# Instalación de Node.js, herramientas de trabajo, uso de paquetes y npm


Modulo1 - Sesión 5

# Entrada.

En la primer sesión realizamos la instalación básica de node.js para trabajar los primeros ejercicios prácticos.

Todo bien hasta el momento de la instalación ¿?

# NPM

npm (Node Package Manager) es el sistema de gestión de paquetes más utilizado en el ecosistema de Node.js. Permite a los desarrolladores instalar, compartir, actualizar y administrar las dependencias de sus proyectos de forma eficiente.

# NPM

npm facilita la gestión de dependencias en proyectos Node.js. Puedes especificar las dependencias necesarias para tu proyecto en un archivo package.json y luego usar npm install para instalar todas las dependencias de forma automática.

Los desarrolladores pueden publicar sus propios paquetes en el registro público de npm para que otros puedan utilizarlos. Esto fomenta la colaboración y el intercambio de código en la comunidad de Node.js.

# NPM

Los desarrolladores pueden publicar sus propios paquetes en el registro público de npm para que otros puedan utilizarlos. Esto fomenta la colaboración y el intercambio de código en la comunidad de Node.js.

npm admite la ejecución de scripts y comandos definidos en el archivo package.json. Por ejemplo, puedes definir scripts para iniciar tu aplicación, ejecutar pruebas, construir el proyecto, entre otros, y luego ejecutarlos usando npm run.

npm permite configurar varias opciones a nivel de proyecto o de usuario, como la versión de Node.js a utilizar, la gestión de versiones de paquetes, la cache de npm, entre otros.

# Creación de proyectos.

Cuando comenzamos con el desarrollo utilizando node.js es una buena practica utilizar el comando init.

npm init crea automáticamente un archivo package.json en el directorio de tu proyecto. Este archivo es importante porque contiene información sobre tu proyecto, como el nombre, la versión, las dependencias, los scripts, el autor y más. Mantener un package.json bien organizado y actualizado facilita la gestión de dependencias y el mantenimiento del proyecto a largo plazo.

# Creación de proyectos.

Al ejecutar npm init, puedes responder preguntas sobre las opciones de configuración de tu proyecto, como el nombre, la versión, la descripción, la entrada principal (archivo JavaScript principal), el autor, la licencia y más. Estas opciones proporcionan información útil y descriptiva sobre tu proyecto, lo que facilita su comprensión y colaboración con otros desarrolladores.

## Siempre debo de usar el comando init

Aunque no es obligatorio utilizar npm init para iniciar un proyecto en Node.js, es una práctica recomendada debido a los beneficios que proporciona, como la creación del archivo package.json, la configuración detallada del proyecto, la gestión de dependencias y la definición de scripts. Esto ayuda a mantener tu proyecto organizado, estructurado y fácil de manejar a medida que crece y se desarrolla.

# Comando npm

En Node.js, puedes utilizar el gestor de paquetes npm (Node Package Manager) para instalar, actualizar y eliminar bibliotecas (paquetes).

Para instalar una biblioteca en tu proyecto Node.js, utiliza el comando npm install seguido del nombre del paquete que deseas instalar.

Para actualizar una biblioteca a la última versión disponible, utiliza el comando npm update seguido del nombre del paquete que deseas actualizar.

Para eliminar una biblioteca de tu proyecto, utiliza el comando npm uninstall seguido del nombre del paquete que deseas eliminar.

# Conclución

Puedes ver una lista de las bibliotecas instaladas en tu proyecto junto con sus versiones utilizando el comando npm list. Por ejemplo, ejecutar npm list te mostrará la estructura de las dependencias instaladas en tu proyecto.

Estos son los comandos básicos que puedes usar en Node.js con npm para instalar, actualizar, eliminar y listar bibliotecas en tu proyecto. Recuerda que es importante mantener tus dependencias actualizadas y gestionarlas correctamente para un desarrollo eficiente y sin problemas.

# Ejecutemos la practica.

¿¿Preguntas??