

Informe de Examen: Creación y Consumo de Librerías NPM para una Calculadora

Jose Daniel Arias Toasa

1 de agosto de 2025

1. Objetivos

Objetivo General

Modularizar las operaciones de una calculadora básica (suma, resta, multiplicación y división) mediante la creación de paquetes NPM independientes y su posterior consumo en un entorno de desarrollo local.

Objetivos Específicos

- Desarrollar y encapsular cada una de las cuatro operaciones aritméticas en su propio paquete NPM, definiendo su funcionalidad y metadatos en los archivos `index.js` y `package.json`.
- Publicar los cuatro paquetes de manera local utilizando el comando `npm link` para hacerlos disponibles en la máquina de desarrollo sin necesidad de un registro público.
- Crear un proyecto consumidor que instale como dependencias los cuatro paquetes locales enlazados.
- Implementar un script en el proyecto consumidor para verificar el correcto funcionamiento de las librerías, ejecutando las operaciones con los valores especificados en el requerimiento del examen.

2. Introducción

Este documento detalla el proceso llevado a cabo para cumplir con los requisitos del examen del segundo bimestre. La tarea consistió en desarrollar cuatro librerías de Node.js, cada una responsable de una operación matemática fundamental: suma, resta, multiplicación y división.

El informe describe el procedimiento completo, desde la creación individual de cada paquete NPM, su publicación en el entorno local mediante `npm link`, hasta la integración y uso de estos módulos en un proyecto principal que consume dichas funcionalidades para realizar un conjunto de cálculos predefinidos. Este enfoque demuestra la capacidad

de modularizar el código y gestionar dependencias locales, un pilar fundamental en el desarrollo de software moderno con Node.js.

3. Paso 1: Desarrollo de los Módulos de Operaciones

Explicación: El primer paso consistió en crear cuatro directorios separados, uno para cada operación. Dentro de cada directorio, se inicializó un proyecto de Node.js con `npm init` para generar el archivo `package.json`. A continuación, se implementó la lógica de cada operación en su respectivo archivo `index.js`.

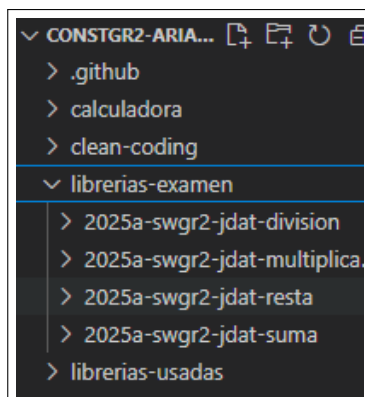


Figura 1: Estructura de directorios para las librerías.

A continuación, se muestra el contenido del archivo `package.json` del módulo de suma como ejemplo representativo.

```
1 {  
2   "name": "2025a-swgr2-jdat-suma",  
3   "version": "1.0.0",  
4   "description": "",  
5   "license": "ISC",  
6   "author": "Jose Daniel Arias Toasa",  
7   "type": "commonjs",  
8   "main": "index.js",  
9   "scripts": {  
10    "test": "echo \"Error: no test specified\" && exit 1"  
11  }  
12 }
```

Listing 1: Contenido de `package.json` para el módulo de suma.

La lógica implementada para cada operación es la siguiente:

```
1 exports.suma = (numeroUno, numeroDos) => {  
2   const numeroUnoCasteado = Number(numeroUno);  
3   const numeroDosCasteado = Number(numeroDos);  
4   if (isNaN(numeroUno) || isNaN(numeroDos)) {  
5     throw new Error('No son numeros validos')  
6   }  
7   return numeroUnoCasteado + numeroDosCasteado;  
8 }
```

Listing 2: Código de la librería de Suma (`2025a-swgr2-jdat-suma/index.js`).

```
1 exports.resta = (numeroUno, numeroDos) => {
2   const numeroUnoCasteado = Number(numeroUno);
3   const numeroDosCasteado = Number(numeroDos);
4   if (isNaN(numeroUno) || isNaN(numeroDos)) {
5     throw new Error('No son numeros validos')
6   }
7   return numeroUnoCasteado - numeroDosCasteado;
8 }
```

Listing 3: Código de la librería de Resta (2025a-swgr2-jdat-resta/index.js).

```
1 exports.multiplicacion = (numeroUno, numeroDos) => {
2   const numeroUnoCasteado = Number(numeroUno);
3   const numeroDosCasteado = Number(numeroDos);
4   if (isNaN(numeroUno) || isNaN(numeroDos)) {
5     throw new Error('No son numeros validos')
6   }
7   return numeroUnoCasteado * numeroDosCasteado;
8 }
```

Listing 4: Código de la librería de Multiplicación (2025a-swgr2-jdat-multiplicacion/index.js).

```
1 exports.division = (numeroUno, numeroDos) => {
2   const numeroUnoCasteado = Number(numeroUno);
3   const numeroDosCasteado = Number(numeroDos);
4
5   if (isNaN(numeroUno) || isNaN(numeroDos)) {
6     throw new Error('No son numeros validos')
7   }
8
9   if (numeroDosCasteado === 0) {
10    return 'No se puede dividir por cero';
11  }
12
13  return numeroUnoCasteado / numeroDosCasteado;
14 }
```

Listing 5: Código de la librería de División (2025a-swgr2-jdat-division/index.js).

4. Paso 2: Publicación Local de los Módulos

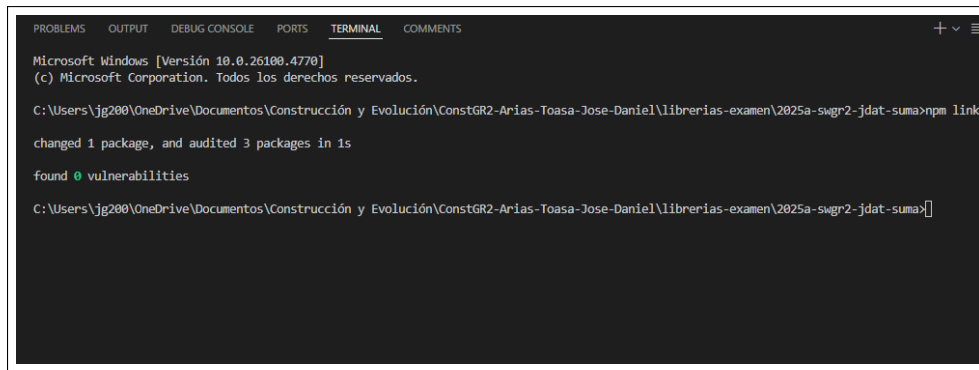
Explicación: Para que los módulos estuvieran disponibles para ser utilizados en otro proyecto sin publicarlos en el registro público de NPM, se utilizó `npm link`. Este comando se ejecutó dentro de cada uno de los cuatro directorios de las librerías.

```
1 npm link
```

Listing 6: Comando para enlazar un paquete localmente.

5. Paso 3: Instalación de Dependencias Locales

Explicación: Se creó un nuevo proyecto llamado `librerias-usadas` y se inicializó con `npm init`. Posteriormente, se enlazaron los cuatro paquetes locales como dependencias utilizando el comando `npm link <nombre-del-paquete>`.



```
Microsoft Windows [Versión 10.0.26100.4770]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\jg200\OneDrive\Documentos\Construcción y Evolución\ConstGR2-Arias-Toasa-Jose-Daniel\librerias-examen\2025a-swgr2-jdat-suma>npm link

changed 1 package, and audited 3 packages in 1s

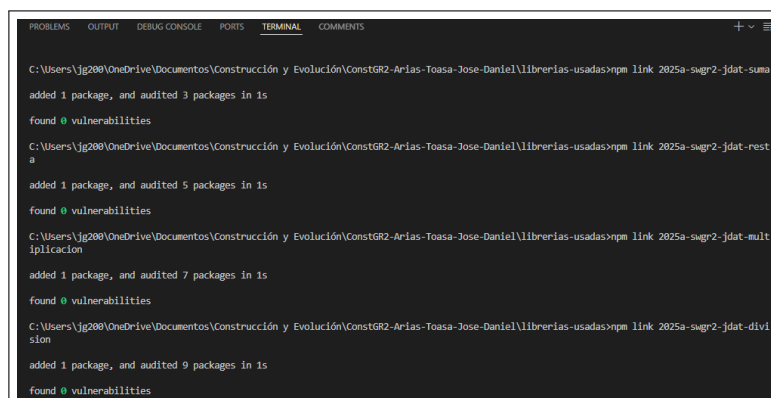
found 0 vulnerabilities

C:\Users\jg200\OneDrive\Documentos\Construcción y Evolución\ConstGR2-Arias-Toasa-Jose-Daniel\librerias-examen\2025a-swgr2-jdat-suma>[]
```

Figura 2: Ejecución del comando ‘npm link’ para uno de los módulos.

```
1 npm link 2025a-swgr2-jdat-suma
2 npm link 2025a-swgr2-jdat-resta
3 npm link 2025a-swgr2-jdat-multiplicacion
4 npm link 2025a-swgr2-jdat-division
```

Listing 7: Comandos para instalar los paquetes locales.



```
C:\Users\jg200\OneDrive\Documentos\Construcción y Evolución\ConstGR2-Arias-Toasa-Jose-Daniel\librerias-usadas>npm link 2025a-swgr2-jdat-suma

added 1 package, and audited 3 packages in 1s

found 0 vulnerabilities

C:\Users\jg200\OneDrive\Documentos\Construcción y Evolución\ConstGR2-Arias-Toasa-Jose-Daniel\librerias-usadas>npm link 2025a-swgr2-jdat-resta

added 1 package, and audited 5 packages in 1s

found 0 vulnerabilities

C:\Users\jg200\OneDrive\Documentos\Construcción y Evolución\ConstGR2-Arias-Toasa-Jose-Daniel\librerias-usadas>npm link 2025a-swgr2-jdat-multiplicacion

added 1 package, and audited 7 packages in 1s

found 0 vulnerabilities

C:\Users\jg200\OneDrive\Documentos\Construcción y Evolución\ConstGR2-Arias-Toasa-Jose-Daniel\librerias-usadas>npm link 2025a-swgr2-jdat-division

added 1 package, and audited 9 packages in 1s

found 0 vulnerabilities
```

Figura 3: Vinculación de las cuatro librerías en el proyecto consumidor.

6. Paso 4: Consumo y Verificación de los Módulos

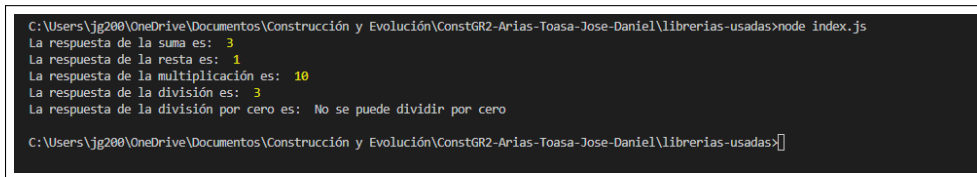
Explicación: Finalmente, se creó un archivo `index.js` en el proyecto `librerias-usadas`. Este script importa cada uno de los módulos y ejecuta las operaciones con los valores solicitados en la tarea para verificar su correcto funcionamiento.

```
1 const paqueteSuma = require('2025a-swgr2-jdat-suma');
2 const paqueteResta = require('2025a-swgr2-jdat-resta');
3 const paqueteMultiplicacion = require('2025a-swgr2-jdat-multiplicacion');
4 const paqueteDivision = require('2025a-swgr2-jdat-division');
5
6 const respuestaSuma = paqueteSuma.suma(2, 1);
7 const respuestaResta = paqueteResta.rest(3, 2);
8 const respuestaMultiplicacion = paqueteMultiplicacion.multiplicacion(2, 5);
9 const respuestaDivision = paqueteDivision.division(9, 3);
10 const respuestaDivisionPorCero = paqueteDivision.division(8, 0);
```

```
11
12 console.log('La respuesta de la suma es: ', respuestaSuma);
13 console.log('La respuesta de la resta es: ', respuestaResta);
14 console.log('La respuesta de la multiplicacion es: ',
    respuestaMultiplicacion);
15 console.log('La respuesta de la division es: ', respuestaDivision);
16 console.log('La respuesta de la division por cero es: ',
    respuestaDivisionPorCero);
```

Listing 8: Código de consumo de las librerías (librerias-usadas/index.js).

La ejecución del script mediante `node index.js` arroja la siguiente salida, confirmando que todos los módulos funcionan como se esperaba.



```
C:\Users\jg200\OneDrive\Documentos\Construcción y Evolución\ConstGR2-Arias-Toasa-Jose-Daniel\librerias-usadas>node index.js
La respuesta de la suma es: 3
La respuesta de la resta es: 1
La respuesta de la multiplicación es: 10
La respuesta de la división es: 3
La respuesta de la división por cero es: No se puede dividir por cero
C:\Users\jg200\OneDrive\Documentos\Construcción y Evolución\ConstGR2-Arias-Toasa-Jose-Daniel\librerias-usadas>
```

Figura 4: Salida en la consola mostrando los resultados correctos de las operaciones.

7. Conclusión

Se cumplieron satisfactoriamente todos los objetivos planteados para el examen. Se logró modularizar una calculadora básica mediante la creación de cuatro paquetes NPM independientes, cada uno responsable de una operación aritmética. Se demostró el uso correcto de `npm link` para la publicación y consumo de paquetes en un entorno de desarrollo local, validando finalmente la funcionalidad integrada con los cálculos y resultados esperados. Este ejercicio ha permitido consolidar los conocimientos sobre la gestión de paquetes y la modularización de código en el ecosistema de Node.js.

Referencias

- [1] NPM. (2025). *NPM CLI Documentation*. Obtenido de <https://docs.npmjs.com/>.
- [2] OpenJS Foundation. (2025). *Node.js Documentation*. Obtenido de <https://nodejs.org/docs/>.