UNIVERSIDAD
REY JUAN CARLOS

GRADO EN INGENERÍA EN TECNOLOGÍA DE LA TELECOMUNICACIÓN

Curso Académico 2014/2015

Trabajo Fin de Grado

VISUALIZACIÓN DE DATOS DE DESARROLLO DE SOFTWARE

Autor : Quan Zhou

Tutor : Dr. Jesús María González-Barahona

Dedicado a

mi familia, amigos

II

Agradecimientos

Dar gracias a mi familia que siempre han estado apoyándome para que no baje la guardia en esos momentos de flaqueza. También a los amigos que siempre me han echo sentir que podía conseguirlo y a los compañeros de clase, especialmente a Jesús Alonso Barrionuevo y a Javier Sanjuan Orgaz que hemos estado los tres codo a codo sacándolo adelante, aunque tampoco olvido al resto de compañeros que han compartido conmigo todos estos años.

Y por último a los profesores de la Universidad Rey Juan Carlos por haberme formado durante todos estos años, muchos me han dado dolores de cabeza durante el curso pero han merecido la pena cuando ves que al final consigues superarlo.

Resumen

Este es un trabajo dedicado a la visualización de datos con diferentes bibliotecas de software libre de manera eficiente y dinámica. Consiste en representar los datos procedentes de ficheros JSON de producción de empresas, permitiendo el filtrado hasta el máximo detalle además de la interacción entre los distintos charts o gráficas que se representan.

Para este propósito, se ha trabajado fundamentalmente con el lenguaje de programación JavaScript y con las siquientes bibliotecas:

- JQuery para interactuar con los documentos HTML, manejar el árbol DOM, etc.
- Crossfilter para el filtrado de datos.
- DC para la visualización de los datos.
- Bootstrap para que funcione en diferentes plataformas(móviles, tabletas y ordenadores).

Aunque solo se mencione dc.js se ha realizado una exploración extensa de otras bibliotecas para finalmente comprobar que era la que mejor se adaptaba a nuestros objetivos.

En el panel de demografía una de las cosas más significativas en cuanto al funcionamiento aunque no tanto en el aspecto visual, es la tabla. Que contienen múltiples filtros de los datos, incluso más que los filtros que se pueden hacer en las propias gráficas. Ya que se pretende hacer que la tabla sea una especie de panel de control que se pueda tener todo sin necesidad de subir a las gráficas. De esta manera tenemos un dasboard más intuitivo al poder filtrar directamente en la tabla, que es donde se muestra toda la información detallada de cada entrada.

En el panel de commits las gráficas son ligeramente diferentes, al tratarse de un fichero JSON diferente al que se van a visualizar. No es tan completo como en el panel de demografía por la falta de tiempo aunque el tamaño de fichero JSON es mucho más grande.

Los ficheros JSON que se han utilizado para este proyecto están ubicados en OpenStack que a su vez usa Metrics Grimoire para la recogida de datos y la creación de esos ficheros JSON.

Summary

It is a work dedicated to display of data with diferents free software libraries

Índice general

Índice de figuras

Capítulo 1

Introducción

Para entender el proyecto, en este capítulo vamos a describir los problemas que hay en la actualidad en el ámbito de la visualización y filtrado de datos. Y ofrecer la mejor solución posible pero con el uso de software libre.

1.1. Descripción del problema

Toda empresa necesita una gran cantidad de datos, ya sea de las ventas, ingresos, progreso, productos, etc. Y que se necesitan visualizarlos para poder entenderlos, como por ejemplo, representar la evolución de la empresa, demografía, ganancias, pérdidas, etc. Como se puede observar si entramos en la página web de cualquier empresa, en alguna parte o directamente en la principal nos muestran dichas gráficas, pero sólo muestra una pequeña parte de los datos. Generalmente no muestran todos, ya que tampoco interesa a la empresa que determinados grupos sepa determinados datos.

También hay empresas que nos ofrecen directamente dashboards dando todo tipo de información sobre ella, con diversas representaciones para la dicha comprensión, como por ejemplo OpenStack `http://activity.openstack.org/dash/browser/`.

Pero en la mayoría de ellas no se pueden filtrar o hacen un filtrado muy pobre. Este proyecto se centrará justo en esa parte del filtrado, para que el usuario entienda mejor los datos y sacar información valiosa sobre ellas.

1.2. Objetivo principal

Una de las cosas que haremos en este proyecto es que el usuario pueda filtrar los datos de una manera sencilla, intuitiva, dinámica y eficientemente. Está hecho para todo tipo de usuarios, ya que no se necesita conocimiento previo para poder usarlo, aunque será enfocado para las empresas más que al púnlico en general.

Nuestro objetivo principal es que se pueda filtrar en profundidad, hasta el punto de que nos quede sólo un dato o ninguno si este no coincide con los parámetros que se han elegido o introducido. De esta manera tenemos un control mucho mayor de los datos y podremos así comprenderlos mucho mejor. Y otra de las cosas también importante es que el filtrado de datos sea rápido, es decir que el repintado de las gráficas sea lo fluido.

Todo esto usando herramientas de software libre con las bibliotecas de dc.js y crossfilter. Y alojado en GitHub.

1.3. Requisitos

Requisitos mínimos que se debe tener el proyecto es que sea rápido y dinámico, para que el filtrado de datos y al repintar las gráficas sean fluidas. El cliente (usuario) quiere ver los efectos del cambio lo más rápido posible, si es al instante mucho mejor, pero cuando usamos grandes cantidades de datos, es casi imposible que el efecto sea instantánea. Las bibliotecas que hacen esto posible, en nuestro proyecto principal son las siguientes:

- Dc.js: Al estar perfectamente incluido la biblioteca `crossfilter` hace que sea muy útil para este proyecto.
- Crossfilter: Filtrado de dato muy eficiente y rápido.

Y respecto a los requisitos del proyecto son las siguientes:

- Diversas gráficas: cada uno con diferente información.
- Tabla: donde se muestran todos los datos.
- Filtrado en la tabla: para que el usuario tenga mayor rapidez en el filtrado, seleccionando directamente sobre ella.

- Buscador: si el usuario sabe directamente a quien buscar, sólo es necesario introducir el nombre del usuario del que se quiere.
- Selector de fechas: elección de las fechas iniciales y finales de los datos que se quiera mostrar. Por dos métodos (datePicker y slider).

1.4. Disponibilidad del software

El proyecto estará alojado en la herramienta de control de versiones `GitHub` y algunos de los demos en la fase de exploración que se han hecho también. Que lo detallaremos en el capítulo cuatro (desarrollo e implementación).

La página web inicial donde se puede encontrar el código fuente del proyecto, los dos paneles, y la memoria en pdf.

`http://zhquan.github.io/TFG/`

El panel de demografía que es el proyecto principal.

`http://zhquan.github.io/TFG/demografia/`

El panel de commit que es el proyecto secundario.

`http://zhquan.github.io/TFG/commit/`

Donde están alojadas los demos de highcharts y charts.js.

`http://zhquan.github.io/qz-dashboard/`

El prototipo3 está alojado la demo de la biblioteca NVD3.js y el prototipo4 está alojado la demo de la biblioteca C3.js.

`https://github.com/zhquan/TFG/Prototipo3`

`https://github.com/zhquan/TFG/Prototipo4`

El resto de prototipo2 constituyen a trabajos futuros, para dar una funcionalidad diferente, pero al final se intentará con esos prototipos la 2-2 y 2-3 se puedan asemejar al proyecto principal, es decir, al panel de demografia.

https://github.com/zhquan/TFG/Prototipo2-2

https://github.com/zhquan/TFG/Prototipo2-3

Capítulo 2

Contexto del proyecto

2.1. Ingeniería del 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.

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 diponga de repositorios de versiones 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 ing del soft 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. Deberá procurarse que las herramientas empleadas para ello estén 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 sí 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 este 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 desarrollCadores, 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 colaboró y horas empleadas— podría establecerse una previsión de costes y recursos para futuros proyectos de software libre.

2.2. Grimoire Dashboard

Aunque no usamos directamente `Grimoire`, pero es interesante mencionar lo que es, ya que los ficheros que se van ha usar en este proyecto están hechos por esta herramienta, que es la encargada de extraer los datos y convertirlos en fichetos JSON.

2.2.1. Metrics Grimoire

MetricsGrimoire 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 error de informes), listas de correo, etc. Los datos y metadatos sobre el proceso de desarrollo de software se recogen de los repositorios (información sobre las confirmaciones, gestión de entradas, la comunicación en listas de correo, etc.), y luego se organizan y se almacenan en bases de datos SQL que luego pueden ser extraídos por los patrones o resúmenes de actividades específicos.

2.2.2. VizGrimoire

VizGrimoire es un conjunto de herramientas y framework para 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 de desarrollo software, pero se trata de un proyecto abierto a la comunidad.

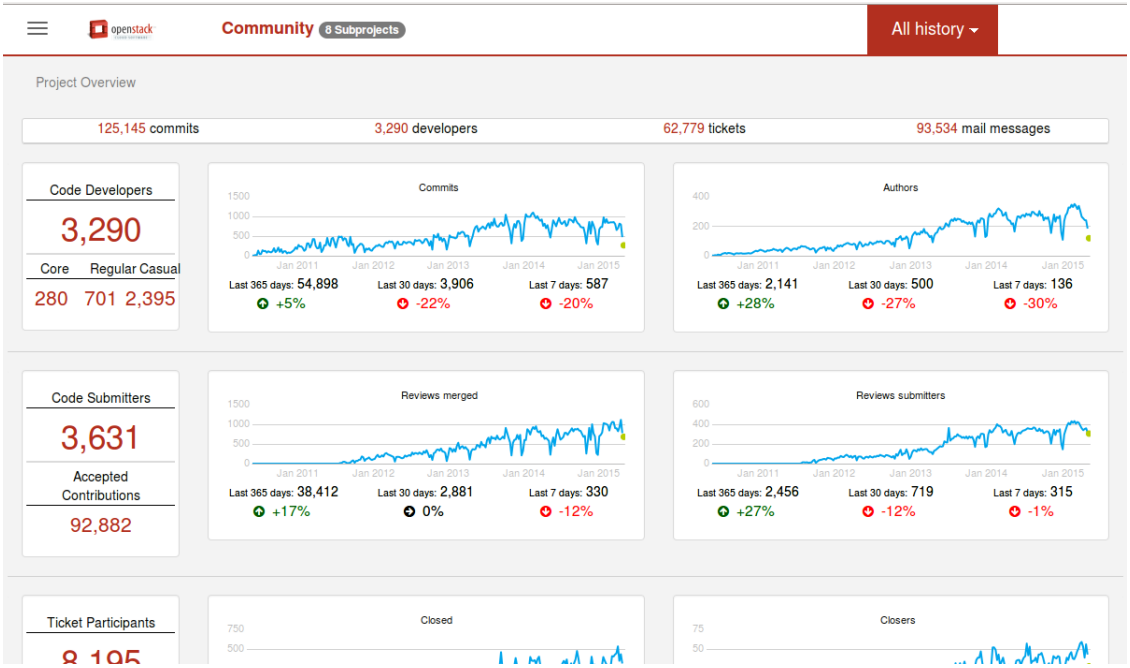


Figura 2.1: OpenStack

2.3. Visualización de datos

2.3.1. OpenStack

OpenStack es un sistema operativo en la nube que controla grandes cantidades de computación, almacenamiento, redes y recursos en un centro de datos, todo ello gestionado a través de un panel de control que le da a los administradores controlar mientras fortalecen a sus usuarios a los recursos de provisión a través de una interfaz web.

En la figura 2.1 veremos un ejemplo de las visualizaciones de OpenStack.

2.3.2. Stackalytics

Stackalytics es un servicio que recoge y procesa los datos de actividad de desarrollo tales como cometa, líneas de código cambiado, y las revisiones de código, planos y permite visualizar en un panel de control web conveniente. El tablero de instrumentos Stackalytics permite ver los datos de proyecto, empresa, colaborador, y otros factores.

El proyecto Stackalytics está trabajando para proporcionar estadísticas transparentes y significativas con respecto a las contribuciones y proyectos relacionados con OpenStack. Pero

¿qué significa transparente y significativa? La transparencia es importante para que la comunidad pueda tener confianza en que todos los cálculos son correctos y justos. Así que transparente significa que cualquiera puede corroborar los métodos de cálculo que utiliza Stackalytics. Mientras tanto, los resultados deben ser significativa para ser útil. Significativa significa que cualquier persona podrá presentar una corrección que ajusta la influencia de los datos estadísticos apropiados. Por ejemplo, el código autogenerado, renombramiento de masas, refactorización automática, los archivos de configuración generados automáticamente, y así sucesivamente pueden inflar artificialmente diversas estadísticas. Stackalytics hace posible evitar estos problemas a medida que se descubren.

Las fuentes de datos primarios para Stackalytics son los repositorios Git OpenStack y la historia crítica Gerrit.

2.3.3. Open Hub

El Black Duck Open Hub (anteriormente Ohloh.net) es una comunidad y directorio público de software libre y código abierto en red (FOSS), afreciendo análisis y busqueda de servicios para descubrir, evaluar, seguimiento y comparar códigos y proyectos de código abierto. Open Hub Code Search es de código libre, el motor de búsqueda de indexación sobre 21 mil millones de líneas de código fuente abierto de proyectos sobre el Black Duck Open Hub.

El Open Hub es editable por todos, como una wiki. Todos son bienvenidos a unirse, añadir nuevos proyectos, y hacer las correcciones a las páginas de los proyectos existentes. Esta opinión pública ayuda a que el Black Duck Open Hub sea uno de los más grandes, más precisa y actualizar FOSS directorios de software libre disponible. Anima a los contribuyentes a unirse al Open Hub y reclamar sus commit en proyectos existentes y añadir proyectos que aún no están en el sitio. De esta manera, los usuarios de Open Hub pueden armar un perfil completo de todas sus contribuciones de código de software libre.

El Open Hub no es una fragua - que no aloja proyectos y código. El Open Hub es un directorio y de la comunidad, que ofrece análisis y servicios de búsqueda y herramientas. Mediante la conexión de proyectar repositorios de código fuente, el análisis de la historia y en curso de actualizaciones del código, y atribuir esos cambios a contribuyentes específicos, el Black Duck Open Hub puede proporcionar informes sobre la composición y la actividad de bases de códigos del proyecto y agregar estos datos para realizar el seguimiento del cambio demográfico del

mundo de software libre.

Para más información visite la página oficial `https://www.openhub.net/`

2.3.4. Many Eyes

IBM Many Eyes, una comunidad web que conecta a expertos en visualización, profesionales, académicos y aficionados, que ofrece esta tecnología y experiencia, junto con la forma de compartir y aprender de los demás.

El atractivo de la página web Many Eyes es que democratiza la visualización. No se necesita programación o conocimientos técnicos, por lo que casi todo el mundo tiene el poder de crear visualizaciones. Sólo tiene que seguir tres pasos:

- Subir el conjunto de datos pública. Las visualizaciones creadas en el sitio web Muchos trabajan Ojos de formatos de datos simples, como una hoja de cálculo o archivos de texto.
- Elegir entre una amplia variedad de visualizaciones o uno recomendado por muchos ojos.
- Dar riendas sueltas a su visión al compartir su visualización a través de Internet. Puede incrustar una visualización en tu blog o compartirlo fácilmente en Facebook y Twitter con un solo clic.

En la figura 2.2 veremos un ejemplo de las visualizaciones de Many Eyes.

2.3.5. Freeboard

Freeboard es una biblioteca gráfica de JavaScript hecha especialmente para gráficas en tiempo real, que combina a su vez diferentes bibliotecas. Tiene una interfaz de usuario bastante conseguida. Una de las ideas de esta biblioteca es el Internet de las cosas, que quiere decir, que todo está conectado a la red como posible ejemplo, el usando REST con esta biblioteca sería bastante útil, se tendría constancia de cada cosa con una url única y todas ellas reflejadas en un mismo panel compuestos de widgets para comprender fácilmente los datos que se tiene.

Es una biblioteca áltamente flixible, dinámica y atractiva para crear dashboards. Los paneles están compuestos de widgets, de esta manera se pueden configurar fácilmente sin interferir al resto de gráficas. Cada widgets se puede configurar como se quiera, también se puede mover los diferentes widgets directamente para componer el dashboard al gusto de cada uno.

Figura 2.2: Many Eyes

En la figura 2.3 veremos un ejemplo de las visualizaciones de Freeboard.

2.3.6. Gephi

Gephi es una visualización interactiva y la plataforma de exploración de todo tipo de redes y sistemas complejos, dinámicos y gráficos jerárquicos.

Gephi es un software de código abierto para el gráfico y el análisis de redes. Utiliza un motor de render 3D para mostrar grandes redes en tiempo real y para acelerar la exploración. Una arquitectura flexible y multi-tarea trae nuevas posibilidades para trabajar con conjuntos de datos complejos y producir resultados visuales valiosos. Se presenta varias características clave de Gephi en el contexto de la exploración y la interpretación de las redes interactivas. Proporciona un acceso fácil y amplio a los datos de red y permite espacializar, filtrar, navegar, manipular y agrupar. Por último, mediante la presentación de las características dinámicas de Gephi, destacamos los aspectos clave de visualización de la red dinámica.

En la figura 2.4 veremos un ejemplo de las visualizaciones de Gephi.

Figura 2.3: Freeboard

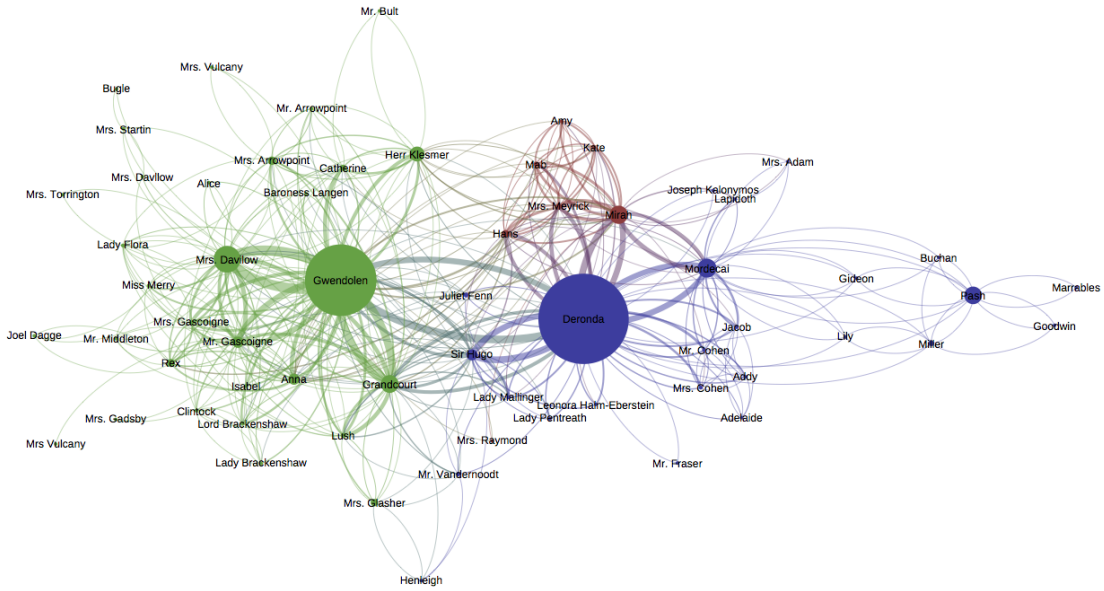


Figura 2.4: Gephi

2.3.7. GGobi

GGobi es un programa de visualización de código abierto para la exploración de los datos de alta dimensión. Proporciona gráficos muy dinámicos e interactivos tales como excursiones, así como gráficos conocidos, como el diagrama de dispersión, barras y gráficas de coordenadas paralelas. Las gráficas son interactivos y vinculado con el cepillado (brushing) y la identificación.

Fue desarrollado inicialmente para mirar matrices de datos. Comúnmente tenemos datos que toma forma de matriz, las filas son las muestras / casos / sujetos, y las columnas son los valores / variables que hemos medido en ellos. La dimensión de un conjunto de datos es el número de variables, y de alta dimensional tiene más de dos variables. Cada variable se asigna a un eje en un sistema de coordenadas euclidiana, y el número de ejes es la dimensión.

Inicialmente surgió de los propios intereses en la búsqueda de maneras de dibujar imágenes de datos de alta dimensión. Ha evolucionado para manejar otros tipos de datos, incluidos los valores perdidos y datos de red / gráfico. Había usuarios en Bellcore, y luego en el AT & T interesados en las redes de telecomunicaciones que impulsaron los métodos de la red. También estan ahora estudiando métodos para tablas relacionadas, que atienden a las mediciones longitudinales. Esto se debe a los intereses generales, consultoría de trabajo y la enseñanza.

En la figura 2.5 veremos un ejemplo de las visualizaciones de GGobi.

2.3.8. Tulip

Tulip es un framework de visualización de la información dedicada al análisis y visualización de datos relacionales. Tulip tiene como objetivo proporcionar al desarrollador con una biblioteca completa, apoyando el diseño de aplicaciones de visualización de información interactivos para datos relacionales que se pueden adaptar a los problemas que él o ella está abordando.

Escrito en C++ el framework permite el desarrollo de algoritmos, codificaciones visuales, técnicas de interacción, modelos de datos y visualizaciones de dominio específico. Uno de los objetivo de Tulip es facilitar la reutilización de componentes y permite a los desarrolladores centrarse en la programación de su aplicación. Esta línea de desarrollo hace que el framework sea eficiente para la creación de prototipos de investigación, así como el desarrollo de aplicaciones

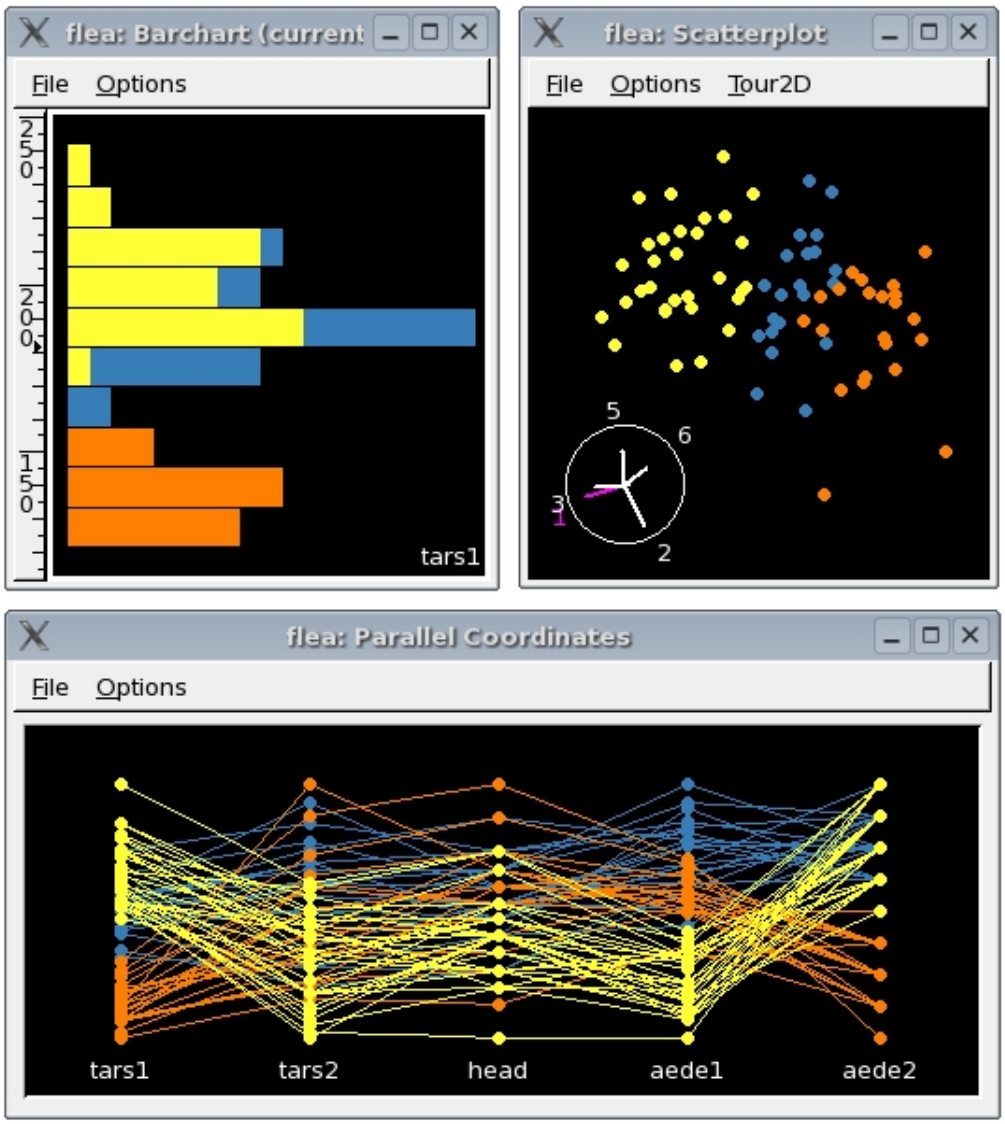


Figura 2.5: GGobi

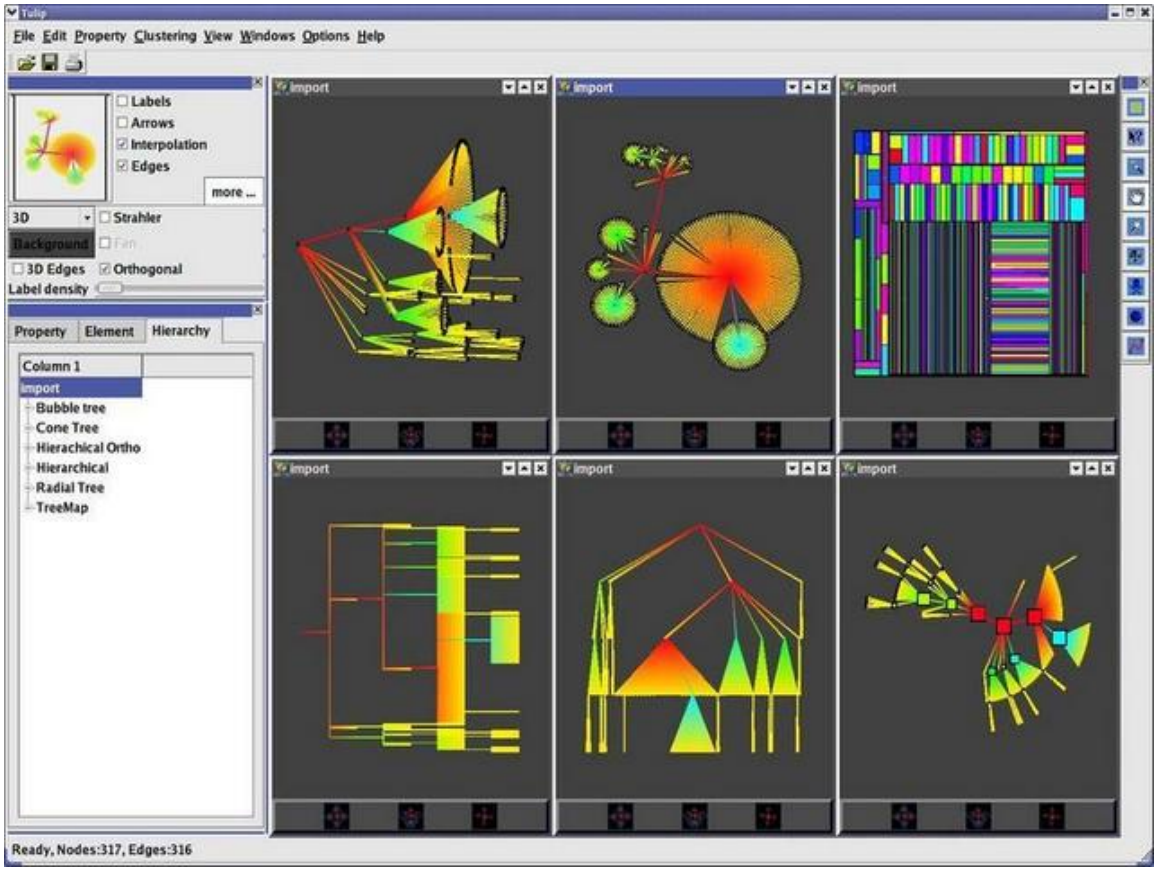


Figura 2.6: Tulip

de usuario final.

En la figura 2.6 veremos un ejemplo de las visualizaciones de Tulip.

Capítulo 3

Tecnologías utilizadas

En este capítulo mencionaremos las tecnologías que hemos usado para desarrollar el proyecto con una breve explicación.

3.1. JavaScript

JavaScript (abreviado comúnmente ”JS”) es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas4 aunque existe una forma de JavaScript del lado del servidor (Server-side JavaScript o SSJS). Su uso en aplicaciones externas a la web, por ejemplo en documentos PDF, aplicaciones de escritorio (mayoritariamente widgets) es también significativo.

JavaScript se diseñó con una sintaxis similar al C, aunque adopta nombres y convenciones del lenguaje de programación Java. Sin embargo Java y JavaScript no están relacionados y tienen semánticas y propósitos diferentes.

Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM).

Tradicionalmente se venía utilizando en páginas web HTML para realizar operaciones y únicamente en el marco de la aplicación cliente, sin acceso a funciones del servidor. JavaScript

se interpreta en el agente de usuario, al mismo tiempo que las sentencias van descargándose junto con el código HTML.

Una cuarta edición está en desarrollo e incluirá nuevas características tales como paquetes, espacio de nombres y definición explícita de clases

3.2. HTML

HTML, siglas de HyperText Markup Language (lenguaje de marcas de hipertexto), 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 lenguaje HTML basa su filosofía de desarrollo en la referenciación. Para añadir un elemento externo a la página (imagen, vídeo, script, entre otros.), este no se incrusta directamente en el código de la página, sino que se hace una referencia a la ubicación de dicho elemento mediante texto. De este modo, la página web contiene sólo texto mientras que recae en el navegador web (interpretador del código) la tarea de unir todos los elementos y visualizar la página final. Al ser un estándar, HTML busca ser un lenguaje que permita que cualquier página web escrita en una determinada versión, pueda ser interpretada de la misma forma (estándar) por cualquier navegador web actualizado.

3.2.1. HTML5

HTML5 es la quinta revisión importante del lenguaje básico de la World Wide Web, HTML. HTML5 especifica dos variantes de sintaxis para HTML: HTML (text/html), la variante conocida como HTML5 y una variante XHTML conocida como sintaxis XHTML5 que deberá ser servida como XML. Esta es la primera vez que HTML y XHTML se han desarrollado en paralelo.

Al no ser reconocido en viejas versiones de navegadores por sus nuevas etiquetas, se recomienda al usuario común actualizar a la versión más nueva, para poder disfrutar de todo el

potencial que provee HTML5.

El desarrollo de este lenguaje de marcado es regulado por el Consorcio W3C.

HTML5 establece una serie de nuevos elementos y atributos. Algunos de ellos son técnicamente similares a las etiquetas $< div >$ y $< span >$, pero tienen un significado semántico, como por ejemplo $< nav >$ (bloque de navegación del sitio web) y $< footer >$. Otros elementos proporcionan nuevas funcionalidades a través de una interfaz estandarizada, como los siguientes elementos:

- $< audio >$ y $< video >$.
- Mejora el elemento $< canvas >$, capaz de renderizar elementos 3D.
- Introduce el elemento $< svg >$ para contener gráficos SVG.
- âAlmacenamiento localâ (Local Storage) de datos en el lado del cliente pero debemos ver si el navegador en el que estamos trabajando es compatible. A diferencia de las cookies esta no consume ancho de banda ya que se almacena en el lado del cliente y no en el lado del servidor.
- La nueva API de geolocalización, que nos permite conocer la ubicación geográfica del usuario, siempre y cuando esté usando un navegador que la tenga implementada y que el usuario dé su permiso.

Lo que más hemos utilizado de estas nuevas características de HTML5 es el elemto $< svg >$, ya que, la biblioteca dc.js se basa en d3 y que a su vez usa usa para generar información gráfica en formato SVG.

3.3. CSS

Hoja de estilo en cascada o CSS (siglas en inglés de cascading style sheets) es un lenguaje usado para definir y crear la presentación de un documento estructurado escrito en HTML o XML2 (y por extensión en XHTML).

La información de estilo puede ser definida en un documento separado o en el mismo documento HTML. En este último caso podrían definirse estilos generales en la cabecera del documento o en cada etiqueta particular mediante el atributo «style».

3.3.1. CSS1

CSS1 es la primera especificación oficial de CSS.

Algunas de las funcionalidades que ofrece son:

- Propiedades de las Fuente, como tipo, tamaño, énfasis...
- Color de texto, fondos, bordes u otros elementos.
- Atributos del texto, como espaciado entre palabras, letras, líneas, etcétera.
- Alineación de textos, imágenes, tablas u otros.
- Propiedades de caja, como margen, borde, relleno o espaciado.
- Propiedades de identificación y presentación de listas.

3.3.2. CSS2

Como ampliaciõn de CSS1, se ofrecieron, entre otras:

- Las funcionalidades propias de las capas ¡div¿como de posicionamiento relativo/absoluto/fijo, niveles (z-index), etcẽtera.
- Soporte para las hojas de estilo auditivas.
- Texto bidireccional, sombras, etcẽtera.

3.3.3. CSS3

A diferencia de CSS2, que fue una gran especificaciõn que definía varias funcionalidades, CSS3 está dividida en varios documentos separados, llamados ”módulos”.

Cada módulo añade nuevas funcionalidades a las definidas en CSS2, de manera que se preservan las anteriores para mantener la compatibilidad.

Vamos a ver algunos de los nuevos elementos de CSS3:

- Bordes: border-color, border-image, border-radius, box-shadow.
- Backgrounds: background-origin, background-clip, background-size, layering multiple background images.

- Color: HSL colors, HSLA colors, RGBA colors opacity.
- Texto: text-shadow, text-overflow.
- Interface: box-sizing, resize.

3.4. Bootstrap

Bootstrap son plantillas echas en base de HTML5, CSS3, JQuery en javaScritp que nos ofrece una apariencia bastante atractiva y que a su vez es compatible con diversas plataformas (móviles, tabletas y ordenadores).

Tienen diversas plantillas orientadas a determinados tipos de páginas o aplicaciones de manera orientativa. Es bastante flexible, ya que se pueden cambiar de aspecto como se quiera para poder ser personalizado al gusto de cada uno. Aunque en ese aspecto no es su punto fuerte ya que existen en la red un montón de plantillas echas con HTML5, CSS3 igual o mejor que este, pero al ser una plantilla altamente adaptado para diferentes plataformas, hace que sea mucho más atractivo. Antes de mostrar la página, primero reconoce el tipo de plataforma que se está accediendo en ese momento y dependiendo de cada una de ellas nos aparecerá de una forma u otra pero con una apariencia muy parecida, con algunos detalles necesarios para su buen funcionamiento. Esto hace que bootstrap sea usado por tantas personas.

Veamos algunos ejemplos de plantilla de esta biblioteca en las figuras 3.1 y 3.2.

3.5. JQuery

jQuery es una biblioteca de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. Fue presentada el 14 de enero de 2006 en el BarCamp NYC. jQuery es la biblioteca de JavaScript más utilizada.

jQuery es software libre y de código abierto, posee un doble licenciamiento bajo la Licencia MIT y la Licencia Pública General de GNU v2, permitiendo su uso en proyectos libres y privados. jQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en

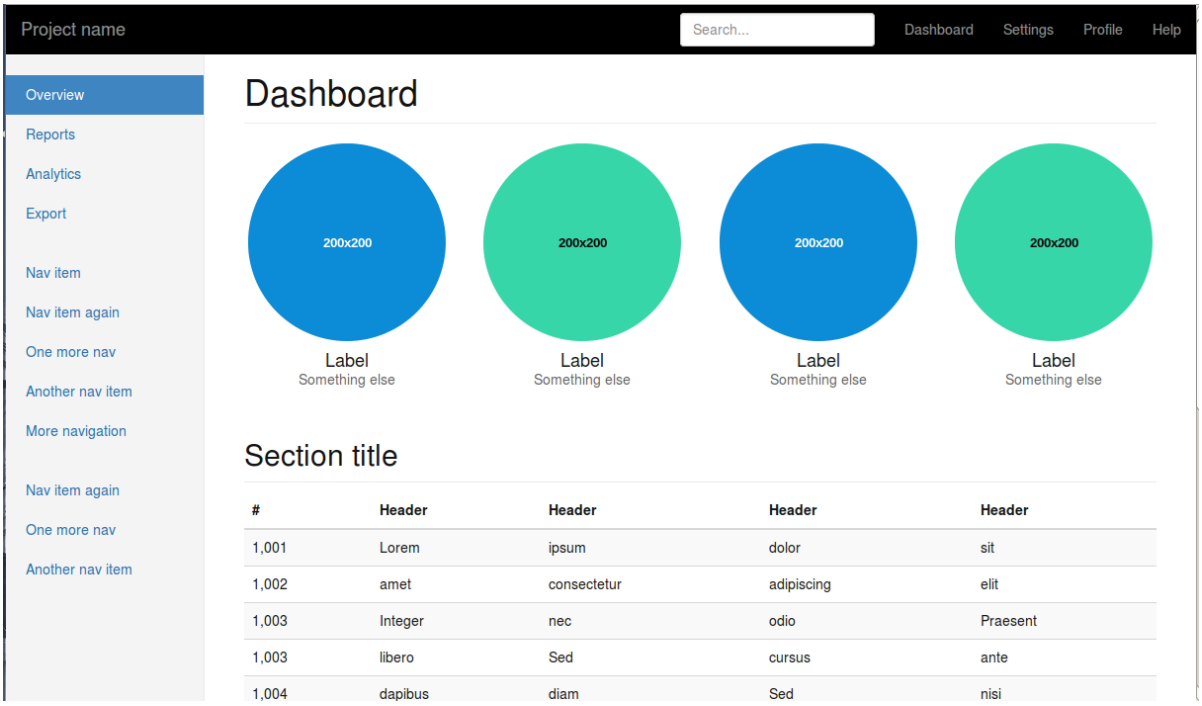


Figura 3.1: Bootstrap Dashboard

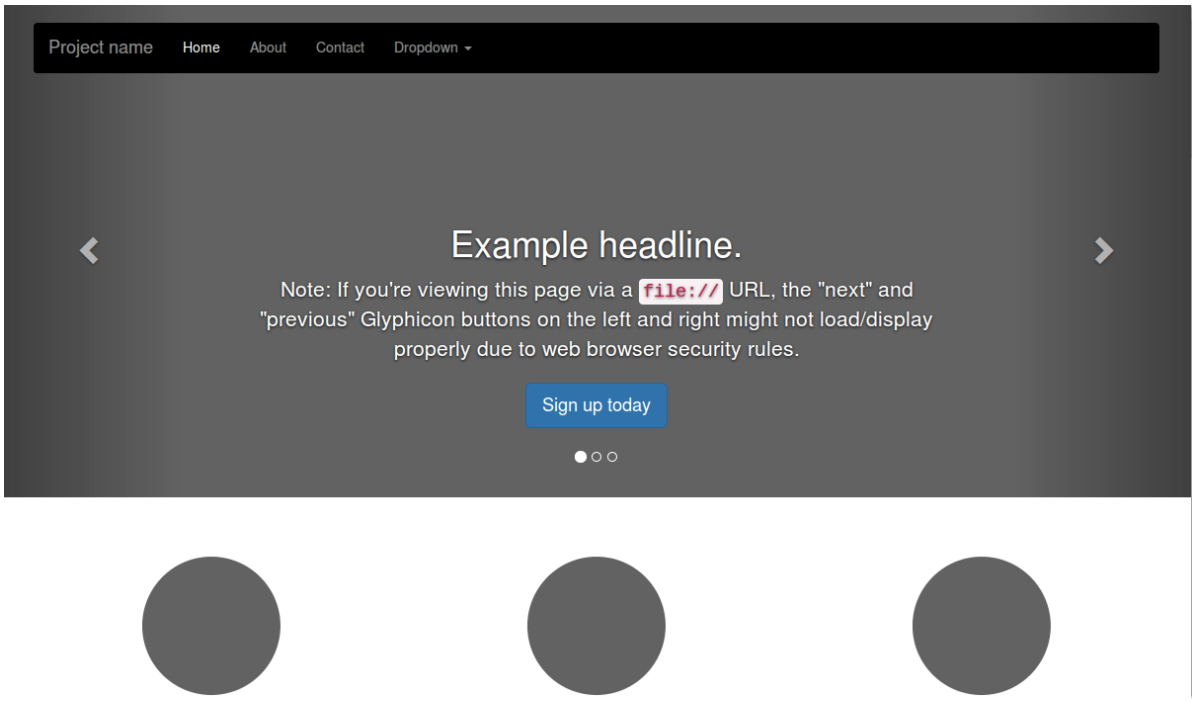


Figura 3.2: Bootstrap Carousel

JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.

3.6. SVG

Gráficos Escalables en Formato Vectorial (Scalable Vector Graphics) es un formato que usa texto para definir imágenes. Cada imagen SVG se define con código similar al de HTML. El código SVG se puede incluir directamente dentro de cualquier documento de HTML. Todos los navegadores leen SVG, con excepción de Internet Explorer 8 y versiones anteriores. SVG está basado en XML, por consiguiente notará que los elementos que no tienen etiquetas de cierre - se auto-cierran. Por ejemplo:

```
<element></element>   <!-- Usa una etiqueta de cierra -->
<element/>            <!-- Se auto-cierra -->
```

Antes de poder dibujar, se debe crear un elemento SVG (es conceptualmente similar elemento canvas de HTML). Por lo menos, se deben especificar los valores de ancho (width) y altura (height). Si éstos no se especifican, SVG utilizará todo el espacio que pueda dentro dentro del elemento que lo contiene. Por ejemplo:

```
<svg width="500" height="50">
</svg>
```

3.7. D3.JS

D3.js (o D3 para Data-Driven Documents) es una biblioteca gráfica de javascript que permite visualizar datos numéricos bajo una forma gráfica y dinámica. Se trata de una herramienta importante, conforme a las normas W3C que utiliza las tecnologías habituales SVG, Javascript y CSS para la visualización de datos.

Contrárіamente a otras bibliotecas, esta permite un control más amplio del resultado visual final.

En este caso nos encontramos con una herramienta capaz de ofrecer visualizaciones interactivas online muy avanzadas con complejos conjuntos de datos. Se trata de una librería de

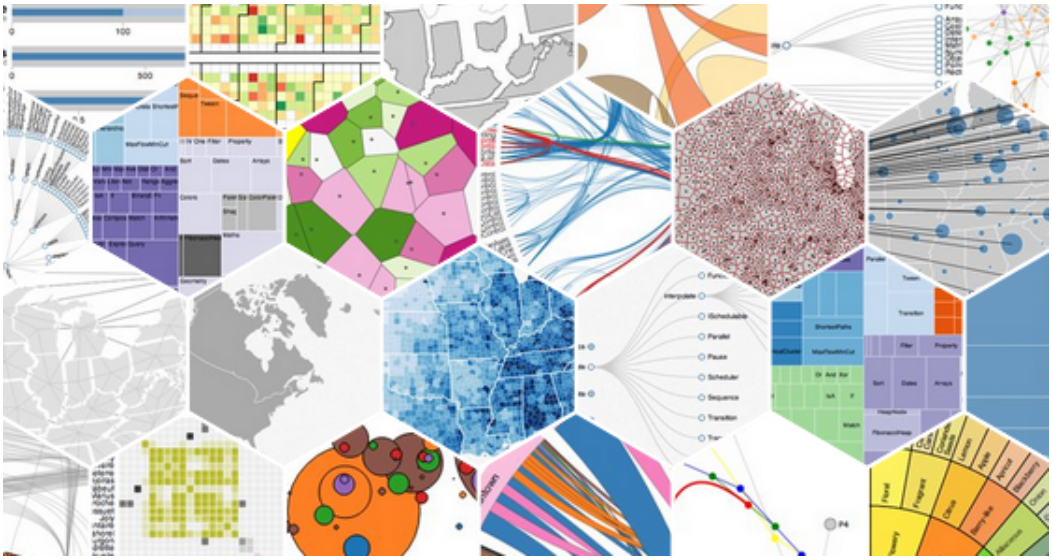


Figura 3.3: D3

Javascript que nos da la posibilidad de crear unos diagramas bastante impresionantes y gráficos a partir de una amplia variedad de fuentes de datos. Es de código abierto, y utiliza los estándares web, hecho que la convierte en muy accesible. El inconveniente que presenta D3 es el hecho de ser tan complejo que se necesita conocer programación y su lenguaje en concreto. Por tanto, su utilización no es tan sencilla como en otras.

EL resto de bibliotecas que siguen, serán hijas de las descritas anteriormente, ya que dependen de las mismas para su desarrollo y funcionamiento. De este modo D3 amplía sus usos en numerosas nuevas bibliotecas. En otras palabras, son capas o niveles superiores basadas en la misma D3. Como veremos en la figura 3.1 el amplio abanico de diferentes tipos de gráficas que tiene esta biblioteca.

Es factible desplegar gráficas con etiquetas $< div >$ y otros elementos nativos de HTML, este método no es el ideal y está sujeto a las inconsistencias que existen entre los diferentes navegadores. El uso de SVG es bastante más confiable, consistente visualmente y más rápido.

3.8. Crossfilter

Crossfilter es una biblioteca JavaScript para explorar grandes conjuntos de datos multivariantes en el navegador. Crossfilter es extremadamente rápido (¡30ms) interacción con vistas coordinadas, incluso con los conjuntos de datos que contiene un millón o más de registros.

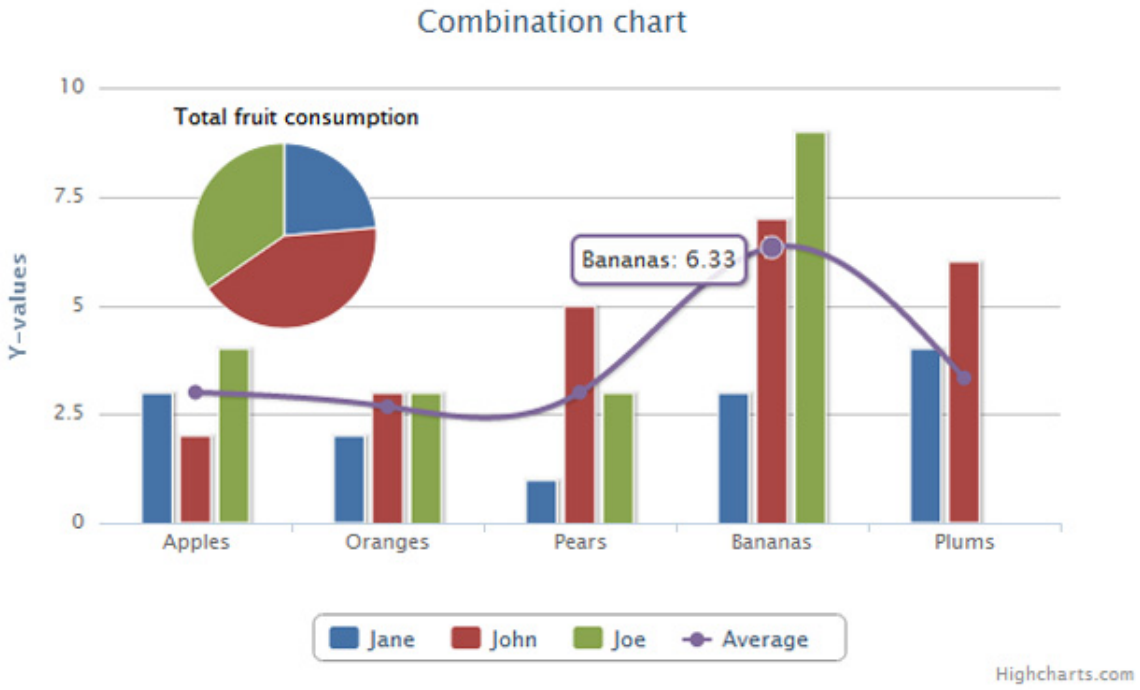


Figura 3.4: Highcharts

Como la mayoría de las interacciones solamente implican una sola dimensión, y luego se hacen sólo pequeños ajustes a los valores de filtro, el filtrado y la reducción gradual es significativamente más rápido que empezar desde cero. Crossfilter utiliza índices ordenados para que esto sea posible, aumentando drásticamente el desempeño de histogramas en vivo. Para más detalles sobre cómo funciona Crossfilter, consulte la referencia de API.

3.9. Highcharts

Highcharts es una biblioteca gráfica escrita en JavaScript, que ofrece una manera fácil de añadir gráficas interactiva para una página web o aplicación web. Highcharts actualmente soporta mucho tipos de gráficas como por ejemplo, linea, área, burbuja, barra, tarta, etc. También es muy f[a]cil combinar diferentes tipos de gr[a]ficas. Tiene una opción de descargar la gráfica en formato PNG, JPG, PDF o SVG, que esta activado por defecto en la misma gráfica.

Una de las cosas muy positivas de esta biblioteca es que tiene una API muy completa, con mucha información y ejemplos. En la figura 3.2 se visualiza un ejemplo de la highcharts

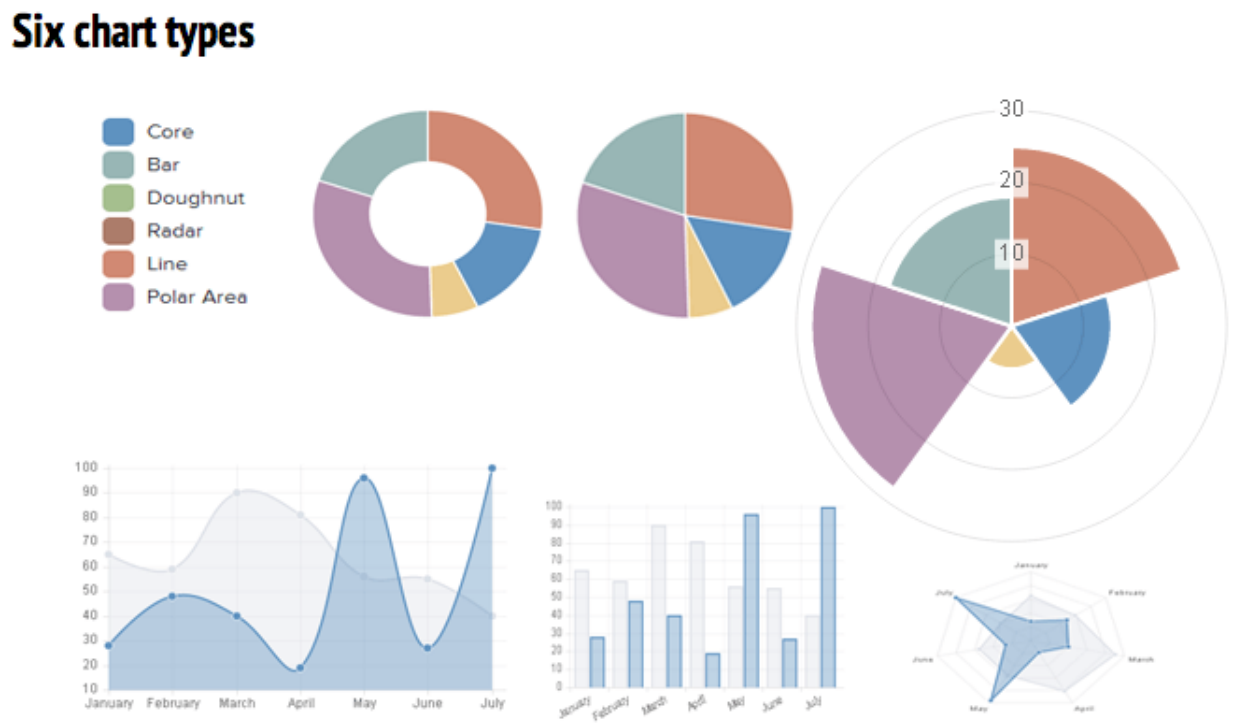


Figura 3.5: Chart JS

3.10. Chart.js

Chart.js es una biblioteca gráfica de JavaScript que usa a su vez d3.js como la mayoría de las bibliotecas gráficas. Tiene una manera bastante sencilla para dibujar las gráficas, ya que, es muy fácil de masajear los datos a como lo quiere esta biblioteca, pero tiene una gran inconveniencia, que es la de introducir leyenda, que no lo tiene por defecto y para ello tienes que crearte un $< div >$ aparte con la leyenda, aunque en la última actualización parece ser que si lo hay. Y tampoco es danámico en el sentido de que no se puede mezclar diferentes tipos, solo algunos predeterminados. Para el uso de nuestro proyecto se nos queda bastante corto.en la Figura 3.3 se ve visualiza con la última actualización aunque sólo nos muestra la leyenda en el tipo PIE.

3.11. NVD3

NVD3 es mantenido actualmente por un equipo de ingenieros de software en Novus Partners. La tecnología de gráficos se utiliza para proporcionar análisis eficaces a los clientes en la industria financiera.

Código base de NVD3 está fuertemente inspirado por el trabajo de Mike Bostock. En particular, su artículo ”Hacia Gráficas reutilizables”sirve como de guía.

NVD3 es una biblioteca gráfica de JavaScript muy flexible para añadir dinámicamente datos una vez que la gráfica este dibujada. Como en muchas de las bibliotecas de visualización y esta también, está echa sobre la biblioteca d3.js

3.12. C3

C3 hace que sea fácil de generar gráficos basados en D3 envolviendo el código necesario para construir todos los gráficos. No necesitamos escribir código D3. También da algunas clases a cada elemento cuando se genera, por lo que puede definir un estilo personalizado de la clase y es posible extender la estructura directamente por D3. Y proporciona una variedad de API y devoluciones de llamada para acceder al estado de la gráfica. Mediante el uso de ellos, se puede actualizar la gráfica, incluso después de que se renderice.

C3 es una biblioteca gráfica de JavaScript muy flexible para añadir dinámicamente datos una vez que la gráfica este dibujada, pero aparte de añadir datos también puede cambiar de tipo de gráficas una vez pintado, como por ejemplo de barra a linea o de cualquier otra forma de representación que tenga la biblioteca.

3.13. DC.js

Dc.js es una biblioteca gráfica de JavaScript que incluye nativamente a crossfilter que permite una exploración altamente eficiente a una base de datos grandes y multidimensional. Aprovecha el motor d3 para hacer gráficos en css formato svg. Gráficos utilizando en dc.js son naturalmente basadas en datos y reactivos, por lo tanto proporciona información instantánea sobre la interacción del usuario. El objetivo principal de este proyecto es proporcionar una potente biblioteca javascript fácil que puede ser utilizado para realizar la visualización y análisis de datos en el navegador , así como en el dispositivo móvil.

3.14. GitHub

Github es una plataforma para alojar poryecto utilizando el sistema de control de versiones Git. Donde está alojado este proyecto.

`http://zhquan.github.io/TFG/Prototipo2`

`https://github.com/zhquan/TFG/tree/master/Prototipo2`

Una herramienta verdaderamente útil para desarrolladores. Puedes observar como evolucionan los proyectos, las nuevas versiones, volver a las versiones anteriores si es necesario. Muchas de las bibliotecas que se han usado para hacer este proyecto también están subidos sus códigos fuentes a `http://github.com`

Todos los prototipos que se han creado están subidos en GitHub. Pero si queremos que GitHub nos la sirva tendremos que crear una rama llamada *gh-pages*, para que nos pueda mostrar la página web que hemos desarrollado. De esta manera podemos ver nuestro proyectos en la red con la url que te proporciona GitHub segun tu nombre de usuario y en el repositorio en el que se encuentre *(nombre de usuario).github.io/(nombre de repositorio)/*.

Capítulo 4

Desarrollo e implementación

En este capítulo explicaremos como se ha hecho el proyecto mediante iteraciones para que sea más fácil de entender. Se clasificarán según las reuniones que se hayan tenido con el cliente (tutor del proyecto).

Se procederá a utilizar el método SCRUM. SCRUM es una metodología ágil y flexible para gestionar el desarrollo de software, cuyo principal objetivo es maximizar el retorno de la inversión para su empresa. Se basa en construir primero la funcionalidad de mayor valor para el cliente y en los principios de inspección continua, adaptación, auto-gestión e innovación. Con este método el cliente se entusiasma y se compromete con el proyecto dado que lo ve crecer iteración a iteración. Y también promueve la innovación, motivación y compromiso del equipo que forma parte del proyecto, por lo que los profesionales encuentran un ámbito propicio para desarrollar sus capacidades.
Aunque en este caso en concreto se ha adaptado este método a tan sólo un cliente, un desarrollador. Que consiste en que al principio de cada reunión se empieza enseñando al cliente con el resultado final del prototipo anterior, para ver si han cumplido los requisitos exigidos. Y en el caso de que sea la primera reunión pasaremos directamente a detallar los requisitos y el objetivo para este prototipo.
Y que constarán de cinco iteraciones para la elaboración de este proyecto.

4.1. Iteración 0. Estudio previo

4.1.1. Requisitos

Antes de comenzar con el proyecto se acordó explorar diferente tipos de bibliotecas de visualización gráfica, para ver cual se ajusta más con el proyecto.

Estas son las siguientes bibliotecas de visualización gráfica que se van ha explorar:

- Highcharts
- Chart.js
- Freeboard
- NVD3
- C3.js
- Dc.js

4.1.2. Desarrollo

La exploración de cada biblioteca consiste en realizar unas simples demos, pero todas con el mismo fichero JSON de demografía para ver la eficiencia. Combinando con `crossfilter` y el rendimiento de cada uno, y seleccionar las que nos ofrecen mayor prestación.

4.1.3. Resultado

La concusión que sacamos al explorar las bibliotecas, es que la biblioteca dc.js es de las bibloiotecas que más se ajusta a lo que queremos aunque no esta nada mal la biblioteca c3.js, asique también haremos algo con esta biblioteca, aunque en un segundo plano y también con la biblioteca NVD3. Las bibliotecas C3.js y NVD3 al no estar totalmente integrado con crossfilter, el tiempo de filtrado es mayor en comparación con dc.js. En las siguientes iteraciones se verán en más profundidad estas tres bibliotecas.

4.2. Iteración 1. Dc.js

En esta segunda fase una vez exploradas las bibliotecas y viendo las diferencias que hay en cada uno. Nos inclinamos principalmente a la biblioteca dc.js por la integración total con crossfilter, una biblioteca altamente potente para el filtrado de datos, esto hace que esta biblioteca sea perfecta para nuestro proyecto, que consiste en visualización y filtrado de datos.

4.2.1. Requisitos

Pasar de una simple demo a algo más consistente y con un cierto orden al representarlos. Gráficas que muestren diferentes datos y que se puedan filtrar por ellas y gracias al `crossfilter` que interactuarán entre todas ellas. Estas serán las gráficas que se harán:

- *Siguen o no retenido con el proyecto (retention)*: Representado en forma de tarta la gráfica y las porciones serán en proporción a la cantidad de datos que coincida a cada uno.
- *Año (year)*: Representado en forma de tarta la gráfica y las porciones serán en proporción a la cantidad de datos que coincida con ese determinado año.
- *Mes (month)*: Representado en forma de tarta la gráfica y las porciones serán en proporción a la cantidad de datos que coincida con ese determinado mes.
- *Día (day)*: Representado en forma de tarta la gráfica y las porciones serán en proporción a la cantidad de datos que coincida con ese determinado día.
- *Días de la semana (days of the week)*: Representado en forma de tarta la gráfica y las porciones serán en proporción a la cantidad de datos que coincida con ese determinado día de la semana.
- *Demografía (demography)*: Representado en forma de barras horizontales la gráfica y el tamaño de las barras serán en proporción a la cantidad de datos que coincida en ese rango de edades (cada barra va de 0-181 días a que se le llama edad).
- *Zona horaria (TZ)*: Representado en forma de barras la gráfica y el tamaño de las barras serán en proporción a la cantidad de datos que coincida en ese zona horaria.

- *Compañía (companies)*: Representado en forma de barras horizontales la gráfica y el tamaño de las barras serán en proporción a la cantidad de datos que coincida con esa compañía.

4.2.2. Desarrollo

Extracción de datos

Para poder empezar a trabajar primero tenemos que sacar los datos, en nuestro caso están sacados de unos ficheros JSON de demografía de OpenStack. Una vez con los datos ya en mano, usamos la función de Jquery (`getJSON`) para extraerlo. Tenemos que juntar dos ficheros JSON en un único diccionario, ya que, nos ofrecen en uno de ellos los proyectos que aun se siguen trabajando y en la otra los que ya están terminados. Pasamos a masajear los datos y acondicionando a como los quiere `crossfilter`, ya que requiere una forma especíca para su correcto uso que es la que se muestra acontinuación:

```
[
    {"name": "user1", "date": 2001-04-23, id: 1},
    {"name": "user2", "date": 2004-02-03, id: 2},
    {"name": "user3", "date": 2011-10-07, id: 3}
]
```

Como en estos JSON no incluyen ni compañías ni zona horaria, lo hemos añadido directamente cuando masajeábamos los datos para que sea más completo. Con los datos ya listos lo pasamos por la biblioteca crossfilter, también incluiremos algunos atributos en esta parte del proceso con el fin de poder representar mejor las gráficas.

Implementación

Una vez que tengamos ya los datos de la forma adecuada para usarlo con dc.js, nos dispondremos a dibujar las gráficas. Para representar las gráficas necesitaremos dos variables:

- `Dimension`: La dimensión de la gráfica que queramos dibujar, es decir, por años, mes, día, etc.
- `Group`: Como queremos agrupar la dimensión.

4.2.3. Resultado

Veamos el resultado de esta iteración en la figura 4.1. Un dasboard con diferentes gráficas que muestra determinada información. Este es un prototipo simple sin muchos detalles, pero se

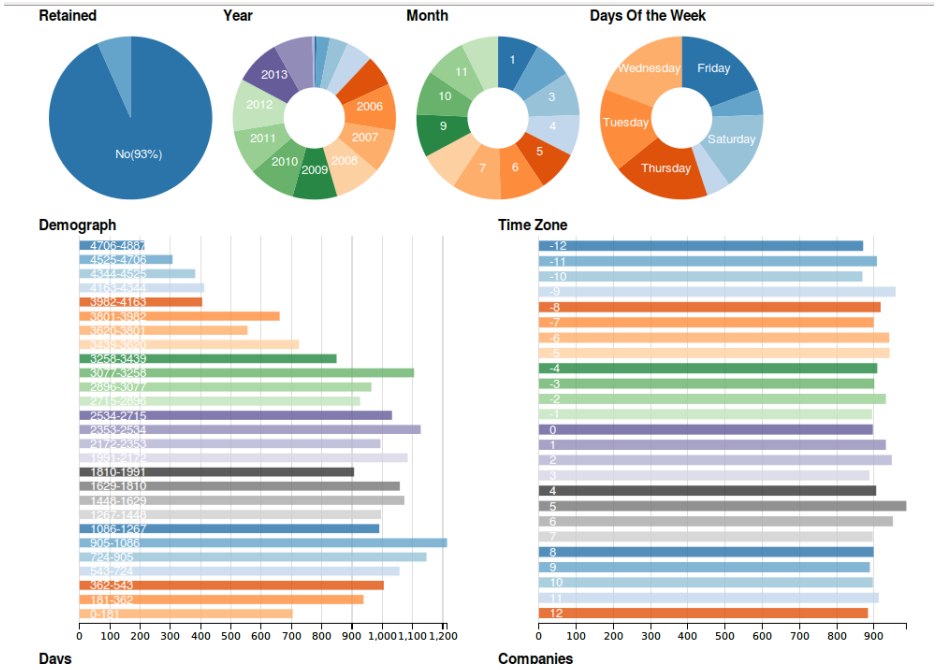


Figura 4.1: Gráficas de la biblioteca dc.js

cumple a los requisitos que se han marcado con todas las gráficas.

Ejemplo con código:

```
var aging = {};
var birth = {};
$.getJSON('its-demographics-aging.json', function (data) {
    aging = data;
}),
$.getJSON('its-demographics-birth.json', function (data) {
    birth = data;
})

... ...

var yearChart = dc.rowChart('#year-chart');
var datas = birAgin(birth['persons'], aging['persons']);
var data = dcFormat(datas);
var ndx = crossfilter(data);

var yearDim = ndx.dimension(function(d) {
    return d.date.getFullYear();
});

var yearGrp = yearDim.group();

yearChart
```

```
    .width(180)
    .height(180)
    .radius(80)
    .innerRadius(30)
    .dimension(yearDim)
    .group(yearGrp);

dc.renderAll();
```

4.3. Iteración 2. C3.js

En esta iteración haremos un cambio radical del proyecto, ya que vamos ha cambiar de biblioteca, concretamente pasamos de la biblioteca dc.js a C3.js para ver las diferencias en más detalle.

4.3.1. Requisitos

Se hará algo parecido a lo que se hizo en la biblioteca dc.js, también experimentar como se comporta crossfilter con esta biblioteca, algo fundamental para el filtrado de datos, que es lo que pretendemos.
Se harán las siguientes gráficas:

- Año: Representado en forma de barras la gráfica y las barras serán en proporción a la cantidad de datos que coincida a cada determinado año.

- Mes: Representado en forma de tarta la gráfica y las porciones serán en proporción a la cantidad de datos que coincida a cada determinado mes.

- Demografía: Representado en forma de barras la gráfica y las barras serán en proporción a la cantidad de datos que coincida a cada determinado rago de edades (cada rango van de 0-181 días llamados edades). Tiene dos barras diferentes en cada rango que son los que siguen retenidos en el proyecto y los que no.

- Día: Representado en forma de barras la gráfica y las barras serán en proporción a la cantidad de datos que coincida a cada determinado día.

4.3.2. Desarrollo

Extracción de datos

La extracción de datos será muy similar a la de dc.js sin decir que es casi idéntica. Primero extraeremos los datos de los correspondientes JSON con la función de jquery (`getJSON`). Como en el mismo caso de dc.js procederemos a juntar los dos JSON en uno, y con todos estos procedimientos ya podremos a masajear los datos a como los quiere crossfilter. Lo pasamos por crossfilter de la misma manera (`var ndx = crossfilter(data)`), estará listo para modularlo de una manera más eficiente los datos. Esta biblioteca es más flexible que la de dc.js, ya que, dispone de varios formatos como se muestran acontinuación:

```
data: {
    columns: [
        ['data1', 30, 20, 50, 40, 60, 50],
        ['data2', 200, 130, 90, 240, 130, 220],
        ['data3', 300, 200, 160, 400, 250, 250]
    ]
}
```

```
data: {
    x : 'x',
    columns: [
        ['x', '1', '2', '3', '4'],
        ['data1', 30, 200, 100, 400],
        ['data2', 90, 100, 140, 200],
    ]
}
```

```
data: {
    rows: [
        ['data1', 'data2', 'data3'],
        [90, 120, 300],
        [40, 160, 240],
        [50, 200, 290],
        [120, 160, 230],
        [80, 130, 300],
        [90, 220, 320],
    ]
}
```

```
data: {
    json: {
        data1: [30, 20, 50, 40, 60, 50],
        data2: [200, 130, 90, 240, 130, 220],
        data3: [300, 200, 160, 400, 250, 250]
    }
}
```

```
data: {
    url: '/data/c3_test.csv'
}
```

Dependiendo del tipo de gráficas queramos representar nos favorecerán un tipo u otro.

Implementación

En esta parte de la implementación es mucho más tediosa que la de dc.js (al estar perfectamente integrado `crossfilter`). Lo primero que se necesita es saber que tipo de gráfica se va ha representar para escoger el formato más adecuado. Se seguirá necesitando las dos variables que se usó en dc.js (`Dimension, Group`). En este momento se pasará ha usar el formato C3.js que más conviene, sacando los datos del parámetro `Group`. Y así se podrá a pasar a la representación de la gráfica.

4.3.3. Resultado

Ejemplo con código.

```
var aging = {};
var birth = {};
$.getJSON('its-demographics-aging.json', function (data) {
    aging = data;
}),
$.getJSON('its-demographics-birth.json', function (data) {
    birth = data;
})

... ...

var chart_year;
var dim = ndx.dimension(function(d) {
    return d.date.getFullYear();
});
var grp = dim.group().reduceSum(function(d) {
    return d.one;
});
var year_data = c3Format('year', grp);
chart_year = c3.generate({
    data: {
        columns: year_data,
        type: 'bar',
```

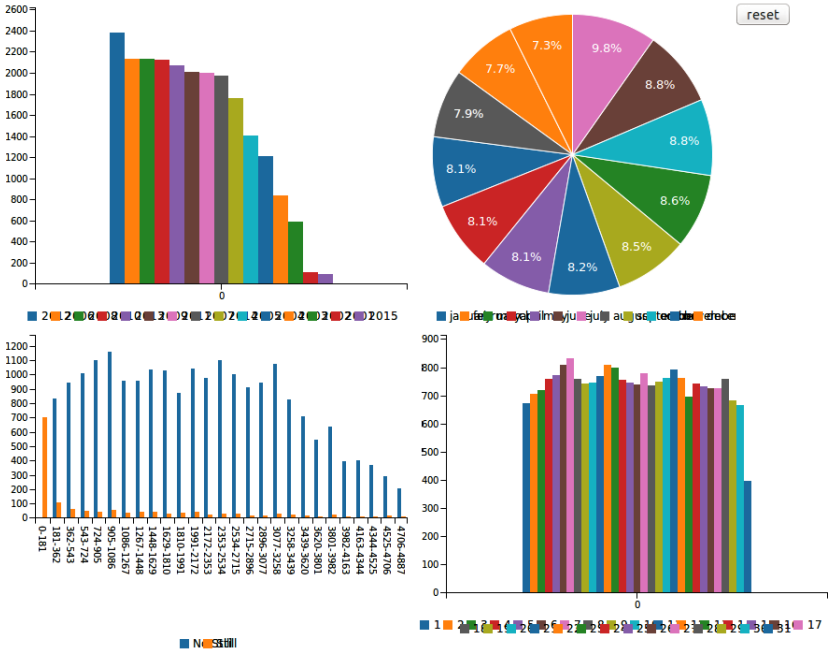


Figura 4.2: Gráficas de la biblioteca c3.js

```
    },
    size: {
height: 350,
width: 450
},
bindto: d3.select('#year')
});
```

Para conseguir el efecto de que se actualicen todas las gráficas a la vez, es necesario hacer ese cambio manualmente, es decir, capturar el evento en donde se hizo el clic y filtrar con `crossfilter` el dato seleccionado y volver a repintar todas las gráficas. Este efecto será más lento y menos eficiente que la biblioteca dc.js.

Veamos como sería el resultado de esta iteración en la figura 4.2

4.4. Iteración 3. NVD3.js

Entraremos más en profundidad en esta biblioteca, que parecía bastante interesante, en la fase de exploración.

4.4.1. Requisitos

Se hará algo parecido a lo que se hizo en la biblioteca c3.js. Si se puede llegar más lejos que con c3.js. Se harán las siguientes gráficas:

- Año: Representado en forma de barras la gráfica y las barras serán en proporción a la cantidad de datos que coincida a cada determinado año.

- Mes: Representado en forma de línea la gráfica y los puntos serán en proporción a la cantidad de datos que coincida a cada determinado mes.

- Demografía: Representado en forma de barras horizontales la gráfica y las barras serán en proporción a la cantidad de datos que coincida a cada determinado rango de edads (los rangos van de 0-181 días llamado edades). Tiene dos barras diferentes en cada rango que son los que siguen retenidos en el proyecto y los que no.

4.4.2. Desarrollo

Extración de datos

Extraer los datos de los correspondientes JSON con la función de Jquery (`getJSON`). Lo pasamos por crossfilter de la misma manera (`var ndx = crossfilter(data)`). Masajear los datos según como lo quiera la biblioteca nvd3 que es de las siguientes formas:

```
var testdata1 = [
    {key: "One", y: 5},
    {key: "Two", y: 2},
    {key: "Three", y: 9},
    {key: "Four", y: 7},
    {key: "Five", y: 4},
    {key: "Six", y: 3},
    {key: "Seven", y: 0.5}
]

var testdata2 = [
    {key: "One", values: [2, 1, 4, 3, 6, 3, 4, 3, 2, 5]},
    {key: "Two", values: [7, 34, 6, 23, 6, 2, 3, 7, 5, 0]},
    {key: "Three", values: [-3, 4, 34, 6, 8, 6, 4, 3, 0 ,5]}
]
```

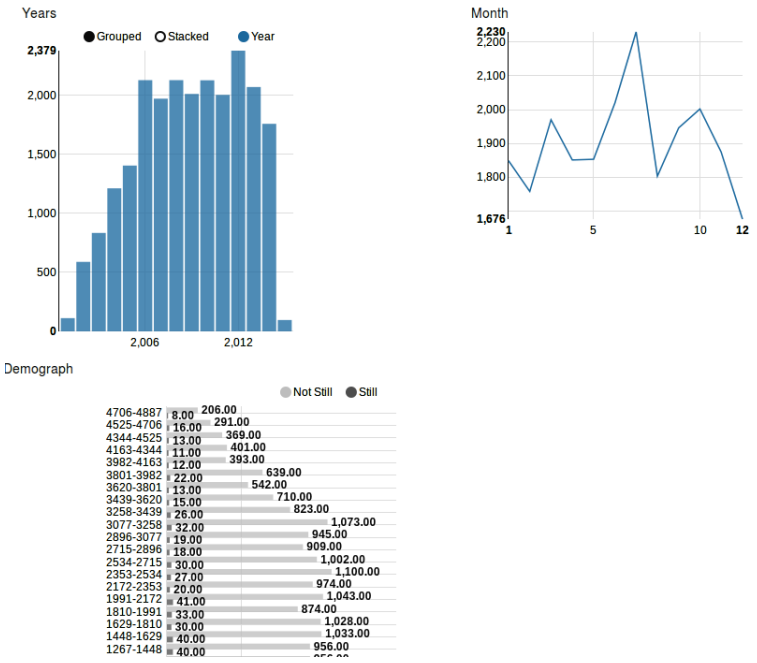


Figura 4.3: Gráficas de la biblioteca nvd3

Implementación

A la hora de implementarlo es más tedioso que en la biblioteca C3.js, por el hecho de que se necesita un parámetro para referenciar a esa gráfica y dentro de la función de dibujar hay que asignárselo, esto es necesario sólo si queremos capturar el evento clic de una gráfica y cambiarlo en el resto de las gráficas que se tenga, y se referencia con ese parámetro.

4.4.3. Resultado

Al concluir las pruebas se decidió abandonar esta biblioteca, porque tanto en dc.js como c3.js nos simplifica bastante más que esta biblioteca en vista a los objetivos de este proyecto. Veamos la imagen de hasta donde se llegó con esta biblioteca con la figura 4.3.

4.5. Iteración 4. Dc.js DataTable

En esta parte de la iteración se vuelve ha usar la biblioteca dc.js. Retomamos de la iteración 1 en donde lo dejamos, pero esta vez se centrará en la creación de la tabla (`DataTable`).

4.5.1. Requisitos

Se creará una tabla como una especie de panel de control, con diferentes maneras de filtrado, incluidas directamente en la misma tabla, para que su uso sea más intuitivo dinámico y versátil. Se incluirán los siguientes filtros en la tabla:

- Checkbox en cada elemto de la tabla
- Filtro AND
- Se pueden hacer clic cada elemento y filtrar sobre ella directamente o información dependiendo del elemento en concreto
- Botón de ordenar en cada columna
- Buscador por nombres de usuarios
- Función de scroll. Cuando se llegue al final la página, cargará cinco elementos más hasta mostrar todos

Y fuera de la tabla también incluiremos dos tipos de filtrado `datePicker y slider`.

4.5.2. Desarrollo

La creación de la tabla no es difícil, es como crear cualquier otro chart. Para capturar los eventos en la tabla tendremos que escribir las funciones después de renderizarlo de la siguiente forma (`table.on('renderlet', function(table) { ... })`), de esta manera podemos capturar eventos de click, mousehover, etc, que es cuando están ya presentes en el árbol DOM.

Después de esto ya podemos empezar a trabajar en lo referente al filtrado. Empecemos explicando cuál es el funcionamiento del filtrado de los datos, de forma que podamos tener la sincronización que buscamos entre los distintos gráficos representados en nuestro entorno.

Al tener una checkbox en cada columna se puede elegir a un elemento determinado o varios, como quiera el usuario. Con esto tenemos un filtrado muy fino de los datos.

El filtro AND que hay en la tabla, sirve para cuando queremos filtrar por varias columnas a la vez, ya que muchas veces es necesario filtrar por varios filtros a la vez para tener una representación más personalizada.

También tenemos algunos elementos de la tabla que al hacer click se abre una venta nueva que muestra dicha información y en todas se pueden filtrar en esa ventana excepto la de los nombres de los usuarios que sólo se muestra su perfil completo.

Hay un buscador por nombre de usuario, muy útil si se sabe el nombre de ese usuario y se quiere saber toda su información.

Y por último para que no tarde demasiado la aplicación, en vez de mostrar todos los elementos que hay en la tabla, sólo se mostrará los 20 primeros, pero cuando se baja al final de la tabla se cargarán cinco elementos más y así hasta que llegue a su totalidad.

4.5.3. Resultado

El resultado final de este proyecto se ha conseguido con éxito, todo lo planeado se han llevado acabo aunque ya se está pensando en ampliar más este proyecto para que su funcionamiento sea más dinámico y genérico para que su uso sea más extenso.

Vamos ha entrar en detalle con los filtros aplicados con ejemplos ilustrativos.

- Tabla sin filtrar se verá en la figura 4.4.
- Hacer clic en el hueco de la fecha (DatePicker) se aparecerá un calendario, para filtrar por fechas (desde - hasta) en la figura 4.5.
- Arrastrar la barra del tiempo (slider) que tiene la misma función que el calendario en la figura 4.6.
- Si se quiere filtrar por nombre de usuario directamente en la figura 4.7.
- Hacemos clic en una de las fechas, aparecerá una ventana que se puede filtrar por año, mes o día en la figura 4.8.
- Hacemos click en (Company), aparece una ventana con todas las companías que se puede filtrar por dichas companías en la figura 4.9.
- Hacemos clic en el nombre del usuario se muestra en una venta su perfil completo en la figura 4.10.

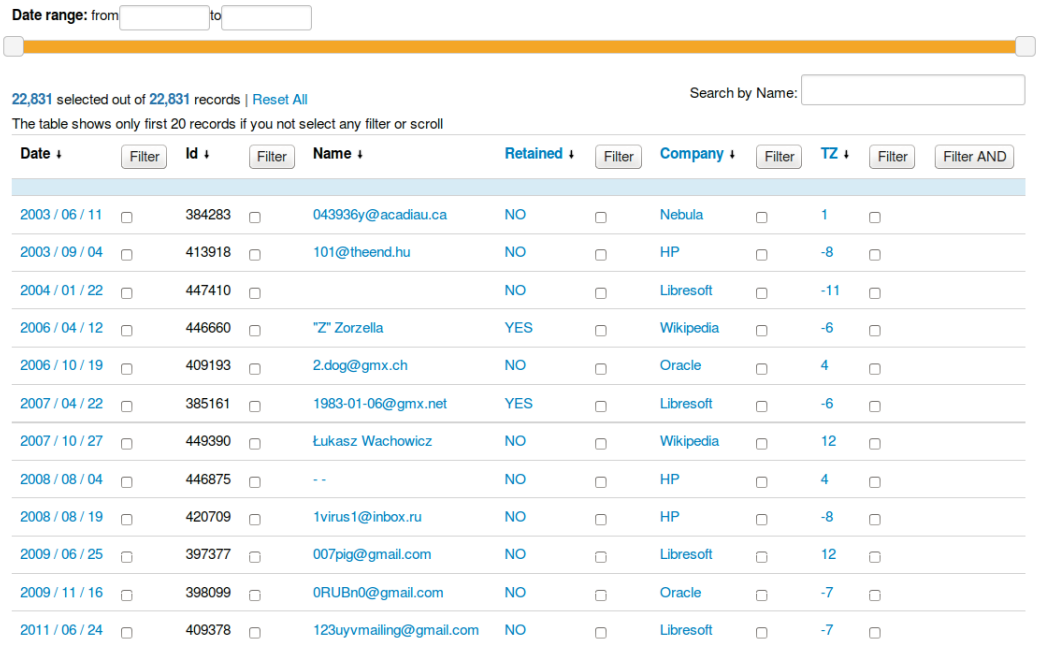


Figura 4.4: Gráficas de la biblioteca dc.js Tabla

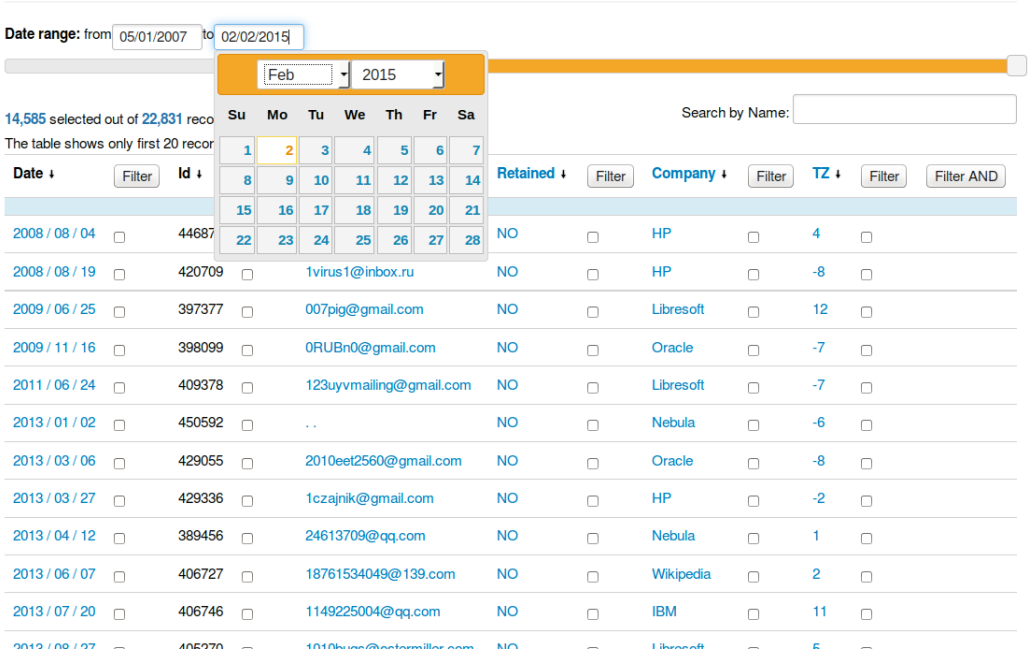


Figura 4.5: Gráficas de la biblioteca dc.js calendario

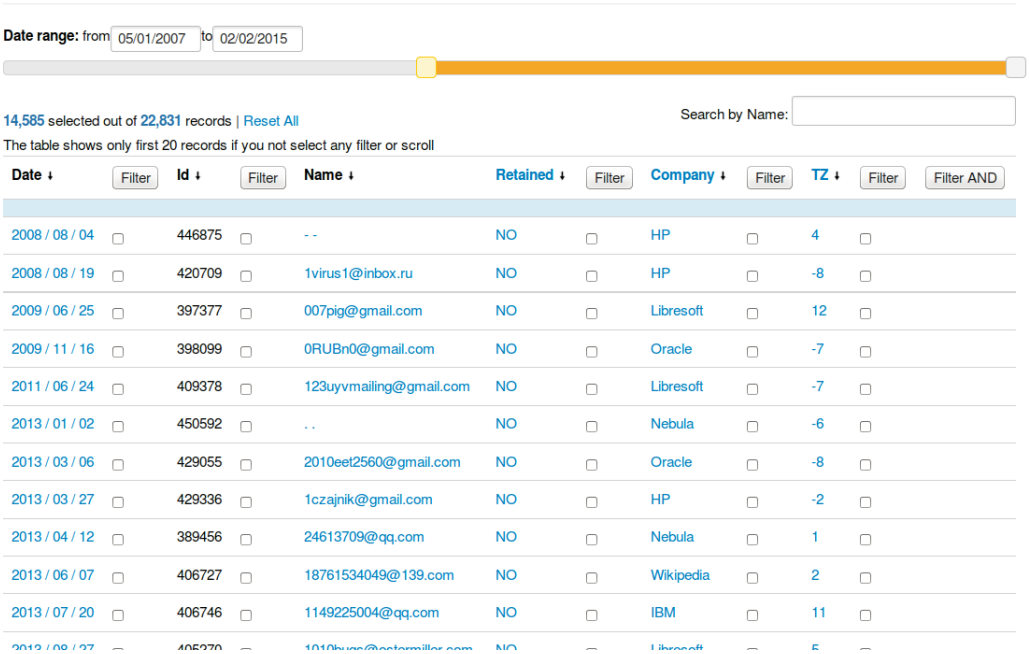


Figura 4.6: Gráficas de la biblioteca dc.js slider

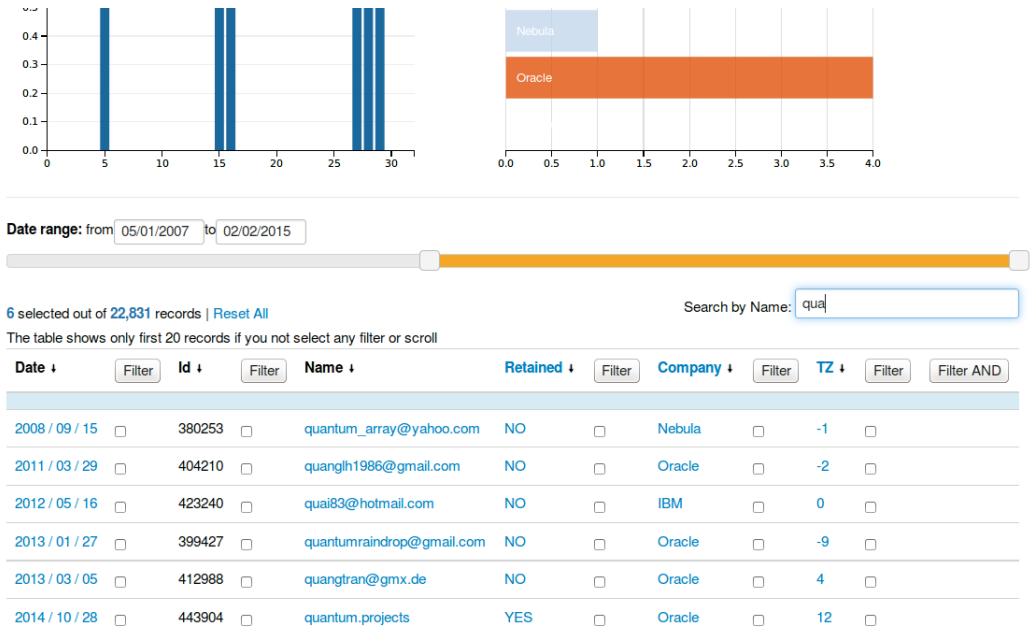


Figura 4.7: Gráficas de la biblioteca dc.js filtrado por nombre de usuario

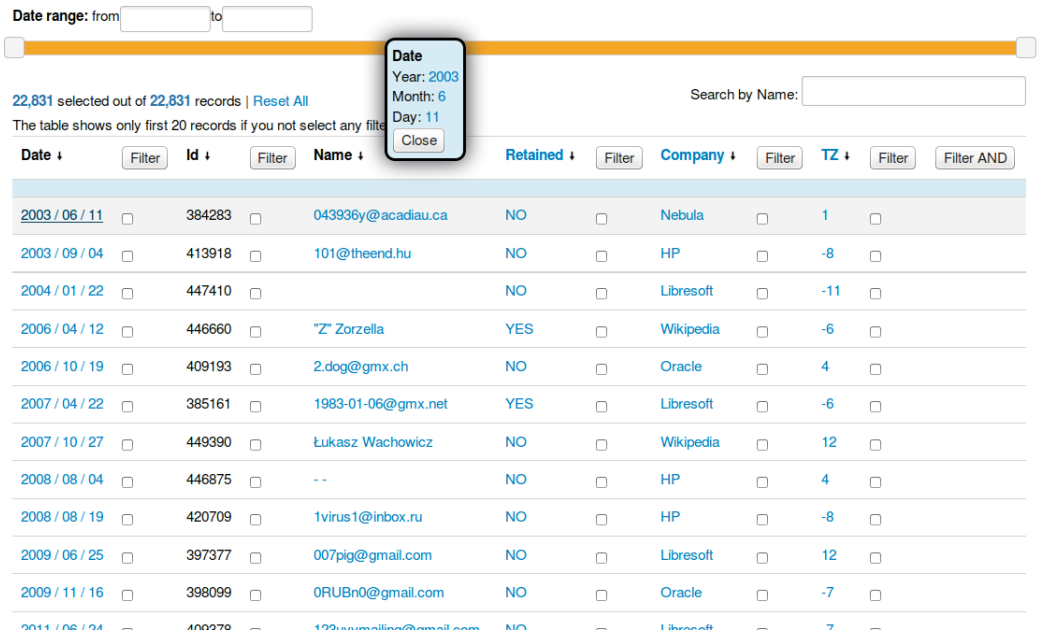


Figura 4.8: Gráficas de la biblioteca dc.js clic en fecha

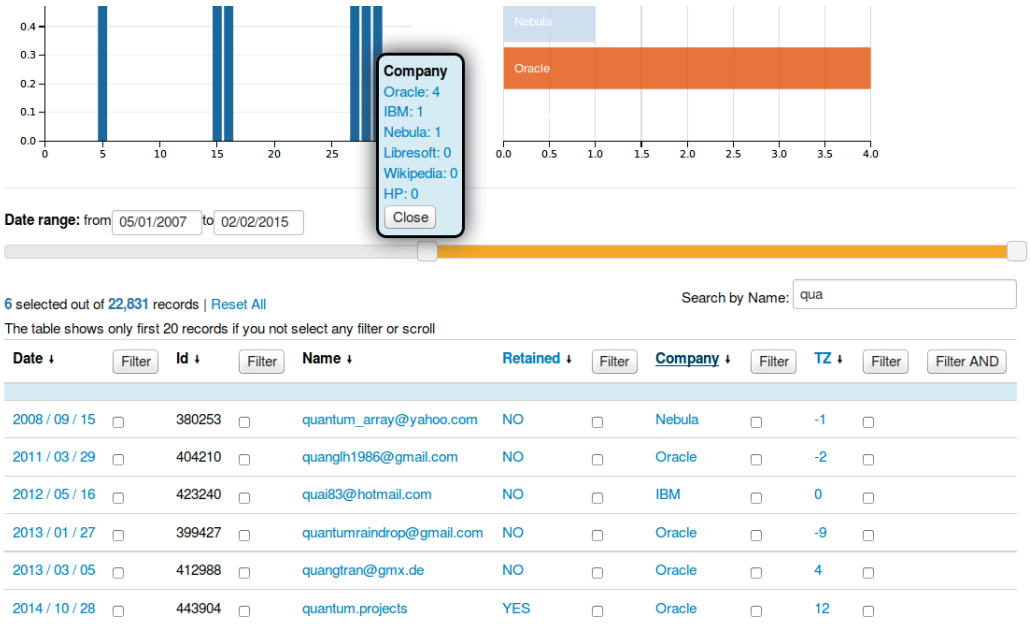


Figura 4.9: Gráficas de la biblioteca dc.js clic en (Company)

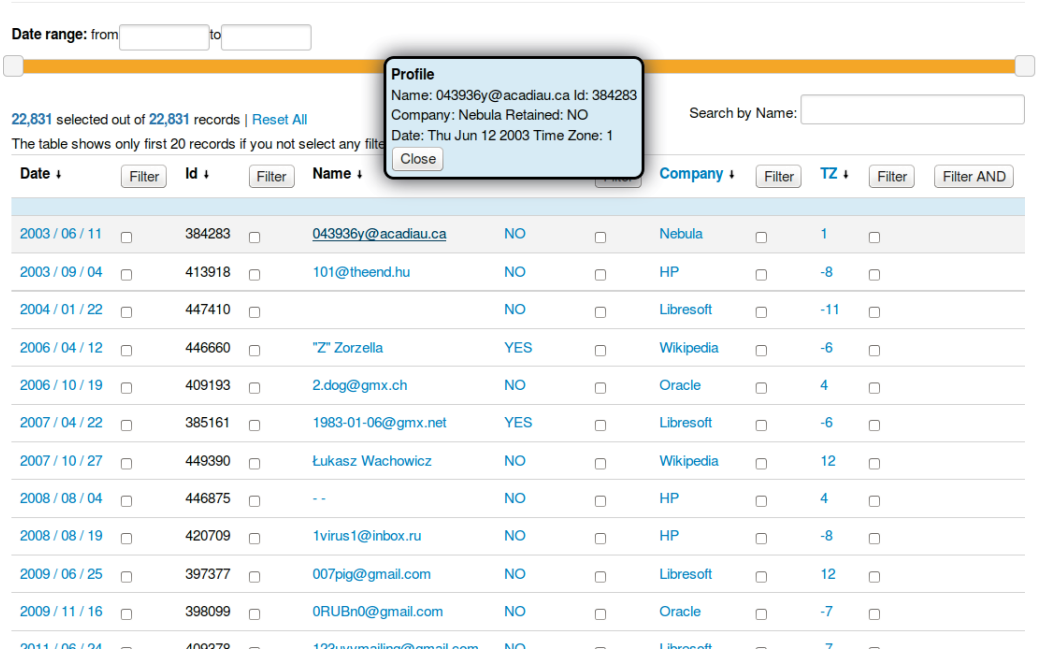


Figura 4.10: Gráficas de la biblioteca dc.js click name

Si nos fijamos bien, si metemos la fecha en el calendario (datePicker) se cambia la barra de tiempo (slider) y viceversa. Con todos estos filtros podemos filtrar todo lo que se quiera, hasta sacar el último dato.

4.6. Integración final

También se han hecho pruebas con diversos ficheros JSON de diferentes tamaños para comprobar la eficiencia de este proyecto con el fin de poder ampliar en un futuro a archivos más pesados. Los archivos adicionales que se han probado tienen los siguientes tamaños:

- 1000 entradas
- 10000 entradas
- 100000 entradas

El resultado fue bastante bueno, ya que en todas las pruebas, el programa ha funcionado de manera eficiente y rápida. En los primeros dos archivos no se esperaban problemas, porque no son archivos muy grandes. Y también funcionó bastante bien en la última de 100000 entradas,

que es algo ya considerable y posiblemente se pueda trabajar en un futuro con tamaños un poco más grande.

Una de los paneles que se van a presentar junto con el proyecto principal es el panel de commits, al que hemos hecho un dashboard de 114583 entradas, no va tan fluido como en el otro debido al tamaño. El fichero JSON que se utilizó para este panel han sido las siguientes:

- scm-commits-distinct.json: El JSON principal donde incluye la id del commit (id), la fecha (date), la id de la persona (person_id), el nombre del usuario (name), la id de la empresa (org_id), el id del repositorio (repo_id), la zona horaria (tz), la zona horaria original sin redondear las horas (tz_orig).
- scm-repos.json: El JSON donde aparecen la id y el nombre del repositorio.
- scm-companies.json: El JSON donde aparece la id y el nombre de la empresa.

Todos ellos están sacados del proyecto OpenStack.

Este nuevo panel consta de las siguientes gráficas:

- *Año (year)*: Representado en forma de tarta la gráfica y las porciones serán en proporción a la cantidad de datos que coincida con ese determinado año.
- *Mes (month)*: Representado en forma de tarta la gráfica y las porciones serán en proporción a la cantidad de datos que coincida con ese determinado mes.
- *Días de la semana (days of the week)*: Representado en forma de tarta la gráfica y las porciones serán en proporción a la cantidad de datos que coincida con ese determinado día de la semana.
- *Compañias (companies)*: Representado en forma de barras verticales la gráfica y el tamaño de las barras serán en proporción a la cantidad de datos que coincida con esa compañia ordenado de mayor a menor. Con otra gráfica debajo de la misma características pero se puede hacer seleccionando un rango con el ratón y la gráfica de arriba se ajustará a ese rango.
- *Repositorios (Repository)*: Representado en forma de barras verticales la gráfica y el tamaño de las barras serán en proporción a la cantidad de datos que coincida con ese repositorio ordenado de mayor a menor. Con otra gráfica debajo de la misma características

pero se puede hacer seleccionando un rango con el ratón y la gráfica de arriba se ajustará a ese rango.

- *Zona horaria (time zone)*: Representado en forma de barras verticales la gráfica y el tamaño de las barras serán en proporción a la cantidad de datos que coincida con esa zona horaria. Con otra gráfica debajo de la misma características pero se puede hacer seleccionando un rango con el ratón y la gráfica de arriba se ajustará a ese rango.

- *Commits (commits)*: Representado en forma de barras verticales la gráfica y el tamaño de las barras serán en proporción a la cantidad de datos que coincida con el número de commits que se han hecho en ese fecha. Con otra gráfica debajo de la misma características pero se puede hacer seleccionando un rango con el ratón y la gráfica de arriba se ajustará a ese rango.

- *Cantidad de compañías diferentes en esa fecha (company)*: Representado en forma de líneas con área la gráfica y la altura de los puntos serán en proporción a la cantidad de diferentes compañías que hay en esa fecha. Con otra gráfica debajo de la misma características pero se puede hacer seleccionando un rango con el ratón y la gráfica de arriba se ajustará a ese rango.

- *Cantidad de repositorios diferentes en esa fecha (repository)*: Representado en forma de líneas con área la gráfica y la altura de los puntos serán en proporción a la cantidad de diferentes repositorios que hay en esa fecha. Con otra gráfica debajo de la misma características pero se puede hacer seleccionando un rango con el ratón y la gráfica de arriba se ajustará a ese rango.

- Una tabla con toda la información.

- Buscador por nombre.

- Un datePicker para seleccionar un rango de fechas (inicio - fin).

Este dashboard se hizo en las últimas semanas, que en un principio sólo iba a ser una prueba para ver si funcionaba razonablemente. Después de ver el rendimiento se decidió ha hacer un dashboard de commits.
No es un panel igual que el de demografía ya que en este no se han incluido los filtros en la tabla

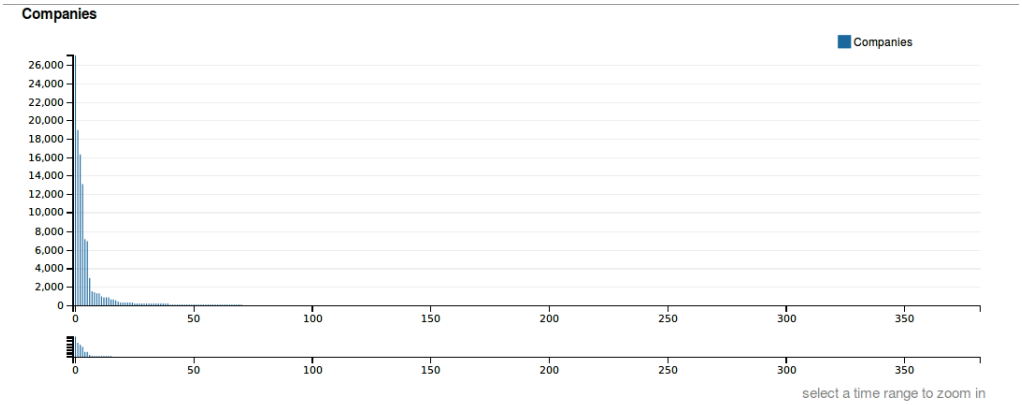


Figura 4.11: Panel commits: Compañías sin filtrar

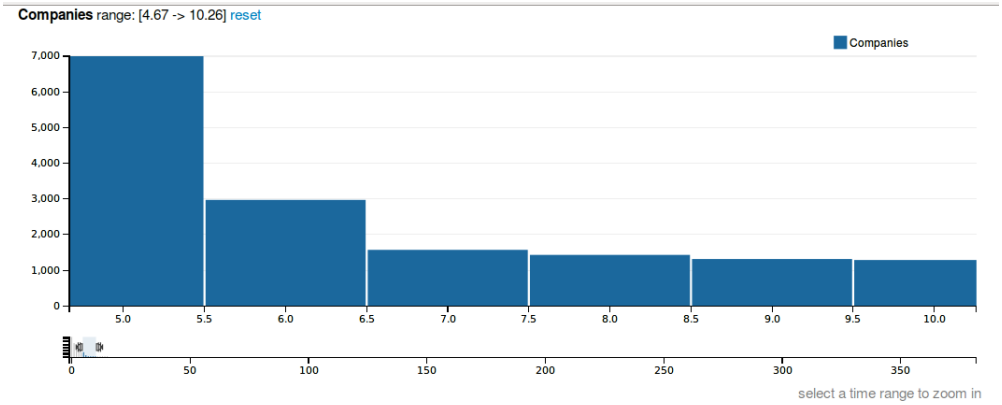


Figura 4.12: Panel commits: Compañías rango

por falta de tiempo, será una versión más reducida pero que nos muestran otro tipo de datos y otras visualizaciones diferentes para entender mejor los datos de los commits.

Vamos a ver algunos ejemplos de visualización de este nuevo panel, sólo los que son diferentes al panel de demografía. Las gráficas de Companies, Repository y Time Zone tienen la misma funcionalidad.

- Compañías (companies): En la figura 4.11 se muestra la gráfica sin filtrar tal cual, como está. Cuando elegimos un rango como se ve en la figura 4.12 vemos que la gráfica se ha ajustado a ese rango.

- Repositorios (repositories): La funcionalidad de esta gráfica es la misma pero vamos a ver como en la figura 4.13 que en la gráfica después de haber seleccionado un rango también podemos seleccionar otro rango más pequeño si queremos ver en más profundidad.

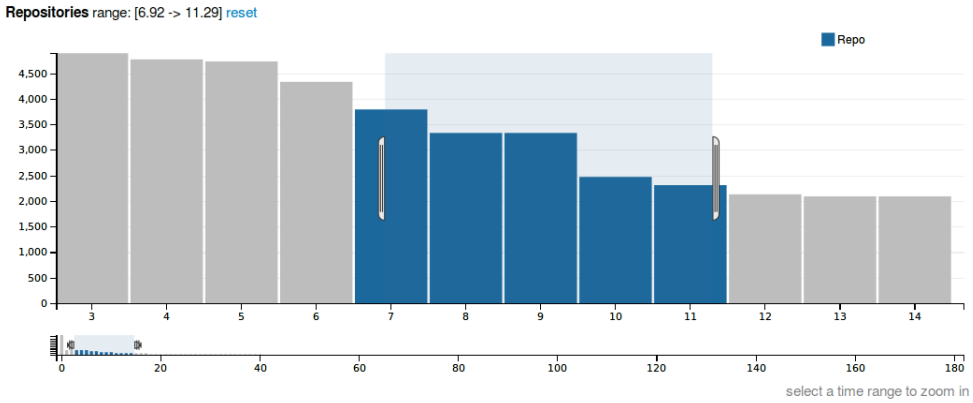


Figura 4.13: Panel commits: Repositorios rango y filtro

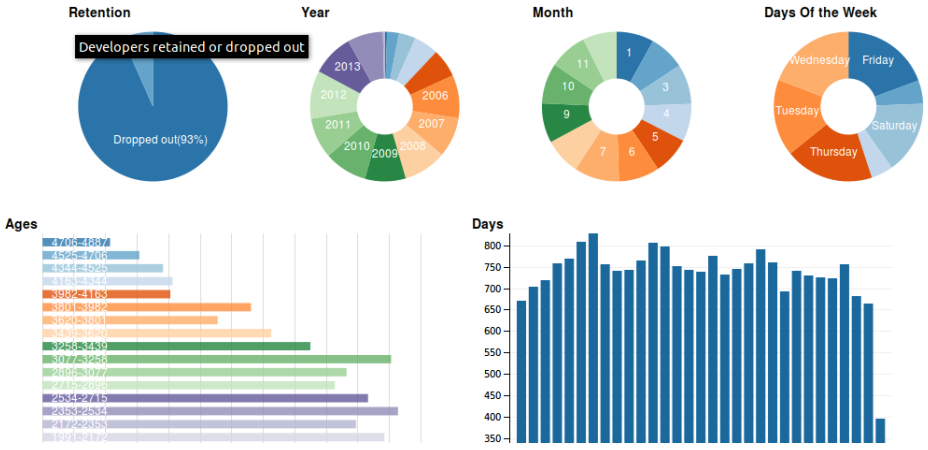


Figura 4.14: Texto en los títulos

Es un dashboard más lento, aunque era de esperar por su tamaño. Con esto vemos que esta biblioteca se puede trabajar con tamaños grandes, aunque esta hecho con poco tiempo se puede decir casi con seguridad que podremos optimizar para que tenga un mejor rendimiento.

En los dos paneles si dejamos el ratón encima de los títulos nos saldrá un texto que informa que tipo de gráfica es, si tenemos dificultad de entender a primera vista. Ejemplo en la figura 4.14.

Capítulo 5

Resultados finales

5.1. Extración de datos

Los datos que se usaron para este proyecto se han sacado de OpenStack (`http://activity.openstack.org/dash/browser/data/json/`), que a su vez usa la tecnología de `Metrics Grimoire` para crear los ficheros JSON de donde se cogen para ser usados en este proyecto. Las gráficas se pintan con la tecnología `VizGrimoire` en el dashboard de OpenStack (`http://activity.openstack.org`).

Lo único que hay que hacer es entrar en la página de OpenStack y descargarse los json que se quiera trabajar, en este caso han sido los ficheros `its-demographics-aging.json, its-demographics-birth.json` para el panel de demografía. Y los ficheros `scm-commits-distir scm-companies.json, csm-repos.json` para el panel de commit.

5.2. Funcionamiento

Este proyecto se enfoca a la visualización y filtrado de los datos. Un dashboard dinámico, sencillo e intuitivo. Tenemos dos paneles donde el funcionamiento es parecido aunque se usan ficheros JSON diferentes, esto hace que apariencia cambia. Debido a que cada panel está hecha específicamente para cada uno.

5.2.1. Panel de demografía

Tenemos el panel de la demografías que es el proyecto principal que consta de los siguientes gráficos:

- *Siguen o no retenido con el proyecto (retention)*: Representado en forma de tarta la gráfica y las porciones serán en proporción a la cantidad de datos que coincida a cada uno.

- *Año (year)*: Representado en forma de tarta la gráfica y las porciones serán en proporción a la cantidad de datos que coincida con ese determinado año.

- *Mes (month)*: Representado en forma de tarta la gráfica y las porciones serán en proporción a la cantidad de datos que coincida con ese determinado mes.

- *Día (day)*: Representado en forma de tarta la gráfica y las porciones serán en proporción a la cantidad de datos que coincida con ese determinado día.

- *Días de la semana (days of the week)*: Representado en forma de tarta la gráfica y las porciones serán en proporción a la cantidad de datos que coincida con ese determinado día de la semana.

- *Demografía (demography)*: Representado en forma de barras horizontales la gráfica y el tamaño de las barras serán en proporción a la cantidad de datos que coincida en ese rango de edades (cada barra va de 0-181 días a que se le llama edad).

- *Zona horaria (TZ)*: Representado en forma de barras la gráfica y el tamaño de las barras serán en proporción a la cantidad de datos que coincida en ese zona horaria.

- *Compañía (companies)*: Representado en forma de barras horizontales la gráfica y el tamaño de las barras serán en proporción a la cantidad de datos que coincida con esa compañía.

Vamos a ver diferentes visualizaciones para detallar en profundidad cómo funciona.

El dashboard sin aplicar ningún filtro y donde aparecen todos los datos representados con las gráficas y la tabla, lo veremos en las figuras 5.1 y 5.2.

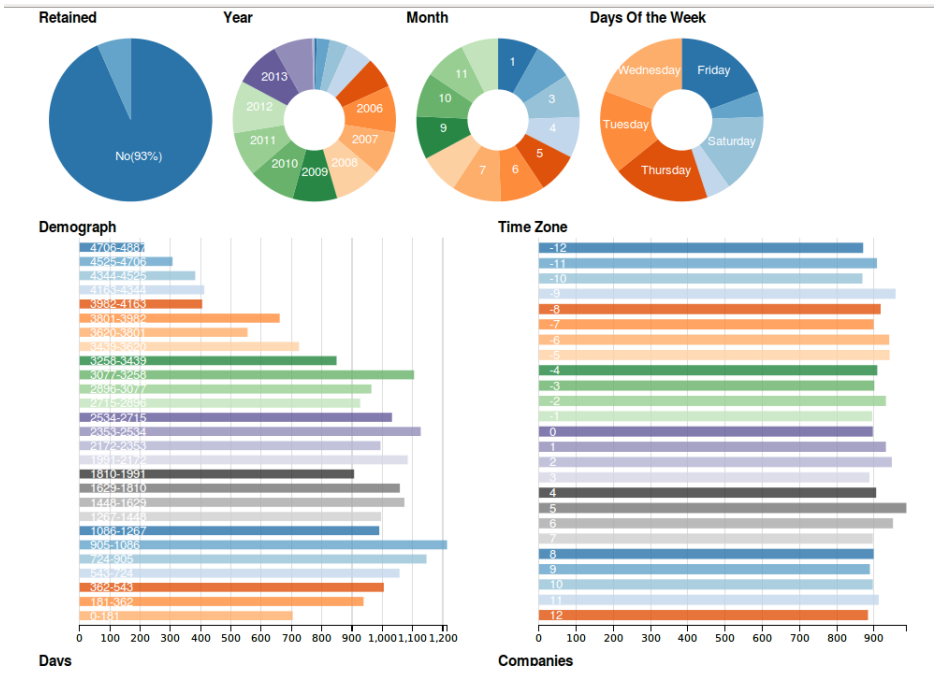


Figura 5.1: Sin filtros aplicados

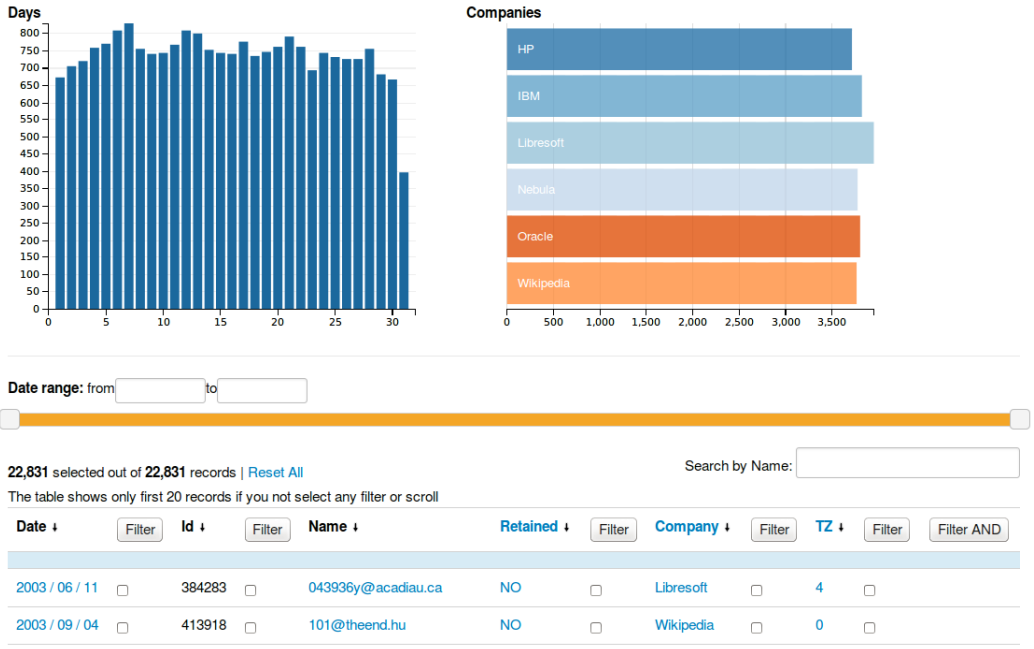


Figura 5.2: Sin filtros aplicados

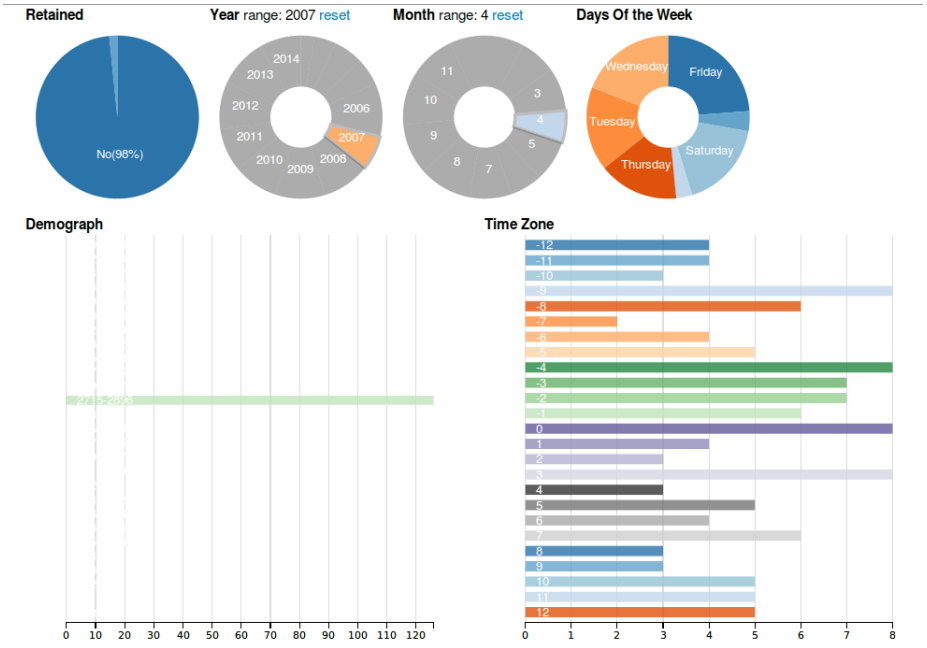


Figura 5.3: Filtrado por la fecha abril de 2007

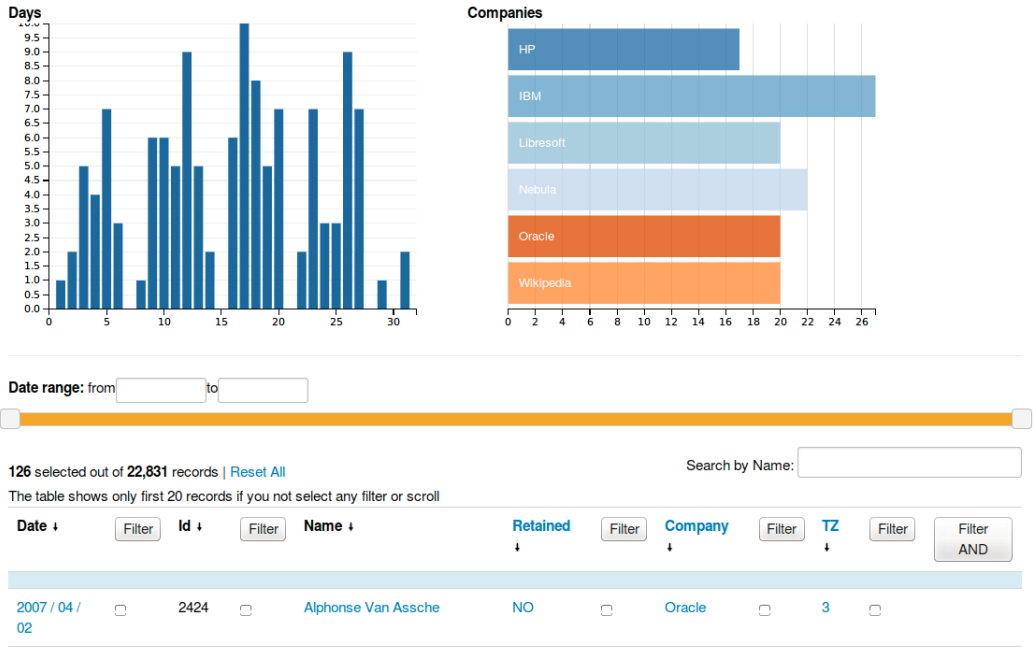
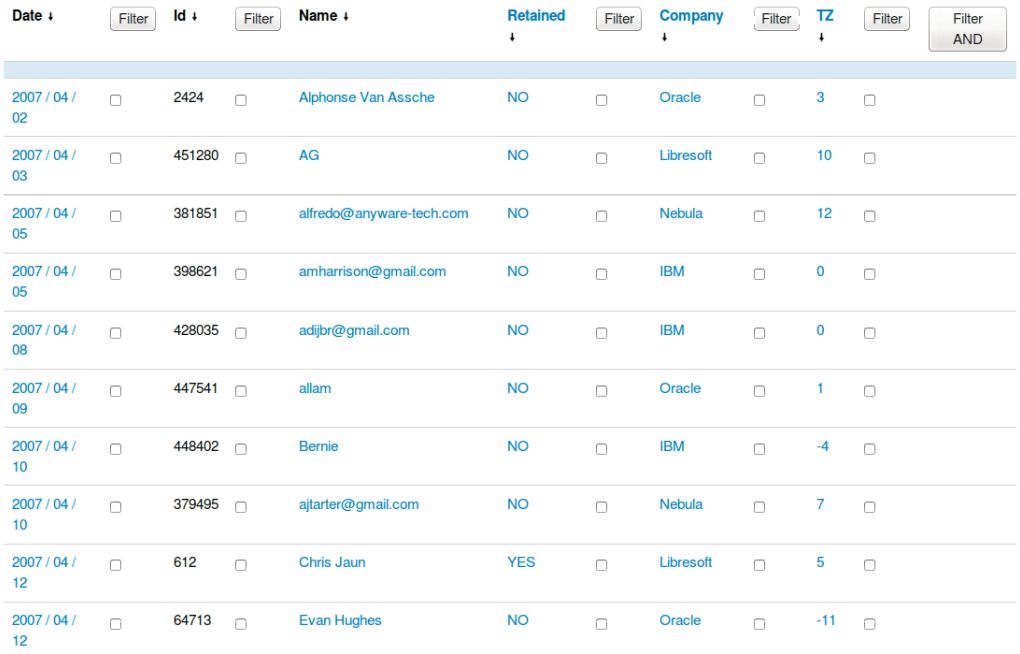


Figura 5.4: Filtrado por la fecha abril de 2007



Date	Filter	Id	Filter	Name	Retained	Filter	Company	Filter	TZ	Filter	Filter AND
2007 / 04 / 02		2424		Alphonse Van Assche	NO		Oracle		3		
2007 / 04 / 03		451280		AG	NO		Libresoft		10		
2007 / 04 / 05		381851		alfredo@anyware-tech.com	NO		Nebula		12		
2007 / 04 / 05		398621		amharrison@gmail.com	NO		IBM		0		
2007 / 04 / 08		428035		adijbr@gmail.com	NO		IBM		0		
2007 / 04 / 09		447541		allam	NO		Oracle		1		
2007 / 04 / 10		448402		Bernie	NO		IBM		-4		
2007 / 04 / 10		379495		ajtarter@gmail.com	NO		Nebula		7		
2007 / 04 / 12		612		Chris Jaun	YES		Libresoft		5		
2007 / 04 / 12		64713		Evan Hughes	NO		Oracle		-11		

Figura 5.5: Filtrado por la fecha abril de 2007

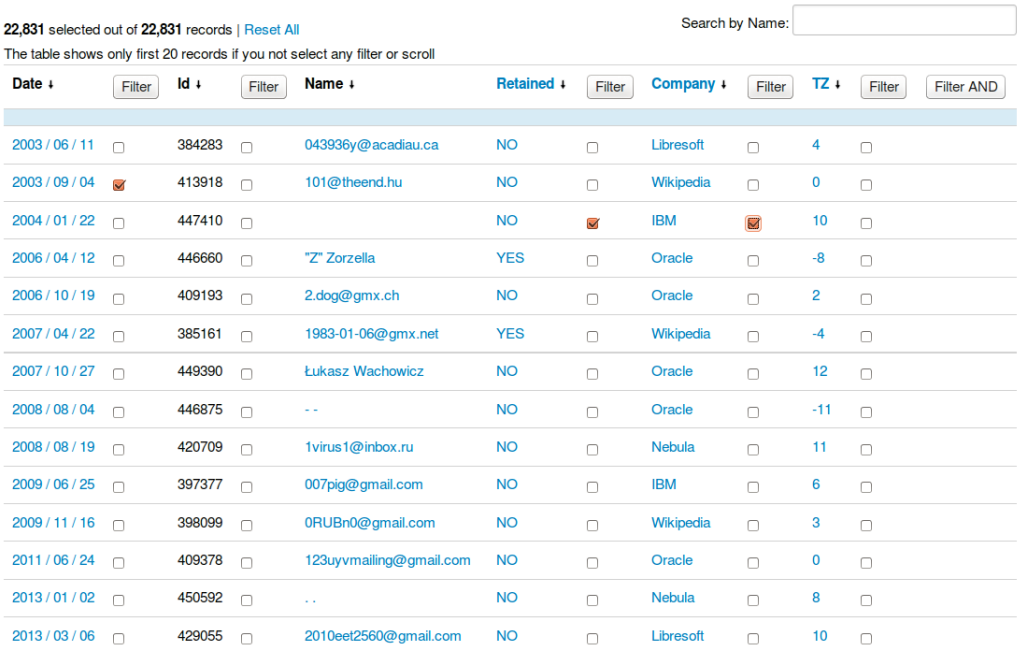
Cuando hacemos click en una de ellas por ejemplo en el año 2007 y en el mes de abril veremos que en el resto de las gráficas se van a cambiar según ese año y mes. Lo veremos en las figuras 5.3, 5.4 y 5.5.

La tabla se muestra al principio los veinte primeros datos, para no colapsar el navegador, ya que, al pintar todos los datos puede tardar bastante tiempo hasta que esté renderizado, por eso se crea la función de `scroll`, que consiste en que cuando bajamos hasta el final de la tabla se añaden a la representación cinco datos más, y así hasta mostrarlo todo.

En la tabla se puede filtrar por filtro AND, que quiere decir que tienen que cumplir todas los filtros aplicados.

Como se puede observar en la figura 5.7 sólo aparecen los datos que cumplen con la fecha 04/09/2009 que NO esté retenido y en la compañía de IBM, que es lo que se seleccionó en la figura 5.6.

Parecerá un error en la figura que tanto en las gráficas de días y de compañías no coinciden. Es debido a que cuando filtras por la tabla por compañías, la gráfica de la compañía no cambia. Veamos un ejemplo visual eligiendo Libresoft en este caso en la figura 5.8.



22,831 selected out of **22,831** records | Reset All

Search by Name:

The table shows only first 20 records if you not select any filter or scroll

Date	Filter	Id	Filter	Name	Retained	Filter	Company	Filter	TZ	Filter	Filter AND
2003 / 06 / 11	☐	384283	☐	043936y@acadiau.ca	NO	☐	Libresoft	☐	4	☐	
2003 / 09 / 04	☑	413918	☐	101@theend.hu	NO	☐	Wikipedia	☐	0	☐	
2004 / 01 / 22	☐	447410	☐		NO	☑	IBM	☑	10	☐	
2006 / 04 / 12	☐	446660	☐	"Z" Zorzella	YES	☐	Oracle	☐	-8	☐	
2006 / 10 / 19	☐	409193	☐	2.dog@gmx.ch	NO	☐	Oracle	☐	2	☐	
2007 / 04 / 22	☐	385161	☐	1983-01-06@gmx.net	YES	☐	Wikipedia	☐	-4	☐	
2007 / 10 / 27	☐	449390	☐	Łukasz Wachowicz	NO	☐	Oracle	☐	12	☐	
2008 / 08 / 04	☐	446875	☐	- -	NO	☐	Oracle	☐	-11	☐	
2008 / 08 / 19	☐	420709	☐	1virus1@inbox.ru	NO	☐	Nebula	☐	11	☐	
2009 / 06 / 25	☐	397377	☐	007pig@gmail.com	NO	☐	IBM	☐	6	☐	
2009 / 11 / 16	☐	398099	☐	0RUBn0@gmail.com	NO	☐	Wikipedia	☐	3	☐	
2011 / 06 / 24	☐	409378	☐	123uyvmailing@gmail.com	NO	☐	Oracle	☐	0	☐	
2013 / 01 / 02	☐	450592	☐	. .	NO	☐	Nebula	☐	8	☐	
2013 / 03 / 06	☐	429055	☐	2010eet2560@gmail.com	NO	☐	Libresoft	☐	10	☐	

Figura 5.6: Antes del filtro AND

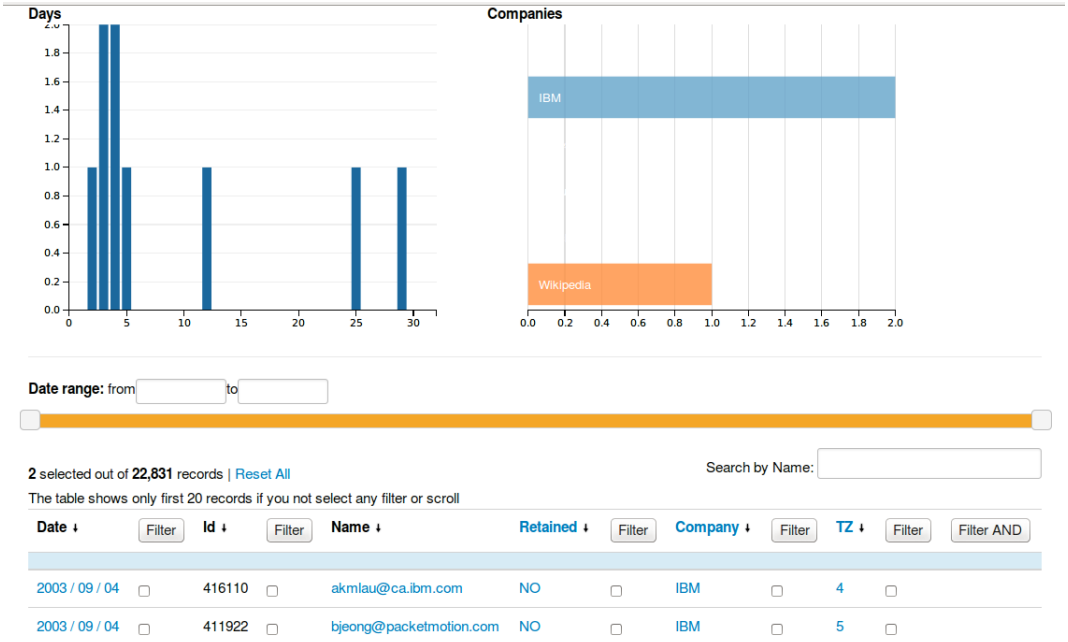


Figura 5.7: Después del filtro AND

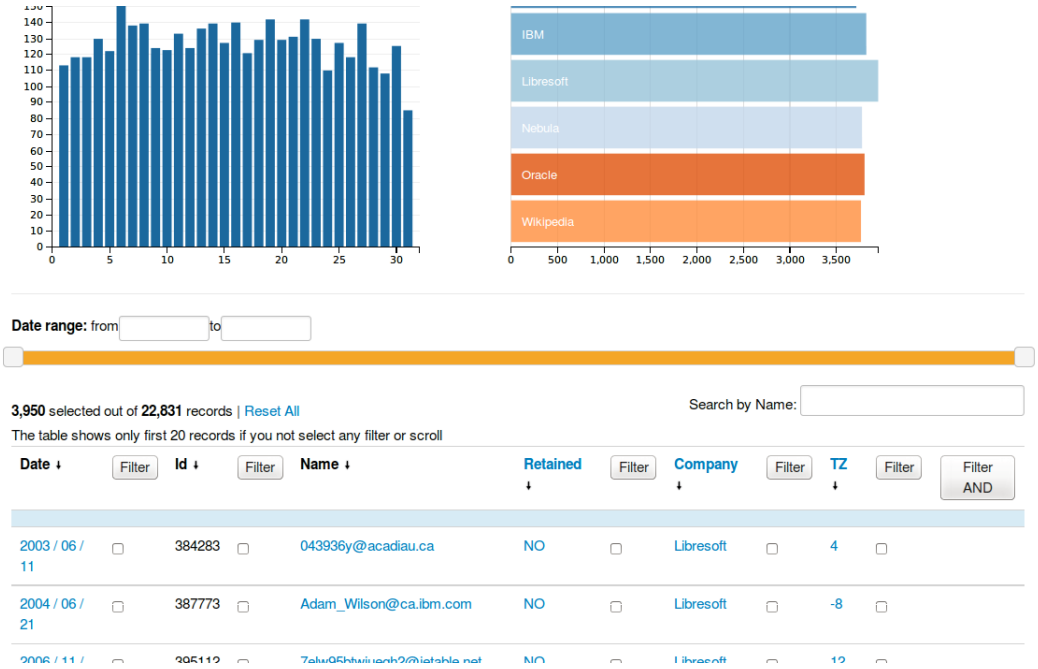


Figura 5.8: Filtrado por tabla

5.2.2. Panel de commits

En este panel que es el de los commits no se ha entrado en tanto detalle que en el panel de demografía por la falta de tiempo, ya que, sólo se le ha dedicado las últimas semanas para hacer este panel. Que en un principio sólo era una prueba para ver si se podía trabajar con ficheros JSON de tamaños más grandes.

Este panel consta de las siguientes gráficas para realizar su funcionamiento:

- *Año (year)*: Representado en forma de tarta la gráfica y las porciones serán en proporción a la cantidad de datos que coincida con ese determinado año.

- *Mes (month)*: Representado en forma de tarta la gráfica y las porciones serán en proporción a la cantidad de datos que coincida con ese determinado mes.

- *Días de la semana (days of the week)*: Representado en forma de tarta la gráfica y las porciones serán en proporción a la cantidad de datos que coincida con ese determinado día de la semana.

- *Compañias (companies)*: Representado en forma de barras verticales la gráfica y el tamaño de las barras serán en proporción a la cantidad de datos que coincida con esa com-

pañia ordenado de mayor a menor. Con otra gráfica debajo de la misma características pero se puede hacer seleccionando un rango con el ratón y la gráfica de arriba se ajustará a ese rango.

- *Repositorios (Repository)*: Representado en forma de barras verticales la gráfica y el tamaño de las barras serán en proporción a la cantidad de datos que coincida con ese repositorio ordenado de mayor a menor. Con otra gráfica debajo de la misma características pero se puede hacer seleccionando un rango con el ratón y la gráfica de arriba se ajustará a ese rango.

- *Zona horaria (time zone)*: Representado en forma de barras verticales la gráfica y el tamaño de las barras serán en proporción a la cantidad de datos que coincida con esa zona horaria. Con otra gráfica debajo de la misma características pero se puede hacer seleccionando un rango con el ratón y la gráfica de arriba se ajustará a ese rango.

- *Commits (commits)*: Representado en forma de barras verticales la gráfica y el tamaño de las barras serán en proporción a la cantidad de datos que coincida con el número de commits que se han hecho en ese fecha. Con otra gráfica debajo de la misma características pero se puede hacer seleccionando un rango con el ratón y la gráfica de arriba se ajustará a ese rango.

- *Cantidad de compañías diferentes en esa fecha (company)*: Representado en forma de líneas con área la gráfica y la altura de los puntos serán en proporción a la cantidad de diferentes compañías que hay en esa fecha. Con otra gráfica debajo de la misma características pero se puede hacer seleccionando un rango con el ratón y la gráfica de arriba se ajustará a ese rango.

- *Cantidad de repositorios diferentes en esa fecha (repository)*: Representado en forma de líneas con área la gráfica y la altura de los puntos serán en proporción a la cantidad de diferentes repositorios que hay en esa fecha. Con otra gráfica debajo de la misma características pero se puede hacer seleccionando un rango con el ratón y la gráfica de arriba se ajustará a ese rango.

- Una tabla con toda la información.

- Buscador por nombre.
- Un datePicker para seleccionar un rango de fechas (inicio - fin).

5.3. Funcionamiento interno

Vamos a detallar algunas cosas sobre el filtrado de datos de `crossfilter` y por qué hace que sea tan rápido. Como se explicó en el capítulo de tecnologías usados. `Crossfilter`, añade unos índices al crear las dimensiones, ya que, trabajar con índices es mucho más rápido que copiar todos los datos cada vez que se use `crossfilter` (es lento al escribir pero extremadamente rápido al leer). Las dimensiones se crean de la siguiente manera (`var dim = ndx.dimension(function(d) ... )`), siendo la variable `ndx` el conjunto donde están todos los datos (`var ndx = crossfilter(data)`), así tendremos todas las dimensiones que queramos. A cada dimensión que se cree se le añadirán unos índices, es decir, si se crea cinco dimensiones, habrá cinco índices, uno por cada dimensión. Con los índices incluidos en cada gráfica el repintado es mucho más fluido, por el hecho de que cada vez que se seleccione algo en una gráfica o en un panel, sólo se tiene que mirar en los índices que es un array mucho más reducido que los datos en sí.

Una vez creado la `dimension`, hay que crear el `group` que en su defecto (`var group = dim.group()`) nos agrupa según la dimensión que se ha creado y cada entrada suma uno. Pero si queremos una agrupación diferente, que siempre es útil tener alguna alternativa, se puede hacer así (`var group = dim.group().reduceSum(function(d) ... )`), muchas veces se tiene que usar ya que no siempre nos sirve la agrupación por defecto.

Este funcionamiento del `crossfilter` hace que sea imprescindible su uso y junto con dc.js que es una biblioteca de visualización hecha específicamente para que `crossfilter` este totalmente integrado.

5.4. Resultado final

Después de haber hecho todas las iteraciones necesarias, se ha obtenido el resultado esperado para este proyecto. Como se ha podido observar en la sección de funcionamiento y con las visualizaciones de diferentes tipos de gráficas para que se puedan filtrar en cada una de ellas,

dependiendo que datos se muestran y que seleccionar en una de ellas, el resto de gráficas se ajustan a lo que se ha seleccionado. Una tabla que muestra la información de los datos seleccionados o si no se ha seleccionado nada, se muestran todos. Para no colapsar el navegador sólo se podrán ver los veinte primeros, aunque si se baja hasta el final se cargarán cinco más, y así hasta que se muestren todos.
También incluye todos los filtros de las gráficas e incluso algunos más para que sea más intuitivo. Otra de las cosas que se puede hacer es seleccionar un rango fechas con un calendario (`datePicker`) o con una barra (`slider`), que nos dan el mismo resultado tanto en una como en otra y se cambian a la vez sin importancia por cual método se hayan introducidos el rango de las fechas.

En definitiva un resultado esperado aunque ya se está trabajando para agrandar el proyecto con ficheros JSON diferentes a las que se han usado y con más datos.

Capítulo 6

Conclusiones

El proyecto supone un intento por construir un conjunto de herramientas capaces de realizar de extraer datos de unos ficheros JSON y representarlos con posibilidades de filtrado de diferentes maneras, usando solo herramientas de software libre.

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

Así, se ha logrado crear una herramienta genérica para la visualización y filtrado de datos desde la fuente de información pública de OpenStack.

Por otra parte, el proyecto resultante de este procesos son alojados en GitHub, la herramienta que se ha utilizado para controlar las versiones.

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

6.1. Lecciones aprendidas

Fueron 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 se tendrá reparos en informar sobre fallos detectados en un programa, siendo consciente de la importancia para el bien común que supone el hacerlo.

- La exploración de las diferentes bibliotecas de visualización. Se adquirió cierta destreza en el aprendizaje de nuevas bibliotecas.
- La experiencia de trabajar en un proyecto de un tamaño considerable y el más grande hasta el momento.
- La autogestión del tiempo y el ritmo para que el proyecto este terminado en una fecha concreta.
- Mayor destreza en el uso de la herramienta github donde se encuentra alojado este proyecto y otros trabajos realizados con anterioridad.
- 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.

6.2. Conocimientos aplicados

Los conocimientos que se han aplicado para lograr con éxito este proyecto han sido mayormente las asignaturas de programación. Especialmente javascript y JQuery que es la base de este trabajo y luego para la maquetación HTML5, CSS3.

Y lo más importante es en la parte de exploración de bibliotecas. Un método de aprendizaje bastante útil para la videl del ingeniero en el mercado laboral. Ser autodidácta es lo más importante una vez terminada la carrera.

Esta es la primera prueba de realizaci'on de un proyecto semejante a lo que podría haber en una empresa real, aunque claro esta, sin tanta presión.

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.

Una posible mejora es optimizar la descarga de datos, que esta sea más rápida. Así se podrían trabajar con datos más grandes, en definitiva un producto mejor.

Resultaría también interesante que el proyecto tenga una plataforma, de esta manera podremos guardar el estado actual del filtrado con una url única y así se podría seguir el trabajo en otro instante o simplemente reproducirlo en otro equipo. Otra cosa que se podría hacer y en este ámbito es exportar el resultado restante en formato JSON y de esta manera enviarse de una manera rápida.

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

Bibliografía

[1] JavaScript.

http://es.wikipedia.org/wiki/JavaScript

http://www.w3schools.com/js/default.asp

http://librosweb.es/libro/javascript/

[2] HTML.

http://es.wikipedia.org/wiki/html

http://www.w3schools.com/html/default.asp

[3] CSS.

http://www.w3schools.com/css/default.asp

http://es.wikipedia.org/wiki/css

http://librosweb.es/libro/css/

[4] JQuery.

https://jquery.com/

http://www.w3schools.com/jquery/default.asp

http://librosweb.es/libro/fundamentos_jquery/

[5] Bootstrap.

http://librosweb.es/libro/bootstrap_3/

http://getbootstrap.com/

[6] D3.js.

https://github.com/mbostock/d3/wiki

[7] SVG.

http://es.wikipedia.org/wiki/Scalable_Vector_Graphics

http://www.w3schools.com/svg/

http://www.w3.org/Graphics/SVG/

[8] Crossfilter.

http://square.github.io/crossfilter/

http://blog.rusty.io/2012/09/17/crossfilter-tutorial/

[9] Dc.js.

http://dc-js.github.io/dc.js/

[10] Highcharts.

http://www.highcharts.com/

[11] Chart.js.

http://www.chartjs.org/

[12] Freeboard.

https://github.com/Freeboard/freeboard

[13] Nvd3.

http://nvd3.org/

[14] C3.js.

http://c3js.org/

[15] GGobi.

http://www.ggobi.org/

[16] Tulip.

http://tulip.labri.fr/TulipDrupal/

[17] GitHub.

https://github.com

[18] Bitergia.

http://bitergia.com/

[19] Grimoire.

`http://metricsgrimoire.github.io/`

[20] 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`

[21] Gregorio Robles, Stefan Koch, Jesús M. González-Barahona. *Remote analysis and measurement of libre software systems by means of the CVSAnalY tool*.

`http://libresoft.dat.escet.urjc.es/html/downloads/cvsanaly-icse.pdf`

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

`http://gsyc.urjc.es`

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

`http://libresoft.urjc.es`