Node, React y Angular

Construcción de software integrador de tecnologías orientadas a servicios

F.T-3223875

Daniel Steven Holguin Soto

Análisis y desarrollo de software

Instructor: Luis Fernando

30/10/2025

Tabla de contenido

Actividad 1 – No se presenta evidencias

POO (Programación orientada a objetos)

En el mundo de la programación, la POO es un paradigma que ha ganado una gran popularidad en los últimos años debido a su capacidad para crear aplicaciones más robustas, flexibles y fáciles de mantener. Esta metodología de desarrollo se basa en la idea de que los programas se pueden organizar como una colección de objetos interconectados, cada uno con su propio conjunto de datos y funcionalidades.

Actividad 2 - Preguntas

¿Qué es un gestor dependencias de código?

Es una herramienta que se encarga de instalar, actualizar y administrar librerías o paquetes que necesita un proyecto para funcionar. Facilita mantener el código ordenado y compatible.

¿Qué es npm?

npm (Node Package Manager) es el gestor de dependencias oficial de Node.js. Permite instalar, compartir y manejar paquetes de JavaScript fácilmente.

¿Para qué se utiliza principalmente npm?

Se usa para instalar librerías y herramientas necesarias en un proyecto, además de gestionar versiones y dependencias de manera automática.

¿Qué es el versionado semántico?

Es una forma estándar de numerar las versiones de un software para indicar qué tipo de cambios se han hecho (mejoras, correcciones o cambios grandes).

Como está especificado el Versionado Semántico.

Se representa con tres números:

MAJOR.MINOR.PATCH

MAJOR: cambios grandes o incompatibles.

MINOR: nuevas funciones sin romper lo anterior.

PATCH: correcciones pequeñas o errores.

¿Qué son las dependencias locales?

Son las librerías instaladas dentro del proyecto (en la carpeta node_modules) y que solo funcionan en ese proyecto.

¿Qué son las dependencias de desarrollo?

Son paquetes que solo se usan durante el desarrollo, no en la ejecución del programa.

¿Qué son las dependencias globales?

Son paquetes instalados en todo el sistema, no solo en un proyecto.

Se usan cuando se necesita ejecutar una herramienta desde cualquier carpeta.

¿Qué es el archivo package.json y que apartados tiene y cuál es su utilidad?

Es un archivo principal del proyecto npm que contiene información como:

name (nombre del proyecto)

version

scripts (comandos para ejecutar tareas)

dependencies y devDependencies

Su utilidad es describir el proyecto y sus dependencias para que otros puedan instalarlo fácilmente.

¿Qué es el archivo package-lock.json y que utilidad tiene?

Guarda las versiones exactas de todas las dependencias instaladas.

Sirve para que todos los desarrolladores usen las mismas versiones, evitando errores por cambios.

¿Qué es la carpeta node_modules en un proyecto de npm?

Es la carpeta donde npm guarda todos los paquetes instalados.

Contiene el código de las librerías que el proyecto necesita para funcionar.

Command Line Cheat Sheet

Como inicializar un proyecto de npm.	**Cómo instalar dependencias locales**	**Cómo instalar dependencias de desarrollo**
Se abre la terminal: Ctrl + ñ	Para instalar las dependencias locales se utiliza el siguiente comando: npm install paquete	Para instalar las dependencias de desarrollo se utiliza el siguiente comando: npm install paquete –save-dev
Creamos y entramos a una carpeta: mkdir mi-proyecto cd mi-proyecto	**Cómo instalar dependencias globales**	**Cómo visualizar las dependencias instaladas**

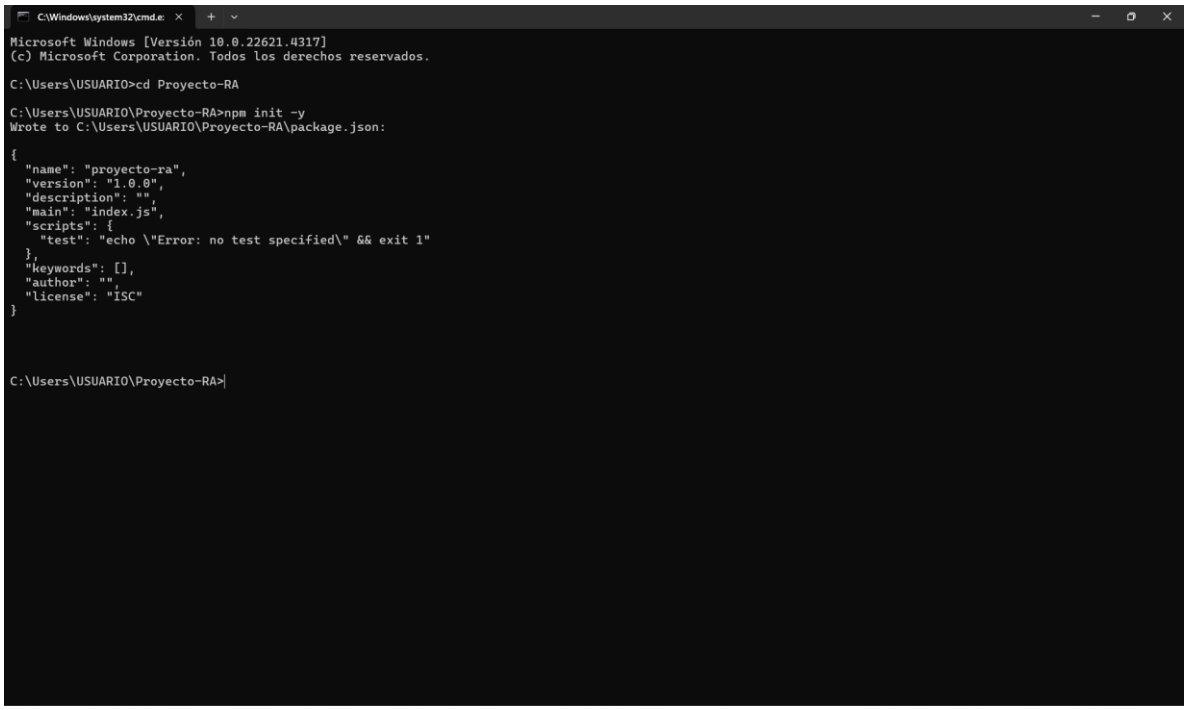
Inicializa el proyecto con npm: npm init	Para instalar las dependencias locales se utiliza el siguiente comando: Npm install -g paquete	Para ver dependencias locales: npm list Para ver dependencias globales: npm list -g –depth=0
Cómo instalar una versión específica de un paquete	**Cómo crear un comando en tu proyecto (scripts en el package.json)**	**Cómo actualizar dependencias**

El formato básico es:
npm install **express@4.17.1**
Instalar una versión de desarrollo especifica:
npm install **nodemon@3.0.1** –save-dev
	Primero se ubica la seccion “scripts” en el package.json
	Luego dentro de “scripts” Por ejemplo:
	
	"scripts": {
	"start": "node index.js",
	"dev": "nodemon index.js",
	"test": "echo \"No hay pruebas definidas\""
	}
		Ver dependencias desactualizadas:
		npm outdated
		Actualizar una dependencia especifica:
		npm install express@latest
		Actualizar todas las dependencias:
		npm update

Cómo eliminar paquetes	**Cómo actualizar node_modules**	
Eliminar una dependencia local: npm uninstall paquete Eliminar una dependencia de desarrollo: npm uninstall paquete –save-dev Eliminar una dependencia local: Npm uninstall -g paquete Verificar que se eliminó correctamente: Npm list –depth=0	Para actualizar node_modules, la forma más segura y efectiva es eliminar la carpeta node_modules y el archivo package-lock.json, y luego ejecutar npm install en la consola para reinstalar todos los paquetes especificados en package.json	

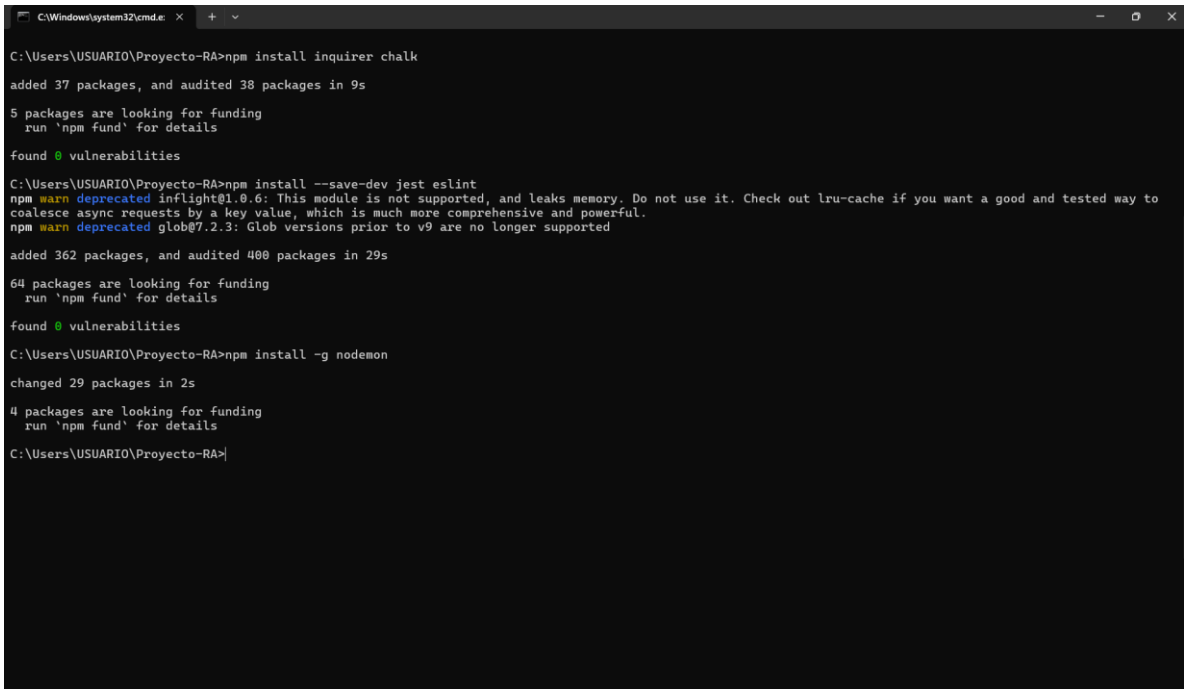
Actividad 3 – Manejo Práctico del gestor de dependencias NPM - Bitácora

Primer paso: Inicialización de un Proyecto npm



Primero se crea una carpeta y se entra en ella. Luego se inicializa el proyecto con npm init -y, lo que genera el archivo package.json, necesario para administrar las dependencias del proyecto.

Segundo paso: Instalación de dependencias

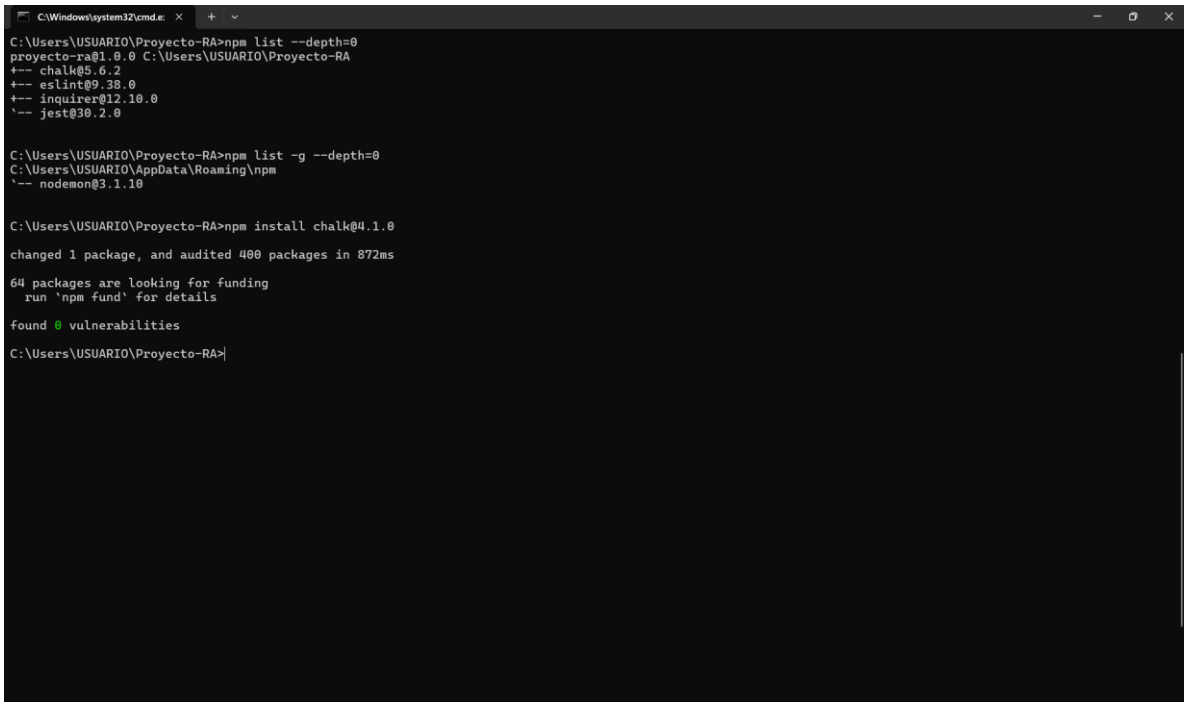


Las dependencias locales se instalan con npm install inquirer chalk y son necesarias para que la aplicación funcione.

Las dependencias de desarrollo se instalan con npm install --save-dev jest eslint y solo se usan durante el desarrollo.

Las dependencias globales se instalan con npm install -g nodemon y quedan disponibles en todo el sistema.

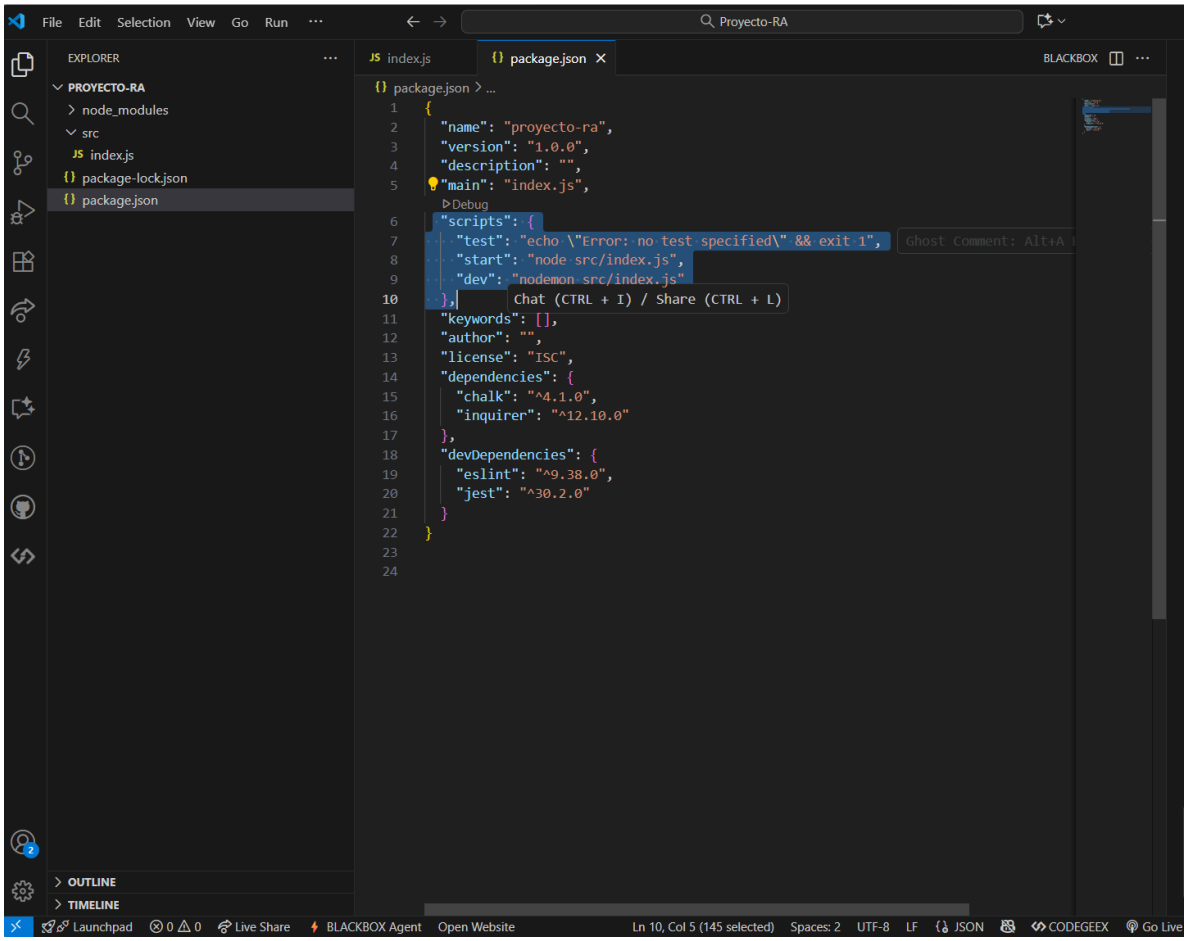
Tercer paso: Gestión de paquetes

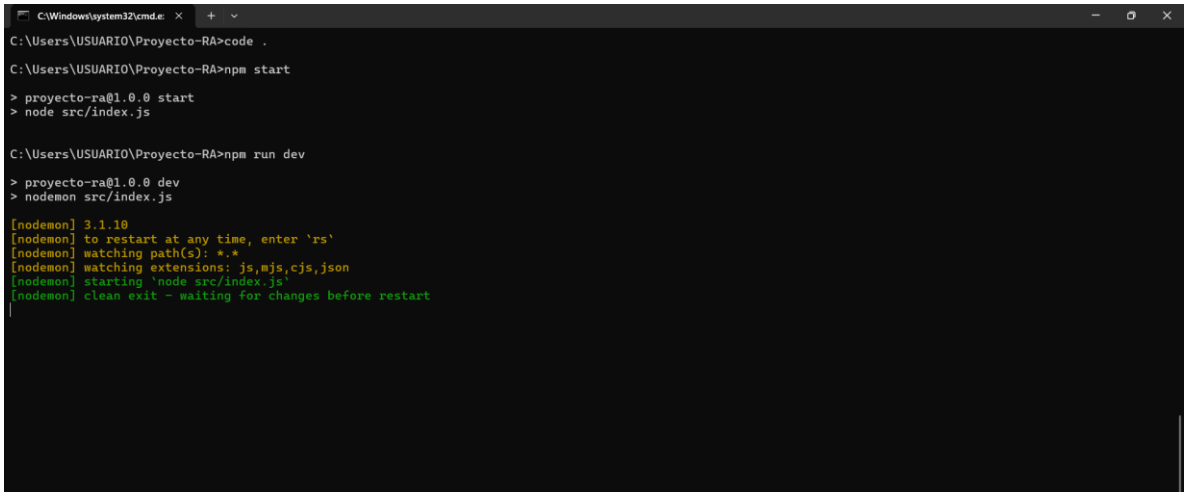


Con npm list --depth=0 se ven las dependencias locales y con npm list -g --depth=0 las globales.

Para instalar una versión específica de un paquete, se usa npm install nombre@versión, por ejemplo, npm install chalk@4.1.0, lo que ayuda a evitar problemas de compatibilidad.

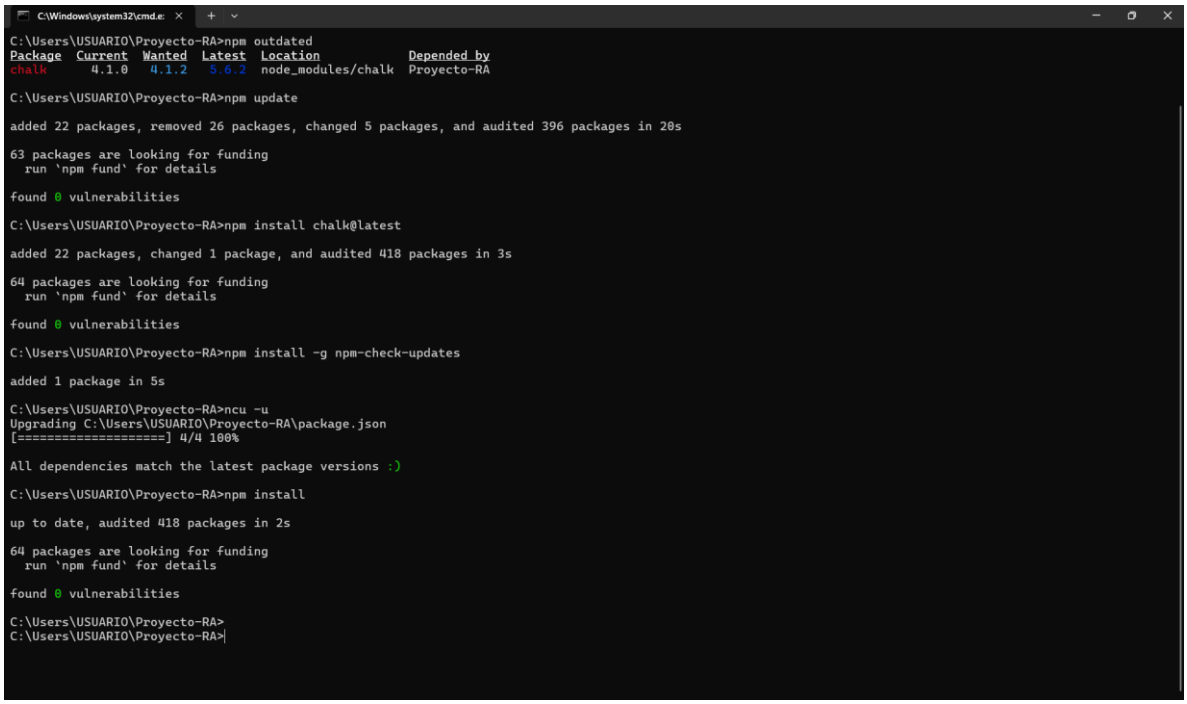
Cuarto paso: Scripts y actualizaciones





Cree la carpeta directamente en visual studio code y cree el archivo index.js, luego me metí al apartado de package.json y puse los scripts que están en azul, y por ultimo lo puse a ejecutar.

Actualizar dependencias



Actualizar dependencias en un proyecto Node.js es importante para mantener las librerías al día, corregir errores y aprovechar nuevas funciones.

npm outdated te permite ver qué paquetes están desactualizados.

npm update actualiza todas las dependencias a las versiones más recientes compatibles según el package.json.

Si solo quieres actualizar una dependencia específica, usas

npm update nombre-paquete

Para forzar la instalación de la última versión absoluta, usas

npm install nombre-paquete@latest

Si quieres que todas las dependencias se actualicen sin importar el rango de versiones, puedes usar la herramienta externa npm-check-updates (ncu).

Actualizar Dependencias Globales

Las dependencias globales son aquellas herramientas instaladas con el flag -g, como nodemon, npm-check-updates, o cualquier herramienta CLI que desees usar desde cualquier lugar del sistema.

a. Ver qué paquetes globales están desactualizados

npm outdated -g --depth=0

Muestra una lista con los paquetes globales que tienen versiones más recientes disponibles.

b. Actualizar todos los paquetes globales

npm update -g

Actualiza todos los paquetes globales a sus últimas versiones compatibles según los rangos definidos.

c. Actualizar un paquete global específico a la última versión

npm install -g nombre-paquete

Actualiza un solo paquete global a la versión más reciente compatible.

Para forzar la actualización a la última versión absoluta, usa:

npm install -g nombre-paquete@latest

Limpieza y Mantenimiento

Eliminar paquetes

Para desinstalar un paquete y eliminarlo del package.json :(docs.npmjs.com)

```
C:\Users\USUARIO\Proyecto-RA>npm uninstall chalk

removed 1 package, and audited 417 packages in 7s

63 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\Users\USUARIO\Proyecto-RA>
```