# **Trabajo Práctico: Objetivos**

## **Parte 1: npm y package.json**

### **1. Introducción a npm**

**npm (Node Package Manager)** es el gestor de paquetes de Node.js, utilizado para instalar, compartir y administrar librerías y herramientas en proyectos de JavaScript.

### **2. Creación de un proyecto con npm**

#### **Pasos:**

1. Abre una terminal y crea una nueva carpeta para el proyecto:  
   mkdir mi-proyecto && cd mi-proyecto
2. Inicializa un proyecto de Node.js con:  
   npm init o npm init -y

### **3. Explorando package.json**

El archivo package.json contiene metadatos del proyecto y dependencias. Ejemplo de un archivo package.json:

{

"name": "mi-proyecto",

"version": "1.0.0",

"description": "Un proyecto de prueba",

"main": "index.js",

"scripts": {

"start": "node index.js"

},

"dependencies": {

"yargs": "^17.7.2"

}

}

### **4. Instalación de paquetes**

Para instalar un paquete, usa:

npm install nombre-del-paquete

Por ejemplo, para instalar yargs:

npm install yargs

Para instalar dependencias de desarrollo:

npm install eslint --save-dev

## **Parte 2: Uso de Yargs**

### **1. Introducción a Yargs**

**Yargs** es una librería que facilita el manejo de argumentos en la línea de comandos en Node.js.

### **2. Creación de un script con Yargs**

Crea un archivo index.js y escribe el siguiente código:

const yargs = require('yargs');

const argv = yargs

.command('saludar', 'Muestra un saludo', {

nombre: {

describe: 'Nombre de la persona a saludar',

demandOption: true,

type: 'string'

}

})

.help()

.argv;

if (argv.\_.includes('saludar')) {

console.log(`Hola, ${argv.nombre}!`);

}

### **3. Ejecutando el script**

Ejecuta el siguiente comando en la terminal:

node index.js saludar --nombre="Juan"

Salida esperada:

Hola, Juan!

## **Ejercicios adicionales**

### **Ejercicios teóricos:**

1. ¿Qué es npm y su función en un proyecto de Node.js?

npm (Node Package Manager) es una herramienta que permite gestionar las dependencias (librerías o paquetes) en proyectos de Node.js. Su función principal es permitir la instalación, actualización, y gestión de estas dependencias, así como la ejecución de scripts definidos en el proyecto.

1. Estructura básica del archivo package.json y tres campos importantes:

{

"name": "nombre-del-proyecto",

"version": "1.0.0",

"description": "Descripción del proyecto",

"main": "index.js",

"scripts": {

"start": "node index.js",

"test": "echo \"Error: no test specified\" && exit 1"

},

"keywords": ["palabra", "clave"],

"author": "Tu nombre",

"license": "ISC",

"dependencies": {

"express": "^4.17.1"

},

"devDependencies": {

"nodemon": "^2.0.7"

}

}

El archivo package.json es el archivo de configuración de un proyecto Node.js. Contiene información sobre el proyecto y las dependencias necesarias.

Tres campos importantes son:

name: El nombre del proyecto.

version: La versión del proyecto.

dependencies: Lista de paquetes o librerías requeridas para el funcionamiento del proyecto.

1. Diferencia entre dependencias normales y dependencias de desarrollo en npm:

Dependencias normales: Son los paquetes que el proyecto necesita para funcionar en producción. Se instalan con el comando npm install <paquete>.

Dependencias de desarrollo: Son los paquetes necesarios solo para el desarrollo (como herramientas de testing o de compilación). Se instalan con el comando npm install <paquete> --save-dev.

1. Comando para actualizar un paquete instalado:

Para actualizar un paquete a su última versión compatible con el archivo package.json, utilizamos el comando:

npm update <paquete>

1. ¿Para qué se utiliza el comando npm link?

El comando npm link se utiliza para vincular un paquete local a tu proyecto. Esto es útil cuando estás desarrollando un paquete y quieres probarlo en otro proyecto sin tener que publicarlo primero. Crea un enlace simbólico entre el paquete y el proyecto.

1. Ver el módulo de inicio de un proyecto:

Para ver el módulo de inicio del proyecto, que es el archivo que Node.js ejecuta al iniciar la aplicación, puedes consultar el campo main en el archivo package.json. Este campo indica el archivo principal de la aplicación. Por defecto, puede ser index.js. Para verlo, ejecuta:

cat package.json

node index.js

### **Ejercicios prácticos:**

1. Agrega otro comando a Yargs llamado despedir que muestre "Adiós, [nombre]!".
2. Crea un script con Yargs que permita sumar dos números desde la línea de comandos. (n1 y n2)
3. Implementa un comando que lea un archivo JSON y muestre su contenido en la terminal.
4. Modifica el script para que, si el usuario no proporciona el argumento nombre, se muestre un mensaje de error adecuado.