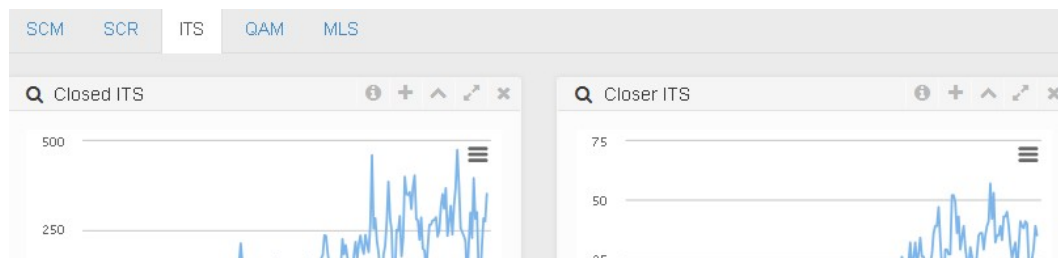


Figura 4.11: Ejemplo de Pestañas utilizadas en la app

4.4.4. Fichero Properties

La información de la gráfica se obtiene mediante un fichero properties, la ventaja de usar este tipo de fichero properties es que, de cara a un futuro si queremos cambiar la información que mostramos con ir al fichero config.properties podemos cambiar los mensajes.

Esto es muy útil si el día de mañana queremos que nuestro dashboard se implemente en diferentes idiomas. Además no necesitaríamos que esta función fuese desarrollada por un programador ya que es un simple fichero por lo que cualquier persona podría cambiar los mensajes.

Para utilizar, dicha configuración de fichero, hemos necesitado utilizar una biblioteca llamada messageResource.

Instalación de la Biblioteca

Esta biblioteca nos permite la lectura de un fichero .properties para poder utilizar dicha biblioteca necesitamos en primer lugar descargar y guardarla en el directorio de nuestro proyecto.

En segundo lugar necesitaremos incluir la ruta en nuestro archivo index.html para ello introduciremos la siguiente línea en el código:

```
<script type="text/javascript" src="js/messageResource.min.js"></script>
```

Una vez tenemos esta línea ya podemos inicializar en nuestro código el elemento que se encargará de leer del fichero:

```
messageResource.init({
    filePath : ''
});
```

En esta inicialización sólo tenemos que rellenar el siguiente parámetro:

Parámetros	Descripción
filePath	Es la ruta donde se encuentra el fichero .properties

Con esto ya estamos listos para obtener los valores del fichero properties, para obtener un valor se hace igual que si fuera un mapa de datos, mediante su clave. En javascript se ejecuta con la siguiente sentencia:

```
messageResource.load(filename, function(){
    var value = messageResource.get(oID, filename);
});
```

Parámetros	Descripción
filename	Nombre del fichero properties sin extensión
oID	Key de la cual queremos obtener su valor

Esta sentencia lo que hace es cargar el fichero properties del nombre que le indicamos y con la función .get obtenemos el valor correspondiente a una clave introducida. Una vez ejecutado este código tendremos en la variable value, nuestro valor para mostrar al usuario o introducir en otra función.

4.4.5. Opción de Volver Atrás

Esta opción que apriori parece ser una cosa bastante sencilla, pero al utilizar una única url que no cambia puede ser algo confuso la implementación.

En el diseño tradicional, esta opción como os podeís imaginar es tan sencilla como guardar en el enlace de dicho boton la url de la que venimos y ya tendríamos implementado nuestro botón de regreso.

Pero ahora, si nos pasamos a una web donde la url no cambia, es decir solo se ocultan o muestran los div. ¿Cómo sabemos donde pinchamos?¿Cómo sabemos donde estábamos?.

El objetivo de esta sección es que el lector entienda y vea como se ha implementado. Para ello es necesario parar a reflexionar antes de escribir líneas y líneas. Una posible idea es guardar en una variable, una etiqueta que haga referencia a la pestaña que vamos a clicar, pero si implementamos esta opción, tenemos un problema, ¿Qué sucede si el usuario quiero navegar dos veces para atrás?...Pues que estaríamos cargando siempre la misma página...

Aunque la solución no termina de ir mal encaminada si a esto le añadimos la palabra array tenemos la fórmula de ejecutarlo.

Con el array lo que vamos ir haciendo es ir guardando en cada posición la etiqueta que hace referencia a la pestaña que el usuario clickea, con esto tenemos el historial entero de navegación.

Esto en el código se implementa de la siguiente forma:

```
\$('#regress').click(function(){
    console.log("Longitud:" + history.length)
    if (history.length == 1){
        //No hacemos nada el usuario no ha navegado
    }
    else{
        history.pop();
        indice = history.length;
        eventRegress = history[indice-1]
        \$("#*").trigger(eventRegress, [eventRegress])
    }
})
```

Ahora, procedemos a explicar el código para que quede del todo claro:

```
if (history.length == 1){
```

Con esta línea de código lo que comprobamos es que si la longitud es igual a uno, el usuario no se ha movido de la página inicial por lo tanto no hacemos ninguna acción, por el contrario si es distinta de uno, esto implica que ha estado navegado por la aplicación.

Una vez que sabemos que la longitud es diferente de uno:

```
        history.pop();  
    })
```

En esta línea lo que hacemos es eliminar el último elemento del array, este último elemento corresponde a la página donde se encuentra actualmente el usuario, además aunque en nuestro caso no lo utilizamos la función `pop()` elimina y guarda ese último registro en una variable si lo deseamos. Una vez hemos borrado el registro:

```
    indice = history.length;  
    eventRegress = history[indice-1]
```

Calculamos la longitud del array con esa última posición eliminada, y sin olvidarnos que los array comienzan en cero por lo que a la longitud total le restamos uno y ya tenemos el elemento que nos llevará a la página anterior.

```
    \$("#*").trigger(eventRegress, [eventRegress
```

Por último sólo nos queda levantar un evento con esa etiqueta, dicho evento será atendido por un manejador que está escuchando para desatar las acciones asignadas.

En alguna parte del código, debemos tener el manejador que desate las opciones, como explicamos en secciones anteriores un manejador de eventos en este caso tendremos la siguiente instrucción:

```
    \$("##regress").on(eventRegress, function(event, trigger) {  
  
    }
```

Y el código del interior del evento, será ocultar y mostrar lo `div` que necesitemos. Con esto queda terminado el proceso de regresión a la página anterior.

4.5. Iteración 4: Deploy en un Servidor

Hasta ahora, nuestra aplicación estaba corriendo en nuestra máquina local, el siguiente paso es subirlo a un servidor de internet para que la aplicación pueda ser consultada en cualquier parte del mundo y por cualquier persona que disponga de acceso a internet.

En nuestro caso la hemos alojado en un servidor gratuito el cual, nos proporciona un almacenamiento limitado, pero suficiente para alojar nuestra aplicación.

Para poder realizar la transferencia de archivos, lo vamos a realizar mediado FTP, ayudándonos de una aplicación para la transferencia de archivos CutFTP.

4.5.1. Configuración CutFTP

Una vez nos hemos registrado, el servidor nos proporcionará una serie de parámetros que son los que tendremos que configurar en nuestro CutFTP. La configuración es la siguiente:

Label for site: Nombre del Sitio. FTP Host Address: ftp.sudominio.com. FTP Site Username: Su nombre de usuario. FTP Site Password: Su contraseña. FTP Site Connection Port: 21. Login Type: Normal.

Una vez tenemos esto configurado ya podemos conectarnos al gestor de archivos, ahora simplemente arrastrando nuestra carpeta donde tenemos la aplicación, ya la tendremos alojada en el servidor.

Capítulo 5

Detalles del Producto Final

En este capítulo el objetivo es explicar la aplicación desde otro punto de vista, el primer punto es entender como esta organizada la arquitectura de la aplicación, la arquitectura es la parte más importante ya que son los cimientos del proyecto, tener una arquitectura clara y precisa evita muchos problemas a posteriori.

Por otro lado también se explicará la aplicación al detalle para que los usuarios entiendan lo que significan todas las siglas y apartados del dashboard.

5.1. Arquitectura general

Antes de empezar el desarrollo de un proyecto es muy importante, dedicar unas cuantas jornadas para plantear como va a ser el proyecto, y como se va a desarrollar así como conocer cual es la arquitectura que mejor se adapta a nuestras necesidades, cuando hablamos de arquitectura nos estamos refiriendo a como va estar dividido el proyecto las capas que va a tener etc. Este proyecto una vez conocida la idea, la forma de abordarla a sido la siguiente:

- **Obtención de Fichero:** Esta capa del proyecto se enfrentará a todo el proceso de coger un archivo así como los problemas que nos podemos encontrar al realizar dichas peticiones.

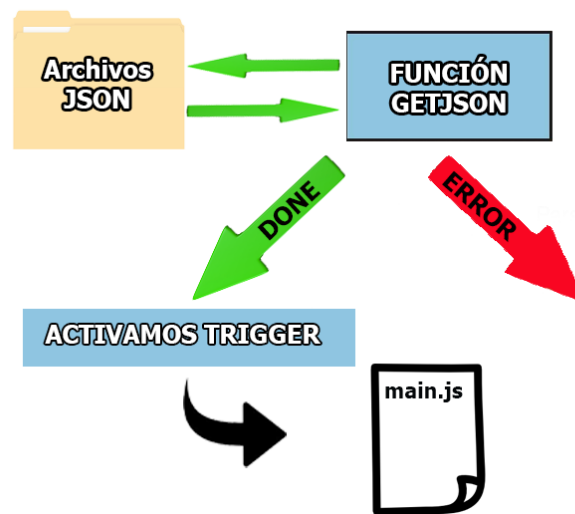


Figura 5.1: Estructura de la Obtención del Fichero

- **Tratamiento de Datos:** Aquí nos encargaremos de una vez obtenidos esos datos, como hay que tratarlos para masajearlos y obtener los datos que estamos buscando.
- **Representación:** En esta sección nos encargaremos de las funciones genéricas que vamos a implementar con el fin de cumplir las reglas de no repetir código y reutilizar las máximas funciones posibles, para ello necesitamos conocer perfectamente el objetivo.
- **Implementación:** Desarrollo del propio código, es decir aclarar que queremos mostrar en cada pantalla, y las colocación de las gráficas
- **Diseño Visual:** Parte muy importante ya que es lo que primero va a ver el usuario, cuando abra nuestra aplicación, para ello utilizaremos tecnologías punteras como BootStrap.

Fruto de esta subdivisión, tenemos el siguiente gráfico, para aclarar y terminar de comprender la arquitectura general:

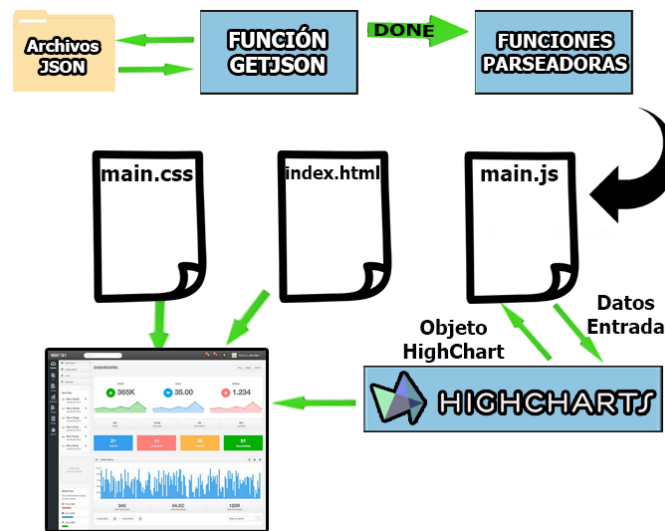


Figura 5.2: Arquitectura General

Dentro de las fases de trabajo, se puede dividir en las siguientes secciones:

- **Iteración 0.** Una fase previa que se limitará al estudio de la estructura de los diferentes documentos a *parsear*, así como las herramientas que disponemos para elaborarlo. En realidad se trata de un paso anterior al desarrollo y, por tanto, no implica un avance en el proceso ni desemboca en un prototipo, pero la vital importancia de esta labor nos obliga a dedicarle un lugar destacado en la evolución del proyecto.
- **Iteración 1.** Una primera fase cuyo objetivo es la creación del sistema de *Obtención de Fichero* básico que establecerá la base para cualquier otro fichero, es decir esta fase bien planteada, en un futuro no debería llevarnos a modificar dicha iteración.
- **Iteración 2.** Una segunda fase que pretende extender la funcionalidad de *HighChart*, para poder conseguir nuestro objetivo final de un dashboard con eventos.
- **Iteración 3.** En esta fase esta orientada al aspecto visual donde introduciremos a nuestro dashboard un aspecto mucho más elegante para conseguir llamar la atención del usuario.
- **Iteración 4.** Una última fase dedica a la instalación de la aplicación en un servidor real, para que pueda estar accesible a cualquier persona que disponga de conexión a internet.

5.2. Descripción funcional completa del Dashboard

En esta sección, se explicara el funcionamiento completo de la aplicación, es decir la guía del usuario.

Una vez hemos entrado, lo primero que nos encontramos es una primera vista de un loading, mientras esto se ejecuta se está procediendo a cargar todos los estilos y al procesado de la página, este proceso puede tardar aproximadamente dos minutos, dependiendo también del ordenador en el que nos encontremos, esto permite al usuario una navegación mucho más fluida. Este es el aspecto que nos encontramos al entrar en la aplicación:



Figura 5.3: Loading Principal

Algún lector, pensará el ¿Por qué de un loading?. El motivo de que se introduzca un loading nada más entrar, tiene su fundamento en la rapidez que demanda un usuario, debido a que estamos manejando una gran cantidad de datos, los cuales tienen que sen procesados con el coste computacional que conlleva, si a esto le juntamos la carga de estilos que se produce en tablas, pestañas, charts...Se ha preferido la opción de mantener al usuario esperando aproximadamente 1 minuto, y que luego la velocidad de navegación por la aplicación se inmediata.

Esto opción se ha elegido porque esta demostrado tras un estudio que se ha realizado, que una gran mayoría de usuarios prefería esperar una vez un poco más y que luego la navegación sea prácticamente inmediata.

Una vez se han cargado los estilos y los datos, el loading desaparece automáticamente mostrando al usuario la página general, donde tenemos un resumen de cada unas de las métricas que se utilizan agrupadas por pestañas.

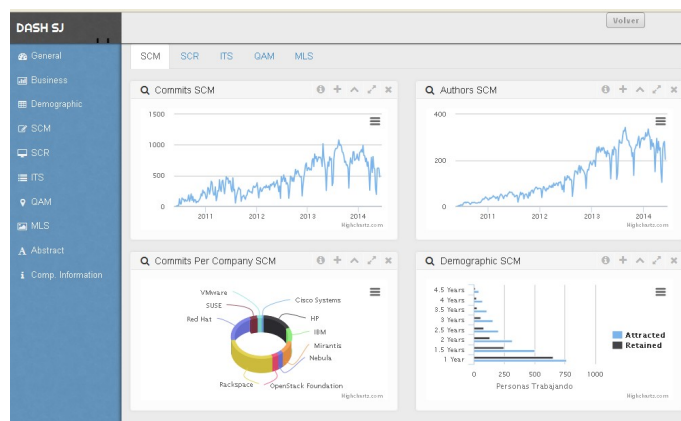


Figura 5.4: Pestaña General

Una vez que estamos en la pestaña podemos escarbar en los datos ampliando la información con el icono + de los chart, cada uno de los gráficos nos lleva a una información diferente, pero siempre relacionado con el widget que hemos seleccionado. Esto permite al usuario moverse por toda la aplicación a través de los widgets, sin necesidad de utilizar el menú.

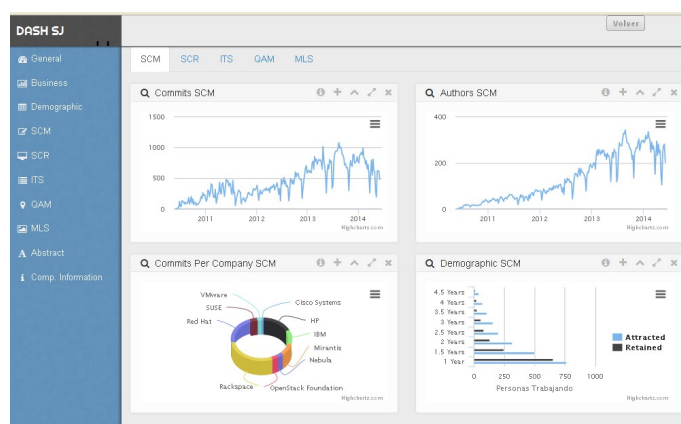


Figura 5.5: Ampliar Informacion Icono +

A continuación procedemos a explicar el menú principal:

- **Business:** En esta pestaña muestra la información de las empresas que proporcionan sus métricas para el estudio, dentro de ellas están clasificadas según el estudio que se haga SCM ITS o MLS.

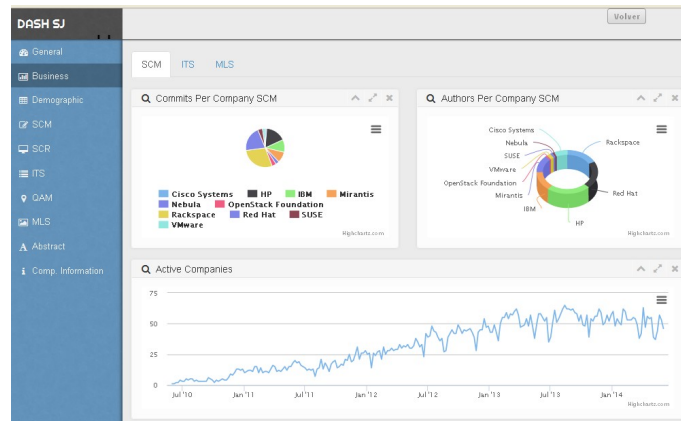


Figura 5.6: Pestaña Business

Además permite disparar eventos para mostrar la información de dicha empresa, clickando en cualquiera de las gráficas circulares.

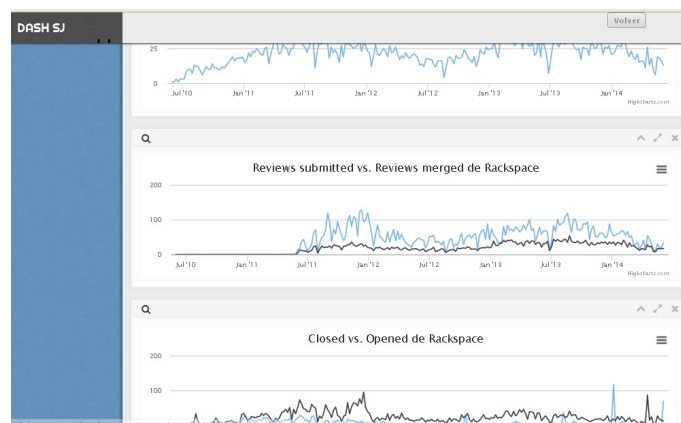


Figura 5.7: Evento Activado Business

- **Demographic:** En esta pestaña, el usuario puede ver la demografía según los períodos de edad, y según si continúan o no en el proyecto.

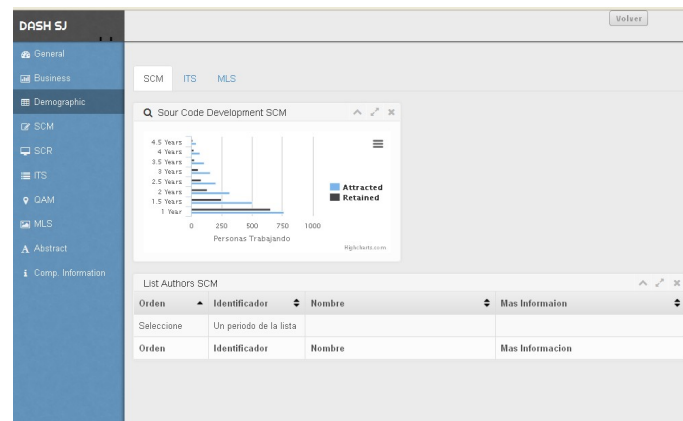


Figura 5.8: Pestaña Demographic

Nuevamente permite escharbar en los datos, ya que si hacemos click en un período de la gráfica, obtendremos en la tabla inferior el nombre de todos los trabajadores de ese período.

Orden	Identificador	Nombre	Mas Informaion
1	10	Mark McLoughlin	Sigue en el proyecto
1	94	Tom Fifield	Sigue en el proyecto
1	138	Anne Gentle	Sigue en el proyecto
2	13	Ghe Rivero	Sigue en el proyecto
2	847	Jordan Rinke	No sigue en el proyecto
2	582707	guyright@gmail.com	No sigue en el proyecto
3	18	Thierry Carrez	Sigue en el proyecto
3	848	gholt	No sigue en el proyecto
3	583622	sonali@semercentralen.se	No sigue en el proyecto
4	22	jakedahn	No sigue en el proyecto

Figura 5.9: Evento Activado Demographic

Y si clickeamos en el nombre de un usuario y dicho usuario dispone de fichero propio de estadística nos mostrará la información de ese usuario seleccionado:

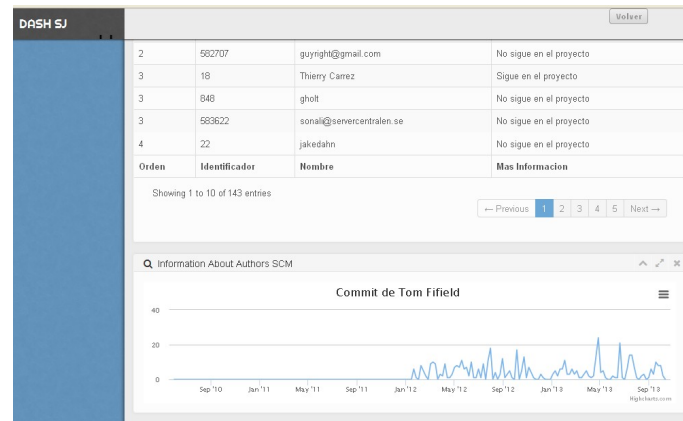


Figura 5.10: Evento Activado Demographic Usuario

- Source Code Management: Muestra la información referente a esta métrica, con la siguiente apariencia:

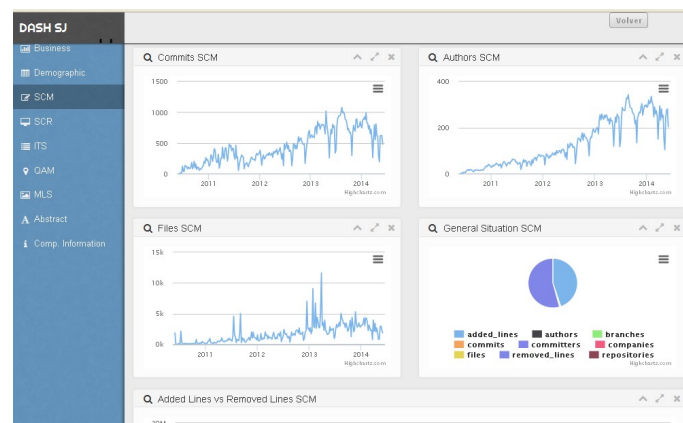


Figura 5.11: Pestaña Source Code Management

De esta pestaña cabe destacar que podemos representar, los datos de la tabla pinchando sobre cada uno de ellos, además también nos permite comparar varios pinchado la tecla Ctrl + Click.

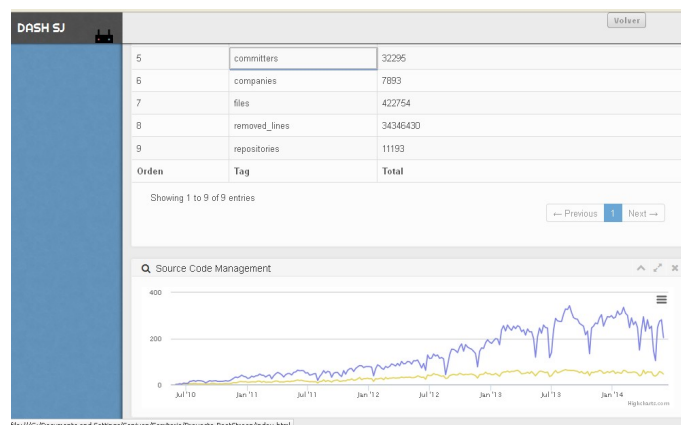


Figura 5.12: Evento activado en la Tabla

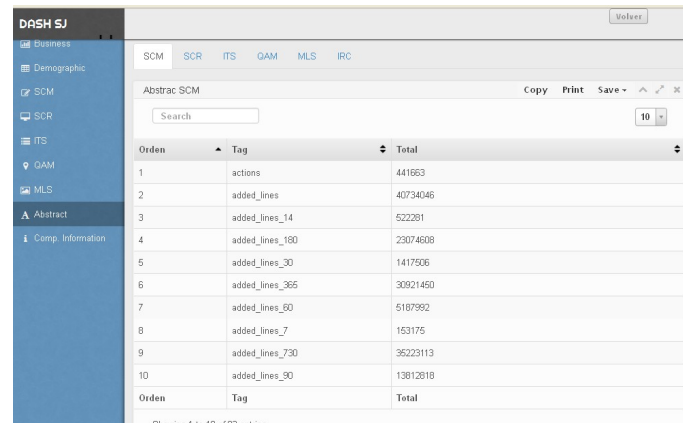
- **Source Code Repository:** Es un sistema de seguimiento, que actúa sobre un repositorio, es decir obtiene los cambios que se han producido en el repositorio por los diferentes desarrolladores.

Esto nos permite sumar un nuevo campo para comparar y obtener nuestros resultados. En esta pestaña obtenemos nuevos gráficos con diferentes parámetros a comparar, todos ellos referentes a un repositorio.

- **Issue Tracking System:** Un sistema de seguimiento de incidentes (denominado en inglés como issue tracking system, trouble ticket system o incident ticket system) es un paquete de software que administra y mantiene listas de incidentes, conforme son requeridos por una institución.

En esta pestaña podemos ver este control en forma de gráficas.

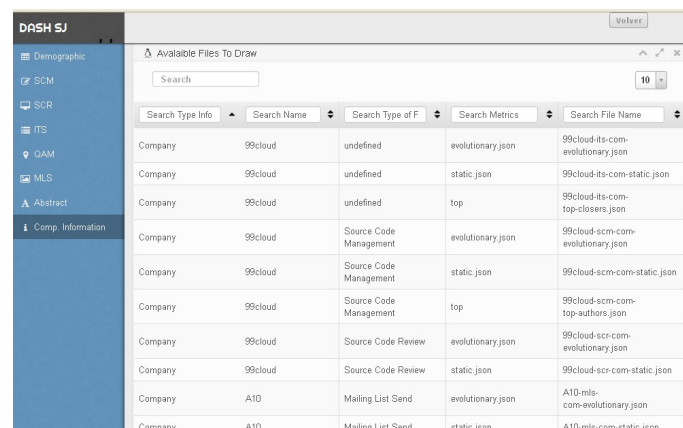
- **Question Answered Forums:** Este sistema de seguimiento, esta basado en los foros, ampliando nuestro sistema para sacar estadísticas.
- **Mailing Lists System:** Son los sistemas de correos que se utilizan entre el equipo de desarrolladores, con esta métrica, podemos visualizar todo lo referente al gestos de correos de los usuarios (enviados, recibidos, respondidos...).
- **Abstract:** Muestra información sobre todas las métricas que se han procesado, nos muestra en una tabla todos los datos analizados y en la gráfica nos muestra la representación de dichos parámetros.



Orden	Tag	Total
1	actions	441663
2	added_lines	40734046
3	added_lines_14	522281
4	added_lines_180	23074608
5	added_lines_30	1417506
6	added_lines_365	30921450
7	added_lines_60	5187992
8	added_lines_7	153175
9	added_lines_730	35223113
10	added_lines_90	13812818
Orden	Tag	Total

Figura 5.13: Pestaña Abstract

- **Company Information:** Nos muestra todos los ficheros de las empresa disponible de todas las métricas que dispones además permite acceder a dichos archivos, para descargarlos.



Search Type Info	Search Name	Search Type of F	Search Metrics	Search File Name
Company	99cloud	undefined	evolutionary.json	99cloud-its-com-evolutionary.json
Company	99cloud	undefined	static.json	99cloud-its-com-static.json
Company	99cloud	undefined	top	99cloud-its-com-top-closers.json
Company	99cloud	Source Code Management	evolutionary.json	99cloud-scm-com-evolutionary.json
Company	99cloud	Source Code Management	static.json	99cloud-scm-com-static.json
Company	99cloud	Source Code Management	top	99cloud-scm-com-top-authors.json
Company	99cloud	Source Code Review	evolutionary.json	99cloud-scr-com-evolutionary.json
Company	99cloud	Source Code Review	static.json	99cloud-scr-com-static.json
Company	A10	Mailing List Send	evolutionary.json	A10-mls-com-evolutionary.json
Company	A10	Mailing List Send	static.json	A10-mls-com-static.json

Figura 5.14: Pestaña Company Information

Capítulo 6

Conclusiones

El proyecto supone un primer paso por construir una herramienta potente, en primer lugar capaz de realizar una recogida de datos, mediante diferentes fuentes de ficheros, y segundo lugar supone el manejo y representación de esos datos.

Podemos decir que su realización se completa con éxito, ya que han sido alcanzados todos los objetivos propuestos en un principio.

Así, se ha logrado crear una herramienta genérica que es capaz de extraer datos desde fuentes de información disponibles públicamente, como pueden ser los ficheros json que son ofrecidos por la página de OpenStack.

Por otra parte, los datos resultantes de este proceso, son masajeados con el fin de conseguir el formato que necesitamos, para la representación. Con el objetivo de que el usuario pueda navegar entre las gráficas ganando más o menos profundidad en función de lo que necesite o el interés que tenga sobre lo que esté buscando.

Además, se ha alcanzado un alto grado de automatización en el proceso de recogida de información: mediante funciones genéricas que son capaces de coger la información de los ficheros sin importar el interior de los datos.

Con respecto a al ámbito personal me ha permitido evaluar numerosas tecnologías y realizar pruebas con ellas, esto me ha permitido ampliar mi campo de conocimiento.

Como es normal en esta vida, todo se puede mejorar y reflexionando un poco y haciendo una autocrítica encontramos algunos inconvenientes:

- Exportación de Datos: Este es un inconveniente a la hora de compartir el dashboard, ya que lo ideal sería crear una url y poderse la enviar a otro usuario y que pudiera ver

nuestro dashboard.

- Autenticación de Usuarios: Gestionar los usuarios con un sistema de login que permitiera tener mas información sobre el usuario que visita la aplicación, así como de guardar configuraciones personalizadas.
- Mejorar la apariencia: Potenciar la apariencia con una grid que fuera editable y que pudieramos colocar las gráficas como el usuario lo deseara.

Como colofón, hemos de recordar que se consiguió desarrollar el proyecto exclusivamente usando software libre, incluyendo documentación y elementos gráficos.

Por último también me gustaría destacar, que una vez concluido el proyecto, esto es una primera versión de un dashboard, que espero que futuros compañeros puedan seguir mejorando esta implementación, para que un futuro no muy lejano dispongamos de un completo dashboard.

6.1. Lecciones aprendidas

Han sido muchas las lecciones aportadas a través de la ejecución del proyecto, por lo que puede considerarse una experiencia formativa enriquecedora. Aquí se presentan algunas de ellas:

- La comprensión de la variedad de medios de colaboración existentes en el desarrollo del software libre y aprendizaje del uso de los estudiados: a partir de ahora no tendré reparos en informar sobre fallos detectados en un programa, siendo consciente de la importancia para el bien común que supone el hacerlo; o en darse de alta y participar en listas de correo.
- He obtenido una gran destreza en lenguaje JavaScript, en dicho lenguaje he trabajado con la carga dinámica de ficheros y con la programación de eventos.
- La creación de un proyecto de software de ciertas dimensiones siguiendo una metodología de desarrollo SCRUM, es decir ver nacer el proyecto desde una idea hasta ver concluido y consolidado.

- Profundizar en el conocimiento del lenguaje del HTML y CSS incluyendo las nuevas versiones de estas tecnologías.
- Implementar el manejo de la tecnología BootStrap.
- Se ha aprendido también a manejar \LaTeX , el sistema de procesamiento de documentos empleado en la realización de esta memoria y de otros tantos documentos en el futuro, con total seguridad.
- La utilización de bibliotecas de visualización gráfica como HighChart, de la cual he obtenido un gran manejo, y de otras como Chart.js, JsLate.

6.2. Conocimientos aplicados

Antes de enfretarme al proyecto me encontraba con una buena base de programación, aunque los lenguajes que iba a utilizar eran completamente nuevos para mí, lo importante es tener la cabeza bien amueblada en el mundo de la programación, dicho trabajo lo han conseguido los profesores a lo largo de la carrera y lo continúan haciendo mis compañeros de trabajo más experimentados de los cuales no dejo de aprender.

En el mundo de la programación no es necesario conocer todos los lenguajes para desarrollar una aplicación sino tener la base ya que como todos sabréis los lenguajes son muy parecidos aunque cada uno tiene sus propias peculiaridades y una sintaxis específica.

Mi situación previa al proyecto era la siguiente: una cierta experiencia programación Ada y C complementándolo con Python, CSS y HTML. Dichos conocimientos han sido adquiridos a lo largo de la carrera. Otros conocimientos no relacionados con la programación pero no por ello menos importantes son el compañerismo, esta facultad yo la resaltaría como muy importante en mi trayectoria, ya que como comente anteriormente a veces en la programación puedes quedarte bloqueado en un proceso y una visión nueva de otra persona puede despejarte el camino.

Tras el proyecto puedo decir que he ampliado mis conocimiento en JavaScript algo que era totalmente nuevo, he mejorado el diseño en hojas de estilo y además he tenido el gusto de poder conocer y trabajar con una tecnología en auge como es BootStrap.

Por otro lado me gustaría resaltar que este reto me ha hecho aprender de una forma autodidacta, esto permite perderle el miedo a lo desconocido y afrontar las cosas y saber donde

buscar.

Por concluir este capítulo de conocimiento, creo que será una base de cara a mi futuro ya que el proyecto se empieza con una gran incognita y un poco confuso con respecto a las tecnologías que usas ya que por norma general son desconocidas, pero puedo afirmar que cuando terminas tienes esa sabiduría de decir, me he buscado la vida y he aprendido por mi cuenta, sin olvidar la figura del tutor el cual inicia este camino y despeja tus dudas.

6.3. Trabajos futuros

Una vez finalizado el proyecto, se plantean varios campos de trabajo que supondrían una extensión de su funcionalidad y contribuirían a consolidarlo como la herramienta *estándar* en su campo.

Una posible mejora es ampliar las clases de fuentes de información a estudiar, creando nuevos módulos, los cuales traten los charts como un objeto.

Otra mejora sería permitir mayor libertad al usuario a la hora de elegir los parametros de representación, con esto conseguiríamos mayor personalización en el dashboard.

Otro aspecto en el que se puede trabajar es integrar un sistema de control de registros de usuarios, esto implica tener acceso a una base de datos. Con esto conseguimos que un usuario pueda personalizar y guardar su propio dashboard con los chart que más le interesan y configurarlo a su medida.

Resultaría también interesante exportar la información, esto consiste en disponer de un botón de guardado que el usuario al clicar sobre el se genere una url en la que se almacene el estado actual, con el fin de compartir con cualquier otro usuario.

Todos estos nuevos objetivos (y los que pueda plantear una tercera persona), podrían implementarse para potenciar la herramienta, que quedaría amparada por una licencia de software libre.

Bibliografía

[1] Marijn Haverbeke. *Eloquent JavaScript*. No Starch Press, 2011.

[2] W3 Schools

<http://www.w3schools.com/>

[3] OpenStack.

<https://www.openstack.org/>

[4] Gregorio Robles, Jesús M. González-Barahona, Rishab A. Ghosh. *GlueTheos: Automating the Retrieval and Analysis of Data from Publicly Available Software Repositories*.

<http://libresoft.dat.escet.urjc.es/html/downloads/gluetheos-icse.pdf>

[5] Manuel de Bootstrap.

https://librosweb.es/libro/bootstrap_3/

[6] Manuel de jQuery.

<http://librojquery.com/>

[7] Manuel de HTML5.

<http://www.tutosytips.com/aprende-html5-desde-0/>

[8] Manuel de CSS.

<http://www.manualdecss.com/section/manualcss/>

[9] Code Academy.

<http://www.codecademy.com>

[10] HighCharts.

<http://www.highcharts.com/>

[11] StackOverFlow.

<http://stackoverflow.com>

[12] Grupo de Sistemas y Comunicaciones - Universidad Rey Juan Carlos.

<http://gsyc.urjc.es>

[13] Web del proyecto Libre Software Engineering - GSYC.

<http://libresoft.urjc.es>

[14] OpenWeBinars.

<https://openwebinars.net/>

[15] Aprendiendo OpentaStack.

<http://aprendiendoopenstack.blogspot.com.es//>

[16] Metrics Grimoire Pages.

<https://metricsgrimoire.github.io//>