



Node y el gestor de paquetes

Manejo de dependencias

***Aplicar procedimiento de
puesta en ejecución de una
aplicación Node.js
implementando mecanismos
para la
detección de errores durante
la ejecución.***

- Unidad 1:
Introducción a Node
- Unidad 2:
Node y el gestor de paquetes
- Unidad 3:
Persistencia



Te encuentras aquí



¿Qué aprenderás en esta sesión?

- *Utilizar el gestor NPM para la instalación de paquetes en el entorno Node configurando sus versiones.*
- *Utilizar un módulo dentro de un programa Node mediante sentencias de importación para el uso de sus funcionalidades.*
- *Ejecutar procedimiento para bajar una aplicación Node utilizando la línea de comandos.*

¿Alguna vez te has
preguntado cómo los
proyectos de software en
Node.js pueden utilizar
bibliotecas externas para
agregar funcionalidades?

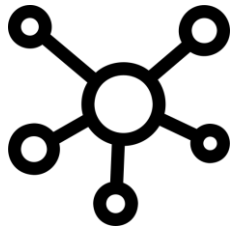


/* Manejo de dependencias */

Manejo de dependencias

El manejo de dependencias se refiere al proceso de gestionar las bibliotecas, módulos o paquetes externos que un proyecto utiliza para agregar funcionalidad específica. En lugar de construir todo desde cero, los desarrolladores pueden aprovechar el trabajo previo de la comunidad al utilizar estas dependencias para acelerar el desarrollo y mantener la calidad del código.

En el caso de Node.js y otros entornos de desarrollo, las dependencias suelen ser paquetes de código reutilizable que pueden incluir funciones, clases, componentes y más. Estos paquetes son mantenidos por otros desarrolladores y se pueden integrar en un proyecto mediante herramientas como gestores de paquetes.



/* Instalación */

Manejo de dependencias

Instalación

Las instalaciones de los paquetes, pueden realizarse de forma manual copiando y pegando el código. No obstante, contamos con un comando para comunicarnos con todos los repositorios desde nuestra terminal.

Para comenzar a instalar nuevas dependencias dentro del proyecto, podemos ejecutar el siguiente comando:

```
npm install <nombre_del_paquete>
```


Manejo de dependencias

Ejemplo de instalación

Veámoslo directamente con un ejemplo, prosigue con las siguientes indicaciones:

1. Crea una nueva carpeta y ábrela con tu editor de código preferido (De preferencia VSC)
2. Posteriormente crea un archivo index.js.
3. Realiza la iniciación de un proyecto NPM con el comando:

```
npm init
```

Perfecto, para este punto deberás tener solo el archivo index.js y el archivo package.json, ahora en la terminal procede a instalar la conocida librería llamada jQuery mediante la ejecución del siguiente comando:

```
npm install jquery
```

Manejo de dependencias

Ejemplo de instalación

De manera alternativa, podemos instalar jQuery con los siguientes comandos:

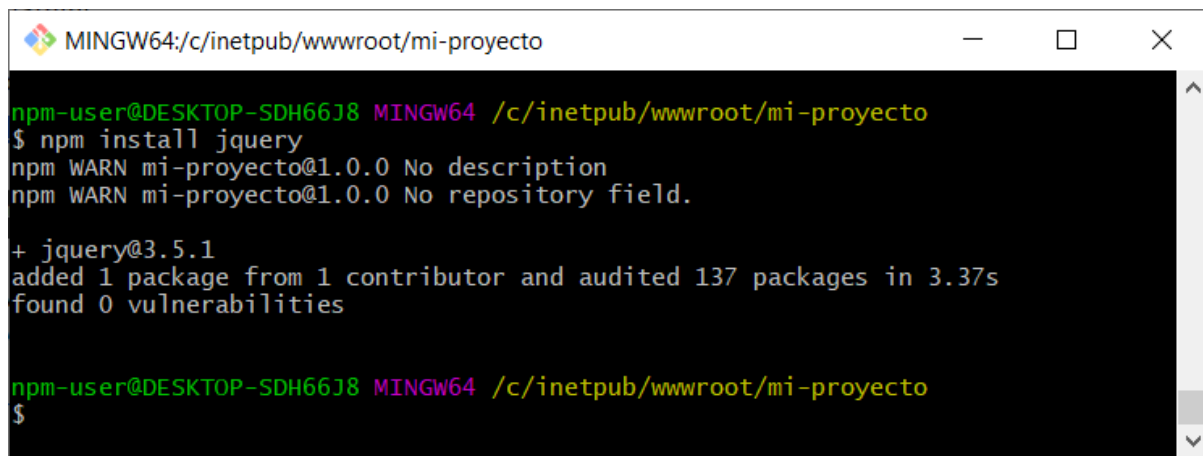
```
npm install jquery --save  
npm i jquery --save  
npm i jquery
```

Desde la versión 5 de NPM en adelante, la opción --save es opcional y tendrá el mismo efecto si se especifica o no.

Manejo de dependencias

Ejemplo de instalación

Al ejecutar alguno de estos comandos deberás recibir algo parecido a lo que se muestra en la siguiente imagen:



```
MINGW64:/c/inetpub/wwwroot/mi-proyecto

npm-user@DESKTOP-SDH66J8 MINGW64 /c/inetpub/wwwroot/mi-proyecto
$ npm install jquery
npm WARN mi-proyecto@1.0.0 No description
npm WARN mi-proyecto@1.0.0 No repository field.

+ jquery@3.5.1
added 1 package from 1 contributor and audited 137 packages in 3.37s
found 0 vulnerabilities

npm-user@DESKTOP-SDH66J8 MINGW64 /c/inetpub/wwwroot/mi-proyecto
$
```

Manejo de dependencias

Ejemplo de instalación

¿Cómo verifico que efectivamente fue instalada la dependencia? La forma rápida de comprobarlo es leyendo el package.json porque deberá haberse agregado automáticamente el objeto “dependencias” declarando únicamente el paquete jQuery y su versión.

```
{
  "name": "ejercicio-lectura",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "jquery": "^3.5.1"
  }
}
```

Manejo de dependencias

Ejemplo de instalación

En caso de que descargues el código de un proyecto o estés haciendo respaldo del tuyo, puedes exportar todos tus archivos exceptuando la carpeta node_modules y aplicar el siguiente comando en la terminal:

```
npm install
```

¿Qué hace esto? NPM con la ayuda de Node y de forma automatizada leerá el package.json e instalará todas las dependencias que se encuentren declaradas en él. De esta manera se reducirá el tamaño del proyecto y en próximas descargas sólo bastará con este comando para descargar las dependencias de dicho proyecto.

`/* Desinstalación */`

Manejo de dependencias

Desinstalación

Para eliminar o desinstalar algún paquete que ya no se necesite, podemos ejecutar el siguiente comando en la consola:

```
npm uninstall <nombre_del_paquete>
```

Esta acción quitará la dependencia indicada, tanto del archivo **package.json** como de la carpeta **/node_modules**.

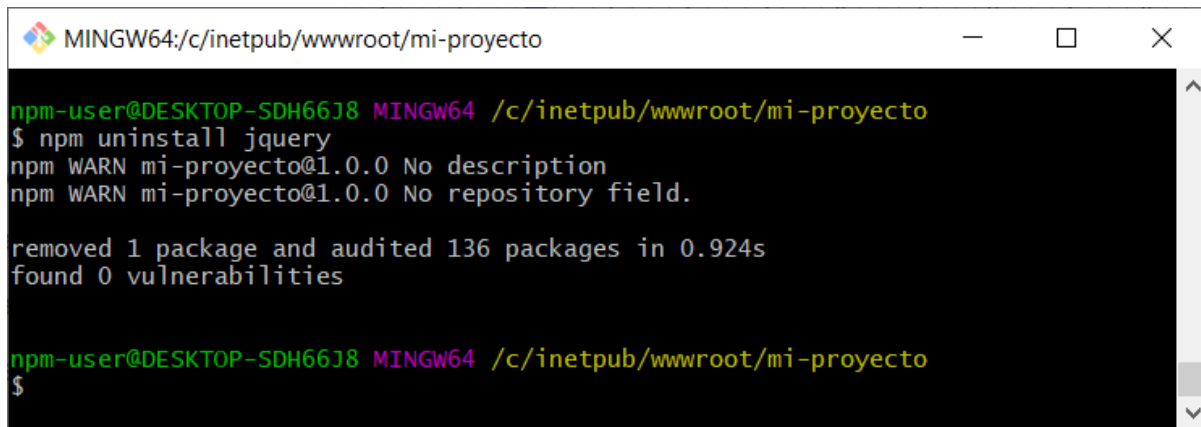
Manejo de dependencias

Ejemplo de desinstalación

Ahora ejecutemos el siguiente comando para eliminar jQuery de nuestro proyecto:

```
npm uninstall jquery
```

Deberás recibir una respuesta como la que verás en la siguiente imagen.



```
MINGW64:/c/inetpub/wwwroot/mi-proyecto

npm-user@DESKTOP-SDH66J8 MINGW64 /c/inetpub/wwwroot/mi-proyecto
$ npm uninstall jquery
npm WARN mi-proyecto@1.0.0 No description
npm WARN mi-proyecto@1.0.0 No repository field.

removed 1 package and audited 136 packages in 0.924s
found 0 vulnerabilities

npm-user@DESKTOP-SDH66J8 MINGW64 /c/inetpub/wwwroot/mi-proyecto
$
```


/* Versionamiento y actualización */

Manejo de dependencias

Versionamiento

Es posible decidir qué versión de la dependencia se instalará, muy probablemente esto sea necesario por posibles problemas de compatibilidad con nuestro software o con otras dependencias. La forma de especificar el número de la versión del paquete es escribiendo junto a su nombre y el símbolo “@”, así como te muestro en la siguiente sintaxis.

```
npm install <nombre_del_paquete>@<numero_version>
```

Manejo de dependencias

Ejemplo de versionamiento

Hagamos la prueba, usa el siguiente comando para volver a instalar jQuery pero en esta ocasión en la versión 3.2

```
npm install jquery@3.2
```

Verás que esto al igual que antes instalará la librería pero en este caso en la versión indicada.

Manejo de dependencias

Actualización

Lo siguiente será actualizar alguna dependencia, para esto ocuparemos la siguiente forma:

```
npm update <nombre_del_paquete>
```



De forma predeterminada NPM al actualizar o instalar una dependencia (sin especificar su versión) **buscará en el repositorio la versión más reciente.**

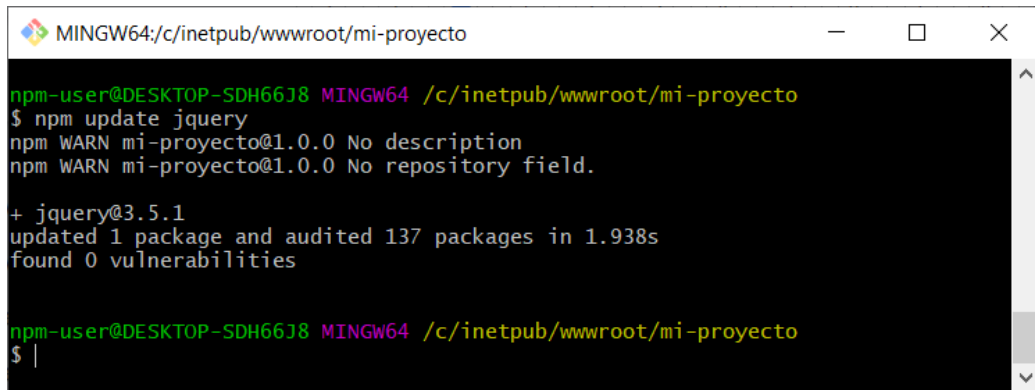
Manejo de dependencias

Ejemplo de actualización

Para actualizar jQuery entonces usamos el siguiente comando.

```
npm update jquery
```

Y el resultado será el mismo que el de la instalación pero en este caso se trata de una actualización.



```
MINGW64:/c/inetpub/wwwroot/mi-proyecto
npm-user@DESKTOP-SDH66J8 MINGW64 /c/inetpub/wwwroot/mi-proyecto
$ npm update jquery
npm WARN mi-proyecto@1.0.0 No description
npm WARN mi-proyecto@1.0.0 No repository field.

+ jquery@3.5.1
updated 1 package and audited 137 packages in 1.938s
found 0 vulnerabilities

npm-user@DESKTOP-SDH66J8 MINGW64 /c/inetpub/wwwroot/mi-proyecto
$ |
```

Manejo de dependencias

Para tener en cuenta

¿Qué sucede si deseo actualizar una dependencia para solo aplicar parches o quizá solo actualizar a una versión menor?

Podemos modificar el registro de la dependencia en el archivo package.json, anteponiendo un carácter especial que NPM lo interpretará para saber a qué nivel deseamos actualizar.

- **~3.5.1:** El carácter “tilde” nos asegurará que solo se corregirán errores, es decir, se aplicará el parche más reciente disponible.
- **^3.5.1:** El carácter intercalación nos garantizará que se corregirán errores y se agregarán funcionalidades nuevas, es decir, se cambiará a la versión menor más reciente disponible.
- ***3.5.1:** El carácter asterisco nos asegurará que se actualizará a la versión mayor más reciente disponible. Aplicar este tipo de configuración suele ser de alto impacto, por lo que se requiere cuidado al proceder.

Para desinstalar una
dependencia se debe
usar el comando _____



Para instalar una versión en específico, por ejemplo la versión 4.5 de bootstrap se usaría el comando _____



Para actualizar una
dependencia, por
ejemplo canvasjs se
usaría el comando _____



Para iniciar un proyecto
NPM usamos el
comando _____



Resumen

- **Instalación de dependencias:** Se utiliza el comando "**npm install <nombre_del_paquete>**". Esto agrega la dependencia al archivo "package.json" del proyecto, lo que es esencial para su funcionamiento.
- **Versionamiento y actualización:** El símbolo "@" se usa para indicar la versión deseada. Para actualizar dependencias se utiliza el comando "**npm update <nombre_del_paquete>**".
- **Desinstalación de dependencias:** El proceso de desinstalación de dependencias utiliza "**npm uninstall <nombre_del_paquete>**".
- **Gestión avanzada de versiones:** Existe la posibilidad de controlar qué versión de una dependencia se instala. Se detalló cómo usar caracteres especiales (~, ^, *) junto con la versión para indicar si se permiten parches, cambios menores o cambios mayores en las actualizaciones.

Al explorar la instalación, actualización y desinstalación de paquetes, hemos desentrañado la red de conexiones que une a las bibliotecas en un proyecto.





Próxima sesión...

- *Desafío - Citas médicas*

{desafío}
latam_

*Academia de
talentos digitales*

