# Trabajo Práctico: Instalación de herramientas

## Descargar Node.js

```
C:\Users\celin>node -v
v22.18.0
```

## Descargar e Instalar Visual Studio Code

```
C:\Users\celin>code -v
1.103.2
6f17636121051a53c88d3e605c491d22af2ba755
x64
```

## Descargar e Instalar Git

```
C:\Users\celin>git --version
git version 2.51.0.windows.1
```

## Descargar e Instalar MongoDB Compass

```
C:\Users\celin>mongosh --version
2.5.6

C:\Users\celin>mongod --version
db version v8.0.12
Build Info: {
    "version": "8.0.12",
    "gitVersion": "b60fc6875b5fb4b63cc0dbbd8dda0d6d6277921a",
    "modules": [],
    "allocator": "tcmalloc-gperf",
    "environment": {
        "distmod": "windows",
        "distarch": "x86_64",
        "target_arch": "x86_64"
    }
}
```

## Descargar e Instalar Docker Desktop

```
C:\Users\celin>docker --version
Docker version 28.3.2, build 578ccf6
```

# Actividad

Responder las siguientes preguntas sobre node

1. **¿Qué es Node.js y en qué se diferencia de JavaScript en el navegador?**

   Node.js es un entorno de ejecución para JavaScript que funciona en el servidor. Está construido sobre el motor V8 de Google Chrome.

   Diferencia principal:

   - En el navegador, JavaScript se usa para manipular el DOM, manejar eventos y crear interfaces de usuario.
   - En Node.js, JavaScript se usa para crear aplicaciones del lado del servidor, manejar archivos, conexiones de red, bases de datos, etc.

2. **¿Qué son las versiones LTS?**

   LTS (Long Term Support) son versiones de Node.js que reciben soporte y actualizaciones de seguridad por un período prolongado (normalmente 30 meses).
   Se usan en proyectos en producción porque ofrecen estabilidad y compatibilidad.

3. **Explica la diferencia entre un proceso síncrono y uno asíncrono**

   - Síncrono: las tareas se ejecutan una tras otra; si una tarda mucho, las siguientes esperan.
   - Asíncrono: varias tareas pueden ejecutarse sin bloquear el flujo principal, permitiendo atender múltiples operaciones (como consultas a bases de datos o llamadas HTTP) de manera eficiente.

4. **¿Cómo se maneja la asincronía en Node.js? Explica las diferencias entre callbacks, Promises y async/await.**

   Node.js maneja la asincronía con diferentes mecanismos:

   - Callbacks: funciones que se pasan como argumento y se ejecutan cuando una operación termina. Inconveniente: el callback hell.

   - Promises: representan un valor que estará disponible ahora, en el futuro, o nunca. Evitan la anidación excesiva de callbacks.

   - async/await: sintaxis moderna que permite escribir código asíncrono de forma legible, como si fuera síncrono, apoyándose en Promises

5. **¿Qué es el objeto `global` en Node.js y en qué se diferencia de `window` en el navegador?**
   En Node.js, el objeto global es global.
   En el navegador, el objeto global es window.

   Diferencia clave:
   - window contiene APIs propias del navegador (DOM, alert, document, etc.).
   - global en Node.js contiene objetos y funciones específicas del servidor (process, Buffer, require, etc.).

6. **¿Node.js es multithreading?**

   Node.js es single-threaded con un solo Event Loop.
   Sin embargo, internamente puede usar múltiples hilos (por ejemplo, en operaciones de E/S o mediante el módulo worker_threads para tareas pesadas).

7. **¿Qué es el Event Loop en Node.js y cuál es su papel en la ejecución de código asíncrono?**

   Es el mecanismo que permite a Node.js manejar operaciones asíncronas.
   Funciona de la siguiente manera:

   1. Node ejecuta el código principal.
   2. Cuando hay operaciones asíncronas (consultas a BD, timers, etc.), las delega a la API interna.
   3. Una vez completadas, el Event Loop coloca los callbacks o promesas en la cola y las ejecuta cuando el call stack está libre.

8. **Mencionar tres ventajas de Node.js**

   1. Alto rendimiento gracias al motor V8 y al modelo asíncrono.
   2. Un solo lenguaje (JavaScript) tanto en frontend como backend.
   3. Ecosistema rico en librerías a través de npm (Node Package Manager).

9. **menciona tres desventajas de Node.js.**

   1. CPU intensiva: no es ideal para tareas pesadas en cálculo matemático.
   2. Callback hell (aunque mitigado con Promises/async-await).
   3. Estabilidad de librerías externas: muchas dependen de la comunidad y no siempre son confiables.

**10. ¿Podrías nombrar algunas bibliotecas que los desarrolladores utilizan frecuentemente con Node.js?**

- Express.js → framework minimalista para crear servidores y APIs.
- Mongoose → manejo de MongoDB.
- Socket.io → comunicación en tiempo real (WebSockets).
- Lodash → utilidades para manipulación de datos.
- Jest / Mocha → frameworks de testing.
- Passport.js → autenticación.