

**PROFESSIONAL WEBMASTER**

Módulo: 2

Node JS



Introducción (2).png

|  |  |
| --- | --- |
|  |  |

[**Usando Bases de datos 4**](#_Toc34806359)

[**Consultas 4**](#_Toc34806360)

[**SESIONES 5**](#_Toc34806361)

[**Usando middleware 6**](#_Toc34806362)

# 

# Usando Bases de datos

Las apps de Express pueden usar cualquier mecanismo de bases de datos suportadas por Node (Express en sí mismo no define ningúna conducta/requerimiento especifico adicional para administración de bases de datos). Hay muchas opciones, incluyendo PostgreSQL, **MySQL**, Redis, SQLite, MongoDB, etc.

La base de datos por si misma puede ser instalada localmente o en un servidor local. (**wamp**) En su código Express requiere el manejador, conectarse a la base de datos, y entonces **ejecutar operaciones crear, leer, actualizar, y borrar**

Antes de continuar es importante saber, que debemos tener el servicio de MySQL instalado en nuestro sistema, y obviamente encendido. (**WAMP**)  
  
**Instalar varios módulos:** mysql, útil, md5

* **mysql:** base de datos
* **util:** El módulo util está diseñado principalmente para soportar las necesidades de las API internas de Node.js.

Sin embargo, muchas de las utilidades también son útiles para los desarrolladores de aplicaciones y módulos**.**

* **md5:** mandar mensaje encriptados.

Conectar Node.js con MySQL  
Crea un script de Node.js. Lo primero que hacemos en este ejemplo es crear un objeto de conexión a hacia la base de datos MySQL.   
Utilizamos 4 parametros: hosting, nombre de la tabla, usuario y contraseña.  
  
*var mysql = require('mysql');*

*var conexion= mysql.createPool({*

*host : 'localhost',*

*database : 'empleados',*

*user : 'USUARIO',*

*password : 'PASS',*

*});*

# ****Consultas****

Empezaremos realizando un **INSERT** en nuestra tabla:

*var query = connection.query('INSERT INTO personaje(nombre, apellido, biografia) VALUES(?, ?, ?)', ['Homero', 'Simpson', 'Esposo de Marge y padre de Bart, Lisa y Maggie.'*

*);*

Este mismo proceso sirve tanto para **UPDATE (modificar)** como **DELETE (eliminar)**.

También podemos hacer un **SELECT**: recuperar un o varios registro.

*var query = connection.query('SELECT nombre, apellido, biografia FROM personaje WHERE personaje\_id = ?', [1]);*

# SESIONES

Las sesiones sirven para guardar información del usuario en nuestra aplicación entre página y página.

Usaremos las dependencias: **express-session.**  
  
Es necesario en Express para el uso de sesiones configurar una clave secreta para así proteger nuestros datos:

*{secret: 'abcd1234'}*

Luego el uso de sesiones es muy sencillo, para crear una sesión debemos usar un objeto llamado **session**. Por ejemplo:

*app.get('/ejemplo', function(req, res){*

*if(req.session.nombre){*

*res.send('Hola ' + req.session.nombre);*

*}else{*

*var nombre = 'Tito';*

*req.session.nombre = nombre;*

res.send('Hola usuario desconocido. De ahora en adelante te llamaremos ' + nombre);

}

});

if(req.session.nombre)

res.send('Hola usuario desconocido. De ahora en adelante te llamaremos ' + nombre);

# Usando middleware

El middleware de Express, puede usarse también para **añadir funcionalidades** para la gestión de cookies, sesiones y usuarios, mediante el uso de parámetros, en los métodos POST/GET.  Puede utilizarse además cualquier sistema de trabajo con bases de datos, que sea soportado por Node.

**