Índice

[**Acerca de NPM, paquetes y módulos** 1](#_Toc93767630)

[**¿Qué es npm?** 1](#_Toc93767631)

[**Node en Window** 2](#_Toc93767632)

[**Iniciar un proyecto** 3](#_Toc93767633)

[**Crear un proyecto** 3](#_Toc93767634)

[**Segunda opción (para hacer package rápido)** 4](#_Toc93767635)

[**Tercera opción (configuramos algunos elementos)** 4](#_Toc93767636)

[**Instalación de dependencias** 5](#_Toc93767637)

[**Algunos comandos** 5](#_Toc93767638)

[**Instalación de dependencias con force** 8](#_Toc93767639)

[**Actualizar y eliminar paquetes** 8](#_Toc93767640)

[**Actualizar paquetes** 8](#_Toc93767641)

[**Eliminar paquetes** 9](#_Toc93767642)

[**Package lock y el uso los símbolos ^ y ~** 9](#_Toc93767643)

[**Versionado Semántico** 9](#_Toc93767644)

[**Ejecutar tareas** 11](#_Toc93767645)

[**Solución de errores** 11](#_Toc93767646)

[**Borrar la carpeta de node\_modules** 12](#_Toc93767647)

[**Seguridad** 12](#_Toc93767648)

[**Crear un paquete para NPM** 13](#_Toc93767649)

[**Publicar un paquete en NPM** 15](#_Toc93767650)

[**Eliminar paquetes de más de 72hs** 16](#_Toc93767651)

[**Paquetes privados** 16](#_Toc93767652)

# **Acerca de NPM, paquetes y módulos**

## **¿Qué es npm?**

Sus siglas en inglés **Node Package Manager**, es un gestor de paquetes, el cual hace más fáciles nuestras vidas al momento de trabajar con **Node**, ya que gracias a él podremos tener **cualquier librería disponible**, **npm** nos ayudará a administrar nuestros **módulos**, distribuir **paquetes** y agregar **dependencias** de una manera sencilla.

Los paquetes más conocidos son:

* [**npm**](https://www.npmjs.com/): Es el más popular de JS.
* [**yarn**](https://yarnpkg.com/): Es una alternativa a npm creada por facebook, cuya mayor ventaja es la velocidad de descarga de los paquetes.

****

# **Node en Window**

Vamos a la página de [node.js](https://nodejs.org/es/) y seleccionar la version current, las cuales trae las ultimas actualizaciones.

**Nota**: la version LTS es la version estable.

Luego de descargar debemos instalarlo como otro cualquier programa de window.

Abrimos la consola cmd y con el comando:



Podemos verificar si se instaló correctamente y la version de node.

Para comprobar si existen versiones recientes de **npm** utilizamos el comando:



Este comando verifica si existe una versión nueva y la instalara.

# **Iniciar un proyecto**

Primero abrimos nuestra terminal y nos posicionamos donde guardamos nuestros archivos.

Podemos crear una carpeta para nuestro proyecto con el comando mkdir **project\_name** nos movemos dentro de la carpeta con **cd** **project\_name** ya dentro de la carpeta podemos iniciar:

## **Crear un proyecto**

* **git init**: Para tener el control de cambios y versiones en nuestro proyecto.
* **npm init**: Con este comando vamos a crear nuestro archivo de configuración del proyecto, el **package.json**, luego nos saldrá lo siguiente:
  + **package name**: el nombre de nuestro proyecto, generalmente es el nombre de la carpeta.
  + **version**: Versión con la que iniciaremos el proyecto, generalmente aparece 1.0.0
  + **description**: Descripcion breve del proyecto.
  + **entry point**: Punto de acceso a nuestro proyecto.
  + **test command**: Comando para gestionar los test.
  + **git repository**: Repositorio de github u otro servicio.
  + **keywords**: Palabras claves del proyecto.
  + **author**: Nombre del autor y **<correo>**
  + **license**: Licencia que tendrá el proyecto.

## **Segunda opción (para hacer package rápido)**

Escribimos **npm init -y** y esto generará un **package.json** vacío para que lo configuremos más adelante.

## **Tercera opción (configuramos algunos elementos)**

Escribimos en nuestra terminal

* **npm set init.author.email "your@email"**
* **npm set init.author.name "your name"**
* **npm set init.license "license name"**
* **npm init -y**

Y se creará un **package.json** con los datos que seleccionamos.

# **Instalación de dependencias**

**node\_modules**: Aquí es donde se van a instalar los módulos que tú estás agregando a tu proyecto. Esta carpeta es importante para que tu proyecto funcione, pero no debe de ser enviada a ningún repositorio y debemos ignorarla apenas se crea.

Para ignorarla se crea un archivo **.gitignore** y escribimos: **node\_modules/**

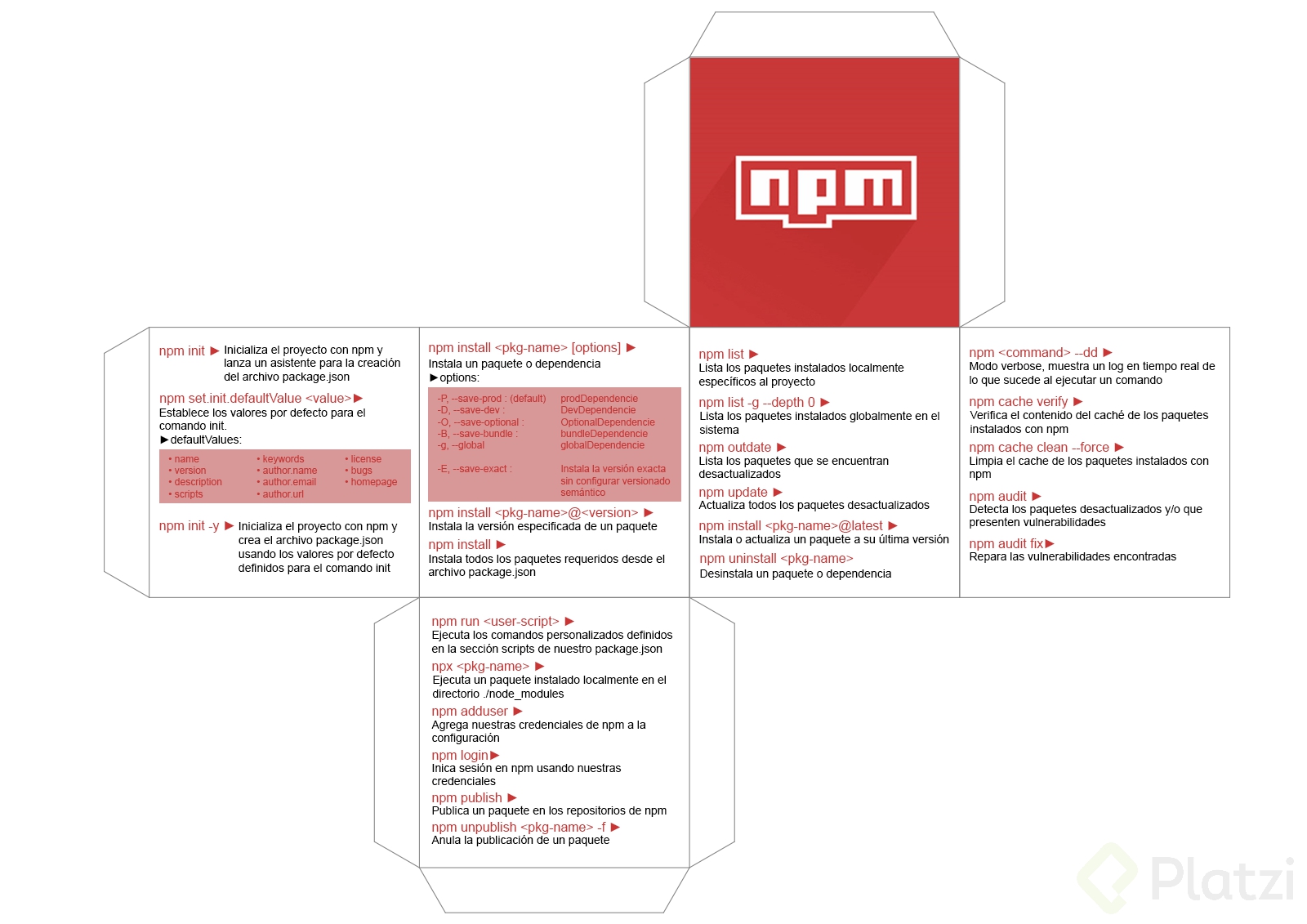
Versión resumida para instalar:



## **Algunos comandos**

* **npm install moment --save**: Paquete que se encarga de manejar las fechas en JavaScript.
* **--save**: El documento que vamos a instalar dentro del proyecto es necesario para vivir en producción.
* **-dev**: El documento que vamos a instalar solo es necesario en nuestro entorno local o en el entorno de desarrollo.
* **npm install date-fns**: Al igual que **moment** se encarga de manejar fechas y datos.
* **npm i**: Abreviación de **npm install**.
* **-D**: Abreviación para **--save-dev**.
* **npm install -g name\_package**: Instala paquetes de forma global.
* **npm list -g –depth 0**: Nos permite listar todos los paquetes instalados de forma global.
* **npm install name\_package -O**: Nos permite instalar un paquete de forma opcional.
* **npm fund**: Nos permite ver los financiamientos de un paquete, es decir, nos muestra la URL para apoyar a los desarrolladores.
* **npm install name\_package --dry-run**: Cuando queremos ver el **output** de una dependencia sin proceder con su instalación, es decir, realiza una simulación de una instalación real.
* **npm install name\_package --force** / **npm install webpack -f**: Instalar la última versión y desde los servidores de npm.
* **npm install json-server@0.15.0**: Instalar una versión exacta de un paquete.
* **npm install:** Revisa el archivo de **package.json** he instala cada una de las dependencias que están dentro de él, no importa si están en desarrollo, en opcionales o las dependencias que requiera el proyecto. Instala todo lo que se encuentra dentro del archivo.
* **npm cache clean --force:** Elimina el cache existente.
* **npm cache verify:** Verifica el cache.





# **Instalación de dependencias con force**

**NPM** decidió agregar un nuevo comando:

* **npm fund**: Brinda más visibilidad a los usuarios de npm sobre qué dependencias buscan activamente formas de financiar su trabajo.

Este comando recupera información sobre cómo financiar las dependencias de un proyecto determinado.

Si no se proporciona el nombre del paquete, enumerará todas las dependencias que buscan financiación en una estructura de árbol en la que se enumeran el tipo de financiación y la URL a visitar.

Si se proporciona un nombre de paquete, entonces intenta abrir su url de financiación utilizando el **--browser** parámetro de configuración.

# **Actualizar y eliminar paquetes**

## **Actualizar paquetes**

* **npm outdate**: Revisar que paquetes disponen de nuevas versiones, es decir, los que están desactualizados.
* **npm outdate --dd**: Para ver un output más detallado.
* **npm update**: Actualizar los paquetes que no están en la última versión.
* **npm install json-server@latest**: Actualizar un paquete especifico.

## **Eliminar paquetes**

* **npm uninstall name\_package**: Eliminar un paquete de **node\_modules** y del archivo **package.json**
* **npm uninstall name\_package --no-save**: Desinstalar un paquete de todo **node\_modules** pero no del archivo **package.json**

# **Package lock y el uso los símbolos ^ y ~**

* **^** (Acento circunflejo): Si mantenemos el caret dentro de la configuración de nuestro package estamos garantizando que cuando realicemos una actualización o tengamos un cambio que podamos realizar, vamos a hacer actualización de los **cambios menores** y de los **parches** o **bug fixes**. Para quedarnos en una sola versión eliminamos el caret.
* **~** (Virgulilla): Establece que vamos a recibir actualizaciones o cambios solamente de los cambios que son **parches** o **bug fixes**.

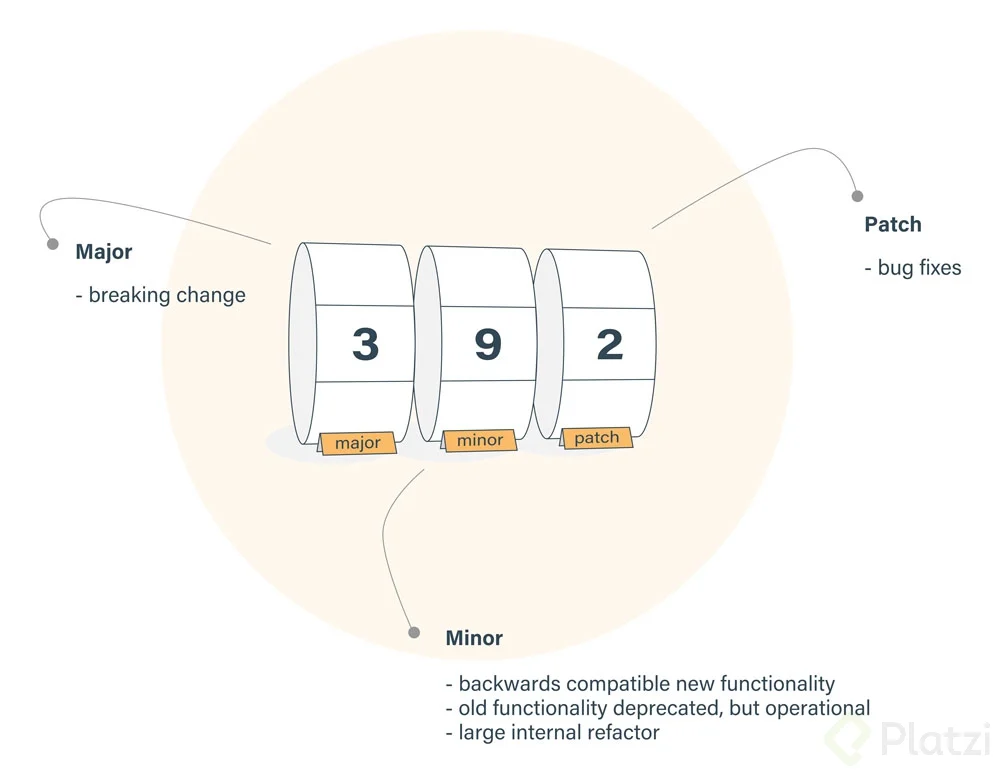
Además de esos símbolos, también tenemos:

* **<**: Versión menor a la indicada.
* **<=**: Versión menor o igual a la indicada.
* **>**: Versión mayor a la indicada.
* **>=**: Versión mayor o igual a la indicada.

## **Versionado Semántico**

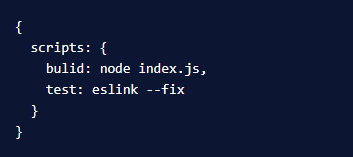
En el versionado tenemos 3 dígitos que significa:

* **Primer dígito**: Son cambios mayores.
* **Segundo dígito**: Añaden ciertas funcionalidades, pero no representan un gran paso para decir que esta es una versión nueva.
* **Tercer dígito**: Estos son patch, bug fixes o cambios menores.



# **Ejecutar tareas**

En nuestro archivo **package.json** encontramos una sección llamada scripts donde podemos nombrar comandos que apuntan a sentencias usadas a lo largo del proyecto. Estos comandos pueden recibir cualquier nombre y contener cualquier sentencia.



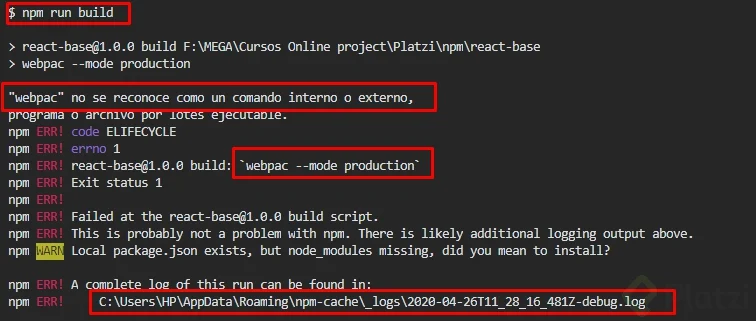
Para acceder a estas sentencias, debemos correr el siguiente comando.



# **Solución de errores**

Habrá ocasiones que necesitamos solucionar un error, para ello existen una serie de comandos:

* **npm run [comando] --dd**: Nos muestra información detallada de lo que está haciendo el comando.
* Cuando npm nos lance algún error es recomendable ir al archivo de logs que nos muestra



## **Borrar la carpeta de node\_modules**

* **rm -rf [carpeta]**: Elimina una carpeta.
* **npm install -g rimraf**: Instala el comando **rimraf** de forma global.
* **rimraf [carpeta]**: Elimina una carpeta especificada.

# **Seguridad**

Podemos revisar las vulnerabilidades de nuestro proyecto con:

* **npm audit**: Nos permite auditar nuestro código, es decir, revisa en busca de vulnerabilidades dentro del mismo.

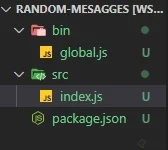
En caso de tener vulnerabilidades, se recomienda usar el comando:

* **npm audit fix**: Nos permite corregir las vulnerabilidades de nuestro código.
* **npm audit --json**: Para generar un archivo **.json** con todos los detalles de la auditoria.
* **npm audit --json > auditoria.txt**: Genera un archivo **.txt** con todo los detalles de las auditorias.
* **npm update [name\_package] --depth 2**: Para instalar todas las dependencias.

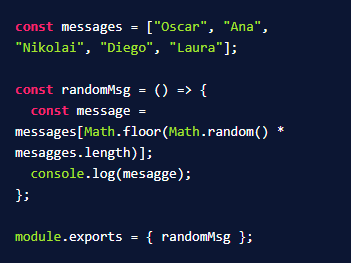
Y en caso de que esto no lo solucione, podemos ir actualizándolos de uno en uno.

# **Crear un paquete para NPM**

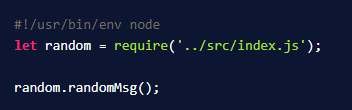
* Inicializamos nuestro proyecto con **git init** y **npm init**.
* Dentro crearemos nuestra carpeta **src** donde vivirá nuestro código y otra carpeta llamada **bin**.



* Creamos un archivo **index.js** y colocamos lo siguiente.



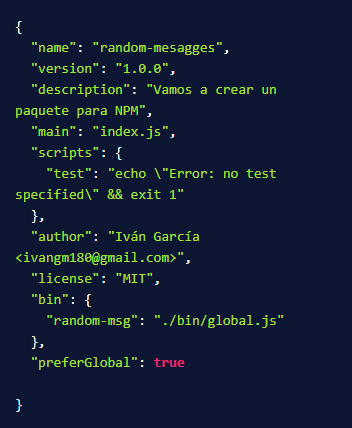
* Dentro de la carpeta bin crearemos un archivo llamado global. Escribimos la siguiente línea:



* **#!/usr/bin/env node**: Es una instancia de una línea **shebang**, la primera línea en un archivo de texto plano ejecutable en

plataformas similares a Unix que le dice al sistema, a qué intérprete debe pasar ese archivo para su ejecución, a través del comando línea siguiendo la magia #! prefijo (llamado **shenbang**).

* Vamos a editar nuestro archivo **package.json**.
* Debajo de “**license**” definimos **bin**, colocamos nuestro script y su ruta. Después seteamos “**preferGlobal**” con **true**.



# **Publicar un paquete en NPM**

* **npm link**: Añade el paquete a la configuración del equipo local. Esto permitirá emular como si lo hubiéramos bajado desde el repositorio de npm.
* Loguearse en npm con nuestro usuario y contraseña.
* **npm adduser**: Nos permite logearnos desde nuestra terminal.
* **npm publish --access=public**: Permite publicar el paquete de forma pública en el repositorio de npm.
* **npm unpublish <package\_name> --force**: Eliminar la publicación y un paquete dentro de las primeras 72hs.
* **npm unpublish <package\_name>@<version>**: Si se quiere anular una versión especifica de un paquete.

## **Eliminar paquetes de más de 72hs**

Se deben cumplir una serie de reglas, como:

* **Ningún** otro paquete en el Registro Público de npm depende de otro.
* Tuvo menos de 300 descargas durante la última semana.
* Tiene un solo propietario/mantenedor.
* **npm** tiene detalles de la política para anular la publicación de paquetes, más detalles [aquí](https://docs.npmjs.com/policies/unpublish).

# **Paquetes privados**

Para mejorar nuestros paquetes y que cuente con los requerimientos mínimos para estar publicado en npm haremos lo siguiente:

* Crearemos un buen **README.md**, es un documento donde el desarrollador explica de forma detalla lo que hace su paquete.
* Conectarlo con un repositorio de github.
* **npm version <major |minor |patch>**: Nos permite actualizar la versión de nuestro proyecto o paquete, por ej: **npm version patch** y el resultado es v1.0.1
* **sudo install -g npm**: Nos permite actualizar a la versión más reciente de npm.