**Implementando Nodemon a nuestro servidor**

Bien, después de haber creado un API sencilla en el anterior ejercicio, nos gustaría introducir una herramienta que nos permitirá ahorrar mucho tiempo a la hora de programar un servidor en Node.js. Esta herramienta se llama Nodemon.

Y resumidamente, Nodemon es una herramienta que monitorea cambios en nuestro código de JavaScript, si algún cambio sucede, Nodemon automáticamente reiniciara el servidor por nosotros, lo que lo convierte en una herramienta perfecta para desarrollar.

Como ejemplo, utilizaremos el código del anterior ejercicio, pero con el pequeño cambio de que tendremos comentado el código de la consulta individual, es decir, la ruta de /:idPendiente:

"use strict";

**const** express = require('express');

**const** app = express();

**const** port = 3000;

**const** host = '127.0.0.1';

**const** pendientesPorHacer = [

{id: 1, descripcion: "Comprar comida", completada: true},

{id: 2, descripcion: "Preparar comida", completada: false},

{id: 3, descripcion: "Pasear al perro", completada: true},

{id: 4, descripcion: "Estudiar React y VueJS", completada: false},

{id: 5, descripcion: "Terminar series pendientes", completada: false},

{id: 6, descripcion: "Reparar el celular", completada: true}

]

app.get('/', (req, res) **=>** {

return res.status(200).json(pendientesPorHacer);

})

app.get('/count', (req, res) **=>** {

**const** miRespuesta = {

"numeroDeRegistros": pendientesPorHacer.length

}

return res.status(200).json(miRespuesta);

})

*// .get('/:idPendiente', (req, res) => {*

*// const idPendiente = req.params.idPendiente;*

*// const pendienteBuscado = pendientesPorHacer.find( pendiente => pendiente.id == idPendiente );*

*// if( pendienteBuscado === undefined )*

*// return res.status(404).json({ mensaje : "No se encontró ese pendiente en el servidor :(" });*

*// else*

*// return res.status(200).json(pendienteBuscado);*

*// })*

app.listen(port, host, () **=>** {

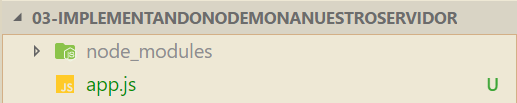
console.log(`¡Escuchando en ${host}:${port}/`);

});

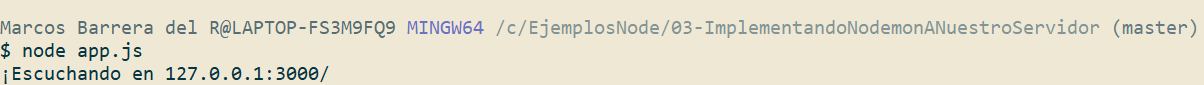
Ahora instalaremos *Express* con el comando ***npm install express@4.16.2 --save*** si no lo tenemos instalado en el directorio de nuestro proyecto.



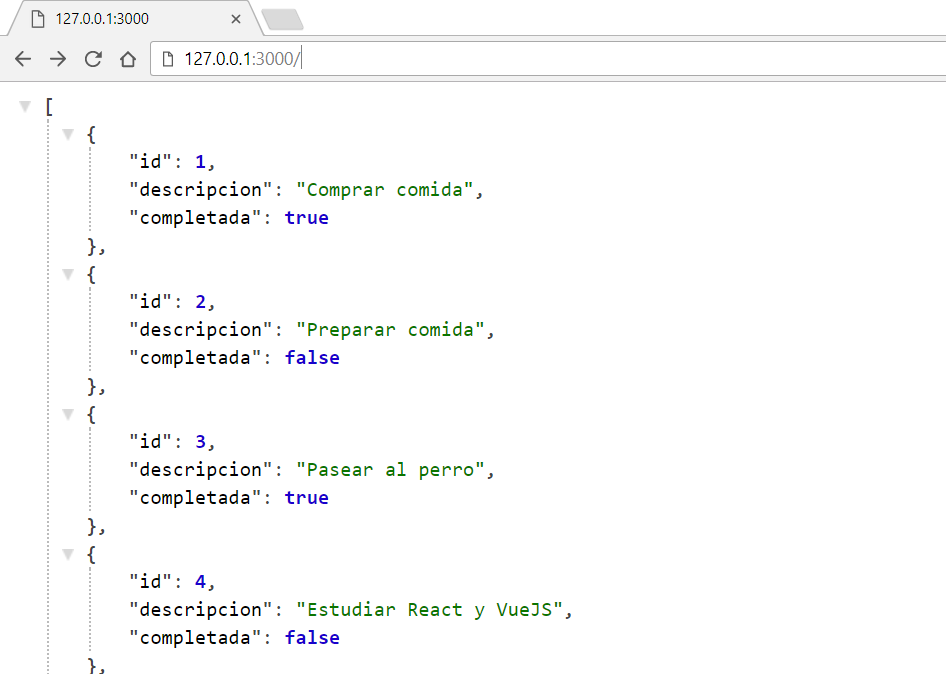
Colocaremos el código JavaScript en un archivo llamado *app.js*



Bien, una vez que tengamos todo listo, lo siguiente será ejecutar el archivo *app.js:*



Y como podemos ver, nuestro servidor estará escuchando peticiones como vimos en el anterior ejercicio, con la diferencia de que tenemos comentado el código de la consulta individual. Si nos dirigimos a la URL de nuestro servidor con el puerto indicado, veremos que la consulta de todos los pendientes funciona perfectamente:



Como ya vimos anteriormente, al ir a esa ruta, el siguiente código es ejecutado:

app.get('/', (req, res) **=>** {

return res.status(200).json(pendientesPorHacer);

})

Pero si nos dirigimos a la siguiente ruta:

*// .get('/:idPendiente', (req, res) => {*

*// const idPendiente = req.params.idPendiente;*

*// const pendienteBuscado = pendientesPorHacer.find( pendiente => pendiente.id == idPendiente );*

*// if( pendienteBuscado === undefined )*

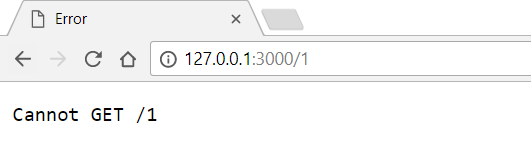
*// return res.status(404).json({ mensaje : "No se encontró ese pendiente en el servidor :(" });*

*// else*

*// return res.status(200).json(pendienteBuscado);*

*// })*

En nuestro navegador nos ocurrirá lo siguiente:



Obviamente no podremos acceder a la consulta individual de los pendientes en el array, ya que está comentado. Así que la solución a esto sería quitar los comentarios del código de la siguiente manera:

"use strict";

**const** express = require('express');

**const** app = express();

**const** port = 3000;

**const** host = '127.0.0.1';

**const** pendientesPorHacer = [

{id: 1, descripcion: "Comprar comida", completada: true},

{id: 2, descripcion: "Preparar comida", completada: false},

{id: 3, descripcion: "Pasear al perro", completada: true},

{id: 4, descripcion: "Estudiar React y VueJS", completada: false},

{id: 5, descripcion: "Terminar series pendientes", completada: false},

{id: 6, descripcion: "Reparar el celular", completada: true}

]

app.get('/', (req, res) **=>** {

return res.status(200).json(pendientesPorHacer);

})

app.get('/count', (req, res) **=>** {

**const** miRespuesta = {

"numeroDeRegistros": pendientesPorHacer.length

}

return res.status(200).json(miRespuesta);

})

.get('/:idPendiente', (req, res) **=>** {

**const** idPendiente = req.params.idPendiente;

**const** pendienteBuscado = pendientesPorHacer.find( pendiente **=>** pendiente.id == idPendiente );

if( pendienteBuscado === undefined )

return res.status(404).json({ mensaje : "No se encontró ese pendiente en el servidor :(" });

else

return res.status(200).json(pendienteBuscado);

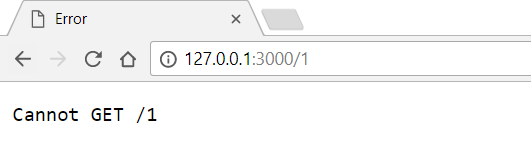
})

app.listen(port, host, () **=>** {

console.log(`¡Escuchando en ${host}:${port}/`);

});

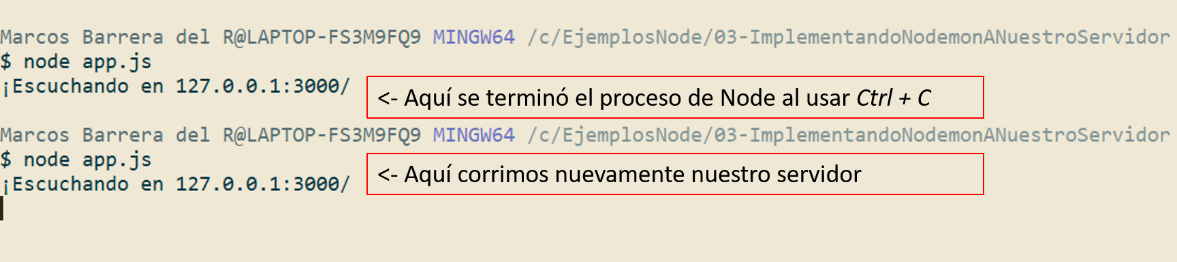
Después de “des-comentar” la ruta de '/:idPendiente', guardamos los cambios en nuestro archivo, y volvemos a intentar hacer uso de '/:idPendiente' desde nuestro navegador:



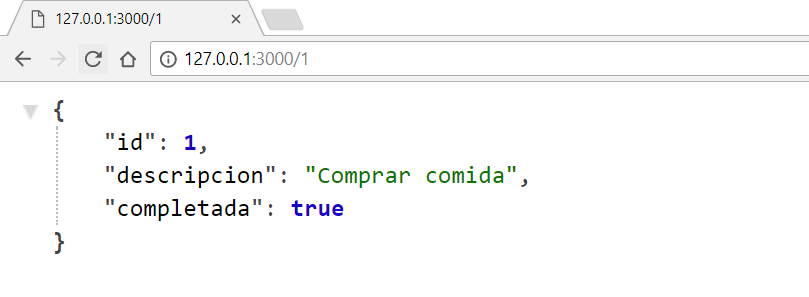
Y nos encontramos con que no se vieron reflejados los cambios, ¿pero por qué?

Lo que sucedió fue que corrimos el programa desde la consola con “*node app.js*” cuando nuestro código fuente tenía comentado el código que funciona como consulta individual. Sin embargo, después “des-comentamos” esa porción de código, pero el servidor que corrimos con “*node app.js*” no se reinició y por lo tanto no se reflejaron los cambios. Una solución sería cerrar el proceso de “*node app.js*” y volver a iniciarlo; todo manualmente.

Así que nos dirigimos a nuestra consola y con “Ctrl + C” terminamos el proceso, después volvemos a ejecutar “*node app.js*”.



Ahora, si nos dirigimos a nuestro navegador web y probamos la consulta individual, podremos ver que ahora sí funciona:



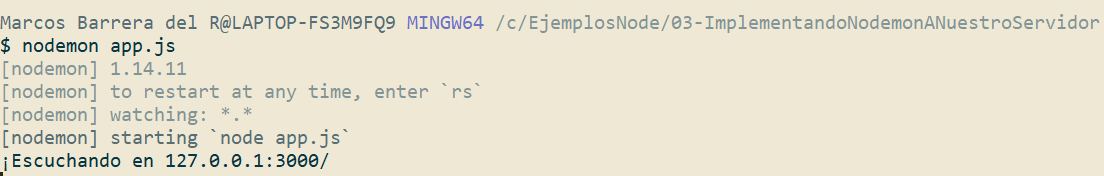
Sin embargo, esa no es la solución adecuada, ya que requiere que cada vez que hagamos un cambio en nuestro código fuente, reiniciemos manualmente nuestro servidor de Node.js. Así que la solución más adecuada es *Nodemon*.

Para instalar *Nodemon,* ejecutaremos el comando ***npm install -g nodemon*** en nuestra consola:



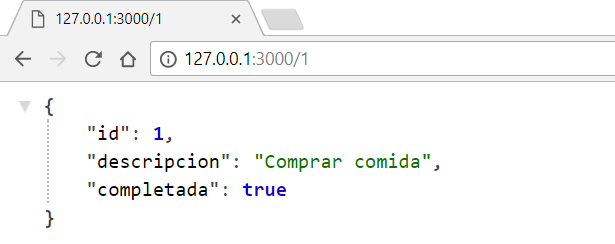
Y como ya vimos en la sección de NPM, la bandera “***-g***” instala globalmente el paquete, en este caso Nodemon fue instalado globalmente en nuestro sistema. Así que no tendremos que volver a instalarlo para poder usarlo en otros proyectos o ejercicios.

Una vez instalado, en lugar de utilizar el comando de *“node app.js”*, utilizaremos el comando ***nodemon app.js***

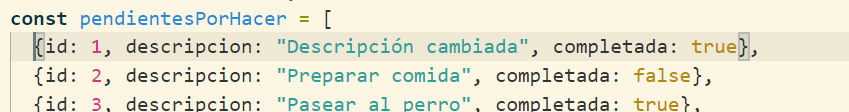


Y como podemos observar, en la consola nos menciona que *Nodemon* estará supervisando los archivos de nuestro código fuente. Así que si realizamos algún cambio, *Nodemon* automáticamente reiniciará el servidor por nosotros.

Entonces probémoslo. Si nos dirigimos a la ruta de “*127.0.0.1:3000/1*” con el código actual, veremos el siguiente elemento:



Ahora, si nos dirigimos a nuestro código fuente, cambiamos la descripción por otra y guardamos el archivo:



Veremos que, sin necesidad de reiniciar el servidor por nosotros mismos, se mostrarán los cambios automáticamente:

