

El objetivo de esta práctica es instalar los requisitos necesarios para seguir el curso, esto es, **Justo**.

Al finalizar la lección, el estudiante:

- Habrá instalado **Justo.js**.
- Habrá creado los archivos requeridos por un proyecto **Justo**, tanto independiente como integrado en otro.

Requisitos de las prácticas

Las prácticas se pueden seguir tanto en sistemas **Windows** como **Linux** como, por ejemplo, **Debian**, **Ubuntu** o **Raspbian**.

Instalación de Node.js

Lo primero es instalar las dependencias. La única dependencia que tiene **Justo.js** es **Node.js** y su administrador de paquetes **NPM**. Se debe tener instalada la versión **5** o superior, la cual se puede descargar de su sitio web oficial, nodejs.org.

En entornos **Windows**, tanto **Node** como **NPM** se instalan fácilmente, mediante el instalador **.msi** que se puede descargar del sitio oficial de **Node**. En el caso de **Linux**, es necesario instalar los paquetes **nodejs** y **npm**, por ejemplo, mediante **APT**. Si el usuario usa una **Raspberry Pi**, bajo una distribución **Debian** como **Raspbian** o **Ubuntu MATE**, hay que decir que no suele haber repositorios con las últimas versiones de **Node** y **NPM** para su instalación con **APT**. A pesar de todo, es muy fácil instalar **Node** y **NPM** a mano.

Instalación de Justo.js

Una vez instalado **Node.js** y **NPM**, podemos pasar a instalar **Justo.js**. Para ello, usaremos el comando **npm**:

1. Abrir una consola de **PowerShell** o terminal *shell* en **Linux**.
De ahora en adelante, nos referiremos a ella como simplemente *consola*. Por otra parte, mostraremos los resultados en **Windows**. Pero en entornos **Linux**, los comandos a ejecutar son exactamente los mismos.
2. Instalar **justo-cli** globalmente en la máquina, mediante **npm**:

```
> npm install -g justo-cli
```
3. Comprobar que tenemos acceso a la utilidad **justo**:

```
> justo -h
```

Justo.js en un proyecto independiente

Una vez instalada la utilidad **justo**, vamos a ver cómo crear los archivos **Justo.json**, **Justo.js** y **package.json**.

Recordemos que existe dos tipos de proyectos donde podemos usar **Justo.js**: un **proyecto independiente** (*standalone project*), autónomo, dedicado exclusivamente a tareas de **Justo** como, por ejemplo, la instalación o configuración de software. Mientras que un **proyecto integrado** (*embedded project*) es aquel que se integra en otro, dedicándose la parte de **Justo** a automatizar ciertas tareas del proyecto principal.

Independientemente de cómo usemos **Justo** en el proyecto, la manera más fácil de crear los archivos requeridos, es mediante el generador **justo-generator-justo**:

1. Ir a la consola.

2. Instalar globalmente el generador `justo-generator-justo`:

```
> npm install -g justo-generator-justo
```
3. Comprobar que el generador se ha instalado correctamente. Para ello, mostrar su ayuda:

```
> justo -g justo help
```

Para invocar un generador, hay que utilizar la opción `-g` de `justo` y el nombre del generador sin el prefijo `justo-generator-`.
4. Crear el directorio de la práctica.
5. Generar los archivos `package.json`, `Justo.json` y `Justo.js` mediante el generador:

```
> justo -g justo
```

El generador recolecta información sobre el proyecto `Justo` a crear, antes de crear sus archivos. Indique `standalone` como tipo de proyecto y responda a las preguntas que le sean formuladas. No se preocupe, puede inventárselas, pero sea creativo.

Una vez recolectada la información, el generador creará los archivos teniendo en cuenta lo respondido por el usuario.
6. Listar el contenido del directorio:

```
> dir
```

En proyectos independientes, el generador crea otros archivos, generalmente, muy útiles como, por ejemplo, `.editorconfig`, `.gitignore`, etc.
7. Mostrar el contenido de los archivos:

```
> cat package.json
> cat Justo.json
> cat Justo.js
```

En la lección siguiente, comenzaremos a trabajar con tareas y su registro en el archivo `Justo.js`.

Justo.js en un proyecto integrado

Ahora, vamos a suponer que usaremos `Justo.js` para automatizar determinadas tareas de otro proyecto. En este caso, el proceso es similar al anterior, usaremos el generador `justo-generator-justo`, pero no indicaremos `standalone` como tipo de proyecto:

1. Ir a la consola.
2. Vaciar el contenido del directorio de la práctica.
3. Invocar el generador:

```
> justo -g justo
```

Ahora, *no* seleccione `standalone` como tipo de proyecto.
4. Listar el contenido del directorio.

Observe que algunos archivos, generados en la sección anterior, no se han creado en este tipo de proyecto. Generalmente, cuando se usa `Justo` en otro proyecto, primero, se crea la estructura del principal y después los archivos de `Justo`. Sólo hay que recordar que si el proyecto principal ya ha creado el archivo `package.json`, habrá que indicarle al generador que *no* lo genere, porque si lo hace, sobrescribirá su contenido. En este caso concreto, habrá que añadir el paquete `justo` como dependencia de desarrollo en la propiedad `devDependencies` del archivo `package.json`.

También hay que decir que otros generadores como, por ejemplo, `justo-generator-express` para la creación de aplicaciones `Express`, ya crean los archivos de `Justo`. Por lo que no habrá que utilizar el generador `justo-generator-justo`.

5. Mostrar el contenido de los archivos `package.json`, `Justo.json` y `Justo.js`.
6. Cerrar la consola.

En la siguiente lección, empezaremos a trabajar con `Justo`, creando tareas e invocándolas. Por ahora, nos quedamos en su proceso de instalación y la utilización del generador `justo-generator-justo`.