

21/06/2021

# ToDo API

Desarrollo del proyecto Final

JAVIER ANASTASIO BARRETO MARTINEZ  
FACULTAD DE TELEMATICA

## INTRODUCCIÓN

En este proyecto se desarrollará una pagina web la cual tenga las siguientes características:

- Vista para implementar tareas  
Como su nombre lo dice, en esta se podrá ingresar tareas y mostrarlas en tiempo real. Debe contener lo siguiente:
  - Un formulario para poder ingresar la información con los campos, nombre, descripción, fecha límite, [fata].
- Vista para revisar el progreso de las tareas.
- El uso de websockets para reflejar los cambios que realice un usuario al otro, en este caso si se crea una nueva actividad o se completa una, se debe de actualizar en tiempo real.

## TECNOLOGÍAS

De tecnologías use varias dependencias de npm las cuales son:

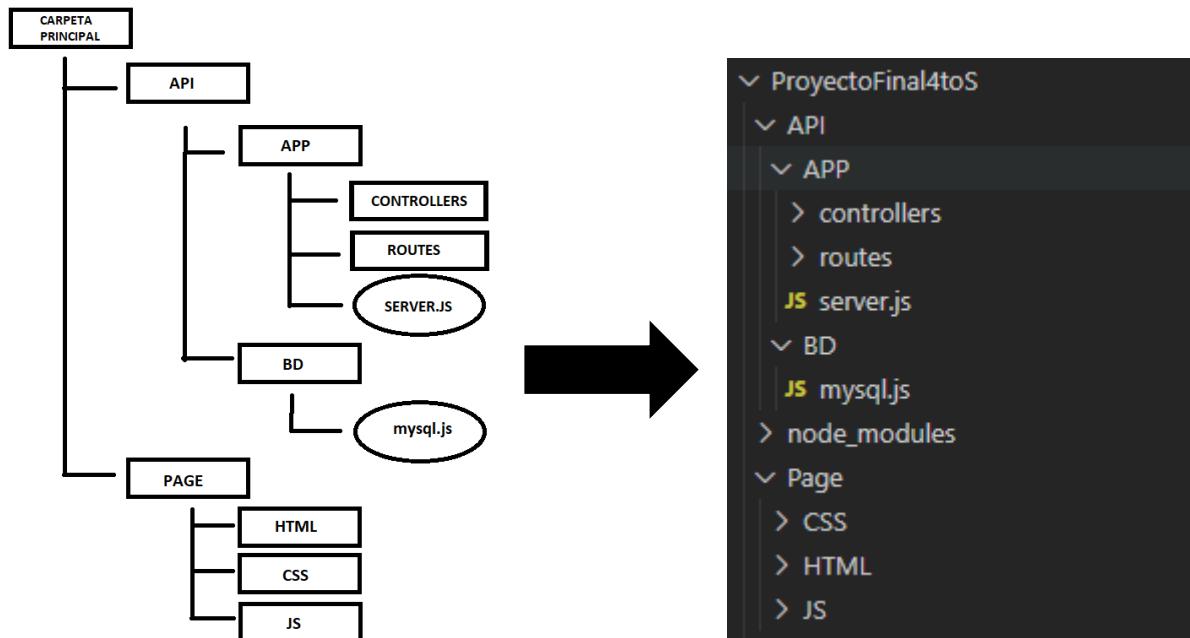
- Cors  
Es una función el cual utiliza cabeceras HTTP para permitir el acceso a recursos seleccionados desde un servidor en un origen distinto (dominios).
- Body-parser  
Permite acceder a los datos, es decir, acceder al cuerpo y analizarlo en un objeto json para poder interactuar con este mismo.
- Express  
Permite el manejo de peticiones diferentes con rutas.
- WS (websockets)  
Permite el uso en vivo de un servidor y que los cambios que hagan los usuarios en una pagina se vean reflejadas en la otra.
- mysql  
Permite crear una o varias conexiones a varias bases de datos.

Tecnologías que instalé para ejecutar/comprobar mi avance en el proyecto.

- Node.js  
Servicio que permite ejecutar códigos de javascript en tiempo real.
- Postman  
Aplicación gratuita para comprobar el uso de rutas y ver como estas responden.

## DESARROLLO

Primero cree la estructura de carpetas en la cual se encontrarán todos los archivos de mi proyecto.



Después instale las dependencias que iba a requerir para la creación de mi proyecto.

```
(1) package.json 1 ×
ProyectoFinal4toS > (1) package.json > ...
1  [
2    "name": "ProyectoFinal4toS",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "test": "echo \\\"Error: no test specified\\\" && exit 1"
8    },
9    "keywords": [],
10   "author": "",
11   "license": "ISC",
12   "dependencies": {
13     "body-parser": "^1.19.0",
14     "cors": "^2.8.5",
15     "express": "^4.17.1",
16     "mysql": "^2.18.1",
17     "ws": "^7.5.0"
18   }
19 ]
20
```

Configuré el archivo **mysql.js** para poder llamar la base de datos llamada dbactividades, y exportando la conexión para poder requerirla más tarde en el servidor.

```
JS mysql.js  X
ProyectoFinal4toS > API > BD > JS mysql.js > ...
1  const mysql = require('mysql');
2
3  const mysqlConnection=mysql.createConnection({
4      host:'localhost',
5      user:'root',
6      password:'@Jbderas12A',
7      database: 'dbactividades'
8  });
9
10 mysqlConnection.connect(function (err){
11     if(err){
12         console.log(err);
13         return;
14     }else{
15         console.log('DB is connected');
16     }
17 });
18
19 module.exports=mysqlConnection;
```

Una vez configurado el archivo **mysql.js**, empecé a configurar el archivo **server.js**, añadiendo las dependencias de cors, express y body-parser para así poder asignar puertos, asignar configuración de la información que se mandara, de la url y al final dándole una ruta principal. Luego, le dije a express en que puerto se ejecutara la app y al final imprimo mensajes para confirmar que el servidor se está ejecutando correctamente.

```
JS server.js  X
ProyectoFinal4toS > API > APP > JS server.js > ...
1  var express = require('express');
2  var app = express();
3  var cors=require('cors');
4  var bodyParser = require('body-parser');
5
6  var port = process.env.PORT || 1332;
7  app.use(cors());
8  app.use(bodyParser.urlencoded({ extended: true }));
9  app.use(bodyParser.json());
10
11 //ruta http://localhost:1332/api
12 var router = require('../routes');
13 app.use('/api', router);
14
15
16 //Info de arranque de servidor
17 app.listen(port);
18 console.log('API escuchando en el puerto ' + port);
```

Una vez creado el servidor, pensé en definir primero el controlador que se usara para las rutas de la API, por lo que creé el archivo **activitiescontroller.js** para que exportara unas funciones que usare más adelante y lo programé de la siguiente manera:

- NuevaA:

Función para agregar una nueva actividad pendiente a la base de datos.

```
var mysql = require('../..//BD/mysql');
//var activity = require('../models/activities');

module.exports = {
  NuevaA:(req,res)=>{// Crear nuevas actividades_pendientes
    mysql.query('INSERT INTO actividades_pendientes(nombre,descripcion,fecha,valor,seleccionada)  Values(?, ?, ?, ?, ?)', [req.body.nombre, req.body.dsc, req.body.fecha, req.body.valor, req.body.seleccionada], (err, rows, fields)=>{
      if(err)
      {
        res.json(err);
      }
      else
      {
        res.json(rows);
      }
    })
  },
};
```

- ListarAP:

Función para obtener las actividades pendientes de la BD.

```
19 |   ListarAP:(req,res)=>{// Obtener las actividades_pendientes
20 |     mysql.query(`SELECT * FROM actividades_pendientes WHERE seleccionada = 'N'`, (err, rows, fields)=>
21 |       if(err)
22 |       {
23 |         res.json(err);
24 |       }
25 |       else
26 |       {
27 |         res.json(rows);
28 |       }
29 |     })
30 |   },
```

- ListarAC:

Función para obtener las actividades completadas de la BD.

```
32 |   ListarAC:(req,res)=>{// Obtener las actividades_completadas
33 |     mysql.query('SELECT * FROM actividades_completadas', (err, rows, fields)=>{
34 |       if(err)
35 |       {
36 |         res.json(err);
37 |       }
38 |       else
39 |       {
40 |         res.json(rows);
41 |       }
42 |     })
43 |   },
```

- ActualizarA:

Se encarga de actualizar una actividad para que le desaparezca de las actividades pendientes.

```

ActualizarA:(req,res)=>{// Actualizar una de las actividades_pendientes
    mysql.beginTransaction((err)=>{
        if(err)
        {
            res.json(err);
        }
        else
        {
            mysql.query('UPDATE actividades_pendientes SET seleccionada = ? WHERE id = ?', [req.body.selec, req.body.id], (err, rows, fields)=>{
                if(err)
                {
                    mysql.rollback(()=>{
                        res.json(err);
                    });
                }
                else
                {
                    mysql.commit(()=>{
                        if(err){
                            mysql.rollback(()=>{
                                res.json(err);
                            });
                        }
                    });
                }
            });
        }
    });
},

```

- Acomp:

Se encarga de insertar una actividad pendiente en la tabla de actividades completadas y elimina esa misma actividad que estaba pendiente.

```

AComp:(req,res)=>{// Crear nuevas actividades_pendientes
    mysql.beginTransaction((err)=>{
        if(err)
        {
            res.json(err);
        }
        else
        {
            mysql.query('INSERT INTO actividades_completadas(nombre,descripcion,fecha,valor,info) Values(?,?,?,?,?)',
            [req.body.nombre, req.body.descripcion, req.body.fecha, req.body.valor, req.body.info], (err, rows, fields)=>{
                if(err)
                {
                    mysql.rollback(()=>{
                        res.json(err);
                    });
                }
                else
                {
                    mysql.query('DELETE FROM actividades_pendientes WHERE id = ?', req.body.id, (err, rows, fields)=>{
                        if(err)
                        {
                            mysql.rollback(()=>{
                                res.json(err);
                            });
                        }
                        else
                        {
                            mysql.commit(()=>{
                                if(err){
                                    mysql.rollback(()=>{
                                        res.json(err);
                                    });
                                }
                            });
                        }
                    });
                }
            });
        }
    });
},

```

Después de eso creé el archivo **actividades.js** en la carpeta de *routes* y en el cual creé las rutas que estarán conectadas a las funciones para interactuar con la base de datos.

```
JS actividades.js ×
ProyectoFinal4toS > API > APP > routes > JS actividades.js > ...
1 var activitycontroller = require('../controllers/activitycontroller');
2 var router = require('express').Router();
3
4 router.get('/actividadesp', function(req, res){
5   //Lista toda las actividades pendientes
6   activitycontroller.ListarAP(req, res);
7 });
8
9 router.get('/actividadesc', function(req, res){
10  //Lista todas las actividades pendientes
11  activitycontroller.ListarAC(req, res);
12 });
13
14 router.post('/', function(req, res){
15  //Agregar nueva actividad pendiente
16  activitycontroller.NuevaA(req,res);
17 });
18
19 router.post('/especial', function(req, res){
20  //Se encarga de insertar la actividad pendiente en la tabla de actividades completadas y borrar esta misma actividad pendiente
21  activitycontroller.Acomp(req,res);
22 });
23
24 router.put('/', function(req, res){
25  //Modifica el estado de una actividad pendiente para que ya no le aparezca a los demás usuarios
26  activitycontroller.ActualizarA(req,res);
27 });
28
29 https://localhost:1332/api/actividades/
30 module.exports = router
```

En la misma carpeta de *routes* creé el archivo de **index.js** en el cual defino cual será la ruta padre en la cual se podrán llamar las otras rutas que interactuaran con la base de datos.

```
JS index.js ×
ProyectoFinal4toS > API > APP > routes > JS index.js > ...
1 var router = require('express').Router();
2
3 var activities = require('../actividades');
4 router.use('/actividades', activities);
5
6 router.get('/', function (req,res){
7   res.status(200).json({message: 'Conectado a la api'});
8 })
9
10 //http://localhost:1332/api
11 module.exports = router;
```

Después probé las rutas con Postman para asegurarme de que las rutas funcionaran correctamente y en caso de detectar fallos, corregirlos.

Una vez creados todos los archivos de la base de datos, empecé a hacer la página web principal **index.js** la cual será de pura decoración para la página ToDo, ya que también cree dos paginas mas, una de estas páginas se llama **nactividades.js** donde estará el formulario para agregar una actividad nueva en tiempo real, y **actividades.js**, la cual permitirá a los usuarios ver las tareas completadas, las tareas pendientes, y ver en tiempo real cual tarea esta disponible, cual tarea esta completada y podrán en esa misma página mediante un popup entregar la actividad pendiente, y a su vez creé el estilo de cada página.

## Index.html



### iBienvenid@ a la Escuela General Juan Benites!

Para agregar una nueva actividad, porfavor dirijase al menu superior y seleccione la opcion de Agregar actividad. Si quiere revisar el progreso de sus actividades, entregarlas o borrarlas, porfavor dirijase al menu superior y seleccione la opcion de Avance de las actividades

```
index.html
ProyectoFinal4toS > Page > HTML > index.html > html > body > div.container > p > strong
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>Escuela General Juan Benites</title>
5          <link rel="stylesheet" href="../CSS/index.css">
6      </head>
7      <body>
8          <header>
9              <div class="headerbox">
10                 <ul>
11                     <a href="./nactividades.html"><li>Agregar actividad</li></a>
12                     <a href="./actividades.html"><li>Avance de las actividades</li></a>
13                 </ul>
14             </div>
15         </header>
16
17         <div class="container">
18             <h1>¡Bienvenid@ a la Escuela General Juan Benites!</h1>
19             <p>Para agregar una nueva actividad, porfavor dirijase al menu superior y seleccione la opcion de <strong>Agregar actividad</strong>. Si quiere revisar el progreso de sus actividades, entregarlas o borrarlas, porfavor dirijase al menu superior y seleccione la opcion de <strong>Avance de las actividades</strong></p>
20
21         </div>
22
23     </body>
24
25 </html>
```

## nactividades.html



A screenshot of a form titled "Agregar nueva actividad". The form contains four input fields: "Nombre" (with placeholder "Nombre:"), "Descripcion" (with placeholder "Descripcion:"), "Fecha de entrega" (with placeholder "dd/mm/aaaa" and a date input field), and "Valor" (with placeholder "Valor:" and a number input field). Below the inputs is a button labeled "Agregar actividad".

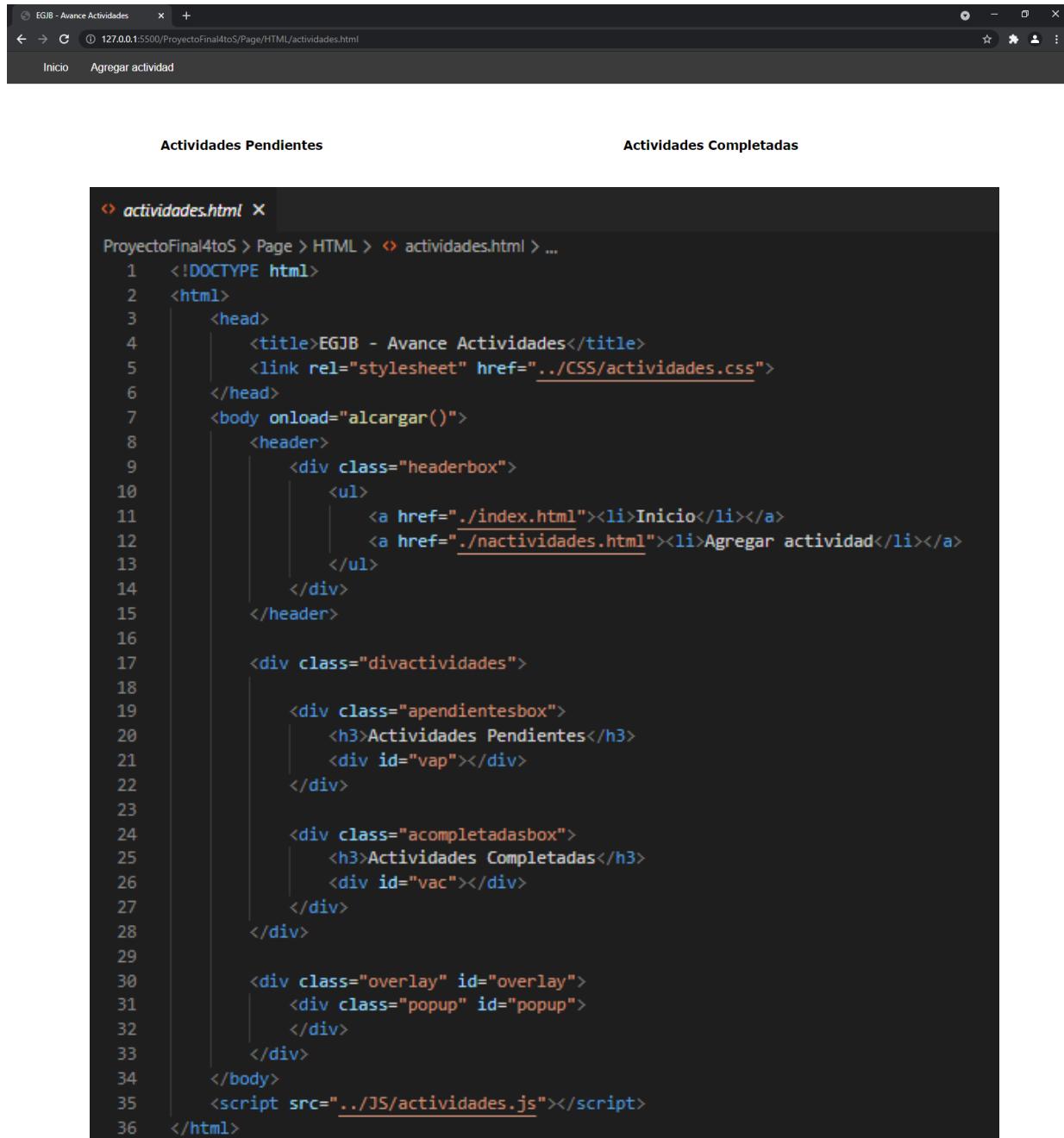
```
<!DOCTYPE html>
<html>
    <head>
        <title>Escuela General Juan Benites</title>
        <link rel="stylesheet" href="../CSS/nactividades.css">
    </head>
    <body>
        <header>
            <div class="headerbox">
                <ul>
                    <a href=".//index.html"><li>Inicio</li></a>
                    <a href=".//nactividades.html"><li>Avance de las actividades</li></a>
                </ul>
            </div>
        </header>
        <div class="divnactividad">
            <h1>Agregar nueva actividad</h1>
            <form>
                <label for="txtname">
                    Nombre:<input type="text" id="txtname">
                    <br>
                </label>

                <label for="txtdsc">
                    Descripcion:<input type="text" id="txtdsc">
                    <br>
                </label>

                <label for="txtdate">
                    Fecha de entrega:<input type="date" id="txtdate">
                    <br>
                </label>

                <label for="txtvalue">
                    Valor:<input type="number" id="txtvalue">
                    <br>
                </label>
                <button type="button" id="AddActivity">Agregar actividad</button>
            </form>
        </div>
        <script src="../JS/nactividades.js"></script>
    </body>
</html>
```

## **actividades.html**



The screenshot shows a browser window with the title "EGJB - Avance Actividades". The address bar displays the URL "127.0.0.1:5500/ProyectoFinal4toS/Page/HTML/actividades.html". Below the address bar, there are two menu items: "Inicio" and "Agregar actividad". The main content area of the browser shows the source code of the "actividades.html" file. The code is presented in a syntax-highlighted format, with line numbers on the left. The code defines an HTML document structure with a header containing navigation links for "Inicio" and "Agregar actividad". It also includes sections for "Actividades Pendientes" and "Actividades Completadas", each with its own heading and a corresponding div element. A script tag at the bottom points to "JS/actividades.js".

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>EGJB - Avance Actividades</title>
5          <link rel="stylesheet" href="../CSS/actividades.css">
6      </head>
7      <body onload="alcargar()">
8          <header>
9              <div class="headerbox">
10                 <ul>
11                     <a href=".//index.html"><li>Inicio</li></a>
12                     <a href=".//nactividades.html"><li>Agregar actividad</li></a>
13                 </ul>
14             </div>
15         </header>
16
17         <div class="divactividades">
18
19             <div class="apendientesbox">
20                 <h3>Actividades Pendientes</h3>
21                 <div id="vap"></div>
22             </div>
23
24             <div class="acompletadasbox">
25                 <h3>Actividades Completadas</h3>
26                 <div id="vac"></div>
27             </div>
28         </div>
29
30         <div class="overlay" id="overlay">
31             <div class="popup" id="popup">
32                 </div>
33             </div>
34         </body>
35         <script src="../JS/actividades.js"></script>
36     </html>
```

Después cree el archivo **nactividades.js** el cual se encargará de que el archivo **nactividades.html** pueda enviar la información a la base de datos.

```
JS nactividades.js X
ProyectoFinal4toS > Page > JS > JS nactividades.js > ...
1  document.getElementById("AddActivity").addEventListener("click", ()=>{
2      let name, dsc, fecha, valor;
3      name=document.getElementById("txtname").value;
4      dsc=document.getElementById("txtdsc").value;
5      fecha=document.getElementById("txtdate").value;
6      valor=document.getElementById("txtvalue").value;
7
8      if(name=="" || dsc=="" || valor=="" || fecha=="")
9      {
10         alert("Porfavor asegurese de haber llenado todos los datos")
11     }
12     else
13     {
14         var data ={nombre:name, dsc:dsc, fecha:fecha, valor:valor, seleccionada:'N'}
15         fetch('http://localhost:1332/api/actividades',{
16             method:'POST',
17             body:JSON.stringify(data),
18             headers:{
19                 'Content-type':'application/json'
20             }
21         })
22         .then(response=>response.json())
23         .catch(error => console.log(error))
24         .then(json=>console.log(json))
25
26         alert("Nueva actividad agregada correctamente.");
27     }
28 });

});
```

Después creé el archivo **actividades.js** el cual se encargará de que el archivo **actividades.html** pueda extraer y colocar la información de las actividades para que los usuarios puedan ver cuales se completaron y las que no se han completado, de misma manera las que no se han completado se podrán completar al momento de darle click a una de las actividades pendientes.

```

JS actividades.js ✘
ProyectoFinal4to > Page > JS > JS actividades.js > ⚡ guitar
  1 function cargar(){
  2   fetch('http://localhost:1332/api/actividades/actividadesp/')
  3     .then(response=>response.json())
  4     .catch(error => console.log(error))
  5     .then(json=>{
  6       let x = document.getElementById("vap");
  7       x.innerHTML="";
  8       test = "";
  9       json.forEach(element => {
 10         x.innerHTML+=
 11           `<button class="actividadesp" id="btnactp" onclick="guitar('Y',${element.id}, '${element.nombre}', '${element.descripcion}', '${element.fecha}', ${element.valor})">${element.nombre}</button>
 12           <br>`;
 13       });
 14     });
 15   }
 16
 17
 18   fetch('http://localhost:1332/api/actividades/actividadesc/')
 19     .then(response=>response.json())
 20     .catch(error => console.log(error))
 21     .then(json=>{
 22       let x = document.getElementById("vac");
 23
 24       x.innerHTML="";
 25       json.forEach(element => {
 26         x.innerHTML+=
 27           `<button class="actividadesc">${element.nombre}</button>
 28           <br>`;
 29       });
 30     });
 31   }
 32 }

33
34
35 function guitar(selec, id, nombre, dsc, fecha, valor){
36   let data = {selec:selec, id:id}
37   let data2 = {id:id, nombre:nombre, dsc:dsc, fecha:fecha, valor:valor}
38   fetch('http://localhost:1332/api/actividades',{
39     method:'PUT',
40     body:JSON.stringify(data),
41     headers:{
42       'Content-type':'application/json'
43     }
44   })
45   .then(response=>response.json())
46   .catch(error => console.log(error))
47   .then(json=>console.log(json));
48
49
50   let popup = document.getElementById("popup");
51
52   popup.innerHTML=
53   <form>
54     <h3>${nombre}</h3>
55     <h4>${dsc}</h4>
56     <div class="contenedor-input">
57       <textarea id="txtinfo" cols="60" rows="10"></textarea>
58     </div>
59     <br>
60     <button type="button" id="btntnentregar" class="btntnentregar" onclick="Entregar(${data2.id}, '${data2.nombre}', '${data2.dsc}', '${data2.fecha}', ${data2.valor})">Entregar</button>
61   </form>
62   ;
63   document.getElementById("overlay").classList.add("active");
64 }
```

```

function Entregar(id, nomb, dsc, fecha, valor){
    let info = document.getElementById("txtinfo").value;
    if(info=="")
    {
        alert("Porfavor, escriba algo en el campo de texto");
    }
    else
    {
        fecha = fecha.split('T',1);
        let data = {id:id, nombre:nomb, descripcion:dsc, fecha:fecha, valor:valor, info:info}
        console.log(data)
        fetch('http://localhost:1332/api/actividades/especial',{
            method:'POST',
            body:JSON.stringify(data),
            headers:{
                'Content-type': 'application/json'
            }
        })
        .then(response=>response.json())
        .catch(error => console.log(error))
        .then(json=>console.log(json));
        document.getElementById("overlay").classList.remove("active");
    }
}

function onlyactpend(){
    fetch('http://localhost:1332/api/actividades/activitiesp/')
    .then(response=>response.json())
    .catch(error => console.log(error))
    .then(json=>{
        let x = document.getElementById("vap");
        x.innerHTML="";
        test = "";
        json.forEach(element => {
            x.innerHTML+=``;
            <br>`;
        });
    });
}

```

Una vez que ya tenia toda la pagina con su funcionamiento con mysql, la probe y corregí errores para que esta funcionara correctamente. Después de que ya corroborara de que la página funcionara correctamente, comencé a añadir websockets por lo que primero agregue los websockets al servidor, dándole un puerto y configurándola para que hacer en caso de recibir mensajes.

```

server.js
ProjectFinal405 > API > APP > # server.js ...
1  var express = require('express');
2  var app = express();
3  var cors=require('cors');
4  var bodyParser = require('body-parser');
5  const ws = require('ws');
6  const wsc = new ws.Server({port:1331});
7
8  var port = process.env.PORT || 1332;
9  app.use(cors());
10 app.use(bodyParser.urlencoded({ extended: true }));
11 app.use(bodyParser.json());
12
13 //ruta http://localhost:1332/api
14 var router = require('./routes');
15 app.use('/api', router);
16
17
18 //Info de arranque de servidor
19 app.listen(port);
20 console.log('API escuchando en el puerto ' + port);
21 console.log('Websocket activo en el puerto 1331');
22
23 //websocket
24 var conexiones = new Array();
25
26 wsc.on('connection', ws=>{
27     conexiones.push(ws);
28
29     ws.on("message", event=>{
30         if(event=="nac"){
31             conexiones.forEach(env =>{
32                 env.send(event);
33             })
34         }
35         else{
36             if(event=="nap"){
37                 conexiones.forEach(env =>{
38                     env.send(event);
39                 })
40             }
41         }
42     })
43 }
44 })
45 })
46 });

```

Después de configurar el servidor para que use también websockets, también configure el archivo de **actividades.js** para que este envíe información al servidor websocket y pueda actualizar la información en tiempo real.

```
JS actividades.js ×
ProyectoFinal4toS > Page > JS > JS actividades.js > Entregar
1 var ws = new WebSocket("ws://127.0.0.1:1331/");
2
3 ws.onmessage = function(e) {
4   console.log(e)
5   if(e.data == "nap")
6   {
7     onlyactpend();
8   }
9   else
10  {
11    if(e.data=="nac")
12    {
13      alcargar();
14    }
15  }
16 };
17
18
function quitar(selec, id, nombre, dsc, fecha, valor){
let data = {selec:selec, id:id}
let data2 = {id:id, nombre:nombre, dsc:dsc, fecha:fecha, valor:valor}
fetch('http://localhost:1332/api/actividades',{
  method:'PUT',
  body:JSON.stringify(data),
  headers:{
    'Content-type':'application/json'
  }
})
.then(response=>response.json())
.catch(error => console.log(error))
.then(json=>console.log(json));
onlyactpend();
ws.send("nap");
}
let popup = document.getElementById("popup");
popup.innerHTML=
<form>
  <h3>${nombre}</h3>
  <h4>${dsc}</h4>
  <div class="contenedor-input">
    <textarea id="txtinfo" cols="60" rows="10"></textarea>
  </div>
  <br>
  <button type="button" id="btntentregar" class="btntentregar" onclick="Entregar(${data2.id}, '${data2.nombre}', '${data2.dsc}')">Entregar</button>
</form>
;
document.getElementById("overlay").classList.add("active");
}

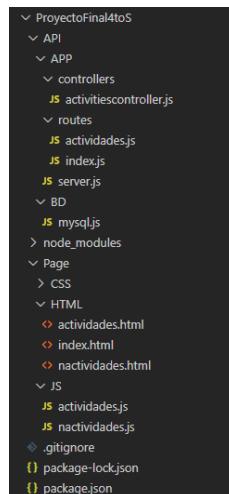
function Entregar(id, nomb, dsc, fecha, valor){
let info = document.getElementById("txtinfo").value;
fecha = fecha.split('T',1);
let data = {id:id, nombre:nomb, descripcion:dsc, fecha:fecha, valor:valor, info:info}
console.log(data)
fetch('http://localhost:1332/api/actividades/especial',{
  method:'POST',
  body:JSON.stringify(data),
  headers:{
    'Content-type':'application/json'
  }
})
.then(response=>response.json())
.catch(error => console.log(error))
.then(json=>console.log(json));
document.getElementById("overlay").classList.remove("active");
ws.send("nac");
}
```

De igual manera, también agregue websockets en el archivo de **nactividades.js**.

```
JS nactividades.js ×
ProyectoFinal4toS > Page > JS > JS nactividades.js > addEventListener("click") callback > headers > 'Content-type'
1 var ws = new WebSocket("ws://127.0.0.1:1331/");
2
3 document.getElementById("AddActivity").addEventListener("click", ()=>{
4     let name, dsc, fecha, valor;
5     name=document.getElementById("txtnname").value;
6     dsc=document.getElementById("txtdsc").value;
7     fecha=document.getElementById("txtdate").value;
8     valor=document.getElementById("txtvalue").value;
9
10    if(name==""||dsc==""||valor==""||fecha=="")
11    {
12        alert("Porfavor asegurese de haber llenado todos los datos")
13    }
14    else
15    {
16        var data ={nombre:name, dsc:dsc, fecha:fecha, valor:valor, seleccionada:'N'}
17        fetch('http://localhost:1332/api/actividades',{
18            method: 'POST',
19            body:JSON.stringify(data),
20            headers:{}
21            |   'Content-type': 'application/json'
22        })
23        .then(response=>response.json())
24        .catch(error => console.log(error))
25        .then(json=>console.log(json))
26
27        ws.send("nap");
28        alert("Nueva actividad agregada correctamente.");
29    }
30 });
31 );
```

Y al hacer eso y probar la pagina ya con todo y websockets, la probé y pude comprobar que todo funcionaba correctamente y que los cambios como agregar una nueva actividad o completar una, si se reflejaban en tiempo real.

Al final, así quedo la carpeta con todos los archivos:



## CONCLUSIONES

En conclusión, gracias a este proyecto pude poner en practica todo lo que vi en este semestre y en semestres anteriores, pude reforzar lo aprendido de mysql con javascript, servidores locales, conexiones a base de datos, creación y funcionamiento de una API y pude aprender a cerca de cómo funcionan los websockets. También gracias a esta practica entiendo como es que se hace la actualización de elementos de un sitio web/aplicacion en tiempo real, como lo seria Messenger y WhatsApp al momento recibir mensajes, o como lo seria classroom que te avisa en tiempo real cuando se sube una actividad.