0. Objetivo de todo esto

Meta: Dejar tu compu lista para poder crear **plataformas** en Node.js (APIs, backends, etc.) y levantar un primer servidor con Express.

---

1. Instalación de Node.js

1.1. Descargar Node.js

1. Entraste a la página oficial de Node.js.
2. Descargaste la versión **LTS** (Long Term Support), que aparece como "Recomendado para la mayoría de los usuarios".
3. Se bajó un instalador tipo:
   node-v24.11.1-x64.msi (la versión puede cambiar, pero el proceso es igual).

1.2. Instalar Node.js

1. Ejecutaste el .msi.
2. Aceptaste términos, diste **Next** a todo.
3. Dejaste las opciones por defecto.
4. Terminó la instalación y Node quedó instalado junto con **npm** (el gestor de paquetes de Node).

---

2. Verificar instalación de Node.js

Ya dentro de **VS Code**:

1. Abriste una **terminal**:
   - Menú superior → Terminal → New Terminal.
2. En la terminal (PowerShell al principio) ejecutaste:
3. node -v

Resultado:
v24.11.1

✅ Esto confirmó que **Node.js estaba correctamente instalado**.

4. Luego probaste:
5. npm -v

Y aquí apareció el problema.

---

3. Error con npm en PowerShell (y cómo lo arreglamos)

El error fue algo así:
npm : File C:\Program Files\nodejs\npm.ps1 cannot be loaded because running scripts is disabled on this system...
Esto significa que **PowerShell bloquea la ejecución de scripts**, y npm en Windows se ejecuta como un script (npm.ps1).

3.1. Qué significa el error

- PowerShell tiene una política llamada **ExecutionPolicy**.

- Cuando está muy restringida, **no permite ejecutar scripts locales**, incluso si son legítimos (como npm).

3.2. Solución que usamos (rápida y segura)

En lugar de modificar políticas de PowerShell, hicimos algo más práctico:

1. Cerraste la terminal de PowerShell.
2. Abriste una nueva terminal en VS Code, pero con tipo:
    - **Command Prompt / CMD** en lugar de PowerShell.
    - Se cambia desde el menú desplegable en la esquina superior derecha de la terminal.
3. En esa terminal CMD ejecutaste:
4. npm -v

Y ahora sí te dio un número (versión de npm).

✅ Esto confirmó que **npm funciona perfecto** usando CMD.

Nota: En el futuro, si quieres usar PowerShell, se puede ejecutar:
Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned
y aceptar con Y. Pero por ahora, **CMD es suficiente y más simple**.

---

4. Instalación y uso básico de Visual Studio Code

4.1. Instalar VS Code

1. Entraste a la página oficial de Visual Studio Code.
2. Descargaste el instalador para Windows.
3. Lo ejecutaste y dejaste las opciones por defecto.
4. Se instaló y lo abriste, viendo la pantalla de bienvenida (“Get started with VS Code”).

4.2. Cerrar la bienvenida y preparar entorno

1. Cerraste la pestaña “Welcome” (o pulsaste “Go Back”).
2. Creaste una carpeta para trabajar, por ejemplo:
3. C:\Users\Angel\Desktop\proyectos node js
4. En VS Code:
    - File → Open Folder…
    - Seleccionaste la carpeta proyectos node js.
5. Abriste la terminal CMD dentro de esa carpeta:
    - Terminal → New Terminal
    - Verificaste que dijera algo como:
    - PS C:\Users\Angel\Desktop\proyectos node js>

(o en CMD con la misma ruta).

---

5. Crear el proyecto Node.js

5.1. Inicializar npm

Dentro de la carpeta del proyecto y usando la terminal CMD:
npm init -y
Esto hizo:

- Crear el archivo **package.json**, que guarda:
  - Nombre del proyecto
  - Versión
  - Dependencias
  - Scripts (comandos) y metadatos

El -y acepta todas las opciones por defecto automáticamente.

---

6. Instalar Express

Express es un framework para crear servidores y APIs de forma sencilla.
En la misma terminal ejecutaste:
npm install express
Esto hizo:

- Descargar la librería express desde el registro de npm.
- Crear la carpeta **node_modules**.
- Agregar express a la sección dependencies del package.json.
- Crear un archivo package-lock.json para asegurar versiones reproducibles.

---

7. Crear tu primer servidor en Node.js con Express

7.1. Crear archivo principal

En el panel izquierdo de VS Code:

1. Clic derecho sobre la carpeta del proyecto → **New File**
2. Le pusiste de nombre:

index.js

7.2. Código del servidor

Dentro de index.js escribiste (o vas a escribir) algo como:

```
const express = require('express');
const app = express();
const port = 3000;

// Middleware para usar JSON
app.use(express.json());

// Ruta principal
app.get('/', (req, res) => {
  res.send('Servidor Node.js funcionando correctamente 😎');
```

```
});

// Ruta de prueba tipo API
app.get('/api/status', (req, res) => {
  res.json({
    ok: true,
    message: 'API lista y corriendo',
    hora: new Date().toISOString()
  });
});

// Levantar el servidor
app.listen(port, () => {
  console.log(`Servidor escuchando en http://localhost:${port}`);
});
```

Este archivo hace varias cosas:

- **Importa Express**
- Crea la app app = express()
- Define el puerto 3000
- Agrega un middleware para entender JSON
- Crea rutas:
  - GET / devuelve un texto.
  - GET /api/status devuelve un JSON con información de estado.
- Arranca el servidor escuchando en http://localhost:3000.

---

8. Ejecutar y probar el servidor

En la terminal CMD, dentro de la carpeta del proyecto:

node index.js

Si todo está bien, ves en la terminal:

Servidor escuchando en http://localhost:3000

Luego pruebas en el navegador:

1. http://localhost:3000/
   → Debe mostrar el texto:
   Servidor Node.js funcionando correctamente 😎
2. http://localhost:3000/api/status
   → Debe mostrar un JSON parecido a:
3. {
4. "ok": true,
5. "message": "API lista y corriendo",

6. "hora": "2025-12-02T..."
7. }

Con eso comprobaste que:

- Node.js está instalado correctamente.
- npm funciona (aunque en CMD).
- Express está instalado.
- Puedes levantar un servidor, definir rutas y responder peticiones.

---

9. ¿Dónde estás parado AHORA?

Ya lograste:

1. Instalar Node.js (v24.11.1) y npm.
2. Resolver un problema real de seguridad de PowerShell (npm.ps1 bloqueado) usando CMD.
3. Instalar y usar VS Code.
4. Crear un proyecto Node con npm init -y.
5. Instalar una dependencia (Express) con npm install express.
6. Crear y ejecutar tu servidor Node/Express que responde en rutas específicas.

---

10. Siguientes pasos posibles

Si quieres seguir avanzando, podemos:

1. **Convertir este servidor en una API más estructurada**, con carpetas como:
    - src/routes
    - src/controllers
    - src/middlewares
2. **Conectar una base de datos** (PostgreSQL, MySQL o MongoDB).
3. Hacer un **sistema de usuarios** (login, registro, JWT).
4. Montar un **frontend simple** (React/Next.js) que consuma tu API.
5. Empezar a diseñar **la plataforma de proyectos para Inymo** con flujos de:
    - Prospección
    - Cotización
    - Ejecución de proyecto
    - Entrega y cierre