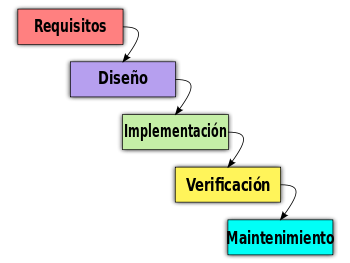
Como objetivo final en primera instancia se estableció la creación de un dashboard dinámico en el que pudiéramos elegir los datos que fuéramos a utilizar a la hora de crear nuestro escritorio personalizado de datos. Hasta la fecha existían diversos dashboards pero estos trabajaban con un tipo de datos muy específico, e impedían el desplazamiento 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 ponernos de acuerdo en el modo y forma de trabajo, se usaría 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.[1](http://es.wikipedia.org/wiki/Desarrollo_en_cascada#cite_note-1) 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

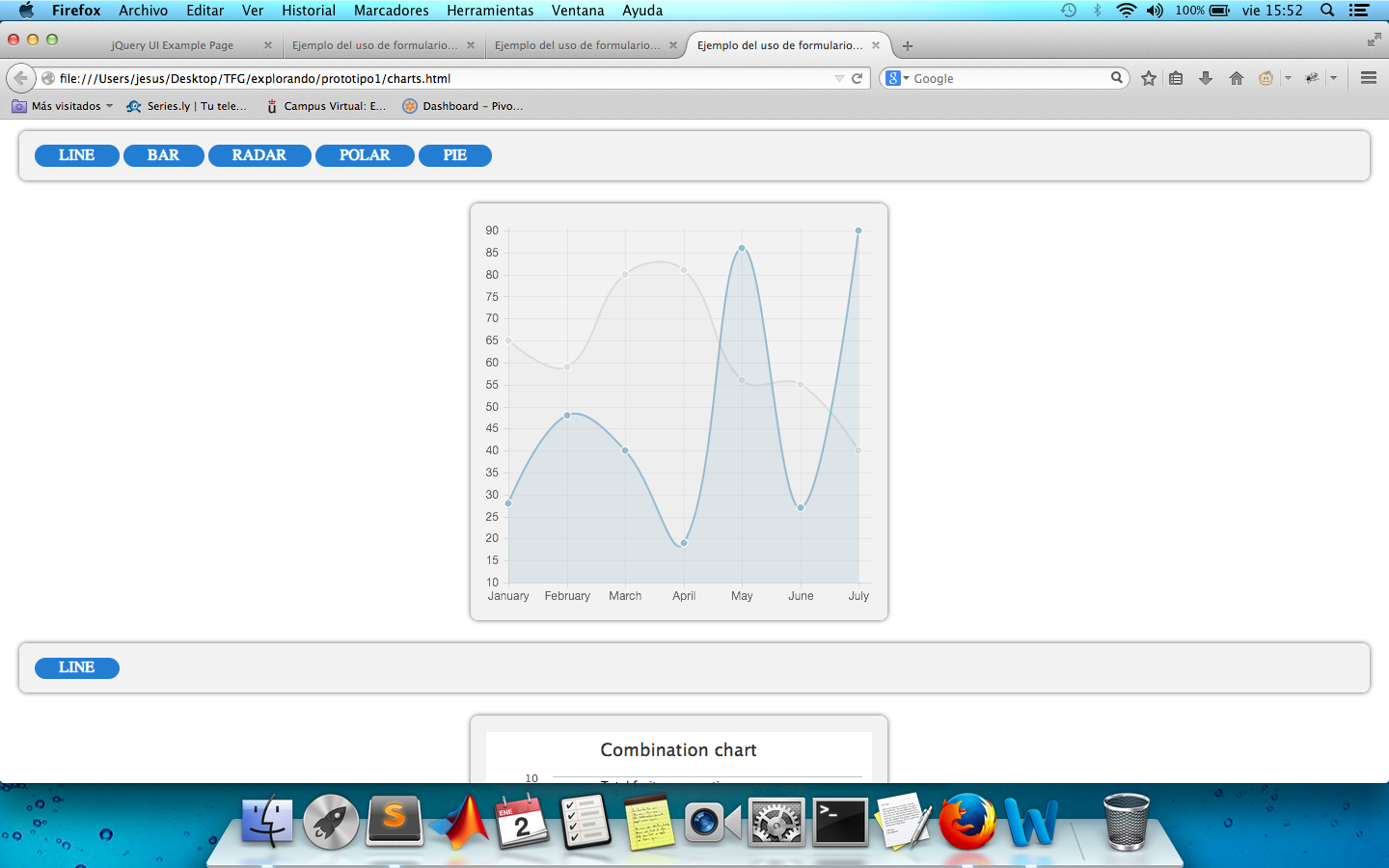
-Creación de un nuevo proyecto en Github

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

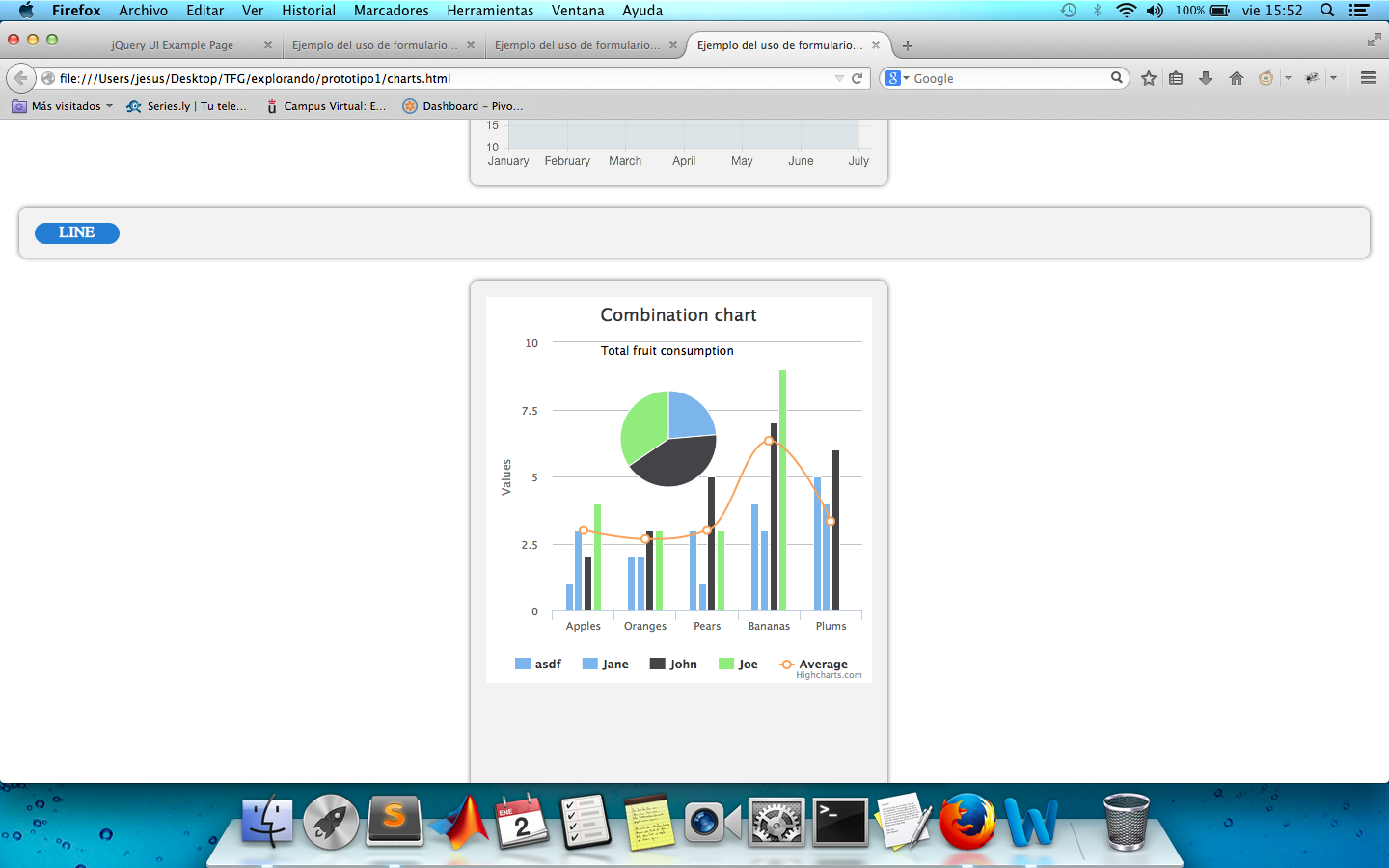
Extras:

-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 en el entorno de Github. 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 experimentar y probar con ellas con el objeto de conseguir discernir cuales de estas podrían adaptarse mejor a lo que buscábamos.



Finalmente y debido a su gran flexibilidad a la hora de representar datos de distinta naturaleza, la posibilidad de acceder a los mismos datos y a la facilidad en cuanto a 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:



A continuación podemos ver un fragmento de código que permite crear este tipo de gráficas, las cuales son perfectas para la representar los datos conforme a lo que necesitemos y en el tipo que necesitemos:

$(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

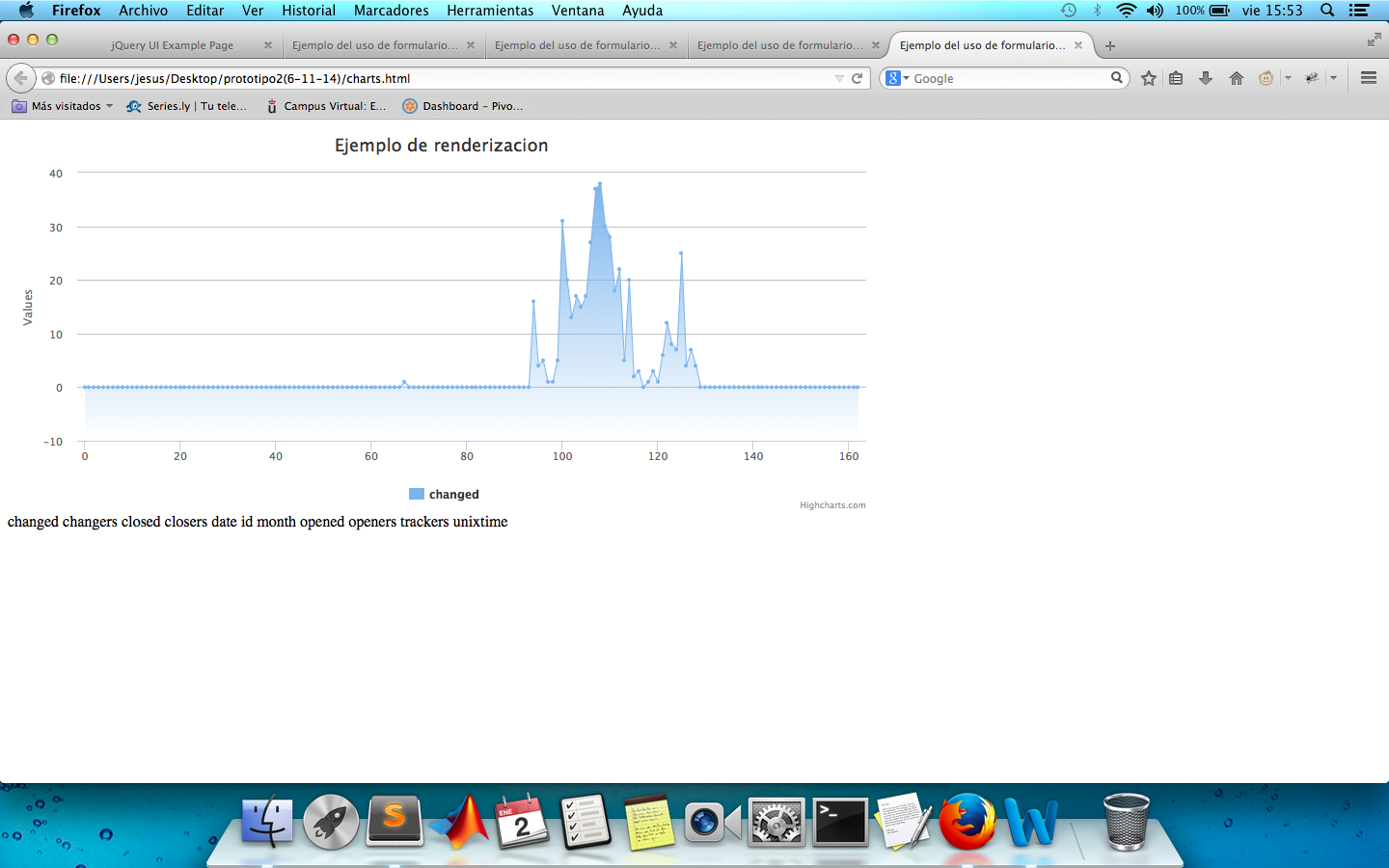
}

}]

});

});

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.



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 que los usuarios pueden utilizar y aprender cómo seleccionar tanto los índices como sus valores por separado.

