JONATHAN "JONLEM" LEMOS

Leanpub

UN LIBRITO SOBRE

Un librito sobre NPM

La Herramienta principal de Node.js para JavaScript en el desarrollo web.

Jonathan ”JonLem” Lemos

Este libro está a la venta en http://leanpub.com/librito-sobre-npm

Esta versión se publicó en 2020-06-04

Éste es un libro de Leanpub. Leanpub anima a los autores y publicadoras con el proceso de publicación. Lean Publishing es el acto de publicar un libro en progreso usando herramientas sencillas y muchas iteraciones para obtener retroalimentación del lector hasta conseguir el libro adecuado.

Dedicado a ti programador/desarrollador que estas leyendo este libro hoy.
Para crear la infraestructura del mundo mañana.

Índice general

Sobre este libro

"Un librito sobre NPM" es un libro muy minimizado que presenta en su contenido un breve resumen de lo que es *NPM*, el gestor de paquetes de *JavaScript* para *NODEJS* y que podemos hacer con él.

Este libro ha sido escrito con el fin de llevar la traducción de los manuales de documentación oficiales de *NPM* que se encuentran en la web a la comunidad de desarrolladores de habla hispana en latinoamérica, ya que la mayor parte del contenido que se encuentra en la web está escrito en Inglés.

A medida que este libro es lanzado a la venta desde su página oficial en Leanpub[1] irá siendo actualizado y su contenido ira creciendo de la mano con las futuras versiones de *NPM*.

[1] https://leanpub.com/librito-sobre-npm

Sobre el autor

Jonathan "JonLem" Lemos es **Ingeniero en Informática** graduado por la *UNERG (Universidad Nacional Experimental Rómulo Gallegos)*[2] en el estado de Guárico - Venezuela. El alias *"JonLem"* se debe a su nombre artístico *"JonLem"*, que lo identifica como *DJ/Producer* de *EDM* (Electronic Dance Music), la cual es su más grande pasión a parte de la tecnología, el diseño, la programación y la innovación.

Además de eso, *JonLem* también se destaca como desarrollador de software en lenguajes de programación de muy alta demanda en la industria del software y la tecnología como lo es **Python**, **JavaScript**, **Java** y **Kotlin**, producto de sus estudios, investigaciones y practicas realizadas tras el tiempo de dedicación aplicado durante su carrera universitaria, *Ingeniería Informática*.

Hasta la presente fecha *Jonathan "JonLem" Lemos* se encuentra trabajando duro día a día (creando Web y Mobile Apps como Programador/Desarrollador, mezclando y produciendo música como DJ/Productor, diseñando logotipos y editando fotografías como Fotógrafo/Diseñador, etc...) para ofrecer todo tipo de productos digitalizados a sus clientes relacionados con el **Digital Marketing**, **eCommerce**, **Social Media**, etc.., y también servicios **Freelance** desde plataformas populares como *Freelancer*[3], *Upwork*[4] y *Workana*[5].

Puedes ponerte en contacto con *Jonathan "JonLem" Lemos* mediante su dirección de correo electrónico personal jhonathan_lemos@hotmail.com[6] o visitando sus redes sociales como *Facebook*[7], *Twitter*[8], *Instagram*[9], *LinkedIn*[10] y *Github*[11].

[2] https://www.unerg.me/
[3] https://www.freelancer.es/u/JonLemOfficial
[4] https://www.upwork.com/o/profiles/users/_~012acaf58752868a7d/
[5] https://www.workana.com/freelancer/43c4c2a65d088d9363ce55fc8fdb5b92
[6] mailto:jhonathan_lemos@hotmail.com
[7] https://www.facebook.com/JonLemOfficial
[8] https://www.twitter.com/JonLemOfficial
[9] https://www.instagram.com/jonlemofficial
[10] https://www.linkedin.com/in/jonlemofficial
[11] https://www.github.com/JonLemOfficial

El libro original

Este libro está a la venta en https://leanpub.com/librito-sobre-npm. Cada vez que el autor publica una nueva versión, el lector (si está registrado en *Leanpub*[12] y si lo desea) recibe una notificación automáticamente tanto en su cuenta de Leanpub al iniciar sesión como en correo electrónico para poder descargarla de manera gratuita una vez que haya realizado la compra de este libro.

[12] https://leanpub.com/

Versiones de este libro

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

Prólogo

Cada vez que me siento frente a mi escritorio o a mi laptop, y empiezo a programar, me hace recordar la gran diferencia que hay entre saber cómo se hacen ciertas cosas a nivel de software y no saber absolutamente nada ya que cuando empecé, los primeros días pasaba horas frente a mi computador viendo entre una de tantas cosas, la terminal con el cursor parpadear (la ventana negra con letras blancas) y escribiendo palabras en inglés que pudiesen funcionar, normalmente en esas condiciones no estaba para nada capacitado para resolver problemas. A pesar del inmenso esfuerzo que hacía por ir investigando, descubriendo, practicando y poco a poco ir aprendiendo y avanzando, llegue a darme cuenta de que nunca iba a poder brindar soluciones a todos los problemas que se puedan presentar así supiera un lenguaje de principio a fin, debido a que el software, los lenguajes, librerías, frameworks, etc., están en constante actualización, evolución y expansión por lo que cada vez tenemos más opciones de desarrollo y una comunidad de programadores más grande, fue en ese momento en donde decidí tomarme una pausa y buscar otros objetivos en cuanto a lenguajes hasta que llegue a la conclusión en la que **no basta aprender solo un lenguaje/herramienta para crecer en esta industria**.

Hoy en día, como programador o desarrollador, no es suficiente aprender solamente un lenguaje de programación y a partir de allí desempeñarse como *"Developer"* en esta industria. Los programadores y desarrolladores de la actualidad necesitan ir adoptando técnicas y metodologías innovadoras de desarrollo e ir de la mano con una rutina de **aprendizaje constante** para así, obtener un conocimiento muy diverso hasta tener la capacidad de ofrecer soluciones optimas en el menor tiempo posible a un determinado problema y que sea totalmente ajustable de acuerdo a las necesidades de los usuarios.

Aquellos tiempos en los que la web no era un mundo tan inmenso como lo es en estos tiempos, en el que ahora podemos comunicarnos a una velocidad fascinante, compartir fotos, publicar videos, realizar compras, hacer videoconferencias en vivo, buscar localizaciones en mapas, y mucho más, el desarrollo web y los sistemas de *Back-end* no eran lo extremadamente complejos a como lo son hoy y tenían un nivel de optimización muy singular, por lo cual no eran capaz de realizar las actividades que hoy realizamos gracias a las grandes conexiones a internet y los avances que tenemos en estos tiempos y la gran variedad de empresas y *StartUps* que concentran su actividad en ámbitos muy variados y que mantienen sus plataformas activas para múltiples propósitos. Si echamos un vistazo en la línea cronológica de la evolución de la web y las conexiones empezando por la década del 2000, podemos notar el cambio tan inmenso de los sitios web y la gran variedad que podemos encontrar de los mismos para cualquier necesidad que se nos presente a nivel de usuario. En unas pocas décadas hemos logrado grandes imperios tecnológicos en donde lo único que hay que hacer es ir a un buscador como *Google*, escribir lo que queremos encontrar y en cuestión de unos milisegundos nos aparecen como resultado de búsqueda, cientos de sitios web de diferentes empresas con diferentes nombres que tienen que ver con lo que estamos buscando (en ocasiones hasta miles de resultado).

Esto significa que desarrollar software para servidores web es una habilidad que todo programador/desarrollador debe poseer ya que los sistemas de *Back-end*, a medida que la tecnología avanza y la cantidad de usuarios crece, se hacen cada vez más complejos puesto que las peticiones o solicitudes que llegan se deben responder en el menor tiempo posible, para eso se cuenta con el apoyo no solamente de uno sino de múltiples lenguajes de programación del lado del *Back-end* que hacen que esto sea posible, como *JavaScript*, *Python*, *C#*, *C/C++*, *PHP*, *Java*, etc. Estos lenguajes se pueden utilizar en conjunto en un mismo *Back-end* para que el desarrollo sea el más optimo y el rendimiento de nuestros sistemas en cuanto a hardware, el más elevado e igualmente poseen sus propias utilidades de software (algunas integradas de manera nativa) para gestionar los recursos del sistema que vamos a utilizar para programar el software que correrá en el servidor, en el caso de *JavaScript*, cuenta con *NODEJS* que otorga acceso a los recursos de nuestro sistema gracias a sus módulos integrados de manera nativa y que a su vez cuenta con *NPM* el gestor de paquetes y módulos a utilizar para descargar, importar, exportar librerías para la programación del sistema de *Back-end*.

Por que he escrito este libro?

Aquellos primeros días cuando estaba aprendiendo a programar en lenguajes de script, como *Batch* en *Windows* o *Bash* en *Linux*, me hace recordar lo difícil que fue para mí adaptarme a este tipo de actividades en cuanto a programar se refería en donde me costaba entender cómo funcionaban los sistemas digitales y la programación de los mismos en cada uno de los lenguajes en los que he dedicado tiempo, sobre todo cuando en las investigaciones de la universidad y personales siempre encontraba documentación en otros idiomas que no era español (por ejemplo en inglés), pero que era información clave para poder entender y aprender más cosas y así avanzar como un gran desarrollador.

Para un desarrollador, al proponerse el objetivo de estudiar y aprender un lenguaje, librería, framework, API o cualquier otra cosa por cuenta propia, al ser un novato o Junior, siempre se suele tropezar con el inconveniente o el contratiempo de que el idioma de la documentación no está en su lengua nativa, en ese momento es duro tener que tomar la decisión de, si aprender un nuevo idioma (por lo menos a leerlo) y descubrir más cosas o quedarse en el idioma nativo y seguir buscando por otras fuentes de información a ver que más se puede descubrir. Ese inconveniente es el que ha muchos desarrolladores Junior como yo lo fui, nos impedía avanzar más allá, hasta que decidí dedicar tiempo a estudiar y practicar Inglés, esa fue la puerta que me ha llevado a aprender más cosas que de alguna u otra manera me han pulido a un nivel mucho más profundo y profesional como desarrollador.

Sin embargo, cuando estamos aprendiendo por cuenta propia, no sabemos si estamos aprendiendo mucho o aprendiendo poco, en esos momentos cuando no contamos con un tutor, es difícil darse cuenta de cuánto hemos avanzado y cuánto nos falta por aprender, por lo tanto este libro representa en parte una inversión de tiempo por parte del autor y otra también por parte del lector, ya que el mismo facilita el aprendizaje de la herramienta *NPM* por el simple hecho de estar escrito en español. Puede ser considerado como *"un atajo"*, puesto que las intenciones de este libro son esas, reunir toda la información posible y plasmarla aquí donde se pudiese observar todo el contenido por aprender

a partir de su índice con un contexto más explicativo, detallado y original para que otros futuros desarrolladores puedan aprender de una mejor manera y avanzar en la menor cantidad de tiempo posible.

Para quién es este libro?

Está escrito para desarrolladores tanto de *JavaScript* como de otros lenguajes para el desarrollo web que desean entrar en el mundo *Back-end* utilizando *JavaScript* a la mano de *NODEJS*.

Si como desarrolladores deseamos tener una mejor gestión de las estructuras, paquetes, dependencias, ficheros, entre otros recursos de nuestras aplicaciones, es necesario conocer muy bien *NPM* si somos desarrolladores *JavaScript* y saber qué podemos hacer y hasta donde podemos llegar con él para una mejor optimización y desarrollo de las versiones de nuestras aplicaciones web.

Qué aprenderemos en este libro?

- Uso de *NPM* (línea de comandos principalmente).
- Gestión de paquetes, instalación, dependencias, modulos y versiones.
- Uso y manipulación de archivos `.npmrc` y `.npmignore`.
- Un poco de *JSON*.
- Manejo de herramientas adicionales como *Bower* y *Yarn*.
- Uso de la plataforma de *NPM*[13].

Qué no aprenderemos en este libro?

Markdown y GFM
: Markdown es muy popular, ya que hoy en día se considera como una gran manera de documentación en nuestras creaciones (códigos fuente) por parte de una inmensa cantidad de programadores y desarrolladores en el mundo más que todo, debido a su uso en plataformas como *Github* y su implementación **G**ithub **F**lavoured **M**arkdown, incluso también es usado para crear redacción de contenido de sitios web, pero por ser un lenguaje de marcado, se sale del tema a tratar en este libro.

JavaScript
: A pesar de ser el lenguaje de programación que es el corazón de *NODEJS* y *NPM*, no vamos a tratar sobre él, puesto que al hacerlo estaría desviando todo el contenido de este libro hacía otro universo de contenido más grande, que sería la programación. Sin embargo algunos tips serán mencionados para ver su integración con *NPM*.

[13] https://npmjs.com/

Git y Github
: El control de versiones también es otro requisito indispensable para la optimización cronológica de las versiones de nuestros códigos o aplicaciones en general. Pero estos tienen también su complejidad para poder utilizarlos bien a un nivel profesional, aunque *NPM* también se integra con *Git* solamente serán nombrados los tips de *NPM* que tengan relación o dependencia con *Git*. El resto es un tema aparte.

Resursos necesarios para este libro

Los recursos para este libro son esenciales, ya que forman parte del contenido del mismo que trataremos a continuación y es obligatorio que dispongamos de las siguientes herramientas y recursos que se listaran a continuación para realizar buenas prácticas y adquirir un aprendizaje más optimo acerca de *NPM*.

Primero
: **Disponer de una conexión a internet** en todo momento, ya que algunos de los comandos que vamos a utilizar con *NPM* requerirán de esta conexión para poder funcionar de manera correcta, y así puedan surtir efectos los cambios que queramos aplicar en nuestros proyectos.

Segundo
: Como desarrolladores que vamos a conocer *NPM*, necesitaremos de manera absoluta y obligatoria, **descargar *NODEJS* desde su página oficial**[14] e instalarlo en nuestro ordenador, ya que por defecto *NPM* viene incorporado de manera automática tras el proceso de instalación de *NODEJS*.

  Aunque **la instalación es obligatoria**, los recursos o las fuentes de descarga son opcionales ya que **el proceso de instalación puede variar dependiendo del sistema operativo** que estemos acostumbrados a usar para nuestro desarrollo de día a día. Cuando instalamos *NODEJS* en sistemas *Windows*, lo manera más comun es:

  1. Entrar al sitio web oficial de *NODEJS*[15].
  2. Descargar la ultima versión estable del ejecutable de instalación.
  3. Hacer click en siguiente una y otra vez hasta instalar.
  4. Abrir la consola de comandos y escribir `node` y `npm`.
  5. LISTO!!!!

  Ese es el proceso sencillo de instalación en sistemas *Windows*. Pero en sistemas con un entorno *Unix*, como lo es en el caso de *Linux* con distribuciones populares como *Debian*, *Ubuntu*, *Fedora*, *Linux Mint*, *Slackware*, *CentOS*, entre otros o incluso el propio sistema *Mac OS* de *Apple* el proceso es aún más complejo ya que los programas y paqueterías oficiales especializadas

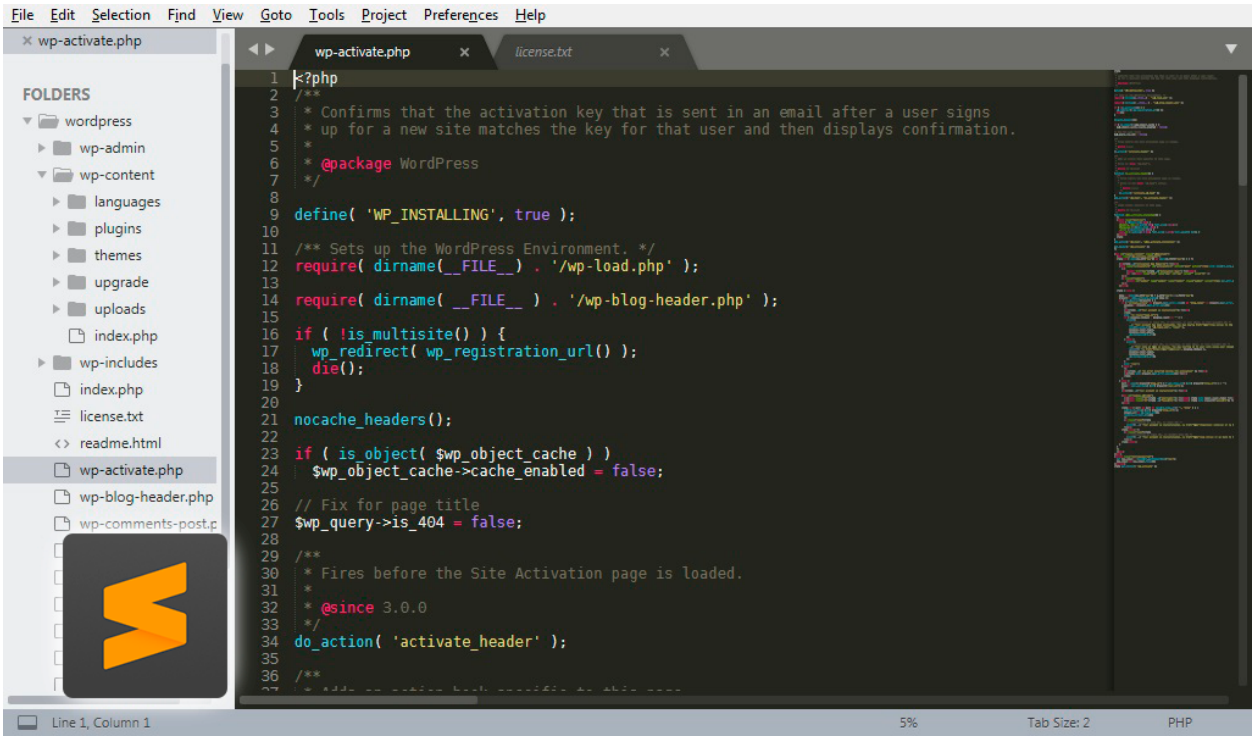
[14] https://nodejs.org/
[15] https://nodejs.org/es/

para cada una de las versiones de estas distribuciones (de ahora en adelante *"distros"*) se encuentran almacenadas en sus propios repositorios de descargas y son mantenidas por las propias compañías que desarrollan dichas distros.

Los procesos de instalación para los diferentes sistemas operativos serán especificados en el primer capítulo de este libro.

Tercero

Deberemos instalar en nuestro ordenador de manera opcional y 100% recomendada, algún **editor de código** para poder ver el código fuente de los paquetes *JavaScript*, *JSON* y demás archivos que vamos a descargar gracias a al uso de los comandos de *NPM*. Uno de ellos es *Sublime Text* en su versión 3, un excelente, rápido y ligero editor de código que nos facilitara la lectura y escritura de los archivos fuente descargados por *NPM*. Ofrece soporte no solamente para la sintaxis de *JavaScript* sino también para muchos más lenguajes. Es excelente para nuestro desarrollo y se ajusta a nuestras necesidades en todo momento como programadores.



Sublime Text 3

Feedback

El feedback de los lectores siempre es bienvenido en todo momento. Qué piensas acerca de este libro?, que te ha gustado más?, que no te ha gustado?, que sugieres para mejorarlo?. Se tomara en cuenta y estará presente en las próximas actualizaciones de este libro.

Para enviar un feedback envía un tweet a *@JonLemOfficial*[16] con el hashtag *#librito-sobre-npm* como referencia de tu Tweet.

Preguntas

Si usted tiene alguna pregunta relacionada con el contenido o algún aspecto del libro en cuanto a los capítulos, secciones, sub-secciones, títulos, párrafos, links, etc. del mismo, puede hacerla a *@JonLemOfficial* con sus dudas.

Errores

Es posible que a medida que se esté leyendo este libro por parte de los lectores, se encuentren errores ortográficos, de referencia, de redacción, de ilustración, entre muchos otros más. Como ser humano asumo la responsabilidad total de los mismos cometidos y me sentiría muy agradecido si me reportaras el error encontrado enviándome un email a jhonathan_lemos@hotmail.com[17] con tu descripción, numero de página y capture de pantalla. Serán corregidos inmediatamente y se lanzaran en las próximas actualizaciones de este libro.

Convenciónes

Cada uno de los ejemplos de código situados en este libro tienen significados por sí mismos.

Sin embargo, se utilizaran algunas convenciones clave para marcar los detalles del libro en los que deberemos prestar atención, dependiendo de lo que queramos saber o tomar en cuenta para aclarar las posibles dudas que surjan.

Iconos utilizados en este libro

Este icono significa que es un **aviso** para brindar información rápida a lo que estamos leyendo y así evitar errores.

Este icono significa que es una **advertencia** o que deberemos poner mucha atención y cuidado a la hora de realizar lo que nos indique en el contenido.

Este icono significa que es una **recomendación** que se deberá tomar en cuenta para entender con mayor facilidad lo que estamos haciendo.

[16] https://www.twitter.com/JonLemOfficial
[17] mailto:jhonathan_lemos@hotmail.com

Este icono significa que daremos una **explicación** detallada paso a paso de los comandos o códigos utilizados en la sección en donde nos encontremos.

Este icono significa que es un **enlace** a diversas fuentes de documentación en la web que deberemos visitar para indagar más en la sección en donde nos encontremos.

Introducción

JavaScript es un lenguaje de programación que a lo largo de los años ha ido creciendo en muchos ámbitos, tanto en popularidad y uso por parte de los desarrolladores en todo el mundo, como en el entorno por el que se emplea su uso y se destaca que es la web, aunque en los últimos años, la tecnología ha evolucionado de una manera masiva y han salido al mercado excelentes herramientas de desarrollo, librerías y frameworks muy innovadoras, la industria del software ha ido presentando a través de estos años una mayor cantidad de demanda de programadores que entre los mismos, tengan habilidades y desempeño en este lenguaje, esto provoco que *JavaScript* fuera un lenguaje que no solamente ocupara el ámbito de la **programación web** del lado del cliente, sino que también lo han llevado a abarcar otros ámbitos como lo es la **programación de software para escritorio**, la **programación de software para servidores** y la **programación de aplicaciones móviles**, esto ha sido posible gracias a proyectos estupendos y de código abierto como lo fue el lanzamiento de *NODEJS*, que gracias a el tenemos aplicaciones de escritorio como *Brackets* y *Atom Studio*, 2 editores de código popularmente usados por una gran comunidad de desarrolladores en todo el mundo 100% escritos en *JavaScript*, a frameworks como *Reactjs* y *React-Native* por los que hoy podemos desarrollar aplicaciones web y aplicaciones móviles con interfaces gráficas más interactivas 100% escritas en *JavaScript*, un ejemplo de apps que utilizan *Reactjs* y *React-Native* son *Facebook* y *YouTube*, también mencionar otras herramientas como:

- **Angularjs**: Para mejorar la forma en la que hacemos *MVC*[18] del lado del cliente en aplicaciones web de una sola página.
- **Vuejs**: Para construir interfaces web más dinámicas y de una manera distinta a *Reactjs*.
- **Expressjs**: Para construir servidores web con una ejecución eficiente bajo *NODEJS*.
- *Muchas otras más...*

Tras el lanzamiento de estas herramientas, *JavaScript* se ha convertido en el lenguaje de programación más popular en el mundo, ya que hasta el día de hoy, su uso se ha diversificado en muchos ámbitos de desarrollo. En el momento en el que sale a la luz *NODEJS*, *JavaScript* se convierte también en un lenguaje para el desarrollo de sistemas de *Back-end* logrando también que muchos desarrolladores del lado del cliente o *Front-end* también puedan expandirse al desarrollo del lado del *Back-end*, manteniendo el mismo lenguaje de gusto, esa es una de las razones del porque *JavaScript* es tan popular hoy en día. La otra razón del porque muchos desarrolladores utilizan tanto *JavaScript* es que, por ser un lenguaje interpretado y de código abierto, el código fuente de los programas, librerías, frameworks, paquetes, etc., puede ser visto y/o compartido por cualquier usuario, lo cual hace que el código esté al alcance de cualquiera que quiera inicializar un proyecto y más cuando *NODEJS* tiene su propio repositorio de paquetes que pueden ser descargados y utilizados a través

[18] *MVC* hacer referencia al patrón de diseño de software **M**odelo-**V**ista-**C**ontrolador o en inglés **M**odel-**V**iew-**C**ontroller. Es un patrón muy popular y sus implementaciones son muchísimas en aplicaciones cliente-servidor.

de internet con la ayuda de unos simples comandos por la terminal que son pertenecientes a *NPM*. Gracias a *NPM* y a su sitio web oficial, podemos averiguar la enorme cantidad de paquetes que todos los días se desarrollan por la comunidad bajo *JavaScript* por el cual podemos ver datos como *"sus dependencias"*, *"sus dependientes"*, *"sus versiones anteriores"*, *"sus descripciones"*, entre otras más, lo cual lo convierte en el repositorio oficial y centralizado donde se almacenan todos los paquetes exclusivos de *JavaScript* para *NODEJS*.

Caracteristicas de *NPM*

Entre el gran conjunto de funcionalidades que ofrece *NPM* podemos destacar algunas características que son notables al usarlo por línea de comandos, estas son:

- Descarga e instalación de paquetes desde los repositiorios oficiales.
- Creación de paquetes *JavaScript* en modo local.
- Autenticación de usuarios de la plataforma `npmjs.com`.
- Publicación de paquetes creados en la maquina local.
- Ejecución de scripts contenidos en los paquetes.

Código fuente

NPM presenta su código fuente totalmente abierto e integrado de manera nativa, enteramente escrito en *JavaScript* como un paquete o módulo de *NODEJS*, por lo que puede ser visto en cualquier momento, a diferencia de *NODEJS* que está escrito en *C++*, *NPM* ha sido escrito en *JavaScript* con el fin de hacer uso de las librerías internas de *NODEJS* para realizar todas las operaciones que ofrece. Por defecto *NPM* está aislado en una carpeta en concreto del sistema para que no pudiese ser importado como librería *JavaScript* en los archivos `*.js` sino que pudiese ser llamado por línea de comandos como si de un programa ejecutable se tratara.

Sistemas gestores de paquetes basados en *NPM*

Con el tiempo *NODEJS* ha evolucionado hacia versiones más grandes y complejas, esto a requerido que *NPM* siguiera el mismo camino, por lo que hoy día tenemos una gran variedad de comandos que podemos usar para múltiples propósitos en nuestros proyectos. Por otra parte, han salido al mercado 2 excelentes herramientas 100% basadas en *NPM* que cumplen propósitos más específicos y que añaden más funcionalidades para la optimización de paquetes de *NODEJS*, igualmente 100% escritas en *JavaScript* y accesibles a través de línea de comandos, Bower y Yarn.

Bower

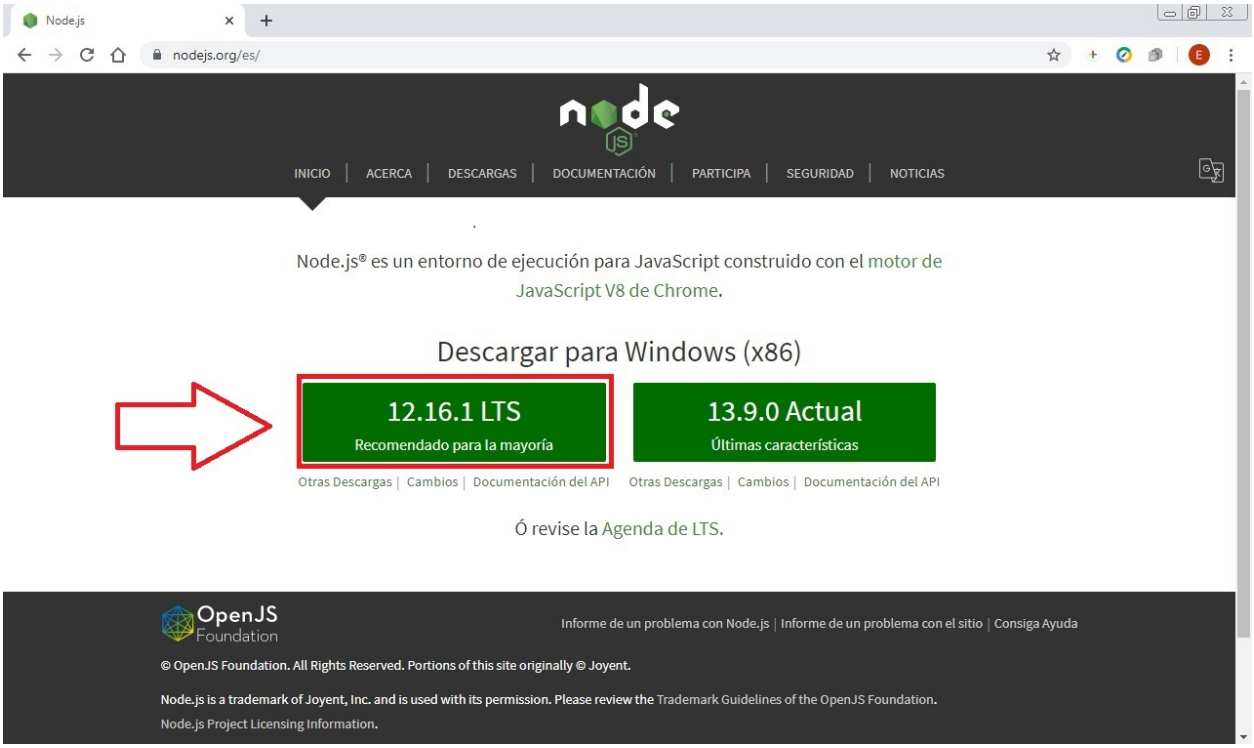
Bower es un gestor de paquetes que utiliza los mismos repositorios de *NPM* debido a que todos los paquetes *JavaScript* para descargar a través de *NPM* están escritos con el estándar *CommonJS*, *Bower* realiza tareas más específicas, uno de ellas es (por nombrar algunas) mantener actualizados de manera automática todos los paquetes *JavaScript* que se registran como dependencias en un proyecto o en una aplicación *NODEDJS*, esta herramienta es desarrollada y mantenida por *Twitter* en la actualidad, en los próximos capítulos de este libro se detallaran todos los usos y funciones de *Bower*.

Yarn

Yarn es otra herramienta que también permite al igual que *Bower*, descargar paquetes *JavaScript* desde sus propios repositorios o desde los de *NPM*, pero *Yarn* tiene un comportamiento diferente y más complejo, puesto a que tiene más comandos que el propio *NPM* para usarlos en línea de comandos, utiliza sus propios archivos de bloqueo y ofrece una mejor gestión de paquetes que *NPM*, es mantenida por *Facebook* en la actualidad y es una de las herramientas a las que más se les da uso a la hora de trabajar con proyectos *NODEJS* por parte de los desarrolladores, igualmente describiremos en los próximos capítulos todas las utilidades de *Yarn* con más profundidad.

Capítulo 1: Instalación de *NPM*

Previamente hicimos una pequeñas menciones acerca de la instalación de *NPM*, por ejemplo, en como se realiza la instalación en *Windows*, todos hemos sido usuarios de *Windows* en algún momento de nuestra vida, así que muchos sabemos que lo normal es hacer click en el ejecutable de instalación de *NODEJS* (tal y como lo hacemos con cualquier otro programa), despues click en el boton "siguiente" que aparece en la interfaz que se despliega una y otra vez, aceptamos los términos y condiciones del contrato e "instalar". En *Windows* no existe el ejecutable de instalación de *NPM* como tal, sino que *NPM* se encuentra en conjunto con el ejecutable de instalación de *NODEJS*, así que si vamos a su página oficial[19] podemos encontrar la versión estable v12.16.1 (o superior para cuando usted este leyendo libro) y descargarla (son aproximadamente 20MB para la versión v12.16.1).



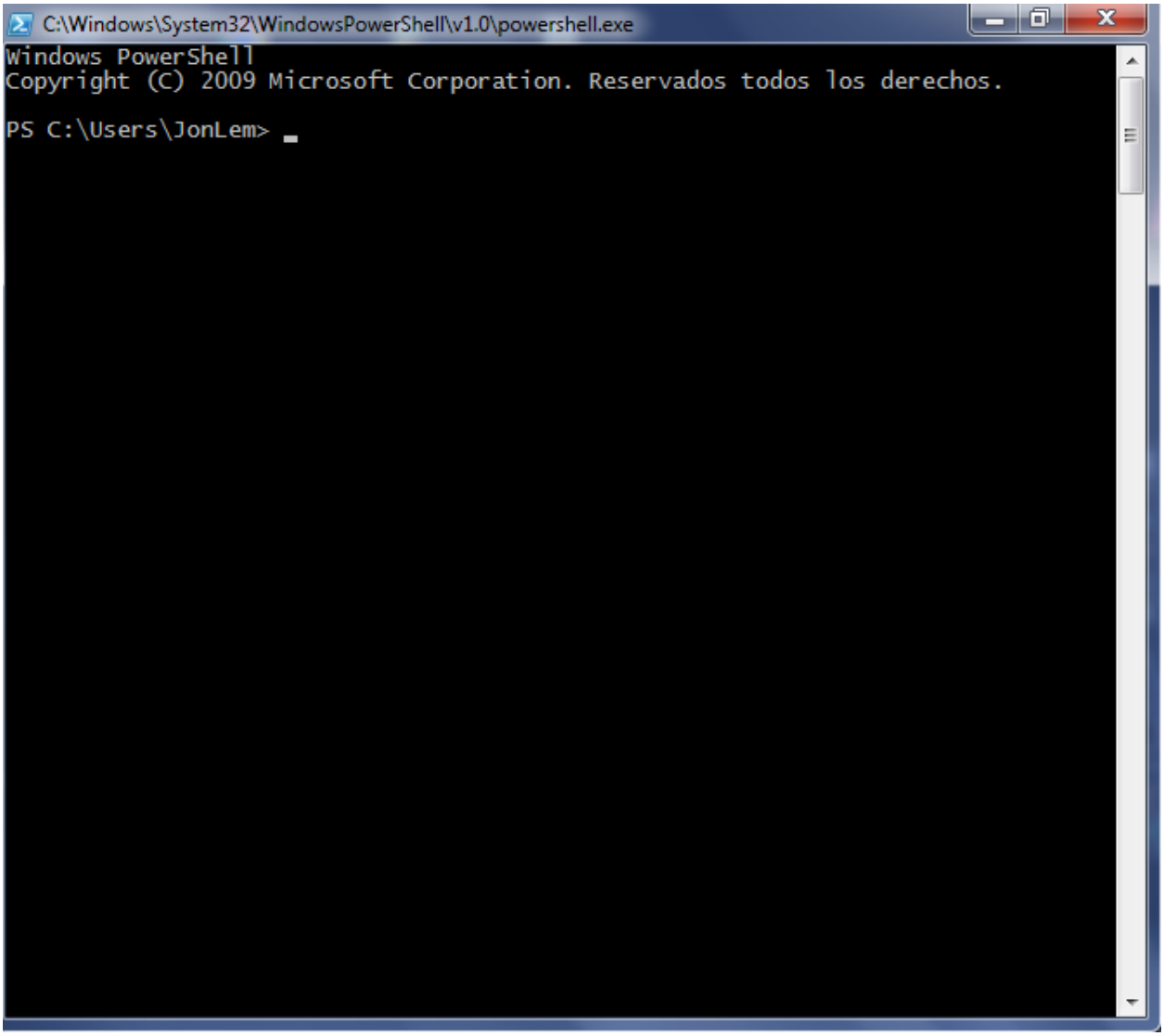
Sin embargo existen diferentes alternativas para instalar *NODEJS* junto a *NPM* desde otras herramientas y recursos. En este capítulo se explicarán los distintos procesos de instalación de *NODEJS* y *NPM* utilizando distintas herramientas de línea de comandos según nuestro sistema operativo.

[19] https://nodejs.org/es/

1.1 Instalación en *Windows*

En *Windows* existe un gestor de paquetes totalmente exclusivo para este sistema operativo que solo se puede utilizar en las versiones de *Windows* 7, 8, 8.1 y 10, su nombre es *Chocolatey*, nos permite instalar programas, paquetes, librerías, etc., por línea de comandos como si de una distribución Linux se tratara (tal y como es en el caso de `apt`).

Chocolatey se instala bajo la *Powershell* de *Windows* con la ayuda de 3 simples comandos (sentencias en términos de programación).



Al tener abierta una sección de powershell, escribimos:

```
> Set-ExecutionPolicy Bypass -Scope Process -Force;

> [System.Net.ServicePointManager]::SecurityProtocol = [System.Net.ServicePointManag\
er]::SecurityProtocol -bor 3072;

> iex ((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/inst\
all.ps1'))
```

Una vez realizado, podra ejecutar el programa *Chocolatey* escribiendo `choco -?` para ver la ayuda que *Chocolatey* nos ofrece, pero solo funcionara bajo la sección de *Powershell*, ya que si sale de la *Powershell* el ejecutable de *Chocolatey* no podrá ser encontrado a pesar de haberse instalado, de otra forma también se puede instalar por *CMD* de *Windows* en una sola línea.

```
> @"%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile -InputFo\
rmat None -ExecutionPolicy Bypass -Command " [System.Net.ServicePointManager]::Secur\
ityProtocol = 3072; iex ((New-Object System.Net.WebClient).DownloadString('https://c\
hocolatey.org/install.ps1'))"
```

Despues de instalarlo por cualquiera de los 2 métodos, añadimos la ruta del ejecutable de *Chocolatey* a la variable de entorno `PATH` del sistema para llamar al ajecutable de *Chocolatey* por línea de comandos.

```
> SET PATH=%PATH%;%ALLUSERSPROFILE%\chocolatey\bin
```

Si no encuentra ningún error, siginifica que todo ha ido bien y *Chocolatey* se ha instalado de forma exitosa. Despues de instalarlo podemos escribir en la terminal lo siguiente:

```
> choco install nodejs-lts
```

Esto instalara la versión estable de *NODEJS* la cual es `v12.16.1` junto a *NPM* en su versión `v6.13.4` o tal vez una versión mayor, pero en el caso de deseemos en algun momento instalar la ultima versión, podremos reemplazar `nodejs-lts` por `nodejs` tal que:

```
> choco install nodejs
```

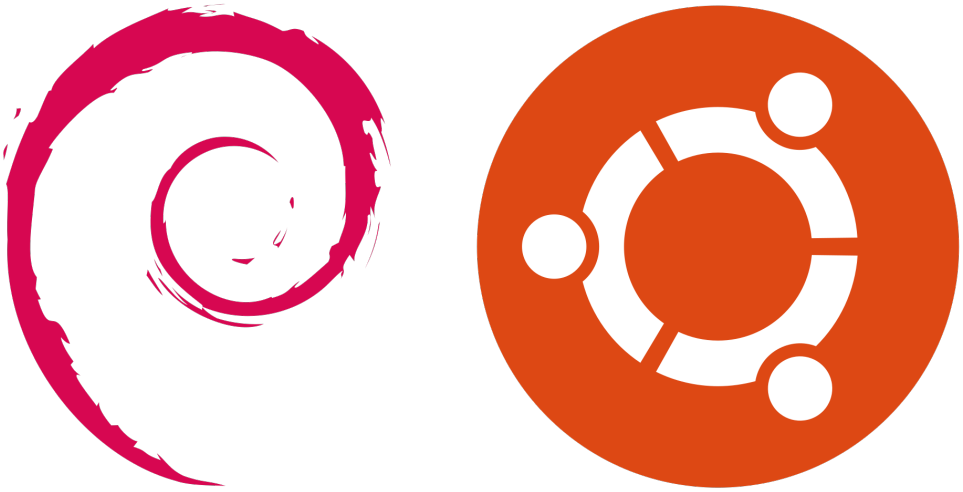
Así obtenemos la ultima versión tanto de *NODEJS* como de *NPM* (aunque no es recomendable para la mayoría de los usuarios ya que son versiones que no han superado todas las pruebas o `tests` en su desarrollo). Finalmente podremos llamar a *NPM* por línea de comandos.

```
> npm -v
6.13.4
```

1.2 Instalación en *GNU/Linux*

La instalación de *NODEJS* y *NPM* en sistemas *GNU/Linux* varían un poco dependiendo de la distro en la que nos encontremos, aunque la instalación siempre se suele ejecutar por línea de comandos, no vamos a obtener la ultima versión del programa, sino más bien, una versión estable antigua para esa distro que es la que se mantiene en los repositorios oficiales.

1.2.1 Debian & Ubuntu



En distros como *Debian* o *Ubuntu* se mantiene una versión antigua pero muy estable para esas distros, que es la versión 4.x de *NODEJS* y 3.x de *NPM* en los repositorios de *Debian* y *Ubuntu*.

```
$ sudo apt update
$ sudo apt install nodejs
$ node -v
v4.2.6

$ npm -v
3.5.2
```

Una solución posible a este problema se descargar el programa *cURL*.

```
$ sudo apt-get update
$ sudo apt-get install curl
```

Una vez instalado podemos agregar el siguiente PPA para obtener la versión LTS actual o la ultima, en este caso vamos agregar la LTS actual que es la v12.16.1.

```
$ cd ~/
$ curl -sL https://deb.nodesource.com/setup_12.x -o nodesource_setup.sh | bash
```

Despues de haber añadido el PPA, podemos instalar el paquete *NODEJS* de la misma manera que lo hicimos anteriormente.

```
$ sudo apt install nodejs
```

Despues verificamos la versión de *NODEJS* y *NPM*.

```
$ node -v
v12.16.1

$ npm -v
6.13.4
```

Para que funcionen algunos paquetes *NPM* (por ejemplo, aquellos que requieren compilar código de origen), deberá instalar el paquete `build-essential`.

```
$ sudo apt install build-essential
```

Finalmente contamos con las herramientas para trabajar con paquetes *NPM* que requieren compilación de código de origen.

Usando `nvm` para la instalación

La segunda solución y mucho más recomendable es utilizar `nvm` en lugar de `apt` la cual su significado es *"Node Version Manager"* (gestor de versiones de node), y la razón por la que es más recomandable es porque al usar un ejecutable de instalación de *NODEJS* como lo es con sistemas *Mac OS X* o *Windows* puede causar errores de permisos al ejecutar paquetes `npm` globales a diferencia de como sería con *NVM* en el que la instalación se realiza a traves de scripts que especifican muy bien la configuración de la instalación y gestión de paquetes en el sistema de archivos. Esto es recomendable hacerlo tanto para sistemas *Windows*, *Linux* y *Mac OS X*.

A continuación se listan los siguientes repositorios oficiales de github para la instalación de *NVM* según nuestro sistema operativo:

Para sistemas *Windows*:

- nodist[20]
- nvm-windows[21]

Para *Linux/Mac OS X*:

- nvm[22]
- n[23]

[20] https://github.com/marcelklehr/nodist
[21] https://github.com/coreybutler/nvm-windows
[22] https://github.com/creationix/nvm
[23] https://github.com/tj/n

NVM nos permite elegir que versión de *NODEJS* instalar en nuestro ordenador, funciona a nivel de un directorio independiente dentro de su directorio de inicio (el directorio principal del usuario actual). Esto significa que puede instalar varias versiones autocontenidas de *NODEJS* sin afectar el sistema operativo. Esto viene muy bien si quieres utilizar una versión LTS o solo la última versión disponible. También es perfecto para que, en el momento en el que estamos desarrollando nuestras aplicaciones o proyectos y estamos trabajando con las ultimas versiones de *NODEJS*, podamos probarlas también con versiones anteriores a ver si son compatibles o no.

Lo primero que debemos hacer es descargar `nvm` usando `wget`. Si no tenemos instalado `wget` en nuestro ordenador podemos hacerlo ejecutando:

```
$ sudo apt update
$ sudo apt install wget
```

Despues de instalarlo, ejecutamos el siguiente comando.

```
$ wget -qO- https://raw.githubusercontent.com/creationix/nvm/v0.34.0/install.sh | ba\
sh
```

O si deseas utilizar `curl` también es posible escribiendo:

```
$ curl -sL https://raw.githubusercontent.com/creationix/nvm/v0.34.0/install.sh | bash
```

Este comando permitirá la instalación de *NVM* agregando un subdirectorio de su directorio de inicio en `~/.nvm`, también agregará las líneas necesarias a su archivo `~/.profile` para usar el archivo, pero para poder ejecutarlo desde la línea de comandos, debemos permitir que *NVM* se use desde el perfil de bash del usuario actual en nuestra sesión.

```
$ source ~/.profile
```

Ya estamos listos para usar *NVM* e instalar las distintas versiones de *NODEJS* y *NPM* en distros *Debian* y *Ubuntu*. Para ver las versiones actuales de *NODEJS* con *NVM* debemos escribir el comando `nvm ls-remote`, esto nos desplegara una lista de todas las versiones de *NODEJS* que queramos instalar desde la web.

```
$ nvm ls-remote

...
...
...
v8.14.0 (LTS: Carbon)
v8.14.1 (LTS: Carbon)
v8.15.0 (LTS: Carbon)
v8.15.1 (LTS: Carbon)
v8.16.1 (LTS: Carbon)
v8.16.2 (LTS: Carbon)
v8.17.0 (Latest LTS: Carbon)
v9.0.0
v9.1.0
v9.2.0
v9.2.1.
v9.3.0
v9.4.0
v9.5.0
v9.6.0
...
...
...
# muchas otras versiones más
```

Al escribir este comando observamos todos los números de versiones de *NODEJS* desde la primera hasta la ultima en la actualidad. Actualmente la ultima versión estable es la `v12.16.1` con el nombre *Erbium* para instalarla con *NVM* basta con escribir `nvm install 12.16.1` esto iniciará la descarga de esa y solo esa versión específica de *NODEJS* y la colocara en los subdirectorios de `~/.nvm` para llevar un control más organizado de las versiones que estamos utilizando de *NODEJS* y *NPM*.

Al escribir en nuestra terminal `node` observamos que *NODEJS* ha sido instalado exitosamente y que esta siendo ejecutado sin problemas.

```
$ node
Welcome to Node.js v12.16.1.
Type ".help" for more information.
>
```

Y al llamar a `npm`.

```
$ npm -v
6.13.4
```

Si en algun momento deseamos utilizar una versión distinta de *NODEJS* y *NPM* ya sea mayor o menor a la que tenemos seleccionada actualmente, basta con volver a ejecutar el comando `nvm ls-remote` y seleccionar la versión que queramos instalar, por ejemplo la versión `v10.13.0` que es la primera versión estable de la serie *Dubnium* de *NODEJS*.

```
$ nvm install 10.13.0
```

Ya tenemos instaladas 2 versiones de *NODEJS*, `v12.16.1` y `v10.13.0`, para cambiarnos de la versión que tenemos configurada para que arranque *NODEJS* a la versión que recien acabamos de instalar simplemente debemos escibir en la terminal el comando `nvm use` seguido del número de versión que queramos usar a continuación, en este caso `v10.13.0`.

```
$ nvm use 10.13.0
```

Al comprobar el número de versión de *NODEJS* y *NPM*.

```
$ node -v
v10.13.0

$ npm -v
6.4.1
```

1.3 Instalación en *Mac OS*

La instalación de *NODEJS* y *NPM* se pueden realizar de la misma forma que como se hace con *Windows*, es decir, visitamos su página oficial y descargamos el ejecutable. Sin ambargo, existe otro metodo por el cual podemos instalar *NODEJS* y *NPM* y no solo estos 2 paquetes sino muchisimos paquetes más, estamos hablando de *Homebrew*.

Homebrew es un gestor de paquetes para sistemas *Mac* que al igual que *APT* y *Chocolatey*, permite la instalación de paquetes totalmente exclusivos para sistemas operativos *Mac OS* directamente por línea de comandos, la instalación tambien se realiza por línea de comandos utilizando `bash` y `curl` de la siguiente manera:

```
$ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/mast\
er/install.sh)"
```

También es posible descargar *Homebrew* con `ruby` escribiendo:

```
$ /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/\
master/install)"
```

Una vez instalado podemos ejecutar el siguiente comando para instalar *NODEJS* y *NPM*.

```
$ brew install node
```

En esta ocasión con solo escribir `node` también nos vendra con el ejecutable de *NPM* en su ultima versión, para verificarla solo escribimos `node -v` o `npm -v`.

Si en algun momento deseamos actualizar *NODEJS*, *NPM* deberemos tener actualizado *Homebrew* y su lista de repositorios escribiendo:

```
$ brew update
```

posteriormente:

```
$ brew upgrade node
```

De esta forma conseguimos que *NODEJS* y *NPM* se actualizen a su ultima versión a la que teniamos anteriormente en nuestra máquina.

Capítulo 2: Empezando con `npm`

En este capítulo vamos a ver todo lo necesario en sus primeros pasos para familiarizarnos con *NPM* poco a poco y por nuestra propia cuenta, haremos menciones acerca de sus comandos, número de argumentos, opciones de los comandos más necesarios, archivos que se utilizan y demás.

Una vez que tenemos instalado *NODEJS* y *NPM* en nuestro ordenador, ya podemos empezar a trabajar con proyectos *JavaScript* del lado del servidor, pero para hacerlo necesitaremos tomar en cuenta que actividades vamos a realizar en nuestra aplicación?, que recursos y/o paquetes deberemos incorporar en los proyectos de nuestras aplicaciones para hacer que funcionen correctamente?. Todos y cada uno de estos recursos los podremos obtener con *NPM* (en cuanto a código *JavaScript* se trate) e incorporarlos en nuestros proyectos de tal manera que queden ser registrados en una archivo especial (el cual explicaremos más adelante) y así no tener que escribir a mano cada uno de los recursos y **dependencias** de nuestro proyecto y descargarlos uno por uno, *NPM* optimiza todo este trabajo por nosotros permitiendonos desarrollar aplicaciones en la menor cantidad de tiempo posible.

2.1 llamando a *NPM* desde linea de comandos

Llego la hora de usar *NPM* por línea de comandos, para ello simplemente escribimos el comando `npm` en cualquier ruta del sistema en la que nos encontremos.

```
$ npm
```

Al pulsar la tecla `Enter` o `return` obtenemos la siguiente salida:

```
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

Specify configs in the ini-formatted file:
    C:\Users\JonLem\.npmrc
or on the command line via: npm <command> --key value
Config info can be viewed via: npm help config

npm@6.13.4 C:\nodejs\node_modules\npm
```

Por defecto el comando `npm` no incorpora las opciones de ayuda popularmente conocidas como los son `-h` y `--help`, así que si escribimos en la terminal `npm -h` o `npm --help`, basicamente obtendremos el mismo resultado que obtuvimos anteriormente.

Sin embargo `npm` ofrece una opción `-l` que nos permite ver de manera detallada y más desplegada, la información de todos los comandos disponibles para su uso, así que al escribir `npm -l` obtenemos los siguiente:

```
Usage: npm <command>

where <command> is one of:

    access          npm access public [<package>]
                    npm access restricted [<package>]
                    npm access grant <read-only|read-write> <scope:team> [<package>]
                    npm access revoke <scope:team> [<package>]
                    npm access 2fa-required [<package>]
                    npm access 2fa-not-required [<package>]
                    npm access ls-packages [<user>|<scope>|<scope:team>]
                    npm access ls-collaborators [<package> [<user>]]
                    npm access edit [<package>]

    adduser         npm adduser [--registry=url] [--scope=@orgname] [--auth-type=leg\
acy] [--always-auth]

                    aliases: login, add-user

    audit
                    npm audit [--json] [--production]
```

```
                    npm audit fix [--force|--package-lock-only|--dry-run|--productio\
n|--only=(dev|prod)]

    bin             npm bin [--global]

    bugs            npm bugs [<pkgname>]

...
...
...
```

Gracias a esto, podemos tener una idea más clara de que argumentos recibe un comando en específico y qué opciones podemos utlizar para ciertas tareas que vayamos a necesitar en nuestros proyectos.

2.1.1 Localizando el ejecutable con `where` o `which`

Localizar el ejecutable de `npm` es muy facil, dependiendo de nuestro sistema operativo podemos usar o uno o el otro. Si nos encontramos trabajando en un sistema tipo *Windows*, usando `where` lo localizamos inmediatamente, de tal manera que escribimos:

```
> where npm
```

Se nos despliega la siguiente salida:

```
C:\nodejs\npm
C:\nodejs\npm.cmd
```

Encontramos cada una de las rutas del sistema en donde se encuentra un archivo llamado `npm`. El archivo que se ejecuta, es el que termina con la extensión `.cmd`, esto es debido a que el archivo `npm` sin extensión, es un archivo de scripting de línea de comandos especial para sistemas *Unix/Linux* ejecutable bajo el lenguaje de comandos *Bash*, del mismo modo, con el archivo `npm.cmd` que es un archivo de scripting de *Windows* que funciona bajo el lenguaje *Batch*.

En sistemas tipo *Linux* usamos `which`:

```
$ which npm
```

Debemos tomar en cuenta que ambos programas `where` y `which` solo ejecutaran su busqueda a partir de las rutas escritas en la variable de entorno `PATH` de nuestro sistema, así que si buscamos por un archivo que no se encuentra en ninguna de las rutas nos dara un error.

2.2 Lista de comandos de npm (API)

NPM posee una gran lista de comandos para realizar diferentes actividades en su ejecución, de los cuales podemos nombrar las más necesarias y ultilizadas por los desarrolladores como: **instalar** paquetes, **actualizar** paquetes, **iniciar** un proyecto, **ejecutar** scripts, **publicar** paquetes en la plataforma online de *NPM*, entre muchas otras más, en los proximos capítulos entraremos más en detalle acerca de la funcionabililidad de cada uno de los comandos de *NPM*.

En la versión `6.13.4` tenemos a nuestra disposición los siguientes comandos para que podamos hacer uso de ellos en nuestros proyectos (de la A a la Z):

- `access`
- `adduser`
- `audit`
- `bin`
- `bugs`
- `cache`
- `ci`
- `completion`
- `config`
- `dedupe`
- `deprecate`
- `dist-tag`
- `docs`
- `doctor`
- `edit`
- `explore`
- `fund`
- `get`
- `help`
- `help-search`
- `hook`
- `init`
- `install`
- `install-ci-test`
- `install-test`
- `link`
- `logout`
- `ls`
- `org`
- `outdated`
- `owner`

- pack
- ping
- prefix
- profile
- prune
- publish
- rebuild
- restart
- root
- run-script
- search
- set
- shrinkwrap
- star
- stars
- start
- stop
- team
- test
- token
- uninstall
- unpublish
- update
- version
- view
- whoami

Para un total de 57 comandos en la versión v6.13.4 de *NPM*.

2.2.1 Comandos con aliases

Algunos comandos se pueden ejecutar con un nombre distinto y/o más corto para que se nos sea más comodo trabajar con *NPM*, me refiero a que en lugar de usar palabras largas (y que de allí empecemos a equivocarnos) usamos abreviaciones, estos son los **alias**, una manera más sencilla de trabajar sin que genere errores.

Un ejemplo de esto lo podemos encontrar con el comando `install` para descargar e instalar un paquete, si por ejemplo escribimos `npm install` pero llegamos a equivocarnos con una sola letra o que nos falte una (por ejemplo que escribamos `instal` que es incorrecto), pues lo más lógico seria un error y *NPM*, gracias a su inteligencia nos deplegará un mensaje de la siguiente manera ofreciendonos sugerencias de los comandos que podrían coincidir con lo que nosotros escribimos:

```
Did you mean one of these?
    install
    unstar
    init

>
```

En el caso de `install` contamos con 3 aliases que son:

- `isntall`: Que es como si escribieramos `install` pero equivocandonos con las 'S' y la 'N'.
- `add`: Por añadir un sinónimo.
- `i`: Para no escribir el comando `install`, ya que cuando lo ejecutamos muchas veces se vuelve algo engorrozo.

Estos aliases funcionan de la misma manera que con el comando original `install`, podemos considerarlo como un accesso corto y directo a la hora de instalar paquetes. Si quisieramos instalar algún paquete o librería por ejemplo *jQuery*, usando los aliases escribiriamos:

```
> npm i jquery
```

Habriamos obtenido el mismo resultado si lo hubiesemos hecho con `install`, a continuación se muestra una tabla donde aparecen todos los comandos de *NPM* que usan **aliases**.

Nº	Comando	Nº de aliases	Aliases
1	adduser	2	login, add-user
2	bugs	1	issues
3	config	1	c
4	dedupe	2	ddp, find-dupes
5	dist-tag	1	dist-tags
6	docs	1	home
7	init	2	create, innit
8	install	3	add, isntall, i
9	install-ci-test	1	cit
10	install-test	1	it
11	link	1	ln
12	ls	3	list, la, ll
13	owner	1	author
14	rebuild	1	rb
15	run-script	3	run, rum, urn
16	search	3	s, se, find
17	star	1	unstar
18	test	2	tst, t
19	unistall	5	un, unlink, remove, rm, r
20	update	3	up, upgrade, udpate
21	view	3	v, info, show

2.2.2 Ayuda adicional para cada comando

Si necesitamos información detallada de algún comando en específico podremos obtenerla escribiendo `npm help` seguido del nombre del comando, por ejemplo: `npm help install`. Esto lo que hara es que abrira nuestro navegador web proporcionandonos un archivo `.html` conteniendo información sobre el comando que especificamos (en este caso `install`), esto también es posible utilizando los aliases de cada uno de los comandos.

La documentación por defecto se encuentra en inglés, además de eso viene solo con información muy resumida puesto que esto no es suficiente para entender la funcionabililidad de los comandos y dificulta un poco nuestro apendizaje. En los próximos capítulos profundizaremos detalladamente la usabilidad de cada uno de los comandos y mostraremos ejemplos de uso con sus respectivas opciones.

2.3 Archivos utilizados por *NPM*

Para que *NPM* pueda demostrar su maximo potencial y eficiencia, se necesitan unos pequeños pero muy poderosos archivos que siempre forman parte de cada uno de los paquetes que descargamos cuando ejecutamos el comando `install`, con el fin de proveer información sobre el autor(es), versión, dependencias, scripts, directorios, archivo de arranque, etc. del paquete o los paquetes que obtuvimos tras la ejecución de `install`. Estamos hablando del archivo `package.json`, este es un archivo JSON[24] que permite almacenar toda esta información, vamos a entrar en detalle sobre el a continuación.

2.3.1 Archivo `package.json`

`package.json` es un archivo que siempre se encuentra en todas las librerías y proyectos *JavaScript* que utilizan *NODEJS* y que se descargan con *NPM* para poder funcionar. Un ejemplo de un archivo de `package.json` puede ser el siguiente:

Ejemplo de package.json

```
{
  "name": "mi-primer-paquete",
  "version": "1.0.0",
  "description": "Un paquete de ejemplo para \"Un librito sobre NPM\".",
  "main": "./lib/index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "repository": {
    "type": "git",
    "url": "git+https://github.com/JonLemOfficial/mi-primer-paquete.git"
  },
```

[24] https://es.wikipedia.org/JSON

```
13    "keywords": [
14      "palabras",
15      "clave",
16      "del",
17      "paquete"
18    ],
19    "author": "Jonathan \"JonLem\" Lemos",
20    "license": "MIT",
21    "bugs": {
22      "url": "https://github.com/JonLemOfficial/mi-primer-paquete/issues"
23    },
24    "homepage": "https://github.com/JonLemOfficial/mi-primer-paquete#readme",
25  }
```

Como se puede observar este archivo posee información muy básica pero suficiente con respecto a nuestro paquete o proyecto, tiene información con respecto al autor(es) del proyecto, número de versión, nombre del paquete, descripción, licensia, palabras clave, entre muchos otros atributos más. Todos estos atributos tienen un proposito o utilidad y suelen estar siempre presente en `package.json` tras la descarga de un paquete o al iniciar cualquier proyecto que queramos en nuestra máquina local. Sin embargo existen otros atributos que tambíen forman parte de `package.json` y que se incorporan en nuestro archivo a medida que nuestro proyecto crece e interactuamos con el, estos son: `dependencies`, `devDependencies`, `peerDependencies`, `bundledDependencies`, `optionalDependencies`. Hablaremos más en detalle sobre cómo estos atributos se incorporan en nuestro archivo en el próximo capítulo.

En resúmen, esta es la configuración básica de un `package.json`, a medida que vamos realizando más operaciones y configuraciones en nuestro proyecto con los comandos de *NPM*, se han de añadir de forma automatica nuevos atributos, unos de los cuales se podría nombrar sería el atributo `bin` el cual su función es localizar el archivo `*.js` en algunos de los sub-directorios de nuestro proyecto y tratarlo como si de un archivo ejecutable se tratara para llamarlo por linea de comandos. En los proximos capitulos veremos cómo `package.json` se modifica en nuestro paquete `mi-primer-paquete` a medida que realizamos ciertas acciones que mencionaremos en este libro.

2.3.2 Carpeta `node_modules/`

Esta es la carpeta principal en donde se encontrarán almacenados todos los paquetes *JavaScript* que fueron descargados e incorporados al proyecto en el que estamos trabajando, dicha carpeta es automáticamente creada tras el uso del comando `npm install`. Una forma muy breve de explicar su aparición tras la creación del paquete `mi-primer-paquete` es la siguiente:

1. Empezamos con una estructura básica

Estructura de directorios del paquete 'mi-primer-paquete'

```
mi-primer-paquete/
├── lib/
│   └── index.js
├── package.json
└── README.md
```

2. Colocarse en el arbol principal de directorios del paquete y ejecutar el comando `npm install` seguido de cualquier nombre de paquete como por ejemplo *jQuery* de la siguiente forma:

```
C:\...\> cd .\mi-primer-paquete
C:\...\mi-primer-paquete> npm install jquery
```

3. Se crea la carpeta `node_modules/` conteniendo los paquetes que incorporamos a nuestro proyecto debido a la ejecución del comando `install`, en este caso *jQuery*. Posteriormente la estructura de directorios de `mi-primer-paquete` se ve alterada de la siguiente forma:

Estructura modificada del paquete 'mi-primer-paquete'

```
mi-primer-paquete/
├── lib/
│   └── index.js
+ ├── node_modules/
+ │   └── jquery/
+ ├── package.json
+ ├── package-lock.json
└── README.md
```

Como se puede observar, ahora se incorpora la carpeta `node_modules/` y se crea automáticamente el archivo `package-lock.json`, cabe mencionar también que tras la instalación de *jQuery* en nuestro proyecto, el archivo `package.json` se vera modificado añadiendo a *jQuery* en la lista de dependencias de la siguiente forma:

```
"dependencies": {
  "jquery": "^3.4.1"
}
```

Y al agregarse al contenido de `package.json` se ve tal que:

package.json con dependencias

```
{
  "name": "mi-primer-paquete",
  "version": "1.0.0",
  "description": "Un paquete de ejemplo para \"Un librito sobre NPM\".",
  "main": "./lib/index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "repository": {
    "type": "git",
    "url": "git+https://github.com/JonLemOfficial/mi-primer-paquete.git"
  },
  "keywords": [
    "palabras",
    "clave",
    "del",
    "paquete"
  ],
  "author": "Jonathan \"JonLem\" Lemos",
  "license": "MIT",
  "bugs": {
    "url": "https://github.com/JonLemOfficial/mi-primer-paquete/issues"
  },
  "homepage": "https://github.com/JonLemOfficial/mi-primer-paquete#readme",
  "dependencies": {
    "jquery": "^3.4.1"
  }
}
```

2.3.3 Archivo `package-lock.json`

Posteriormente nos encontramos con el archivo adicional `package-lock.json`, este archivo es generado de manera automática a medída que estamos interactuando con ordenes de línea de comandos de *NPM* despues de iniciar un proyecto que contenga el archivo `package.json`, en donde alguna orden de *NPM* que deba realizar modificaciones en `package.json` o la carpeta `node_modules/` creará este archivo automáticamente. `package-lock.json` no puede ser modificado manualmente porque su contenido es gestionado por *NPM* y presenta toda la informacion individual de todos y cada uno de los paquetes que fueron añadidos al proyecto en el que estamos trabajando con la orden `npm install ...`. toda esta información está directamente vinculada con el registro de paquetes (o repositorio) de la plataforma de *NPM* el cual es manejada de manera cifrada de punto a punto con

algoritmos espciales para llevar un control más exacto de la gestión de nuestros proyectos y tratar de evitar con mayor eficacia posibles conflictos.

Un archivo `package-lock.json` sencillo puede ser el siguiente:

Ejemplo de package-lock.json

```
{
  "name": "mi-primer-paquete",
  "version": "1.0.0",
  "lockfileVersion": 1,
  "requires": true,
  "dependencies": {
    "jquery": {
      "version": "3.4.1",
      "resolved": "https://registry.npmjs.org/jquery/-/jquery-3.4.1.tgz",
      "integrity": "sha512-36+AdBzCL+y6qjw5Tx7HgzeGCzC81MDDgaUP8ld2zhx58HdqXGoBd+tHd\
rBMiyjGQs0Hxs/MLZTu/eHNJJuWPw=="
    }
  }
}
```

En este archivo encontramos atributos muy básicos de cada uno de los paquetes añadidos al proyecto por causa de la ejecución de `npm install [...]`. Todos los paquetes instalados se localizan dentro del atributo `dependencies`, `devDependencies`, `optionalDependencies`, `peerDependencies` o `bundledDependencies` dependiendo de que opción hayamos utilizado para indexarlas (`--save-dev`, `--save-opt`, `--no-save`) etc.

En este caso tenemos el paquete *jQuery* que se encuentra dentro de `dependencies`, al revisarlo nos ofrece información acerca de su número de versión (`version`), localización en el repositorio de la plataforma de *NPM* (`resolved`) y cifrado (`integrity`) para la comprobación de integridad del archivo en la plataforma de *NPM*. Esta información es necesaria tenerla a la mano porque nos permite ubicar con mayor exactitud y rapidez los paquetes requeridos para que nuestro proyecto pueda ejecutarse de manera exitosa y sin problemas y al momento de plublicarse en los repositiorios (de la plataforma de *NPM*) solo deba subirse la información (`package.json` y package-lock.json) y no el directorio `node_modules/` entero.

Capítulo 3: Descarga e instalación de paquetes

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

3.1 Uso del comando `install`

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

3.1.1 Sinopsis de `install`

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

3.1.2 Instalar paquetes de forma típica

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

Utilizando Ambitos (`<@scope>`)

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

3.1.3 Instalar paquetes desde un tarball (`.tgz`)

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

3.1.4 Instalar paquetes desde un repositorio de Github

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

3.2 Opciones del comando `install`

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

3.2.1 Opción `--save`

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

3.2.2 Opción `--save-dev` o `-D`

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

3.2.3 Opción `--save-prod` o `-P`

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

3.2.4 Opción `--save-opt` o `-O`

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

3.2.5 Opción `--no-save`

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

3.2.6 Opción `--save-exact` o `-E`

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

3.2.7 Opción `--save-bundle` o `-B`

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

3.2.8 Opción `--global`

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

3.3 Abreviando con aliases

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

Capítulo 4: Ejecución de Scripts en linea de comandos

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

4.1 Uso del comado `run-script`

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

4.2 Abreviando con aliases

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

Capítulo 5: Creación de paquetes en modo local

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

5.1 Uso del comando `init`

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

5.2 Opciones del comando `init`

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

Capítulo 6: Publicación de paquetes en `npmjs.com`

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

6.1 Uso del comando `publish`

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

6.1 Opciones del comando `publish`

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

Capítulo 7: Un vistazo al website

`npmjs.com`

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

7.1 Busqueda de paquetes

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

7.1.1 Uso de palabras clave (`keywords`)

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

7.2 Registro de usuario e inicio de sesión

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

7.3 Adminitración del `dashboard` del usuario

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

Capítulo 8: Herramienta adicional *Bower*

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

8.1 Como funciona *Bower*?

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

8.1.1 Sistema de archivos utilizados por *Bower*

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

8.1.2 Lista de Comandos (`API`)

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

8.2 Deferencias y comparaciones entre *Bower* y *NPM*

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

8.3 Ventajas de usar de *Bower*

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

8.4 Cuando y para que tipo de proyectos usar *Bower*?

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

8.5 Un vistazo al website de *Bower*

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

Capítulo 9: Herramienta adicional *Yarn*

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

9.1 Como funciona *Yarn*?

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

9.1.1 Sistema de archivos utilizados por *Yarn*

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

9.1.2 Lista de Comandos (`API`)

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

9.2 Deferencias y comparaciones entre *Yarn* y *NPM*

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

9.3 Ventajas de usar de *Yarn*

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

9.4 Cuando y para que tipo de proyectos usar *Yarn*?

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

8.5 Un vistazo al website de *Bower*

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

Conclusión

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

Anexo: Contenido adicional

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.

Glosario de términos

Este contenido no está disponible en el libro de muestra. El libro se puede comprar en Leanpub en http://leanpub.com/librito-sobre-npm.