Primeros pasos con Node.js

¿Qué es Node.js?

Node.js es un entorno de ejecución de JavaScript que te permite desarrollar aplicaciones del lado del servidor.

¿Por qué utilizar Node.js?

Node.js se ha convertido en una tecnología popular para el desarrollo de aplicaciones web modernas gracias a sus características únicas. Aquí te explico las principales razones para usarlo:

Ventajas de Node.js

1. **Arquitectura Basada en Eventos y No Bloqueante**
    - Node.js utiliza un modelo de entrada/salida no bloqueante basado en eventos, lo que permite manejar múltiples solicitudes simultáneamente sin esperar a que se completen las operaciones.
    - Ideal para aplicaciones en tiempo real (chats, notificaciones, streaming).
2. **Un Lenguaje para Todo el Stack (JavaScript)**
    - Permite usar el mismo lenguaje (JavaScript) tanto en el cliente (frontend) como en el servidor (backend), simplificando el desarrollo y facilitando la colaboración entre equipos.
3. **Rendimiento**
    - Basado en el motor V8 de Chrome, Node.js convierte el código JavaScript a código máquina, lo que mejora la velocidad de ejecución.
4. **Ecosistema Rico con npm**
    - Con más de un millón de paquetes disponibles, npm (Node Package Manager) proporciona herramientas y librerías que facilitan el desarrollo, desde frameworks como Express hasta soluciones para bases de datos y autenticación.
5. **Comunidad Activa**
    - Una gran comunidad de desarrolladores garantiza la constante mejora de Node.js y una amplia variedad de recursos (tutoriales, foros, plugins).
6. **Escalabilidad**
    - Node.js está diseñado para aplicaciones escalables gracias a su capacidad para manejar conexiones simultáneas con un uso eficiente de los recursos.

Partes de una Aplicación Web

Una aplicación web, sin importar el lenguaje o entorno utilizado, generalmente consta de las siguientes partes:

---

1. Cliente (Frontend)

- **Descripción**: Es la interfaz con la que interactúa el usuario.
- **Responsabilidades**:

    - Mostrar datos (HTML, CSS, JavaScript).
    - Enviar solicitudes al servidor (a través de APIs o formularios).
    - Recibir y procesar respuestas del servidor.
- **Tecnologías comunes**:
    - HTML, CSS, JavaScript.
    - Frameworks/librerías como React, Angular o Vue.js.

---

2. Servidor (Backend)

- **Descripción**: Es el "cerebro" de la aplicación que gestiona la lógica de la aplicación y procesa las solicitudes del cliente.
- **Responsabilidades**:
    - Procesar solicitudes HTTP (GET, POST, PUT, DELETE, etc.).
    - Interactuar con bases de datos.
    - Gestionar autenticación, autorización y sesiones.
    - Servir archivos estáticos o generar contenido dinámico.
- **Tecnologías comunes**:
    - Node.js, Django, Flask, Laravel, etc.
    - En Node.js, frameworks como Express o NextJS facilitan la creación del backend.

---

3. Base de Datos

- **Descripción**: Almacena y gestiona la información que utiliza la aplicación.
- **Responsabilidades**:
    - Guardar datos estructurados o no estructurados (según el tipo de base de datos).
    - Permitir la recuperación, actualización y eliminación de información.
- **Tipos de bases de datos**:
    - Relacionales (SQL): MySQL, PostgreSQL, SQLite.
    - No relacionales (NoSQL): MongoDB, Firebase, Redis.

Ejemplo de Flujo

1. El cliente envía una solicitud (ejemplo: buscar productos).
2. El servidor procesa la solicitud y consulta la base de datos.
3. El servidor devuelve una respuesta en formato JSON (API).
4. El cliente procesa la respuesta y la muestra al usuario.

Primeros pasos con Node.js

1. Instalar Node.js

1. Ve a la página oficial de Node.js.
2. Descarga e instala la versión LTS (Long Term Support) para mayor estabilidad.

Verifica la instalación ejecutando estos comandos en tu terminal:

```
node -v
npm -v
```

3. Esto debería mostrar las versiones instaladas de Node.js y npm (Node Package Manager).

---

2. Crear un Proyecto

Crea una carpeta para tu proyecto:

```
mkdir mi-app-node
cd mi-app-node
```

1. Inicializa un proyecto Node.js:

   ```
   npm init -y
   ```
2. Esto generará un archivo `package.json` con la configuración básica del proyecto.

---

3. Instalar Dependencias (opcional)

Para esta guía, instalaremos un paquete llamado **Express**, que simplifica la creación de aplicaciones web:

```
npm install express
```

---

4. Crear el Archivo Principal

1. Crea un archivo llamado `app.js` (o el nombre que prefieras).

Abre el archivo y añade el siguiente código básico para un servidor HTTP con Express:

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

// Ruta básica
```

```
app.get('/', (req, res) => {
    res.send('¡Hola, mundo desde Node.js!');
});

// Inicia el servidor
const PORT = 3000;
app.listen(PORT, () => {
    console.log(`Servidor escuchando en http://localhost:${PORT}`);
});
```

2.

---

5. Ejecutar la Aplicación

1. En la terminal, ejecuta:
   `node app.js`