

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

<b>Actividad 1 – No se presenta evidencias .....</b>	3
<b>POO (Programación orientada a objetos) .....</b>	3
<b>Actividad 2 - Preguntas .....</b>	3
<b>¿Qué es un gestor dependencias de código?.....</b>	3
<b>¿Qué es npm?.....</b>	3
<b>¿Para qué se utiliza principalmente npm? .....</b>	4
<b>¿Qué es el versionado semántico? .....</b>	4
<b>¿Qué son las dependencias locales?.....</b>	4
<b>¿Qué son las dependencias de desarrollo?.....</b>	5
<b>¿Qué son las dependencias globales?.....</b>	5
<b>¿Qué es el archivo package.json y que apartados tiene y cuál es su utilidad? ...</b>	5
<b>¿Qué es el archivo package-lock.json y que utilidad tiene?.....</b>	6
<b>¿Qué es la carpeta node_modules en un proyecto de npm?.....</b>	6
<b>Command Line Cheat Sheet.....</b>	6
<b>Actividad 3 – Manejo Práctico del gestor de dependencias NPM - Bitácora .....</b>	7
<b>Primer paso: Inicialización de un Proyecto npm .....</b>	8
<b>Segundo paso: Instalación de dependencias.....</b>	8
<b>Tercer paso: Gestión de paquetes.....</b>	9
<b>Cuarto paso: Scripts y actualizaciones.....</b>	10
<b>Actualizar dependencias.....</b>	12
<b>Actualizar Dependencias Globales.....</b>	13
a. Ver qué paquetes globales están desactualizados .....	13
b. Actualizar todos los paquetes globales.....	13
c. Actualizar un paquete global específico a la última versión .....	14
<b>Limpieza y Mantenimiento.....</b>	14
<b>Eliminar paquetes .....</b>	14

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.

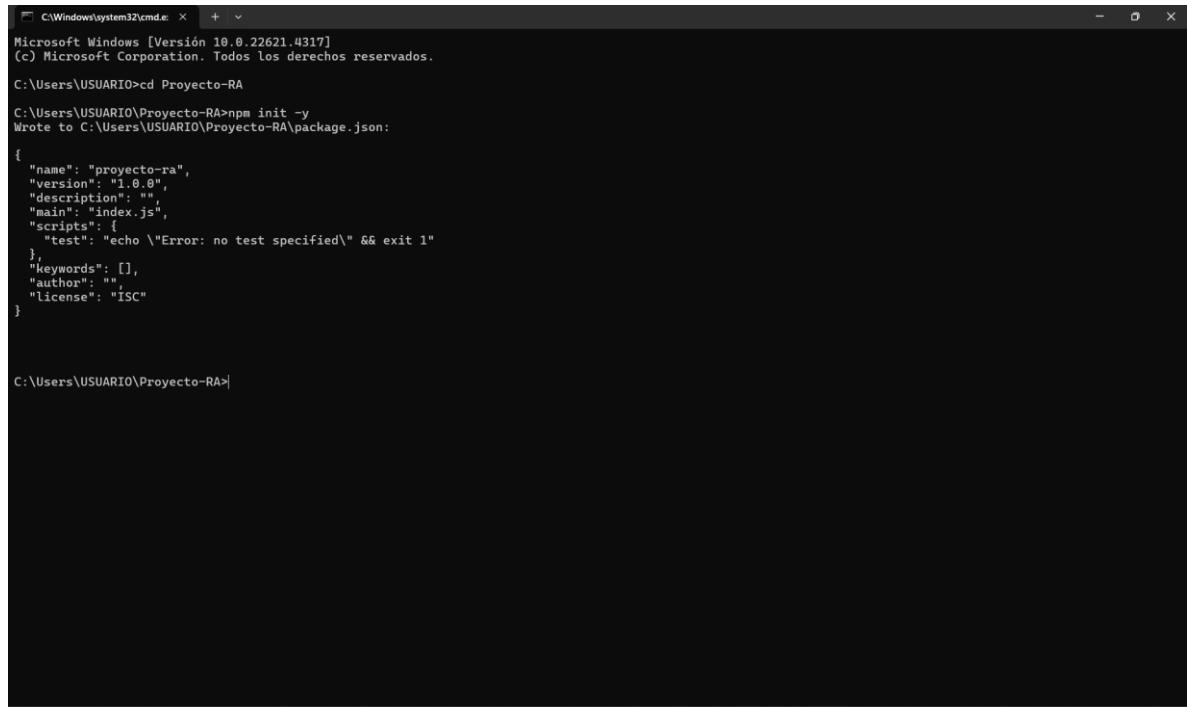
Contiene el código de las librerías que el proyecto necesita para funcionar.

### 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	<b>Cómo instalar dependencias globales</b>	<b>Cómo visualizar las dependencias instaladas</b>

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
<b>Cómo instalar una versión específica de un paquete</b>	<b>Cómo crear un comando en tu proyecto (scripts en el package.json )</b>	<b>Cómo actualizar dependencias</b>
El formato básico es: npm install <a href="#">express@4.17.1</a> Instalar una versión de desarrollo específica: npm install <a href="#">nodemon@3.0.1</a> – save-dev	Primero se ubica la sección “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 específica: npm install express@latest Actualizar todas las dependencias: npm update
<b>Cómo eliminar paquetes</b>	<b>Cómo actualizar node_modules</b>	
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	

## Primer paso: Inicialización de un Proyecto npm



A screenshot of a Windows Command Prompt window titled 'C:\Windows\system32\cmd.e...' showing the output of an 'npm init -y' command. The window title bar also shows '+ ^'. The command prompt text is as follows:

```
Microsoft Windows [Versión 10.0.22621.4317]
(c) Microsoft Corporation. Todos los derechos reservados.

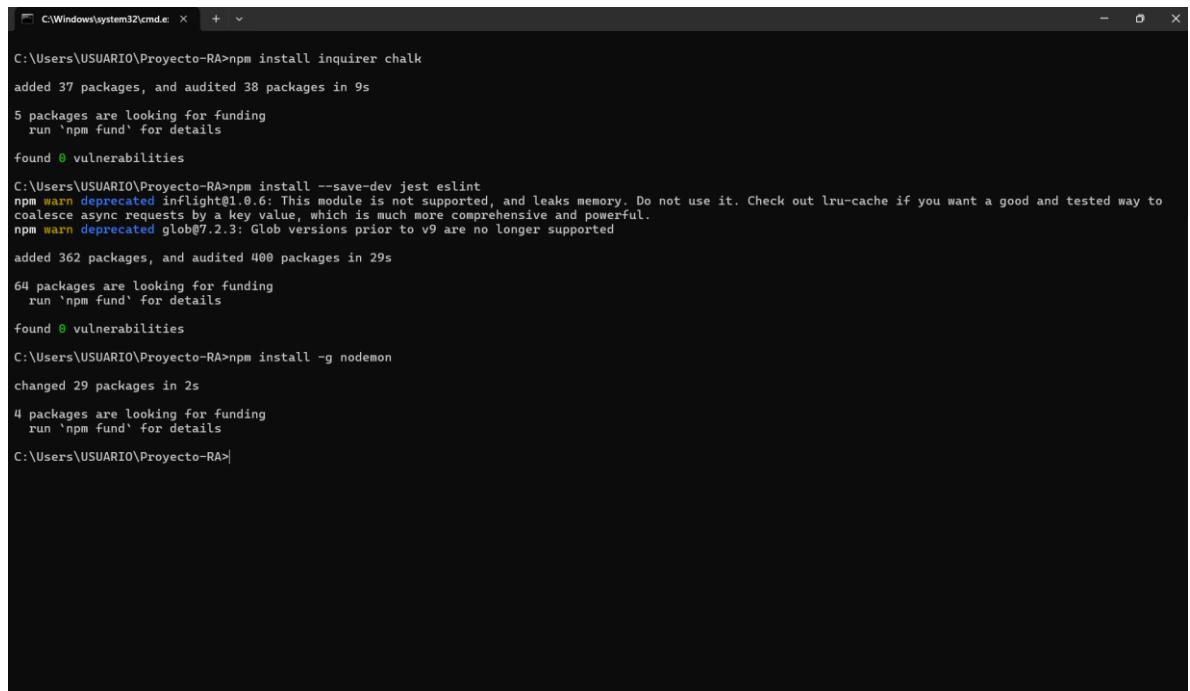
C:\Users\USUARIO>cd Proyecto-RA
C:\Users\USUARIO\Proyecto-RA>npm init -y
Wrote to C:\Users\USUARIO\Proyecto-RA\package.json:

{
  "name": "proyecto-ra",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}

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

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



```
C:\Users\USUARIO\Proyecto-RA>npm install inquirer chalk
added 37 packages, and audited 38 packages in 9s
5 packages are looking for funding
  run 'npm fund' for details
found 0 vulnerabilities

C:\Users\USUARIO\Proyecto-RA>npm install --save-dev jest eslint
npm warn deprecated inflight@1.0.6: This module is not supported, and leaks memory. Do not use it. Check out lru-cache if you want a good and tested way to
coalesce async requests by a key value, which is much more comprehensive and powerful.
npm warn deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported
added 362 packages, and audited 400 packages in 29s
64 packages are looking for funding
  run 'npm fund' for details
found 0 vulnerabilities

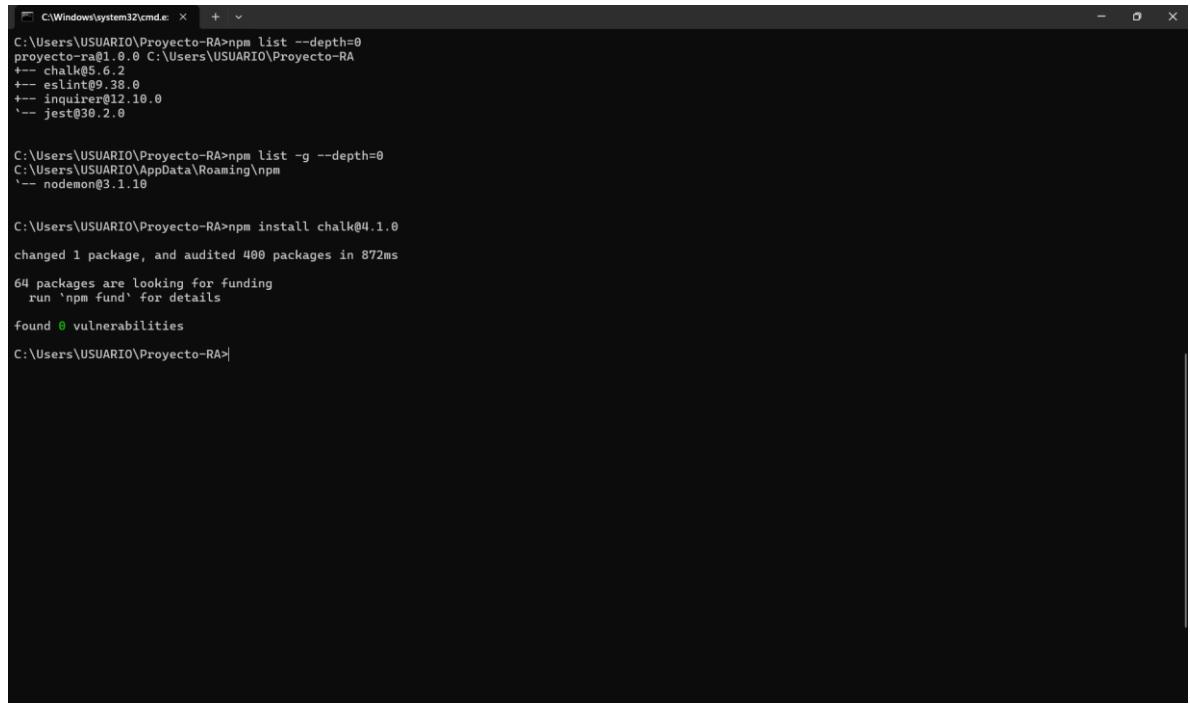
C:\Users\USUARIO\Proyecto-RA>npm install -g nodemon
changed 29 packages in 2s
4 packages are looking for funding
  run 'npm fund' for details
C:\Users\USUARIO\Proyecto-RA>
```

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



```
C:\Windows\system32\cmd.exe + v
C:\Users\USUARIO\Proyecto-RA>npm list --depth=0
proyecto-ra@1.0.0 C:\Users\USUARIO\Proyecto-RA
+-- chalk@5.6.2
+-- eslint@9.38.0
+-- inquirer@12.10.0
`-- jest@38.2.0

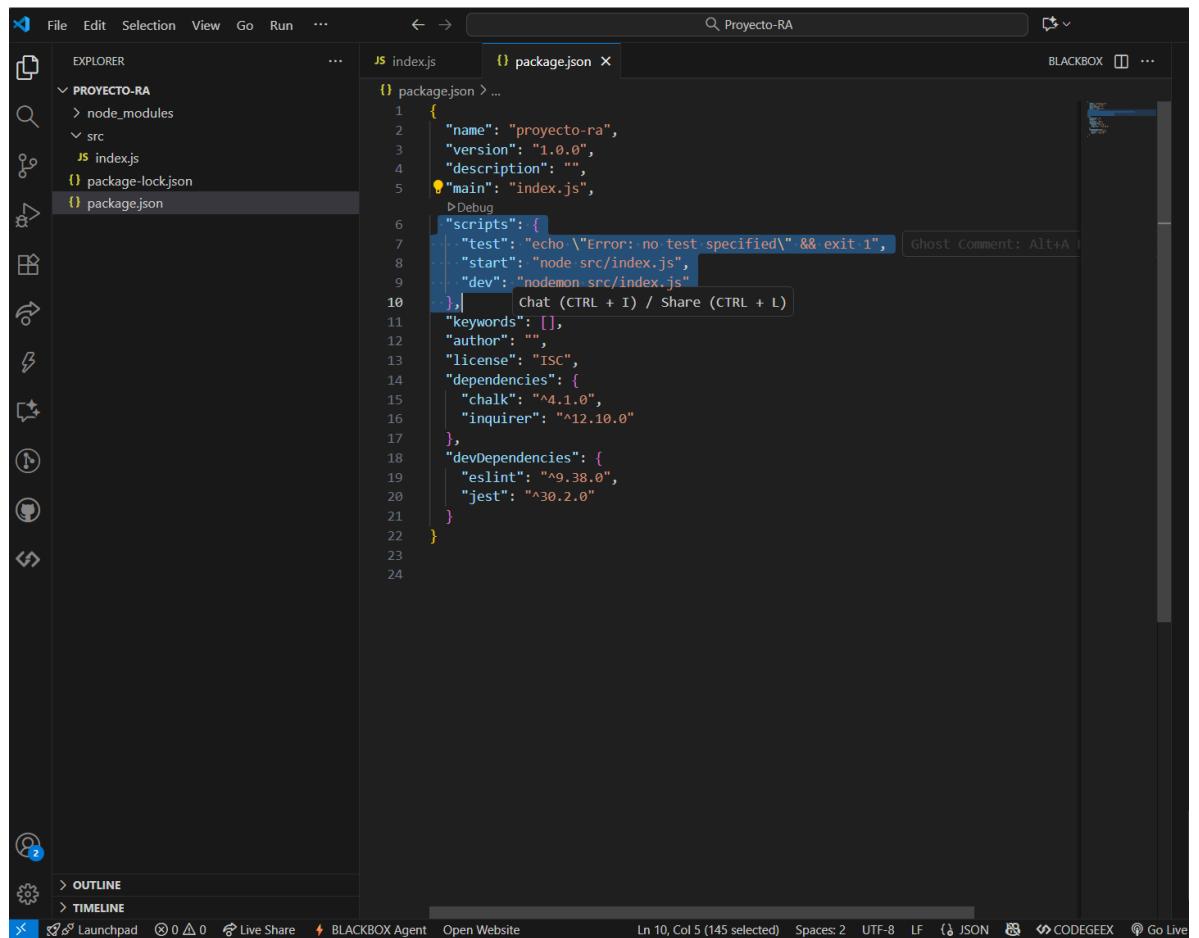
C:\Users\USUARIO\Proyecto-RA>npm list -g --depth=0
C:\Users\USUARIO\AppData\Roaming\npm
`-- nodemon@3.1.10

C:\Users\USUARIO\Proyecto-RA>npm install chalk@4.1.0
changed 1 package, and audited 400 packages in 872ms
64 packages are looking for funding
  run 'npm fund' for details
found 0 vulnerabilities
C:\Users\USUARIO\Proyecto-RA>
```

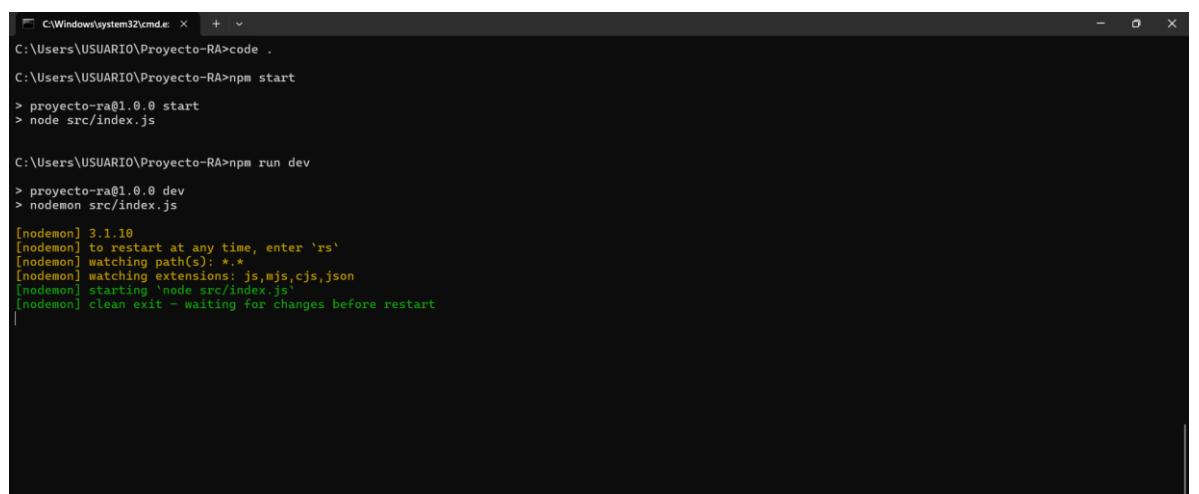
Con `npm list --depth=0` se ven las dependencias locales y con `npm list -g --depth=0` las globales.

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



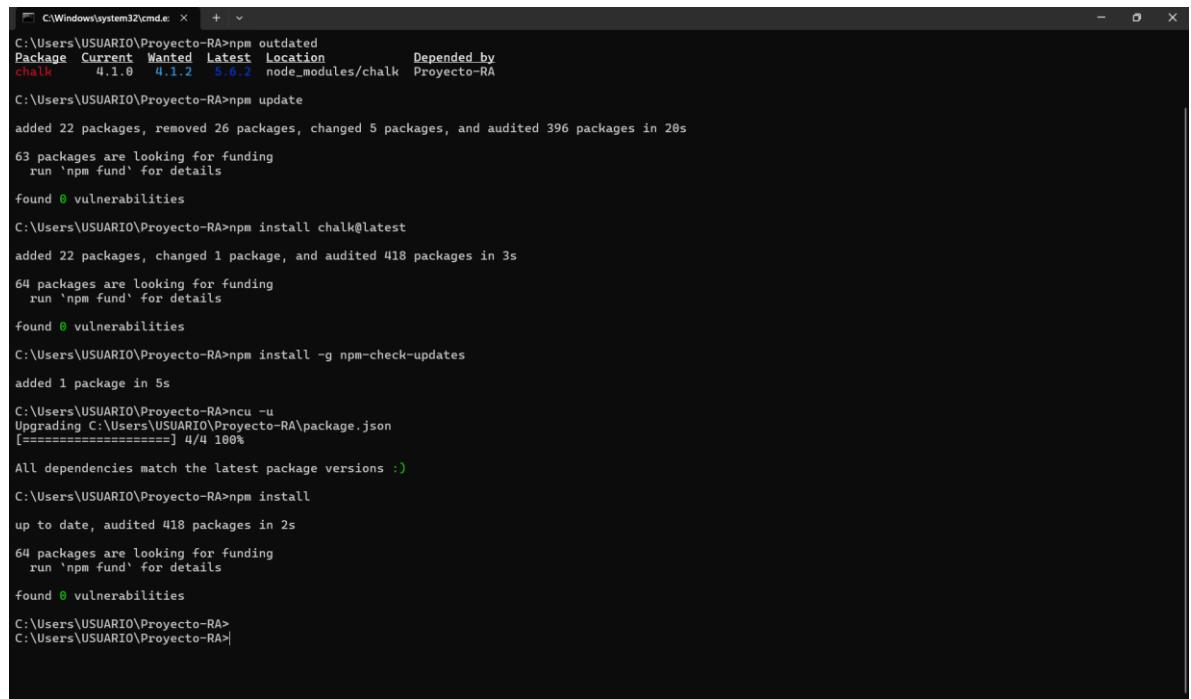
```
1 {  
2   "name": "proyecto-ra",  
3   "version": "1.0.0",  
4   "description": "",  
5   "main": "index.js",  
6   "scripts": {  
7     "test": "echo \"Error: no test specified\" && exit 1",  
8     "start": "node src/index.js",  
9     "dev": "nodemon src/index.js"  
10    },  
11    "keywords": [],  
12    "author": "",  
13    "license": "ISC",  
14    "dependencies": {  
15      "chalk": "^4.1.0",  
16      "inquirer": "^12.10.0"  
17    },  
18    "devDependencies": {  
19      "eslint": "^8.0.0",  
20      "jest": "^30.2.0"  
21    }  
22  }  
23  
24
```



```
C:\Windows\system32\cmd.exe +   
C:\Users\USUARIO\Proyecto-RA>code .  
C:\Users\USUARIO\Proyecto-RA>npm start  
> proyecto-ra@1.0.0 start  
> node src/index.js  
  
C:\Users\USUARIO\Proyecto-RA>npm run dev  
> proyecto-ra@1.0.0 dev  
> nodemon src/index.js  
[nodemon] 3.1.10  
[nodemon] to restart at any time, enter 'rs'  
[nodemon] watching path(s): *.*  
[nodemon] watching extensions: js,mjs,cjs,json  
[nodemon] starting 'node src/index.js'  
[nodemon] clean exit - waiting for changes before restart
```

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



```
C:\Windows\system32\cmd.exe: + -
C:\Users\USUARIO\Proyecto-RA>npm outdated
Package Current Wanted Latest Location      Depended_by
chalk   4.1.0  4.1.2  5.6.2  node_modules(chalk)  Proyecto-RA

C:\Users\USUARIO\Proyecto-RA>npm update
added 22 packages, removed 26 packages, changed 5 packages, and audited 396 packages in 20s
63 packages are looking for funding
  run 'npm fund' for details
found 0 vulnerabilities

C:\Users\USUARIO\Proyecto-RA>npm install chalk@latest
added 22 packages, changed 1 package, and audited 418 packages in 3s
64 packages are looking for funding
  run 'npm fund' for details
found 0 vulnerabilities

C:\Users\USUARIO\Proyecto-RA>npm install -g npm-check-updates
added 1 package in 5s

C:\Users\USUARIO\Proyecto-RA>ncu -u
Upgrading C:\Users\USUARIO\Proyecto-RA\package.json
[=====] 4/4 100%
All dependencies match the latest package versions :)
```

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

## Actualizar Dependencias Globales

```
C:\Users\USUARIO\Proyecto-RA>npm outdated -g --depth=0
C:\Users\USUARIO\Proyecto-RA>npm update -g
changed 30 packages in 5s
4 packages are looking for funding
run 'npm fund' for details
C:\Users\USUARIO\Proyecto-RA>npm install -g nodemon@latest
changed 29 packages in 850ms
4 packages are looking for funding
run 'npm fund' for details
C:\Users\USUARIO\Proyecto-RA>
```

Las dependencias globales son aquellas herramientas instaladas con el flag -g, como nodemon, npm-check-updates, o cualquier herramienta CLI que deseas 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 :[\(docs.npmjs.com\)](https://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>
```

## Actividad 4 – Trabajo con vite

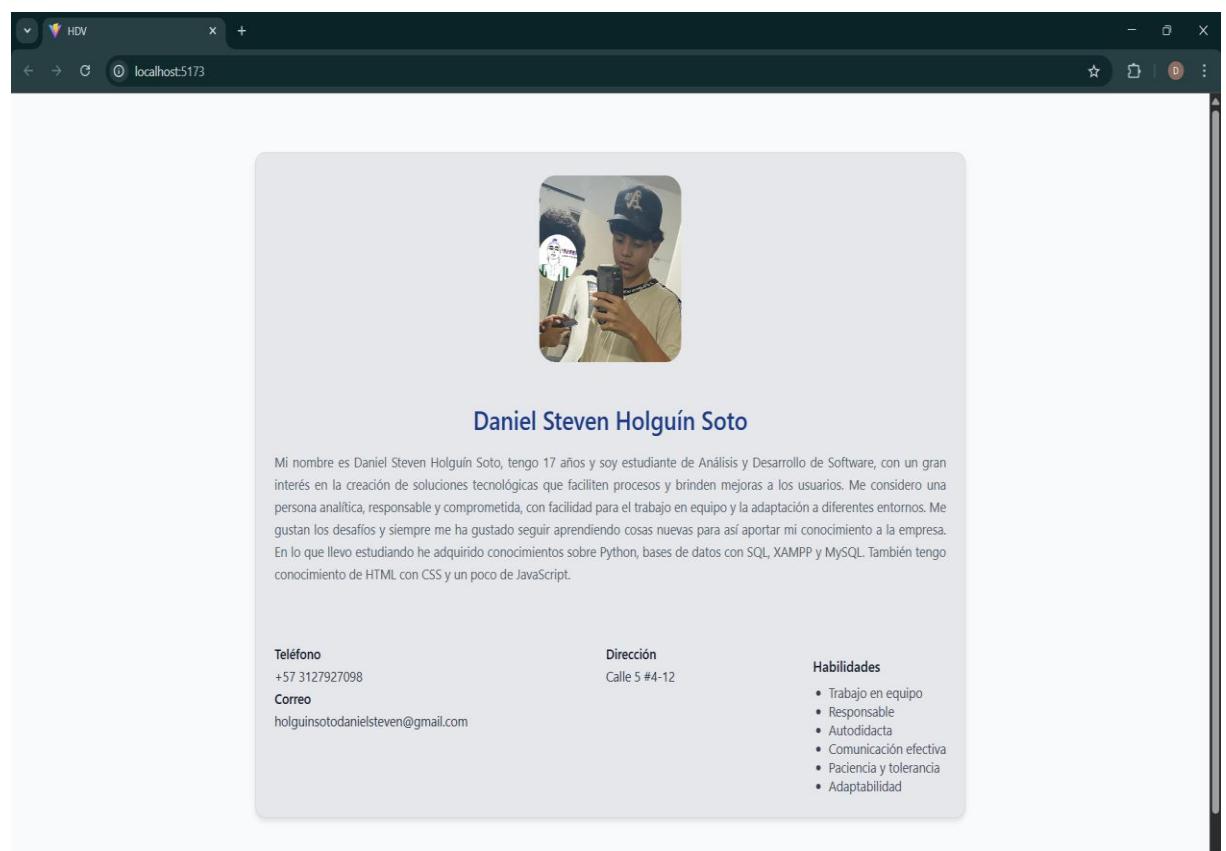
En este punto instale la librería de Vite con el siguiente comando:

```
npm create vite@latest exp_app -- --template react
```

Luego cree un archivo llamado hdv.jsx y otro hdv.css, tuve que conectar el index.html con el main.jsx, y por último este main llama a hdv.jsx y termina de importar la decoración (hdv.css).

Lo que hice se puede visualizar en la primera carpeta que tengo en el GitHub.

Interfaz grafica:

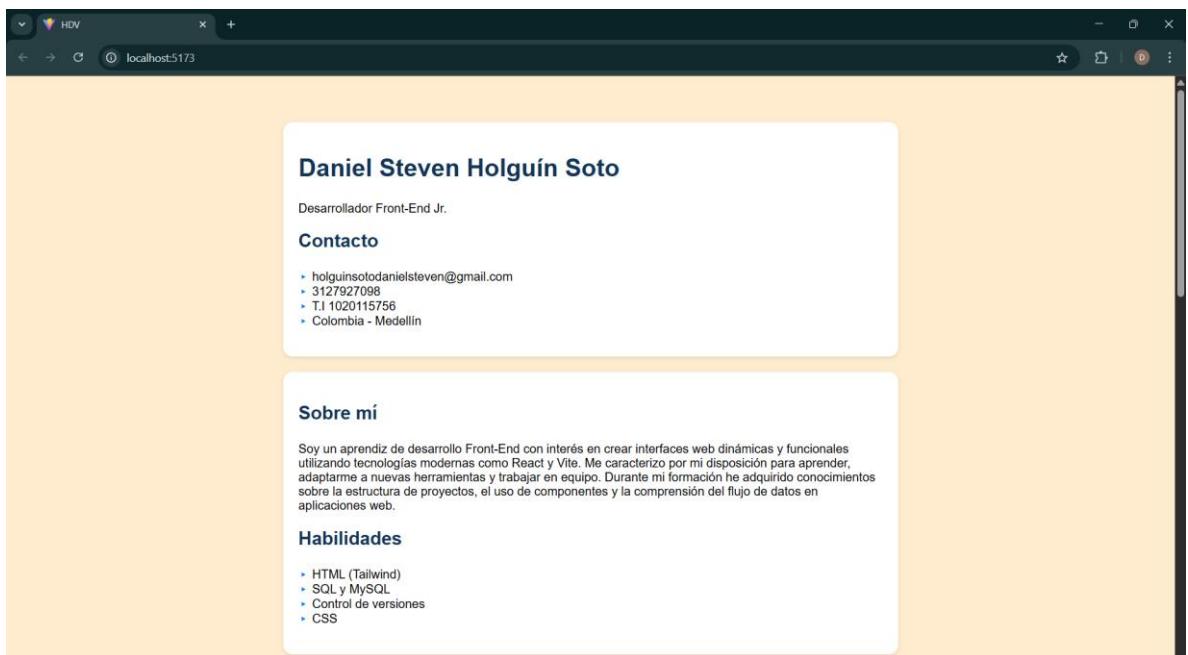


## Actividad 5 – Componentes, anidación y control de versiones.

En este punto tuve que crear una carpeta llamada componentes dentro de la carpeta src, luego cree los archivos que pide en el documento (Experiencia.jsx, Habilidades.jsx, etc). Cree un archivo principal llamado App.jsx que va a importar todos estos y por último le puse el diseño en el archivo App.css.

En este punto aprendí un poco más sobre la conexión del main con la App y la manera correcta de hacer dicha conexión, si por ejemplo esta esta en una carpeta, etc.

### Interfaz grafica:



HDV

localhost:5173

**Tecnología en Análisis y Desarrollo de Software — SENA (2025 - 2027)**

Formación enfocada en el desarrollo de aplicaciones web y móviles, bases de datos y metodologías ágiles para la gestión de proyectos de software.

**Bachillerato Académico — I.E Alberto Diaz Muñoz (2024)**

Finalicé mis estudios de educación media con énfasis en ciencias y tecnología, desarrollando habilidades en trabajo en equipo y pensamiento lógico.

**Curso de Inglés Nivel 1 — SENA (2025)**

Introducción al idioma inglés con enfoque en vocabulario básico, estructuras gramaticales y comunicación oral sencilla.

**Curso de Inglés Nivel 2 — SENA (2025)**

Continuación del aprendizaje del idioma, con énfasis en comprensión auditiva, lectura y conversación a nivel intermedio básico.

**Desarrollador Front-End Jr. — TechNova Solutions (2025)**

Participé en la creación de una página web corporativa utilizando React y Vite, implementando componentes reutilizables y optimizando el rendimiento. Colaboré con el equipo de diseño para asegurar una experiencia de usuario fluida y moderna.

**Asistente de Desarrollo Web — CodeFactory (2024)**

Apoyé en la maquetación de sitios web con HTML, CSS y JavaScript. Realicé pruebas de compatibilidad entre navegadores y ayudé a mejorar la accesibilidad del contenido.

HDV

localhost:5173

**Asistente de Desarrollo Web — CodeFactory (2024)**

Apoyé en la maquetación de sitios web con HTML, CSS y JavaScript. Realicé pruebas de compatibilidad entre navegadores y ayudé a mejorar la accesibilidad del contenido.

**Proyecto Académico: "Gestor de Tareas" — SENA (2024)**

Desarrollé una aplicación web para gestionar tareas diarias, empleando React para el manejo de estados y componentes. Implementé un diseño responsive con CSS y funciones básicas de CRUD.

**Prácticas en InnovApp — (2023)**

Colaboré en el desarrollo de una aplicación web de portafolio personal, utilizando Vite como herramienta de compilación y GitHub para control de versiones. Aprendí sobre organización de código y despliegue en entornos web.

