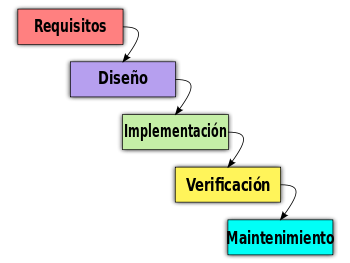
Como objetivo final en primera instancia se estableció la creación de un dashboard dinámico en el que pudiéramos elegir los datos que fuéramos a utilizar a la hora de crear nuestro escritorio personalizado de datos. Hasta la fecha existían diversos dashboards pero estos trabajaban con un tipo de datos muy específico, e impedían la personalización de los resultados obtenidos. Una herramienta útil y aplicable al desarrollo del software que nos permitiría realizar un trabajo interesante y ameno.

Antes de ponernos manos a la obra se decidió por consenso que el modo y forma de trabajo, se basaría en el método cascada.

* Modelo cascada:

En [Ingeniería de software](http://es.wikipedia.org/wiki/Ingenier%C3%ADa_de_software) el desarrollo en cascada, también llamado modelo en cascada (denominado así por la posición de las fases en el desarrollo de esta, que parecen caer en cascada *“por gravedad”* hacia las siguientes fases), es el enfoque metodológico que ordena rigurosamente las etapas del [proceso para el desarrollo de software](http://es.wikipedia.org/wiki/Proceso_para_el_desarrollo_de_software), de tal forma que el inicio de cada etapa debe esperar a la finalización de la etapa anterior. Al final de cada etapa, el modelo está diseñado para llevar a cabo una revisión final, que se encarga de determinar si el proyecto está listo para avanzar a la siguiente fase. Este modelo fue el primero en originarse y es la base de todos los demás modelos de ciclo de vida.



Acordado dicho modelo se procedió a la búsqueda de los requisitos que se esperaban para la primera fase:

-Uso de Javascript.

-Uso de jqueryUI

-Búsqueda de bibliotecas gráficas de representación para los datos

-Creación de un nuevo proyecto en Github

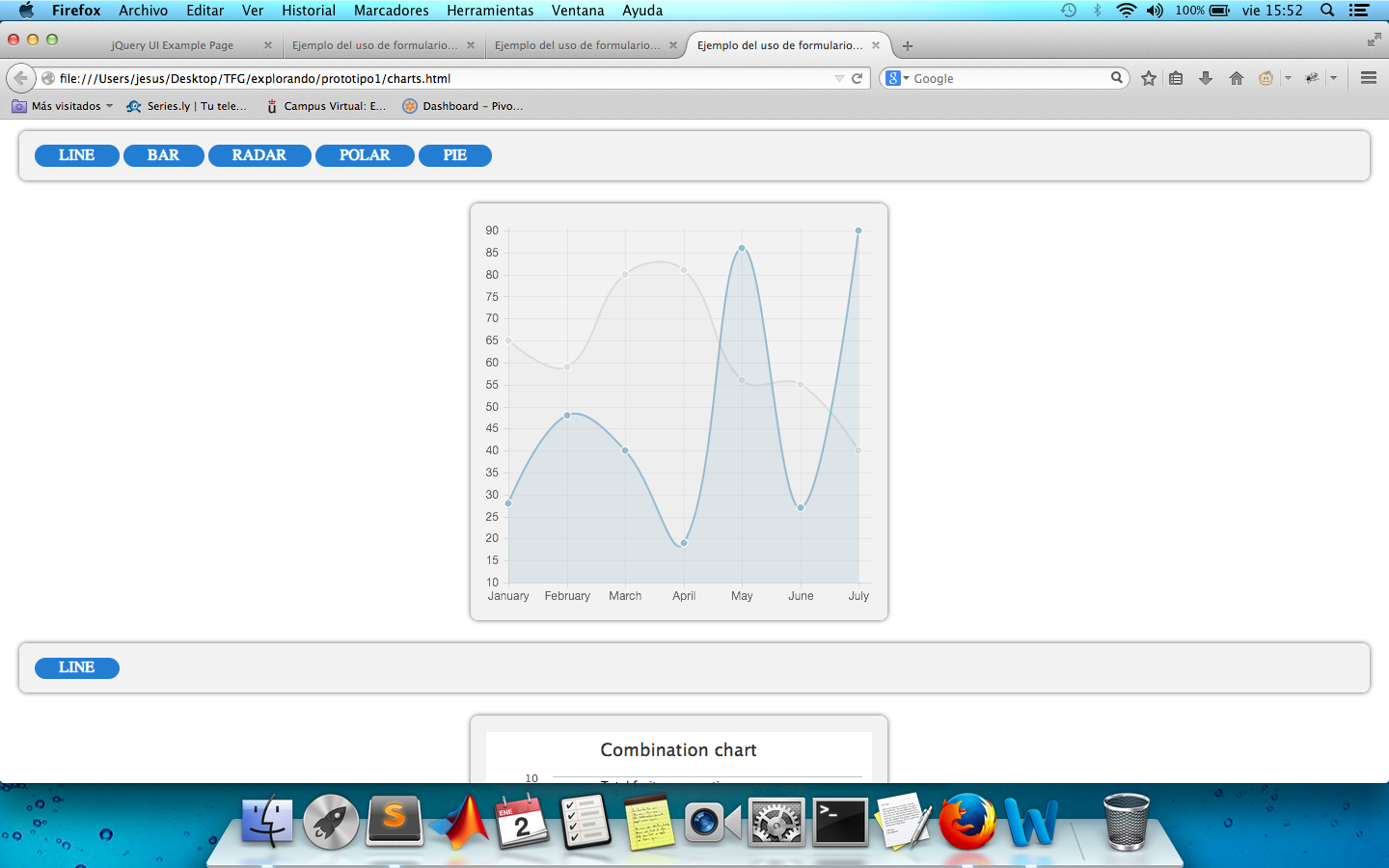
-Uso y experimentación con todo lo anterior para pasar a utilizar los verdaderos datos.

Extras:

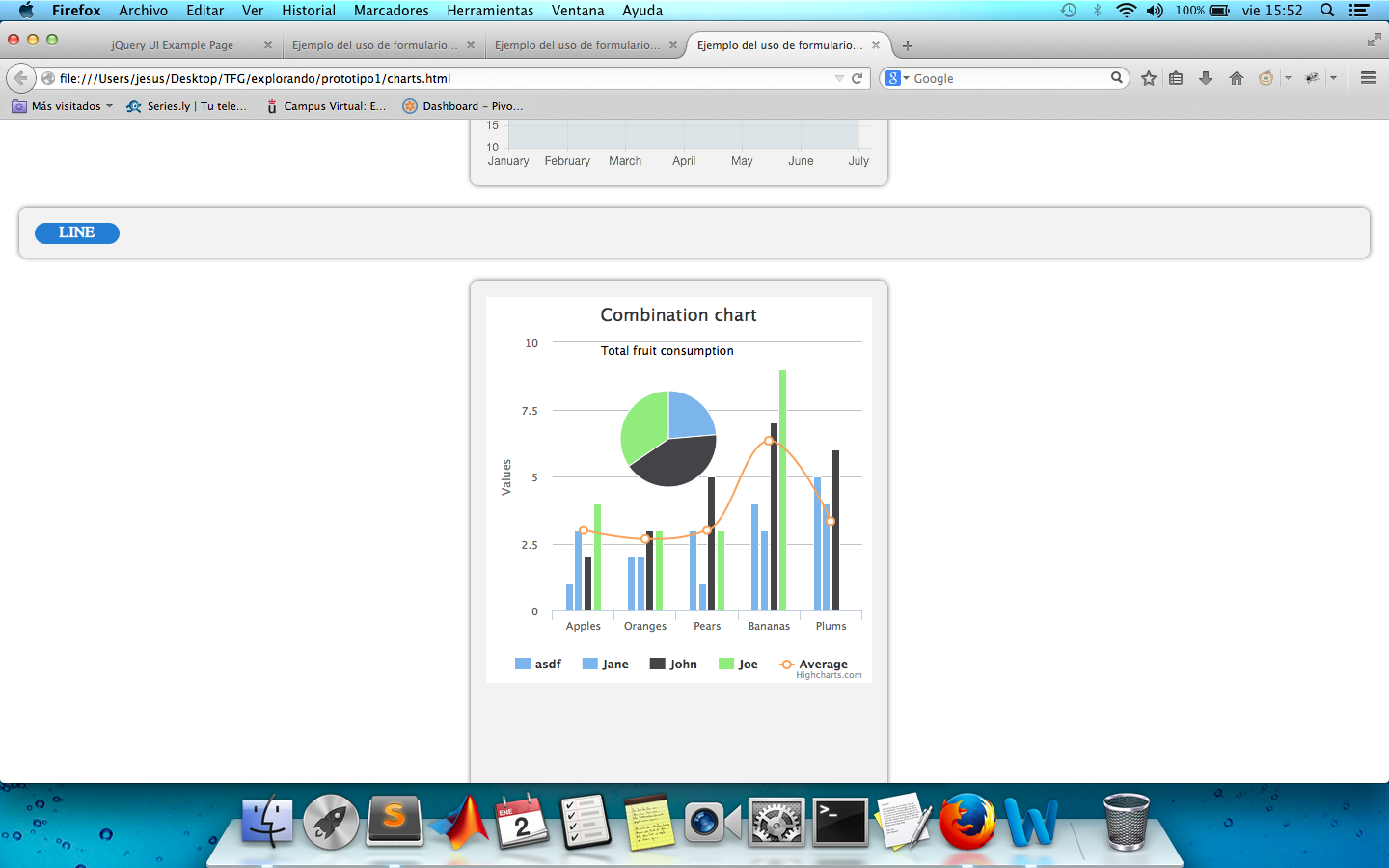
-Creación de un proyecto en pivotal traker.

Iniciando el ciclo se decidió empezar por la experimentación. El número de bibliotecas exploradas fue bastante extenso debido a la gran variedad que ofrecía la red, en especial desarrolladas en el entorno de Github, y cuyo software era libre. La mayoría de estas bibliotecas ofrecían unos usos muy específicos y se trataba de un entorno poco flexible en el que el trabajo podría convertirse en algo tedioso debido a la complejidad de las mismas, la falta de documentación y la poca forma en cuanto al trabajo realizado con estas. De las mismas se hizo una basta selección para probar con ellas con el objeto de conseguir discernir cuales de estas podrían adaptarse mejor a lo que buscábamos. Como se puede ver en la siguiente figura, se hicieron varias pruebas, en este caso con charts.js, mirando sus posibilidades de representación pero sobre todo la versatilidad del código que ofrecía para adaptarlo a nuestras necesidades.

Cabe destacar que lo importante no era en sí la representación de los datos, si no dar la oportunidad de una vez creada la plataforma y estructura de los objetos en nuestra jerarquía personal, que cualquier usuario interesado pudiera crear sus propios widgets enriqueciendo así el trabajo realizado.



Finalmente y debido a su gran flexibilidad a la hora de representar datos de distinta naturaleza, la posibilidad de acceder a los mismos datos con relativa sencillez y a las facilidades que ofrecía documentación se decidió hacer uso de highcharts.js . Como añadido la biblioteca no sólo permitía la creación de gráficas específicas si no que también permitía la posibilidad de la creación de gráficas que mezclaban propiedades de otras muchas como se puede comprobar en la siguiente imagen:



El uso de una biblioteca que permitiera la creación de una gráfica totalmente personalizada, cuya creación en el código sólo dependiera de un array de datos la hacía perfecta. Además de este tipo de gráficas highcharts.js ofrece numerosos tipos extras de gráficas, en cuanto a código algo más específicas, poco flexibles y cerradas, pero que también podrían suponer un aliciente en cuanto a la representación de los datos.

A diferencia de otras bibliotecas, esta en concreto daba la oportunidad de tener un acceso sencillo y limpio a los datos. La personalización de la representación de datos a priori y posteriori suponía un aliciente en cuanto al interés de la misma teniendo en cuenta que además podíamos saber con facilidad qué estábamos representando una vez la gráfica estuviera construida.

A continuación podemos ver un fragmento de código que permite crear este tipo de gráficas:

$(function () {

$('#container').highcharts({

title: {

text: 'Combination chart'

},

xAxis: {

categories: ['Apples', 'Oranges', 'Pears', 'Bananas', 'Plums']

},

labels: {

items: [{

html: 'Total fruit consumption',

style: {

left: '50px',

top: '18px',

color: (Highcharts.theme && Highcharts.theme.textColor) || 'black'

}

}]

},

series: [{

type: 'column',

name: 'Jane',

data: [3, 2, 1, 3, 4]

}, {

type: 'column',

name: 'John',

data: [2, 3, 5, 7, 6]

}, {

type: 'column',

name: 'Joe',

data: [4, 3, 3, 9, 0]

}, {

type: 'spline',

name: 'Average',

data: [3, 2.67, 3, 6.33, 3.33],

marker: {

lineWidth: 2,

lineColor: Highcharts.getOptions().colors[3],

fillColor: 'white'

}

}, {

type: 'pie',

name: 'Total consumption',

data: [{

name: 'Jane',

y: 13,

color: Highcharts.getOptions().colors[0] // Jane's color

}, {

name: 'John',

y: 23,

color: Highcharts.getOptions().colors[1] // John's color

}, {

name: 'Joe',

y: 19,

color: Highcharts.getOptions().colors[2] // Joe's color

}],

center: [100, 80],

size: 100,

showInLegend: false,

dataLabels: {

enabled: false

}

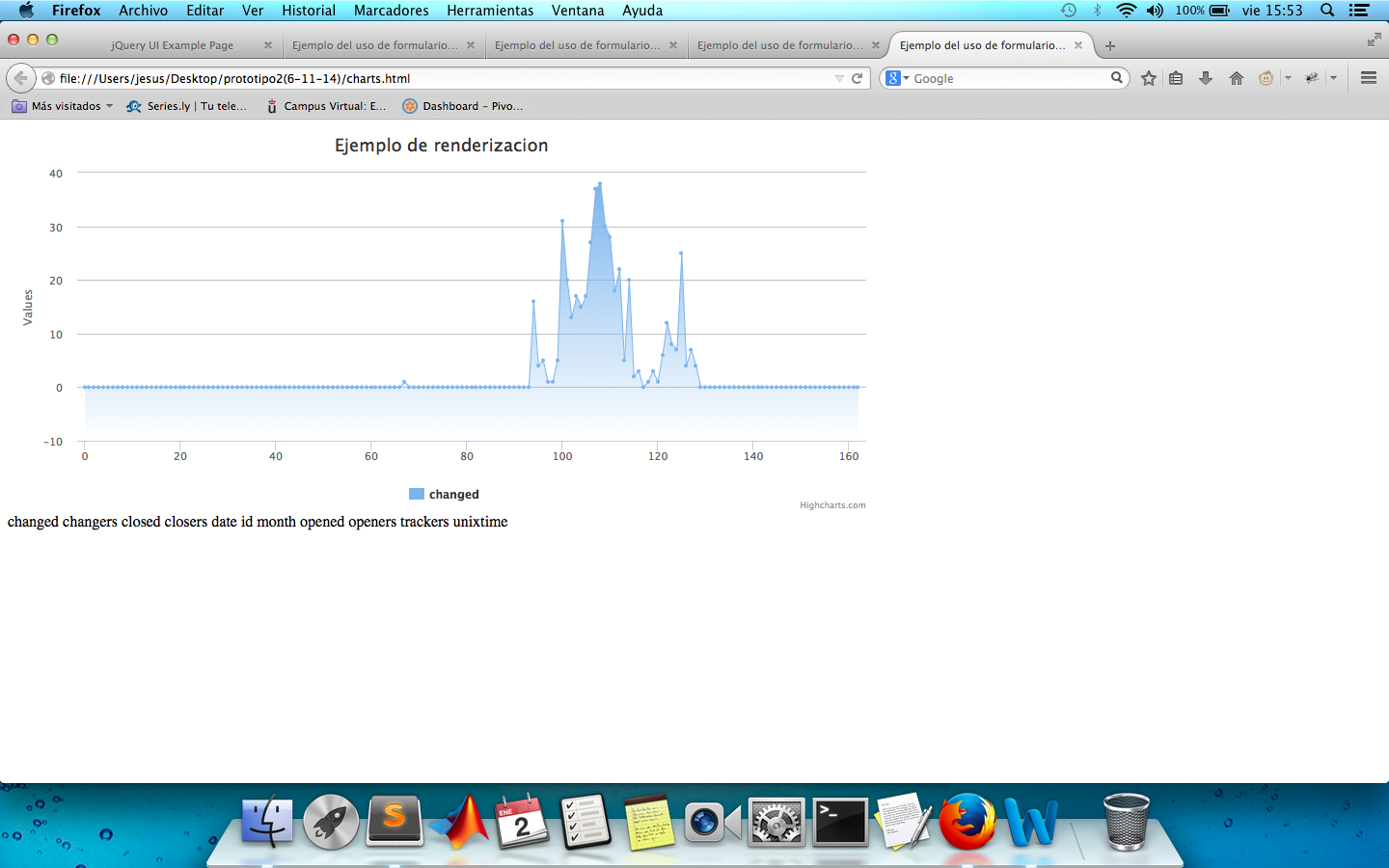
}]

});

});

Como se puede comprobar en el código cada elemento añadido en el chart permite ser incluido mediante el array “series” dentro del cual podemos añadir desde un “pie” hasta un formato de datos estándar formado por un simple array de números.

Highcharts.js por su parte también permite la correcta representación de grandes cantidades de datos a diferencia de otras bibliotecas y dashboards de código abierto y adaptado a nuestra página web, pero fijos en cuanto a funcionalidad. Así se puede ver en la siguiente figura, donde demuestra su flexibilidad a la hora de adaptarse a los datos impidiendo la superposición de los mismos si se trata de archivos con una gran cantidad de valores.



Concluida la parte de búsqueda de biblioteca y seleccionada highcharts para nuestros propósitos el siguiente paso era poner en práctica lo aprendido en la red y buscar la forma de recrear una primera gráfica personalizada con el objetivo de empezar a crear nuestro dashboard. Para ello es necesario trabajar con los ficheros json reales. En este punto era importante conocerlos a fondo debido a que su formato sería importante en cuanto a la selección de datos para la representación de los mismos.

Eso si, se debía tener en cuenta que a la hora de escribir el código las funciones de filtrado de datos debían estar debidamente divididas, siendo estas un ente independiente del resto de la funcionalidad del código. Esto debió hacerse de este modo debido que ya en este punto se barajaba con la posibilidad de la creación de nuestro propio formato de seguimiento de datos debido a que no existía un estándar equitativo entre todos los jsons existentes de distintas empresas.

Navegando por internet se descubrió que la mayoría de las empresas que hacen uso de esta clase de recursos para hacer estudios de mercado sobre sí mismas y saber los beneficios que obtienen durante un periodo de tiempo hacían caso omiso a los datos de otras empresas más allá de sus resultados finales dónde se podían comparar directamente y saber en qué posición de la competición se encontraban. Esta situación daba lugar a lo ya mencionado de la existencia de jsons de distinto formato para la creación de gráficas que nos permitan hacer un seguimiento exhaustivo acerca de un proyecto.

Por otro lado, dada la complejidad de los datos con los que se estaba trabajando suponía que la representación de nuestra gráfica no dependerá de un único fichero, si no que dependerá de varios, los cuales pueden proceder de distintas fuentes, y que además el programa debe saber administrar para ofrecer al usuario una experiencia sencilla y amena.

En la siguiente figura se puede observar un primer prototipo de dashboard el cual permite la representación personalizada de los datos de tres empresas distintas. Permite la selección personalizada de la representación de los propios datos. Es aquí dónde hablábamos de la flexibilidad de highcharts para permitir representar datos distintos de maneras distintas sobre el mismo widget. Por otro lado y pese a que en la figura no se muestre highcharts ofrece algunos extras para poder descargar la gráfica, imprimirla o exportarla en el formato que nosotros deseemos.

Por otro lado permite la interactuación con los datos permitiendo seleccionar dentro de os datos ya existentes en la gráfica cuál queremos tener visible.

