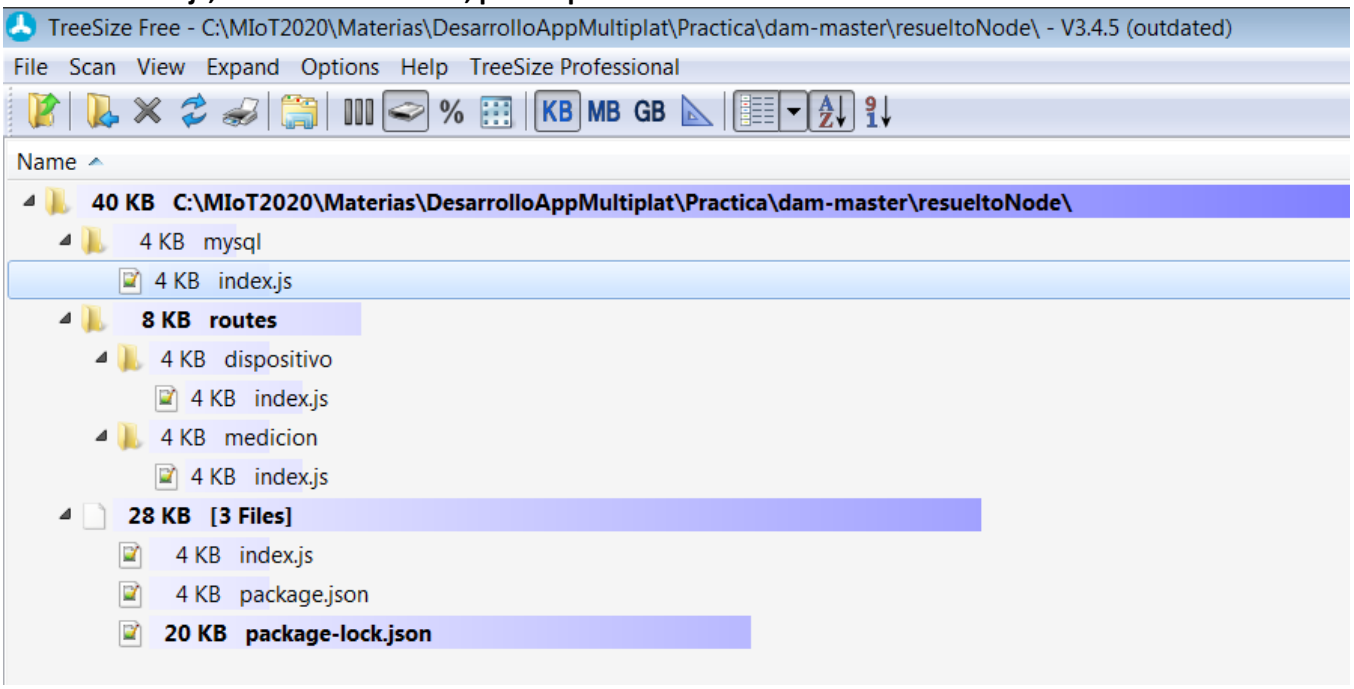
Backend resuelto Node detalles - R.Oliva 08.2020

Resuelto Node upd15-08-20. Se usa como base para el TP

1) Resuelto Node en Backend

El backend se encarga de recibir las consultas del front-end y derivarlas a la base de datos. En lugar de trabajar con un único index.js, see usa un ruteador, para separa los accesos de acuerdo a las solicitudes.



Express JS

Express es un framework web, escrito en JavaScript y alojado dentro del entorno de ejecución NodeJS. Es robusto, rápido, flexible y muy simple. Soporta los métodos GET, POST, PUT, DELETE entre otros y posee un método de direccionamiento especial que no se deriva de ningún método HTTP (.all). La definición de ruta tiene la siguiente estructura:

```
app.method(path,handler)
```

Donde:

`app` es una instancia de express.

`method` es un método de solicitud HTTP.

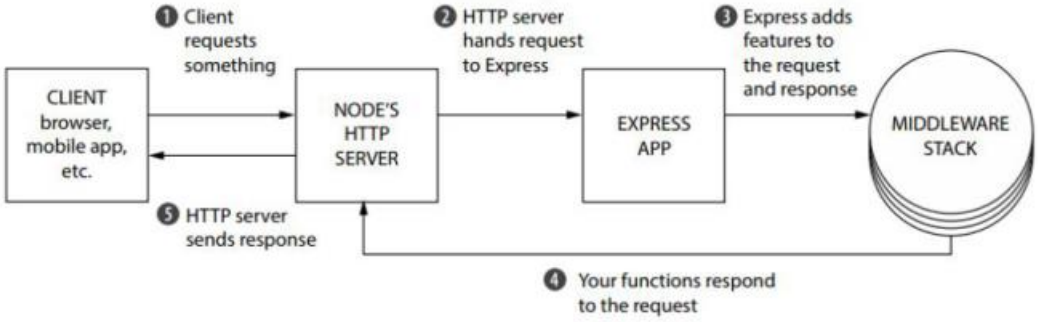
`(path,..)` es una vía de acceso a un servidor

`(..,handler)` es la función que se ejecuta cuando se correlaciona la ruta.

Dentro de los handler, uno posible es el denominado *middleware*. Las funciones de middleware son funciones que tienen acceso al objeto de solicitud (req), al objeto de respuesta (res) y a la siguiente función de middleware en el ciclo de solicitud/respuestas de la aplicación. La siguiente función de middleware se denota normalmente con una variable denominada next. Las funciones de middleware pueden realizar las siguientes tareas:

- Ejecutar cualquier código.
- Realizar cambios en la solicitud y los objetos de respuesta.
- Finalizar el ciclo de solicitud/respuestas.
- Invocar la siguiente función de middleware en la pila.

Si la función de middleware actual no finaliza el ciclo de solicitud/respuestas, debe invocar next() para pasar el control a la siguiente función de middleware. De lo contrario, la solicitud quedará colgada.



Objeto Request
Representa a la request HTTP y algunas de sus properties más útiles son:

- **`req.body`**

Contiene un par de clave-valor de los datos que se enviaron en el cuerpo de la request. Por defecto, su valor es "undefined" y se va a llenar con valores cuando usemos el "body-parsing" middleware

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

app.use(express.json()) // para parsear application/json

app.post('/perfil, function (req, res, next) {
  console.log(req.body)
  res.json(req.body)
})
```

- **req.ip**

Contiene la dirección IP de la request

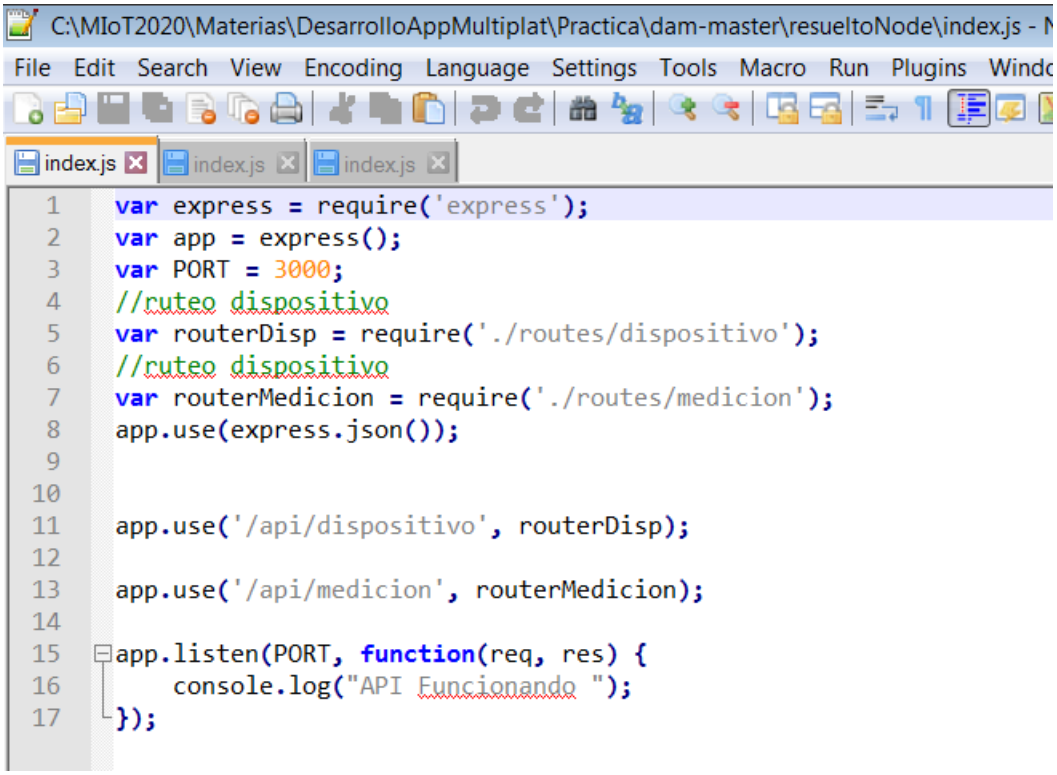
- **req.params**

Esta propiedad es un objeto de propiedades vinculadas a los parámetros de ruta (GET), por ejemplo, si tenemos la ruta "/usuario/:id", la propiedad "id" la podremos ver usando "req.params.id". Por defecto el valor del objeto es {}

- **req.secure**

Booleano para ver si se estableció una conexión TLS.

2) index.js Raiz:
Se sigue utilizando Express, pero se divide las rutas de acceso de acuerdo a la consulta que llega. Si la consulta es a dispositivo, se deriva a /routes/dispositivo/index.js. Si la consulta es a una medición, se deriva a /routes/medición/index.js – esto permite separar cualquier problema que pueda existir con los accesos, y tratar por separado cada consulta. También con Postman se pueden verificar partes y ruteo adecuado de las consultas.



3) /mysql/index.js

Este archivo configura y crea el pool de conexiones (se establece un máximo de 10) a la base de datos mysql, que debe estar previamente "levantada" y cargada la base de datos requerida, por ejemplo con phpmyadmin.

4) Routes

4.1) /routes/dispositivo/index.js

Los accesos a dispositivos se rutean por aquí:

C:\MIoT2020\Materias\DesarrolloAppMultiplat\Practica\dam-master\resueltoNode\routes\dispositivo\index.js - Not

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

index.js index.js index.js

```
1  var express = require('express');
2  var routerDispositivo = express.Router();
3  var pool = require('../../mysql');
4
5  //Devuelve un array de dispositivos
6  routerDispositivo.get('/', function(req, res) {
7      pool.query('Select * from Dispositivos', function(err, result, fields) {
8          if (err) {
9              res.send(err).status(400);
10             return;
11         }
12         res.send(result);
13     });
14 });
15
16 module.exports = routerDispositivo;
```

4.2) /routes/medicion/index.js

C:\MIoT2020\Materias\DesarrolloAppMultiplat\Practica\dam-master\resueltoNode\routes\medicion\index.js - Notepad++ [Administrator]

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

index.js index.js index.js index.js

```
1  var express = require('express');
2  var routerMedicion = express.Router();
3  var pool = require('../../mysql');
4
5  //Espera recibir por parámetro un id de dispositivo y devuelve su última medición
6  routerMedicion.get('/:idDispositivo', function(req, res) {
7      pool.query('Select * from Mediciones where dispositivoId=? order by fecha desc', [req.params.idDispositivo], function(err, result, fields) {
8          if (err) {
9              res.send(err).status(400);
10             return;
11         }
12         res.send(result[0]);
13     });
14 });
15
16 //Espera recibir por parámetro un id de dispositivo y devuelve todas sus mediciones
17 routerMedicion.get('/:idDispositivo/todas', function(req, res) {
18     pool.query('Select * from Mediciones where dispositivoId=? order by fecha desc', [req.params.idDispositivo], function(err, result, fields) {
19         if (err) {
20             res.send(err).status(400);
21             return;
22         }
23         res.send(result);
24     });
25 });
26
27 //Espera recibir por parámetro un id de dispositivo y un valor de medición y lo inserta en base de datos.
28 routerMedicion.post('/agregar', function(req, res) {
29     pool.query('Insert into Mediciones (fecha,valor,dispositivoId) values (?,?,?)', [req.body.fecha, req.body.valor, req.body.dispositivoId], function(err, result, fields) {
30         if (err) {
31             res.send(err).status(400);
32             return;
33         }
34         res.send(result);
35     });
36 });
37
38 module.exports = routerMedicion;
```