Creación de proyecto de NodeJs con SocketIo en Visual Studio Code:

Para este primer ejemplo haremos un sitio web cuyo servidor se alojará localmente.

El sitio web tendrá un formulario el cual el cliente completará con sus datos personales para luego ser enviado al servidor.

Una vez recibido el formulario serán mostrados los datos desde el lado del servidor y el mismo le responderá al cliente confirmando la llegada de los datos.

Para iniciar el proyecto crearemos una carpeta en cualquier directorio de nuestra PC. En este caso la misma tendrá el nombre **`Ejemplo_NodeJs_SocketIo`**. Hacemos click derecho sobre la carpeta recientemente creada y seleccionamos Abrir con Code. Si esta opción no aparece al darle click derecho a la carpeta simplemente se debe abrir el **`Visual Studio Code`** y arrastraremos la carpeta hacia la pantalla de Bienvenida del editor.

Una vez parados dentro de la carpeta para iniciar el proyecto debemos abrir una terminal con **`Ctrl+Shift+ñ`** y allí ejecutar el siguiente comando

```
npm init --yes
```

De esta forma habremos generado un archivo con el nombre **`package.json`** el cual contendrá lo siguiente:

```
{
  "name": "Ejemplo_NodeJs_SocketIo",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

Completaremos los atributos `"description"` con una breve descripción del proyecto y `"author"` con nuestro nombre.

Lo siguiente será instalar los módulos que serán necesarios para la ejecución del código.

En nuestro caso vamos a utilizar los siguientes módulos:

`socket.io:` este módulo nos facilita la comunicación entre el cliente y el servidor haciendo uso de websokets.

path: este módulo nos permitirá ubicar los distintos directorios dentro de nuestro proyecto.

express: con este módulo podremos configuraremos las características del propio servidor.

nodemon: este último modulo permitirá hacer cambios en el código de forma más cómoda ya que reiniciará la ejecución del código cada vez que se detecte un y el mismo sea guardado.

Para su instalación a través de la línea de comandos escribimos los siguiente:

```
npm install socket.io path express nodemon
```

Al instalar los módulos se agregara atributo "dependencies" al archivo **package.json**

```
  "dependencies": {
    "express": "^4.17.1",
    "nodemon": "^2.0.12",
    "path": "^0.12.7",
    "socket.io": "^4.1.3"
  }
```

De esta forma cada vez que compilaremos el proyecto el mismo sabrá cuales serán las dependencias necesarias para la ejecución. Las dependencias son agregadas con la versión más reciente encontrada al momento de instalarlas por lo cual si se quisiera posteriormente tener la versión más reciente se debe cambiar el valor de la versión por la palabra "latest". De esta forma cada vez que queramos actualizar las dependencias solo bastara con ejecutar el comando:

```
npm install
```

Lo siguiente será crear un archivo JavaScript en donde comenzaremos a escribir nuestro código. Dentro de la carpeta del proyecto creamos un archivo con el nombre **index.js**.

Una vez creado dicho archivo editaremos el atributo "scripts" en el **package.json**

```
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  }
```

Borraremos el comando "test" y lo completaremos con lo siguiente:

```
  "scripts": {
    "start": "node index.js",
    "dev": "nodemon index.js"
  }
```

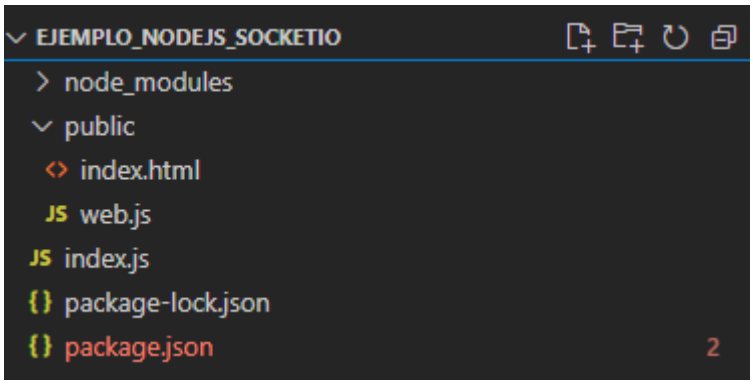
De esta forma al ejecutar el comando **npm start** ejecutaremos el código de forma normal mientras que al usar **npm run deb** lo haremos en forma desarrollador haciendo uso del módulo **nodemon**.

Con todos los cambios realizados hasta el momento el archivo **`package.json`** debería lucir como la imagen de abajo. Una vez realizados todos los cambios guardamos **`Ctrl+S`**.

```
{
  "name": "Ejemplo_NodeJs_SocketIo",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "start": "node index.js",
    "dev": "nodemon index.js"
  },
  "keywords": [],
  "author": "Jhon Doe",
  "license": "ISC",
  "dependencies": {
    "express": "^4.17.1",
    "nodemon": "^2.0.12",
    "path": "^0.12.7",
    "socket.io": "^4.1.3"
  }
}
```

Con esto listo lo que haremos es crear una carpeta dentro del directorio del proyecto que se llame **`public.`** Dentro de ella crearemos dos archivos, un código en `html` y otro en `JavaScript`. Esta carpeta será enviada al cliente cuando se comunica por primera vez con el servidor y contendrá la propia página web. Los archivos que crearemos se llamaran `index.html` y `web.js`.

La estructura de nuestro proyecto deberá contener los siguientes archivos y carpetas:



Tener en cuenta que la carpeta **`node_modules`** y el archivo **`package-lock.json`** son añadidos al proyecto automáticamente cuando instalamos las dependencias del proyecto

Comenzaremos ahora a escribir el código del servidor en el archivo **`index.js`**.

Primero requeriremos los módulos instalados de la siguiente forma

```
const path = require('path');
const express = require('express');
const SocketIo = require('socket.io');
```

Una vez requeridos los módulos configuraremos el servidos de la siguiente forma

```
const app = express();
app.set('port',process.env.PORT || 3000);
app.use(express.static(path.join(__dirname, 'public')))
```

Con esta porción de código lo que hacemos es elegir un puerto para atender las consultas que llegaran al servidor el cual será el puerto 3000 si no es que hay un puerto establecido por el sistema operativo.

En la tercera línea lo que le indicamos al servidor es en donde buscar la carpeta con la página web para enviarla al cliente al establecer la conexión.

Ya teniendo todo configurados iniciamos el servidor e imprimimos una notificación por consola

```
const server = app.listen(app.get('port'), () => {
    console.log('Server on port', app.get('port'));
})
```

Por ultimo escribimos el código correspondiente al manejo de los webSockets

```
// manejo de sockets
const io = SocketIo(server);

io.sockets.on('connection', (Socket) =>{
    console.log("New Connection");

    Socket.on('envioDatos', (datos) =>{
        console.log("Nueva inscripcion: ", datos);

    Socket.emit("datosRecibidos");
    });
});
```

Esta porción de código lo que hace es atender al cliente al conectarse con el servidor. Dentro del webSocket que se crea al conectarse haremos 3 cosas: imprimiremos un mensaje por consola el cartel `New Connection`, escucharemos el evento `envioDatos` y emitiremos un evento al cliente llamado `datosRecibidos`.

Solo resta escribir el código que ejecutara el cliente para escuchar y enviar los eventos con el servidor (no analizaremos el archivo index.htm pero se encuentra adjunto con con este documento). En el archivo **`index.js`** escribiremos el siguiente código:

```
const socket = io();

const form = document.querySelector('#formulario');
form.addEventListener('submit', e => {
socket.emit('envioDatos', datos);

// Todo el código que va en esta parte extrae los datos enviados
      por el usuario y los almacena en una objeto llamado datos

});

socket.on('datosRecibidos', () => {
    alert("Inscripción completada");
});
```

En este código asociamos el formulario con la constante `form`. Luego lo que hacemos es escuchar el evento `'submit'`(es decir cuando el formulario ha sido enviado) y enviamos al servidor un evento con el nombre `'envioDatos'` el cual será atendido por este último.

Por ultimo indicamos que al recibir un evento con el nombre `'datosRecibidos'` notificaremos al usuario que el servidor ha recibido nuestro formulario.

Ya teniendo nuestro proyecto listo solo queda probarlo por lo cual ejecutaremos en consola el comando

```
npm start
```

En el navegador ingresamos la siguiente url: http://localhost:3000/

Completamos con datos el formulario y le damos a Enviar Inscripción. Luego de enviar el formulario el servidor nos responderá con el siguiente aviso:

Por ultimo en la consola del Visual Studio Code podremos verificar que efectivamente llegaron los datos enviados por el cliente.

```
PS C:\Users\Justo\Desktop\Ejemplo_NodeJs_SocketIo> npm start

> Ejemplo_NodeJs_SocketIo@1.0.0 start C:\Users\Justo\Desktop\Ejemplo_NodeJs_SocketIo
> node index.js

Server on port 3000
New Connection
Nueva inscripcion:  {
  correo: 'jhon.doe@gmail.com',
  nombre: 'Doe Jhon',
  nalumno: '631863',
  materiasAdeudadas: [ 'materiaA', 'materiaB' ],
  comentarios: 'Solicito la inscripción a la cursada.'
}
```