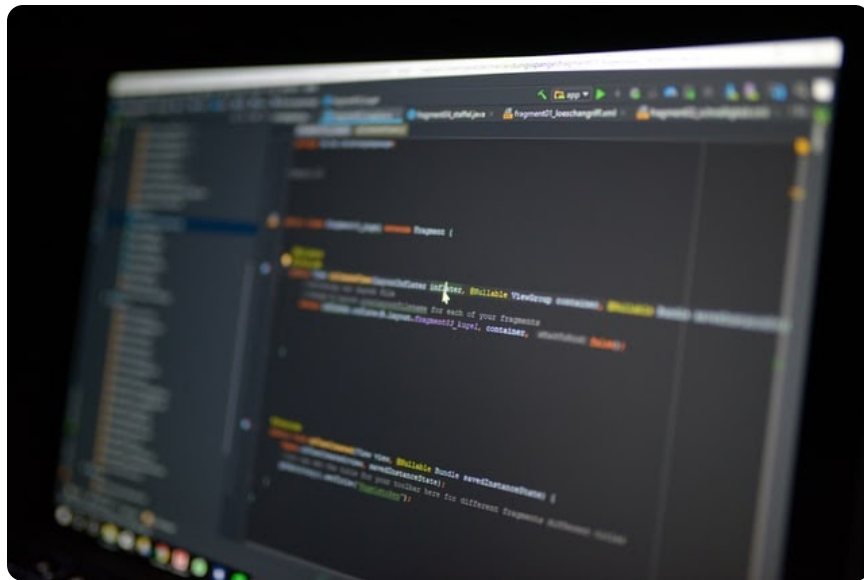


 Grabación Presentación

# La clave está en la comunidad NodeJS

"El software libre construye una sociedad mejor." —[Richard Stallman](#). Desarrollador de software.



Así como creamos nuestros propios módulos con NodeJS en la bitácora anterior, al momento de programar también puedes aprovechar los módulos creados por otros/as que traten de estructuras genéricas. Ahorrarás tiempo y podrás utilizarlo para acceder a un código de mejor calidad.

Cuando utilizamos librerías, existen diferentes formas de importarlas. Una opción es hacerlo directo desde la web, copiando y pegando el código en nuestro proyecto. Pero ¿te imaginas si tuvieras que buscar los archivos uno por uno en cada página, de cada librería y descargarlos? ¡Por

Además de estas dos formas, Node cuenta con una más ordenada: **NPM (Node Package Manager)**. Es un **gestor de paquetes que viene instalado junto con Node y se utiliza desde la consola**.

Un **paquete** es una versión encapsulada de un módulo de librería. Todo el contenido en NPM tiene el beneficio de ser *open source*, es decir, su código fuente original está disponible de forma gratuita y puede ser modificado, redistribuido por la comunidad y hasta puedes modificar el paquete y enviarle tus mejoras al desarrollador/a para que lo actualice. Puedes crear tus propios paquetes y publicarlos en este espacio que es como una gran biblioteca donde los/as developers pueden compartir, consultar y tomar elementos para usar e instalar en sus proyectos / trabajos. La gran ventaja es que contamos con un gestor que hace esto con tan solo con una línea de código. Accediendo a la [versión web](#) puedes encontrar disponibles todos los paquetes y también su documentación, su versión de descargas semanales e incluso el acceso al repositorio.

NPM también cuenta con librerías o paquetes, y entre las más populares, se encuentran:

- [Angular](#): framework para crear aplicaciones web.
- [Reactjs](#): librería para crear interfaces de usuario.
- [Vue.js](#): framework MVVM para crear interfaces de usuario.
- [Expressjs](#): framework para crear APIs muy fácilmente.
- [Lodash](#): set de utilidades JavaScript para el manejo de objetos, arrays, etc

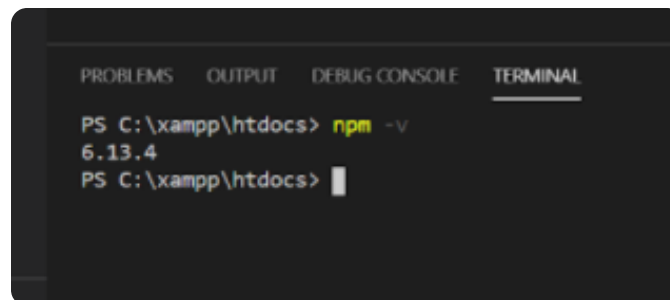
Particularmente, nos enfocaremos en **ExpressJS** a lo largo del Bloque para crear servidores con NodeJS. En esta bitácora encontrarás cómo reutilizar librerías de otros developers, cómo instalarlas y también cómo inicializar tus proyectos con **NPM**.

## ¿Cómo crear proyectos con NodeJS?

## #1. npm [acción] [paquete]

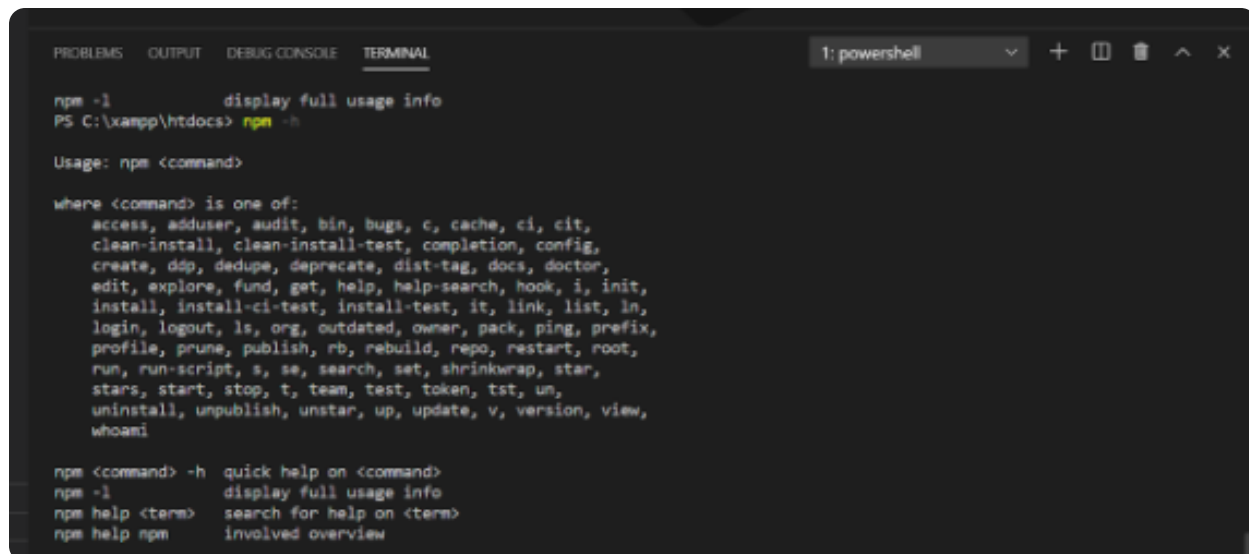
Mediante el comando **npm** podemos acceder a la gran cantidad de librerías *open source* listas para usar.

1. Primero, chequea el correcto funcionamiento de npm mirando su versión: `npm -v`



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\xampp\htdocs> npm -v
6.13.4
PS C:\xampp\htdocs> |
```

2. Para recibir ayuda tipea: `npm -h`



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: powershell
npm -h display full usage info
PS C:\xampp\htdocs> npm -h

Usage: npm <command>

where <command> is one of:
  access, adduser, audit, bin, bugs, c, cache, ci, cit,
  clean-install, clean-install-test, completion, config,
  create, ddp, dedupe, deprecate, dist-tag, docs, doctor,
  edit, explore, fund, get, help, help-search, hook, i, init,
  install, install-ci-test, install-test, it, link, list, ln,
  login, logout, ls, org, outdated, owner, pack, ping, prefix,
  profile, prune, publish, rb, rebuild, repo, restart, root,
  run, run-script, s, se, search, set, shrinkwrap, star,
  stars, start, stop, t, team, test, token, tst, un,
  uninstall, unpublish, unstar, up, update, v, version, view,
  whoami

npm <command> -h quick help on <command>
npm -l display full usage info
npm help <term> search for help on <term>
npm help npm involved overview
```

3. Para instalar un paquete: `npm i [nombre del paquete]`

```
PS C:\xampp\htdocs\acamica_npm> npm i random-quotes
npm WARN acamica_npm@1.0.0 No description
npm WARN acamica_npm@1.0.0 No repository field.

+ random-quotes@1.3.0
updated 1 package and audited 3 packages in 0.454s
found 0 vulnerabilities

PS C:\xampp\htdocs\acamica_npm> █
```

4. Para desinstalar un paquete: `npm uninstall [nombre del paquete]`

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\xampp\htdocs\acamica_npm> npm uninstall random-quotes
npm WARN acamica_npm@1.0.0 No description
npm WARN acamica_npm@1.0.0 No repository field.

removed 3 packages in 0.806s
found 0 vulnerabilities

PS C:\xampp\htdocs\acamica_npm> █
```

## #2. Package Json

**Package Json** es un archivo de configuración, necesario para que NPM guarde toda la información de nuestros proyectos y se pueda instalar en otras computadoras.

Si planeas publicar el proyecto en NPM tienes que considerar los campos de nombre y versión. Según la [página de documentación](#) de NPM, el nombre y la versión juntos forman un identificador completamente únicos. Los cambios en el paquete deberían venir junto con cambios en la versión. Si no planeas publicarlo, los campos de nombre y versión son opcionales.

Para **nombrarlo** existen algunas reglas:

- Los paquetes nuevos no deben tener letras mayúsculas.
- Evita caracteres que no sean seguros para URL.

Aquí van 4 consejos que te resultarán útiles:

- Usa diferentes nombres para un módulo y un nodo central.
- Evita el nombre "js" o "nodo" en el nombre. Ya que estás escribiendo un archivo `package.json`, y puede especificar el motor utilizando el campo "motores".
- El nombre probablemente se pasará como un argumento que requiere (). Por eso, debe ser algo corto, pero también razonablemente descriptivo.
- Puedes verificar el registro npm para ver si ya existe ese nombre en <https://www.npmjs.com/>

Si planeas publicar tu versión, asegúrate de que pueda analizarse mediante `node-semver`.  
¡Instálalo vía NPM!

Ten en cuenta que cuando instalamos cualquier librería en nuestro proyecto, se genera un archivo llamado `package.json`: ahí estarán listadas todas las **dependencias** de tu proyecto. Como su nombre lo indica, tu proyecto va a depender de las librerías allí listadas para funcionar correctamente. En este código podrías indicar una versión en particular para evitar, si en algún momento se modifica, que se te rompa el proyecto. También puedes indicarle que siempre se actualice a la última versión, que incluirá mejoras en sus funcionalidades.

Toda la magia la realiza NPM, ¡no tenemos que preocuparnos por actualizar el archivo `package.json`!

### #3. Node modules

`node_modules` es una carpeta que se genera cuando instalamos un paquete vía NPM, que a su vez contiene carpetas con el código de cada uno de los paquetes que instalamos. Funciona como si

lo deja en la carpeta `node_modules` disponible para que utilizar en tu aplicación.

Cuando necesitamos mover la aplicación a otro entorno —como por ejemplo, un servidor de stage o de producción, o subir una versión estable a nuestro repositorio de GIT— no podemos subir la carpeta `node_modules` por la gran cantidad de archivos y peso que tiene. ¡Para esto tenemos **NPM Install!** Este comando es el encargado de leer todas las dependencias del archivo `package.json` e instalar todos los módulos necesarios para que la aplicación corra de igual manera que en nuestro entorno de desarrollo.

Una buena práctica de developer consiste en agregar siempre en `.gitignore` todos los archivos y carpetas que no son necesarios subir al repositorio, como por ejemplo, la carpeta `node_modules`.

## Variables de entorno

Las variables de entorno se utilizan para definir parámetros básicos en la configuración de tu aplicación y podrá tener diferentes valores de acuerdo al entorno donde esté corriendo tu aplicación.

Uno de los ejemplos más comunes es la conexión a una base de datos. Mientras desarrolles tu aplicación estarás trabajando de manera local con tu base de datos, pero al subir tu aplicación a producción a un servidor para que quede accesible para todo el mundo, la conexión a la base de datos será otra.

Entonces, ¿cómo puede nuestro código fuente saber si debe conectarse a una base de datos u a otra? ¡Nuestro código nunca lo sabrá! Solo se conectará a una base de datos y cambiarán las credenciales de conexión de acuerdo a dónde se esté ejecutando la aplicación.

En el próximo encuentro aprenderás cómo trabajar con variables de entorno y cuáles son los casos de usos más comunes.

Como te contamos, existe una gran cantidad de librerías disponibles para utilizar en tus proyectos. A medida que vayas conociendo más cómo funciona el backend, considera empezar a aportar las tuyas a la comunidad de developers de una forma sencilla a partir de la herramienta que tiene NodeJS: NPM.

## ¡Prepárate para el próximo encuentro!



### Profundiza

Te invitamos a conocer más sobre el tema de esta bitácora.



### Herramientas

Programas necesarios para facilitar tu experiencia.



### Challenge

Te proponemos el siguiente desafío, ¿te animas?