## Pretest

DEV.F.:
DESARROLLAMOS(PERSONAS);

## **Práctica 1**

- Crearse una cuenta en Quizizz.
- Realizar el quiz de Node, npm y APIs.





# Npm

DEVIEW DESARROLLAMOS (PERSONAS);

#### Npm

**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 uninstallnpm search <package-name>
- npm ls
- npm update -save
- npm list
- npm list -g –depth 0
- npm outdated

g <package-name>



# **Paquetes**

DEV.F.:
DESARROLLAMOS(PERSONAS);

## **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



## 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

DEVIEW DESARROLLAMOS (PERSONAS);

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

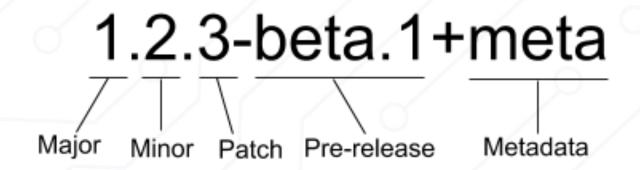


# Semantic Versioning SEMVER



### **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.





#### 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

