

La última vez que programamos para la web, creamos nuestro archivo de JavaScript para controlar que los datos de un formulario cumplan con ciertas condiciones, **Validamos** que los datos estuviesen correctos. Luego conectamos el formulario para que agregara un registro en la tabla que tenemos creada en **MySQL**. Vimos algunos conceptos importantes de la arquitectura MVC y a partir de hoy vamos a ponerlos en práctica.

Como vimos, la razón de utilizar **MVC** es que nos permite separar los componentes de nuestra aplicación **dependiendo de la responsabilidad que tienen**, esto significa que cuando hacemos un cambio en alguna parte de nuestro código, esto no afecte otra parte del mismo.

Vamos a desarrollar una **aplicación web** que servirá para usar como **Agenda**. Haremos un **CRUD** de los contactos, utilizando **Node.js** y el framework **Express**.

Node.js



Es un **entorno de ejecución JavaScript de código abierto y multiplataforma** para desarrollar aplicaciones del lado del servidor. Fue creado por los desarrolladores originales de JavaScript para transformar algo que sólo podía ejecutarse en un navegador en una aplicación que se pueda ejecutar en una computadora.

Express

Express es un **framework** (marco de desarrollo) minimalista para **Node.js** que permite estructurar una aplicación de una manera ágil. Nos proporciona funcionalidades como el enrutamiento, opciones para gestionar sesiones y cookies, etc. La ventaja de usarlo es que contiene funciones ya probadas y nos facilita hacer muchas cosas con poco esfuerzo y escribiendo poco código.



npm

npm responde a las siglas de **Node Package Manager** o **manejador de paquetes de Node**, es la herramienta por defecto de JavaScript para la tarea de compartir e instalar paquetes (librerías).

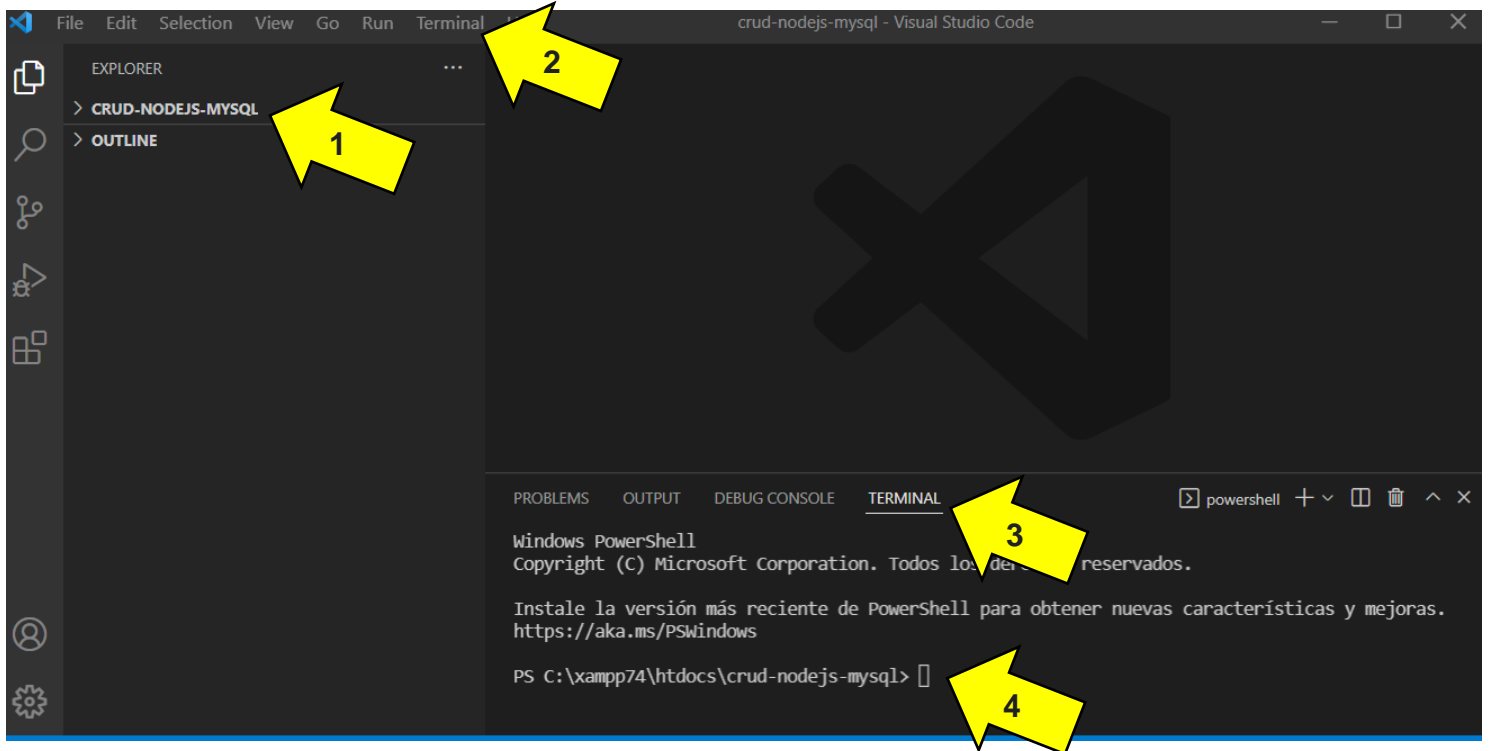
Antes de empezar, será necesario instalar node. Para ello vamos a descargar el instalador que está en las carpetas del aula. La carpeta se llama: NodeJS y Express - Instalador Servidor Node. Una vez descargado, ejecutá el instalador.



¿Comenzamos?

- 1) Lo primero que vamos a hacer es abrir el **Explorador de Archivos de Windows** y dirigirnos a la carpeta donde tenemos el instalado **Xampp** e ingresaremos en la carpeta **htdocs**.
Por ejemplo: c:\xampp\htdocs
- 2) Una vez posicionados allí, crearemos una carpeta con el nombre **crud-nodejs-mysql**. Esta será la carpeta de nuestro proyecto.
- 3) Entramos en **Visual Studio Code**.

- 4) Iremos a la opción **File** y luego elegimos **Open Folder**. Buscamos y seleccionamos la carpeta **crud-nodejs-mysql** que acabamos de crear.
- 5) (Miren el siguiente gráfico) Una vez posicionados en la carpeta **crud-nodejs-mysql** (flecha 1) vamos a abrir una **Terminal** (flechas 2 y 3) dentro de **Visual Studio Code** y nos **posicionamos para escribir** (flecha 4):



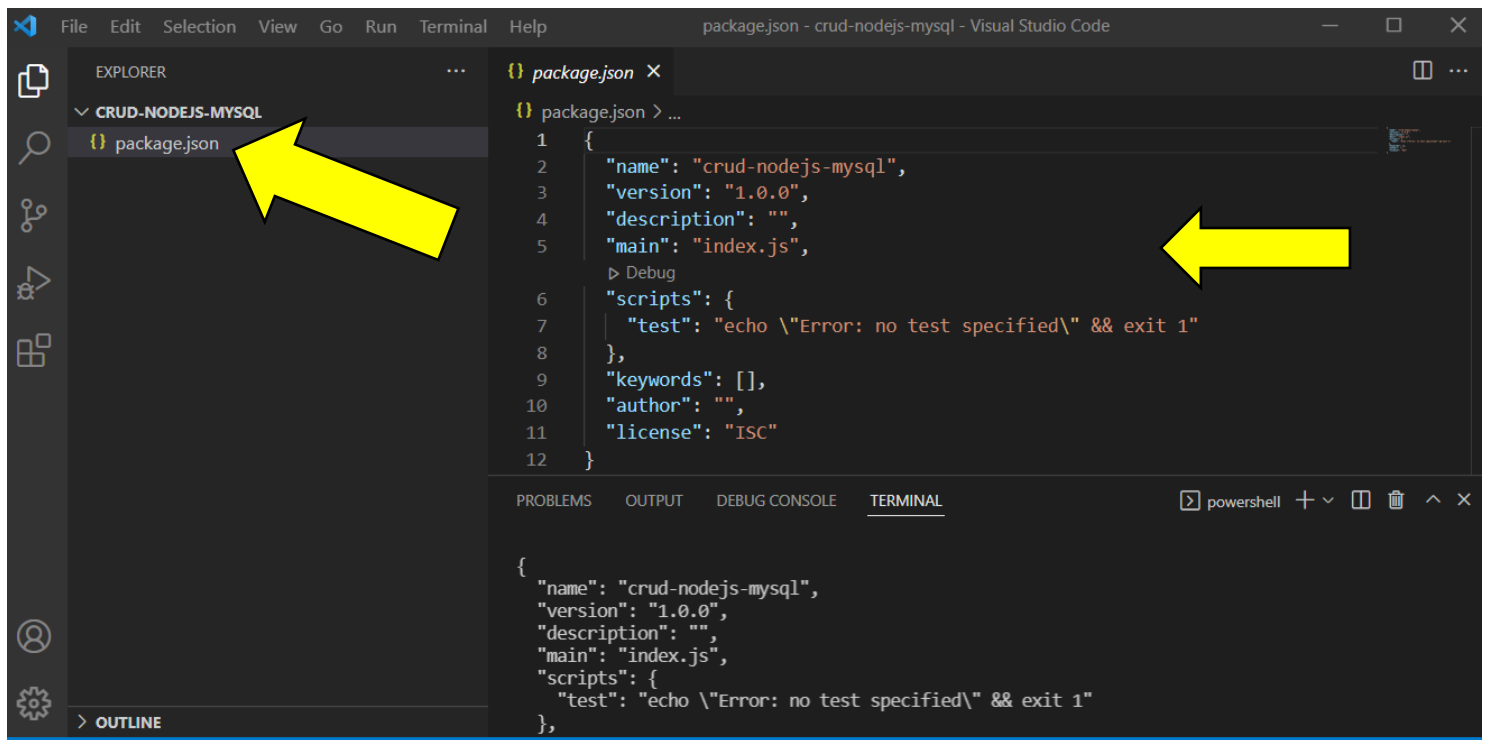
NOTA: Es importante verificar que estemos posicionados en la carpeta antes de continuar.

package.json

Como vimos en el Trabajo Práctico, los archivos **json** contienen metadatos. El archivo **json** de nuestra aplicación nos permitirá administrar ciertas dependencias que utilizaremos, entre ellas, acceder a la librería para conectarnos con la base de datos **MySQL**.

- 6) Vamos a ejecutar un comando que va a crear nuestro archivo **package.json** en forma automática escribiendo:

npm init -y



Como podemos observar, este comando creó automáticamente el archivo json con información de nuestro proyecto: metadatos, dependencias y módulos que vamos a utilizar.

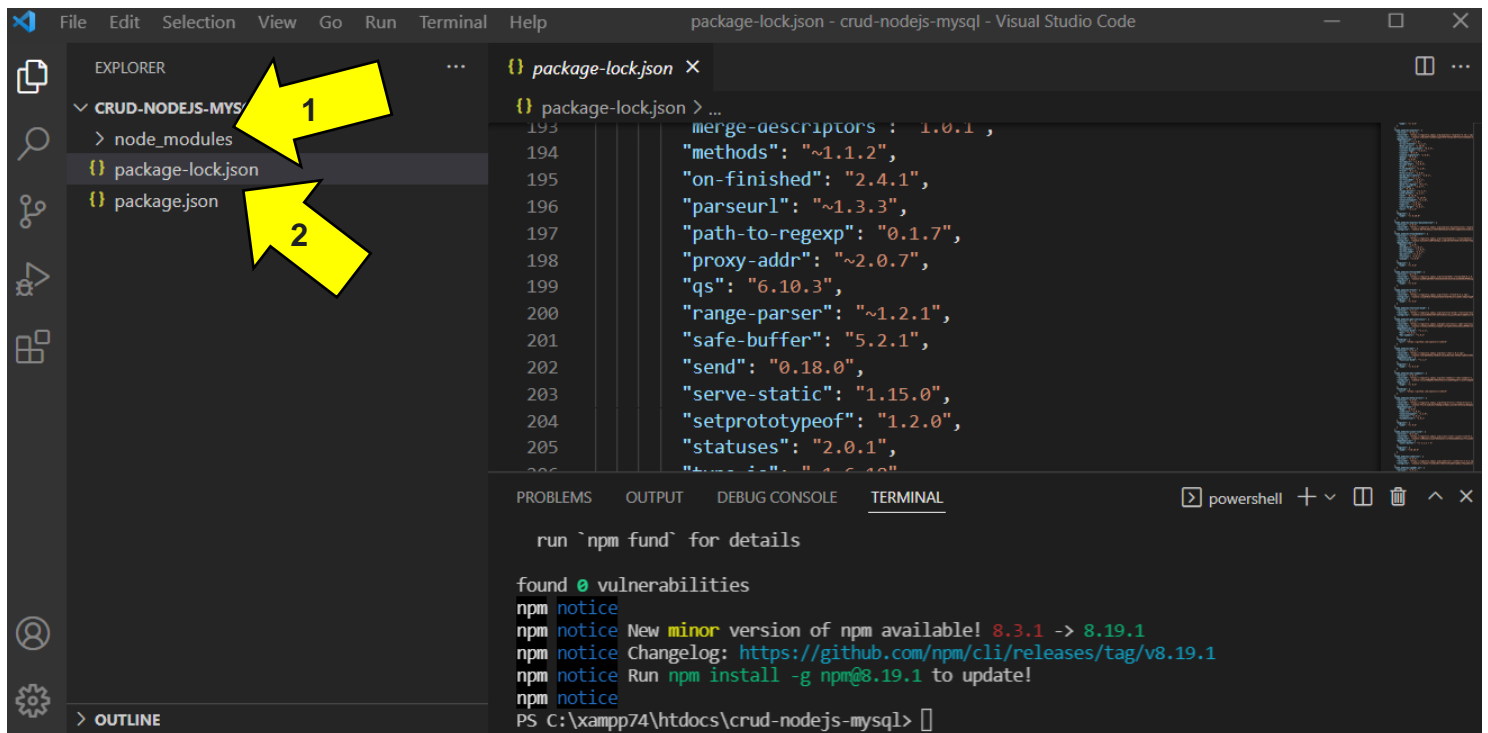
- 7) **JavaScript** del lado del cliente, no puede conectarse directamente a la base de datos **MySQL** del lado del servidor. Por lo tanto nos hace falta un **"puente"**. Vamos a instalar ahora el framework **express**, la librería **mysql** y el modulo **express-myconnection** (para conectar con la capa intermedia de express).

Veamos... ¿Qué hace cada uno?

Como dijimos, **Express** es el framework (marco de desarrollo) para node.js. El módulo **mysql** nos permitirá conectar con la base de datos y con el módulo **express-myconnection** vamos a interactuar con la capa del medio, teniendo a mano la conexión.

Vamos a instalar estos módulos escribiendo en la terminal el comando:

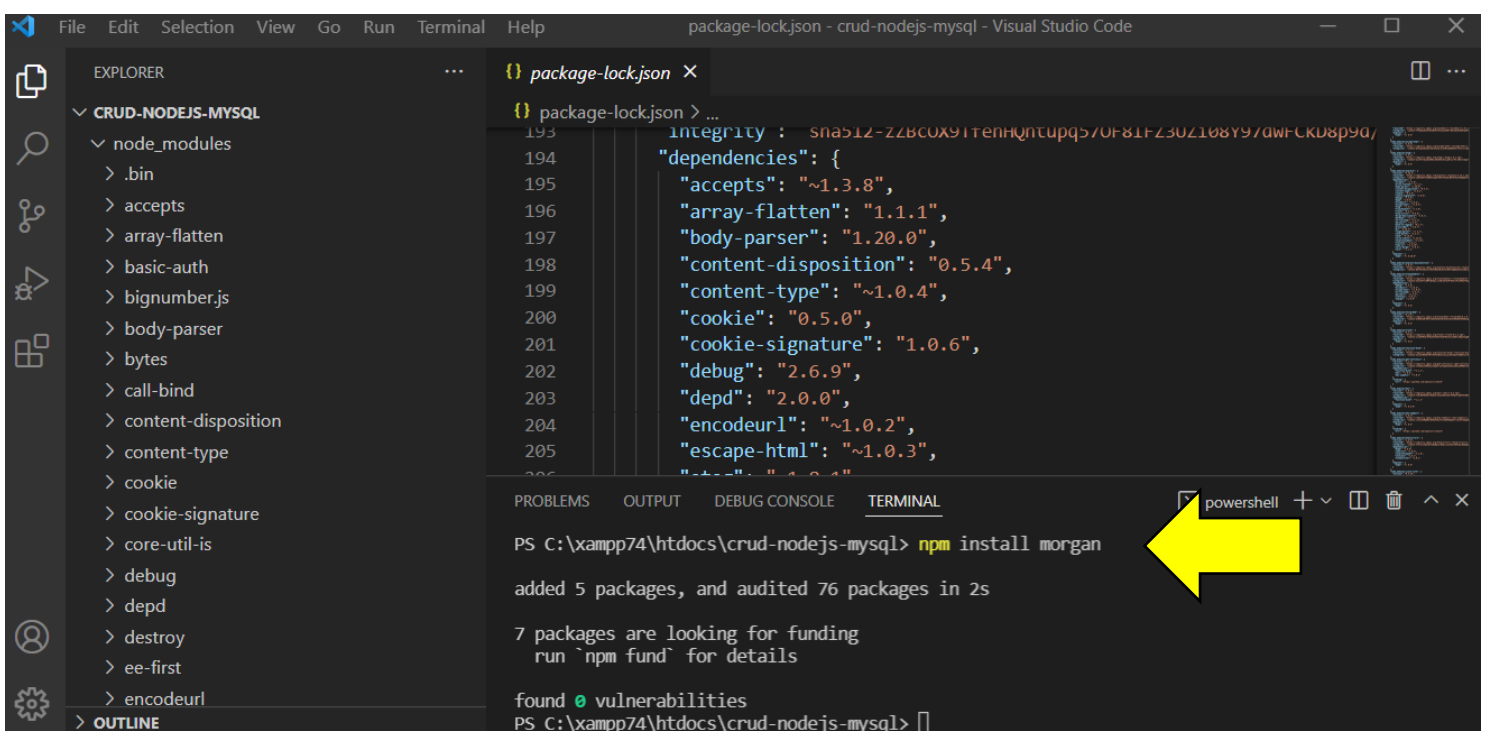
npm install express mysql express-myconnection



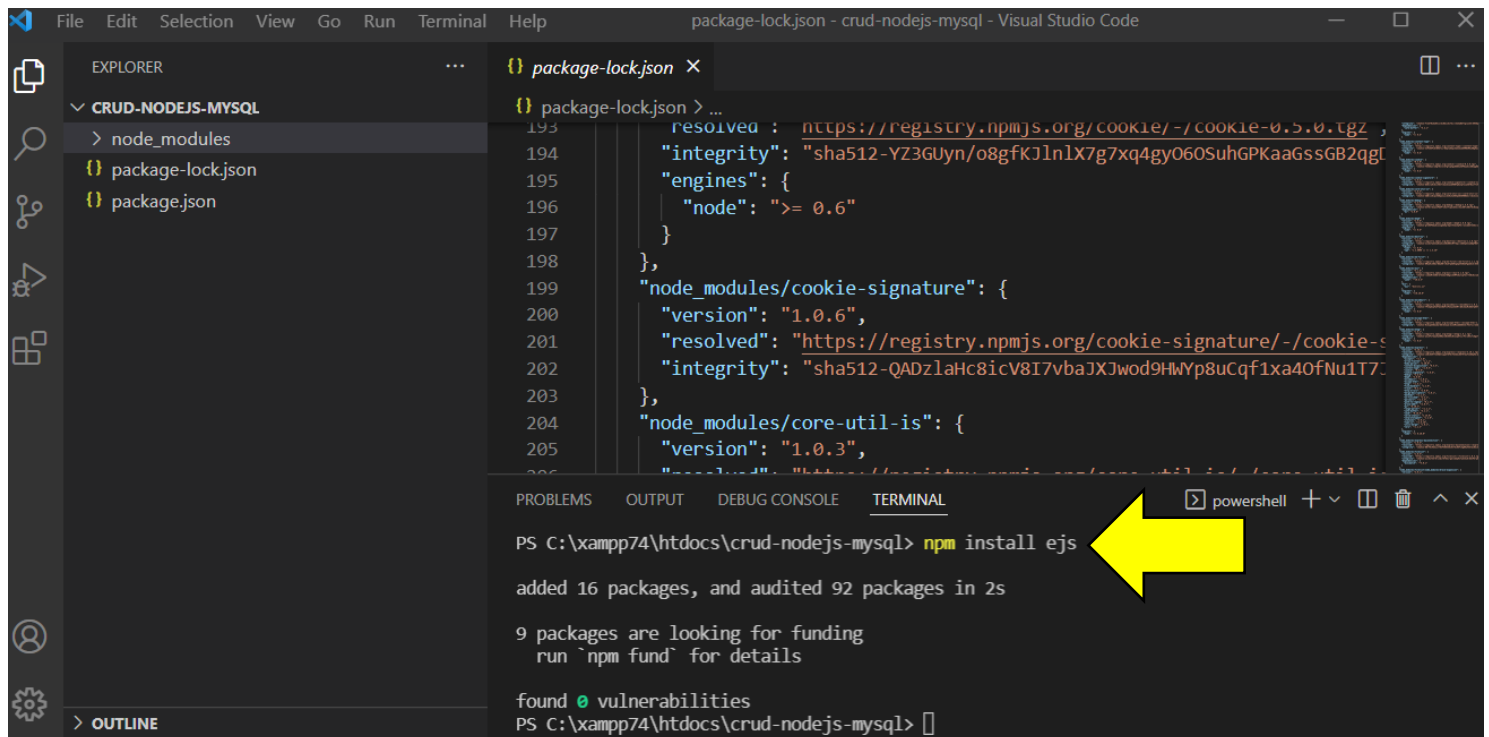
Al ejecutar este comando, se crearon automáticamente la carpeta **node_modules** (1) que va a contener todas las dependencias relacionadas a los paquetes que instalemos con Node y se agregan las dependencias en el archivo **package-lock.json** (2).

- 8) Aunque no es totalmente necesario, vamos a instalar otro módulo que nos va a permitir ver las peticiones que llegan al servidor. Este módulo se llama **morgan**.

npm install morgan



- 9) Vamos a instalar también el módulo **ejs** que es un **motor de plantillas**, que nos permite enviar html pero antes es procesado en el servidor. De esta manera podemos ejecutar en el servidor condiciones, bucles, etc y podemos ejecutar la lógica de programación antes de armar el html.



```
package-lock.json > ...
  "resolved": "https://registry.npmjs.org/cookie/-/cookie-0.5.0.tgz",
  "integrity": "sha512-YZ3GUyn/o8gfKJlNlX7g7xq4gyO60SuhGPKaaGssGB2qgT",
  "engines": {
    "node": ">= 0.6"
  }
},
"node_modules/cookie-signature": {
  "version": "1.0.6",
  "resolved": "https://registry.npmjs.org/cookie-signature/-/cookie-signature-1.0.6.tgz",
  "integrity": "sha512-QADzlaHc8icV8I7vbaJXJwod9HwYp8uCqf1xa40fNu1T7C",
  "engines": {
    "node": ">= 0.6"
  }
},
"node_modules/core-util-is": {
  "version": "1.0.3",
  "resolved": "https://registry.npmjs.org/core-util-is/-/core-util-is-1.0.3.tgz",
  "integrity": "sha512-9n1a72a207c77f7a8b8AC4F6CET0Ye6tG05cB36vP00Ac3eZgK4lC1+8X6gIIZquzttVhKtjAzdEiUjQ8A=="
},
...
}

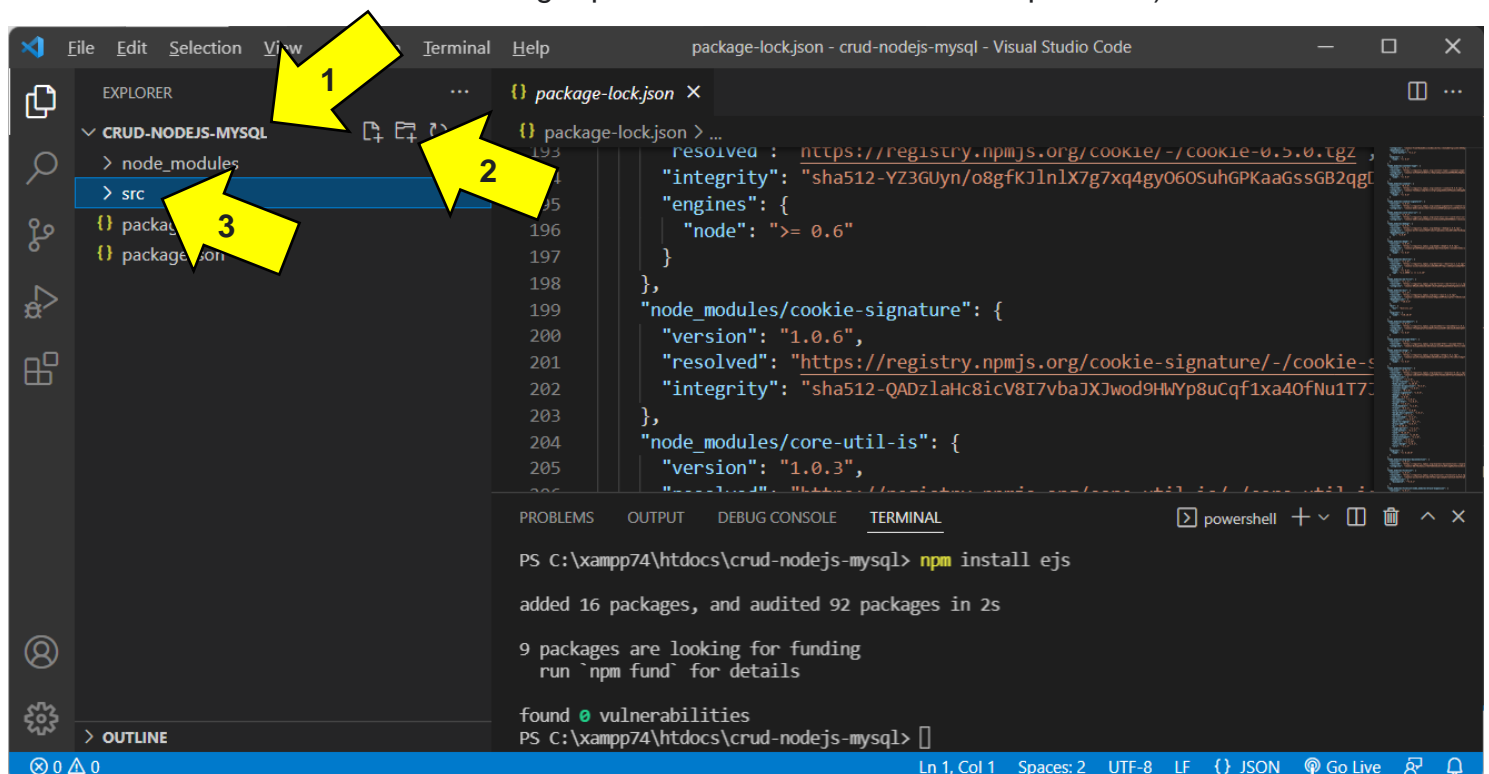
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\xampp74\htdocs\crud-nodejs-mysql> npm install ejs
added 16 packages, and audited 92 packages in 2s

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

found 0 vulnerabilities
PS C:\xampp74\htdocs\crud-nodejs-mysql>
```

Ya tenemos todos los módulos instalados. Empecemos a programar!

- 1) Lo primero que vamos a hacer es: estando parados en la carpeta **crud-nodejs-mysql** (1) vamos a hacer click en el botón "New Folder" (2) para crear una nueva carpeta que llamaremos **src** (3) (significa **SOURCE = FUENTE**). (Los programadores la nombramos así para saber que ahí adentro va a estar todo el código que nos hace falta en nuestra aplicación).



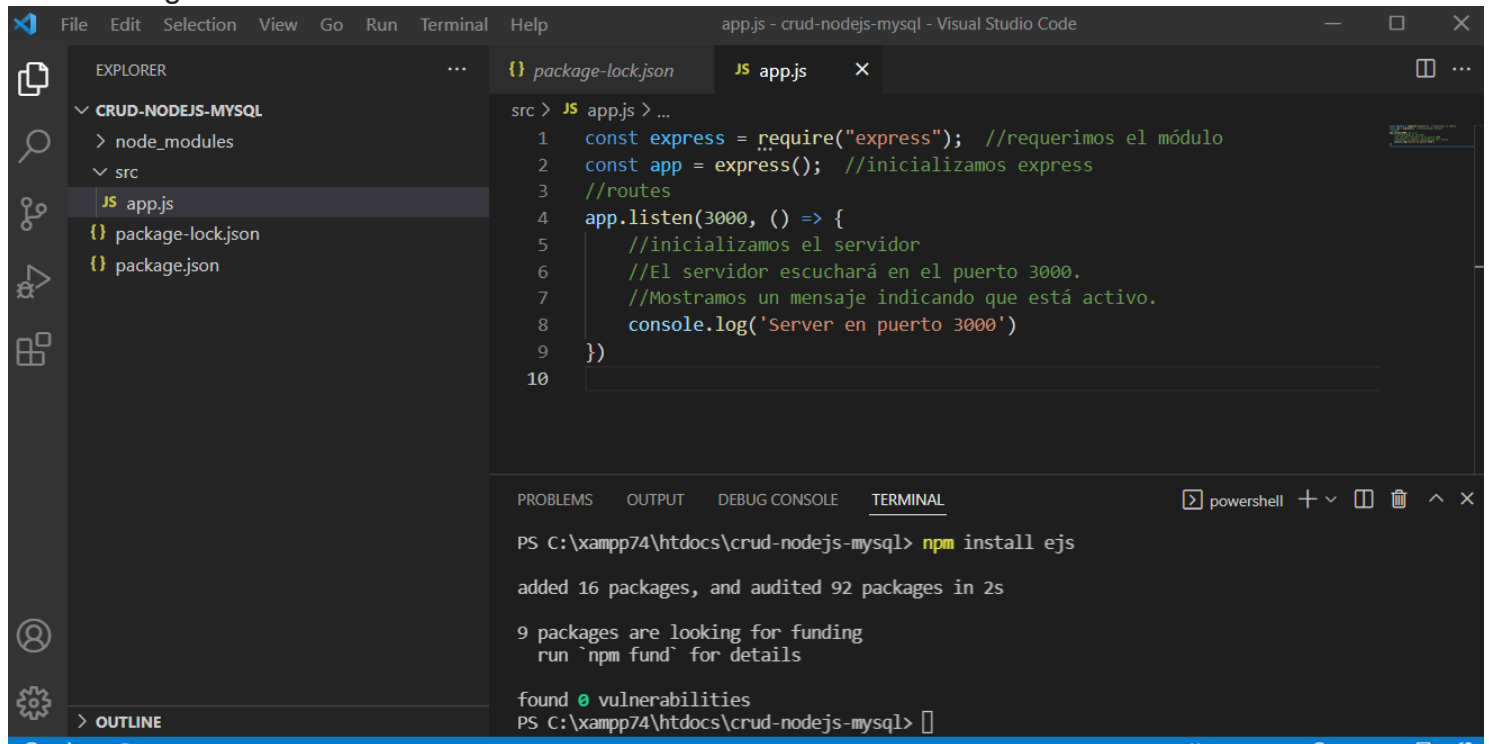
```
package-lock.json > ...
  "resolved": "https://registry.npmjs.org/cookie/-/cookie-0.5.0.tgz",
  "integrity": "sha512-YZ3GUyn/o8gfKJlNlX7g7xq4gyO60SuhGPKaaGssGB2qgT",
  "engines": {
    "node": ">= 0.6"
  }
},
"node_modules/cookie-signature": {
  "version": "1.0.6",
  "resolved": "https://registry.npmjs.org/cookie-signature/-/cookie-signature-1.0.6.tgz",
  "integrity": "sha512-QADzlaHc8icV8I7vbaJXJwod9HwYp8uCqf1xa40fNu1T7C",
  "engines": {
    "node": ">= 0.6"
  }
},
"node_modules/core-util-is": {
  "version": "1.0.3",
  "resolved": "https://registry.npmjs.org/core-util-is/-/core-util-is-1.0.3.tgz",
  "integrity": "sha512-9n1a72a207c77f7a8b8AC4F6CET0Ye6tG05cB36vP00Ac3eZgK4lC1+8X6gIIZquzttVhKtjAzdEiUjQ8A=="
},
...
}

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\xampp74\htdocs\crud-nodejs-mysql> npm install ejs
added 16 packages, and audited 92 packages in 2s

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

found 0 vulnerabilities
PS C:\xampp74\htdocs\crud-nodejs-mysql>
```

- 2) Dentro de la carpeta **src**, vamos a crear un **archivo** nuevo que se llame **app.js** que será el encargado de manejar el servidor de la capa intermedia **Express**. Escribimos el siguiente código:

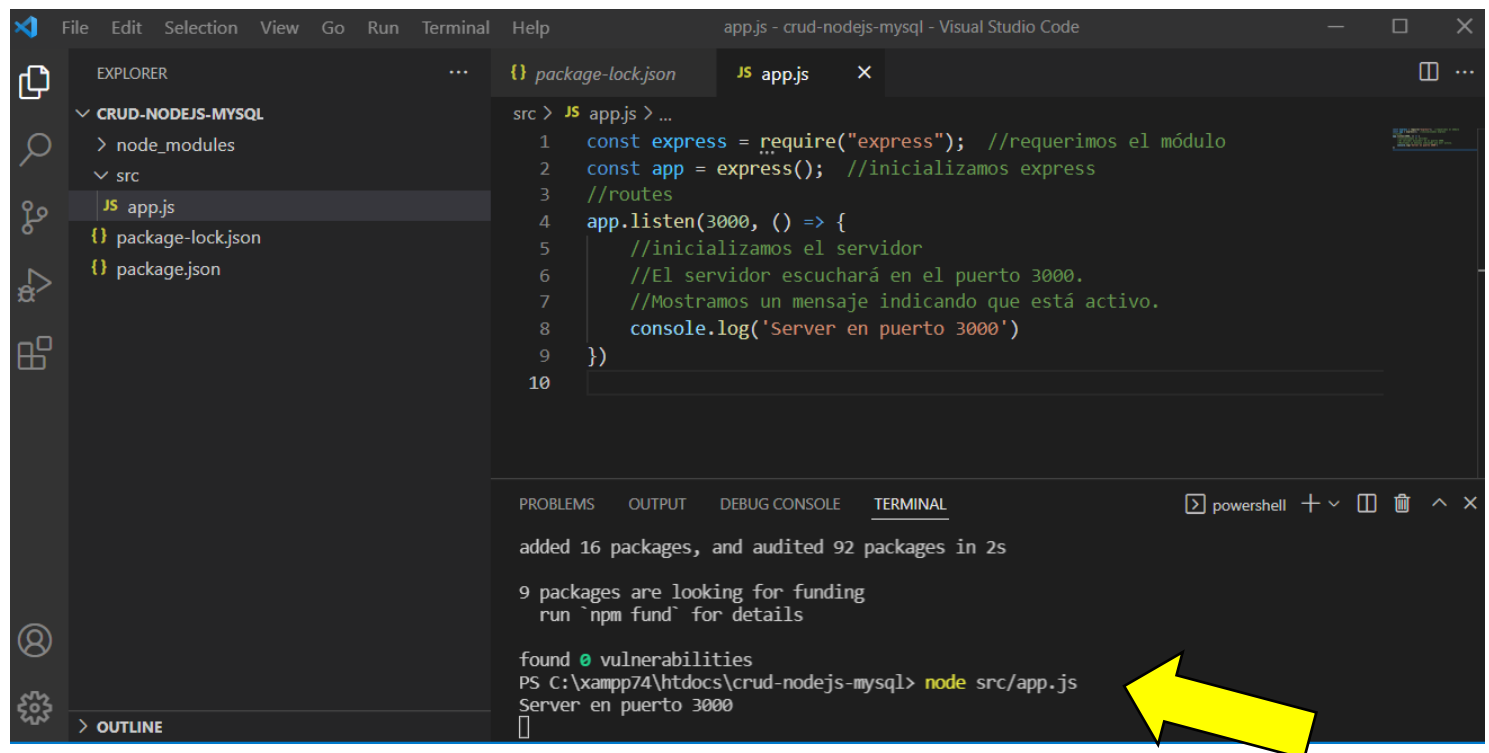


The screenshot shows the Visual Studio Code interface. The Explorer on the left shows the project structure: `CRUD-NODEJS-MYSQL` with subfolders `node_modules` and `src`. Inside `src`, there is a new file `app.js` and existing files `package-lock.json` and `package.json`. The main editor shows the content of `app.js`:

```
src > JS app.js > ...
1  const express = require("express"); //requerimos el módulo
2  const app = express(); //inicializamos express
3  //routes
4  app.listen(3000, () => {
5    //inicializamos el servidor
6    //El servidor escuchará en el puerto 3000.
7    //Mostramos un mensaje indicando que está activo.
8    console.log('Server en puerto 3000')
9  })
10
```

The Terminal at the bottom shows the command `npm install ejs` being executed, with output indicating that 16 packages were added and 92 packages were audited in 2 seconds. It also shows that 9 packages are looking for funding and that no vulnerabilities were found.

- 3) Vamos a **arrancar el servidor de express** ejecutando en la **terminal** el siguiente comando:
node src/app.js



The screenshot shows the Visual Studio Code interface with the same project structure as before. The main editor still shows the content of `app.js`. The Terminal at the bottom shows the command `node src/app.js` being executed. The output shows the message `Server en puerto 3000`, indicating that the server is running. A large yellow arrow points to this message in the terminal.

...y nos mostrará el mensaje que indica que el servidor **express** ya está escuchando en el puerto 3000.



Hasta acá la primera parte de nuestra aplicación. Asegurémonos de tener todo funcionando para poder continuar con la Parte 2.