DESPLIEGUE DE SERVICIOS MULTIMEDIA
Máster Universitario en Ingeniería de Telecomunicación

AppGaztaroa

Commit 08: "Servidor JSON"

Marko Galarza Galarza
Mikel Sagues García

Servidor JSON

Commit 08: "Servidor JSON"

Este ejercicio tiene como objetivo instalar un servidor REST API local que nos permita simular un servidor real, sin necesidad de complicarnos demasiado con la configuración del mismo.

Emplearemos dicho servidor para obtener toda la información que hasta ahora recabábamos de ficheros JavaScript locales. De este modo, los siguientes ficheros ya no serán necesarios en nuestra aplicación:

- *comun/excursiones.js*
- *comun/cabeceras.js*
- *comun/actividades.js*
- *comun/comentarios.js*

Todos ellos, serán sustituidos por el archivo *db.json* que se encuentra adjunto a este ejercicio y en el que podemos encontrar la misma información. En todo caso, esto es algo que haremos en el siguiente ejercicio, ¡por lo que **no debéis borrarlos todavía**!

Además, el servidor JSON se encargará también de almacenar y enviar las imágenes que hasta ahora obteníamos desde la carpeta "*componentes/imagenes*". A partir de ahora, esta carpeta únicamente almacenará el logotipo de Gaztaroa (*logo.png*), entendiendo que éste no varía con el tiempo y que, en consecuencia, merece la pena que se encuentre entre los ficheros nativos de los que hará uso la aplicación.

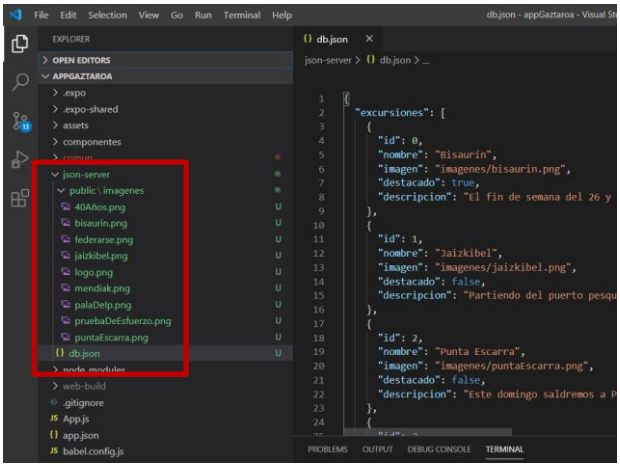
1. Simple JSON Server

En primer lugar, instalaremos el siguiente módulo global, lo que nos permitirá "lanzar" el servidor JSON desde cualquier directorio de nuestro ordenador[1]:

npm install -g json-server@0.17.4

A continuación, creamos la carpeta *json-server* en el directorio principal de nuestra aplicación y guardamos en dicha carpeta el fichero *db.json* que se adjunta al ejercicio. Además, creamos una carpeta de nombre *public* y dentro de ella otra carpeta de nombre *imagenes*, en la que copiaremos todas las imágenes a las que hasta ahora la aplicación accedía desde la carpeta *componentes/imagenes.* En definitiva:

[1] La última versión estable a día de hoy: json-server --version -> 0.17.4



Para lanzar el servidor JSON debemos:

- Conocer la dirección IP (Wifi) de nuestro ordenador. En este ejemplo, imaginemos que la IP del ordenador es la siguiente: 192.168.1.109.
- Abrir un segundo terminal en *Visual Studio Code* (emplearemos el primer terminal para lanzar la aplicación React Native y el segundo para lanzar el servidor JSON).
- Navegar en dicho terminal hasta la carpeta *json-server* que acabamos de crear y ejecutar la siguiente instrucción:

json-server --host 192.168.1.109 db.json -p 3001 -d 2000

De este modo, el servidor hará accesible para nosotros el contenido del fichero *db.json* a través de una interfaz REST. Por lo tanto, los recursos a los que queremos acceder estarán disponibles en (ver fichero *db.json*):

- http://192.168.1.109:3001/excursiones
- http://192.168.1.109:3001/comentarios
- http://192.168.1.109:3001/cabeceras
- http://192.168.1.109:3001/actividades

Además, *json-server* nos permite servir, de manera estática, cualquier contenido que albergue la carpeta *public*. Por lo tanto, las imágenes podrán descargarse desde[2]:

- http://192.168.1.109:3001/imagenes/40Años.png

2. Cambiar acceso a imágenes y colores

En este segundo apartado del ejercicio, modificaremos el código de nuestra aplicación para que acceda de manera centralizada a los colores y para que acceda a las imágenes (excepto *logo.png*) a través del servidor que acabamos de configurar.
En definitiva, haremos uso del servidor estático que nos proporciona *json-server*, dejando para el siguiente ejercicio el acceso a la REST API.

[2] Se muestra el caso de la imagen "*40años.png"*, pero el acceso al resto se hará de manera análoga.

Como primer paso, crearemos un fichero de nombre *comun.js* dentro de la carpeta *comun* en el que definiremos una serie de constates a las que después accederemos desde nuestra aplicación. El código de dicho fichero es el siguiente:

```
export const baseUrl = "http://192.168.1.109:3001/";
export const colorGaztaroaOscuro = '#015afc';
export const colorGaztaroaClaro = '#c2d3da';
```

De este modo, no es necesario escribir en diferentes puntos de nuestro código un dato "variable" como es la dirección IP de nuestro ordenador (esta cambiará cada vez que cambiemos de red wifi). También almacenaremos los dos colores que empleamos para configurar nuestra aplicación, de tal forma que estos puedan ser modificados de forma centralizada.

Para que nuestra aplicación reconozca las constantes definidas en *comun.js*, debemos importarlas en cada uno de los ficheros en que deseemos utilizarla. Por ejemplo, en *CampobaseComponent.js* modificaremos el código de este modo para actualizar la forma en que cargamos el color de fondo del *Drawer Menu*:

```
import { colorGaztaroaClaro } from '../comun/comun';
[...]
backgroundColor: '#c2d3da',
backgroundColor: colorGaztaroaClaro,
```

Una vez creado dicho fichero, podríamos sustituir los accesos a imágenes que hasta ahora venían haciéndose mediante el método *require()*, para que sean importadas desde el servidor estático, ajustando además el código para que cada componente (*ListItem*¸ por ejemplo) muestre la imagen que le corresponde, atendiendo al fichero JavaScript desde el que se importa el resto de su información. Nótese que hasta ahora empleábamos la imagen *40Años.png* en todos los componentes. Por ejemplo, en el método *renderExcursion()* dentro de *DetalleExcursionComponent.js,* haríamos lo siguiente:

```
<Card.Image source={require('./imagenes/40Años.png')}>
<Card.Image source={{uri: baseUrl + excursion.imagen}}>
```

En los estilos de *DetalleExcursionComponent.js* también modificaremos el color de los títulos en las *cards*, cambiando de "chocolate" a "white".

En definitiva, para completar este ejercicio, deberemos:

- Modificar el código de toda la aplicación para que en lugar de ajustar el color de los menús con valores numéricos (#015afc y #c2d3da) se realice mediante las constantes definidas en *comun.js*.
- Modificar el código de toda la aplicación para que las imágenes se obtengan desde el servidor estático proporcionado por *json-server*. Además, será necesario seleccionar las imágenes programáticamente de tal forma que cada

componente (*ListItem*) haga uso de la imagen que le corresponde, atendiendo al fichero JavaScript desde el que se importa el resto de su información.

Una vez completada la tarea, la aplicación debe tener la apariencia que se muestra en el vídeo adjunto a este ejercicio.

Si todo es correcto, ya podemos hacer el *commit* correspondiente a este capítulo y subir a nuestro tablero de Trello un breve documento con las claves del trabajo realizado.

3. Bibliografía

- **JSON Server**
  https://github.com/typicode/json-server/tree/v0
- **JSON Server v1.x** no utilizado en este proyecto:
  https://github.com/typicode/json-server