

Título del documento:

Bitácora N°1 preparación de Entorno de Desarrollo

Nombre del estudiante:

Liliana Panesso

Institución:

SENA

Curso:

Análisis y Desarrollo de Software

Profesor:

Luis Sanchez

Ficha:

3223873

Introducción

La presente bitácora documenta el proceso de desarrollo de un proyecto práctico en React, siguiendo una metodología ordenada y registrando cada etapa con capturas de pantalla y notas explicativas bajo las normas APA 7ª edición. El objetivo principal fue instalar y configurar las dependencias necesarias, organizar la estructura del proyecto y validar el funcionamiento de las herramientas utilizadas. Asimismo, se realizaron pruebas de instalación y desinstalación de librerías, modificaciones temporales en el archivo package.json y verificaciones de actualizaciones, con el fin de comprender el impacto de cada acción en el entorno de desarrollo. Este registro busca reflejar el aprendizaje progresivo y la importancia de mantener una documentación clara y profesional.

Desarrollo (Bitácora)

- ***Inicialización del proyecto:*** Se generó el archivo package.json mediante npm init -y.
- ***Instalación de dependencias:*** Se añadieron las librerías inquirer y chalk para interacción en consola y estilos visuales.
- ***Herramientas de desarrollo:*** Se instalaron jest y eslint para pruebas automáticas y control de calidad del código.
- ***Instalación global:*** Se incorporó nodemon para reiniciar automáticamente el servidor al detectar cambios.
- ***Verificación de dependencias:*** Se usaron los comandos npm list y npm list -g para confirmar las librerías locales y globales.
- ***Organización del proyecto:*** Se crearon nuevas carpetas para mejorar la estructura y facilitar el desarrollo.
- ***Actualización de dependencias:*** Se ejecutaron npm outdated y npm outdated -g confirmando que todas las librerías estaban al día.
- ***Modificación de scripts:*** Se añadió un comando "start": "node src/index.js" en package.json como prueba inicial.

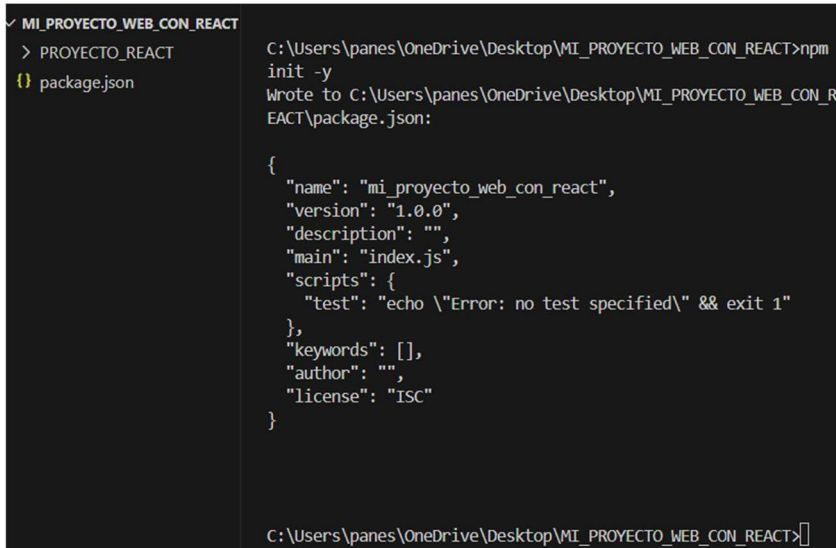
□ ***Prueba de desinstalación:*** Se eliminó temporalmente la librería chalk para comprobar el comportamiento del proyecto, con la intención de reinstalarla posteriormente.

- ***Reinstalación:*** Se volvió a instalar chalk junto con inquirer para restablecer el entorno.

Cada acción fue acompañada de capturas de pantalla presentadas como *Figuras*, numeradas y descritas según APA 7ª edición.

Bitácora N°1 preparación de Entorno de Desarrollo

Inicializar Un Proyecto npm



```
✓ MI_PROYECTO_WEB_CON_REACT
> PROYECTO_REACT
{} package.json

C:\Users\panes\OneDrive\Desktop\MI_PROYECTO_WEB_CON_REACT>npm
init -y
Wrote to C:\Users\panes\OneDrive\Desktop\MI_PROYECTO_WEB_CON_R
EACT\package.json:

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

C:\Users\panes\OneDrive\Desktop\MI_PROYECTO_WEB_CON_REACT>
```

Figura 1

Archivo package.json generado automáticamente tras la inicialización del proyecto.

Nota. Captura de pantalla propia del entorno de desarrollo.

Una vez creadas las carpetas correspondientes para la estructura del proyecto, se procedió a inicializarlo mediante el comando `npm init -y`, lo cual generó automáticamente el archivo `package.json`. Este archivo contiene la configuración básica del proyecto, incluyendo su nombre, versión, punto de entrada y otros metadatos esenciales para la gestión de dependencias y scripts.

2. Instalación De Dependencias

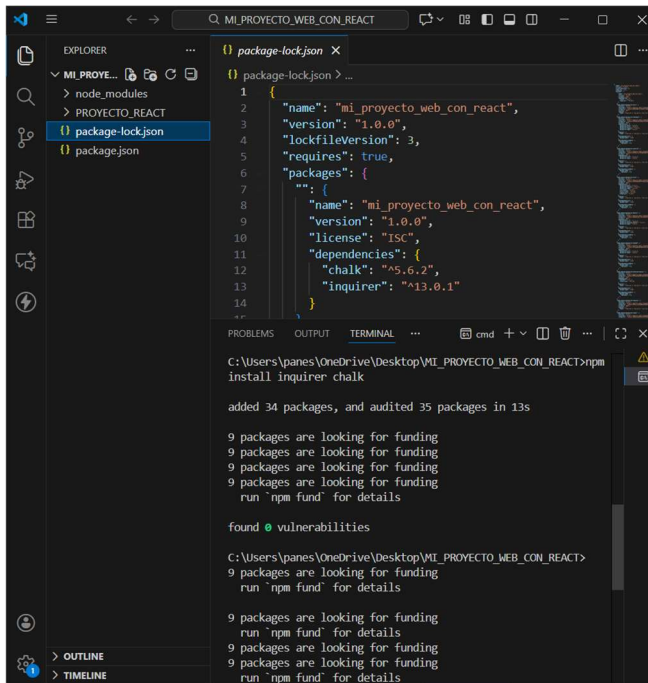


Figura 2

Instalación de las librerías inquirer y chalk en el proyecto.

Nota. Captura de pantalla propia del entorno de desarrollo.

Después de la creación de la estructura inicial del proyecto y la generación del archivo package.json, se procedió a instalar las dependencias necesarias mediante el comando `npm install inquirer chalk`. Esta acción incorporó las librerías **Inquirer**, utilizada para construir interfaces interactivas en la línea de comandos, y **Chalk**, destinada a aplicar colores y estilos al texto en consola. La instalación se completó correctamente, añadiendo 34 paquetes al entorno y actualizando el archivo package-lock.json para registrar las versiones exactas de cada dependencia.

Instalar Dependencias de Desarrollo

```
C:\Users\panes\OneDrive\Desktop\MI_PROYECTO_WEB_CON_REACT>npm
install --save-dev jest eslint
npm warn deprecated inflight@1.0.6: This module is not support
ed, and leaks memory. Do not use it. Check out lru-cache if yo
u want a good and tested way to coalesce async requests by a k
ey 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 402 packages, and audited 437 packages in 28s

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

found 0 vulnerabilities
```

Figura 3

Instalación de dependencias de desarrollo jest y eslint.

Nota. Captura de pantalla propia del entorno de desarrollo.

Se instalaron las herramientas de desarrollo Jest y ESLint mediante el comando `npm install --save-dev jest eslint`, con el objetivo de realizar pruebas automáticas y mantener el código limpio y ordenado. Aunque se mostraron algunas advertencias por paquetes obsoletos, la instalación fue exitosa, se añadieron 402 paquetes y no se detectaron vulnerabilidades.

Instalar Dependencias Globales

```
C:\Users\panes\OneDrive\Desktop\MI_PROYECTO_WEB_CON_REACT>npm
install -g nodemon

changed 29 packages in 4s

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

C:\Users\panes\OneDrive\Desktop\MI_PROYECTO_WEB_CON_REACT>
```

Figura 4

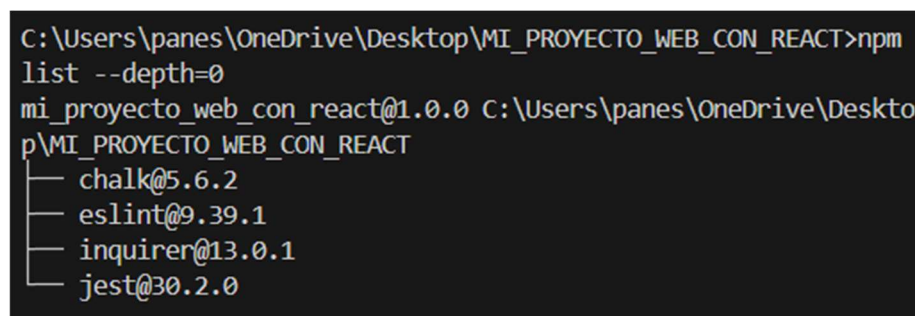
Instalación global de la herramienta nodemon para reinicio automático del servidor.

Nota. Captura de pantalla propia del entorno de desarrollo.

Se instaló la herramienta **nodemon** de forma global utilizando el comando `npm install -g nodemon`. Esta utilidad permite reiniciar automáticamente el servidor cada vez que se detectan cambios en los archivos del proyecto, facilitando el flujo de trabajo durante el desarrollo. La instalación fue rápida y exitosa, y aunque se mostraron algunos paquetes en búsqueda de financiación, no se reportaron errores ni vulnerabilidades.

3. Gestión de Paquetes

Visualizar Paquetes Instalados



```
C:\Users\panes\OneDrive\Desktop\MI_PROYECTO_WEB_CON_REACT>npm
list --depth=0
mi_proyecto_web_con_react@1.0.0 C:\Users\panes\OneDrive\Desкто
p\MI_PROYECTO_WEB_CON_REACT
├── chalk@5.6.2
├── eslint@9.39.1
├── inquirer@13.0.1
└── jest@30.2.0
```

Figura 5

Listado de dependencias locales instaladas en el proyecto mediante npm list.

Nota. Captura de pantalla propia del entorno de desarrollo.

Se ejecutó el comando `npm list --depth=0` para verificar las dependencias instaladas en el proyecto. El resultado mostró que están presentes las librerías chalk, eslint, inquirer y jest, todas correctamente registradas en el entorno. Esta verificación permite confirmar que las herramientas necesarias para el desarrollo, pruebas y estilo de código están disponibles y listas para usarse.


```
C:\Users\panes\OneDrive\Desktop\MI_PROYECTO_WEB_CON_REACT>npm
list -g --depth=0
C:\Users\panes\AppData\Roaming\npm
├─ nodemon@3.1.11

C:\Users\panes\OneDrive\Desktop\MI_PROYECTO_WEB_CON_REACT>
```

Figura 6

Verificación de dependencias globales instaladas con npm list -g.

Nota. Captura de pantalla propia del entorno de desarrollo.

Se ejecutó el comando `npm list -g --depth=0` para verificar las dependencias instaladas globalmente en el sistema. El resultado confirmó que la herramienta **nodemon**, en su versión 3.1.11, está instalada correctamente. Esta utilidad permite reiniciar automáticamente el servidor cada vez que se detectan cambios en los archivos, lo que facilita el desarrollo continuo sin necesidad de reinicios manuales.

4. Scripts y Actualizaciones

Crear la Estructura de Directorios del Proyecto

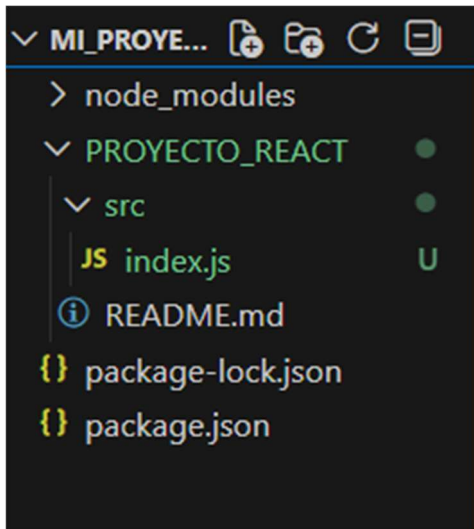


Figura 7

Estructura de carpetas del proyecto MI_PROYECTO_WEB_CON_REACT en VS Code.

Nota. Captura de pantalla propia del entorno de desarrollo.

Después de instalar las dependencias necesarias, se realizó la creación de las carpetas nuevas propuestas para organizar mejor la estructura del proyecto.

Actualizar Dependencias

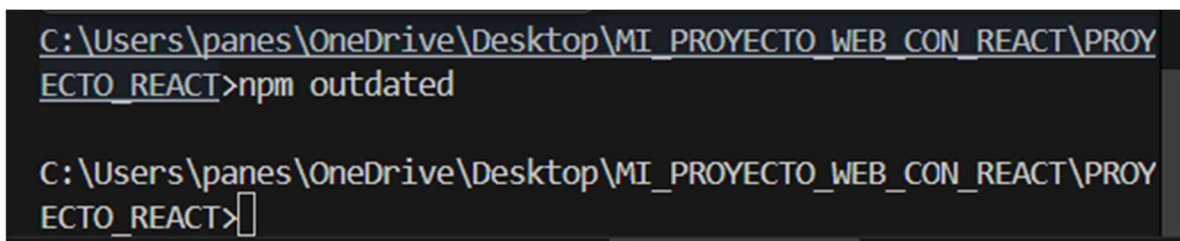


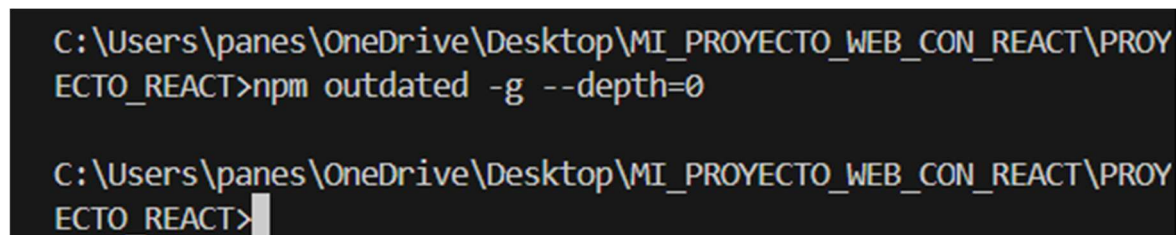
Figura 8

Verificación de actualizaciones de dependencias locales con npm outdated.

Nota. Captura de pantalla propia del entorno de desarrollo.

Se ejecutó el comando `npm outdated` para comprobar si alguna de las dependencias del proyecto requería actualización. El resultado indicó que todas las librerías instaladas están al día, lo que confirma que el entorno de desarrollo se mantiene actualizado y listo para continuar con las siguientes etapas del proyecto.

2. Actualizar Dependencias Globales



```
C:\Users\panes\OneDrive\Desktop\MI_PROYECTO_WEB_CON_REACT\PROYECTO_REACT>npm outdated -g --depth=0

C:\Users\panes\OneDrive\Desktop\MI_PROYECTO_WEB_CON_REACT\PROYECTO_REACT>
```

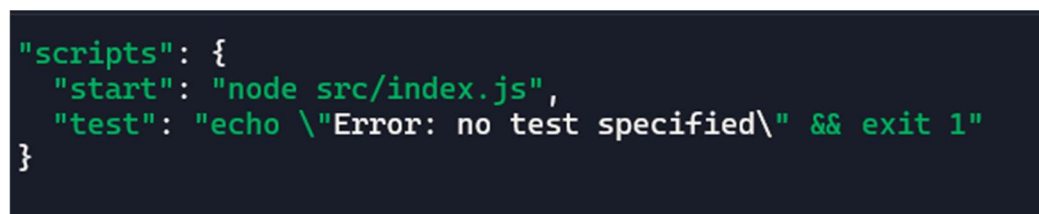
Figura 9

Verificación de actualizaciones de dependencias globales con `npm outdated -g`.

Nota. Captura de pantalla propia del entorno de desarrollo.

Se ejecutó el comando `npm outdated -g --depth=0` para verificar si alguna de las herramientas instaladas globalmente requería actualización. El resultado indicó que todas las dependencias globales están actualizadas, lo que garantiza que el entorno de desarrollo cuenta con versiones recientes y funcionales para continuar trabajando sin inconvenientes.

Crear un Comando (Script) en el Proyecto



```
"scripts": {
  "start": "node src/index.js",
  "test": "echo \"Error: no test specified\" && exit 1"
}
```

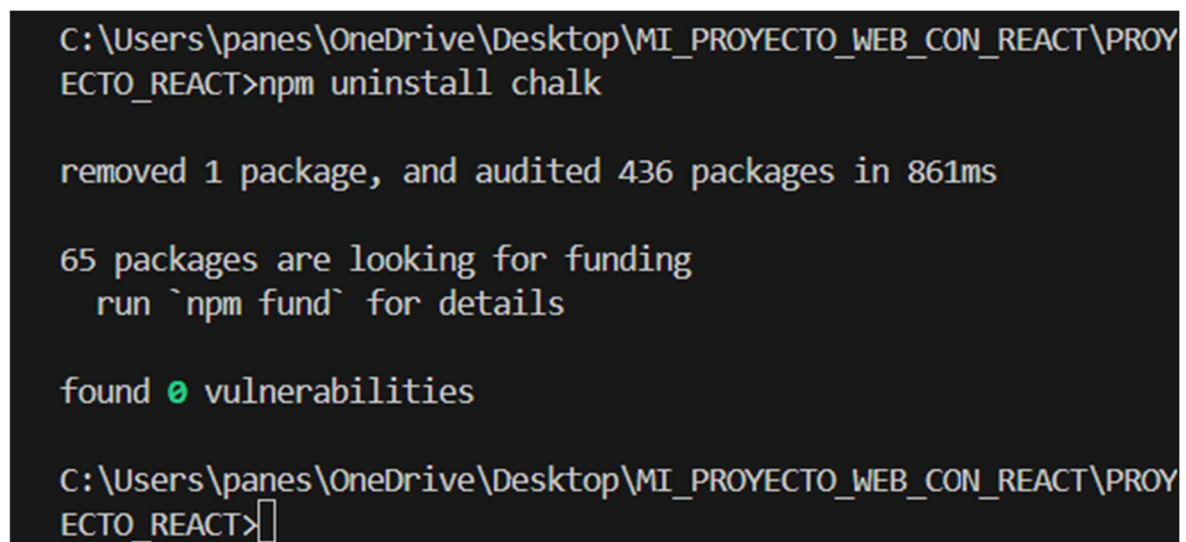
Figura 10

Modificación temporal del script "start" en package.json para pruebas iniciales.

Nota. Captura de pantalla propia del entorno de desarrollo.

Como parte del proceso de configuración inicial, se realizó una modificación en el archivo package.json para agregar un comando personalizado en la sección "scripts". Se definió el script "start" con el valor "node src/index.js", lo que permite ejecutar el archivo principal del proyecto desde la terminal utilizando npm start. Esta configuración se implementó únicamente con fines de prueba y será actualizada conforme avance el desarrollo del proyecto y se definan nuevas necesidades técnicas

5. Limpieza y Mantenimiento



```
C:\Users\panes\OneDrive\Desktop\MI_PROYECTO_WEB_CON_REACT\PROYECTO_REACT>npm uninstall chalk

removed 1 package, and audited 436 packages in 861ms

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

found 0 vulnerabilities

C:\Users\panes\OneDrive\Desktop\MI_PROYECTO_WEB_CON_REACT\PROYECTO_REACT>
```

Figura 11

Desinstalación temporal de la librería chalk para pruebas internas.

Nota. Captura de pantalla propia del entorno de desarrollo.

Como parte de una prueba puntual dentro del entorno de desarrollo, se ejecutó el comando `npm uninstall chalk` para comprobar el comportamiento del proyecto sin esta dependencia. Esta acción no representa una modificación definitiva, ya que se tiene previsto reinstalar la librería más adelante. La desinstalación se completó correctamente, sin generar errores ni vulnerabilidades, y permitió validar el impacto de su ausencia en el flujo del proyecto

Conclusiones

El desarrollo del taller práctico permitió afianzar conocimientos sobre la gestión de dependencias en proyectos con Node.js y React, así como la importancia de mantener un entorno de trabajo ordenado y actualizado. La instalación y configuración de librerías como inquirer, chalk, jest, eslint y nodemon facilitaron tanto la interacción en consola como la validación del código. Las pruebas de desinstalación y modificación de scripts demostraron la flexibilidad del entorno y la necesidad de documentar cada cambio para evitar confusiones futuras. Finalmente, la aplicación de las normas APA 7ª edición en la presentación de la bitácora aseguró una estructura clara, profesional y estandarizada, que respalda el proceso de aprendizaje y desarrollo técnico.