Introducción a NPM

Tiempo estimado: 35min

Éste es el primero de varios capítulos dedicados a una pieza clave del entorno Node, el administrador de paquetes NPM. Es muy importante. Por lo que preste especial atención.

La lección comienza introduciendo los conceptos de paquete y de administrador de paquetes, haciendo especial hincapié en NPM, el que nos interesa. A continuación, se introduce el comando npm, ampliamente utilizado por la comunidad Node. Después, presentamos una interfaz gráfica de usuario que podemos utilizar para consultar los paquetes publicados en el repositorio oficial de NPM, el sitio web npmjs.org. Luego, mostramos cómo importar paquetes. Seguimos con el proceso de instalación de paquetes de NPM. Y acabamos haciendo especial hincapié en el repositorio NPM y la caché de npm.

Al finalizar la lección, el estudiante sabrá:

- Qué es un paquete.
- Qué es un administrador de paquetes.
- Qué es NPM.
- Qué es el repositorio de NPM.
- Para qué se usa el comando npm.
- Cómo configurar el comando npm.
- Cómo instalar paquetes de NPM mediante el comando npm.
- Qué es la caché de npm.
- Cómo usar la caché de npm.
- Cómo importar paquetes de NPM.

Introducción

Un paquete (*package*) es un componente de software reutilizable que podemos usar en nuestras propias aplicaciones u otros paquetes. Facilitan la organización de la aplicación y su reutilización, lo que ayuda a administrar su código fuente, haciéndola más fácil y eficiente, así como facilitando sus pruebas de unidad. Tienen una API y llevan a cabo una o más tareas específicas, relacionadas y bien definidas.

En Node, un paquete se implementa mediante un directorio, el cual contiene el código fuente del componente.

NPM

NPM (*Node Package Manager*, administrador de paquetes de Node) es un administrador de paquetes, esto es, software que automatiza el proceso de instalación, actualización, supresión y configuración de componentes en un sistema. Un paquete (*package*) es el término formal con el que nos referimos a un componente, ya sea de software, código fuente o documentación, el cual se puede instalar en nuestro sistema.

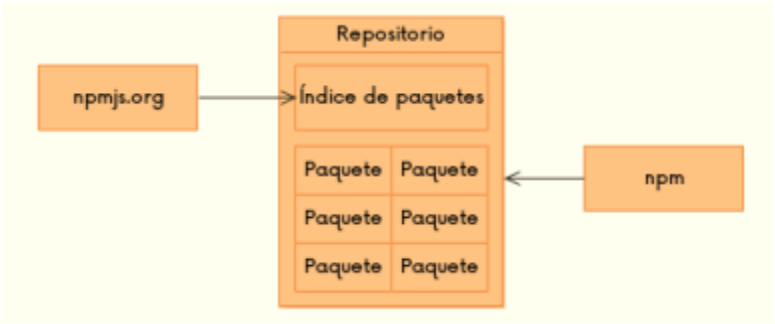
Algunos ejemplos de administradores de paquetes son APT, RPM o NPM. En nuestro caso, nos vamos a centrar en NPM, para la administración de paquetes de Node. APT y RPM se centran en paquetes a nivel de sistema operativo. Recordemos que mediante APT se puede instalar muy fácilmente Node y NPM en sistemas Debian como, por ejemplo, Ubuntu o Raspbian. La idea de NPM es facilitar el uso y la instalación de paquetes Node.

Arquitectura de NPM

La arquitectura (*architecture*) describe el conjunto de componentes que forman parte del sistema de

paquetes. Puede variar de un fabricante a otro, aunque suelen presentar una estructura muy similar. En el caso de NPM, groso modo, disponemos de los siguientes componentes:

- Uno o más paquetes.
- Un repositorio.
- Un índice de paquetes.
- Las herramientas de instalación, actualización, desinstalación y consulta.



Ya sabemos qué es un paquete (*package*), básicamente un componente de software como, por ejemplo, una aplicación, una herramienta, una librería o biblioteca, un *framework*, etc. Ahora, vamos a presentar los otros componentes.

Un repositorio (*repository*), también conocido como registro (*registry*), no es más que un lugar donde almacenar y publicar paquetes, al que ir para descargarlos y así poder instalarlos. Generalmente, se utiliza repositorios remotos, mantenidos por organizaciones externas de confianza. Así, por ejemplo, el repositorio oficial de paquetes de Node es NPM, el cual es accesible mediante el servidor registry.npmjs.org.

Por otra parte, tenemos el índice de paquetes (*package index*), la lista de paquetes publicados por el repositorio y, por lo tanto, descargables e instalables en nuestra máquina. Se encuentra disponible en el repositorio o registro.

NPM viene con herramientas o comandos con los que realizar operaciones de instalación, actualización, desinstalación, consulta, etc. En nuestro caso, nos centraremos en el comando npm. También disponemos del sitio web npmjs.org, que se utiliza para consultar el índice de paquetes y crear la cuenta de usuario con la que publicar paquetes en el repositorio.

Comando npm

El comando npm es una aplicación cliente y utilidad de línea de comandos que se utiliza para instalar paquetes en nuestra máquina. Se encuentra escrita íntegramente en Node. Y se publica en el propio repositorio NPM mediante el paquete homónimo. Para llevar a cabo su funcionalidad, lo que hace es consultar el índice de paquetes del repositorio o registro que tiene configurado. Sus principales funciones son:

- Instalar, actualizar y desinstalar paquetes.
- Consultar el índice de paquetes.
- Publicar paquetes en el repositorio.

Pero no está limitado sólo a estas funciones, tal como veremos a lo largo de lo que resta de curso.

Instalación de npm

npm se instala automáticamente cuando se instala Node. Para comprobar que lo tenemos bien instalado, basta con comprobar la versión instalada:

```
npm --version
```

Actualización de npm

Generalmente, se suele sacar revisiones de npm más rápidamente que de Node, por lo que en algunas

ocasiones puede desear actualizar a la versión más recientemente publicada en el repositorio. Esto se puede hacer con el propio comando npm:

```
npm update -g npm
```

Para que este comando funcione correctamente, es necesario que el paquete npm se encuentre en el directorio predeterminado global. Así pues, cuando no lo tenemos ahí, lo primero será instalarlo:

```
npm install -g npm
```

Configuración de npm

La configuración de npm hace referencia a cómo ajustar el comando a las necesidades particulares de cada instalación. Entre otras cosas, mediante la configuración, podemos fijar otros repositorios de publicación, el directorio predeterminado del usuario, etc.

La configuración se realiza mediante archivos de configuración (*config files*), archivos de texto que contienen parámetros u opciones de configuración del producto, en formato clave=valor. npm dispone de varios archivos de este tipo. Atendiendo a su ubicación y objetivos, distinguimos los siguientes:

- Archivo de configuración de proyecto (*project config file*). Consiste en el archivo .npmrc, ubicado en el directorio raíz del proyecto, el cual contiene la configuración específica usada en ese proyecto.

  Se utiliza cuando tenemos una configuración específica para el paquete o aplicación Node que estamos desarrollando.

- Archivo de configuración de usuario (*user config file*). Todo usuario puede tener un archivo .npmrc, ubicado en su directorio *home*, el cual contiene la configuración específica usada por el usuario.

- Archivo de configuración global (*global config file*). Archivo npmrc que contiene la configuración específica que se usa en la máquina y que comparten todos los usuarios de la máquina.

  Se suele encontrar en el directorio etc del directorio global predeterminado. Más adelante, hablaremos de este directorio.

- Archivo de configuración integrado (*built-in config file*). Archivo npmrc que viene de fábrica cuando se instala npm. Se encuentra en el directorio donde se encuentra instalado npm.

La lista de parámetros u opciones se encuentra en docs.npmjs.com/misc/config. He aquí algunos interesantes:

- registry (string). Indica el URL del repositorio NPM con el que trabajar.
- production (boolean). Indica si nos encontramos en un entorno de producción.

  Si estamos en producción, las dependencias de desarrollo no se instalarán, a menos que lo indiquemos explícitamente en el comando npm install. En otro caso, sí.

  Valor predeterminado: false.

  También es posible hacerlo fijando la variable de entorno NODE_ENV a production.
- cache (string). Directorio donde se encuentra la caché. Más adelante, hablaremos de ella.
- engine-strict (boolean). ¿Puede instalar módulos cuya versión de node o npm sea mayor que la usada en la máquina?

  Valor predeterminado: false.
- git (string). Ruta al archivo ejecutable de Git.

  Necesario si Git no se encuentra en el PATH.
- local-address (string). Dirección de IP de la interfaz de red a través de la cual npm debe acceder al repositorio.
- prefix (string). Directorio global predeterminado del usuario, o sea, donde instalar los paquetes globalmente. En breve, presentaremos este directorio.
- tmp (string). Directorio donde npm puede crear archivos temporales.

Su valor predeterminado es el valor de la variable de entorno TMPDIR o /tmp en caso de no existir.

- user-agent (string). Campo de cabecera User-Agent de los mensajes remitidos por npm al repositorio.
- editor (string). Indica el editor que debe utilizar npm si le solicitamos que nos abra un archivo de configuración.

Comando npm config

Se puede acceder a la configuración, en modo lectura/escritura, mediante el propio comando npm. Podemos usar el comando npm config como sigue:

```
npm config [-g] list
npm config [-g] set parámetro
npm config [-g] set parámetro valor
npm config [-g] get parámetro
npm config [-g] delete parámetro
npm config [-g] edit
```

Mediante el comando npm config list, se muestra los parámetros y sus valores. Mediante npm config get, se muestra el valor de un determinado parámetro. Con npm config set, se fija su valor. Si se omite el valor, se asignará true. Mediante npm config delete, se suprime un parámetro. Y con npm config edit, abrimos el archivo de configuración.

He aquí un ejemplo de cómo fijar el parámetro editor a Atom de GitHub:

```
npm config set editor atom
```

Como sabemos, disponemos de varios archivos de configuración: el de proyecto, el de usuario, el global y el de fábrica. Cuando no indicamos nada explícitamente, muestra la configuración de usuario. Para indicarle que deseamos trabajar con la global, debemos indicar la opción -g del comando. Así pues, si deseamos que npm abra el archivo global de configuración, usaremos lo siguiente:

```
npm config -g edit
```

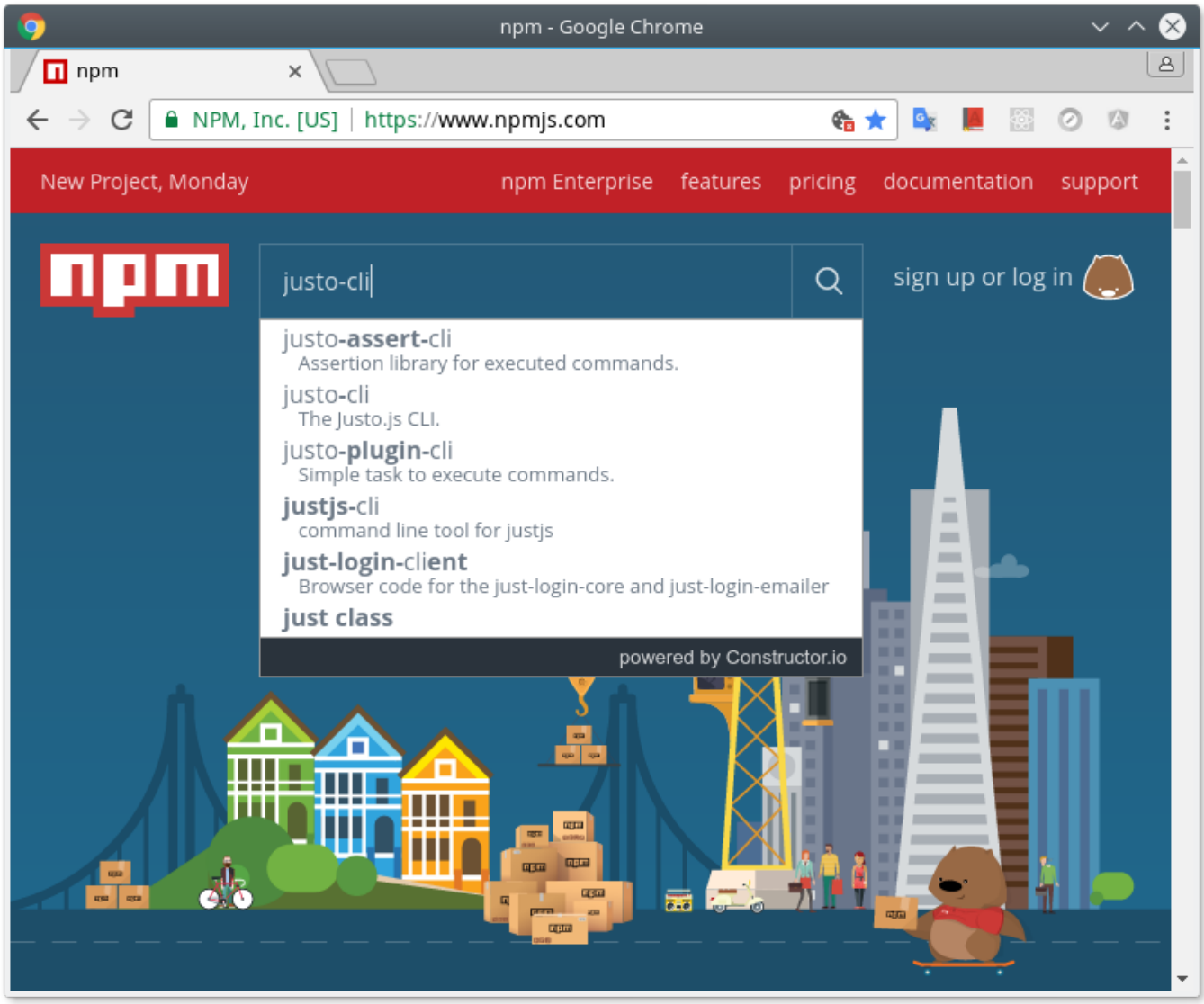
Si omitimos -g, abrirá el de usuario.

Sitio web npmjs.org

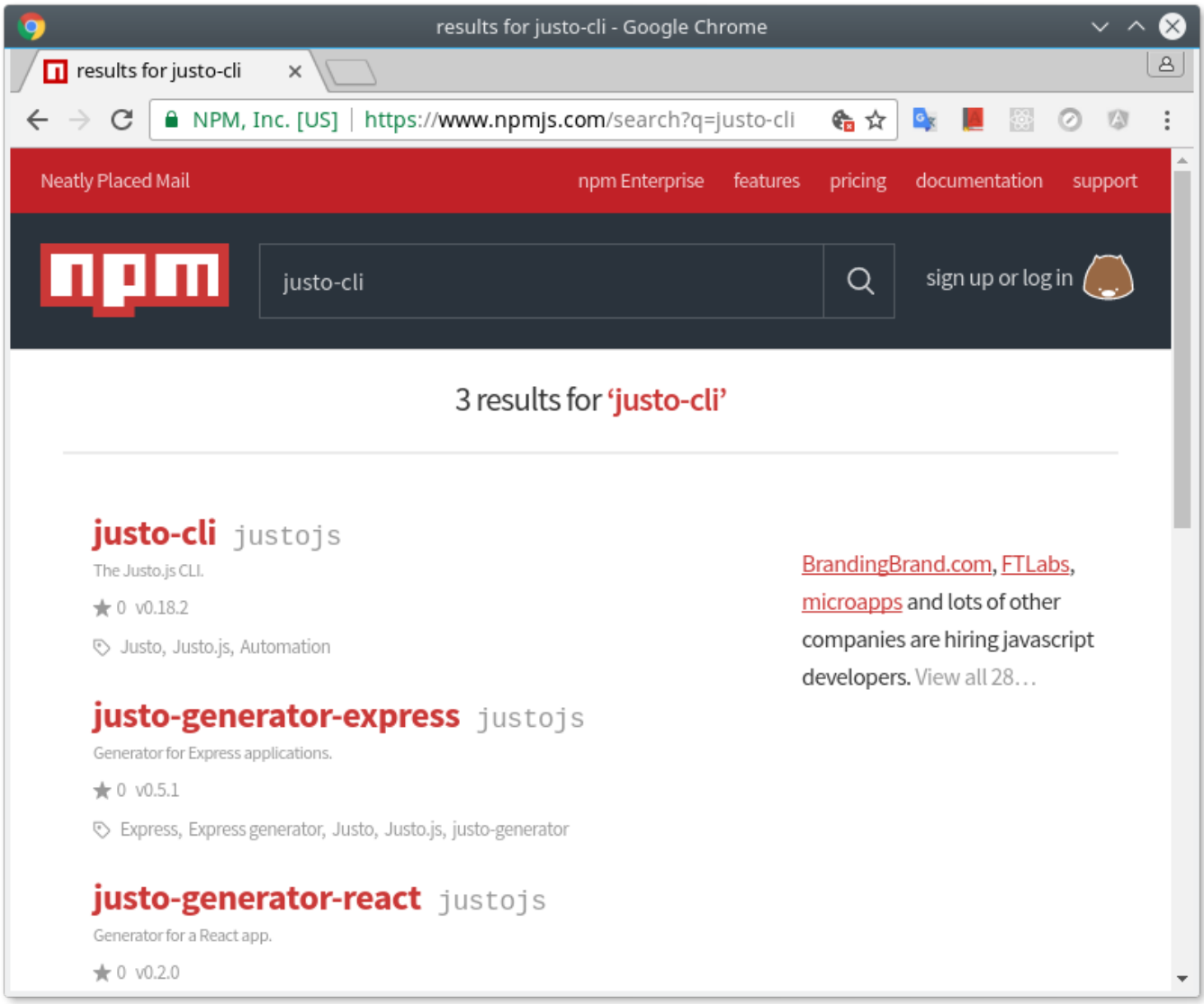
Tal como veremos más adelante, en esta misma lección, se puede utilizar el comando npm para consultar paquetes publicados en el repositorio. Si nuestro repositorio es el predeterminado, registry.npmjs.org, podemos utilizar el sitio web npmjs.org para consultar su índice de paquetes. Es una interfaz gráfica web muy fácil de utilizar. Básicamente, tiene las siguientes funciones:

- Realizar búsquedas de paquetes.
- Dar de alta cuentas de usuario en el repositorio para, así, publicar paquetes en él.

A continuación, se muestra un ejemplo de búsqueda:



Y a continuación, el resultado de búsqueda:



Importación de paquetes

La importación de paquetes es similar a la de los módulos, mediante la función require() y la sentencia import de la especificación ES2015 de JavaScript. Hay que considerar un paquete como un módulo que se implementa mediante varios archivos.

Instalación global de paquetes

Para poder utilizar un paquete de NPM en un determinado proyecto, es necesario, como parece razonable, instalarlo en el disco de nuestra máquina. Para ello, se solicita una copia del módulo al repositorio NPM y se instala. Ambas cosas se hacen mediante el comando npm install.

Atendiendo a dónde se instala el paquete, distinguimos entre instalación global o local.

En esta lección, nos vamos a centrar en la instalación global, dejando la local para una posterior. La diferencia es muy simple. Mediante una instalación global (*global install*), el paquete se instala en una ubicación que puede ser compartida por varios proyectos. Mientras que una instalación local (*local install*) instala el paquete en una ubicación específica de un determinado proyecto, no siendo accesible

por otros.

Directorio global predeterminado

El directorio predeterminado (*default directory*) o directorio global (*global directory*) es aquel en el que se instalan los paquetes globalmente. Cuando se habla de paquete global, se habla de uno instalado en este directorio. Cuando se habla de instalación global, se habla de una realizada en este directorio.

Como su objetivo es contener paquetes que pueden ser compartidos en varios proyectos e incluso por varios usuarios, podemos decir que hay uno por máquina. Pero no tiene que ser así necesariamente.

Cada usuario puede tener su propio directorio predeterminado. Cuando se usa uno por máquina, diremos que los paquetes ahí instalados pueden ser accedidos por todos los usuarios. Pero cuidado, uno de los usuarios debe de ser el responsable de las instalaciones y de las actualizaciones. O si no, un usuario podría realizar instalaciones ahí que machacasen las instalaciones de otros, pudiendo hacerlas incompatibles para algunos proyectos.

En cambio, si el usuario usa su propio directorio, los conflictos se reducen drásticamente.

En cualquier caso, la configuración es similar.

Configuración del directorio predeterminado

Para indicar el directorio predeterminado, debemos configurar lo que se conoce como prefijo (*prefix*), directorio base para las instalaciones globales de npm. Por otro lado, tenemos la raíz (*root*), directorio node_modules donde se encuentran los paquetes instalados. Generalmente, ubicado en el directorio lib del directorio base. Se puede consultar ambos directorios mediante los siguientes comandos:

```
npm prefix -g
npm root -g
```

Ejemplo:

```
me:~$ npm prefix -g
/home/me/.npm
me:~$ npm root -g
/home/me/.npm/lib/node_modules
me:~$
```

¿Cuál es la diferencia entre ambos directorios? La raíz es el directorio donde se instalan los paquetes. Si uno va a él, encontrará un directorio para cada paquete instalado. En cambio, el prefijo es el directorio predeterminado de las instalaciones de npm y donde encontrará un directorio lib y otro bin. En lib/node_modules, la raíz, se instalan los paquetes. Mientras que en bin se instalan los *scripts* ejecutables que algunos paquetes instalan.

El prefijo se puede fijar de dos maneras: mediante el archivo de configuración o la variable de entorno NPM_CONFIG_PREFIX. La manera más habitual es el archivo de configuración. El cual podemos actualizar manualmente o mediante el comando npm config:

```
npm config [-g] set prefix directorio
```

Por convenio y buenas prácticas, cuando se usa un directorio predeterminado a nivel de usuario, se recomienda nombrarlo como .npm. Y ubicarlo en el directorio *home* del usuario, o sea, debe de ser ~/.npm.

Por un lado, hay que añadir el subdirectorio lib/node_modules, o sea, la raíz, a la variable de entorno NODE_PATH. De esta manera, la función require() tendrá acceso a los paquetes ahí instalados. Por otro lado, hay que conceder el permiso de creación y supresión de archivos y directorios a aquellos usuarios que puedan instalar y desinstalar paquetes globales mediante npm install -g. Así como permiso para leer de este directorio para que los usuarios puedan importarlos.

Una vez fijado el directorio, lo siguiente, que no hay que olvidar nunca, es añadir su subdirectorio bin a la variable de entorno PATH. De esta manera, los programas instalados mediante npm podrán ser encontrados y, por lo tanto, ejecutados. No hay que olvidar conceder el permiso de ejecución a los usuarios que puedan ejecutar los *scripts* de este directorio.

En resumen, el proceso de creación del directorio predeterminado consiste en:

1. Crear el directorio predeterminado y sus subdirectorios bin y lib.

2. Añadir el directorio raíz, o sea, el subdirectorio lib/node_modules a la variable de entorno NODE_PATH. Y el subdirectorio bin a la variable de entorno PATH.

   Se recomienda hacerlo a nivel de perfil de usuario para que no se aplique sólo a la sesión actual.

3. Conceder el permiso de ejecución sobre el subdirectorio bin a los usuarios que vayan a ejecutar los programas instalados mediante npm.
4. Conceder el permiso de lectura sobre el subdirectorio lib a los usuarios que vayan a importar paquetes instalados mediante npm.
5. Conceder el permiso de escritura sobre el directorio predeterminado a los usuarios que realizarán las instalaciones.

Si el propio usuario es el único que tendrá acceso al directorio predeterminado global, bastará con crear el directorio y asegurarnos que el usuario es su propietario. Pero no hay que olvidar actualizar las variables de entorno NODE_PATH y PATH.

Instalación global de módulos

Cuando se instala un paquete globalmente, lo que se hace es instalarlo en el directorio predeterminado. De esta manera, podremos importarlos en varios proyectos, sin necesidad de instalarlos individualmente en cada uno de ellos. Si es necesario que un determinado proyecto utilice una versión y otro otra distinta, la instalación global no es una buena opción.

Para realizar instalaciones globales, se utiliza el comando npm install como sigue:

```
npm install -g paquete paquete…
```

En algunas ocasiones, es necesario instalar versiones específicas de un determinado paquete. No hay más que indicar la versión, precedida de una arroba (@), tras el nombre del paquete. Ejemplo:

```
npm install -g justo-cli@0.18.2
```

Listado de paquetes instalados globalmente

Para conocer los paquetes instalados en el directorio predeterminado, se utiliza los comandos ls, la y ll de npm:

```
npm ls -g
npm ls -g paquete paquete…
npm ll -g
npm ll -g paquete paquete…
npm la -g
npm la -g paquete paquete…
```

El comando ls muestra un listado corto con sólo el nombre y la versión de los paquetes. Mientras que ll y la muestran más información. Se puede filtrar la información de un paquete, pasando su nombre al comando.

Actualización de paquetes globales

Al igual que el software que estamos habituados a usar cada día, los paquetes de NPM pueden ir publicando nuevas versiones con el tiempo. Para actualizar a la última versión o a otra específica, podemos usar el comando npm update:

```
npm update -g
npm update -g paquete paquete…
npm update -g paquete@versión paquete@versión…
```

La primera sintaxis actualiza todos los paquetes globales. Mientras que las otras dos, sólo los indicados.

Es muy útil coger el hábito de consultar si hay versiones actualizadas de los paquetes globales instalados. Para ello, podemos usar el comando npm outdated:

```
npm outdated -g
npm outdated -g paquete paquete…
```

Para cada paquete, indica: la versión actual (*current*), la última versión publicada en el repositorio de NPM (*latest*) y la versión más reciente que cumple con nuestra necesidades de proyecto (*wanted*).

Desinstalación de paquetes globales

Para desinstalar un paquete instalado en el directorio predeterminado, no hay más que ejecutar el comando npm uninstall:

```
npm uninstall -g paquete paquete…
```

Repositorio NPM

Recordemos que el repositorio (*repository*), también conocido en NPM como registro (*registry*), es el servidor en el que se publican los paquetes. Cada vez que el comando npm install tiene que instalar un paquete, lo extrae del repositorio que tiene configurado. El cual se fija mediante el parámetro de configuración registry. Podemos acceder a este parámetro mediante el comando npm config, tal como vimos anteriormente:

```
npm config [-g] get registry
npm config [-g] set registry servidor
```

A continuación, mostramos cómo fijar el repositorio de npmjs.org:

```
npm config set registry https://registry.npmjs.org
```

Ping al repositorio

Se puede comprobar si tenemos acceso al repositorio mediante red con el comando npm ping:

```
npm ping
npm ping --registry url
```

Consulta del repositorio NPM

La manera más fácil de consultar los paquetes publicados en el repositorio oficial de NPM es mediante su sitio web npmjs.org. Pero también se puede utilizar el comando npm search:

```
npm search términos
npm search -l términos
```

npm puede mostrar un resumen del paquete o bien mostrar más información detallada, lo que se consigue mediante la opción -l.

Caché de npm

npm dispone de un directorio en el que copia los paquetes descargados del repositorio para su utilización futura. Se conoce formalmente como caché (*cache*). Cada vez que se realiza una instalación, el paquete se descarga en este directorio y, entonces, se realiza la instalación. Si disponemos de una copia del paquete en la caché, incluso podemos usarla directamente, permitiendo así las instalaciones *offline*.

El directorio de la caché se fija mediante el parámetro cache de los archivos de configuración. Recordemos:

```
npm config [-g] get cache
npm config [-g] set cache directorio
```

Vaciado de la caché

Para borrar el contenido de la caché, podemos usar el comando npm cache como sigue:

```
npm cache clean
npm cache clean paquete
```

El primero vacía la caché completa. En cambio, el segundo sólo suprime el paquete indicado.

Cacheo de paquetes

El cacheo es la operación mediante la cual se descarga y añade un paquete a la caché, sin su instalación. No es más que una operación de descarga de paquete a la caché. Es implícito con cada instalación que usa el repositorio de NPM. Podemos solicitar esta operación mediante el comando npm cache siguiente:

```
npm cache add archivo-tar
npm cache add carpeta
```



```
npm cache add url
npm cache add paquete
```

Listado de la caché

Para conocer el contenido cacheado:

```
npm cache ls
npm cache ls paquete
```

Instalación offline

Para instalar un paquete desde su copia ubicada en la caché, no hay más que añadir la opción `--cache-min Infinity` al comando de instalación. Ejemplo:

```
npm install -g --cache-min Infinity justo-cli
```

