

Secuencia TPFinal - R. Oliva – a partir de Ej12Node y borrador iniciado 14.06.2020

I Parte

I.a) Estructura

Name	Size	Allocated	Files	Fold...	% of Par...	Last Modifi...
C:\MioT2020\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

I.a.1) En ROOT_

Idem Ej12 de TS:

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

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

I.a.1.3) Root Scripts:

I.a.1.3a)serve_http.sh

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

I.a.1.3b) serve_node_app.sh (se agrega)

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

Por ejemplo en el trabajo final se usa:

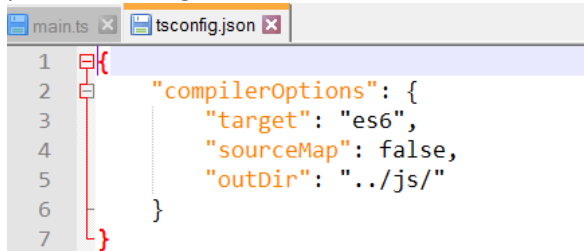
```
./serve_node_app_net.sh "$PWD" ws/index.js 8000 mysql-net
```

I.a.2) en /src

I.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

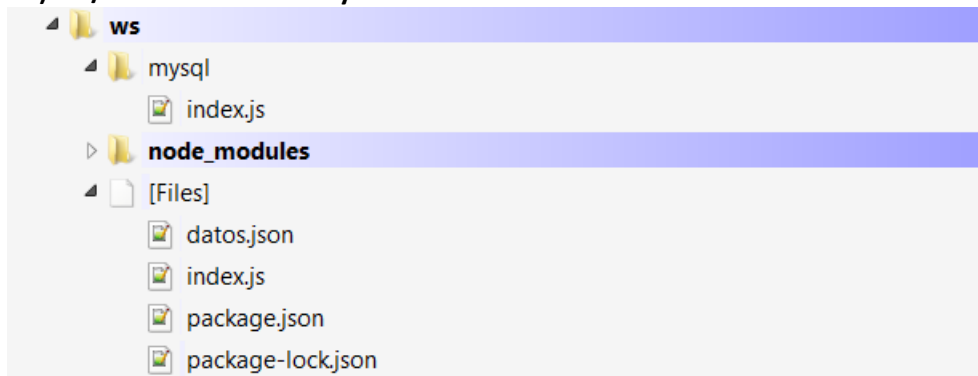
I.a.2.2) myFramework.ts -> impreso a PDF (Ej12_NodeMyFramework_js.pdf) → difiere en el método requestPOST(), sobre todo en líneas finales 72-74 de envío de data.

I.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 }
```

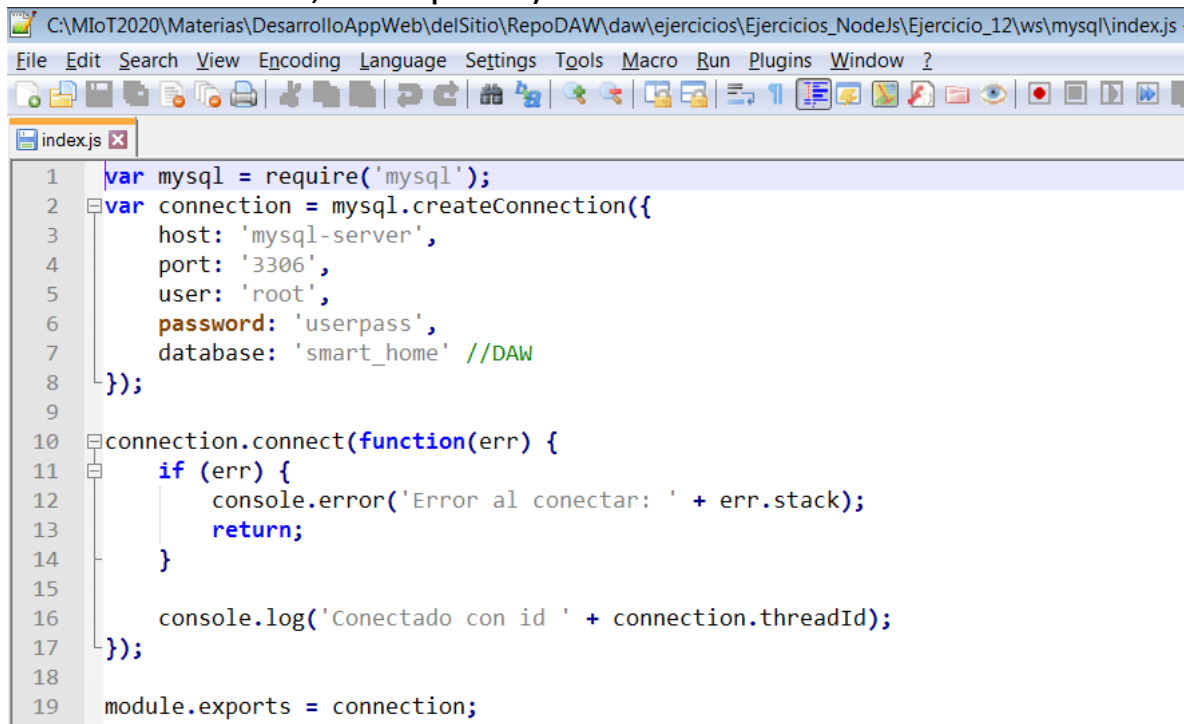
I.b) en /ws – como se ve hay 3 directorios:



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

I.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;
```

I.b.3) En /ws

I.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 ]
```

I.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
```

I.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

II Parte

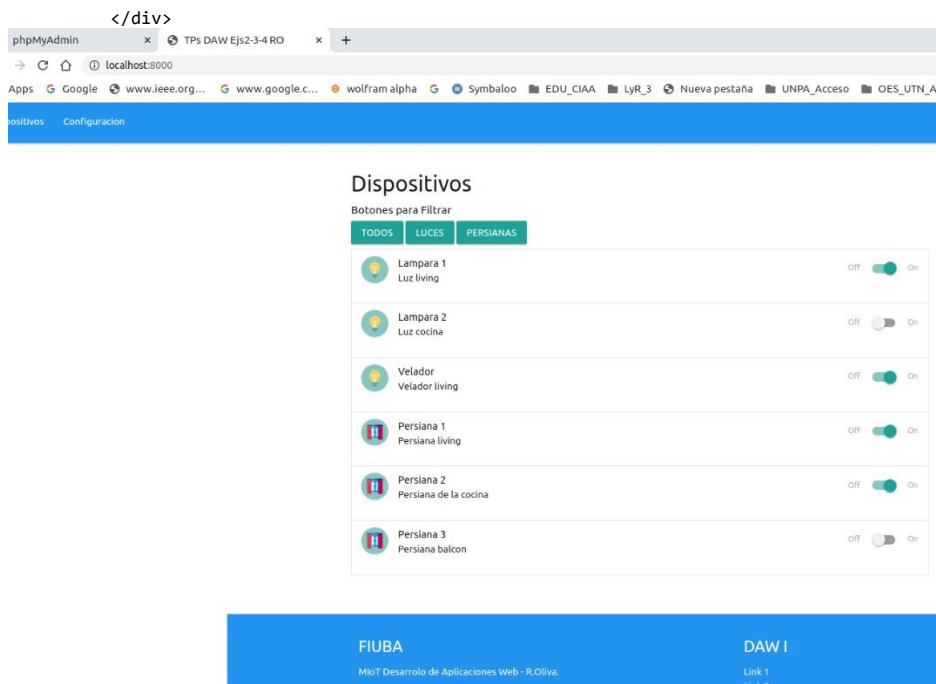
II.a) Copiamos Ej12_Node y ensayamos uso de contenedores en MaqJ2. según documento ReplicaTPFinal_14062020b.pdf

II.b) Importamos index.html de repositorio de maquetado /sandbox y Ejercicios iniciales, y removemos parte de listado fija

II.c) Agregamos a index.html botones de Filtrado: TODOS, LAMPARAS; PERSIANAS

```
<!-- 10.5.20 Aquí agregamos el container, restringe a 70% del ancho -->
<div class="container">
  <!-- Nuevos Botones TPFinal para filtrado, removemos lista fija de dispositivos -->
  <div class="container">
    <br/>
    <h4>Dispositivos</h4>
    <h6>Botones para Filtrar</h6>
    <div>
      <a id="bTodos" class="waves-effect btn">TODOS</a>
      <a id="bLuces" class="waves-effect btn">LUCES</a>
      <a id="bPersianas" class="waves-effect btn">PERSIANAS</a>
    </div>
    <ul class="collection" id='devicesList'>
    </ul>

    <br/>
    <br/>
  </div>
```



II.d) Copiamos Ej12_Node con nuevo index.html y ensayamos en Probook OK -> commit desde GK

II.e) Ensayos con node en puerto 8000 con nuevo index.html → Ok → pusheamos cambios a github, repo iniciado 14.6.20 pero todo el trabajo fue en repoSandbox.

II.f) Se modifica en primer lugar dentro de main.ts el handleEvent, ya que ahora no hay un único Boton como en Ej12 sino 3: TODOS, LUCES y PERSIANAS. Se define un HTMLElement llamado elem asociado a getElemByEvent, cuyo elem.id es el que cambia de acuerdo al botón de filtrado apretado. Cada una dispara un requestGET con filter=0,1 o 2. Se mantiene la clase ViewMainPage del Ej12, como así también el myFramework.ts sin cambios.

II.g) Dentro de main() se agrega a cada uno de los botones definidos un EventListener, de acuerdo a su id, como se muestra , y esto compila correctamente:

```

125 main():void
126 {
127   this.myf = new MyFramework();
128
129   this.view = new ViewMainPage(this.myf);
130
131   this.myf.requestGET("devices",this);
132
133   // en clase se resolvió con configClick() pero lo hacemos mas crudo
134   // no se toca el MyFramework.ts de Ej12
135   // primero handler para boton "bTodos"
136   let b:HTMLElement = this.myf.getElementById("bTodos");
137   b.addEventListener("click",this);
138   // luego para boton "bLuces"
139   b = this.myf.getElementById("bLuces");
140   b.addEventListener("click",this);
141   // finalment para boton "bPersianas"
142   b = this.myf.getElementById("bPersianas");
143   b.addEventListener("click",this);
144
145 }

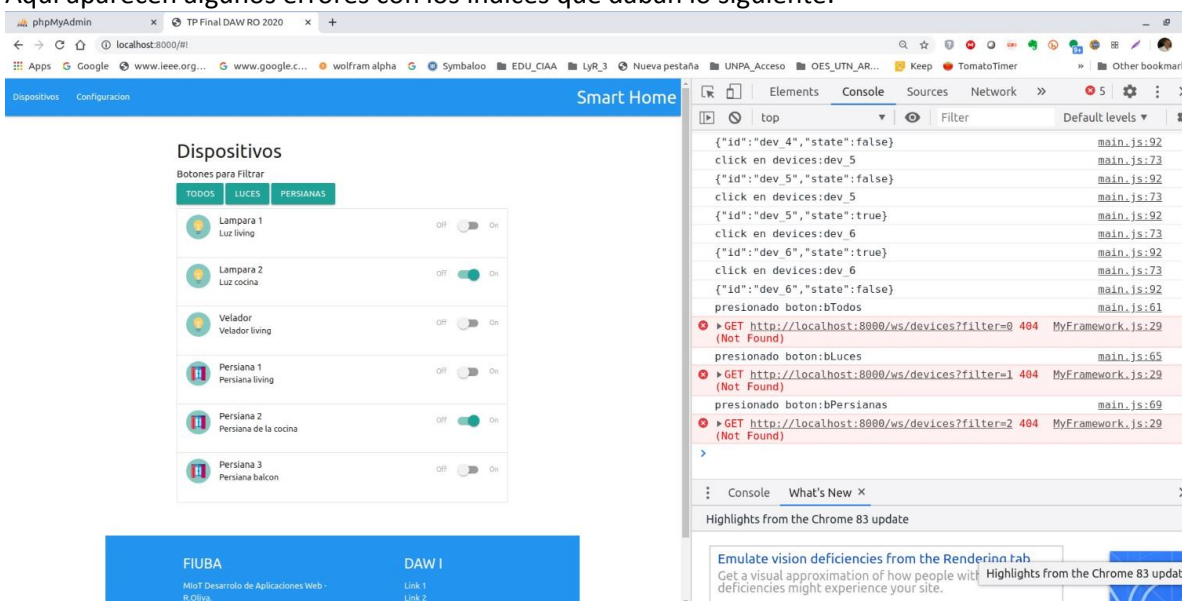
```

OUTPUT DEBUG CONSOLE PROBLEMS TERMINAL

[7:27:42 PM] Starting compilation in watch mode...

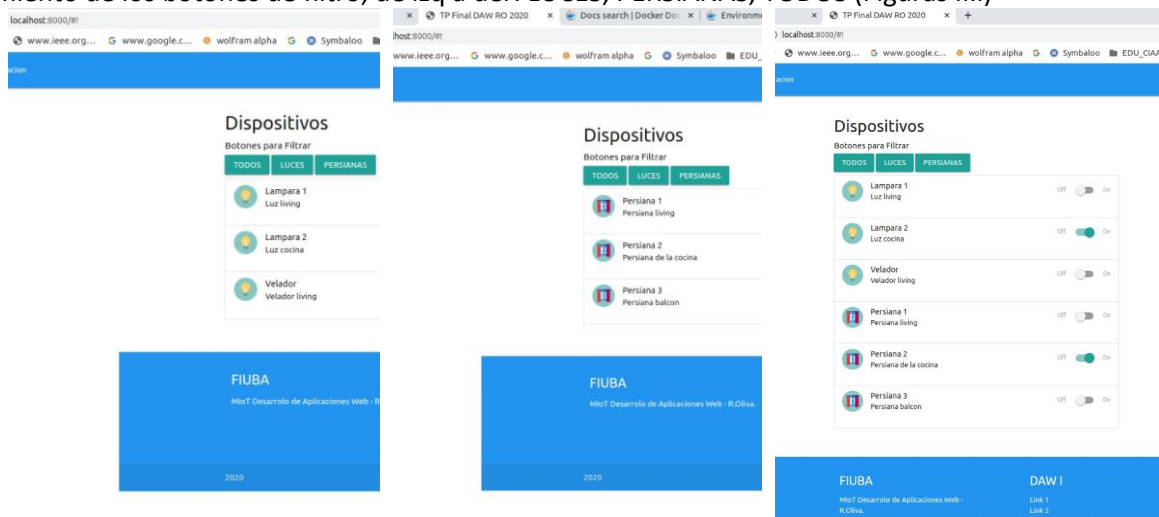
[7:27:47 PM] Compilation complete. Watching for file changes.

II.h) Mantenemos el mismo datos.json generado en clase. Dentro del backend, index.js continua utilizando el framework express y mysql como base de datos, pero modificamos el `app.get('/devices',..)` para que filtremos el query por tipo 0, tipo 1 o nada (todos). Por lo demás, queda igual. Aquí aparecen algunos errores con los índices que daban lo siguiente:

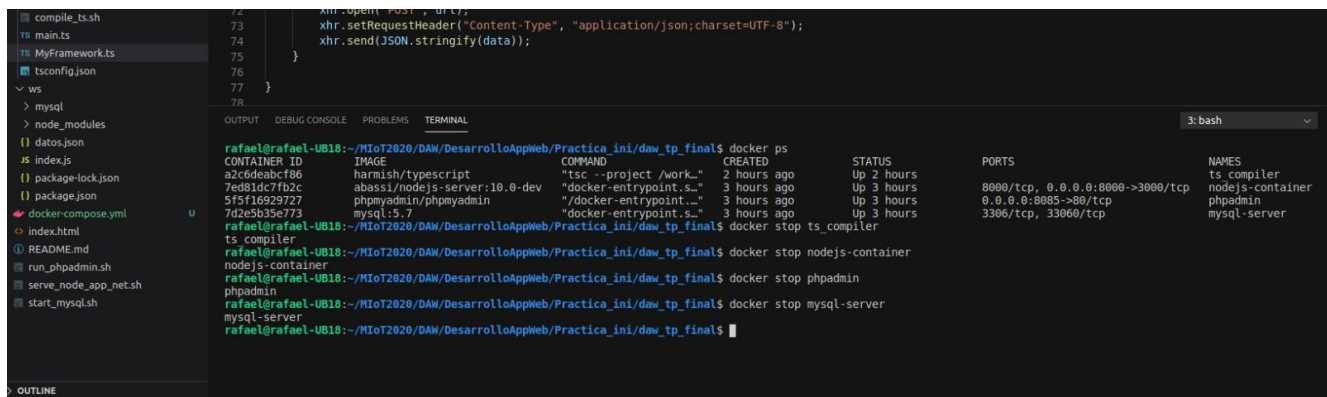


Después se encontró el error en el índice y funcionó correctamente. Además, viendo la consola se encontró que había quedado en el index.htm una referencia a User.js de los ejercicios iniciales, que se eliminó.

II.i) Se verificó montando en el orden indicado en la consigna el montaje de los contenedores Docker, y el funcionamiento de los botones de filtro, de izq a der: LUCES, PERSIANAS, TODOS (Figuras II.i)



II.j) Para el armado del compose, es importante dar de baja primero individualmente a todos los componentes, con el comando `docker stop id_container`:



```
rafael@rafael-UB18:~/MIoT2020/DAW/DesarrolloAppWeb/Practica_ini/daw_tp_final$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED              STATUS              PORTS              NAMES
a2cdeabcf866       harimish/typescript   "tsc --project /work..."   2 hours ago         Up 2 hours         8000/tcp, 0.0.0.0:8000->3000/tcp   ts_compiler
7ed81dc7fb2c       abassi/nodejs-server:10.0-dev   "/docker-entrypoint.s..."   3 hours ago         Up 3 hours         0.0.0.0:8085->80/tcp   nodejs-container
5f5f16929727       phpmyadmin/phpmyadmin   "/docker-entrypoint.s..."   3 hours ago         Up 3 hours         3306/tcp, 33060/tcp   phpadmin
7d2e5b35e773       mysql:5.7             "docker-entrypoint.s..."   3 hours ago         Up 3 hours         3306/tcp, 33060/tcp   mysql-server

rafael@rafael-UB18:~/MIoT2020/DAW/DesarrolloAppWeb/Practica_ini/daw_tp_final$ docker stop ts_compiler
rafael@rafael-UB18:~/MIoT2020/DAW/DesarrolloAppWeb/Practica_ini/daw_tp_final$ docker stop nodejs-container
rafael@rafael-UB18:~/MIoT2020/DAW/DesarrolloAppWeb/Practica_ini/daw_tp_final$ docker stop phpadmin
rafael@rafael-UB18:~/MIoT2020/DAW/DesarrolloAppWeb/Practica_ini/daw_tp_final$ docker stop mysql-server
rafael@rafael-UB18:~/MIoT2020/DAW/DesarrolloAppWeb/Practica_ini/daw_tp_final$ docker-compose down
```

II.k) Instalación de docker-compose: en la máquina inicial de desarrollo no estaba instalado:

```
rafael@rafael-UB18:~/MIoT2020/DAW/DesarrolloAppWeb/Practica_ini/daw_tp_final$ docker-compose up
```

Command 'docker-compose' not found, but can be installed with:

```
sudo snap install docker          # version 19.03.11, or
sudo apt install docker-compose
```

See 'snap info docker' for additional versions.

```
rafael@rafael-UB18:~/MIoT2020/DAW/DesarrolloAppWeb/Practica_ini/daw_tp_final$ sudo apt install docker-compose
[sudo] password for rafaël:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  efibootmgr libfwupd1
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  golang-docker-credential-helpers python-asn1crypto python-backports.ssl-match-hostname python-cached-property python-certifi python-ffi-backend python-
  charDET python-cryptography python-docker python-dockerpty python-dockerpycreds python-docopt python-enum34 python-funcsigs python-functools32 python-idna python-ipaddress python-
  jsonschema python-mock
python-openssl python-pbr python-pkg-resources python-requests python-six python-texttable python-urllib3 python-websocket python-yaml
Suggested packages:
  python-cryptography-doc python-cryptography-vectors python-enum34-doc python-funcsigs-doc python-mock-doc python-openssl-doc python-openssl-dbg python-
  setuptools python-socks python-ntlm
Recommended packages:
  docker.io
The following NEW packages will be installed:
  docker-compose golang-docker-credential-helpers python-asn1crypto python-backports.ssl-match-hostname python-cached-property python-certifi python-ffi-
  backend python-charDET python-cryptography python-docker python-dockerpty python-dockerpycreds python-docopt python-enum34 python-funcsigs python-functools32 python-idna python-
  ipaddress python-jsonschema python-mock python-openssl python-pbr python-pkg-resources python-requests python-six python-texttable python-urllib3 python-websocket python-yaml
0 upgraded, 29 newly installed, 0 to remove and 26 not upgraded.
Need to get 1.947 kB of archives.
After this operation, 9.337 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Al acceder se instala y queda listo para operar. El orden de secuencia está dado por el archivo `.yaml` en el raíz del directorio.

II.l) Para el armado del archivo `/doc/notes.txt` y el archivo de YAML (<https://yaml.org/>) para docker compose se usaron las recomendaciones de las páginas:

<https://iot-es.herokuapp.com/static/posts/docker-compose-applications/content-spanish.html#h.cz3ui1pvhtir>
<https://docs.docker.com/compose/compose-file/>

Se consideraron los parámetros:


```

docker-compose.yml
1  # YAML archivo para componer los servicios de SmartHome 2020 R.O.
2  # https://docs.docker.com/compose/yml/
3  # Version
4  version: '3'
5  # Servicios
6  services:
7  # Server mysql - 5.7
8  mysql-server:
9    image: mysql:5.7
10   hostname: mysql-server
11   container_name: mysql-server
12   restart: always
13   environment:
14     MYSQL_ROOT_PASSWORD: userpass
15   volumes:
16     - ./db/dumps:/docker-entrypoint-initdb.d
17     - ./db/data:/var/lib/mysql
18   networks:
19     - mysql-net
20  # PHP para administracion de la base de datos
21  phpmyadmin:
22    image: phpmyadmin/phpmyadmin
23    hostname: phpmyadmin
24    container_name: phpmyadmin
25    restart: always
26    environment:
27      PMA_HOST: mysql-server
28      PMA_PORT: 3306
29      MYSQL_ROOT_PASSWORD: userpass
30    networks:
31      - mysql-net
32    ports:
33      - "8085:80"
34    depends_on:
35      - mysql-server
36  # Node.js para el backend
37  node-app:
38    image: abassi/nodejs-server:10.0-dev
39    hostname: nodejs-container
40    container_name: nodejs-container
41    restart: always
42    volumes:
43      - ./:/home/node/app
44      #/home/node/app/ws/index.js
45    networks:
46      - mysql-net
47    depends_on:
48      - mysql-server
49    ports:
50      - "8000:3000"
51    command: nodemon ws/index.js
52  # Red interna
53  networks:
54    mysql-net:
55      driver: bridge
56

```

II.m) Verificación (i): En la máquina de desarrollo se notó que la primera vez con docker-compose upn no llegaba a montar el Node-js, aunque luego de hacer docker-compose down y luego up nuevamente lo lograba como en (Figuras II.i).

DOWN SEQUENCE FROM ANOTHER TERMINAL

```

rafael@rafael-UB18:~/MIoT2020/DAW/DesarrolloAppWeb/Practica_ini/daw_tp_final$ docker-compose down
Stopping phpmyadmin    ... done
Stopping nodejs-container ... done
Stopping mysql-server  ... done
Removing phpmyadmin    ... done
Removing nodejs-container ... done
Removing mysql-server  ... done
Removing network dawtpfinal_mysql-net
rafael@rafael-UB18:~/MIoT2020/DAW/DesarrolloAppWeb/Practica_ini/daw_tp_final$

```

UP SEQUENCE REPEAT

```

rafael@rafael-UB18:~/MIoT2020/DAW/DesarrolloAppWeb/Practica_ini/daw_tp_final$ docker-compose up
Creating network "dawtpfinal_mysql-net" with driver "bridge"
Creating mysql-server ...
Creating mysql-server ... done
Creating nodejs-container ...
Creating phpmyadmin ...
Creating nodejs-container
Creating phpmyadmin ... done
Attaching to mysql-server, nodejs-container, phpmyadmin
mysql-server | 2020-06-20 21:20:52+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 5.7.30-1debian10 started.
mysql-server | 2020-06-20 21:20:55+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'

-----

mysql-server | Version: '5.7.30' socket: '/var/run/mysqld/mysqld.sock' port: 3306 MySQL Community Server (GPL)
phpmyadmin   | AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.19.0.4. Set the 'ServerName' directive globally to suppress this message

-----

```

```

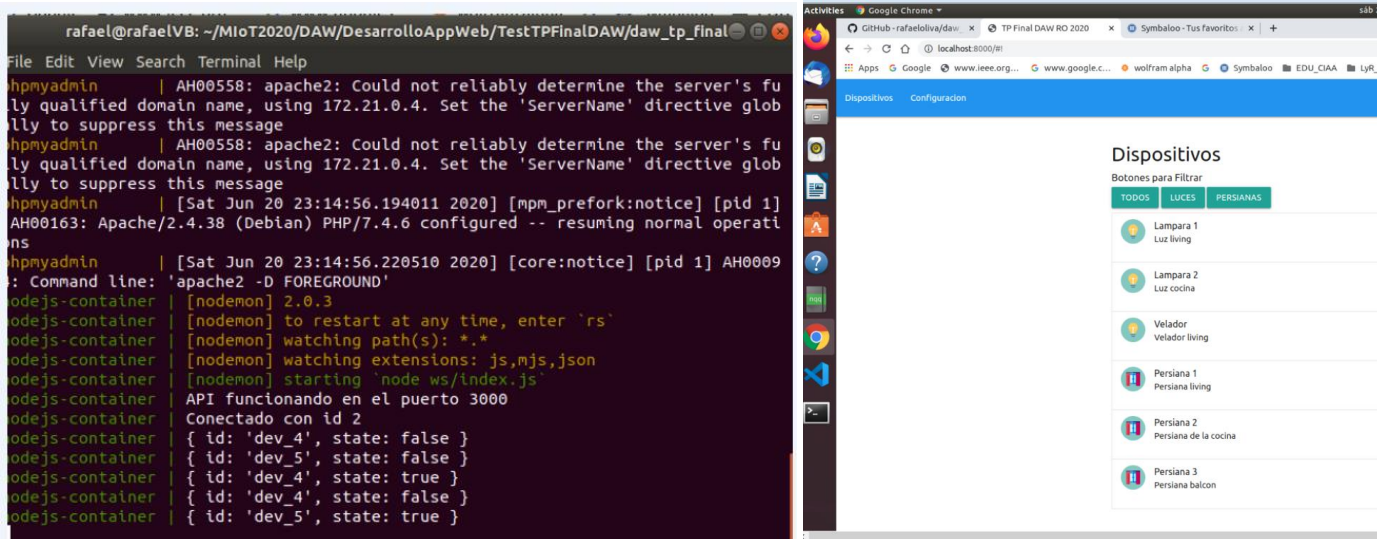
nodejs-container | [nodeemon] 2.0.3
nodejs-container | [nodeemon] to restart at any time, enter `rs`
nodejs-container | [nodeemon] watching path(s): *.*
nodejs-container | [nodeemon] watching extensions: js,mjs,json
nodejs-container | [nodeemon] starting `node ws/index.js`
phpmyadmin      | [Sat Jun 20 21:21:12.387849 2020] [mpm_prefork:notice] [pid 1] AH00163: Apache/2.4.38 (Debian) PHP/7.4.6 configured -- resuming normal
operations
phpmyadmin      | [Sat Jun 20 21:21:12.428055 2020] [core:notice] [pid 1] AH00094: Command line: 'apache2 -D FOREGROUND'
nodejs-container | API funcionando en el puerto 3000
nodejs-container | Conectado con id 2

```

y ahí corre sin problemas tanto la SPA como el phpadmin en localhost:8085

II.n) Verificación (ii):

Se hizo la descarga en otra máquina A) corriendo UB18.04 / VM sobre Win10 y en otra corriendo UB16.04 / VM sobre Win7. En la primera se logró hacerlo funcionar repitiendo la secuencia indicada:



En la segunda no se logró instalar Docker Compose, probablemente por tratarse de una versión de 2018 de Linux (16.04) u algún otro problema:

NO CORRE EN UBUNTU 16.04/VM WIN7 20.6.20, AUN CAMBIANDO VERSION DE YML DE 3 A 2, O REMOVIÉNDOLA..

```

rafael@rafael-VirtualBox:~/MIOT2020/TestRepoDAW/daw_tp_final$ docker-compose up
ERROR: Couldn't connect to Docker daemon at http://docker://localhost:2376 - is it running?

```

If it's at a non-standard location, specify the URL with the DOCKER_HOST environment variable.

```

rafael@rafael-VirtualBox:~/MIOT2020/TestRepoDAW/daw_tp_final$ sudo apt install docker-compose

```

[sudo] password for rafaél:

Reading package lists... Done

Building dependency tree

Reading state information... Done

docker-compose is already the newest version (1.8.0-2~16.04.1).

The following packages were automatically installed and are no longer required:

```

libappindicator1 libindicator7 liblvm5.0 liblvm5.0:i386
linux-headers-4.13.0-36 linux-headers-4.13.0-36-generic
linux-headers-4.13.0-37 linux-headers-4.13.0-37-generic
linux-headers-4.13.0-38 linux-headers-4.13.0-38-generic
linux-headers-4.13.0-39 linux-headers-4.13.0-39-generic
linux-headers-4.13.0-41 linux-headers-4.13.0-41-generic
linux-headers-4.13.0-45 linux-headers-4.13.0-45-generic
linux-headers-4.15.0-24 linux-headers-4.15.0-24-generic
linux-headers-4.15.0-29 linux-headers-4.15.0-29-generic
linux-headers-4.15.0-30 linux-headers-4.15.0-30-generic
linux-headers-4.15.0-33 linux-headers-4.15.0-33-generic
linux-headers-4.15.0-34 linux-headers-4.15.0-34-generic
linux-headers-4.15.0-46 linux-headers-4.15.0-46-generic
linux-headers-4.15.0-50 linux-headers-4.15.0-50-generic
linux-headers-4.15.0-51 linux-headers-4.15.0-51-generic
linux-image-4.13.0-36-generic linux-image-4.13.0-37-generic
linux-image-4.13.0-38-generic linux-image-4.13.0-39-generic
linux-image-4.13.0-41-generic linux-image-4.13.0-45-generic
linux-image-4.15.0-24-generic linux-image-4.15.0-29-generic
linux-image-4.15.0-30-generic linux-image-4.15.0-33-generic
linux-image-4.15.0-34-generic linux-image-4.15.0-46-generic
linux-image-4.15.0-50-generic linux-image-4.15.0-51-generic
linux-image-extra-4.13.0-36-generic linux-image-extra-4.13.0-37-generic
linux-image-extra-4.13.0-38-generic linux-image-extra-4.13.0-39-generic
linux-image-extra-4.13.0-41-generic linux-image-extra-4.13.0-45-generic
linux-modules-4.15.0-24-generic linux-modules-4.15.0-29-generic
linux-modules-4.15.0-30-generic linux-modules-4.15.0-33-generic
linux-modules-4.15.0-34-generic linux-modules-4.15.0-46-generic
linux-modules-4.15.0-50-generic linux-modules-4.15.0-51-generic
linux-modules-extra-4.15.0-30-generic linux-modules-extra-4.15.0-33-generic
linux-modules-extra-4.15.0-34-generic linux-modules-extra-4.15.0-46-generic

```



```
linux-modules-extra-4.15.0-50-generic linux-modules-extra-4.15.0-51-generic
```

Use 'sudo apt autoremove' to remove them.

0 upgraded, 0 newly installed, 0 to remove and 415 not upgraded.

```
rafael@rafael-VirtualBox:~/MIoT2020/TestRepoDAW/daw_tp_final$ docker-compose upERROR: Version in "./docker-compose.yml" is unsupported. You might be seeing this error because you're using the wrong Compose file version. Either specify a version of "2" (or "2.0") and place your service definitions under the `services` key, or omit the `version` key and place your service definitions at the root of the file to use version 1. For more on the Compose file format versions, see https://docs.docker.com/compose/compose-file/
```

CAMBIANDO VERSION DE YML DE 3 A 2, O REMOVIÉNDOLA..

```
rafael@rafael-VirtualBox:~/MIoT2020/TestRepoDAW/daw_tp_final$ docker-compose up
```

```
ERROR: The Compose file './docker-compose.yml' is invalid because:
```

```
Unsupported config option for networks: 'mysql-net'
```

II.o) Se hicieron estas aclaraciones en el README.md

README.md

Autor: Rafael Oliva - 2020

Introducción

Este proyecto es el trabajo final de la materia Desarrollo de Aplicaciones Web (DAW) de la EspIoT / MIoT 2020 (FIUBA), y consiste en un Sistema Smart Home para control on/off de luces y persianas en una casa. Mediante los tres botones situados en el frente es posible visualizar todos los dispositivos, solo las luces o solo las persianas. Cada dispositivo tiene su propio switch on/off. El sistema se programó utilizando como base material de la cátedra, con un front-end basado en HTML y CSS (Materialize), y parte dinámica en Typescript. Para el backend, se consulta a una base de datos MySQL utilizando node-js. A través de la utilización de contenedores Docker y la herramienta Docker-Compose el sistema puede ponerse en marcha y detenerse con un único par de comandos.

Correr la aplicación

Para correr la aplicacion es necesario primero clonar el repositorio en la carpeta local deseada utilizando:

```
git clone https://github.com/rafaeloliva/daw_tp_final
```

y luego ejecutar los siguientes comandos con un terminal desde la carpeta seleccionada:

```
cd daw_tp_final
docker-compose up
```

Esperar que termine de cargar, y luego la aplicación podrá verse desde Chrome con <http://localhost:8000>

Para detener ordenadamente la aplicación, desde otro terminal ejecutar:

```
docker-compose down
```

Notas / Known issues

-Testeado en maquinas con Ubuntu 18.04, previamente ejecutar el Software Updater.

-Aun así puede requerir instalación de docker-compose, con sudo apt install docker-compose

-Se testeó en una máquina con Ubuntu 16.04 pero docker-compose no permitía ejecutar la secuencia, aun bajando la Versión YAML a 2 u omitiéndola completamente.

-Puede ocurrir que el arranque se salga de secuencia, el último container debe ser el de node. Si no ocurre, correr desde otro terminal docker-compose down, y luego nuevamente docker-compose up.

Contribuir

Para contribuir realizar un pull request con las sugerencias. Dado que se trata de un proyecto didáctico no se prevé realizar mantenimiento del mismo.

Licencia

GPL