# Instalación de librerías y paquetes con NPM

Julian F. Latorre

1 de noviembre de 2024

# Contenido

# ¿Qué es NPM?

- NPM = Node Package Manager
- Gestor de paquetes estándar para Node.js
- Facilita compartir e integrar código
- Gestiona dependencias de proyectos

# Importancia de NPM

- Reutilización de código
- Gestión simplificada de dependencias
- Versionado y actualización fácil de paquetes
- Amplia comunidad y repositorio de paquetes

# Instalación en Windows

1. Visitar `https://nodejs.org`
2. Descargar versión LTS
3. Ejecutar instalador
4. Verificar instalación:

```
node --version
npm --version
```

# Instalación en macOS

1. Instalar Homebrew: `https://brew.sh`
2. Ejecutar en terminal:

```
brew install node
```

3. Verificar instalación como en Windows

# Instalación en Linux (Ubuntu/Debian)

1. Actualizar repositorios:

```
sudo apt update
```

2. Instalar Node.js y NPM:

```
sudo apt install nodejs npm
```

3. Verificar instalación

# El archivo package.json

- Corazón de cualquier proyecto Node.js
- Contiene metadatos del proyecto
- Lista dependencias
- Creación:

```
npm init
```

# Estructura básica de package.json

```
{
  "name": "mi-proyecto",
  "version": "1.0.0",
  "description": "Descripción breve",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit
        1"
  },
  "author": "Tu Nombre",
  "license": "ISC",
  "dependencies": {},
  "devDependencies": {}
}
```

# Semántica de versionado

- NPM usa versionado semántico (SemVer)
- Formato: MAJOR.MINOR.PATCH
- MAJOR: cambios incompatibles
- MINOR: nuevas funcionalidades compatibles
- PATCH: correcciones de errores compatibles

# Instalación local vs. global

- Local: añade al directorio `node_modules` del proyecto

```
npm install nombre-del-paquete
```

- Global: disponible en todo el sistema

```
npm install -g nombre-del-paquete
```

# Flags comunes en la instalación

- `--save` o `-S`: Guarda como dependencia
- `--save-dev` o `-D`: Guarda como dependencia de desarrollo
- `--no-save`: Instala sin añadir a `package.json`
- `--exact` o `-E`: Instala versión exacta

# Dependencias vs. devDependencies

- **dependencies**: Necesarias en producción
- **devDependencies**: Necesarias solo en desarrollo

# Actualización y eliminación de paquetes

Actualizar un paquete:

```
npm update nombre-del-paquete
```

Actualizar todos los paquetes:

```
npm update
```

Eliminar un paquete:

```
npm uninstall nombre-del-paquete
```

# Definición y ejecución de scripts

En `package.json`:

```
{
  "scripts": {
    "start": "node index.js",
    "test": "jest",
    "build": "webpack"
  }
}
```

Ejecución:

```
npm run nombre-del-script
```

# Uso de npm audit

Revisar vulnerabilidades:

```
npm audit
```

Corregir automáticamente:

```
npm audit fix
```

# Buenas prácticas

- Mantener `package-lock.json` versionado
- Usar `.npmrc` para configuraciones personalizadas
- Revisar regularmente las dependencias
- Mantener Node.js y NPM actualizados

# Conclusión

- NPM es esencial en el desarrollo web moderno
- Facilita la gestión de dependencias
- Permite compartir código eficientemente
- Mantiene las aplicaciones actualizadas y seguras
- Documentación oficial: `https://docs.npmjs.com/`