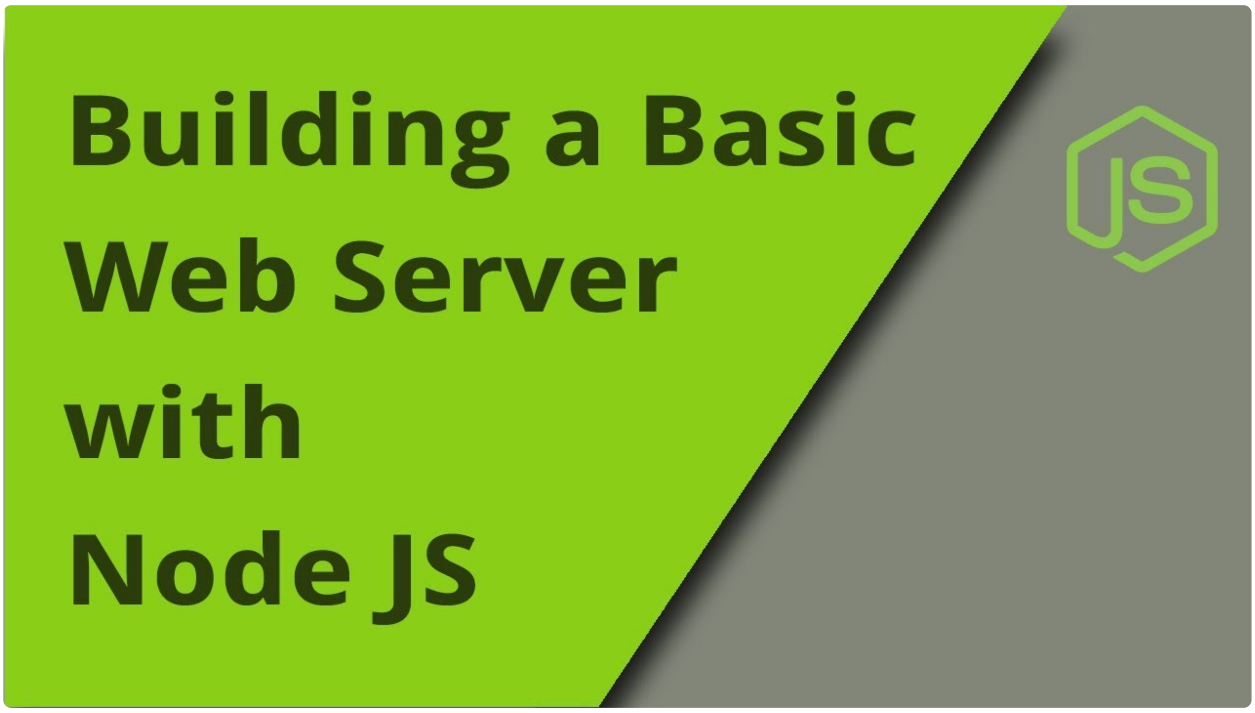
05 - Create Servers and Get Data - PT



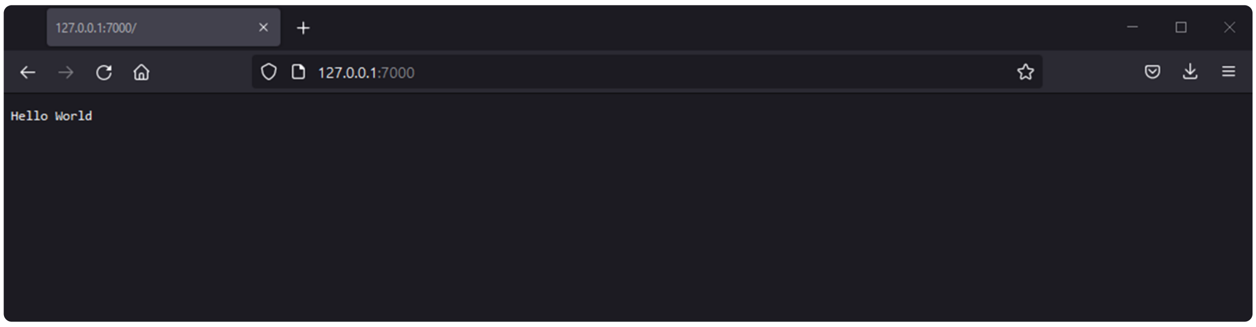
NODE AS HTTP WEB SERVER

Node.js is primarily used to build server-based applications. The framework can be easily used to build web servers, which can serve content to users.

There are a variety of modules, such as the "http" and "request" modules, that help in processing web requests.

In this chapter we will see how we can create a basic web server with Node.js.

```
var http=require('http')
var server=http.createServer((function(request,response)
{
    response.writeHead(200,
    {"Content-Type" : "text/plain"});
    response.end("Hello World\n");
}));
server.listen(7000);
```



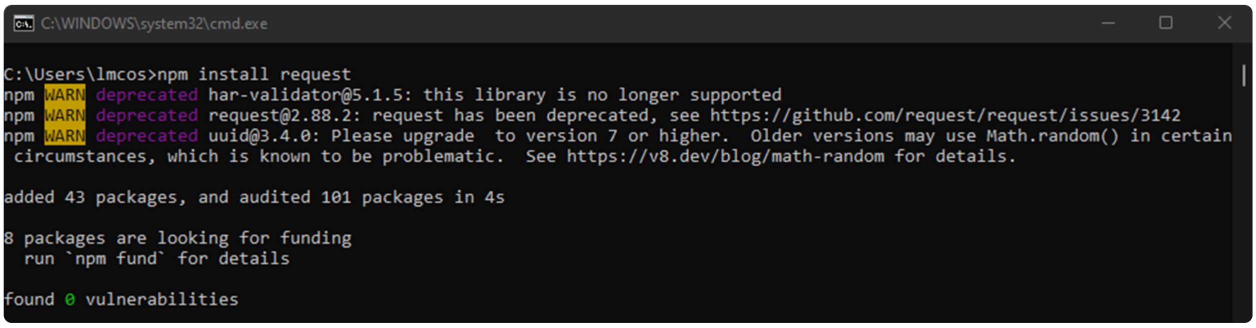
```
Hello World
```

GET REQUEST IN NODE.JS

Making GET requests to get data from another website is relatively straightforward in Node.js. To make a GET request in node, we first need to have the “request” module installed.

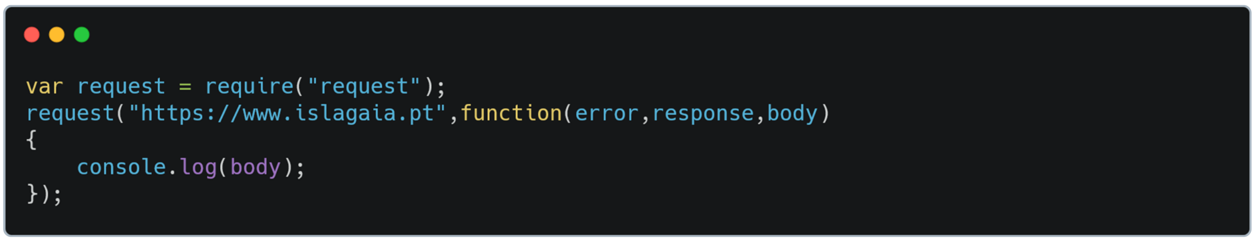
This can be done by running the following command in the terminal:

```
npm install request
```



```
C:\Users\lmcos>npm install request
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142
npm WARN deprecated uuid@3.4.0: Please upgrade  to version 7 or higher.  Older versions may use Math.random() in certain
 circumstances, which is known to be problematic.  See https://v8.dev/blog/math-random for details.

added 43 packages, and audited 101 packages in 4s

8 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Let's now test a block of code that uses the request module.



```javascript
var request = require("request");
request("https://www.islagaia.pt",function(error,response,body)
{
    console.log(body);
});
```

We are using the 'request' module which has the necessary functions to make GET requests. We made a GET request towww.islagaia.ptand then we call a function when a response is received.

When a response is received, the parameters (error, response and body) will have the following values:

- error - If there is any error received when using the GET request
- response - The response will have the http headers that are sent in the response. body - Will
- contain the entire content of the response sent by the website.

Then we write the content received in the body parameter to the console.

SUMMARY

Node.js can be used to develop web servers using the 'http' module.

The application can be created to listen on a specific port and send a response to the client whenever a request is made.

The 'request' module can be used to get information from websites.

CREATE A WEB SERVER WITH NODE.JS

A Web Server is software that handles HTTP requests sent by HTTP clients, such as web browsers, and returns web pages in response to the client requests.

Web servers usually respond with html documents along with images, style sheets (css) and scripts.

Most web servers support server-side scripting, using a scripting language, or redirect to the application server which performs the specific task of fetching data from a database, executing complex logic, etc., and then returning a result to the HTTP client.

ARCHITECTURE OF A WEB APPLICATION

Client layer

The client layer contains web browsers, mobile browsers, or applications that can make HTTP requests to the web server.

Server layer

The server layer contains the web server which can intercept the request made by the clients and return the response.

Business layer

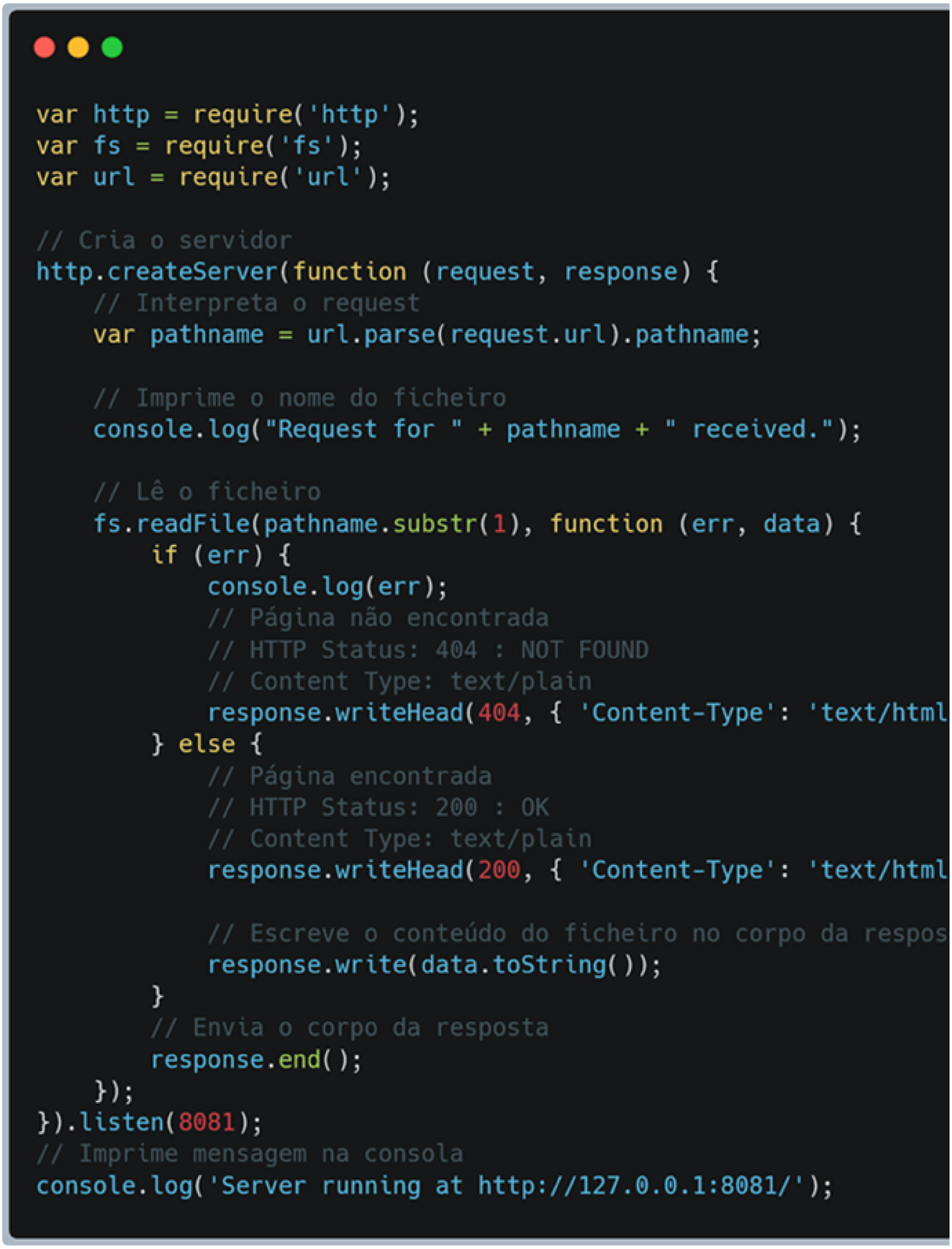
The business layer contains the application server that is used by the web server to perform the necessary processing. This layer interacts with the data layer via a database or external programs.

Data layer

The data layer contains the database or any data source.

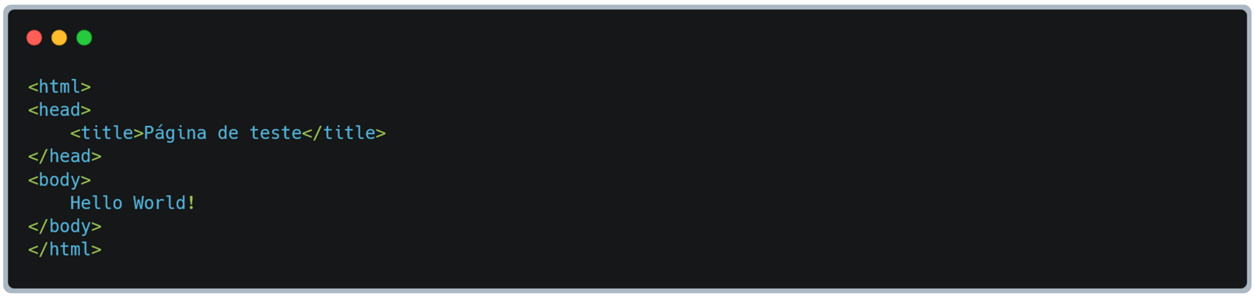
Node.js provides the http module which can be used to create an HTTP client or server.

Create a .js file called server.js with the following code:



```
var http = require('http');
var fs = require('fs');
var url = require('url');

// Cria o servidor
http.createServer(function (request, response) {
    // Interpreta o request
    var pathname = url.parse(request.url).pathname;

    // Imprime o nome do ficheiro
    console.log("Request for " + pathname + " received.");

    // Lê o ficheiro
    fs.readFile(pathname.substr(1), function (err, data) {
        if (err) {
            console.log(err);
            // Página não encontrada
            // HTTP Status: 404 : NOT FOUND
            // Content Type: text/plain
            response.writeHead(404, { 'Content-Type': 'text/html
        } else {
            // Página encontrada
            // HTTP Status: 200 : OK
            // Content Type: text/plain
            response.writeHead(200, { 'Content-Type': 'text/html

            // Escreve o conteúdo do ficheiro no corpo da respos
            response.write(data.toString());
        }
        // Envia o corpo da resposta
        response.end();
    });
}).listen(8081);
// Imprime mensagem na consola
console.log('Server running at http://127.0.0.1:8081/');
```

Create the index.html file inside the project folder:



```
<html>
<head>
    <title>Página de teste</title>
</head>
<body>
    Hello World!
</body>
</html>
```

CREATE A MYSQL CONNECTION

If you don't already have Mysql on your computer, install it now. A (recommended) alternative to Mysql would be to install MariaDB.

You can find the installation packages at the following addresses:

Mysql:https://www.mysql.com/downloads/

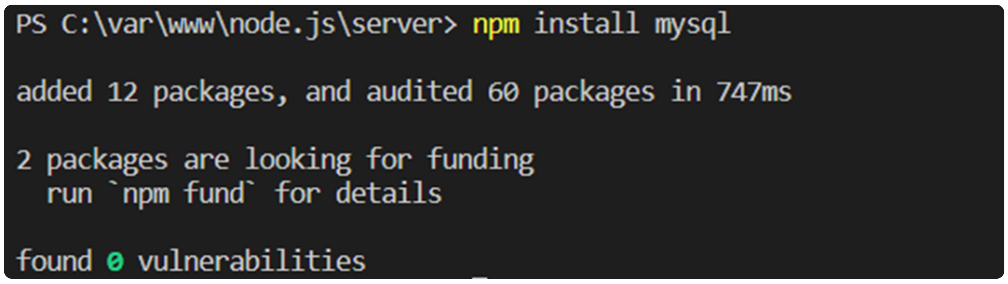
MariaDB:https://mariadb.org/download/

INSTALL DRIVER ON NODE.JS

You now need to install the MySQL driver to access the database with Node.js.

Download the Mysql module via npm. To download and install the "mysql" module, open the terminal and run the following command: npm install mysql

```
npm install mysql
```



```
PS C:\var\www\node.js\server> npm install mysql

added 12 packages, and audited 60 packages in 747ms

2 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

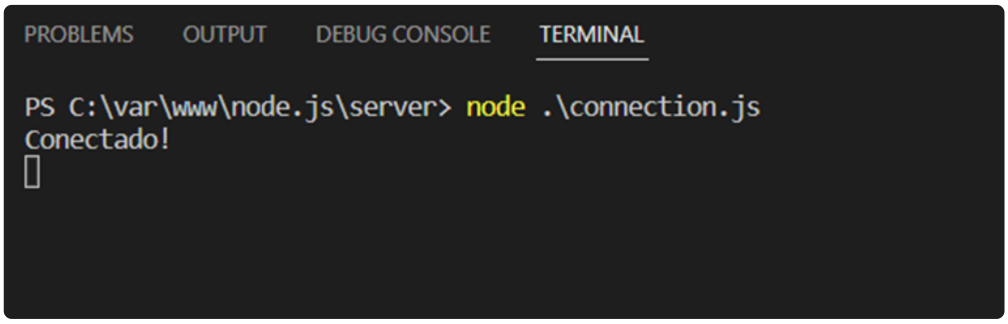
CREATE A CONNECTION

Inside the project folder, create a file named connection.js and enter the following code:

```
var mysql = require('mysql');
var con = mysql.createConnection({
    host: "o_meu_servidor",
    user: "o_meu_username",
    password: "a_minha_password"
});
con.connect(function (err) {
    if (err) throw err;
    console.log("Conectado!");
});
```

TEST THE CONNECTION

To test the connection run: node connection.js. If everything worked correctly you will get the message “Connected!”.



```
PS C:\var\www\node.js\server> node .\connection.js
Conectado!
```

CREATE A DATABASE

Create a new file named createdatabase.js inside your project folder. Enter the following code:

```
var mysql = require('mysql');
var con = mysql.createConnection({
    host: "o_meu_servidor",
    user: "o_meu_username",
    password: "a_minha_password"
});
con.connect(function (err) {
    if (err) throw err;
    console.log("Conectado!");
    con.query("CREATE DATABASE escoladb", function (err, result) {
        if (err) throw err;
        console.log("Base de dados criada com sucesso!");
    });
});
```

Verify that the database was created successfully.

CREATE A TABLE

Now let's create a table called students and reference the database in the connection.

Create a file named createtable.js and insert the following code:

```
var mysql = require('mysql');
var con = mysql.createConnection({
    host: "o_meu_servidor",
    user: "o_meu_username",
    password: "a_minha_password",
    database: "escoladb"
});
con.connect(function (err) {
    if (err) throw err;
    console.log("Conectado!");
    var sql = "CREATE TABLE alunos (id INT, nome VARCHAR(255), idade INT(3), morada VARCHAR(255), codigo_postal
VARCHAR(50))";
    con.query(sql, function (err, result) {
        if (err) throw err;
        console.log("Tabela criada com sucesso!");
    });
});
```

Verify that the table was created successfully.

INSERT 1 RECORD

Now let's insert a record. Create a file insertrecord.js

```
var mysql = require('mysql');
var con = mysql.createConnection({
    host: "o_meu_servidor",
    user: "o_meu_username",
    password: "a_minha_password",
    database: "escoladb"
});
con.connect(function (err) {
    if (err) throw err;
    console.log("Conectado!");
    var sql = "INSERT INTO alunos (id, nome, idade, morada, codigo_postal) VALUES ('1', 'Luis Osorio', '50', 'Rua
do Pombal', '4430-612 VNG')";
    con.query(sql, function (err, result) {
        if (err) throw err;
        console.log("1 registo inserido");
    });
});
```

Verify that the record was created successfully.

CHALLENGE

Create 2 scripts with the following functionality:

- Change id field to primary key Add a
- field for email

INSERT MULTIPLE RECORDS

Create a file named insertvarios.js and insert the following code:

```
var mysql = require('mysql');
var con = mysql.createConnection({
    host: "o_meu_servidor",
    user: "o_meu_username",
    password: "a_minha_password",
    database: "escoladb"
});
con.connect(function (err) {
    if (err) throw err;
    console.log("Conectado!");
    var sql = "INSERT INTO alunos (id, nome, idade, morada, codigo_postal, email) VALUES ?";
    var values = [
        ['2', 'Elon Musk', '55', 'Los Angeles', '54144', 'elon@tesla.com' ],
        ['3', 'Bill Gates', '62', 'San Francisco', '35255', 'bill@microsoft.com'],
        ['4', 'Jeff Bezos', '59', 'San Francisco', '354781', 'jeff@amazon.com']
        ];
    con.query(sql, [values], function (err, result) {
        if (err) throw err;
        console.log("Total de registos inseridos: " + result.affectedRows);
    });
});
```

SELECT RECORDS

Create a file named select.js and enter the following code:

```
var mysql = require('mysql');
var con = mysql.createConnection({
    host: "o_meu_servidor",
    user: "o_meu_username",
    password: "a_minha_password",
    database: "escoladb"
});
con.connect(function (err) {
    if (err) throw err;
    con.query("SELECT * FROM alunos", function (err, result) {
        if (err) throw err;
        console.log(result);
    });
});
```

CHALLENGE

- Change the address of student “Jeff Bezos” to “New York”
- Delete student with ID 1