

# NodeJS R. Oliva – Ej12 06.2020

## a) Estructura

Name	Size	Allocated	Files	Fold...	% of Par...	Last Modifi...
C:\MloT2020\Materias\DesarrolloAppWeb\delSitio\RepoDAW\daw\ejercicios\Ejercicios_NodeJs\Ejercicio_12\	3.569,4 KB	4.744,0 KB	476	101	100,0 %	13/06/2020
css	313,6 KB	316,0 KB	2	0	6,7 %	03/06/2020
materialize.css	175,0 KB	176,0 KB	1	0	55,7 %	03/06/2020
materialize.min.css	138,5 KB	140,0 KB	1	0	44,3 %	03/06/2020
images	6,5 KB	8,0 KB	2	0	0,2 %	03/06/2020
lightbulb.png	3,3 KB	4,0 KB	1	0	50,0 %	03/06/2020
window.png	3,2 KB	4,0 KB	1	0	50,0 %	03/06/2020
js	544,6 KB	552,0 KB	4	0	11,6 %	03/06/2020
main.js	4,0 KB	4,0 KB	1	0	0,7 %	03/06/2020
materialize.js	362,3 KB	364,0 KB	1	0	65,9 %	03/06/2020
materialize.min.js	176,9 KB	180,0 KB	1	0	32,6 %	03/06/2020
MyFramework.js	1,4 KB	4,0 KB	1	0	0,7 %	03/06/2020
src	6,9 KB	20,0 KB	4	0	0,4 %	03/06/2020
compile_ts.sh	0,4 KB	4,0 KB	1	0	20,0 %	03/06/2020
main.ts	4,5 KB	8,0 KB	1	0	40,0 %	03/06/2020
MyFramework.ts	1,9 KB	4,0 KB	1	0	20,0 %	03/06/2020
tsconfig.json	0,1 KB	4,0 KB	1	0	20,0 %	03/06/2020
ws	2.694,3 KB	3.832,0 KB	460	96	80,8 %	13/06/2020
mysql	0,4 KB	4,0 KB	1	0	0,1 %	03/06/2020
index.js	0,4 KB	4,0 KB	1	0	100,0 %	03/06/2020
node_modules	2.675,2 KB	3.796,0 KB	455	94	99,1 %	13/06/2020
[Files]	18,7 KB	32,0 KB	4	0	0,8 %	03/06/2020
datos.json	0,8 KB	4,0 KB	1	0	0,1 %	03/06/2020
index.js	1,2 KB	4,0 KB	1	0	0,1 %	03/06/2020
package.json	0,3 KB	4,0 KB	1	0	0,1 %	03/06/2020
package-lock.json	16,5 KB	20,0 KB	1	0	0,5 %	03/06/2020
[Files]	3,5 KB	16,0 KB	4	0	0,3 %	03/06/2020
devices.txt	0,7 KB	4,0 KB	1	0	0,1 %	03/06/2020
index.html	2,2 KB	4,0 KB	1	0	0,1 %	03/06/2020
serve_http.sh	0,2 KB	4,0 KB	1	0	0,1 %	03/06/2020
serve_node_app.sh	0,4 KB	4,0 KB	1	0	0,1 %	03/06/2020

### a.1) En ROOT\_

Idem Ej12 de TS:

a.1.1) index.htm -> impreso a PDF

a.1.2) devices.txt -> impreso a PDF

a.1.3) Root Scripts:

a.1.3a) serve\_http.sh

```
serve_http.sh
1  #!/bin/bash
2  CONTAINER_NAME=http-server
3  DIR_TO_SERVE=`realpath $1`
4  HOST_PORT=$2
5  docker run --rm --interactive --name $CONTAINER_NAME --volume $DIR_TO_SERVE:/dir_to_serve -p $HOST_PORT:8080 abassi/node-http-server
6
```

a.1.3b) serve\_node\_app.sh (se agrega)

```
serve_http.sh  serve_node_app.sh  compile_ts.sh  tsconfig.json
1  #!/bin/bash
2  CONTAINER_NAME=nodejs-container
3  APP_DIR=`realpath $1`
4  FILE=$2
5  HOST_PORT=$3
6  NET=$4
7
8  CONTAINER_WORKDIR=/usr/src/app
9  CONTAINER_PORT=3000
10
11  echo "${CONTAINER_NAME}, dir:$APP_DIR, file:$FILE, port:$HOST_PORT}"
12
13  docker run --rm --interactive \
14  --name $CONTAINER_NAME \
15  --network $NET \
16  --publish $HOST_PORT:$CONTAINER_PORT \
17  --volume $APP_DIR:$CONTAINER_WORKDIR \
18  abassi/nodejs-server:dev \
19  nodemon $CONTAINER_WORKDIR/$FILE
20
```

Por ejemplo en el trabajo final se usa:

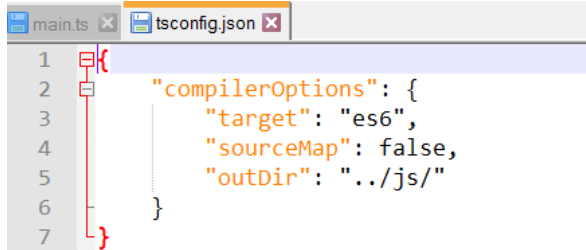
./serve\_node\_app\_net.sh "\$PWD" ws/index.js 8000 mysql-net

## a.2) en /src

a.2.1) main.ts -> impreso a PDF (Main\_NodeJsEj12.pdf) → casi idéntico al de TSEj12, sólo difiere en línea 85 que evita el uso de devices.py

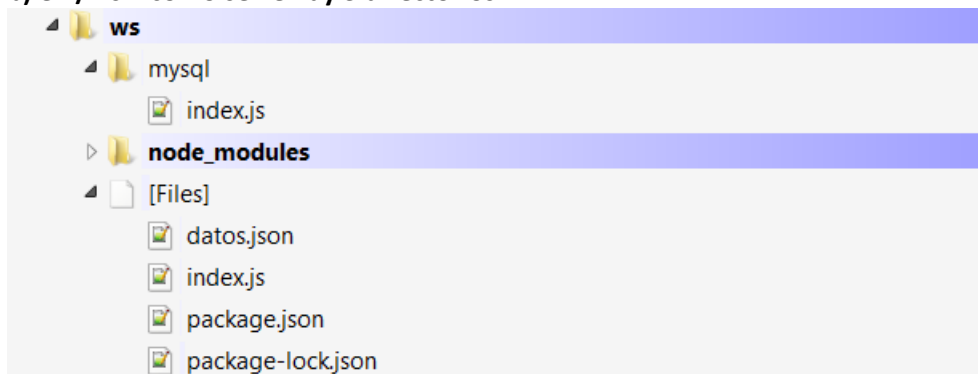
a.2.2) myFramework.ts -> impreso a PDF (Ej12\_NodeMyFramework\_js.pdf) → difiere en el método requestPOST(), sobre todo en líneas finale 72-74 de envío de data.

a.2.3) /src scripts: el compile\_ts que se vienen usando en /src desde el principio: pero con config de es6:



```
1 {
2   "compilerOptions": {
3     "target": "es6",
4     "sourceMap": false,
5     "outDir": "../js/"
6   }
7 }
```

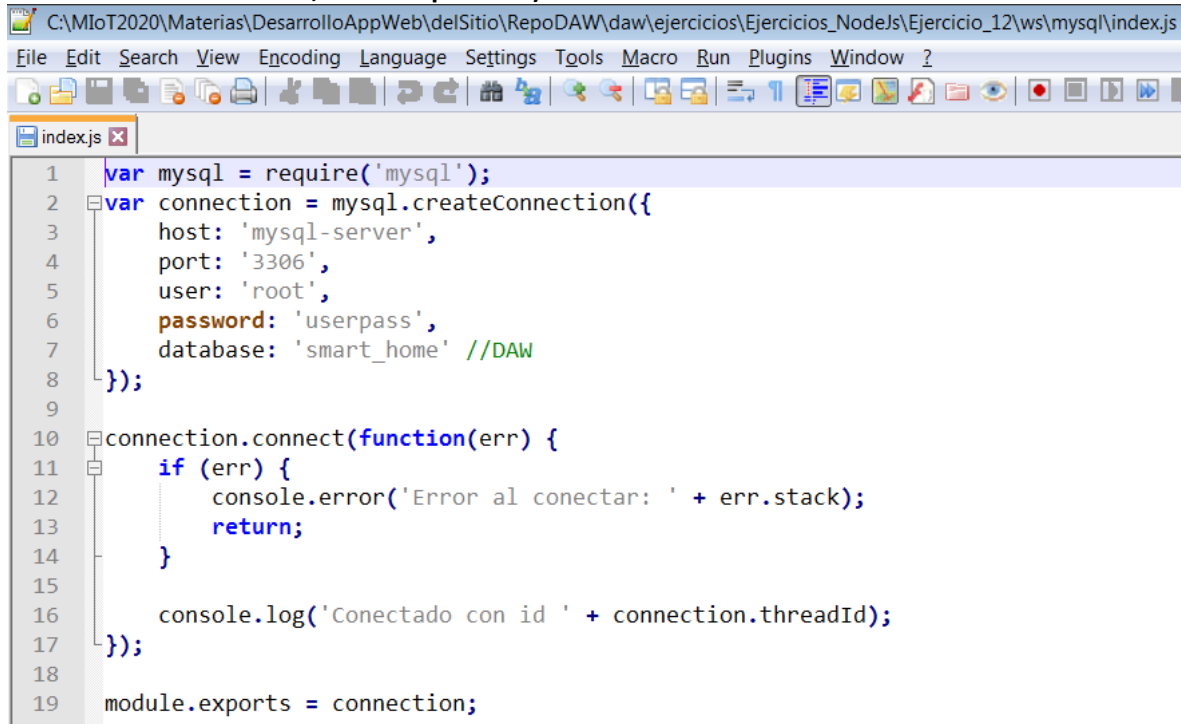
## b) en /ws – como se ve hay 3 directorios:



b.1) En /ws/node\_modules es generado por node-js y descarga muchos módulos que tiene que usar.

b.2) En /ws/mysql está un index.js específico para mysql

(solicita el módulo mysql, establece la conexión mysql-server, en port 3306, con database “smart\_home”, si la conexión es correcta, la hace “publica”)



```
1 var mysql = require('mysql');
2 var connection = mysql.createConnection({
3   host: 'mysql-server',
4   port: '3306',
5   user: 'root',
6   password: 'userpass',
7   database: 'smart_home' //DAW
8 });
9
10 connection.connect(function(err) {
11   if (err) {
12     console.error('Error al conectar: ' + err.stack);
13     return;
14   }
15   console.log('Conectado con id ' + connection.threadId);
16 });
17
18 module.exports = connection;
```

### b.3) En /ws

#### b.3.1) datos.json – sigue siendo la misma que en Ej.3-6 de node

```
package.json x index.js x datos.json x MyFramework.ts x index.html x
1  [
2    { "id": 1, "name": "Lámpara 1", "description": "Luz Living", "state": 1, "type": 0 },
3    { "id": 2, "name": "Lámpara 2", "description": "Luz Cocina", "state": 0, "type": 0 },
4    { "id": 3, "name": "Velador", "description": "Velador Living", "state": 1, "type": 0 },
5    {
6      "id": 4,
7      "name": "Persiana 1",
8      "description": "Persiana Living",
9      "state": 1,
10     "type": 1
11   },
12   {
13     "id": 5,
14     "name": "Persiana 2",
15     "description": "Persiana Cocina",
16     "state": 0,
17     "type": 1
18   },
19   {
20     "id": 6,
21     "name": "Persiana 3",
22     "description": "Persiana Balcón",
23     "state": 1,
24     "type": 1
25   },
26   { "id": 7, "name": "Lámpara 3", "description": "Luz Balcón", "state": 1, "type": 0 }
27 ]
```

#### b.3.2) Idem: package.json en Ej.3-6 de node, define dependencias de terceros (es este caso Express y mysql)

```
package.json x index.js x datos.json x MyFramework.ts x index.html x
1  {
2    "name": "ejercicios",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1"
8    },
9    "author": "",
10   "license": "ISC",
11   "dependencies": {
12     "express": "^4.17.1",
13     "mysql": "^2.18.1"
14   }
15 }
16
```

#### b.3.3) El index.js de esta aplicación define el uso de express y su combinación con mysql, el inicio es: (ver Ej12\_Node\_index\_js.pdf)

```
var PORT=3000;
var express = require('express');
var app = express();
var mysql = require('./mysql');
app.use(express.json()); // para parsear application/json
app.use(express.static('.')); // para servir archivos estáticos
```

y luego los app.get(), app.post(), app.listen() → igual imprimimos en un PDF