Conociendo ya una primera visualización de nuestra aplicación, y viendo que la creación de las gráficas funcionaba correctamente se procedió a la ampliación del número de funcionalidades que aportaba el entorno.

De esta manera se mejoró ligeramente la visualización del gridster y los widgets, para intentar pasar al siguiente ciclo, la implementación de un servidor, guardado y aplanamiento de las gráficas y la reproducción de las mismas si se quiere acceder de nuevo a ese entorno de trabajo. Este pequeño paso intermedio era importante para hacerle un uso semejante al de traza con el que discernir durante su programación si el funcionamiento era correcto o no.

Para lograr esto se procedió al uso de django y python. Un servidor a través del cual se añadirían las urls y del cual se haría uso de sus bases de datos en primera instancia para poder hacer las pruebas pertinentes. Véase que si en un futuro se quisiera subir la aplicación a un entorno web permanente existen herramientas que nos pueden hacer la vida más fácil pero la mayoría de ellas no hacen uso de las bases de datos sqlite de las que hace uso django por defecto.

Es de este modo que se presenta la siguiente organización de urls que nos permitirán interactuar con nuestra aplicación:

A vista del usuario:

../ : Dashboard vacío, página home desde el que empezar a utilizar nuestra aplicación

../\d+: Dashboard ya creado que se carga para seguir modificándolo a nuestro gusto

Desde el punto de vista del servidor:

../:

Get:

Dashboard vacío, página home desde el que empezar a utilizar nuestra

/db/:

Get:

Listado de ids de los dashboards creados

Post:

Creación de un nuevo dashboard.

/db/\d+:

Get:

Carga de un dashboard ya creado, devuelve el json de configuración del mismo.

Put:

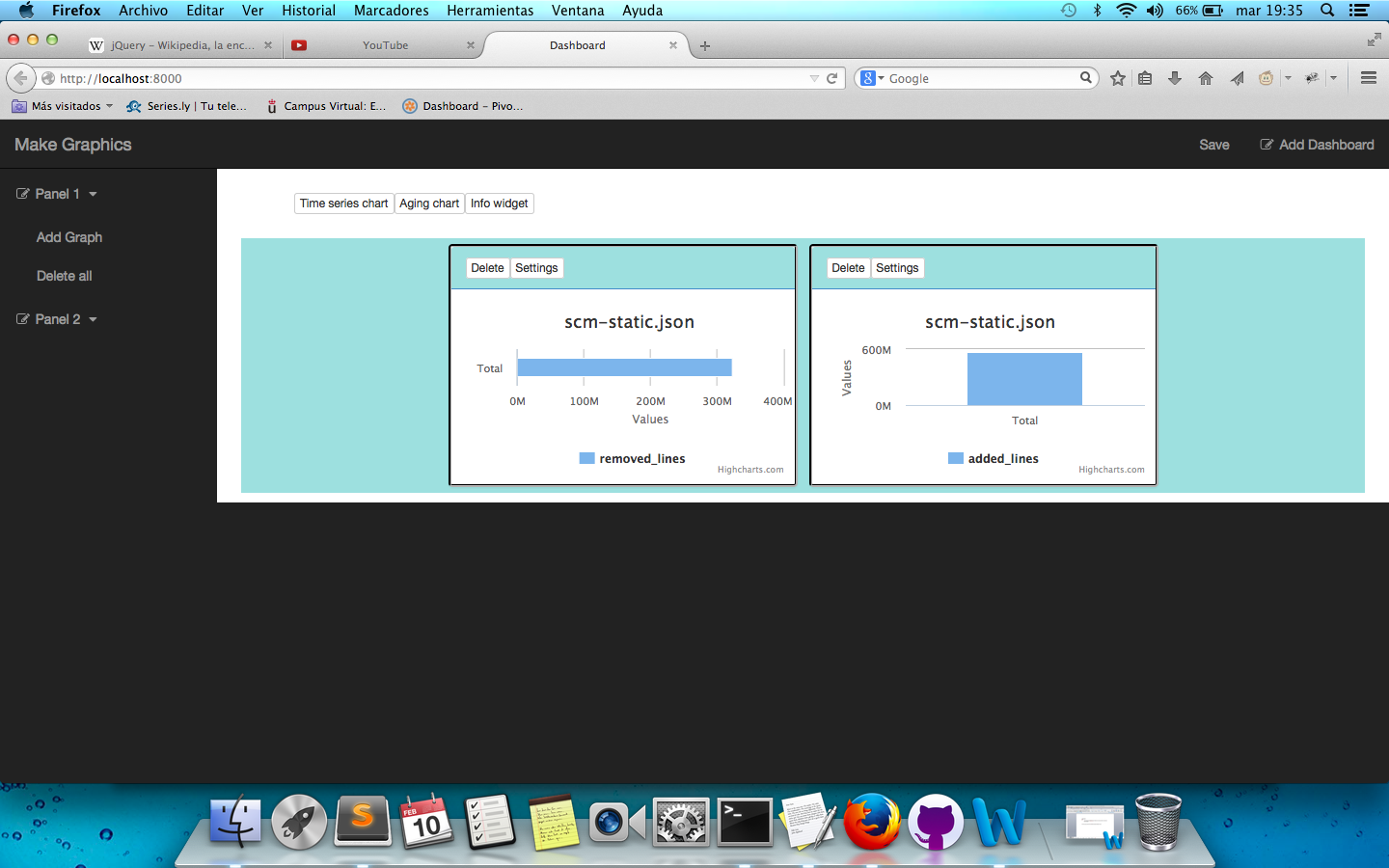
Sobreescritura de un dashboard ya creado con una nueva configuración.

/db/default:

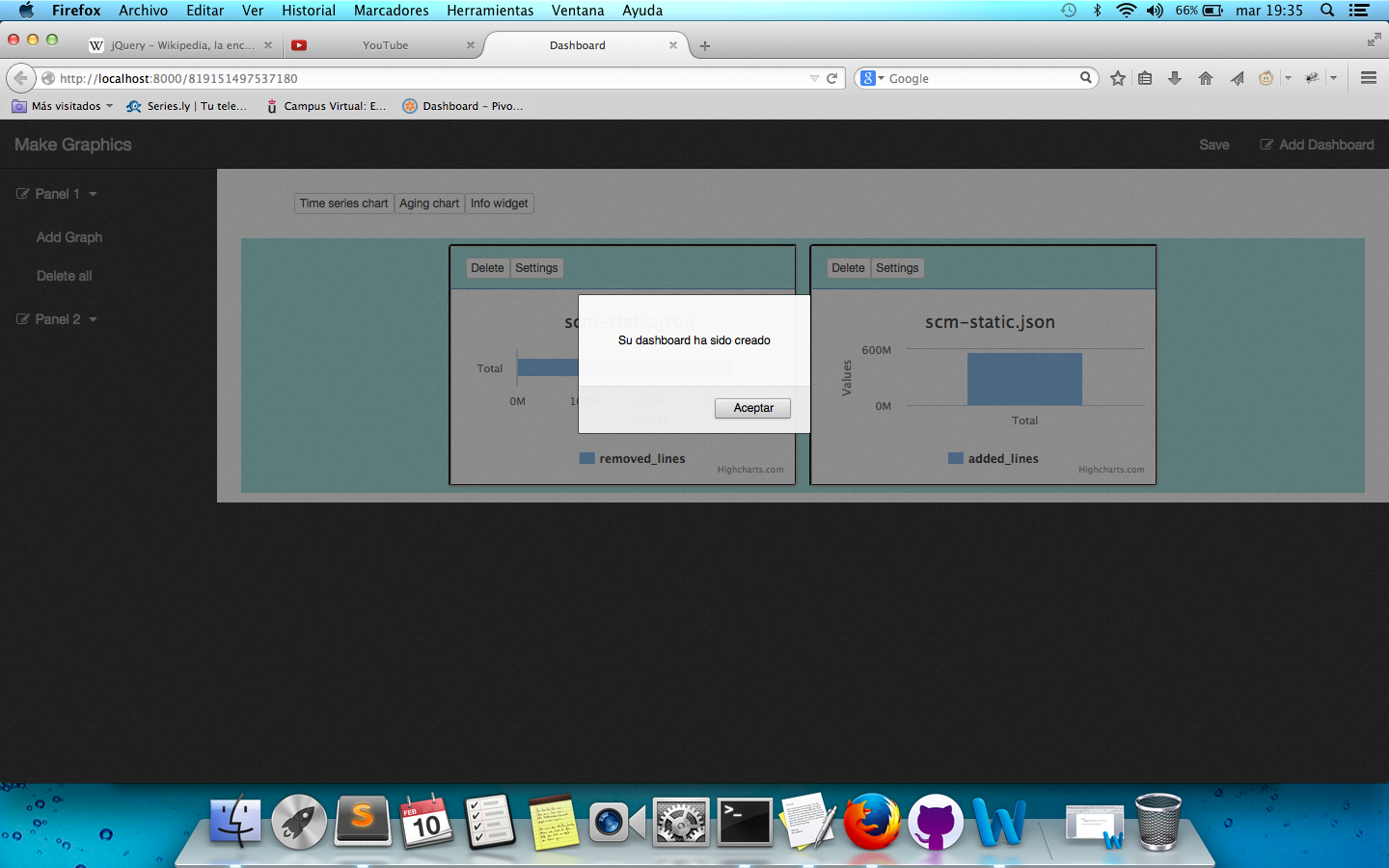
Get:

Presentación del dashboard por defecto con el que mostrar al usuario la funcionalidad.

Con esta nueva estructura tendríamos algo semejante a esto:



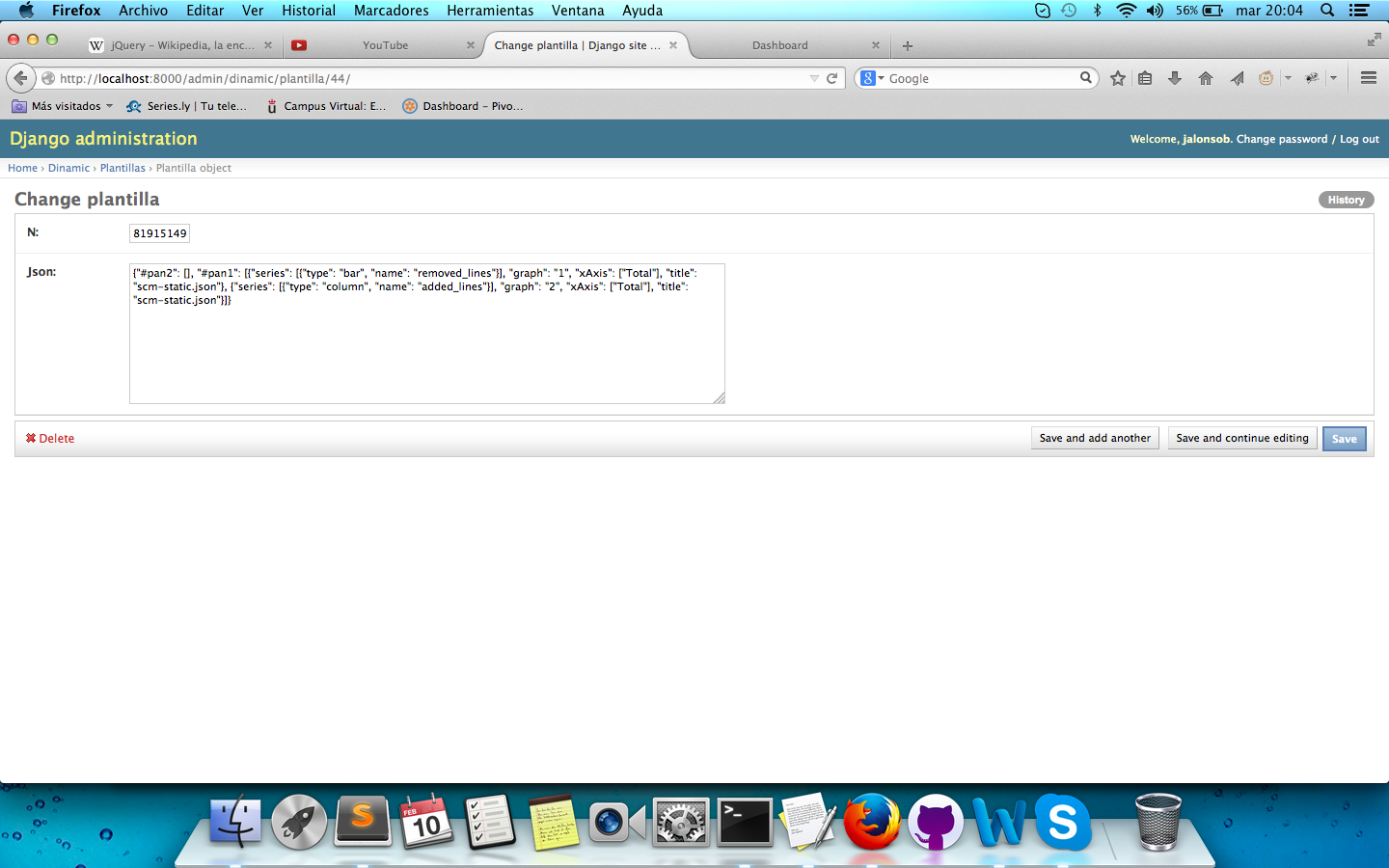
Como observamos en la figura anterior hemos creado dos gráficas básicas sobre un dashboard vacío aún no creado. Las aplicación permite el guardado de las mismas mediante el botón “Save” situado arriba a la derecha. Al pulsarlo…



Se creará un nuevo dashboard en nuestro servidor, un mensaje aparecerá en pantalla para avisarnos de tal evento.

Como podemos ver en la segunda imagen la url en la que estamos posicionados ha cambiado conforme hemos guardado, situándonos ya en el lugar correspondiente de trabajo.

Si entramos en el funcionamiento del fichero de configuración tendríamos actualmente algo de este formato.



Cada dashboard está identificado por un número único N y por un json de configuración. Dicho json es un objeto cuyas claves son los paneles que dibujamos, y cuyo contenido son arrays. Dentro de cada posición de los arrays tenemos objetos con las características necesarias para guardar debidamente una gráfica. El título, las métricas y su formato así como el identificador de gráfica son elementos importantes a la hora de guardarlo dentro del servidor para posteriormente reproducirlo debidamente.

Uno de los mayores problemas que existen al hacer uso de una aplicación que necesita de peticiones de datos de gran envergadura es el tiempo de carga con respecto al tiempo de espera de los datos. Por eso fue necesaria una reestructuración completa del código orientada a la espera de los datos para después hacer uso de los mismos. Es decir, pido los datos si son necesarios, si están pedidos espero a recibirlos, si no los recibo lanzo un evento orientado a un error en la descarga, y si ha resultado correcta la petición pinto la gráfica sobre la que estaba trabajando y las subsiguientes que tal vez dependían del mismo tipo de json.

Establecido nuestro servidor, nuestro guardado y reproducción de charts, pasamos al siguiente ciclo donde puliremos algunos defectos que actualmente posee nuestra aplicación.