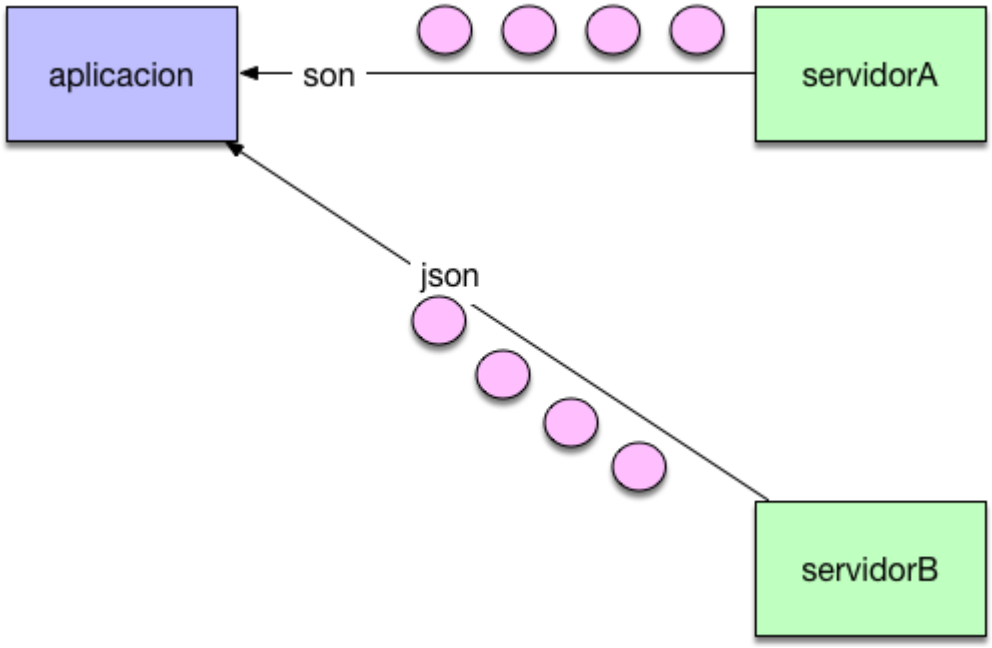
El uso de Google Chrome Speed Test para revisar el rendimiento de nuestras aplicaciones es cada día más común. Hoy por hoy muchas de las aplicaciones que construimos adquieren un montón de información en formato JSON y muchas de ellas necesitan acceder a esa información desde situaciones en las que el ancho de banda puede estar limitado.



Es decir podemos tener un móvil que no tenga cobertura 4G sino una mala cobertura 3G o cosas bastante peores . ¿Como se va a comportar nuestra aplicación cuando adquiramos los datos? . Vamos a ver como configurar la herramienta de Google Chrome y poder valorar de una forma muy directa el rendimiento real que la aplicación tendrá en la descarga de datos para un ancho de banda determinado. Vamos a partir de un sencillo ejemplo en node.js que nos devuelve una estructura JSON de 5 mil registros muy sencillos con 4 campos.

```
var express = require('express');
var app = express();
```

```
var lista=[];
for (var i=0;i<5000;i++) {

  lista.push({"cif":"123456A",
  "nombre":"juan manuel"+i,
  "apellidos":"apellido 1 apellido2 "+i,
  "edad":30,
  "informe":"se trata de un cliente clasico con el cual"+
   "llevamos trabajando tiempo y necesita una actualizacion" })
}


var cors = require('cors')
app.use(cors())
app.get('/', function (req, res) {


  res.send(lista);
});


app.listen(3000, function () {
  console.log('Example app listening on port 3000!');
});
```

Si cargamos esto en un navegador veremos salir de forma instantanea los datos en la pantalla:

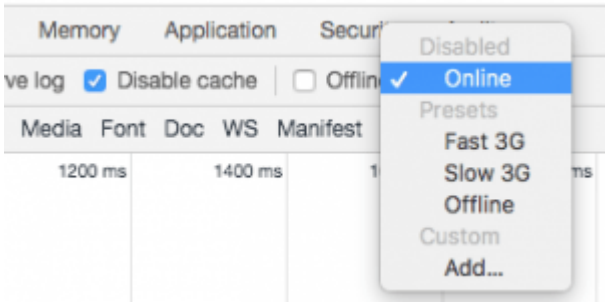
[{"cif":"123456A","nombre":"juan manuel0","apellidos":"ap
0","edad":30,"informe":"se trata de un cliente clasico co
tiempo y necesita una actualizacion"},{"cif":"123456A","n
manuel1","apellidos":"apellido 1 apellido2 1","edad":30,"
cliente clasico con el cual llevamos trabajando tiempo y
{"cif":"123456A","nombre":"juan manuel2","apellidos":"ape

No hay ningún problema y todo funciona correctamente . Si abrimos el inspector de chrome y nos situamos en la pestaña de red podremos ver una información a detalle de la petición que hemos hecho.

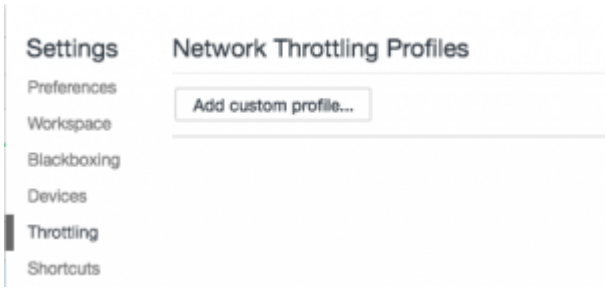
Name	Status	Type	Initiator	Size	Time
localhost	200	docum...	Other	1004 KB	843 ms
tpc-check.html	200	docum...	buffer-tpc-check.js:39	(from disk cache)	31 ms
buffer-hover-i...	200	png	Other	(from disk cache)	10 ms
tpc-check-ad...	200	script	chrome-extension://no...	(from disk cache)	5 ms
postmessage.js	200	script	chrome-extension://no...	(from disk cache)	5 ms

Google Chrome Speed Test

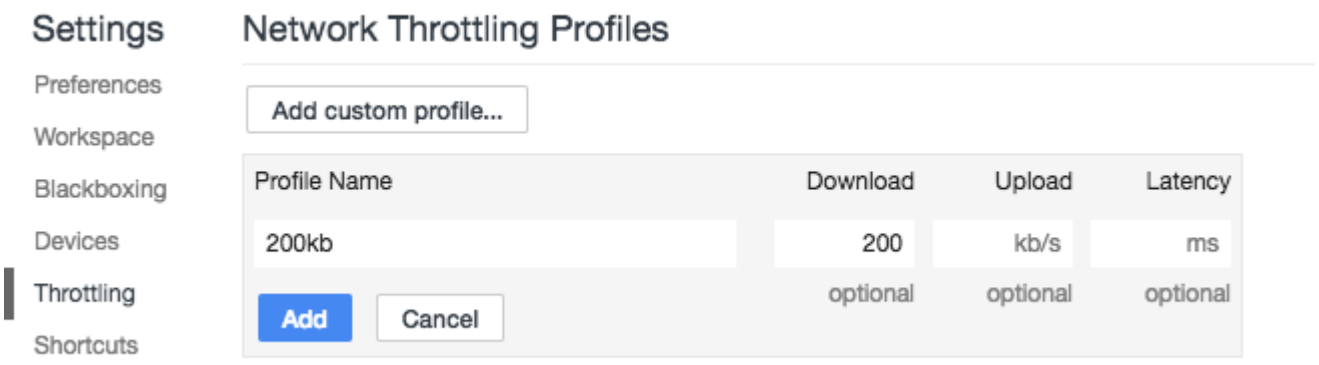
Los datos son bastante esclarecedores el fichero JSON ocupa 1mg de tamaño (1004 KB) y hemos tardado en cargarlo no llega a un segundo. Con esta información nos podemos hacer una idea de cuanto tardaría el fichero en cargarse en un navegador que por ejemplo usara una conexión de ADSL de 3 mg . La respuesta suele ser que también sería instantaneo ya que el fichero es de 1 mg . La realidad es que en ADSL por ejemplo solo nos aseguran un 10% del caudal en un momento determinado. Así que es probable que no sea tan instantaneo como nosotros creemos ya que realmente nos aseguran solo 300 kb . Con esta medida nos podríamos hacer una idea de lo que la aplicación tarda . Sin embargo hay que recordar que Google Chrome es muy flexible y en la pestaña de red nos permite definir la velocidad que deseamos simular para el navegador (opción de online).



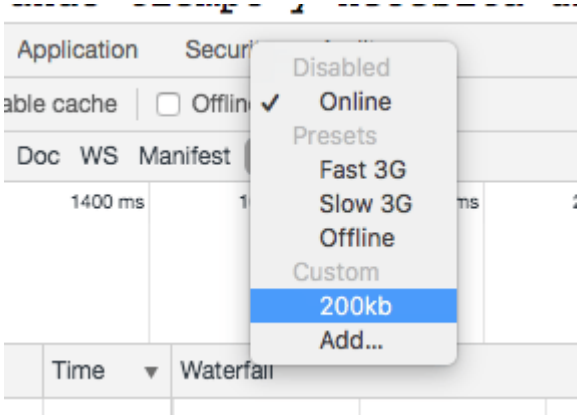
Rapidamente aparecen las opciones de 3G como las más comunes . Sin embargo tenemos la opción de pulsar sobre add y añadir una a medida.



Rellenamos la información , yo en este caso voy a marcar 200kb en vez de 300 para cubrirme las espaldas en el funcionamiento del mínimo del ADSL.



Volvemos a la pestaña de network principal y ya lo podremos seleccionar:



Volvemos a cargar nuestro fichero JSON con este ancho de banda nuevo tan maravilloso y veremos como la descarga de datos se eterniza. Aprendamos a utilizar este tipo de herramientas como google chrome speed test para poder valorar de mejor forma el ancho de red que disponemos.

1. JSON Viewer Awesome y chrome
2. JSON-P y Java Enterprise Edition 8
3. Java EE JSON y procesamiento (JSR 353)
4. Java y JSON sencillo
5. Google Chrome Tools