

Ejercicios D.A.W1 R.Oliva 06.2020

Ejercicios / Upd 16.6.2020

Ejercicio 1:

Desarrollo de Aplicaciones I – FIUBA

Ejercicio 1

- 1) Generar un sitio Web con la misma estructura que el ejercicio de maquetado y agregarle la carpeta src.
- 2) Dentro de esta carpeta generar el archivo “main.ts” e incluir el archivo “tsconfig.json” con la configuración para el compilador de TS como se detalla a continuación.

```
{
  "compilerOptions": {
    "target": "es6",
    "sourceMap": false,
    "outDir": "../js/"
  }
}
```

- 3) Incluir en el archivo “index.html” el archivo de JS generado “main.js” para que se ejecute al cargar la página.

- 4) Agregar dentro del archivo main.ts el código:

```
console.log("Hola mundo");
```

- 5) Compilar el src* con el comando:

```
tsc -watch
```

*o si utilizamos Docker, ejecutar el script que lanza el contenedor dentro de src:

```
./compile_ts.sh . ../js/
```

- 6) Hostear el sitio web* con el servidor estático:

```
http-server . -c-1
```

Siendo “.” el path al directorio actual.

*o si utilizamos docker, ejecutar el script que lanza el contenedor:

```
./serve_http.sh . 8000
```

- 7) Abrir el navegador y el “inspector” y chequear el funcionamiento del programa.

Layout:

📁 C:\MIoT2020\Materias\DesarrolloAppWeb\delSitio\zip_deGhub22042020\da1-...	12.061,4 KB	12.516,0 KB	172
📁 ts_ej1	855,7 KB	884,0 KB	10
📁 css	313,6 KB	316,0 KB	2
materialize.css	175,0 KB	176,0 KB	1
materialize.min.css	138,5 KB	140,0 KB	1
📁 js	539,6 KB	552,0 KB	4
main.js	0,0 KB	4,0 KB	1
materialize.js	362,3 KB	364,0 KB	1
materialize.min.js	176,9 KB	180,0 KB	1
MyFramework.js	0,4 KB	4,0 KB	1
📁 src	0,5 KB	12,0 KB	3
compile_ts.sh	0,4 KB	4,0 KB	1
main.ts	0,0 KB	4,0 KB	1
tsconfig.json	0,1 KB	4,0 KB	1
📁 [Files]	2,1 KB	4,0 KB	1
index.html	2,1 KB	4,0 KB	1

Todos los CSS files se obtienen del sitio MaterializeCSS.com

In the /src

a) main.ts

```
tsconfig.json x compile_ts.sh x main.ts x
1 console.log("Hola mundo");
2
3
4
```

b) compile_ts.sh (Script in Linux)

```
C:\MIoT2020\Materias\DesarrolloAppWeb\delSitio\zip_deGhub22042020\da1-master\ejercicios\Ejercicios_ts\ts_ej1\src\compile_domingo, 17 de mayo de 2020 13:18
1 #!/bin/bash
2 CONTAINER_NAME=ts_compiler
3 INPUT_DIR=`realpath $1`
4 OUTPUT_DIR=`realpath $2`
5 echo "Init TypeScript compiler {container:$CONTAINER_NAME,
6   input:$INPUT_DIR,output:$OUTPUT_DIR}"
7 docker run -it --rm --name $CONTAINER_NAME --user "$(id -u):$(id -g)" --volume
8   $INPUT_DIR:/workspace --volume $OUTPUT_DIR:/output harmish/typescript tsc --project
9   /workspace --outDir /output --watch --pretty true
```

c) tsconfig.json (version de 2019, con ES5)

```
tsconfig.json x compile_ts.sh x main.ts x
1 {
2   "compilerOptions": {
3     "target": "es5",
4     "sourceMap": false,
5     "outDir": "../js/"
6   }
7 }
8
```

d) El compilador typescript genera el archivo main.js (aquí resulta idéntico al main.ts), que a su vez es llamado por el index.html

d.1) index.html original de la cátedra

```
<!DOCTYPE html>
<html>
  <head>
    <!--Import Google Icon Font-->
    <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
    <!--Import materialize.css-->
    <link type="text/css" rel="stylesheet" href="css/materialize.min.css" media="screen,projection"/>

    <!--Let browser know website is optimized for mobile-->
    <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
```

```

</head>

<body>

    <nav class="light-blue darken-4">
        <div class="nav-wrapper">
            <a href="#" class="brand-logo right">Smart HOME</a>
            <ul id="nav-mobile" class="left hide-on-med-and-down">
                <li class="active"><a href="#">Dispositivos</a></li>
                <li><a href="#">Configuracion</a></li>
            </ul>
        </div>
    </nav>

    <div class="container">
        <br/>
        <br/>
        <br/>
    </div>

    <footer class="page-footer light-blue darken-4">
        <div class="container">
            <div class="row">
                <div class="col l6 s12">
                    <h5 class="white-text">FIUBA</h5>
                    <p class="grey-text text-lighten-4">Especializacion IoT - Gestion de Datos y Aplicaciones</p>
                </div>
                <div class="col l4 offset-l2 s12">
                    <h5 class="white-text">Desarrollo de Aplicaciones I</h5>
                    <ul>
                        <li><a class="grey-text text-lighten-3" href="#">Link 1</a></li>
                        <li><a class="grey-text text-lighten-3" href="#">Link 2</a></li>
                        <li><a class="grey-text text-lighten-3" href="#">Link 3</a></li>
                        <li><a class="grey-text text-lighten-3" href="#">Link 4</a></li>
                    </ul>
                </div>
            </div>
            <div class="footer-copyright">
                <div class="container">
                    2019
                    <!-- <a class="grey-text text-lighten-4 right" href="#">More Links</a> -->
                </div>
            </div>
        </div>
    </footer>
    <!--JavaScript at end of body for optimized loading-->
    <script type="text/javascript" src="js/materialize.min.js"></script>
    <script type="text/javascript" src="js/main.js"></script>

</body>

</html>

```

Or with correct indentation:

```
<doctype html>
<html>
<head>
<!--Import Google Icon Font-->
<link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
<!--Import materialize.css-->
<link type="text/css" rel="stylesheet" href="css/materialize.min.css" media="screen,projection"/>

<!--Let browser know website is optimized for mobile-->
<meta name="viewport" content="width=device-width, initial-scale=1.0"/>
</head>

<body>

<nav class="light-blue darken-4">
<div class="nav-wrapper">
<a href="#" class="brand-logo right">Smart HOME</a>
<ul id="nav-mobile" class="left hide-on-med-and-down">
<li class="active"><a href="#">Dispositivos</a></li>
<li><a href="#">Configuracion</a></li>
</ul>
</div>
</nav>

<div class="container">
<br/>
<br/>
<br/>
</div>

<footer class="page-footer light-blue darken-4">
<div class="container">
<div class="row">
<div class="col l6 s12">
<h5 class="white-text">FIUBA</h5>
<p class="grey-text text-lighten-4">Especializacion IoT - Gestion de Datos y Aplicaciones</p>
</div>
<div class="col l4 offset-l2 s12">
<h5 class="white-text">Desarrollo de Aplicaciones I</h5>
<ul>
<li><a class="grey-text text-lighten-3" href="#">Link 1</a></li>
<li><a class="grey-text text-lighten-3" href="#">Link 2</a></li>
<li><a class="grey-text text-lighten-3" href="#">Link 3</a></li>
<li><a class="grey-text text-lighten-3" href="#">Link 4</a></li>
</ul>
</div>
</div>
<div class="footer-copyright">
<div class="container">
2019
<!-- <a class="grey-text text-lighten-4 right" href="#">More Links</a> -->
</div>
</div>

<!--JavaScript at end of body for optimized loading-->
<script type="text/javascript" src="js/materialize.min.js"></script>
<script type="text/javascript" src="js/main.js"></script>

</body>

</html>
```

Ejercicio 2: modifica el Ejercicio 1, con similar estructura, pero dice:

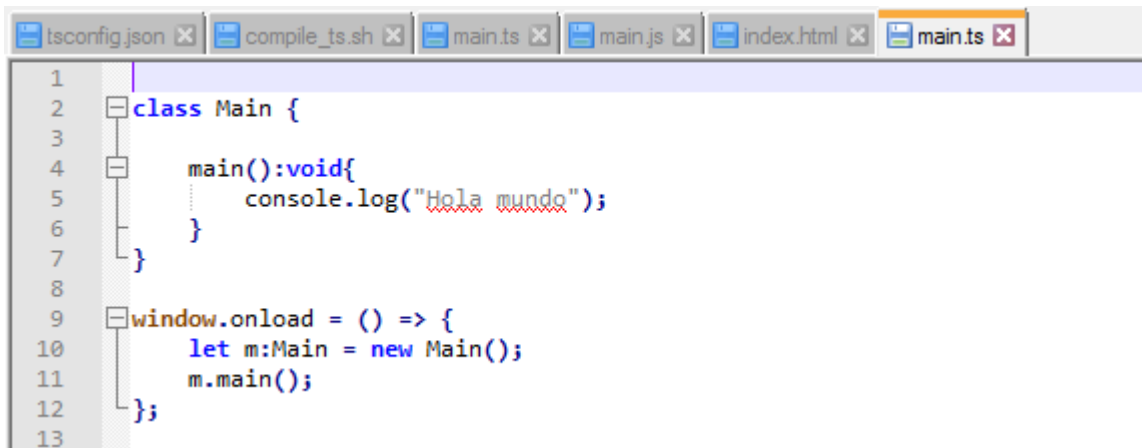
Ejercicio 2

- 1) Dentro del archivo main.ts del ejercicio anterior, definir la clase “Main” y dentro de la misma definir un método “main”. Dentro del método imprimir un mensaje.
- 2) Utilizar la clásica función “onload” de JS para crear un objeto del tipo Main y ejecutar el método “main()”.

Entonces:

ts_ej2	856,1 KB	884,0 KB	10	3	7,1 %	21/08/2019
css	313,6 KB	316,0 KB	2	0	35,7 %	21/08/2019
materialize.css	175,0 KB	176,0 KB	1	0	55,7 %	21/08/2019
materialize.min.css	138,5 KB	140,0 KB	1	0	44,3 %	21/08/2019
js	539,8 KB	552,0 KB	4	0	62,4 %	21/08/2019
main.js	0,2 KB	4,0 KB	1	0	0,7 %	21/08/2019
materialize.js	362,3 KB	364,0 KB	1	0	65,9 %	21/08/2019
materialize.min.js	176,9 KB	180,0 KB	1	0	32,6 %	21/08/2019
MyFramework.js	0,4 KB	4,0 KB	1	0	0,7 %	21/08/2019
src	0,6 KB	12,0 KB	3	0	1,4 %	21/08/2019
compile_ts.sh	0,4 KB	4,0 KB	1	0	33,3 %	21/08/2019
main.ts	0,1 KB	4,0 KB	1	0	33,3 %	21/08/2019
tsconfig.json	0,1 KB	4,0 KB	1	0	33,3 %	21/08/2019
[Files]	2,1 KB	4,0 KB	1	0	0,5 %	21/08/2019
index.html	2,1 KB	4,0 KB	1	0	0,5 %	21/08/2019

Pero main.ts se modifica a:



```
1
2 class Main {
3
4     main():void{
5         console.log("Hola mundo");
6     }
7 }
8
9 window.onload = () => {
10     let m:Main = new Main();
11     m.main();
12 };
13
```

Y esto se traduce (versión 2019) a un main.js mas complejo

```
var Main = /** @class */ (function () {
    function Main() {
    }
    Main.prototype.main = function () {
        console.log("Hola mundo");
    };
    return Main;
})();
window.onload = function () {
    var m = new Main();
    m.main();
};
```

Probablemente porque todavía está compilando para es5 y no es6 que tiene clases:

Este es el config.json

```
{
  "compilerOptions": {
    "target": "es5",
    "sourceMap": false,
    "outDir": "../js/"
  }
}
```

Ejercicio 3: Ejercicio 1, con similar estructura pero:

Ejercicio 3

- 1) Definir la clase “User” con los atributos privados “id” (numérico) “name”, “email” (textos) y “isLoggedIn” (booleano). El constructor de la clase debe recibir el ID, el email y el nombre.
- 2) Agregar getters y setters para los atributos.
- 3) Agregar el método “printInfo” el cual imprime por consola los tres atributos.
- 4) En la clase Main, en el método “main” definir un array de 3 objetos “User”. Luego iterar el array y ejecutar el método printInfo() de cada objeto.

Dentro del main.ts:

```
class User {
    private _id:number;
    private _name:string;
    private _email:string;
    private _isLoggedIn:boolean;

    constructor(id:number, email:string, name:string){
        this.id = id;
        this.email = email;
        this.name = name;
        this.isLoggedIn = false;
    }
}
```

```

set id(id:number){
  this._id = id;
}
get id():number {
  return this._id;
}
set email(email:string){
  this._email = email;
}
get email():string {
  return this._email;
}
set name(name:string){
  this._name = name;
}
get name():string {
  return this._name;
}
set isLogged(flag:boolean){
  this._isLogged = flag;
}
get isLogged():boolean{
  return this._isLogged;
}

public printInfo():void{
  console.log("Nombre:"+this.name+" email:"+this.email+" logged:"+this.isLogged);
}
}

class Main {
  main():void{
    console.log("Hola mundo");
    let usuarios:Array<User>;
    usuarios = new Array<User>();
    usuarios.push(new User(1,"juan@juan.com","Juan"));
    usuarios.push(new User(2,"pepe@juan.com","Pepe"));
    usuarios.push(new User(3,"carlos@juan.com","Carlos"));

    for(let i in usuarios){
      usuarios[i].printInfo();
    }
  }
}

window.onload = () => {
  let m:Main = new Main();
  m.main();
};

```

EJECUCION EN CHROME / F12:

The screenshot shows a web browser window with the address bar displaying a file path. The page has a blue header with 'Smart HOME' and a main content area with a dark blue background. The console output shows the following messages:

Message	Source
Hola mundo	main.js:57
Nombre:Juan email:juan@juan.com logged:false	main.js:48
Nombre:Pepe email:pepe@juan.com logged:false	main.js:48
Nombre:Carlos email:carlos@juan.com logged:false	main.js:48

Vemos como queda formateado el main.ts:

```
2 class User {
3   private _id:number;
4   private _name:string;
5   private _email:string;
6   private _isLoggedIn:boolean;
7
8   constructor(id:number, email:string, name:string){
9     this.id = id;
10    this.email = email;
11    this.name = name;
12    this.isLoggedIn = false;
13  }
14
15  set id(id:number){
16    this._id = id;
17  }
18  get id():number {
19    return this._id;
20  }
21  set email(email:string){
22    this._email = email;
23  }
24  get email():string {
25    return this._email;
26  }
27  set name(name:string){
28    this._name = name;
29  }
30  get name():string {
31    return this._name;
32  }
33  set isLoggedIn(flag:boolean){
34    this._isLoggedIn = flag;
35  }
36  get isLoggedIn():boolean{
37    return this._isLoggedIn;
38  }
39
40  public printInfo():void{
41    console.log("Nombre:" +this.name+ " email:" +this.email+ " logged:" +this.isLoggedIn);
42  }
43 }
44
45
46 class Main {
47
48   main():void{
49     console.log("Hola mundo");
50     let usuarios:Array<User>;
51     usuarios = new Array<User>();
52     usuarios.push(new User(1,"juan@juan.com","Juan"));
53     usuarios.push(new User(2,"pepe@juan.com","Pepe"));
54     usuarios.push(new User(3,"carlos@juan.com","Carlos"));
55
56     for(let i in usuarios){
57       usuarios[i].printInfo();
58     }
59   }
60 }
61
62 window.onload = () => {
63   let m:Main = new Main();
64   m.main();
65 };
66
```

En la versión ES5 esto se traduce en un main.js mas "enrevesado":

```

1 var User = /** @class */ (function () {
2     function User(id, email, name) {
3         this.id = id;
4         this.email = email;
5         this.name = name;
6         this.isLogged = false;
7     }
8     Object.defineProperty(User.prototype, "id", {
9         get: function () {
10             return this._id;
11         },
12         set: function (id) {
13             this._id = id;
14         },
15         enumerable: true,
16         configurable: true
17     });
18     Object.defineProperty(User.prototype, "email", {
19         get: function () {
20             return this._email;
21         },
22         set: function (email) {
23             this._email = email;
24         },
25         enumerable: true,
26         configurable: true
27     });
28     Object.defineProperty(User.prototype, "name", {
29         get: function () {
30             return this._name;
31         },
32         set: function (name) {
33             this._name = name;
34         },
35         enumerable: true,
36         configurable: true
37     });
38     Object.defineProperty(User.prototype, "isLogged", {
39         get: function () {
40             return this._isLogged;
41         },
42         set: function (flag) {
43             this._isLogged = flag;
44         },
45         enumerable: true,
46         configurable: true
47     });
48     User.prototype.printInfo = function () {
49         console.log("Nombre:" + this.name + " email:" + this.email + " logged:" + this.isLogged);
50     };
51     return User;
52 }());
53 var Main = /** @class */ (function () {
54     function Main() {
55     }
56     Main.prototype.main = function () {
57         console.log("Hola mundo");
58         var usuarios;
59         usuarios = new Array();
60         usuarios.push(new User(1, "juan@juan.com", "Juan"));
61         usuarios.push(new User(2, "pepe@juan.com", "Pepe"));
62         usuarios.push(new User(3, "carlos@juan.com", "Carlos"));
63         for (var i in usuarios) {
64             usuarios[i].printInfo();
65         }
66     };
67     return Main;
68 }());
69 window.onload = function () {
70     var m = new Main();
71     m.main();

```

JavaScript file

Ejercicio 4, 4.2: Ejercicios 1 y 3, con similar estructura pero:

Ejercicio 4

- 1) Agregar en la maqueta HTML un botón con el id “boton”.
- 2) Crear la clase MyFramework la cual deberá tener el método “getElementById()” y devolverá el tipo de dato “HTMLElement”.
- 3) En el main, crear un objeto MyFramework y ejecutar el método para obtener la referencia del botón que se encuentra en el DOM.
- 4) Modificar el texto del botón desde el programa. (utilizar el atributo textContent del objeto)

Ejercicio 4-2

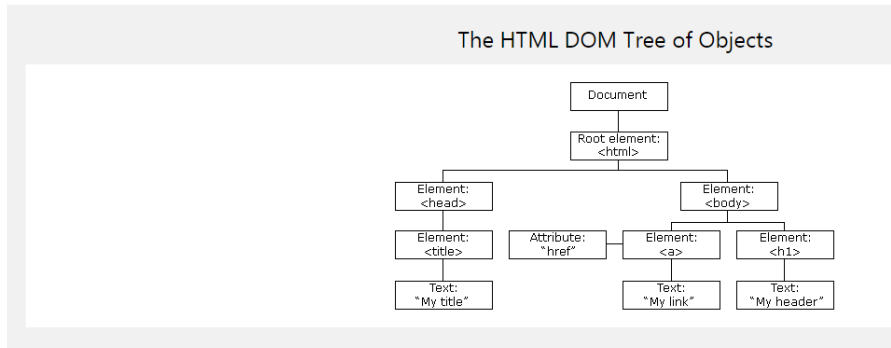
- 1) Agregar la clase User definida en el Ejercicio 3 al proyecto del ejercicio 4.
- 2) A la clase Main, agregarle el método “mostrarUsers” que reciba un array de objetos User y los imprima por la consola.
- 3) En el método main, definir un array de 3 objetos User e invocar al método “mostrarUsers” y pasarle el array.

Repaso:

The HTML DOM (Document Object Model)

When a web page is loaded, the browser creates a **Document Object Model** of the page.

The **HTML DOM** model is constructed as a tree of **Objects**:



With the object model, JavaScript gets all the power it needs to create dynamic HTML:

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

Para el Ejercicio 4:

Finding HTML Element by Id

The easiest way to find an HTML element in the DOM, is by using the element id.

This example finds the element with `id="intro"`:

Example

```
var myElement = document.getElementById("intro");
```

Try it Yourself »

If the element is found, the method will return the element as an object (in myElement).

If the element is not found, myElement will contain `null`.

```

<!DOCTYPE html>
<html>
<head>
<!--Import Google Icon Font-->
<link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
<!--Import materialize.css-->
<link type="text/css" rel="stylesheet" href="css/materialize.min.css" media="screen,projection"/>

<!--Let browser know website is optimized for mobile-->
<meta name="viewport" content="width=device-width, initial-scale=1.0"/>
</head>

<body>

<nav class="light-blue darken-4">
<div class="nav-wrapper">
<a href="#" class="brand-logo right">Smart HOME</a>
<ul id="nav-mobile" class="left hide-on-med-and-down">
<li class="active"><a href="#">Dispositivos</a></li>
<li><a href="#">Configuracion</a></li>
</ul>
</div>
</nav>

<div class="container">
<br/>
<a id="boton" class="waves-effect btn">Boton</a>
<br/>
<br/>
</div>

<footer class="page-footer light-blue darken-4">
<div class="container">
<div class="row">
<div class="col l6 s12">
<h5 class="white-text">FIUBA</h5>
<p class="grey-text text-lighten-4">Especializacion IoT - Gestion de Datos y Aplicaciones</p>
</div>
<div class="col l4 offset-l2 s12">
<h5 class="white-text">Desarrollo de Aplicaciones I</h5>
<ul>
<li><a class="grey-text text-lighten-3" href="#">Link 1</a></li>
<li><a class="grey-text text-lighten-3" href="#">Link 2</a></li>
<li><a class="grey-text text-lighten-3" href="#">Link 3</a></li>
<li><a class="grey-text text-lighten-3" href="#">Link 4</a></li>
</ul>
</div>
</div>
<div class="footer-copyright">
<div class="container">
2019
<!-- <a class="grey-text text-lighten-4 right" href="#">More Links</a> -->
</div>
</div>

<!--JavaScript at end of body for optimized loading-->
<script type="text/javascript" src="js/materialize.min.js"></script>
<script type="text/javascript" src="js/main.js"></script>
<script type="text/javascript" src="js/MyFramework.js"></script>

</body>

</html>

```

R4.1:Boton

MyFramework.js

main.ts

```

1  class Main
2  {
3      myf:MyFramework;
4
5      main():void
6      {
7          this.myf = new MyFramework();
8
9          let b:HTMLElement = this.myf.getElementById("boton");
10
11          console.log(b);
12
13          b.textContent = "Haceme click!";
14      }
15  }
16
17
18  window.onload = () => {
19      let obj = new Main();
20      obj.main();
21  };

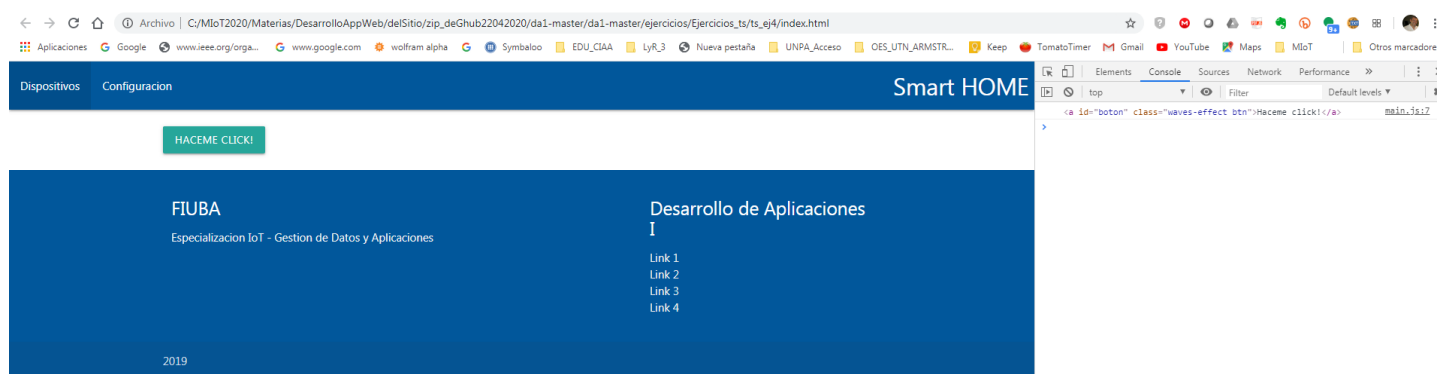
```

Modifico atributo textContent del Boton

El MyFramework está en un archivo aparte:

```
MyFramework.ts
1 class MyFramework{
2
3
4   /**
5    * getElementById: Busca un elemento del DOM por su ID
6    * @param id : String con el id a buscar
7    * @returns : Objeto HTMLElement encontrado
8    */
9   getElementById(id:string):HTMLElement
10  {
11    let el:HTMLElement;
12    el = document.getElementById(id);
13    return el;
14  }
15
16
17 }
18
```

Resultado: En el main.ts se modificó el texto interno del boton



Para el Ejercicio 4.2, agregamos la clase User en un archivo User.ts, que compila a User.js:

```
<!DOCTYPE html>
<html>
<head>
  <!--Import Google Icon Font-->
  <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
  <!--Import Materialize CSS-->
  <link type="text/css" rel="stylesheet" href="css/materialize.min.css" media="screen,projection"/>

  <!--Let browser know website is optimized for mobile-->
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
</head>

<body>

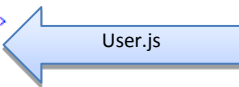
  <nav class="light-blue darken-4">
    <div class="nav-wrapper">
      <a href="#" class="brand-logo right">Smart HOME</a>
      <ul id="nav-mobile" class="left hide-on-med-and-down">
        <li class="active"><a href="#">Dispositivos</a></li>
        <li><a href="#">Configuracion</a></li>
      </ul>
    </div>
  </nav>

  <div class="container">
    <br/>
    <a id="boton" class="waves-effect btn">Boton</a>
    <br/>
    <ul id="listaUsuarios">
    </ul>
  </div>

  <footer class="page-footer light-blue darken-4">
    <div class="container">
      <div class="row">
        <div class="col l6 s12">
          <h5 class="white-text">FIUBA</h5>
          <p class="grey-text text-lighten-4">Especializacion IoT - Gestion de Datos y Aplicaciones</p>
        </div>
        <div class="col l4 offset-l2 s12">
          <h5 class="white-text">Desarrollo de Aplicaciones I</h5>
          <ul>
            <li><a class="grey-text text-lighten-3" href="#">Link 1</a></li>
            <li><a class="grey-text text-lighten-3" href="#">Link 2</a></li>
            <li><a class="grey-text text-lighten-3" href="#">Link 3</a></li>
            <li><a class="grey-text text-lighten-3" href="#">Link 4</a></li>
          </ul>
        </div>
      </div>
      <div class="footer-copyright">
        <div class="container">
          2019
          <!-- <a class="grey-text text-lighten-4 right" href="#">More Links</a> -->
        </div>
      </div>
    </div>

  <!--JavaScript at end of body for optimized loading-->
  <script type="text/javascript" src="js/materialize.min.js"></script>
  <script type="text/javascript" src="js/User.js"></script>
  <script type="text/javascript" src="js/main.js"></script>
  <script type="text/javascript" src="js/MyFramework.js"></script>

</body>
</html>
```



MyFramework.ts queda igual:

```
1 class MyFramework{
2
3  /**
4   * getElementById: Busca un elemento del DOM por su ID
5   * @param id : String con el id a buscar
6   * @returns : Objeto HTMLElement encontrado
7   */
8  getElementById(id:string):HTMLElement
9  {
10   let el:HTMLElement;
11   el = document.getElementById(id);
12   return el;
13 }
14
15
16
17
18 }
```

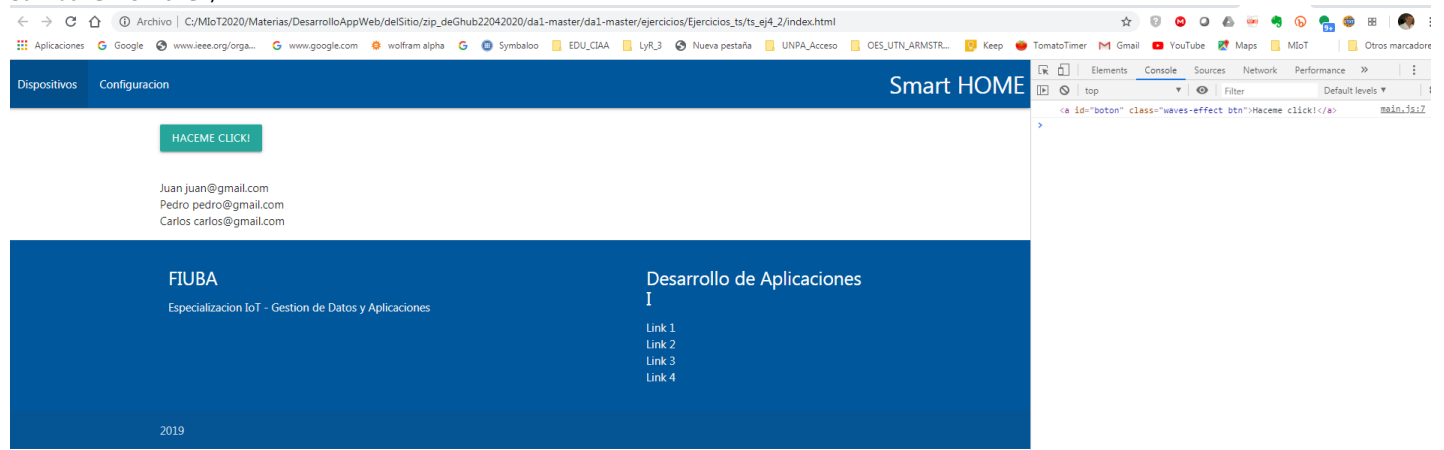
Ahora los User.ts, main.ts modificados son:

```
Users.ts | MyFramework.ts | main.ts
C:\MIoT2020\Materias\DesarrolloAppWeb\delSitio\zip_deGithub22042020\da1-master\da1-master\ejercicios\Ejercicios_ts\ts_ej4_2\src\Users.ts
2 class User {
3     private _id:number;
4     private _name:string;
5     private _email:string;
6     private _isLoggedIn:boolean;
7
8     constructor(id:number, email:string, name:string){
9         this.id = id;
10        this.email = email;
11        this.name = name;
12        this.isLoggedIn = false;
13    }
14
15    set id(id:number){
16        this._id = id;
17    }
18    get id():number {
19        return this._id;
20    }
21    set email(email:string){
22        this._email = email;
23    }
24    get email():string {
25        return this._email;
26    }
27    set name(name:string){
28        this._name = name;
29    }
30    get name():string {
31        return this._name;
32    }
33    set isLoggedIn(flag:boolean){
34        this._isLoggedIn = flag;
35    }
36    get isLoggedIn():boolean{
37        return this._isLoggedIn;
38    }
39
40    public printInfo():void{
41        console.log("Nombre: "+this.name+ " email: "+this.email+ " logged: "+this.isLoggedIn);
42    }
43
44 }
```

```
Users.ts | MyFramework.ts | main.ts
C:\MIoT2020\Materias\DesarrolloAppWeb\delSitio\zip_deGithub22042020\da1-master\da1-master\ejercicios\Ejercicios_ts\ts_ej4_2\src\main.ts
1 class Main
2 {
3     myf:MyFramework;
4
5     main():void
6     {
7         this.myf = new MyFramework();
8
9         let b:HTMLElement = this.myf.getElementById("boton");
10
11         console.log(b);
12
13         b.textContent = "Hazeme click!";
14
15         let users:Array<User> = [];
16         users.push(new User(1,"juan@gmail.com","Juan"));
17         users.push(new User(2,"pedro@gmail.com","Pedro"));
18         users.push(new User(3,"carlos@gmail.com","Carlos"));
19
20         this.mostrarUsers(users);
21     }
22
23     mostrarUsers(users:Array<User>):void {
24         /*
25         let items:string="";
26         for(let i in users){
27             users[i].printInfo();
28             items+="- " +users[i].name+ " "+users[i].email+"</li>";
29         }
30         */
31
32         let strTemplate:string=`${
33             users.map((item) =>`<li>${item.name} ${item.email}</li>`).join('')
34         }`;
35
36         let ul:HTMLElement = this.myf.getElementById("listaUsuarios");
37         ul.innerHTML = strTemplate;
38     }
39 }
40
41 window.onload = () => {
42     let obj = new Main();
43     obj.main();
44 };
45

```

Salida en Chrome / F12



Gitflow Ernesto Clase 5

<https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow>

No incorporó obligatoriamente, pero interesante

Ejercicio 5: Retomamos Ejercicios 16.06.2020, intento con tsc desde Win10 complicado, volvemos a MV Ubuntu18.04 y actualizamos aquí.

Utilizamos los ejercicios hechos en clase, no los remitidos al repo privado que no fue visualizado por los profes.

Ejercicio 5

1) Partiendo del ejercicio 4, agregar un listener para escuchar el evento de click del botón, para ello, definir en la clase Main el siguiente método:

```
evento(ev:Event):void {  
    console.log("se hizo click!");  
}
```

2) Setear el listener el elemento botón mediante el método “addEventListener” de la siguiente manera:

```
objBoton.addEventListener("click",this.evento);
```

3) Verificar el funcionamiento.

4) Imprimir dentro del método “evento” el objeto “this” y verificar si es del tipo Main.

```
console.log(this);
```

SOLUCION 4.2a REPASO main.ts, con agregados EJ5. Users.ts ya no se utiliza

```
class Main  
{  
    myf:MyFramework;  
  
    main():void  
    {  
        this.myf = new MyFramework();  
  
        let b:HTMLElement = this.myf.getElementById("boton");  
  
        console.log(b);  
  
        b.textContent = "Haceme click!";  
  
        let users:Array<User> = [];  
        users.push(new User(1,"juan@gmail.com","Juan"));  
        users.push(new User(2,"pedro@gmail.com","Pedro"));  
        users.push(new User(3,"carlos@gmail.com","Carlos"));  
  
        this.mostrarUsers(users);  
    }  
  
    mostrarUsers(users:Array<User>):void {  
        /*  
        let items:string="";  
        for(let i in users){  
            users[i].printInfo();  
        }  
        */  
    }  
}
```

```
class Main  
{  
    myf:MyFramework;  
  
    main():void  
    {  
        this.myf = new MyFramework();  
  
        let b:HTMLElement = this.myf.getElementById("boton");  
        console.log(b);  
        b.textContent = "Haceme click!";  
  
        b.addEventListener("click", this.evento );  
  
        console.log("this en main:");  
        console.log(this);  
    }  
  
    evento(ev:Event):void {  
        console.log("se hizo click!");  
        console.log("this en evento:");  
        console.log(this);  
    }  
}
```

Esto se
reemplaza

2

1

3

```

        items+="- " + users[i].name + " " + users[i].email + "</li>";
    }
    */

    let strTemplate:string=`${
        users.map((item) =>`<li>${item.name} ${item.email}</li>`).join('')
    }`;

    let ul:HTMLElement = this.myf.getElementById("listaUsuarios");
    ul.innerHTML = strTemplate;
}
}

window.onload = () => {
    let obj = new Main();
    obj.main();
};

```

SOLUCION 4.2 REPASO MyFramework.ts

```

class MyFramework{

    /**
     * getElementById: Busca un elemento del DOM por su ID
     * @param id : String con el id a buscar
     * @returns : Objeto HTMLElement encontrado
     */
    getElementById(id:string):HTMLElement
    {
        let el:HTMLElement;
        el = document.getElementById(id);
        return el;
    }
}

```

Index.html del Ej 5 (elimina Users.js)

```

<!DOCTYPE html>
<html>
  <head>
    <!--Import Google Icon Font-->
    <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
    <!--Import materialize.css-->
    <link type="text/css" rel="stylesheet" href="css/materialize.min.css" media="screen,projection"/>

    <!--Let browser know website is optimized for mobile-->
    <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  </head>

  <body>

```



```

<nav class="light-blue darken-4">
  <div class="nav-wrapper">
    <a href="#" class="brand-logo right">Smart HOME</a>
    <ul id="nav-mobile" class="left hide-on-med-and-down">
      <li class="active"><a href="#">Dispositivos</a></li>
      <li><a href="#">Configuracion</a></li>
    </ul>
  </div>
</nav>

<div class="container">
  <br/>
  <a id="boton" class="waves-effect btn">Boton</a>
  <br/>
  <br/>
</div>

<footer class="page-footer light-blue darken-4">
  <div class="container">
    <div class="row">
      <div class="col l6 s12">
        <h5 class="white-text">FIUBA</h5>
        <p class="grey-text text-lighten-4">Especializacion IoT -
Gestion de Datos y Aplicaciones</p>
      </div>
      <div class="col l4 offset-l2 s12">
        <h5 class="white-text">Desarrollo de Aplicaciones I</h5>
        <ul>
          <li><a class="grey-text text-lighten-3" href="#">Link 1</a></li>
          <li><a class="grey-text text-lighten-3" href="#">Link 2</a></li>
          <li><a class="grey-text text-lighten-3" href="#">Link 3</a></li>
          <li><a class="grey-text text-lighten-3" href="#">Link 4</a></li>
        </ul>
      </div>
    </div>
  </div>
  <div class="footer-copyright">
    <div class="container">
      2019
      <!-- <a class="grey-text text-lighten-4 right" href="#">More Links</a> -->
    </div>
  </div>
</footer>

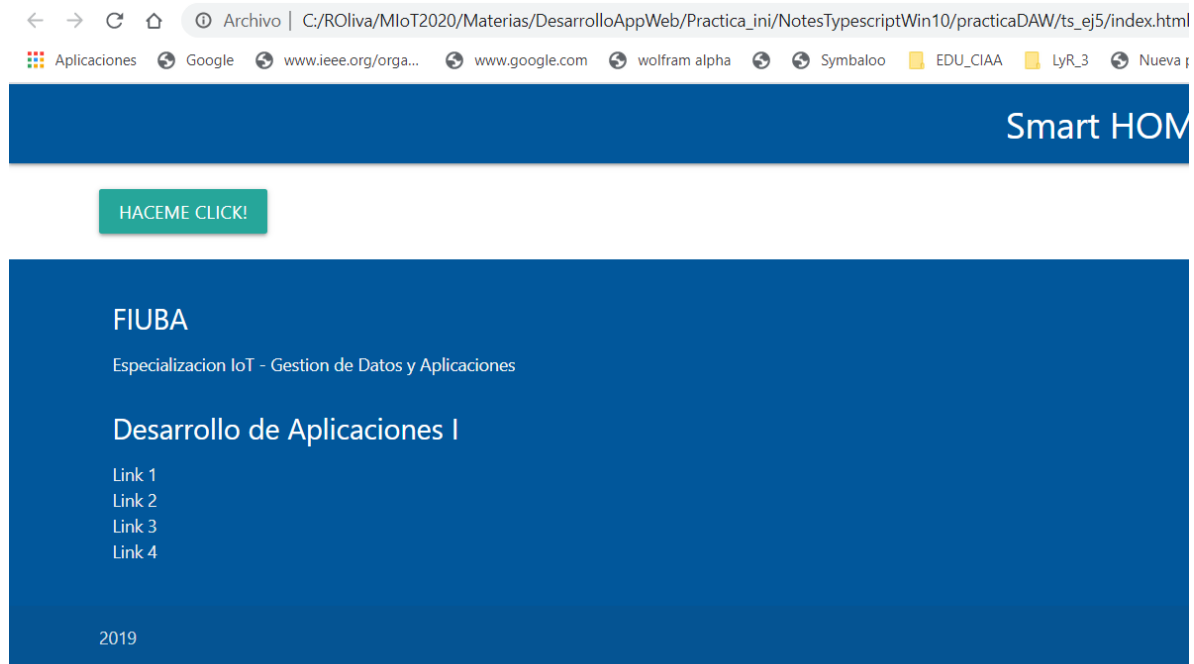
<!--JavaScript at end of body for optimized loading-->
<script type="text/javascript" src="js/materialize.min.js"></script>
<script type="text/javascript" src="js/main.js"></script>
<script type="text/javascript" src="js/MyFramework.js"></script>

</body>

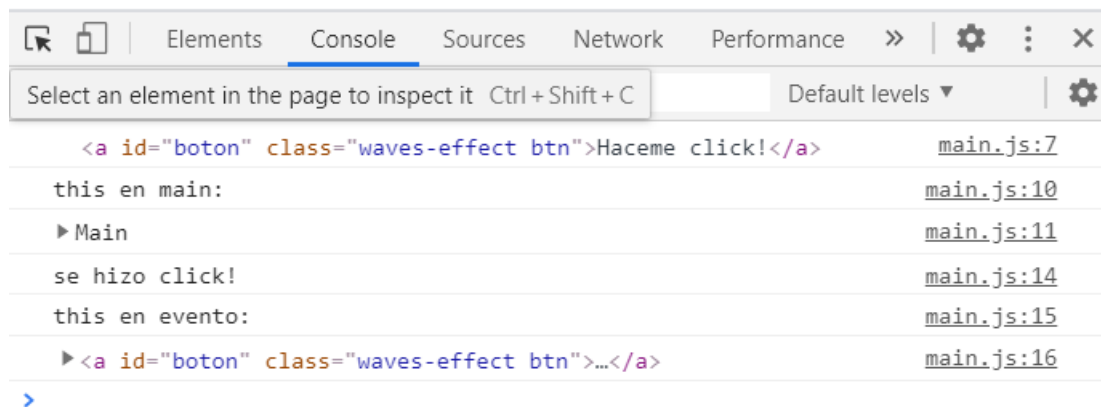
```

</html>

El resultado se muestra:



Y en la consola:



Ejercicio 6: Modifica el Ejercicio 5

Ejercicio 6

1) Partiendo del ejercicio 5, cambiar la manera de escuchar el evento utilizando interfaces. Para ello, se deberá implementar la interface “EventListenerObject”, esto nos obligará a implementar el método:

```
handleEvent(evt: Event): void
{
}
```

2) Escribir dentro del método un mensaje e imprimir el valor de “this”.

3) En la llamada a “addEventListener” ahora se deberá pasar el objeto que implementa la interfaz (this):

```
objBoton.addEventListener("click", this);
```

4) Probar y verificar que el objeto “this” que se imprime en el evento sea del tipo Main

```
src > TS main.ts > ...
1  class Main implements EventListenerObject
2  {
3      myf:MyFramework;
4
5
6      handleEvent(evt: Event): void
7      {
8          console.log("handleEvent");
9          console.log("this en handleEvent:");
10         console.log(this);
11     }
12
13     main():void
14     {
15         this.myf = new MyFramework();
16
17         let b:HTMLElement = this.myf.getElementById("boton")
18         console.log(b);
19         b.textContent = "Haceme click!";
20
21         b.addEventListener("click", this );
22
23         console.log("this en main:");
24         console.log(this);
25     }
26 }
27
28
29 window.onload = () => {
30     let obj = new Main();
31     obj.main();
32 };
```

Agregado en Ej. 6 :Antes evento era una función aparte, ahora la incorporamos a la clase Main

```
class Main
{
    myf:MyFramework;

    main():void
    {
        this.myf = new MyFramework();

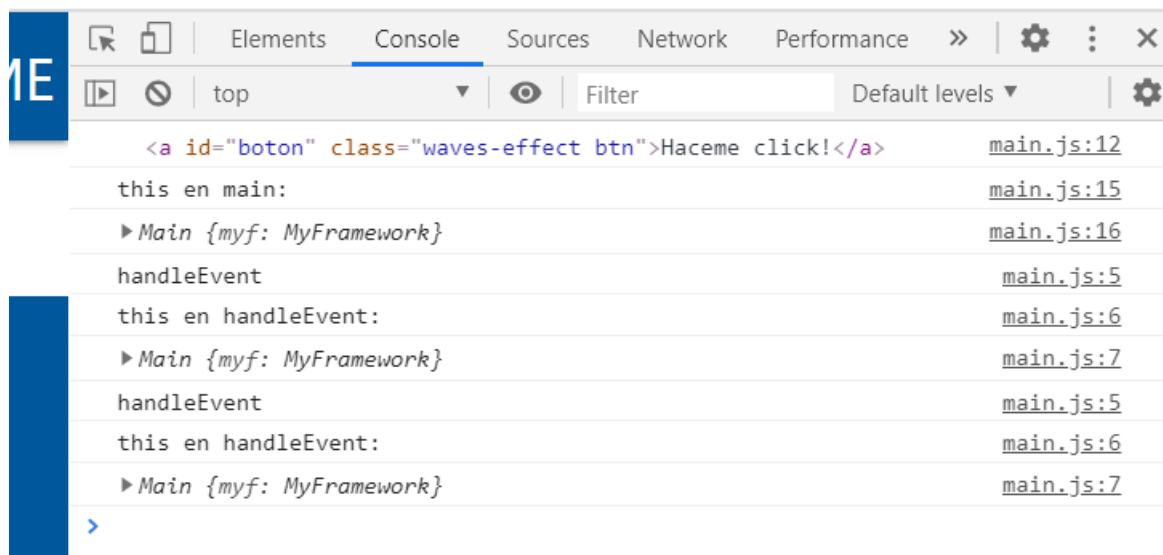
        let b:HTMLElement = this.myf.getElementById("boton");
        console.log(b);
        b.textContent = "Haceme click!";

        b.addEventListener("click", this.evento );

        console.log("this en main:");
        console.log(this);
    }

    evento(ev:Event):void {
        console.log("se hizo click!");
        console.log("this en evento:");
        console.log(this);
    }
}
```

Ahora Al arrancar, se imprime “this en main”



Ejercicio 7: Modifica el Ejercicio 6

Ejercicio 7

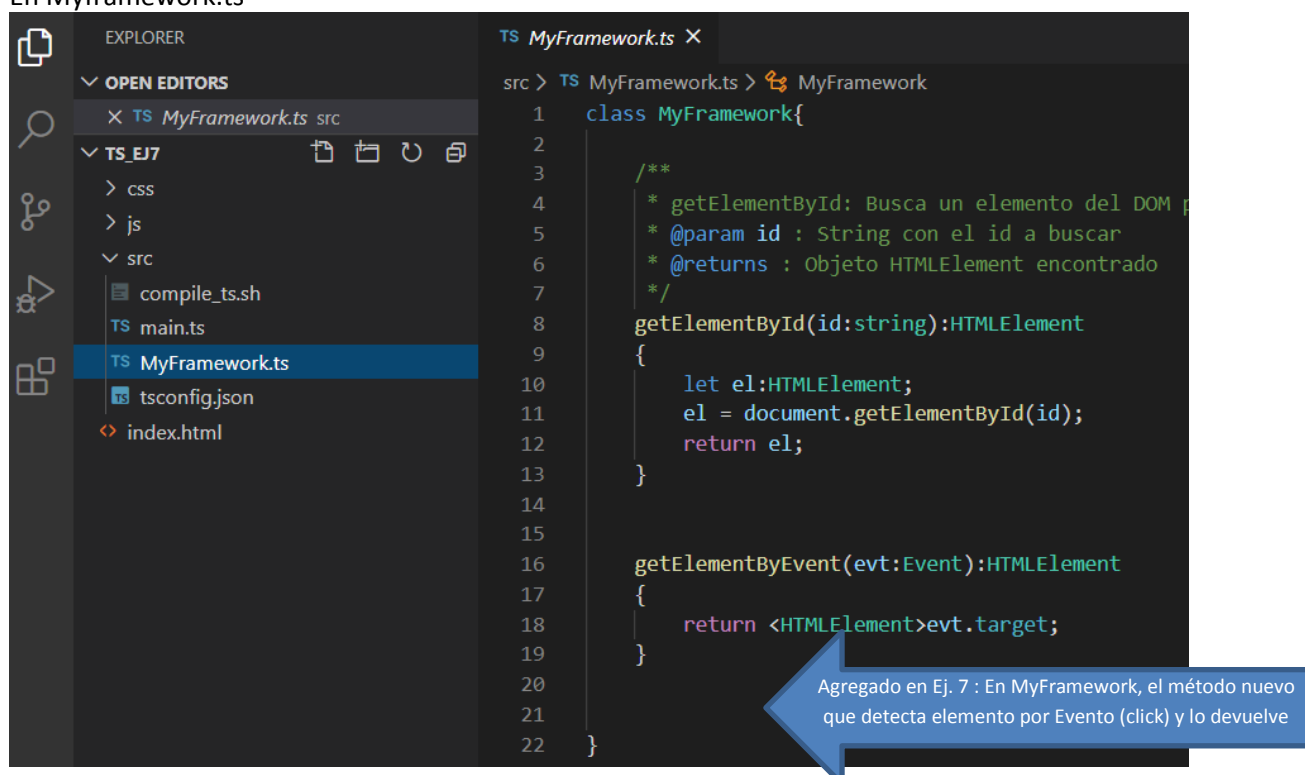
1) Agregar a la clase “MyFramework” un método que reciba un objeto “Event” y devuelva el objeto HTMLElement asociado, se debe llamar “getElementByEvent”.

Dentro del método colocar:

```
return <HTMLElement>evt.target
```

2) Utilizar este método en el ejercicio anterior para obtener el elemento sobre el que se produjo el evento. Cambiar el texto al hacer click sobre el botón y mostrar un contador de clicks.

En Myframework.ts



```
src > TS MyFramework.ts > MyFramework
1 class MyFramework{
2
3   /**
4    * getElementById: Busca un elemento del DOM
5    * @param id : String con el id a buscar
6    * @returns : Objeto HTMLElement encontrado
7    */
8   getElementById(id:string):HTMLElement
9   {
10    let el:HTMLElement;
11    el = document.getElementById(id);
12    return el;
13  }
14
15
16   getElementByEvent(evt:Event):HTMLElement
17   {
18    return <HTMLElement>evt.target;
19  }
20
21
22 }
```

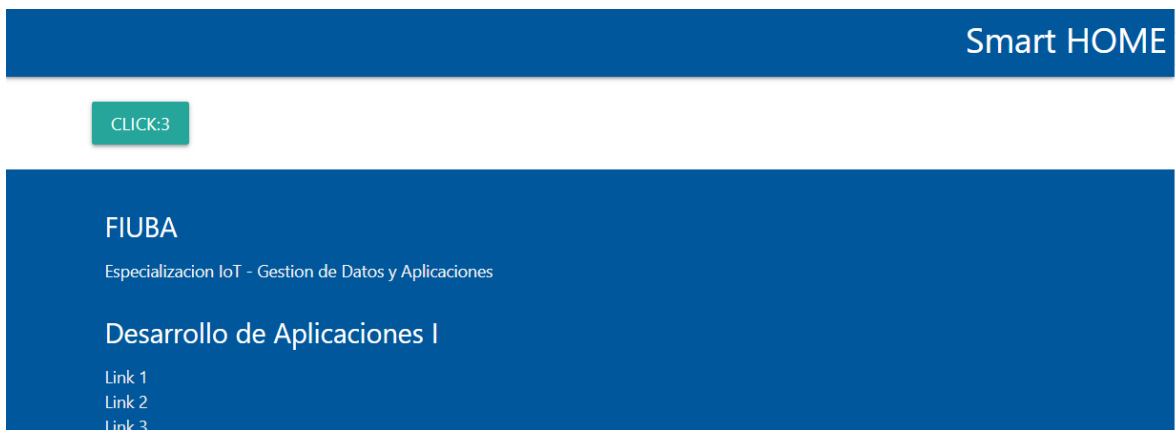
Agregado en Ej. 7 : En MyFramework, el método nuevo que detecta elemento por Evento (click) y lo devuelve

En Main.ts

```
1 class Main implements EventListenerObject
2 {
3   myf:MyFramework;
4   counter:number=0;
5
6   handleEvent(evt: Event): void
7   {
8     this.counter++;
9     let btn:HTMLElement = this.myf.getElementByEvent(evt);
10    btn.textContent = "click:"+this.counter;
11  }
12
13  main():void
14  {
15    this.myf = new MyFramework();
16
17    let b:HTMLElement = this.myf.getElementById("boton");
18    console.log(b);
19    b.textContent = "Haceme click!";
20
21    b.addEventListener("click", this );
22  }
23 }
24
25
26 window.onload = () => {
27   let obj = new Main();
28   obj.main();
29 };
30
```

Agregado en Ej. 7 : en main.ts, contador. Del Framework, obtenemos el Boton, e incrementamos el contador, además lo mostramos en el texto del boton

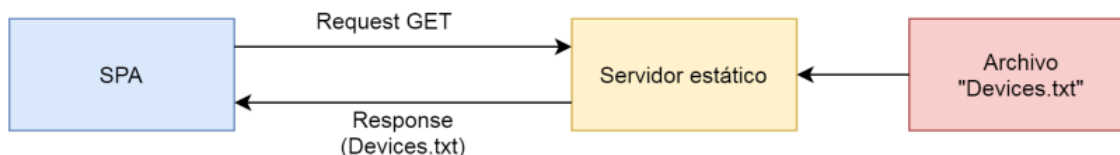
No muestra nada por consola, solo el incremento del contador en el texto del botón, ante cada click



Ejercicio 8: Modifica el Ejercicio 7 → requests tipo GET a un server estático

Ejercicio 8

Agregaremos a “MyFramework” un método que nos permita hacer requests del tipo GET. De esta forma, al iniciar la aplicación se ejecutará un request GET al archivo “devices.txt” (el cual deberemos crear) para que la página pueda leer la información que hay en él y posteriormente mostrar la lista de dispositivos de la “smart home”.



1) En MyFramework, definiremos una interfaz que nos permitirá definir el evento que se ejecute al llegar la respuesta del servidor:

```
interface GETResponseListener{
    handleGETResponse(status:number, response:string):void;
}
```

La función que realice al request, deberá recibir un objeto del tipo “GETResponseListener” como argumento, cuando se reciba la respuesta del server, a dicho objeto se le ejecutará el método “handleGETResponse”.

2) Crear el archivo “Devices.txt” en la raíz del sitio web, dentro del mismo colocar el texto “Hola mundo”.

En Myframework.ts,

8.a) al inicio se agrega la interface GETResponseListener{..}, que contiene el Handler para el GET.

```
src > TS MyFramework.ts > GETResponseListener
1 interface GETResponseListener{
2     handleGETResponse(status:number,response:string):void;
3 }
4
5 class MyFramework{
6
7     /**
8      * getElementById: Busca un elemento del DOM por su ID
9      * @param id : String con el id a buscar
10     * @returns : Objeto HTMLElement encontrado
11     */
12     getElementById(id:string):HTMLElement
```

8. → 2) devices.txt por ahora solo contiene “Hola Mundo”

Continuacion para parte 8b):

3) Agregar el método “requestGET” a la clase “MyFramework” con la siguiente firma:

```
requestGET(url:string, listener: GETResponseListener):void
```

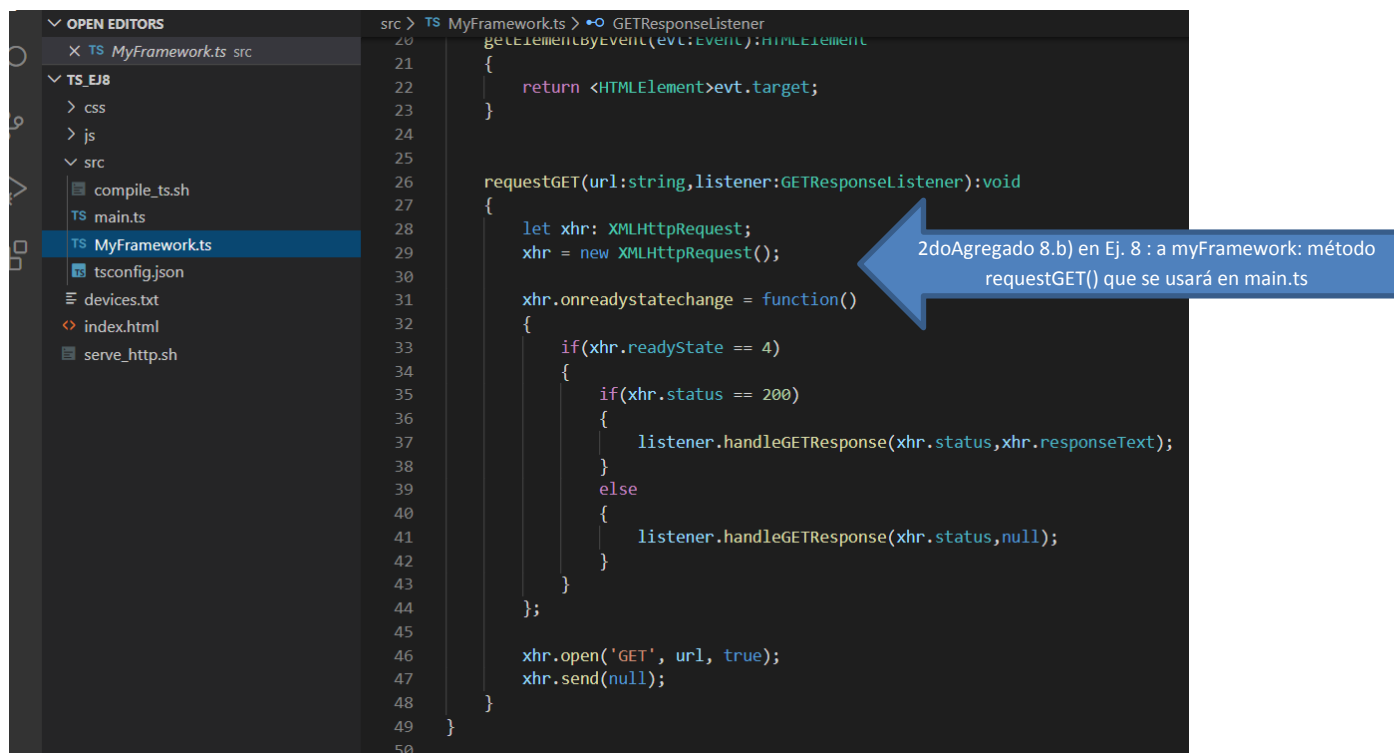
Dentro del método, podremos usar el clásico objeto de JS XMLHttpRequest para realizar el request del tipo GET.

8.b) al final del MyFramework.ts, agregamos el método requestGET() que toma dos argumentos:

→ 1 un string url, que por ahora será devices.txt

→ 2 un objeto del tipo definido en 8a) o sea GETResponseListener{..},

Dentro del método, se usa un objeto de Javascript llamado XMLHttpRequest, del cual se define una instancia llamada Xhr. Si el status es OK ==200, devuelve un texto, sinó un NULL, y deja “Abierto” el GET en esa url.



```
20  GETResponseListener
21  {
22      handleGETResponse(status: number, responseText: string): void
23  }
24
25
26  requestGET(url:string,listener:GETResponseListener):void
27  {
28      let xhr: XMLHttpRequest;
29      xhr = new XMLHttpRequest();
30
31      xhr.onreadystatechange = function()
32      {
33          if(xhr.readyState == 4)
34          {
35              if(xhr.status == 200)
36              {
37                  listener.handleGETResponse(xhr.status,xhr.responseText);
38              }
39              else
40              {
41                  listener.handleGETResponse(xhr.status,null);
42              }
43          }
44      };
45
46      xhr.open('GET', url, true);
47      xhr.send(null);
48  }
49
50
```

8.c) Puntos 4 y 5 del ejercicio, modificar main.ts para usar ese método:

4) Para usar este método al iniciar la aplicación web, agregaremos una llamada al mismo en el método “main”:

```
objMyFramework.requestGET("Devices.txt",this);
```

Como primer argumento pasamos la url (en este caso el nombre del archivo ya que estará hosteado en la raíz del sitio), y como segundo argumento, un objeto que implemente la interfaz “GETResponseListener”, por lo que deberemos hacer que la clase “Main” implemente la interfaz, así podremos pasarle “this” como argumento, y el método que se ejecutará al recibir la respuesta del server, estará en la clase “Main”.

```
class Main implements EventListenerObject, GETResponseListener {
```

5) Al implementar la interfaz, estaremos obligados a agregar el método definido en la interfaz, en la clase Main, por lo que se debe agregar el método “handleGetResponse()”. Imprimir dentro del mismo el texto que llegó desde el server.

En Main.ts,

```
src > TS main.ts > Main
1 class Main implements EventListenerObject, GETResponseListener
2 {
3     myf: MyFramework;
4     counter: number = 0;
5
6     handleEvent(evt: Event): void
7     {
8         this.counter++;
9         let btn: HTMLInputElement = this.myf.getElementByEvent(evt);
10        btn.textContent = "click:" + this.counter;
11    }
12
13    handleGETResponse(status: number, response: string) {
14        console.log("Llego status:" + status + " response:" + response);
15    }
16
17    main(): void
18    {
19        this.myf = new MyFramework();
20
21        let b: HTMLInputElement = this.myf.getElementById("boton");
22        console.log(b);
23        b.textContent = "Haceme click!";
24
25        b.addEventListener("click", this);
26
27        this.myf.requestGET("devices.txt", this);
28    }
29
30 }
31
32
33 window.onload = () => {
34     let obj = new Main();
35     obj.main();
36 };
37
```

Agregado 8.c1) hacemos que la clase Main implemente la interfaz GetResponseListener

4b

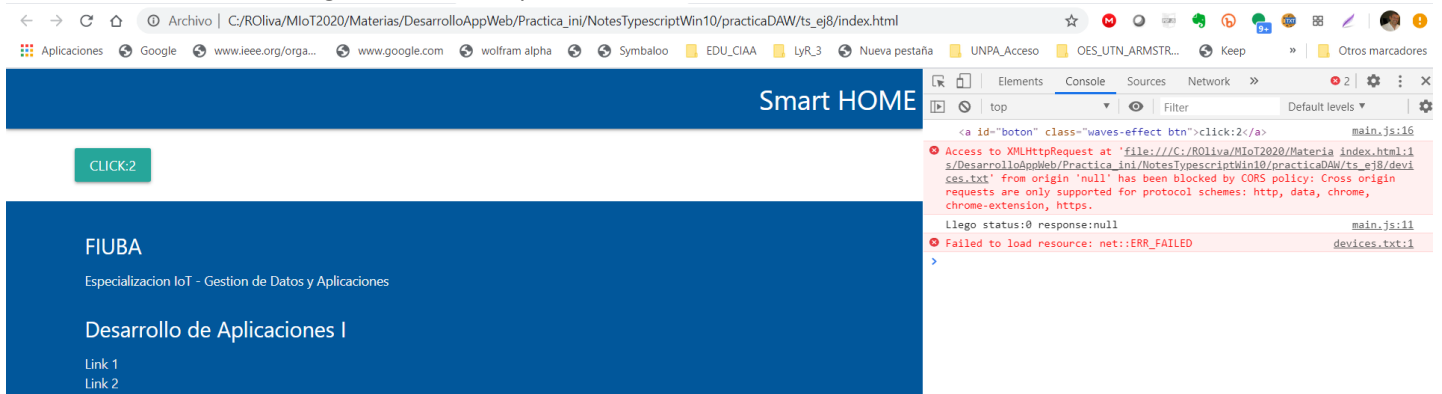
5

Agregado 8.c2) en Ej. 8 : llamado a myf:myFramework método requestGET() Argumentos: i) url = devices.txt ii) "this", porque por 8.c1 implementamos en Main el Get ResponseListener

4a

8.d) Ensayo:

Con el Chrome, viola las reglas de CORS → por eso tira un null



Desde el ejercicio 8) está el script `serve_http.sh` que contiene el container para verlo desde Linux con un servidor Node de Agustín:

```
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
```


8.d.1) Corremos en Probook compilando Ej8 (este es E8_promises.. levemente distinto?)

```
17
18     main():void
19     {
20         this.myf = new MyFramework();
21
22         let buttonElement = this.myf.getButtonOutd(8000);
23     }
24 }
```

OUTPUT DEBUG CONSOLE PROBLEMS TERMINAL 1: bash

```
rafael@rafael-UB18:~/MIoT2020/DAW/DesarrolloAppWeb/delSitio/RepoDAW/daw/ejercicios/Ejercicios_ts/ts_ej8_promises$ ./compile_ts.sh . ./js
bash: ./compile_ts.sh: No such file or directory
rafael@rafael-UB18:~/MIoT2020/DAW/DesarrolloAppWeb/delSitio/RepoDAW/daw/ejercicios/Ejercicios_ts/ts_ej8_promises$ cd src
rafael@rafael-UB18:~/MIoT2020/DAW/DesarrolloAppWeb/delSitio/RepoDAW/daw/ejercicios/Ejercicios_ts/ts_ej8_promises/src$ ./compile_ts.sh . ./js
Init TypeScript compiler {container:ts_compiler, input:/home/rafael/MIoT2020/DAW/DesarrolloAppWeb/delSitio/RepoDAW/daw/ejercicios/Ejercicios_ts/ts_ej8_promises/src,output:/home/rafael/MIoT2020/DAW/DesarrolloAppWeb/delSitio/RepoDAW/daw/ejercicios/Ejercicios_ts/ts_ej8_promises/js}
docker: Error response from daemon: Conflict. The container name "/ts_compiler" is already in use by container "bff813c87b877f33e78caa072c095cc8b7712b0da7b2e1fe885fbb0f740d5" to remove (or rename) that container to be able to reuse that name.
See 'docker run --help'.
rafael@rafael-UB18:~/MIoT2020/DAW/DesarrolloAppWeb/delSitio/RepoDAW/daw/ejercicios/Ejercicios_ts/ts_ej8_promises/src$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED            STATUS              PORTS              NAMES
1fb781b52d9d       abassi/node-http-server  "/bin/sh -c 'http-se..." 20 minutes ago     Up 20 minutes      0.0.0.0:8000->8080/tcp  http-server
bff813c87b87       harmish/typescript    "tsc --project /work..." 24 minutes ago     Up 24 minutes              ts_compiler
rafael@rafael-UB18:~/MIoT2020/DAW/DesarrolloAppWeb/delSitio/RepoDAW/daw/ejercicios/Ejercicios_ts/ts_ej8_promises/src$ docker stop ts_compiler
ts_compiler
rafael@rafael-UB18:~/MIoT2020/DAW/DesarrolloAppWeb/delSitio/RepoDAW/daw/ejercicios/Ejercicios_ts/ts_ej8_promises/src$
```

Compila OK, para correrlo hay que hacer:

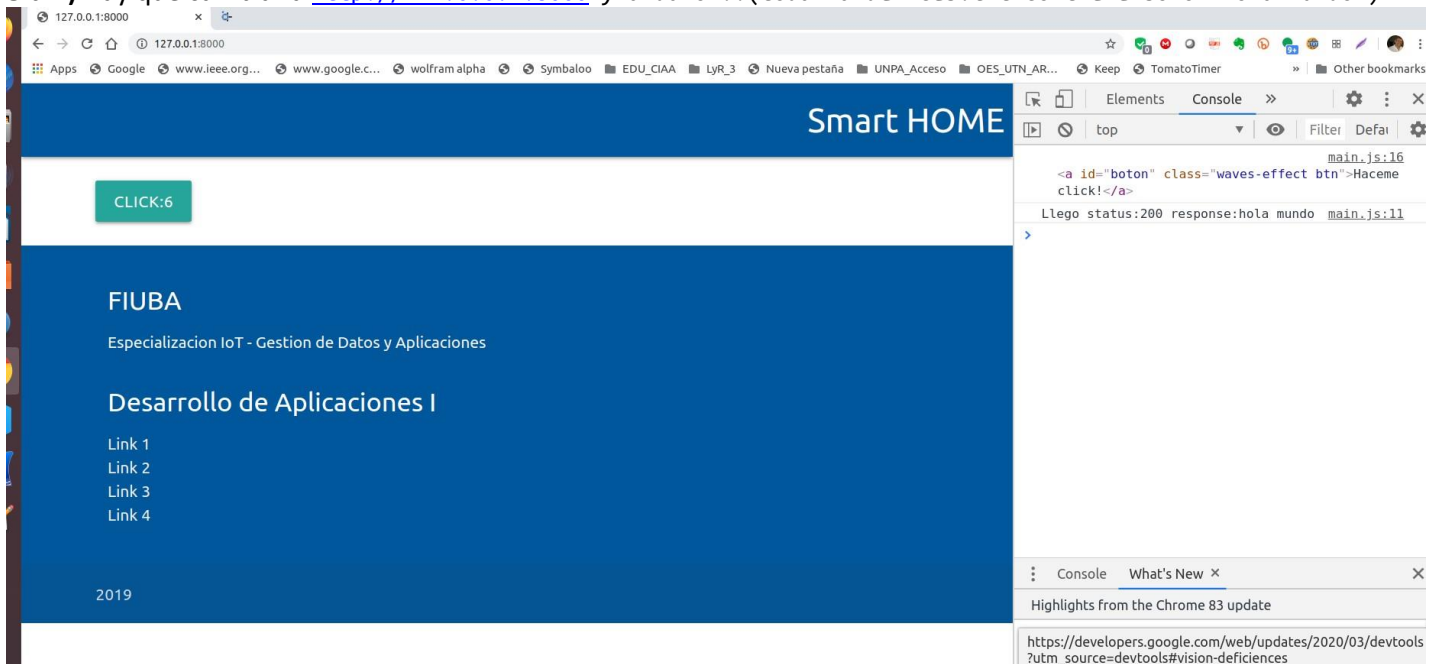
EJ8

```
rafael@rafael-UB18:~/MIoT2020/DAW/DesarrolloAppWeb/delSitio/RepoDAW/daw/ejercicios/Ejercicios_ts/ts_ej8$ ./serve_http.sh . 8000
Starting up http-server, serving /dir_to_serve
Available on:
  http://127.0.0.1:8080
  http://172.17.0.3:8080
(OJO, FUNCIONA CON http://127.0.0.1:8000)
Hit CTRL-C to stop the server
```

```
[Wed Jun 17 2020 11:02:36 GMT+0000 (UTC)] "GET /" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.97 Safari/537.36"
[Wed Jun 17 2020 11:02:37 GMT+0000 (UTC)] "GET /css/materialize.min.css" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.97 Safari/537.36"
[Wed Jun 17 2020 11:02:37 GMT+0000 (UTC)] "GET /js/materialize.min.js" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.97 Safari/537.36"
[Wed Jun 17 2020 11:02:37 GMT+0000 (UTC)] "GET /js/main.js" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.97 Safari/537.36"
[Wed Jun 17 2020 11:02:37 GMT+0000 (UTC)] "GET /js/MyFramework.js" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.97 Safari/537.36"
[Wed Jun 17 2020 11:02:38 GMT+0000 (UTC)] "GET /devices.txt" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.97 Safari/537.36"
[Wed Jun 17 2020 11:02:38 GMT+0000 (UTC)] "GET /favicon.ico" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.97 Safari/537.36"
[Wed Jun 17 2020 11:02:38 GMT+0000 (UTC)] "GET /favicon.ico" Error (404): "Not found"
[Wed Jun 17 2020 11:02:48 GMT+0000 (UTC)] "GET /css/materialize.min.css" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.97 Safari/537.36"
```

HAY QUE HACER docker stop del ts_compiler y del http-server al cargar un ejercicio nuevo, se corren desde VSC en terminales distintos.

8.d.1) Hay que cambiar a <http://127.0.0.1:8000> y anda ok..(todavía devices.txt contiene solo “hola mundo”)



NOTA: Ver clase 5 para ver método configClick() agregados a MyFramework que no esta en sitio <https://drive.google.com/file/d/1l2nINjEEXH5oOHoyYEyD39SbmENngbWr/view>

```
getElementByEvent(e:Event):HTMLElement {
    return <HTMLElement> e.target;
}

configClick(id:string,listener:EventListenerObject):void {
    // buscar el elemento por su id
    let el:HTMLElement = this.getElementById(id);
    el.addEventListener("click",listener);
}

requestGET(url:string, listener: GETResponseListener):void
{
```

Ejercicio 9: Retomamos

Ejercicio 9

1) Cambiar el contenido del archivo “Devices.txt” por un texto con formato JSON. De esta forma se podrán convertir en variables de TS/JS y la información podrá ser representada en el sitio web. El formato del JSON sera:

```
[
  {
    "id":"1",
    "name":"Lampara 1",
    "description":"Luz living",
    "state":"0",
    "type":"0"
  },
  {...}
]
```

2) Definir la interfaz para parsear el texto en formato JSON, llamada “DeviceInt”

3) En el método de la clase Main, “handleGETResponse” parsear el texto JSON y transformarlo en un array de objetos “DeviceInt”:

```
let data: DeviceInt[] = JSON.parse(response);
```

4) Imprimir el objeto data para verificar que los datos sean los cargados en el archivo “Devices.txt”.

9.1: Devices.txt como JSON

```
[
  {
    "id":"1",
    "name":"Lampara 1",
    "description":"Luz living",
    "state":"0",
    "type":"0"
  },
  {
    "id":"2",
```

```

    "name":"Lampara 2",
    "description":"Luz cocina",
    "state":"1",
    "type":"0"
  } ,
  {
    "id":"3",
    "name":"Lampara 3",
    "description":"Luz habitacion",
    "state":"1",
    "type":"0"
  } ,
  {
    "id":"4",
    "name":"Lampara 4",
    "description":"Luz living",
    "state":"1",
    "type":"0"
  } ,
  {
    "id":"5",
    "name":"Persiana habitacion",
    "description":"Frente a la cama",
    "state":"0",
    "type":"1"
  },
  {
    "id":"6",
    "name":"Lampara 5",
    "description":"Luz baño",
    "state":"0",
    "type":"0"
  }
]

```

9.2: Interfaz para parseo de devices.txt, al inicio de main.ts

```

interface DeviceInt{
  id:string;
  name:string;
  description:string;
  state:string;
  type:string;
}

```

9.3.4: Solo agrega un if(status == 200) y el nuevo data que contiene lo parseado

```

handleGETResponse(status:number,response:string){
  console.log("Llego status:"+status+" response:"+response);
  if(status==200)
  {
    let data:DeviceInt[] = JSON.parse(response);
    console.log(data);
  }
}

```

9.5: Verificación desde Ubuntu / http_server de docker con Chrome

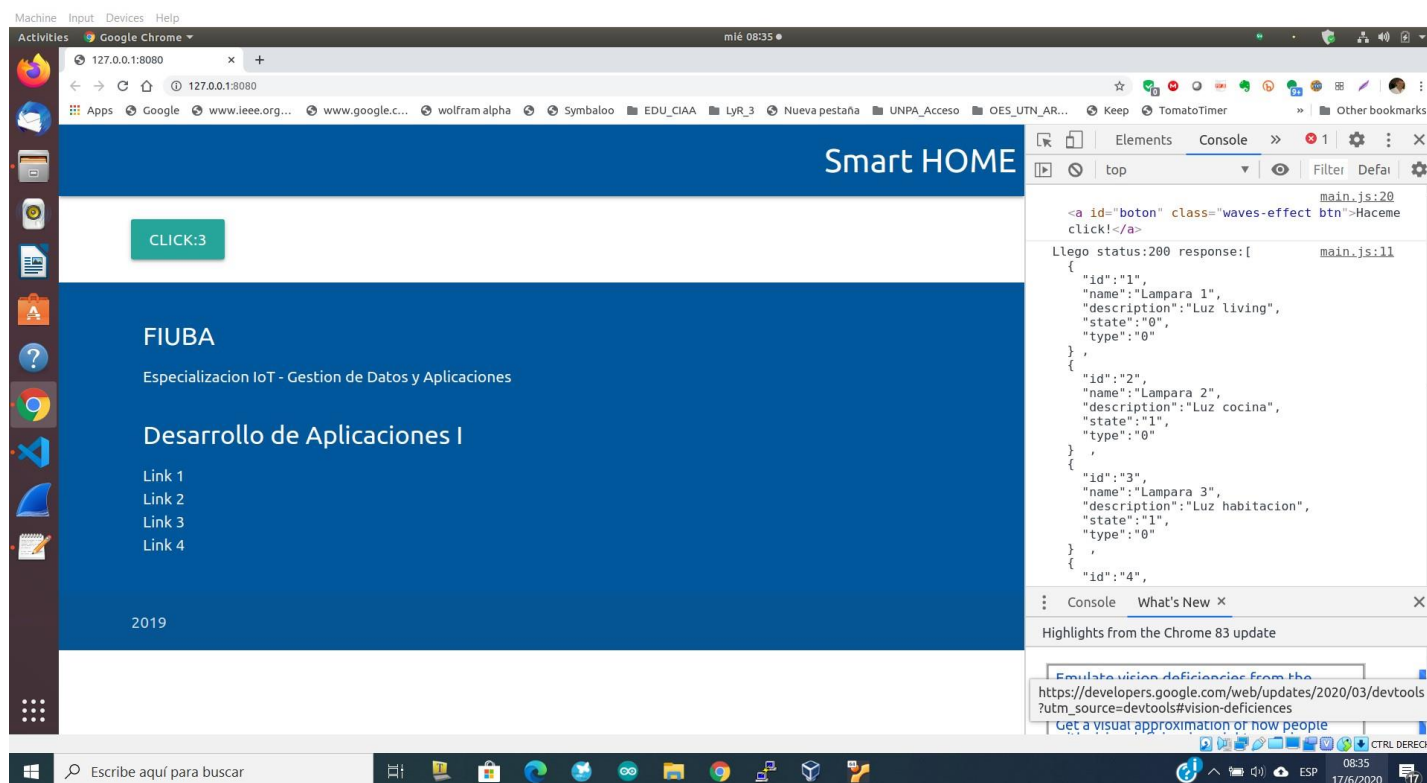
EJ 9:

```
rafael@rafael-UB18:~/MIoT2020/DAW/DesarrolloAppWeb/delSitio/RepoDAW/daw/ejercicios/Ejercicios_ts/ts_ej9$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS                    NAMES
a77d31b2c174   harmish/typescript                  "tsc --project /work..." 5 minutes ago  Up 5 minutes  ts_compiler
1fb781b52d9d   abassi/node-http-server            "/bin/sh -c 'http-se..." 34 minutes ago Up 34 minutes  0.0.0.0:8080->8080/tcp  http-server

rafael@rafael-UB18:~/MIoT2020/DAW/DesarrolloAppWeb/delSitio/RepoDAW/daw/ejercicios/Ejercicios_ts/ts_ej9$ docker stop http-server
http-server
rafael@rafael-UB18:~/MIoT2020/DAW/DesarrolloAppWeb/delSitio/RepoDAW/daw/ejercicios/Ejercicios_ts/ts_ej9$ ./serve_http.sh . 8080
Starting up http-server, serving /dir_to_serve
Available on:
  http://127.0.0.1:8080 (AQUÍ SI FUNCIONA PORQUE PUSIMOS 8080 AL LLAMAR)
  http://172.17.0.3:8080
```

Hit CTRL-C to stop the server

```
[Wed Jun 17 2020 11:34:03 GMT+0000 (UTC)] "GET /" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.97 Safari/537.36"
[Wed Jun 17 2020 11:34:03 GMT+0000 (UTC)] "GET /css/materialize.min.css" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.97 Safari/537.36"
[Wed Jun 17 2020 11:34:03 GMT+0000 (UTC)] "GET /js/materialize.min.js" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.97 Safari/537.36"
[Wed Jun 17 2020 11:34:03 GMT+0000 (UTC)] "GET /js/main.js" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.97 Safari/537.36"
[Wed Jun 17 2020 11:34:03 GMT+0000 (UTC)] "GET /js/myFramework.js" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.97 Safari/537.36"
[Wed Jun 17 2020 11:34:05 GMT+0000 (UTC)] "GET /devices.txt" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.97 Safari/537.36"
[Wed Jun 17 2020 11:34:05 GMT+0000 (UTC)] "GET /favicon.ico" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.97 Safari/537.36"
[Wed Jun 17 2020 11:34:05 GMT+0000 (UTC)] "GET /favicon.ico" Error (404): "Not found"
[Wed Jun 17 2020 11:34:11 GMT+0000 (UTC)] "GET /css/materialize.min.css" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.97 Safari/537.36"
```



Ejercicio 10: Retomamos

Ejercicio 10

En este ejercicio, mostraremos la lista de dispositivos en la página web.

1) Tomar de la maqueta original de “smart house” la lista “ul”, definida en el HTML, y dejarla sin elementos “li” (es decir, sin ítems). Agregarle el id “devicesList”:

```
<ul class='collection' id='devicesList'></ul>
```

2) en Main.ts, crear la clase “ViewMainPage” la cual tendrá el método “showDevices” con la siguiente firma:

```
showDevices(list: DeviceInt[]):void
```

Dentro de este método, que recibe el array de objetos creados desde el texto JSON, se deberán agregar elementos “li” al elemento “ul” que tiene el id “devicesList”.

NOTA1: Para obtener el elemento ul, usar el método “getElementById” de la clase MyFramework.

NOTA2: Para agregar html a un elemento, cargar el texto en el atributo “innerHTML”

3) Ejecutar el método “showDevices” luego de parsear el JSON para que se vean en la página.

10.1: En el index.htm del 4.2, reemplazamos la parte del botón por

```
<div class="container">
  <br/>
  <h4>Devices</h4>

  <ul class="collection" id='devicesList'>
  </ul>

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

10.2: En main.ts, tenemos que crear la clase ViewMainPage, con el método showDevices():

10.2.1: (con switch de type colapsado). Crea un elemento HTML (devicesUl) y un string items.

Luego un índice i para recorrer la lista con un for. Dentro de ese for, crea otro string checkedStr que se pone en "checked" si el state del elemento está en 1.

```
class ViewMainPage
{
    myf:MyFramework;

    constructor(myf:MyFramework)
    {
        this.myf = myf;
    }

    showDevices(list:DeviceInt[]):void
    {
        // cargo la lista de objetos en el DOM
        let devicesUl:HTMLElement = this.myf.getElementById("devicesList");

        let items:string="";
        for(let i in list)
        {
            let checkedStr="";
            if(list[i].state=="1")
                checkedStr="checked";

            switch(list[i].type) ...

        }

        devicesUl.innerHTML=items;
    }
}
```

showDevices es el método requerido:
lleva el parámetro list

Nota1

Switch colapsado, ponemos resumen interno mas adelante, depende del type leído el parámetro list

Nota2

10.2.2: Dentro del switch, se asigna HTML a items según si tipo es 0=lámpara; 1= persiana

```
switch(list[i].type)
{
    case "0": // Lámpara
        items+="- 

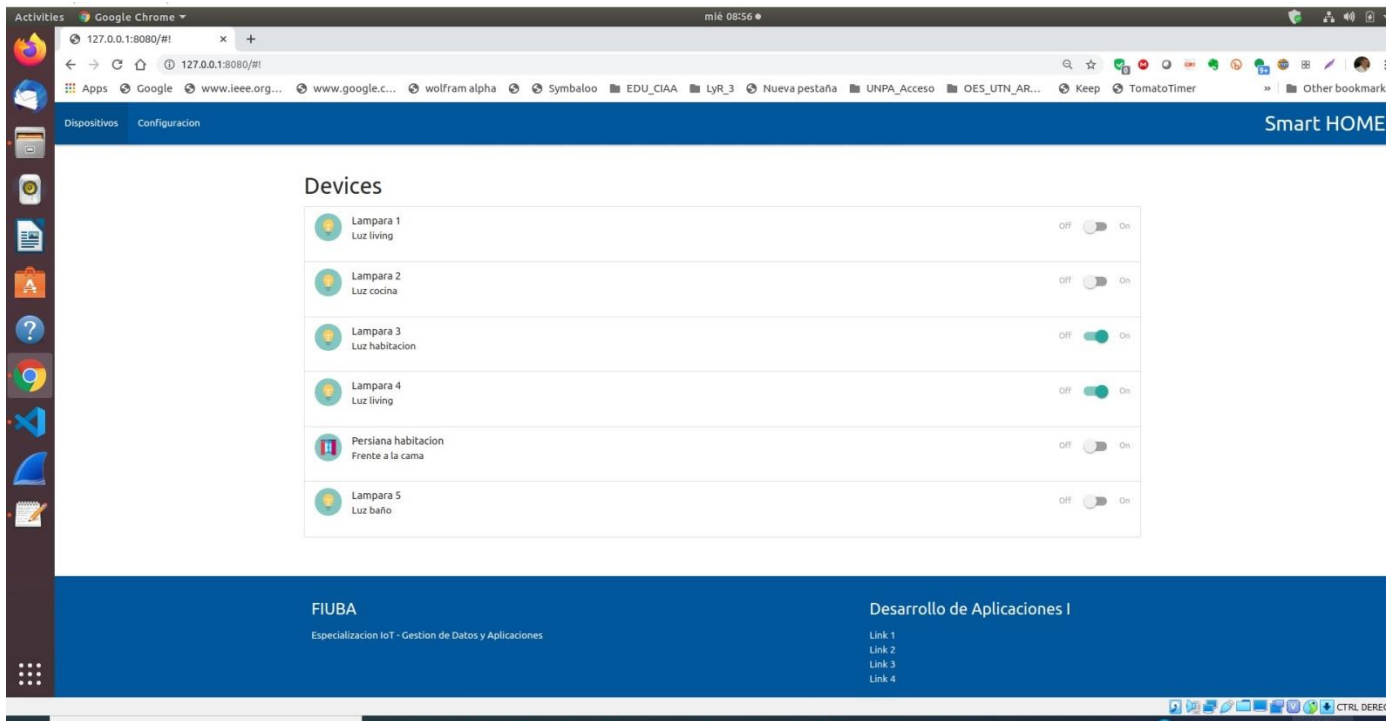
```

10.3 Dentro del class Main, vemos que ahora eliminamos el botón (y su EventListenerObject), y nos concentramos en probar “view” que es una instancia del ViewMainPage(). Mantenemos el mismo requestGET del archivo JSON devices.txt

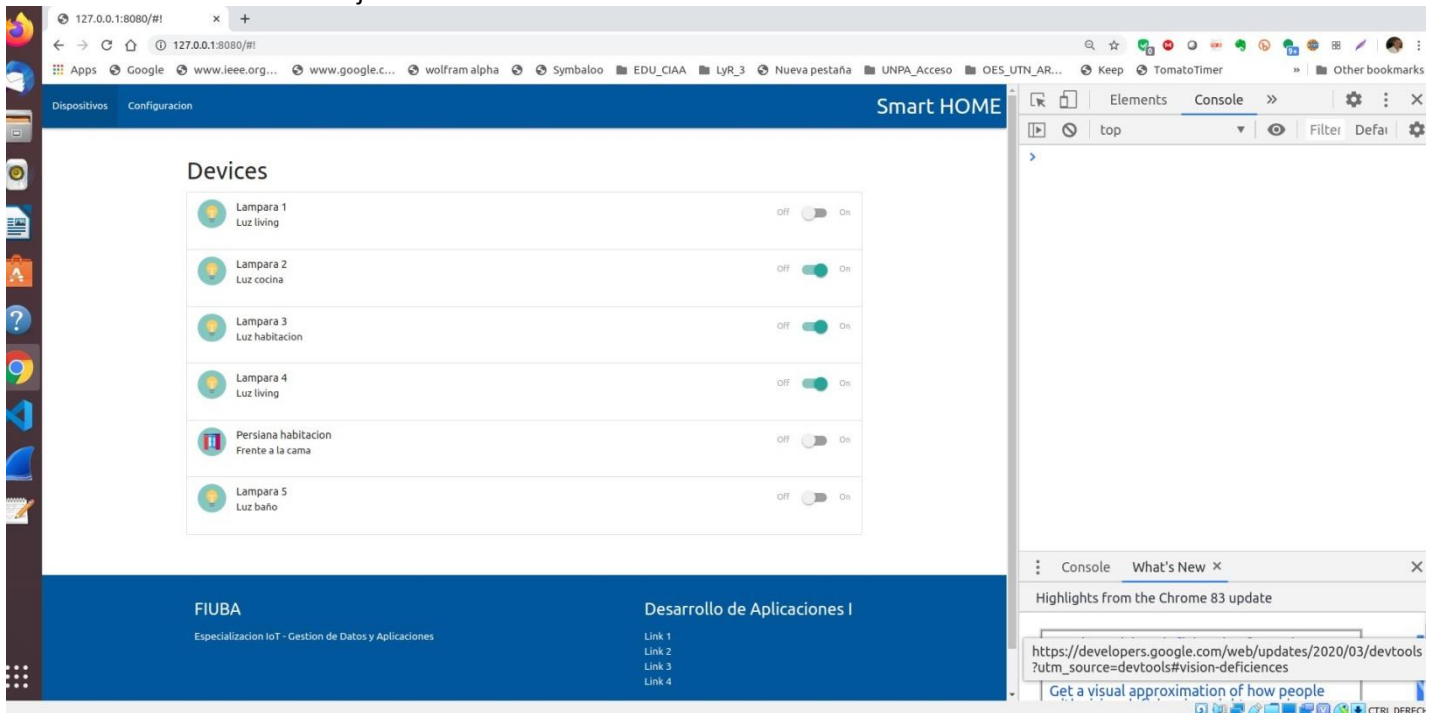
```
9  class Main implements GETResponseListener
10 {
11     myf:MyFramework;
12     view:ViewMainPage;
13
14     handleGETResponse(status:number,response:string){
15         if(status==200)
16         {
17             let data:DeviceInt[] = JSON.parse(response);
18             this.view.showDevices(data);
19         }
20     }
21
22     main():void
23     {
24         this.myf = new MyFramework();
25
26         this.view = new ViewMainPage(this.myf);
27
28         this.myf.requestGET("devices.txt",this);
29     }
30 }
31
32 window.onload = () => {
33     let obj = new Main();
34     obj.main();
35 };
36
```

10.4 Ensayo: (desde UB) – ya no usa consola, es todo hacia el DOM, compilamos en Probook:

```
EJ 10
rafael@rafael-UB18:~/MIoT2020/DAW/DesarrolloAppweb/delSito/RepoDAW/daw/ejercicios/Ejercicios_ts/ts_ej10$ docker stop http-server
http-server
rafael@rafael-UB18:~/MIoT2020/DAW/DesarrolloAppweb/delSito/RepoDAW/daw/ejercicios/Ejercicios_ts/ts_ej10$ ./serve_http.sh . 8080
Starting up http-server, serving /dir_to_serve
Available on:
  http://127.0.0.1:8080
  http://172.17.0.3:8080
Hit CTRL-C to stop the server
[Wed Jun 17 2020 11:54:54 GMT+0000 (UTC)] "GET /" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.97 Safari/537.36"
[Wed Jun 17 2020 11:54:55 GMT+0000 (UTC)] "GET /css/materialize.min.css" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.97 Safari/537.36"
[Wed Jun 17 2020 11:54:55 GMT+0000 (UTC)] "GET /js/materialize.min.js" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.97 Safari/537.36"
[Wed Jun 17 2020 11:54:55 GMT+0000 (UTC)] "GET /js/main.js" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.97 Safari/537.36"
[Wed Jun 17 2020 11:54:55 GMT+0000 (UTC)] "GET /js/MyFramework.js" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.97 Safari/537.36"
[Wed Jun 17 2020 11:54:58 GMT+0000 (UTC)] "GET /devices.txt" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.97 Safari/537.36"
[Wed Jun 17 2020 11:54:58 GMT+0000 (UTC)] "GET /images/lightbulb.png" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.97 Safari/537.36"
[Wed Jun 17 2020 11:54:58 GMT+0000 (UTC)] "GET /images/window.png" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.97 Safari/537.36"
[Wed Jun 17 2020 11:57:00 GMT+0000 (UTC)] "GET /css/materialize.min.css" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.97 Safari/537.36"
[Wed Jun 17 2020 11:57:12 GMT+0000 (UTC)] "GET /" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.97 Safari/537.36"
[Wed Jun 17 2020 11:57:12 GMT+0000 (UTC)] "GET /css/materialize.min.css" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.97 Safari/537.36"
[Wed Jun 17 2020 11:57:12 GMT+0000 (UTC)] "GET /js/materialize.min.js" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.97 Safari/537.36"
[Wed Jun 17 2020 11:57:12 GMT+0000 (UTC)] "GET /js/main.js" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.97 Safari/537.36"
[Wed Jun 17 2020 11:57:12 GMT+0000 (UTC)] "GET /js/MyFramework.js" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.97 Safari/537.36"
[Wed Jun 17 2020 11:57:14 GMT+0000 (UTC)] "GET /devices.txt" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.97 Safari/537.36"
[Wed Jun 17 2020 11:57:14 GMT+0000 (UTC)] "GET /images/lightbulb.png" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.97 Safari/537.36"
[Wed Jun 17 2020 11:57:14 GMT+0000 (UTC)] "GET /images/window.png" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.97 Safari/537.36"
```

No tiene nada en consola el Ej10:



11: Encaramos este, que detecta clicks sobre los switches.

Ejercicio 11

En este ejercicio, detectaremos el click sobre los elementos “switch” de cada device de la lista.

1) Agregar un id a cada switch de la lista (elementos “input” generados por showDevices()) con el formato “dev_”+id, donde “id” es el id del dispositivo, por lo que los ids de los elementos html quedarán : “dev_1”, “dev_2”, etc.

2) En el método “handleGETResponse”, luego de ejecutar “showDevices”, iterar los elementos del array de devices y según el id de cada uno, obtener la referencia del objeto HTMLElement de cada “input”. Luego asignar un listener para el evento “click” para cada uno. (Volver a ver Ejercicio 6)

3) Al producirse un click sobre uno de los switch, imprimir por consola el id del dispositivo sobre el que se hizo click.

11.1: En cada switch agregamos un id: id_1, id_2.. etc

```
class Main implements GETResponseListener, EventListenerObject
{
    myf:MyFramework;
    view:ViewMainPage;

    handleEvent(evt:Event):void
    {
        let sw: HTMLElement = this.myf.getElementByEvent(evt);
        console.log("click en device:"+sw.id);
    }

    handleGETResponse(status:number,response:string){
        if(status==200)
        {
            let data:DeviceInt[] = JSON.parse(response);
            this.view.showDevices(data);

            for(let i in data)
            {
                let sw:HTMLElement = this.myf.getElementById("dev_"+data[i].id);
                sw.addEventListener("click",this);
            }
        }
    }

    main():void
    {
        this.myf = new MyFramework();

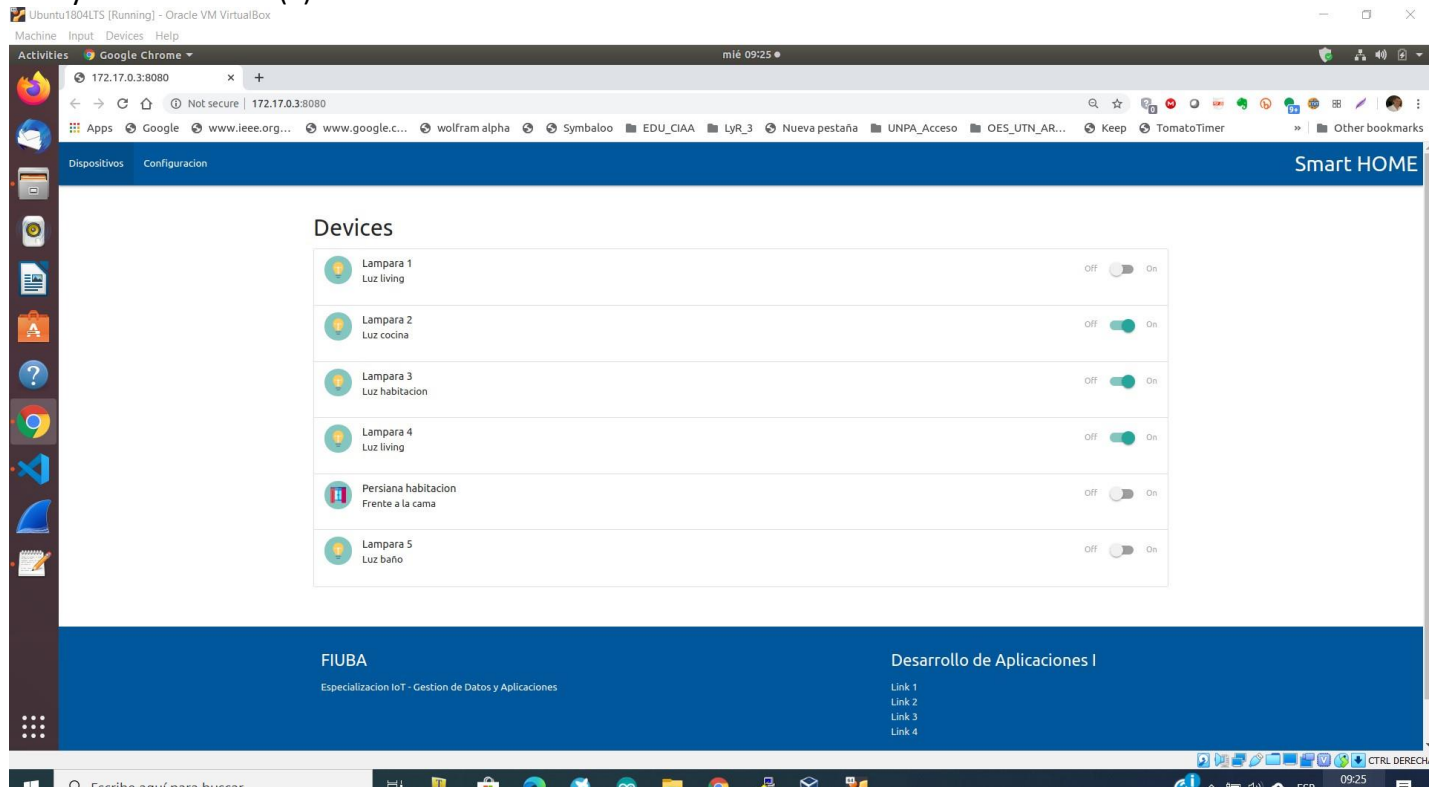
        this.view = new ViewMainPage(this.myf);

        this.myf.requestGET("devices.txt",this);
    }
}
```

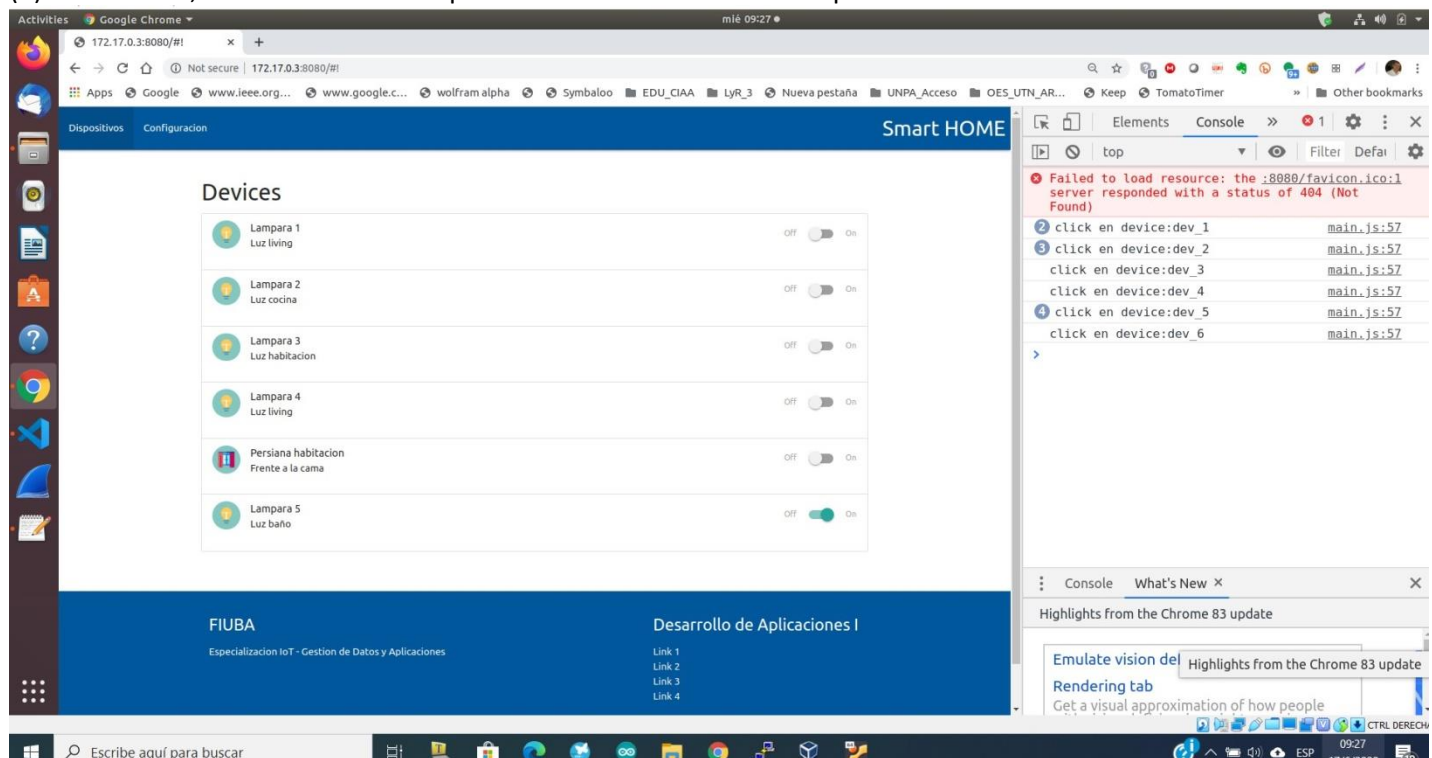


11.2: Ensayo

Ensayos en Probook (a) sin consola



(b) con consola, los numeritos a la izq de cada click indican las veces que se conmutó cada switch:



12: Encaramos este, que tiene varios ítems, integramos por partes:

Ejercicio 12

En este ejercicio se agregará un método para hacer requests “POST” hacia el servidor. Enviando datos asociados.

1) Definiremos una interface que nos permitirá definir el evento que se ejecute al llegar la respuesta del servidor:

```
interface POSTResponseListener{
    handlePOSTResponse(status:number, response:string):void;
}
```

12.1 → esto se agrega al inicio de MyFramework.ts

```
src > TS MyFramework.ts > GETResponseListener
1  interface GETResponseListener{
2      handleGETResponse(status:number,response:string):void;
3  }
4  interface POSTResponseListener{
5      handlePOSTResponse(status:number,response:string):void;
6  }
7
8  class MyFramework{
9
10     /**
11     * getElementById: Busca un elemento del DOM por su ID
```

2) Agregar el método “requestPOST” a la clase MyFramework, el mismo deberá realizar un request del tipo POST hacia el server, enviando un diccionario de datos. El método debe tener la siguiente firma:

```
requestPOST(url:string, data:object, listener:POSTResponseListener):void
```

Este es el ejemplo que se da.. la parte de FormData no está incluida en los ejemplos de la clase..ver?

Ejemplo de request POST:

```
let formData:FormData = new FormData();
for(let key in data) {
    formData.append(key, data[key]);
}

let xhr = new XMLHttpRequest();

xhr.onreadystatechange = function() {
    if(xhr.readyState == 4) {
        if(xhr.status == 200)
            listener.handlePOSTResponse(xhr.status,xhr.responseText);
        else
            listener.handlePOSTResponse(xhr.status,null);
    }
};

xhr.open("POST", url);
// envio JSON en body de request (Usar con NODEJS)
//xhr.setRequestHeader("Content-Type", "application/json;charset=UTF-8");
//xhr.send(JSON.stringify(data));
//
//
// envio Formdata en body de request (Usar con Apache,PythonWS,etc.)
let formData:FormData = new FormData();
for(let key in data) {
    formData.append(key, data[key]);
}
xhr.send(formData)
//
```

12.2 → esto es lo que tenemos en MyFramework.ts

```
requestPOST(url:string,data:object,listener:POSTResponseListener):void
{
    let formData:FormData = new FormData();

    for(let key in data) {
        formData.append(key, data[key]);
    }

    let xhr = new XMLHttpRequest();

    xhr.onreadystatechange = function()
    {
        if(xhr.readyState == 4)
        {
            if(xhr.status == 200)
            {
                listener.handlePOSTResponse(xhr.status,xhr.responseText);
            }
            else
            {
                listener.handlePOSTResponse(xhr.status,null);
            }
        }
    };

    xhr.open("POST", url);
    xhr.send(formData);
}
```

Continuamos con 12.3, 12.4

3) Implementar la interfaz “POSTResponseListener” en la clase Main y escribir el método que define dicha interfaz. (handlePOSTResponse) dentro del mismo imprimir la respuesta del server.

4) Antes de poder enviar un request POST al presionar un switch, deberemos obtener en qué estado está el switch, ya que los datos que deberán enviarse al server son:

- id del dispositivo.
- Nuevo estado del switch.

Para ello, agregar en la clase ViewMainPage el método “getSwitchStateById()” el cual deberá tener la siguiente firma:

```
getSwitchStateById(id:string):boolean
```

NOTA: Castear el elemento input al tipo “HTMLInputElement” y leer el atributo “checked” para saber si está activo o no.

```
4    }
5
6    devicesUl.innerHTML=items;
7  }
8
9  getSwitchStateById(id:string):boolean {
10    let el:HTMLInputElement = <HTMLInputElement>this.myf.getElementById(id);
11    return el.checked;
12  }
13 }
```

12.4 getSwStatebyID, al final
de class ViewMainPage +
NOTA

```

74 class Main implements GETResponseListener, EventListenerObject, POSTResponseListener
75 {
76     myf:MyFramework;
77     view:ViewMainPage;
78
79     handleEvent(evt:Event):void
80     {
81         let sw: HTMLElement = this.myf.getElementByEvent(evt);
82         console.log("click en device:"+sw.id);
83
84         let data:object = {"id":sw.id,"state":this.view.getSwitchStateById(sw.id)};
85         this.myf.requestPOST("cgi-bin/device.py",data,this);
86     }
87
88     handleGETResponse(status:number,response:string):void{
89         if(status==200)
90         {
91             let data:DeviceInt[] = JSON.parse(response);
92             this.view.showDevices(data);
93
94             for(let i in data)
95             {
96                 let sw:HTMLElement = this.myf.getElementById("dev_"+data[i].id);
97                 sw.addEventListener("click",this);
98             }
99         }
100     }
101
102     handlePOSTResponse(status:number,response:string):void{
103         if(status==200)
104         {
105             console.log(response);
106         }
107     }
108
109     main():void
110     {
111         this.myf = new MyFramework();
112
113         this.view = new ViewMainPage(this.myf);
114
115         this.myf.requestGET("devices.txt",this);
116     }
117 }

```

12.3 Post
Listener

Usamos 12.4

Continuamos con 12.5

5) Para poder probar esta funcionalidad, deberemos crear un script que reciba el request y responda algo. Para eso en forma provisoria utilizaremos el container de docker con apache+php.

a) Crear en la raiz del proyecto el archivo **Device.php** con el siguiente contenido:

```
<?php
print_r($_POST);
?>
```

b) Copiar en la raiz del proyecto el script para lanzar el container de Docker “serve_php_app.sh”

c) Lanzar el container para que sirva nuestro sitio en un puerto determinado:

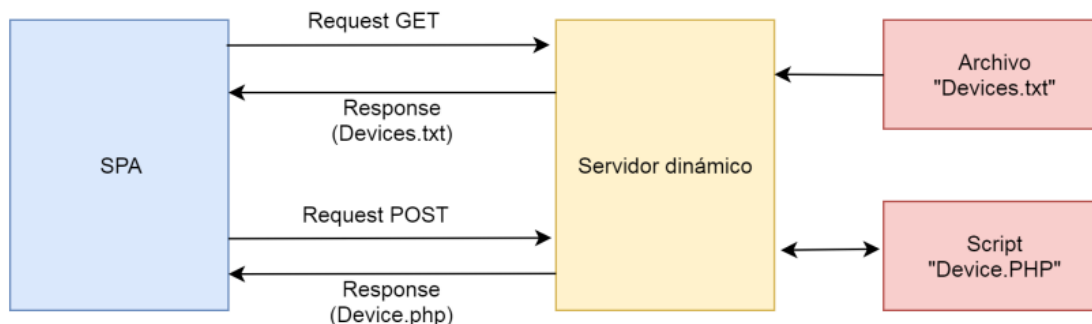
```
./serve_php_app.sh . 8002
```

d) En un navegador, ingresar la URL localhost:8002, deberá aparecer el sitio web.

Al realizar el request POST deberemos ver como respuesta los datos enviados, por ejemplo:

```
dev_1 true
```

El diagrama de comunicación entre el cliente y el server quedaría:



De esta manera se servirán los archivos estáticos como antes, y además los archivos .php se interpretarán y ejecutaran.

(Esto debe hacerse desde Ubuntu 1804)

12.6

6) En el método `handleEvent` (evento de click sobre un switch) en la clase `Main`, ejecutar el método “`getSwitchStateById`” para obtener el estado del switch sobre el que se hizo click, crear el diccionario para enviar por POST con los datos de id y estado, y **realizar el request POST a la URL “Device.php”**, la cual contestará un “eco” de lo que recibió por post.

En el práctico se usa `device.py` en un directorio `/cgi-bin` → hay que mirar el Video.