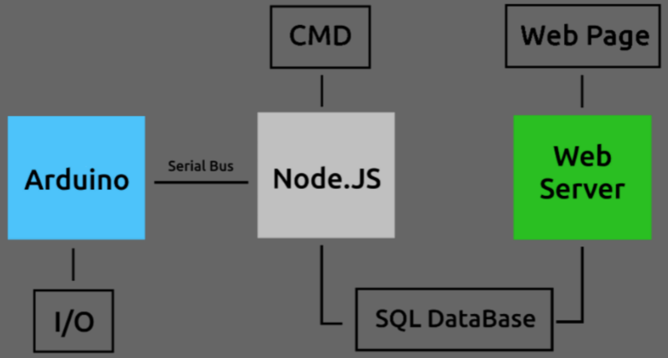
Introduction to NodeJS with WebSockets and DB Connection

Professor:
Bladimir Bacca Cortes Ph.D.
Bladimir.bacca@correounivalle.edu.co

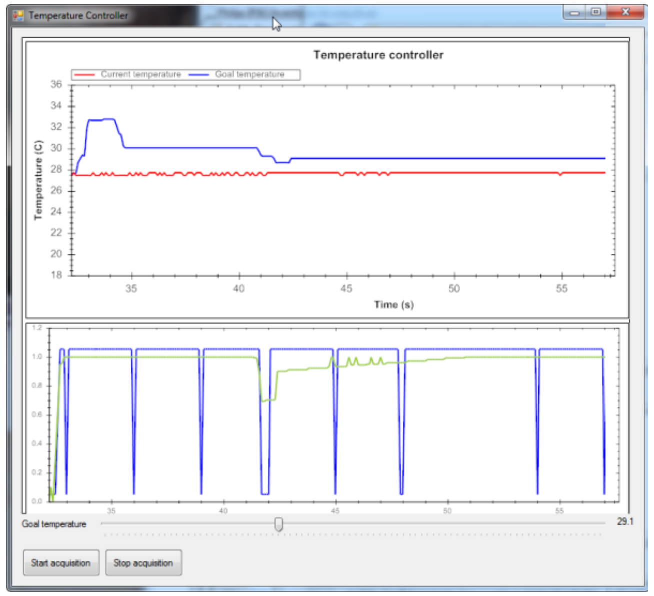
Grupo de Investigación en Percepción y Sistemas Inteligentes



Contents

- NodeJS and WebSockets
  - WebSockets
  - Connecting with serial ports.

- NodeJS and MySQL databases.
  - Connection
  - Handling inserts
  - Handling selects
  - Handling updates



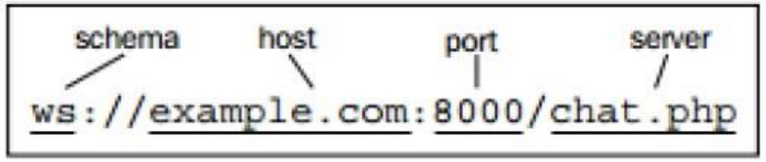
NodeJS and WebSockets

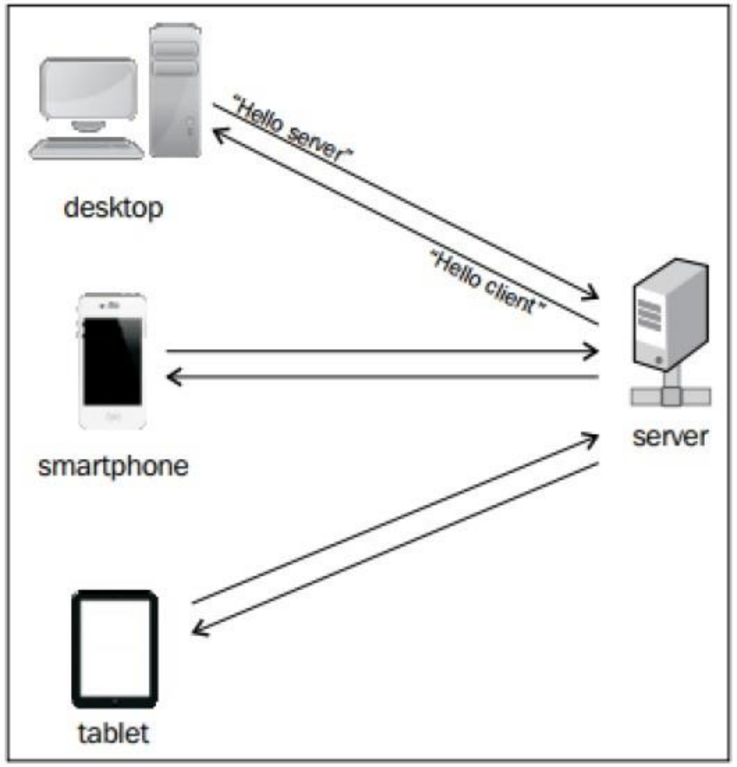
- **WebSockets**: They are defined as a two-way communication between the servers and the clients, which mean both the parties communicate and exchange data at the same time.

- **WebSockets Protocol**:

- It is standard, it means real time communication between web servers and clients is possible
- The only requirement on the browser-side is to run a JavaScript library that can interpret the Web Socket handshake
- Web Socket is an independent TCP-based protocol, but it is designed to support any other protocol that would traditionally run only on top of a pure TCP connection
- Browser support RFC-6455

- **URL**:



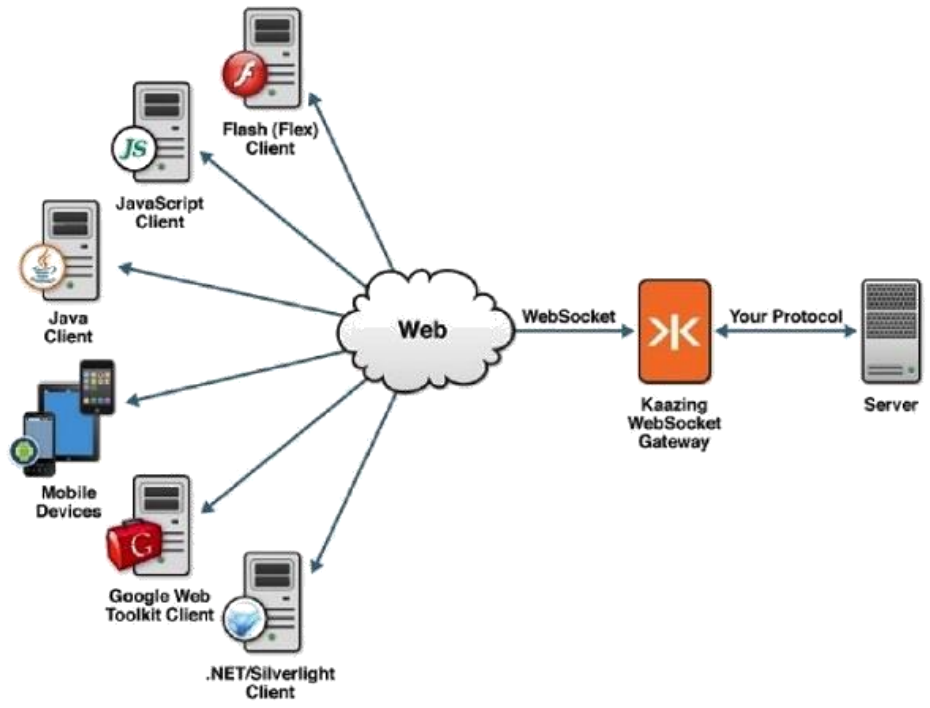


NodeJS and WebSockets – Functionalities

- Web Socket connections are initiated via HTTP; HTTP servers typically interpret Web Socket handshakes as an Upgrade request.
- Web Sockets can both be a complementary add-on to an existing HTTP environment and can provide the required infrastructure to add web functionality

- **Process**:

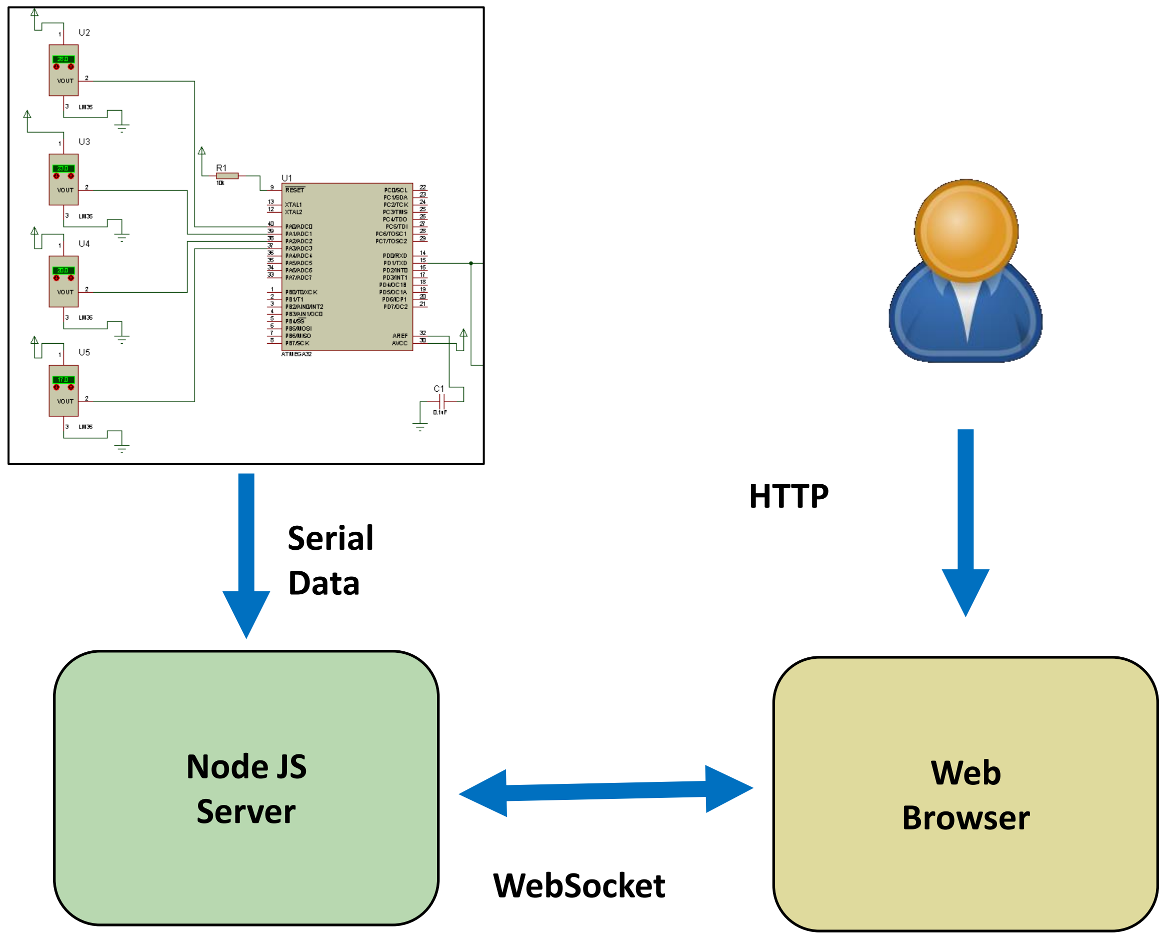
- The client establishes a connection through a process known as Web Socket handshake.
- The process begins with the client sending a regular HTTP request to the server.
- An Upgrade header is requested. In this request, it informs the server that request is for Web Socket connection.
- Web Socket URLs use the **ws** scheme.
- To install it in NodeJS: *npm install ws*



NodeJS and WebSockets

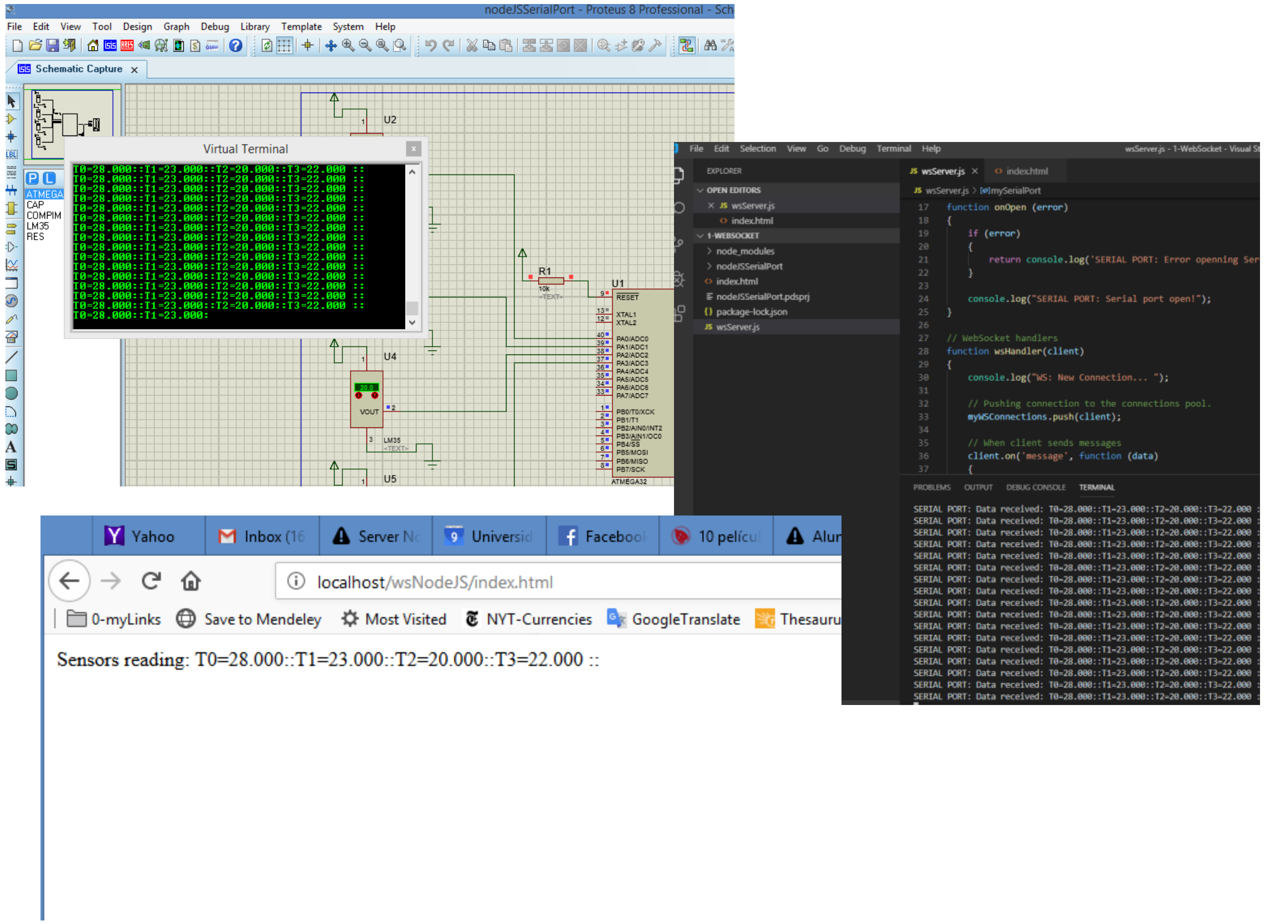
- **Procedure**:

- Start Apache using Xampp Control.
- Using a file explorer, to create a folder called **wsNodeJS** into the **htdocs**.
- Copy the **index.html** file into the **wsNodeJS** folder.
- Start Proteus with the simulation **nodeJSSerialPort.pdsprj**.
- Start Visual Code Studio using the folder **1-WebSocket**.
- Start a browser.
- Run the simulation **nodeJSSerialPort.pdsprj**.
- Run the file **wsServer.js**.
- In the URL, put **http://localhost/wsNodeJS/index.html**.



NodeJS and WebSockets

- **Expected result:**



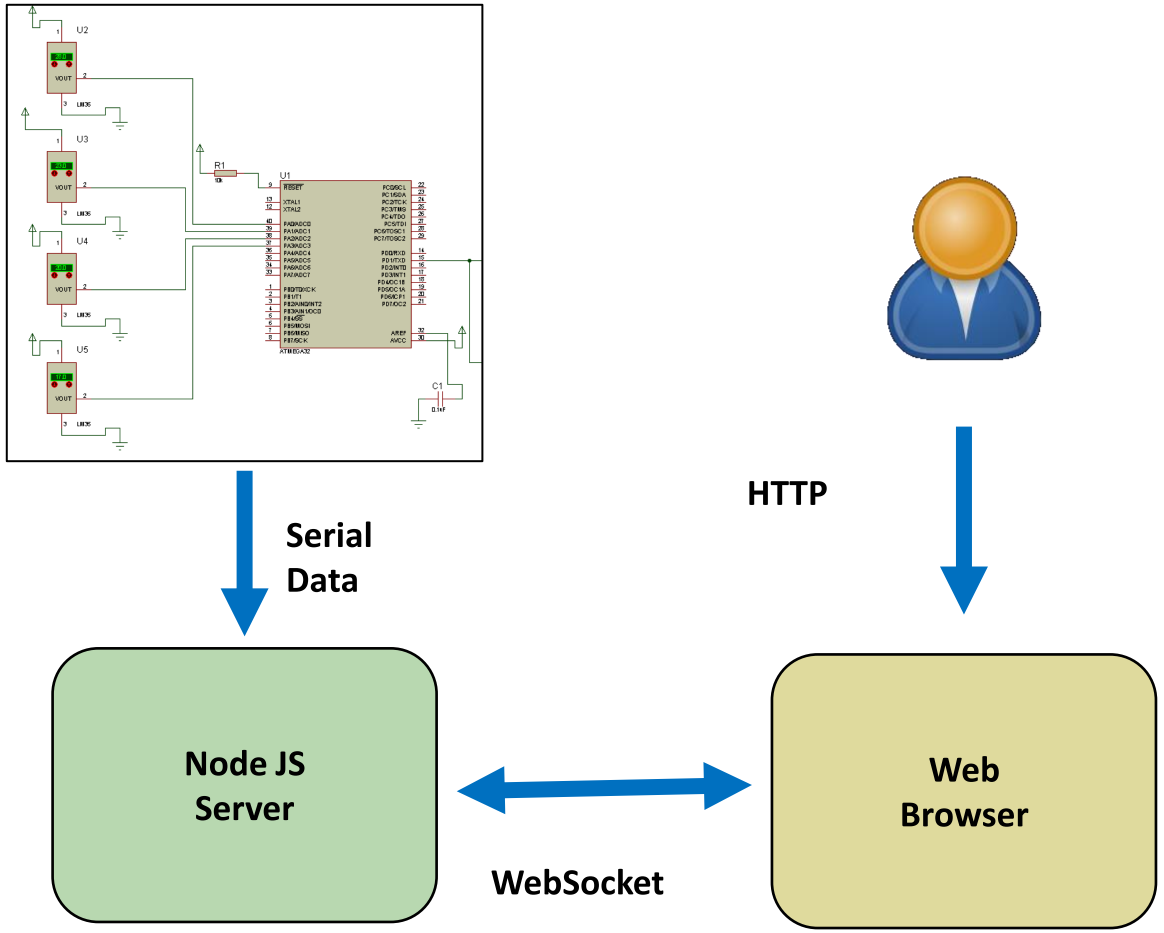
```
T0=28.000::T1=23.000::T2=20.000::T3=22.000 ::
T0=28.000::T1=23.000::T2=20.000::T3=22.000 ::
T0=28.000::T1=23.000::T2=20.000::T3=22.000 ::
T0=28.000::T1=23.000::T2=20.000::T3=22.000 ::
T0=28.000::T1=23.000::T2=20.000::T3=22.000 ::
T0=28.000::T1=23.000::T2=20.000::T3=22.000 ::
T0=28.000::T1=23.000::T2=20.000::T3=22.000 ::
T0=28.000::T1=23.000::T2=20.000::T3=22.000 ::
T0=28.000::T1=23.000::T2=20.000::T3=22.000 ::
T0=28.000::T1=23.000::T2=20.000::T3=22.000 ::
T0=28.000::T1=23.000::T2=20.000::T3=22.000 ::
T0=28.000::T1=23.000::T2=20.000::T3=22.000 ::
T0=28.000::T1=23.000::T2=20.000::T3=22.000 ::
T0=28.000::T1=23.000::T2=20.000::T3=22.000 ::
T0=28.000::T1=23.000::T2=20.000::T3=22.000 ::
T0=28.000::T1=23.000:
```

```
JS wsServer.js > [e]mySerialPort
function onOpen (error)
{
    if (error)
    {
        return console.log('SERIAL PORT: Error openning Ser
    }

    console.log("SERIAL PORT: Serial port open!");
}

// WebSocket handlers
function wsHandler(client)
{
    console.log("WS: New Connection... ");

    // Pushing connection to the connections pool.
    myWSConnections.push(client);

    // When client sends messages
    client.on('message', function (data)
    {
```

```
SERIAL PORT: Data received: T0=28.000::T1=23.000::T2=20.000::T3=22.000 :
SERIAL PORT: Data received: T0=28.000::T1=23.000::T2=20.000::T3=22.000 :
SERIAL PORT: Data received: T0=28.000::T1=23.000::T2=20.000::T3=22.000 :
SERIAL PORT: Data received: T0=28.000::T1=23.000::T2=20.000::T3=22.000 :
SERIAL PORT: Data received: T0=28.000::T1=23.000::T2=20.000::T3=22.000 :
SERIAL PORT: Data received: T0=28.000::T1=23.000::T2=20.000::T3=22.000 :
SERIAL PORT: Data received: T0=28.000::T1=23.000::T2=20.000::T3=22.000 :
SERIAL PORT: Data received: T0=28.000::T1=23.000::T2=20.000::T3=22.000 :
SERIAL PORT: Data received: T0=28.000::T1=23.000::T2=20.000::T3=22.000 :
SERIAL PORT: Data received: T0=28.000::T1=23.000::T2=20.000::T3=22.000 :
SERIAL PORT: Data received: T0=28.000::T1=23.000::T2=20.000::T3=22.000 :
SERIAL PORT: Data received: T0=28.000::T1=23.000::T2=20.000::T3=22.000 :
SERIAL PORT: Data received: T0=28.000::T1=23.000::T2=20.000::T3=22.000 :
SERIAL PORT: Data received: T0=28.000::T1=23.000::T2=20.000::T3=22.000 :
SERIAL PORT: Data received: T0=28.000::T1=23.000::T2=20.000::T3=22.000 :
SERIAL PORT: Data received: T0=28.000::T1=23.000::T2=20.000::T3=22.000 :
SERIAL PORT: Data received: T0=28.000::T1=23.000::T2=20.000::T3=22.000 :
```

localhost/wsNodeJS/index.html

Sensors reading: T0=28.000::T1=23.000::T2=20.000::T3=22.000 ::

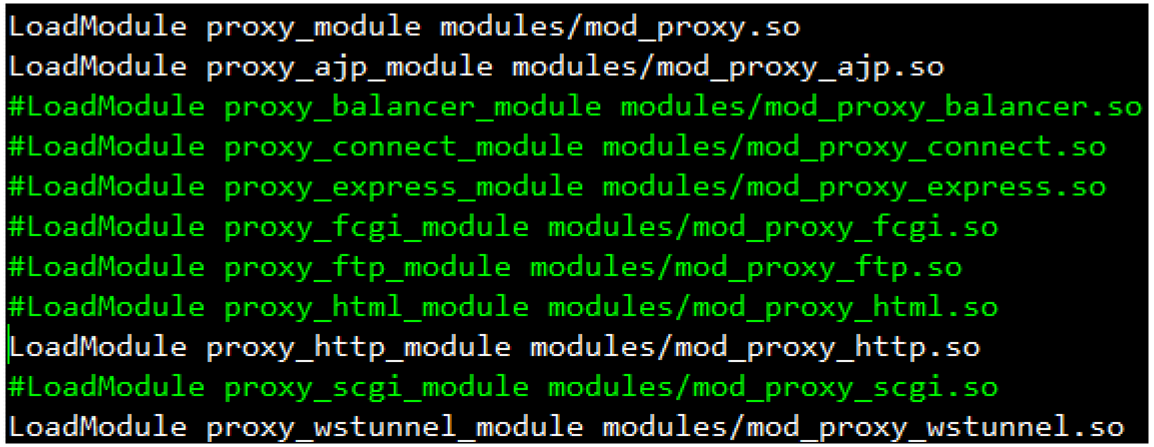


NodeJS and WebSockets

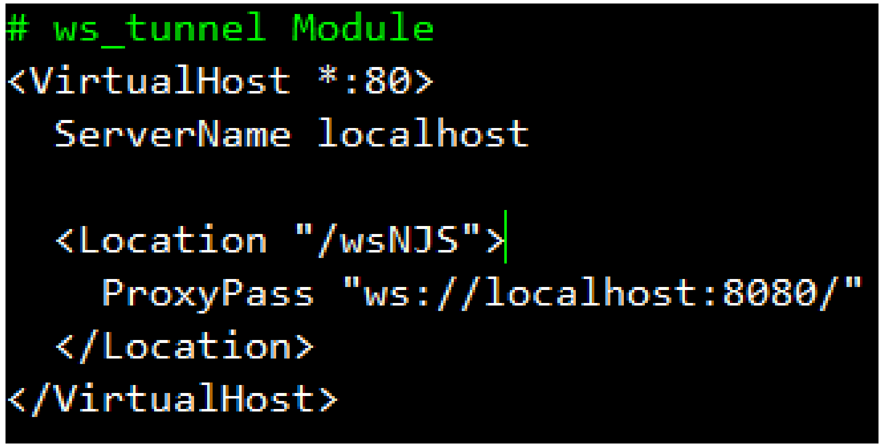
- **Problem**: the websocket request URL looks like ws://localhost:8080. Then, it can be filtered by most firewalls or network elements.

- **Solution**: Proxy servers, specifically Websocket tunneling proxy.

- Steps to configure apache to do that:
  - Open **httpd.conf** file.
  - Uncomment the module loading for **proxy_module**, **proxy_http_module** and **proxy_wstunnel_module**.
  - Configure a virtual host in the port 80 as shown in the figure.
  - Re-start apache web server.
  - Load **index-v2.html** instead **index.html**.



```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_ajp_module modules/mod_proxy_ajp.so
#LoadModule proxy_balancer_module modules/mod_proxy_balancer.so
#LoadModule proxy_connect_module modules/mod_proxy_connect.so
#LoadModule proxy_express_module modules/mod_proxy_express.so
#LoadModule proxy_fcgi_module modules/mod_proxy_fcgi.so
#LoadModule proxy_ftp_module modules/mod_proxy_ftp.so
#LoadModule proxy_html_module modules/mod_proxy_html.so
LoadModule proxy_http_module modules/mod_proxy_http.so
#LoadModule proxy_scgi_module modules/mod_proxy_scgi.so
LoadModule proxy_wstunnel_module modules/mod_proxy_wstunnel.so
```

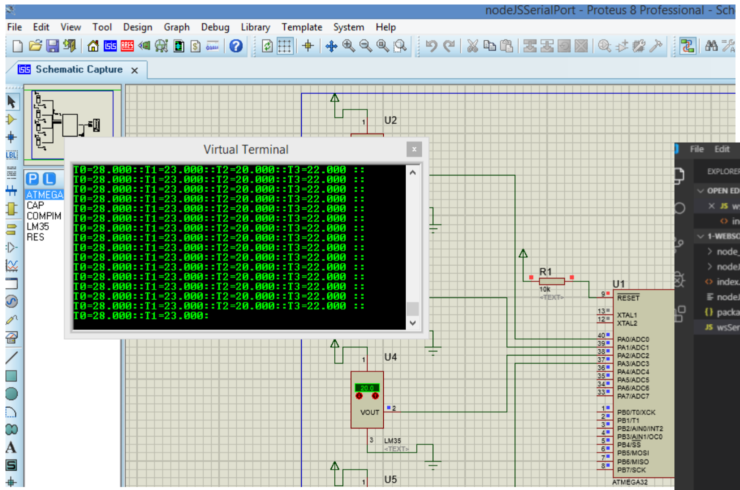


```
# ws_tunnel Module
<VirtualHost *:80>
  ServerName localhost

  <Location "/wsNJS">
    ProxyPass "ws://localhost:8080/"
  </Location>
</VirtualHost>
```

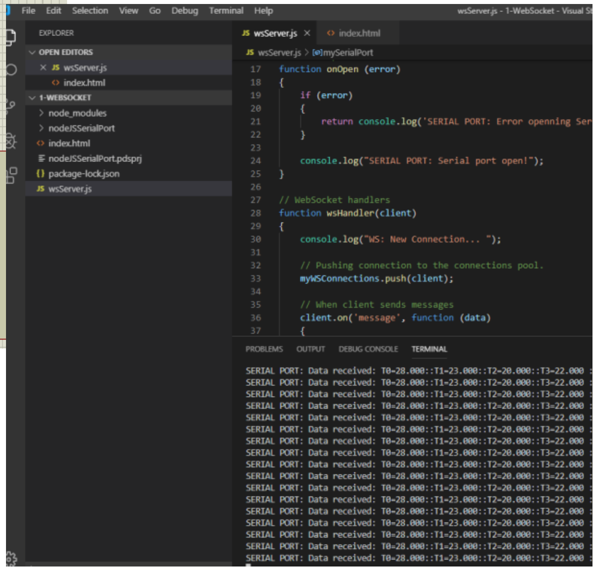
NodeJS and WebSockets

- Load **index-v2.html** instead **index.html**.

- **Expected result**:


nodeJSSerialPort - Proteus 8 Professional - Sch
File Edit View Tool Design Graph Debug Library Template System Help
Schematic Capture
Virtual Terminal
ATMEGA
CAP
COMPIM
LM35
RES
U2
U4
U5
R1
10k
U1
RESET
XTAL1
XTAL2
VOUT
LM35
ATMEGA32

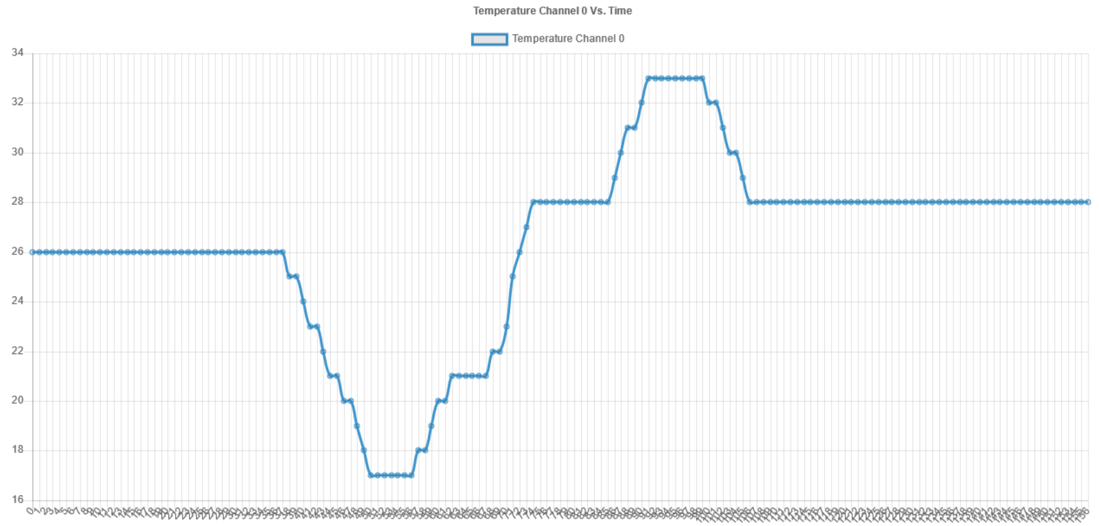
```
T0=28.000::T1=23.000::T2=20.000::T3=22.000 ::
T0=28.000::T1=23.000::T2=20.000::T3=22.000 ::
T0=28.000::T1=23.000::T2=20.000::T3=22.000 ::
T0=28.000::T1=23.000::T2=20.000::T3=22.000 ::
T0=28.000::T1=23.000::T2=20.000::T3=22.000 ::
T0=28.000::T1=23.000::T2=20.000::T3=22.000 ::
T0=28.000::T1=23.000::T2=20.000::T3=22.000 ::
T0=28.000::T1=23.000::T2=20.000::T3=22.000 ::
T0=28.000::T1=23.000::T2=20.000::T3=22.000 ::
T0=28.000::T1=23.000::T2=20.000::T3=22.000 ::
T0=28.000::T1=23.000::T2=20.000::T3=22.000 ::
T0=28.000::T1=23.000::T2=20.000::T3=22.000 ::
T0=28.000::T1=23.000::T2=20.000::T3=22.000 ::
T0=28.000::T1=23.000::T2=20.000::T3=22.000 ::
T0=28.000::T1=23.000::T2=20.000::T3=22.000 ::
T0=28.000::T1=23.000:
```



```javascript
function onOpen (error)
{
    if (error)
    {
        return console.log('SERIAL PORT: Error openning Ser
    }

    console.log("SERIAL PORT: Serial port open!");
}

// WebSocket handlers
function wsHandler(client)
{
    console.log("WS: New Connection... ");

    // Pushing connection to the connections pool.
    myWSConnections.push(client);

    // When client sends messages
    client.on('message', function (data)
    {
```

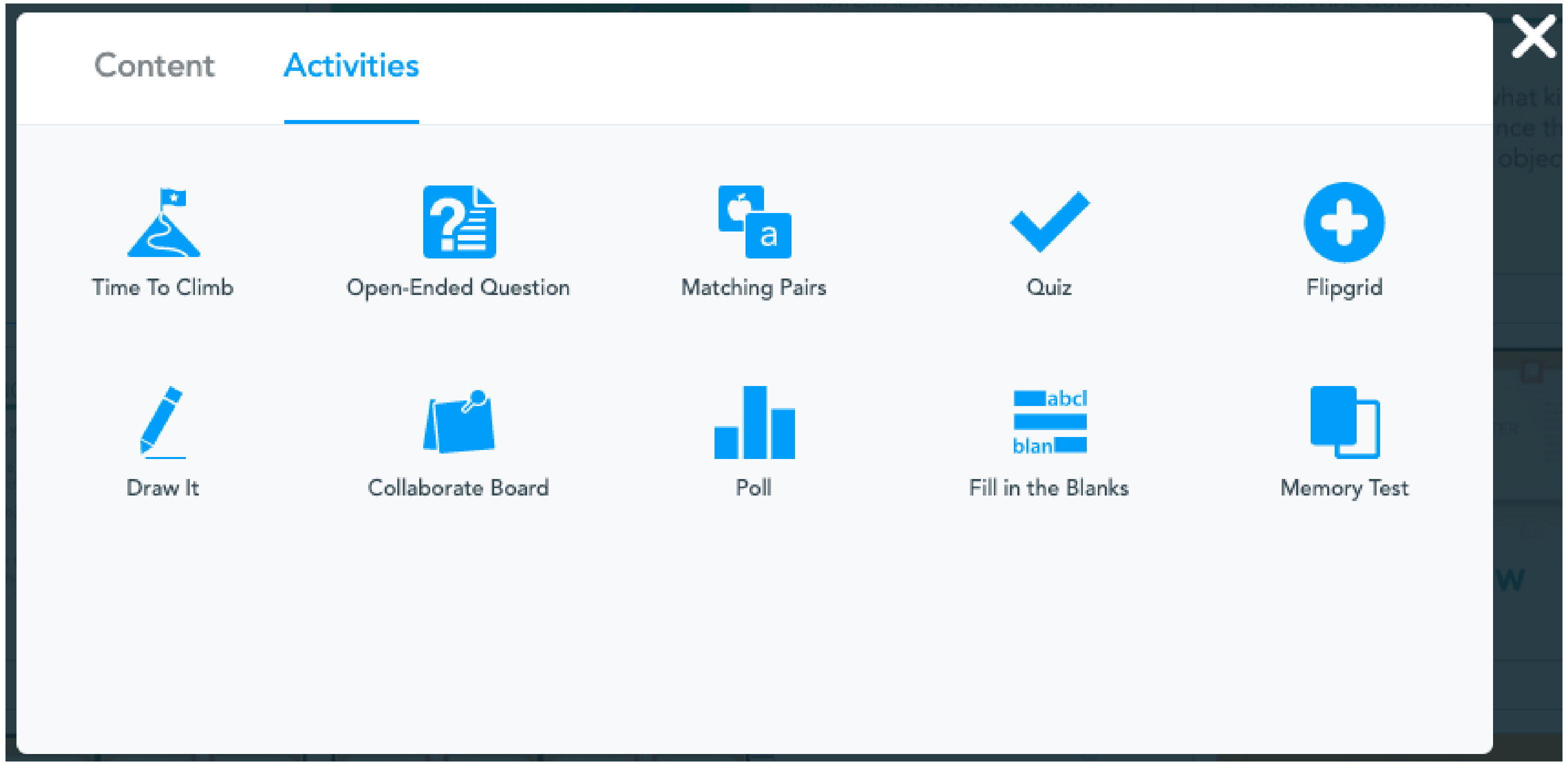
```
SERIAL PORT: Data received: T0=28.000::T1=23.000::T2=20.000::T3=22.000 :
SERIAL PORT: Data received: T0=28.000::T1=23.000::T2=20.000::T3=22.000 :
SERIAL PORT: Data received: T0=28.000::T1=23.000::T2=20.000::T3=22.000 :
SERIAL PORT: Data received: T0=28.000::T1=23.000::T2=20.000::T3=22.000 :
SERIAL PORT: Data received: T0=28.000::T1=23.000::T2=20.000::T3=22.000 :
SERIAL PORT: Data received: T0=28.000::T1=23.000::T2=20.000::T3=22.000 :
SERIAL PORT: Data received: T0=28.000::T1=23.000::T2=20.000::T3=22.000 :
SERIAL PORT: Data received: T0=28.000::T1=23.000::T2=20.000::T3=22.000 :
SERIAL PORT: Data received: T0=28.000::T1=23.000::T2=20.000::T3=22.000 :
SERIAL PORT: Data received: T0=28.000::T1=23.000::T2=20.000::T3=22.000 :
SERIAL PORT: Data received: T0=28.000::T1=23.000::T2=20.000::T3=22.000 :
SERIAL PORT: Data received: T0=28.000::T1=23.000::T2=20.000::T3=22.000 :
SERIAL PORT: Data received: T0=28.000::T1=23.000::T2=20.000::T3=22.000 :
SERIAL PORT: Data received: T0=28.000::T1=23.000::T2=20.000::T3=22.000 :
SERIAL PORT: Data received: T0=28.000::T1=23.000::T2=20.000::T3=22.000 :
SERIAL PORT: Data received: T0=28.000::T1=23.000::T2=20.000::T3=22.000 :
SERIAL PORT: Data received: T0=28.000::T1=23.000::T2=20.000::T3=22.000 :
```

REMOTE SENSOR READING USING NODE JS



Nearpod Activity

- Please go to the Nearpod link shared in the chat.
- Fulfil the Nearpod activity.
- Analyze the results with your teacher.

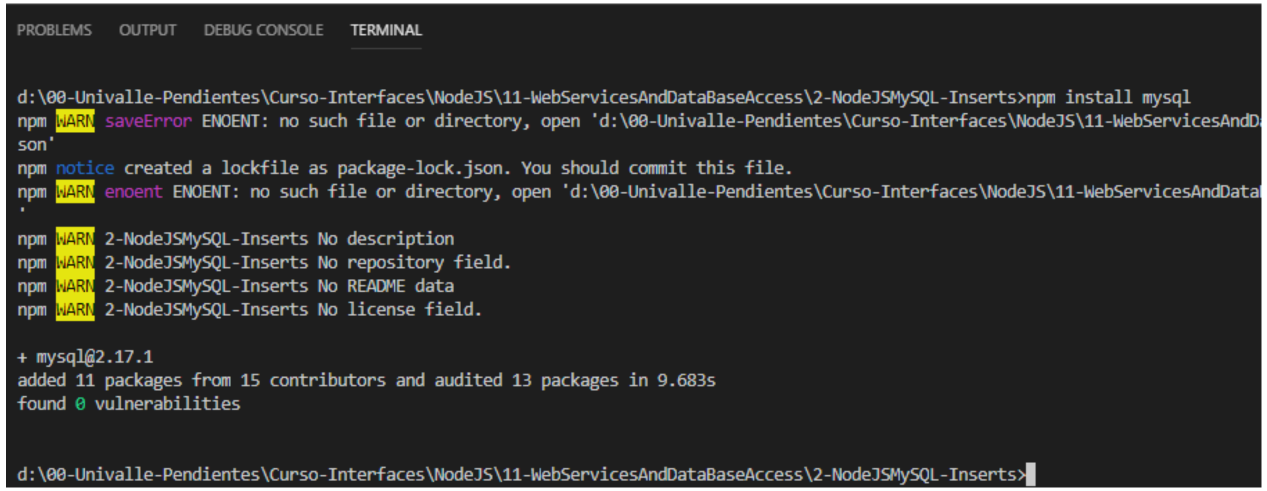


NodeJS and MySQL Databases – Setup

- **Node JS tools for MySQL**:

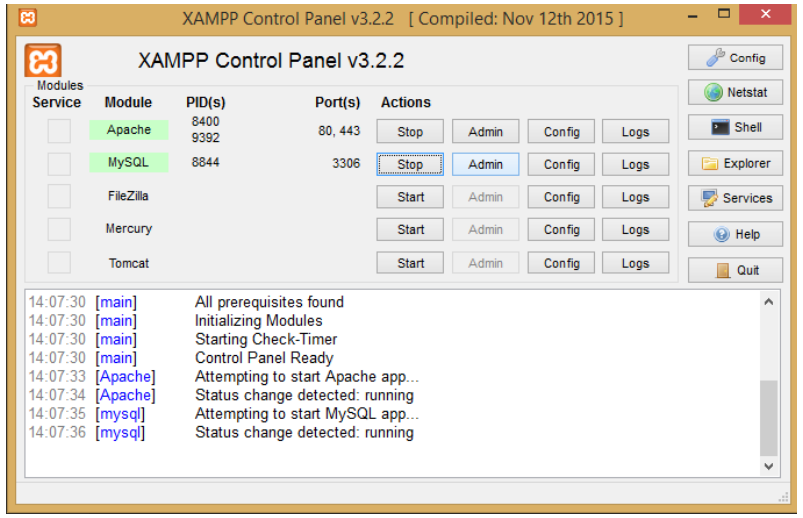
>npm install mysql

- Also, run **Xampp Control** and:
  - Run Apache and execute PHPmyADMIN
  - Run MySQL server.



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
d:\00-Univalle-Pendientes\Curso-Interfaces\NodeJS\11-WebServicesAndDataBaseAccess\2-NodeJSMySQL-Inserts>npm install mysql
npm WARN saveError ENOENT: no such file or directory, open 'd:\00-Univalle-Pendientes\Curso-Interfaces\NodeJS\11-WebServicesAndD
son'
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN enoent ENOENT: no such file or directory, open 'd:\00-Univalle-Pendientes\Curso-Interfaces\NodeJS\11-WebServicesAndData
'
npm WARN 2-NodeJSMySQL-Inserts No description
npm WARN 2-NodeJSMySQL-Inserts No repository field.
npm WARN 2-NodeJSMySQL-Inserts No README data
npm WARN 2-NodeJSMySQL-Inserts No license field.

+ mysql@2.17.1
added 11 packages from 15 contributors and audited 13 packages in 9.683s
found 0 vulnerabilities


d:\00-Univalle-Pendientes\Curso-Interfaces\NodeJS\11-WebServicesAndDataBaseAccess\2-NodeJSMySQL-Inserts>
```



XAMPP Control Panel v3.2.2 [Compiled: Nov 12th 2015]

XAMPP Control Panel v3.2.2

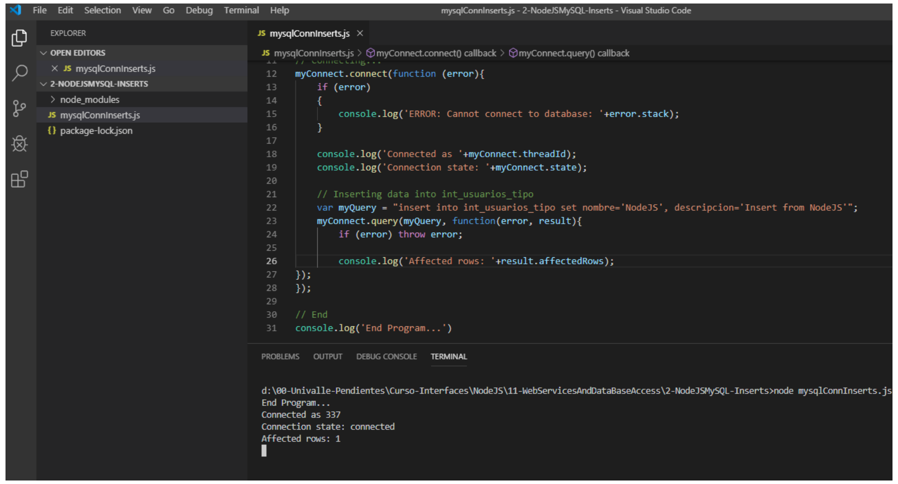
Modules Service	Module	PID(s)	Port(s)	Actions			
	Apache	8400 9392	80, 443	Stop	Admin	Config	Logs
	MySQL	8844	3306	Stop	Admin	Config	Logs
	FileZilla			Start	Admin	Config	Logs
	Mercury			Start	Admin	Config	Logs
	Tomcat			Start	Admin	Config	Logs

```
14:07:30 [main]    All prerequisites found
14:07:30 [main]    Initializing Modules
14:07:30 [main]    Starting Check-Timer
14:07:30 [main]    Control Panel Ready
14:07:33 [Apache]  Attempting to start Apache app...
14:07:34 [Apache]  Status change detected: running
14:07:35 [mysql]   Attempting to start MySQL app...
14:07:36 [mysql]   Status change detected: running
```

NodeJS and MySQL Databases – Connection and Inserts

- **Connecting with a database**: mysql.createConnection()
  - Main parameters: Host, User, Password, Database
  - It has other 18 parameters.
- **Terminating database connections**:
  - *end() method*: It ensures previous queued queries are sent to server.
  - *destroy() method*: This will cause an immediate termination of the underlying socket. It guarantees that no more events or callbacks will be triggered for the connection.
- **Pooling of connections**: Cache of database connections which are useful to send multiple queries to database.

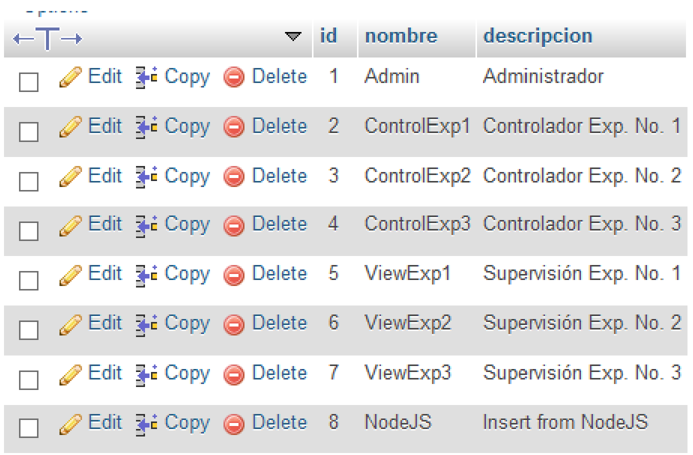
- Open folder **2-NodeJSMySQLInserts**, and run **mysqlConnInserts.js**.



```javascript
myConnect.connect(function (error){
    if (error)
    {
        console.log('ERROR: Cannot connect to database: '+error.stack);
    }

    console.log('Connected as '+myConnect.threadId);
    console.log('Connection state: '+myConnect.state);

    // Inserting data into int_usuarios_tipo
    var myQuery = "insert into int_usuarios_tipo set nombre='NodeJS', descripcion='Insert from NodeJS'";
    myConnect.query(myQuery, function(error, result){
        if (error) throw error;

        console.log('Affected rows: '+result.affectedRows);
});
});

// End
console.log('End Program...')
```

```
d:\00-Univalle-Pendientes\Curso-Interfaces\NodeJS\11-WebServicesAndDataBaseAccess\2-NodeJSMySQL-Inserts>node mysqlConnInserts.js
End Program...
Connected as 337
Connection state: connected
Affected rows: 1
```

```javascript
var mysql = require('mysql');
var pool  = mysql.createPool({
  connectionLimit : 10,
  host            : 'example.org',
  user            : 'bob',
  password        : 'secret',
  database        : 'my_db'
});
```



				id	nombre	descripcion
☐	Edit	Copy	Delete	1	Admin	Administrador
☐	Edit	Copy	Delete	2	ControlExp1	Controlador Exp. No. 1
☐	Edit	Copy	Delete	3	ControlExp2	Controlador Exp. No. 2
☐	Edit	Copy	Delete	4	ControlExp3	Controlador Exp. No. 3
☐	Edit	Copy	Delete	5	ViewExp1	Supervisión Exp. No. 1
☐	Edit	Copy	Delete	6	ViewExp2	Supervisión Exp. No. 2
☐	Edit	Copy	Delete	7	ViewExp3	Supervisión Exp. No. 3
☐	Edit	Copy	Delete	8	NodeJS	Insert from NodeJS

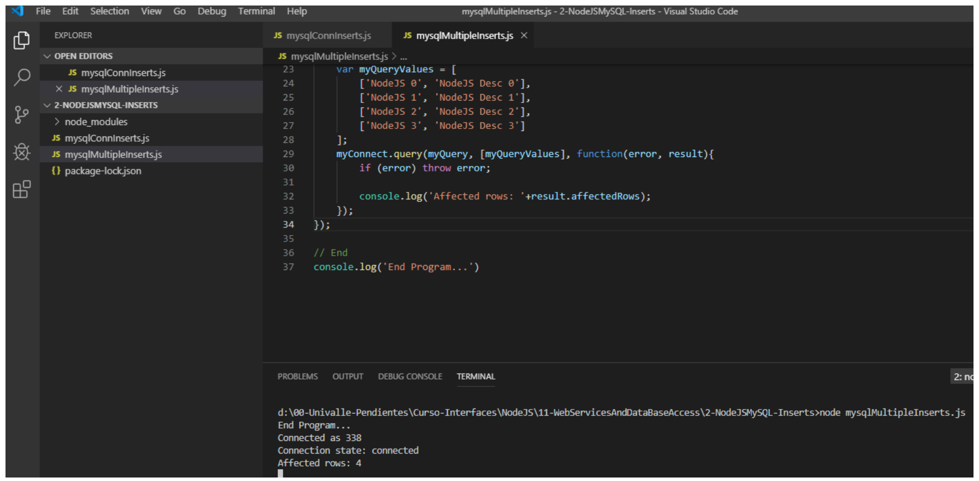
NodeJS and MySQL Databases – Multiple Inserts

- **To insert more than one record in one table**:
  - Define an array with the values.
  - Define the SQL statement with a '?'
  - The '?' will be replaced by the value array.

- Open folder **2-NodeJSMySQLInserts**, and run **mysqlMultipleInserts.js**.

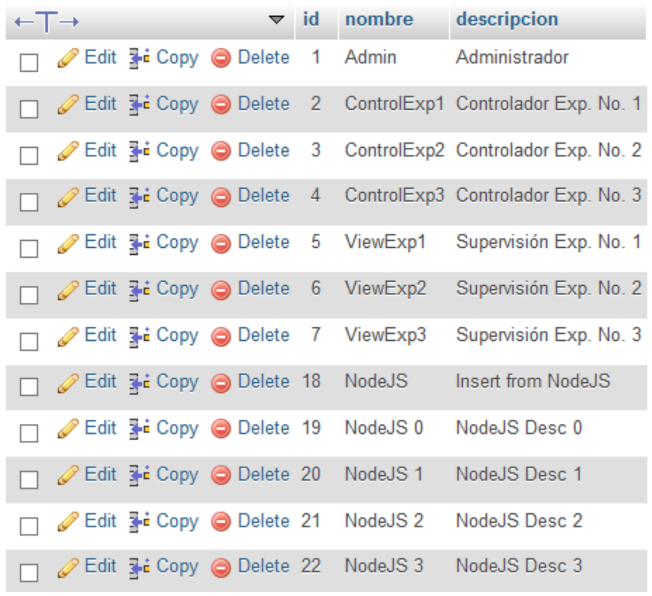
- **Result object**: The result object contains information about how the query affected the table. For instance, it would look like this:

```
{
  fieldCount: 0,
  affectedRows: 14,
  insertId: 0,
  serverStatus: 2,
  warningCount: 0,
  message: '\'Records:14  Duplicated: 0  Warnings: 0',
  protocol41: true,
  changedRows: 0
}
```



```
    var myQueryValues = [
        ['NodeJS 0', 'NodeJS Desc 0'],
        ['NodeJS 1', 'NodeJS Desc 1'],
        ['NodeJS 2', 'NodeJS Desc 2'],
        ['NodeJS 3', 'NodeJS Desc 3']
    ];
    myConnect.query(myQuery, [myQueryValues], function(error, result){
        if (error) throw error;

        console.log('Affected rows: '+result.affectedRows);
    });
});

// End
console.log('End Program...')
```

```
d:\00-Univalle-Pendientes\Curso-Interfaces\NodeJS\11-WebServicesAndDataBaseAccess\2-NodeJSMySQL-Inserts>node mysqlMultipleInserts.js
End Program...
Connected as 338
Connection state: connected
Affected rows: 4
```



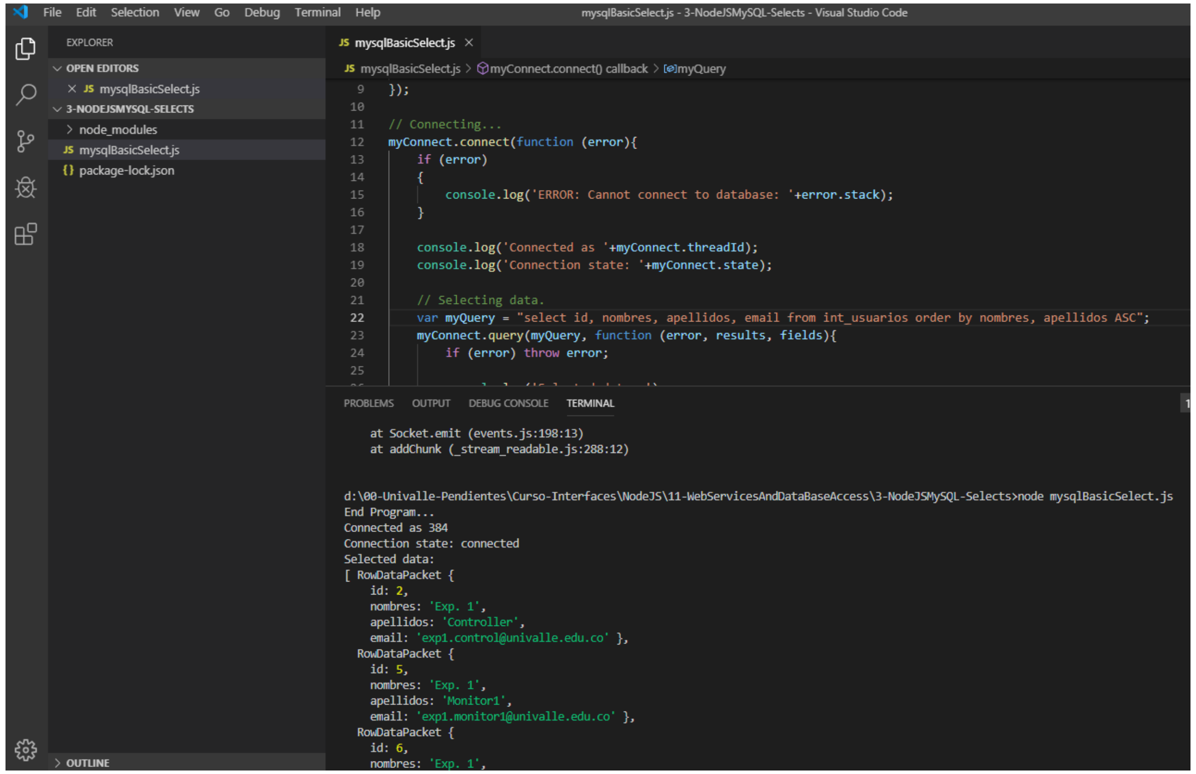
	id	nombre	descripcion
Edit Copy Delete	1	Admin	Administrador
Edit Copy Delete	2	ControlExp1	Controlador Exp. No. 1
Edit Copy Delete	3	ControlExp2	Controlador Exp. No. 2
Edit Copy Delete	4	ControlExp3	Controlador Exp. No. 3
Edit Copy Delete	5	ViewExp1	Supervisión Exp. No. 1
Edit Copy Delete	6	ViewExp2	Supervisión Exp. No. 2
Edit Copy Delete	7	ViewExp3	Supervisión Exp. No. 3
Edit Copy Delete	18	NodeJS	Insert from NodeJS
Edit Copy Delete	19	NodeJS 0	NodeJS Desc 0
Edit Copy Delete	20	NodeJS 1	NodeJS Desc 1
Edit Copy Delete	21	NodeJS 2	NodeJS Desc 2
Edit Copy Delete	22	NodeJS 3	NodeJS Desc 3

NodeJS and MySQL Databases – Selects

- **Selecting data from tables**:

    - Objects to deal with **results** and **fields**.
    - **Results** is an array of values extracted from table.
    - Table rows can be accessed using array notation.
    - Fields in a row can be accessed using the dot operator.
    - **Fields** object contains information about each field in the result.

- Open folder **3-NodeJSMySQL-Selects**, and run **mysqlBasicSelect.js**.


File Edit Selection View Go Debug Terminal Help
mysqlBasicSelect.js - 3-NodeJSMySQL-Selects - Visual Studio Code
EXPLORER
OPEN EDITORS
mysqlBasicSelect.js
3-NODEJSMYSQL-SELECTS
node_modules
mysqlBasicSelect.js
package-lock.json
OUTLINE
mysqlBasicSelect.js > myConnect.connect() callback > myQuery
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```javascript
});

// Connecting...
myConnect.connect(function (error){
    if (error)
    {
        console.log('ERROR: Cannot connect to database: '+error.stack);
    }

    console.log('Connected as '+myConnect.threadId);
    console.log('Connection state: '+myConnect.state);

    // Selecting data.
    var myQuery = "select id, nombres, apellidos, email from int_usuarios order by nombres, apellidos ASC";
    myConnect.query(myQuery, function (error, results, fields){
        if (error) throw error;

```

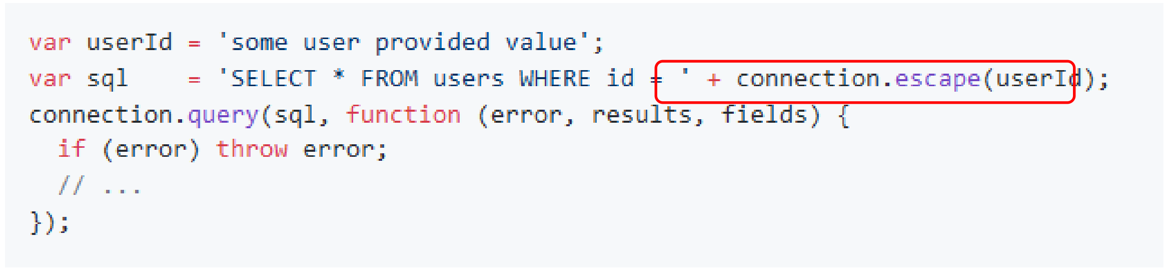
```
    at Socket.emit (events.js:198:13)
    at addChunk (_stream_readable.js:288:12)


d:\00-Univalle-Pendientes\Curso-Interfaces\NodeJS\11-WebServicesAndDataBaseAccess\3-NodeJSMySQL-Selects>node mysqlBasicSelect.js
End Program...
Connected as 384
Connection state: connected
Selected data:
[ RowDataPacket {
    id: 2,
    nombres: 'Exp. 1',
    apellidos: 'Controller',
    email: 'exp1.control@univalle.edu.co' },
  RowDataPacket {
    id: 5,
    nombres: 'Exp. 1',
    apellidos: 'Monitor1',
    email: 'exp1.monitor1@univalle.edu.co' },
  RowDataPacket {
    id: 6,
    nombres: 'Exp. 1',
```

NodeJS and MySQL Databases – Selects

- **Escaping Query Values.**

- In order to avoid SQL **Injection attacks**, you should always **escape any** user **provided data** before using it inside a SQL query.
- **Injection Attacks**: It occurs when an attacker supplies untrusted input to a program to induce a bad-function of a web application.

- Methods to prevent this:
  - *mysql.escape()*
  - *connection.escape*() or
  - *pool.escape*() methods



```
var userId = 'some user provided value';
var sql    = 'SELECT * FROM users WHERE id = ' + connection.escape(userId);
connection.query(sql, function (error, results, fields) {
  if (error) throw error;
  // ...
});
```

- **How to escape values:**

- Numbers are left untouched
- Booleans are converted to true / false
- Date objects are converted to 'YYYY-mm-dd HH:ii:ss' strings
- Buffers are converted to hex strings
- Strings are safely escaped
- Arrays are turned into list, e.g. ['a', 'b'] turns into 'a', 'b'
- Nested arrays are turned into grouped lists,e.g. [['a', 'b'], ['c', 'd']] turns into ('a', 'b'), ('c', 'd')
- Objects that have a toSqlString method will have .toSqlString() called
- Objects are turned into key = 'val' pairs for each enumerable property on the object.
- undefined / null are converted to NULL
- NaN / Infinity are left as-is.

NodeJS and MySQL Databases – Selects

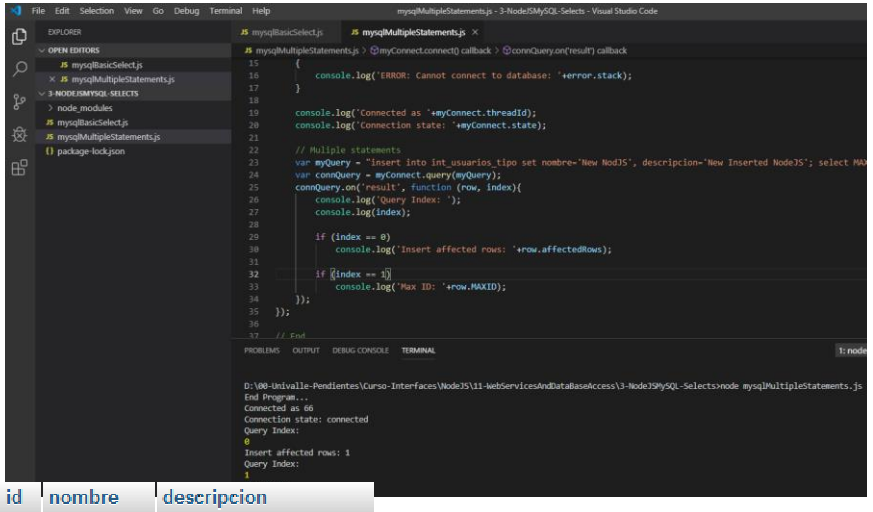
- **Multiple statements query**:

- Support for multiple statements is disabled for security reasons.
- You must enable it explicitly:

```
var connection = mysql.createConnection({multipleStatements: true});
```

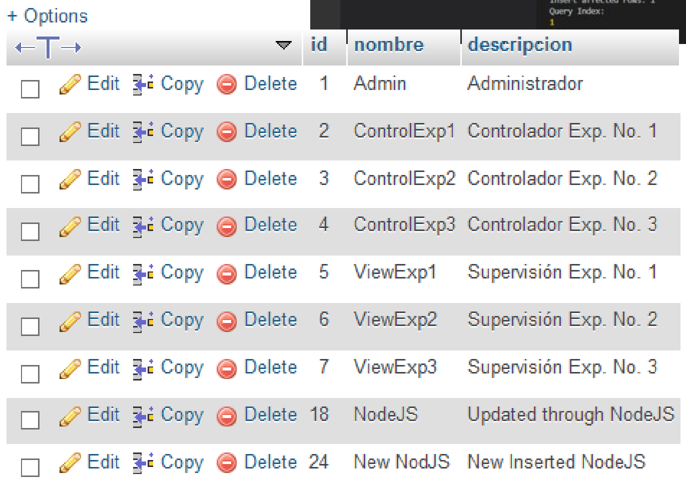
```
connection.query('SELECT 1; SELECT 2', function (error, results, fields) {
  if (error) throw error;
  // `results` is an array with one element for every statement in the query:
  console.log(results[0]); // [{1: 1}]
  console.log(results[1]); // [{2: 2}]
});
```

- Open folder **3-NodeJSMySQL-Selects**, and run **mysqlMultipleStatements.js**.



```
        {
            console.log('ERROR: Cannot connect to database: '+error.stack);
        }

        console.log('Connected as '+myConnect.threadId);
        console.log('Connection state: '+myConnect.state);

        // Muliple statements
        var myQuery = "insert into int_usuarios_tipo set nombre='New NodJS', descripcion='New Inserted NodeJS'; select MA
        var connQuery = myConnect.query(myQuery);
        connQuery.on('result', function (row, index){
            console.log('Query Index: ');
            console.log(index);

            if (index == 0)
                console.log('Insert affected rows: '+row.affectedRows);

            if (index == 1)
                console.log('Max ID: '+row.MAXID);
        });
    });

// End
```

```
D:\00-Univalle-Pendientes\Curso-Interfaces\NodeJS\11-WebServicesAndDataBaseAccess\3-NodeJSMySQL-Selects>node mysqlMultipleStatements.js
End Program...
Connected as 66
Connection state: connected
Query Index:
0
Insert affected rows: 1
Query Index:
1
```



+ Options

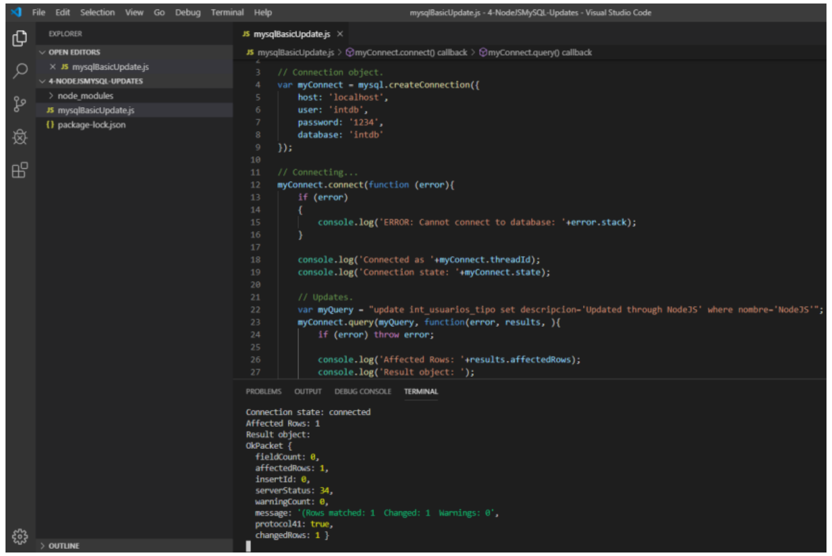
		id	nombre	descripcion
☐	Edit Copy Delete	1	Admin	Administrador
☐	Edit Copy Delete	2	ControlExp1	Controlador Exp. No. 1
☐	Edit Copy Delete	3	ControlExp2	Controlador Exp. No. 2
☐	Edit Copy Delete	4	ControlExp3	Controlador Exp. No. 3
☐	Edit Copy Delete	5	ViewExp1	Supervisión Exp. No. 1
☐	Edit Copy Delete	6	ViewExp2	Supervisión Exp. No. 2
☐	Edit Copy Delete	7	ViewExp3	Supervisión Exp. No. 3
☐	Edit Copy Delete	18	NodeJS	Updated through NodeJS
☐	Edit Copy Delete	24	New NodJS	New Inserted NodeJS

NodeJS and MySQL Databases – Updates

- **Updating data from tables**:

  - Object to deal with **results**.
  - The result object contains information about how the query affected the table.

- Open folder **4-NodeJSMySQL-Updates**, and run **mysqlBasicUpdate.js**.

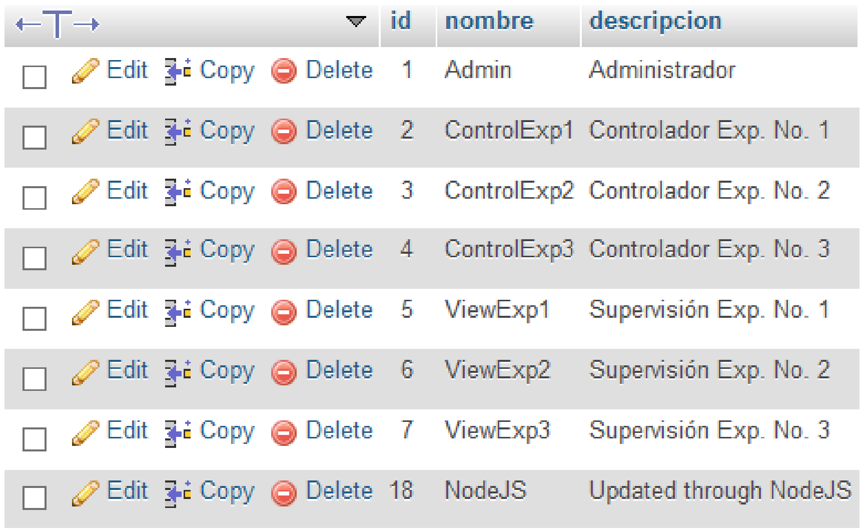


File Edit Selection View Go Debug Terminal Help
mysqlBasicUpdate.js - 4-NodeJSMySQL-Updates - Visual Studio Code
EXPLORER
OPEN EDITORS
mysqlBasicUpdate.js
4-NODEJSMYSQL-UPDATES
node_modules
mysqlBasicUpdate.js
package-lock.json
OUTLINE
mysqlBasicUpdate.js > myConnect.connect() callback > myConnect.query() callback
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
// Connection object.
var myConnect = mysql.createConnection({
    host: 'localhost',
    user: 'intdb',
    password: '1234',
    database: 'intdb'
});

// Connecting...
myConnect.connect(function (error){
    if (error)
    {
        console.log('ERROR: Cannot connect to database: '+error.stack);
    }

    console.log('Connected as '+myConnect.threadId);
    console.log('Connection state: '+myConnect.state);

    // Updates.
    var myQuery = "update int_usuarios_tipo set descripcion='Updated through NodeJS' where nombre='NodeJS'";
    myConnect.query(myQuery, function(error, results, ){
        if (error) throw error;

        console.log('Affected Rows: '+results.affectedRows);
        console.log('Result object: ');
```

```
Connection state: connected
Affected Rows: 1
Result object:
OkPacket {
  fieldCount: 0,
  affectedRows: 1,
  insertId: 0,
  serverStatus: 34,
  warningCount: 0,
  message: '(Rows matched: 1  Changed: 1  Warnings: 0',
  protocol41: true,
  changedRows: 1 }
```

```
{
  fieldCount: 0,
  affectedRows: 1,
  insertId: 0,
  serverStatus: 34,
  warningCount: 0,
  message: '(Rows matched: 1 Changed: 1 Warnings: 0',
  protocol41: true,
  changedRows: 1
}
```



←T→				id	nombre	descripcion
☐	Edit	Copy	Delete	1	Admin	Administrador
☐	Edit	Copy	Delete	2	ControlExp1	Controlador Exp. No. 1
☐	Edit	Copy	Delete	3	ControlExp2	Controlador Exp. No. 2
☐	Edit	Copy	Delete	4	ControlExp3	Controlador Exp. No. 3
☐	Edit	Copy	Delete	5	ViewExp1	Supervisión Exp. No. 1
☐	Edit	Copy	Delete	6	ViewExp2	Supervisión Exp. No. 2
☐	Edit	Copy	Delete	7	ViewExp3	Supervisión Exp. No. 3
☐	Edit	Copy	Delete	18	NodeJS	Updated through NodeJS

Questions?

El problema de las mentes cerradas, es que siempre tienen la boca abierta...