

EXERCISES QUE TRABAJAREMOS EN LA CUE

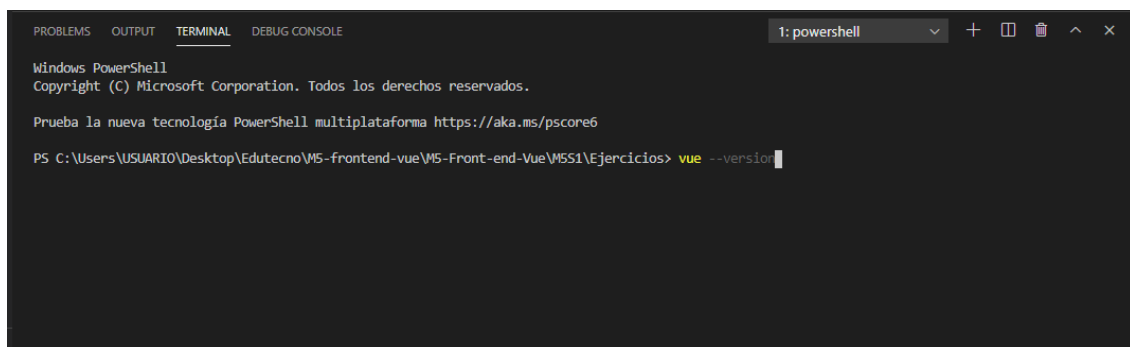
- EXERCISE 1: INTALANDO JEST Y VUE TEST UTILS POR VUE/CLI.
- EXERCISE 2: INTALANDO JEST PARA PROYECTO JS.

EXERCISE 1: INSTALANDO JEST Y VUE TEST UTILS POR VUE/CLI

INTRODUCCIÓN

Usando la interfaz de línea de comandos (command line interface o CLI), **Vue** nos permite iniciar proyectos empleando todas las herramientas necesarias para desarrollar y testear. De esta forma, podemos agregar las dependencias necesarias que se ocupan en sus proyectos, tales como: **Vue-Test-Utils**, **Jest**, **Vue router**, **vuex**, entre otros, con solo seleccionar unas opciones.

Para comenzar un proyecto con **Vue**, previamente debemos tener instalado **Vue-CLI**. En el Drill anterior ya lo hicimos, así que vamos a abrir nuestro editor de texto, y desde ahí la terminal. Vamos a escribir:



```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
1: powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\USUARIO\Desktop\EduTecno\MS-frontend-vue\MS-Front-end-Vue\MSS1\Ejercicios> vue --version
```

Si la instalación fue correcta, ahora podremos ver la versión instalada.



```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE 1: powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\USUARIO\Desktop\Edutecno\VS-front-end-vue\VS-Front-end-Vue\VS1\Ejercicios> vue --version
@vue/cli 4.5.12
PS C:\Users\USUARIO\Desktop\Edutecno\VS-front-end-vue\VS-Front-end-Vue\VS1\Ejercicios>
```

UTILIZANDO VUE/CLI PARA CREAR PROYECTO

Para crear un nuevo proyecto, debemos posicionarnos en la carpeta donde queremos trabajar, y escribir en la terminal el comando que sigue: **“vue create nombre_proyecto”**. Aquí debes reemplazar “nombre_proyecto”, por el que le quieras dar.

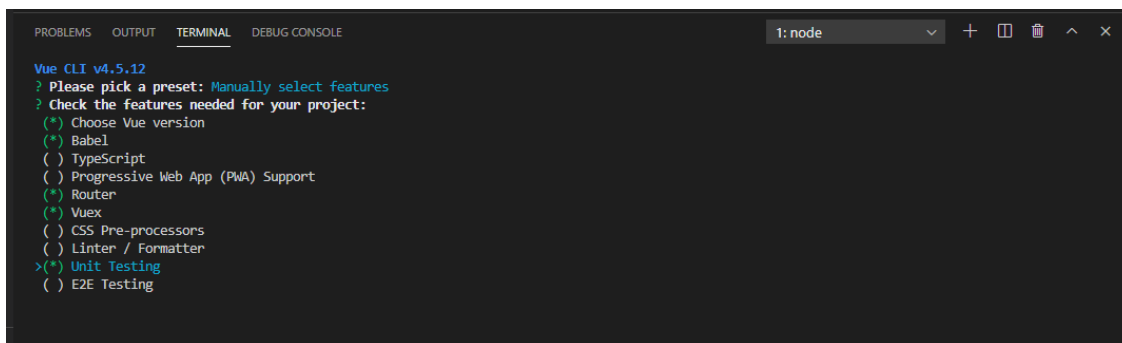
```
vue create nombre_proyecto
```

Cuando tecleamos “enter”, comienzan a aparecer diferentes vistas. En la primera vamos a elegir, utilizando para ello las flechas del teclado arriba y abajo, **“Manually select features”**, así podremos seleccionar según nuestras necesidades, las herramientas que instalaremos y, de nuevo “enter”.



```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE 1: node
Vue CLI v4.5.12
? Please pick a preset:
  Default ([Vue 2] babel, eslint)
  Default (Vue 3 Preview) ([Vue 3] babel, eslint)
> Manually select features
```

Ahora, para seleccionar una dependencia a instalar, en la vista que sigue debemos presionar la tecla “espacio”. En este caso, agregaremos **Babel**, para poder transformar nuestro código JS a versiones más antiguas, compatibles con todos los navegadores, **Router**, para generar las rutas a las diferentes secciones, **Vuex**, para administrar los estados, y **Unit Testing**, para poder realizar pruebas unitarias. Una vez marcadas nuestras preferencias, tecleamos “enter”.



```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE 1: node
Vue CLI v4.5.12
? Please pick a preset: Manually select features
? Check the features needed for your project:
(*) Choose Vue version
(*) Babel
(*) TypeScript
(*) Progressive Web App (PWA) Support
(*) Router
(*) Vuex
(*) CSS Pre-processors
(*) Linter / Formatter
>(*) Unit Testing
(*) E2E Testing
```

En la vista que sigue, vamos a seleccionar la versión de **“vue 2.x”**, y enseguida oprimimos “enter”.

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
1: node

Vue CLI v4.5.12
? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel, Router, Vuex, Unit
? Choose a version of Vue.js that you want to start the project with (Use arrow keys)
> 2.x
  3.x (Preview)
```

Allí nos preguntará: **“Use history mode for router?”**, le escribimos **“Y”**, y nuevamente tecla **“enter”**.

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
1: node

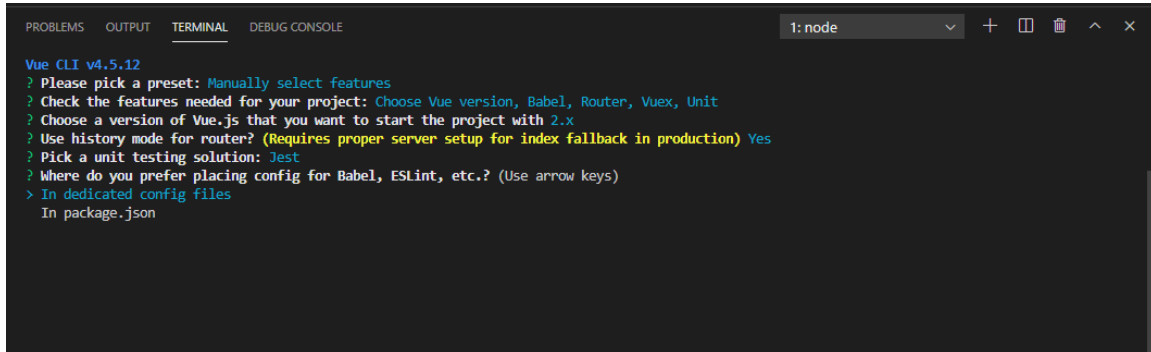
Vue CLI v4.5.12
? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel, Router, Vuex, Unit
? Choose a version of Vue.js that you want to start the project with 2.x
? Use history mode for router? (Requires proper server setup for index fallback in production) (Y/n) |
```

En la siguiente pregunta, nos dan a elegir entre usar **Mocha + Chai** o **Jest**, para realizar testing. Seleccionaremos la segunda opción para realizar nuestras pruebas unitarias, y tecleamos **“enter”**.

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
1: node

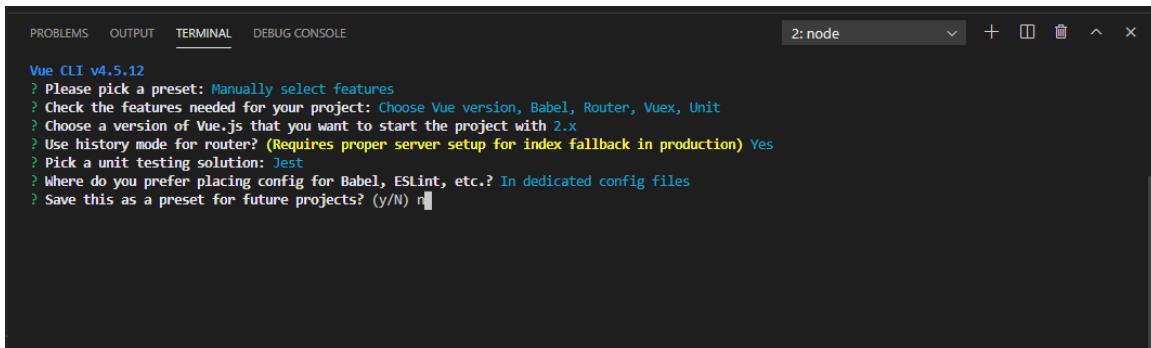
Vue CLI v4.5.12
? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel, Router, Vuex, Unit
? Choose a version of Vue.js that you want to start the project with 2.x
? Use history mode for router? (Requires proper server setup for index fallback in production) Yes
? Pick a unit testing solution:
  Mocha + Chai
> Jest
```

En la vista que sigue, nos preguntará donde preferimos colocar los archivos de configuración para **Babel** y otros, seleccionamos la alternativa **"In dedicated config files"**, y así cada uno tendrá su archivo propio de configuración personalizado, y de nuevo "enter".



```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE 1: node
Vue CLI v4.5.12
? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel, Router, Vuex, Unit
? Choose a version of Vue.js that you want to start the project with 2.x
? Use history mode for router? (Requires proper server setup for index fallback in production) Yes
? Pick a unit testing solution: Jest
? Where do you prefer placing config for Babel, ESLint, etc.? (Use arrow keys)
> In dedicated config files
  In package.json
```

Por último, cuando nos pregunte si queremos guardar esta configuración para futuros proyectos, le pondremos **"N"**, y le oprimimos "enter".



```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE 2: node
Vue CLI v4.5.12
? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel, Router, Vuex, Unit
? Choose a version of Vue.js that you want to start the project with 2.x
? Use history mode for router? (Requires proper server setup for index fallback in production) Yes
? Pick a unit testing solution: Jest
? Where do you prefer placing config for Babel, ESLint, etc.? In dedicated config files
? Save this as a preset for future projects? (y/N) n
```

Inmediatamente, **Vue CLI** comenzará a crear el proyecto. El tiempo que le tome dependerá de nuestra conexión a internet, y del equipo en el que estemos trabajando. Si todo sale bien, al finalizar, deberíamos ver dos instrucciones. La primera, que nos indica que debemos ingresar a la carpeta del proyecto recién creada usando el comando **"cd nombre_proyecto"**; y la segunda, que nos indicará el comando para levantar el proyecto en un servidor local **"npm run serve"**.

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
1: powershell

added 7 packages from 3 contributors in 13.481s

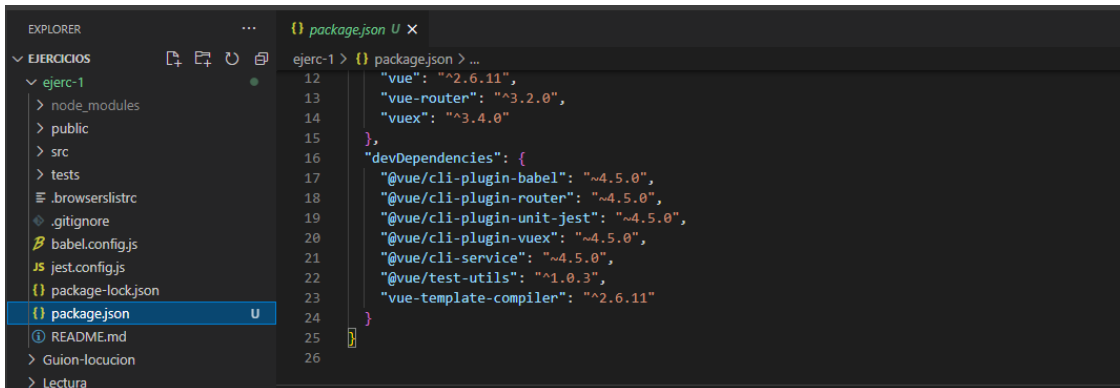
71 packages are looking for funding
  run `npm fund` for details

🔄 Running completion hooks...
📄 Generating README.md...
🟢 Successfully created project ejerc-1.
👉 Get started with the following commands:

$ cd ejerc-1
$ npm run serve

PS C:\Users\USUARIO\Desktop\Edutecno\W5-frontend-vue\W5-Front-end-Vue\W51\Ejercicios>
```

Al terminar la instalación, nos daremos cuenta de que **vue/cli** ha creado y configurado la estructura inicial. Aquí podemos encontrar el archivo **package.json**, en el cual están nuestras dependencias, incluidas **Vue-test-utils**, que es instalada por defecto por **CLI**, cuando seleccionamos la opción **Unit-testing**.



The screenshot shows the VS Code interface. On the left, the Explorer sidebar shows a project named 'EJERCICIOS' with a subfolder 'ejerc-1'. Inside 'ejerc-1', the 'package.json' file is selected and highlighted. On the right, the editor displays the content of 'package.json' for the 'ejerc-1' directory. The file lists dependencies for 'vue', 'vue-router', and 'vuex', and a 'devDependencies' section listing various Vue CLI plugins and 'vue-template-compiler'.

```
ejerc-1 > {} package.json > ...
12  "vue": "^2.6.11",
13  "vue-router": "^3.2.0",
14  "vuex": "^3.4.0"
15  },
16  "devDependencies": {
17    "@vue/cli-plugin-babel": "~4.5.0",
18    "@vue/cli-plugin-router": "~4.5.0",
19    "@vue/cli-plugin-unit-jest": "~4.5.0",
20    "@vue/cli-plugin-vuex": "~4.5.0",
21    "@vue/cli-service": "~4.5.0",
22    "@vue/test-utils": "^1.0.3",
23    "vue-template-compiler": "^2.6.11"
24  }
25
26
```

De inmediato nos dirigimos a nuestra carpeta del proyecto, escribiendo el comando: **"cd nombre_proyecto"**.

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
1: powershell

added 7 packages from 3 contributors in 13.481s

71 packages are looking for funding
  run `npm fund` for details

🔗 Running completion hooks...

📄 Generating README.md...

🎉 Successfully created project ejerc-1.
👉 Get started with the following commands:

$ cd ejerc-1
$ npm run serve

PS C:\Users\USUARIO\Desktop\Edutecno\W5-frontend-vue\W5-Front-end-Vue\W5S1\Ejercicios> cd ejerc-1
```

Ahora podemos escribir en la consola **“npm run serve”**, y tecleamos “enter”. Así, estaremos compilando el proyecto para levantarlo.

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
1: powershell

71 packages are looking for funding
  run `npm fund` for details

🔗 Running completion hooks...

📄 Generating README.md...

🎉 Successfully created project ejerc-1.
👉 Get started with the following commands:

$ cd ejerc-1
$ npm run serve

PS C:\Users\USUARIO\Desktop\Edutecno\W5-frontend-vue\W5-Front-end-Vue\W5S1\Ejercicios> cd ejerc-1
PS C:\Users\USUARIO\Desktop\Edutecno\W5-frontend-vue\W5-Front-end-Vue\W5S1\Ejercicios\ejerc-1> npm run serve
```

Una vez terminada la compilación, nos da el enlace para verlo en un servidor local.

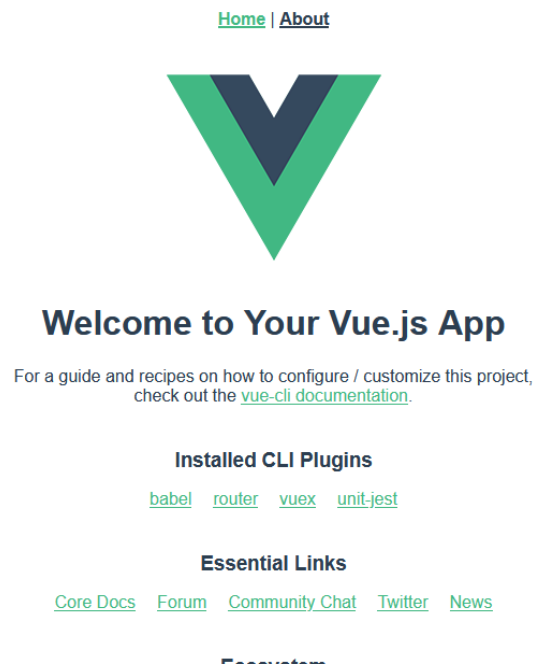
```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
1: node

🟢 DONE Compiled successfully in 2736ms 12:45:34

App running at:
- Local: http://localhost:8080/
- Network: http://192.168.181.23:8080/

Note that the development build is not optimized.
To create a production build, run npm run build.
```

Si abrimos ese enlace en un navegador, podremos ver algo similar a esto:



Con ello, ya hemos terminado de crear un proyecto usando las herramientas para testeo unitario.

EXERCISE 2: INSTALANDO JEST PARA CUALQUIER PROYECTO JS

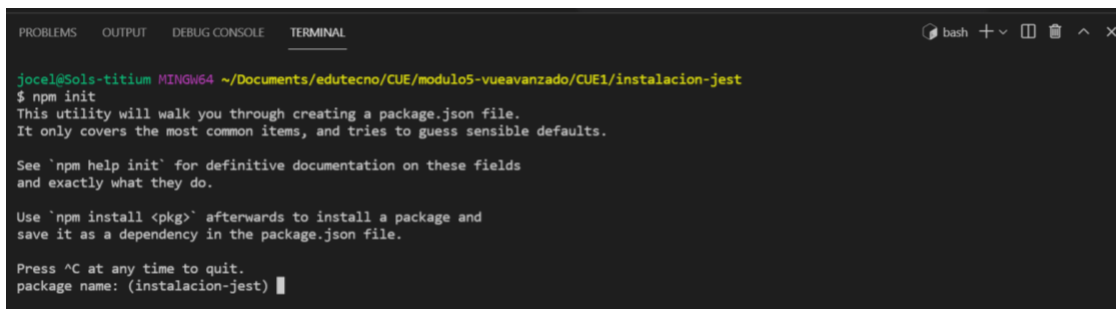
INTRODUCCIÓN

En el siguiente ejercicio, veremos cómo instalar **Jest** en proyectos **JS**, con los comandos y dependencias necesarias para su correcto funcionamiento.

CREANDO ARCHIVO PACKAGE.JSON

En primer lugar, abriremos el ejercicio al que le queremos hacer pruebas en VSC. Iniciamos la consola del editor de texto, y verificamos que estemos ubicados en la carpeta de nuestro proyecto. Lo siguiente que haremos, es crear nuestro archivo **"package.json"**. Para eso, en la consola de nuestro

editor, escribiremos el comando `"npm init"` y presionamos "enter". Obtendremos el siguiente resultado, donde se solicitará el valor `"name"` de nuestro proyecto. Por defecto, la terminal prevé que es el nombre de la carpeta en la que nos encontramos. Los valores predeterminados de cada propiedad se muestran entre paréntesis; dejaremos el predeterminado de `"name"`, y nuevamente "enter".



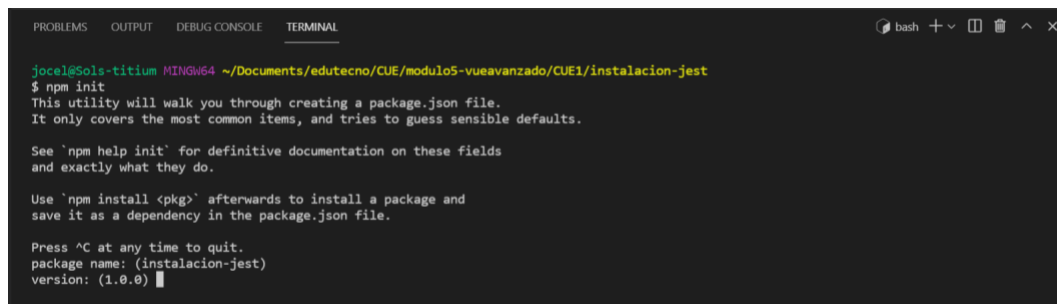
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
jocel@Sols-titium MINGW64 ~/Documents/edutecno/CUE/modulo5-vueavanzado/CUE1/instalacion-jest
$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See 'npm help init' for definitive documentation on these fields
and exactly what they do.

Use 'npm install <pkg>' afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (instalacion-jest) |
```

Ahora, el siguiente valor que se debe introducir es: `"versión"`. Al igual que con `"name"`, este campo se requiere si el proyecto se compartirá con otros en el repositorio de paquetes npm. Presionaremos "enter" para aceptar la versión predeterminada.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
jocel@Sols-titium MINGW64 ~/Documents/edutecno/CUE/modulo5-vueavanzado/CUE1/instalacion-jest
$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See 'npm help init' for definitive documentation on these fields
and exactly what they do.

Use 'npm install <pkg>' afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (instalacion-jest)
version: (1.0.0) |
```

El siguiente campo es `"description"`, una cadena útil para explicar lo que hace nuestro módulo de `Node.js`. Pondremos `"instalación Jest"` y oprimimos "enter".

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
```

```
jocel@Sols-titium MINGW64 ~/Documents/edutechno/CUE/modulo5-vueavanzado/CUE1/instalacion-jest
$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (instalacion-jest)
version: (1.0.0)
description: Instalacion Jest
```

En la siguiente solicitud, se pedirá el `"entry point"`. Si alguien instala y usa `"requires"` para su módulo, lo que configure en `"entry point"` será lo primero que se cargará de nuestro programa. El valor debe ser la ubicación relativa de un archivo de `JavaScript`, y se añadirá a la propiedad `"main"` de `"package.json"`. Teclearemos `"enter"` para conservar el valor predeterminado.

```

jocel@Sols-titium MINGW64 ~/Documents/edutechno/CUE/modulo5-vueavanzado/CUE1/instalacion-jest
$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See 'npm help init' for definitive documentation on these fields
and exactly what they do.

Use 'npm install <pkg>' afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (instalacion-jest)
version: (1.0.0)
description: Instalacion Jest
entry point: (index.js)

```

A continuación, se solicitará un **“test command”**, que es una secuencia de comandos ejecutable, o un comando para ejecutar las pruebas del proyecto. Lo dejaremos vacío, y presionaremos “enter”.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
jocel@Sols-titium MINGW64 ~/Documents/edutecno/CUE/modulo5-vueavanzado/CUE1/instalacion-jest
$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See 'npm help init' for definitive documentation on these fields
and exactly what they do.

Use 'npm install <pkg>' afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (instalacion-jest)
version: (1.0.0)
description: Instalacion Jest
entry point: (index.js)
test command:
```

Luego, la consola solicitará el repositorio **GitHub** del proyecto. En este ejemplo, no lo usaremos. Por lo tanto, lo dejamos vacío y oprimimos “enter”.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
jocel@Sols-titium MINGW64 ~/Documents/edutecno/CUE/modulo5-vueavanzado/CUE1/instalacion-jest
$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See 'npm help init' for definitive documentation on these fields
and exactly what they do.

Use 'npm install <pkg>' afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (instalacion-jest)
version: (1.0.0)
description: Instalacion Jest
entry point: (index.js)
test command:
git repository:
```

Después del repositorio, la terminal solicita **“keywords”**. Esta propiedad es una serie de cadenas, con términos útiles, que las personas pueden usar para encontrar nuestro repositorio. Como no lo vamos a utilizar, lo dejaremos vacío y nuevamente presionamos “enter”.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
jocel@Sols-titium MINGW64 ~/Documents/edutecno/CUE/modulo5-vueavanzado/CUE1/instalacion-jest
$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (instalacion-jest)
version: (1.0.0)
description: Instalacion Jest
entry point: (index.js)
test command:
git repository:
keywords:
```

El siguiente campo de la solicitud es **“author”**. Por ejemplo, si alguien descubre una vulnerabilidad de seguridad en nuestro módulo, puede usarlo para indicarnos el problema, a fin de que lo solucionemos. El campo **“author”**, es una cadena que tiene el formato: **"Name \<Email\> (sito web)"**. También podemos colocar nuestro nombre. Lo dejaremos vacío, y confirmamos oprimiendo **“enter”**.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
jocel@Sols-titium MINGW64 ~/Documents/edutecno/CUE/modulo5-vueavanzado/CUE1/instalacion-jest
$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

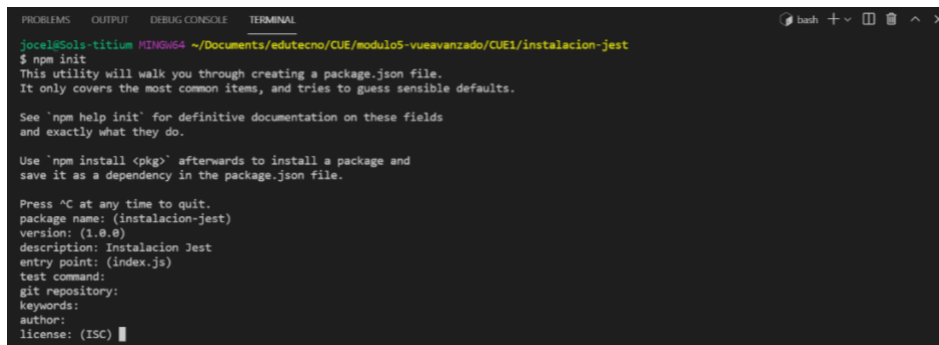
Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (instalacion-jest)
version: (1.0.0)
description: Instalacion Jest
entry point: (index.js)
test command:
git repository:
keywords:
author:
```

Por último, nos solicitará indicar la **“license”**. Con esto, se determinan los permisos legales, y las limitaciones que los usuarios tendrán al usar nuestro módulo. Muchos módulos de **Node.js** son de código abierto. Por lo tanto, **NPM** establece **ISC** como valor predeterminado. **ISC**, es una licencia de software libre permisiva, publicada por Internet Software Consortium, hoy en día llamado Internet Systems Consortium (**ISC**). Otorga permiso para usar, copiar, modificar y / o distribuir el software,

para cualquier propósito con o sin tarifa, siempre que el aviso de derechos de autor y el de permiso aparezcan en todas las copias.

Para este ejemplo, utilizaremos la licencia **ISC** predeterminada, y presionaremos “enter” para terminar el proceso.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
jocel@Sols-titium MINGW64 ~/Documents/edutecno/CUE/modulo5-vueavanzado/CUE1/instalacion-jest
$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See 'npm help init' for definitive documentation on these fields
and exactly what they do.

Use 'npm install <pkg>' afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (instalacion-jest)
version: (1.0.0)
description: Instalacion Jest
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
```

La terminal, ahora, mostrará el archivo **“package.json”** que creará, y tendrá el siguiente aspecto.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
keywords:
author:
license: (ISC)
About to write to C:\Users\jocel\Documents\edutecno\CUE\modulo5-vueavanzado\CUE1\instalacion-jest\package.json:

{
  "name": "instalacion-jest",
  "version": "1.0.0",
  "description": "Instalacion Jest",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}

Is this OK? (yes)
```

Una vez que la información coincida con lo que se ve aquí, presionamos “enter” para completar el proceso, y crear el archivo **“package.json”**. Con éste, se puede llevar un registro de los módulos que formen parte del proyecto.

INSTALANDO JEST

Lo siguiente que haremos será instalar **Jest**. Escribiremos en la consola el comando: **"npm install --save-dev jest"** y teclearemos "enter".

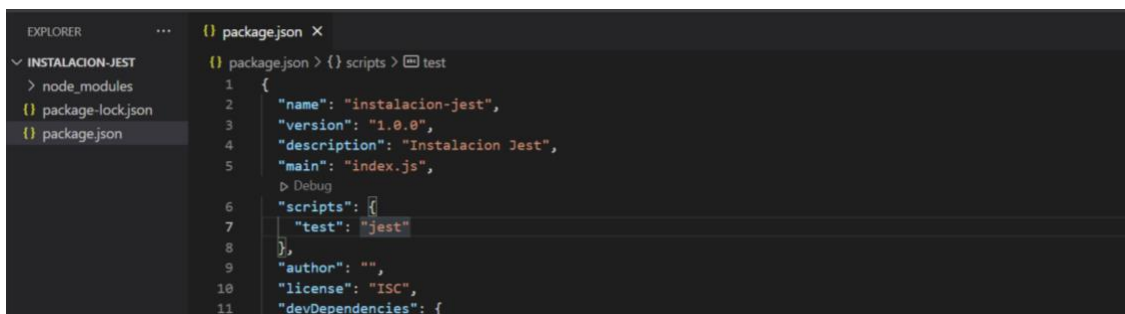


```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
keywords:
author:
license: (ISC)
About to write to C:\Users\jocel\Documents\edutecno\CUE\modulo5-vueavanzado\CUE1\instalacion-jest\package.json:

{
  "name": "instalacion-jest",
  "version": "1.0.0",
  "description": "Instalacion Jest",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}

Is this OK? (yes)
jocel@Sols-titium MINGW64 ~/Documents/edutecno/CUE/modulo5-vueavanzado/CUE1/instalacion-jest
$ npm install --save-dev jest
```

Por último, abriremos el archivo **"package.json"**, y en la sección **"scripts"**, agregaremos a **Jest** escribiendo **"test: jest"** y guardamos. De esta manera, podemos instalar **Jest** en cualquier proyecto **JavaScript** al que queramos hacerle pruebas unitarias.



```
EXPLORER package.json X
INSTALACION-JEST
> node_modules
package-lock.json
package.json
package.json
1 {
2   "name": "instalacion-jest",
3   "version": "1.0.0",
4   "description": "Instalacion Jest",
5   "main": "index.js",
6   "scripts": {
7     "test": "jest"
8   },
9   "author": "",
10  "license": "ISC",
11  "devDependencies": {
```