

Pretest

DEV.F
DESARROLLAMOS(PERSONAS);

Práctica 1

- Crearse una cuenta en [Quizizz](#).
- Realizar el quiz de Node, npm y APIs.



Npm

DEV.F
DESARROLLAMOS(PERSONAS);

dev

Node Package Manager es un manejador de paquetes de node (el más popular de JavaScript). Permite para compartir e instalar paquetes.

Se compone de 2 partes:

- **Un repositorio online para publicar paquetes** de software libre.
- **Una CLI para la terminal** que permite interactuar instalar, gestionar y publicar paquetes.

NOTA: Se puede considerar un gestor de dependencias de proyectos de tipo npm.

Comandos de Npm

Inicialización de un proyecto npm

- **npm init:** Inicializa una carpeta como un proyecto de npm.

Levantar un proyecto npm

- **npm start:** Es el único comando por defecto que no requiere la palabra run (npm run start). Permite “iniciar” un proyecto de node.

Comandos de Npm

Instalar/desinstalar dependencias

- `npm i`: Instala todas las dependencias del `package.json`
- `npm install -g <package-name>`: Instala un paquete globalmente.
- `npm install <package-name>`: Instala un paquete y lo agrega al `package.json`
- `npm install --save <package-name>`: Instala un paquete y lo agrega al `package.json`
- `npm install -D <package-name>`: Instala un paquete y lo agrega al `package.json` en la parte de `devDependencies`.

Comandos de Npm

Gestión de dependencias

- `npm uninstall <package-name>`
- `npm -g uninstall <package-name>`
- `npm search <package-name>`
- `npm ls`
- `npm update -save`
- `npm list`
- `npm list -g --depth 0`
- `npm outdated`

Paquetes

DEV.F
DESARROLLAMOS(PERSONAS);

dev

Paquetes

Son módulos distribuidos en forma de librerías que resuelven alguna necesidad de desarrollo. A continuación se listan los más populares al 2022:

- npm.
- create-react-app.
- vue-cli.
- grunt-cli.
- mocha.
- react-native-cli.
- gatsby-cli.
- forever.

Scripts

DEV.F
DESARROLLAMOS(PERSONAS);

Scripts

Son comandos propios que se pueden agregar al package.json para poderlos ejecutar con **npm run <my-comand>**.

```
"scripts": {  
  "test": "echo \"Error: no test specified\" && exit 1",  
  "start": "node index.js",  
  "dev": "nodemon index.js"  
},
```

Scaffold Npm

DEV.F
DESARROLLAMOS(PERSONAS);

dev

Estructura de proyecto npm

- **node_modules:** Carpeta donde se instalan las dependencias de un proyecto npm, normalmente está carpeta se agrega al .gitignore.
- **package.json:** Guardan las dependencias y los comandos de node.
- **package-lock.json:** Guarda un snapshot de las dependencias que se instalaron en un determinado momento.

Yarn

DEV.F
DESARROLLAMOS(PERSONAS);

dev

Yarn

Es un gestor de dependencias de javascript, con mejoras de velocidad y seguridad en comparación con el cliente npm.

- Creado en 2016 por ingenieros de facebook y google.
- Es más amigable en su uso (sintaxis y colores).
- Utiliza npm y depende de node.
- Npm sigue siendo el repositorio central de paquetes, solo se cambia el cliente (gestor de paquetes/dependencias).



Ventajas de Yarn

- **Velocidad:** La mayoría de la instalación de paquetes por npm toma minutos, en yarn toma segundos.
- **Seguridad:** Yarn verifica la integridad de cada paquete y comprueba que sea seguro.
- **Fiabilidad:** La red de npm suele tener issues, yarn tiene un CDN de mayor fiabilidad.
- **Múltiple registro:** Tiene acceso al registry de bower y npm, mientras que npm solo al de npm.

Comandos Yarn

Comando	Descripción
<code>npm install -g yarn</code>	Instalar globalmente yarn.
<code>yarn --version</code>	Validar versión de yarn.
<code>yarn init</code>	Inicializar un proyecto yarn.
<code>yarn add <package-name></code>	Agregar una dependencia.
<code>yarn install</code>	Instalar una dependencia.
<code>yarn publish</code>	Publicar un paquete.

Semantic Versioning

SEMVER

DEV.F
DESARROLLAMOS(PERSONAS);

dev

Semantic Versión

Es un conjunto simple de reglas y requerimientos que dictan cómo asignar e incrementar los números de la versión de un software. Evitan la pérdida de versiones y mejoran la gestión de dependencias.

1.2.3-beta.1+meta

The diagram illustrates the components of the Semantic Versioning string '1.2.3-beta.1+meta'. It features a horizontal line with five vertical tick marks. Below each tick mark is a label: 'Major' under the first tick, 'Minor' under the second, 'Patch' under the third, 'Pre-release' under the fourth, and 'Metadata' under the fifth. Lines connect each label to its corresponding tick mark on the horizontal line.

Major Minor Patch Pre-release Metadata

Funcionamiento de semantic versión

Dado un número de versión **MAYOR.MENOR.PARCHE**, se incrementa:

- La versión **MAYOR** cuando realizas un cambio incompatible en el proyecto.
- La versión **MENOR** cuando añades funcionalidad que compatible con versiones anteriores.
- La versión **PARCHE** cuando reparas errores compatibles con versiones anteriores.

MAJOR.MINOR.PATCH = MAYOR.MENOR.PARCHE

Ejemplo: 1.2.1