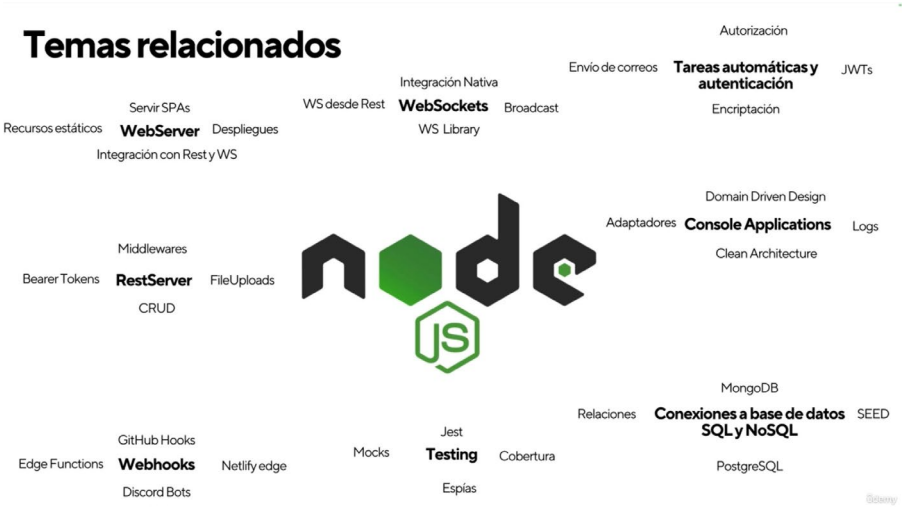

GUÍA DE TRABAJO NODE JS

Desarrollo de aplicaciones back-end

NODE JS:

Node.js® es un entorno de ejecución de JavaScript gratuito, de código abierto y multiplataforma que permite a los desarrolladores crear servidores, aplicaciones web, herramientas de línea de comandos y scripts.

Victor Vladimir Cortes Arevalo
Instructor SENA CEET

GUIA DE TRABAJO EN NODE JS

Para crear e inicializar un proyecto en Node.js, sigue estos pasos utilizando los comandos en tu terminal:

1. **Crear una carpeta para tu proyecto:**

```
mkdir nombre-del-proyecto
cd nombre-del-proyecto
```

2. **Inicializar el proyecto con un archivo package.json:** Este archivo contendrá información básica sobre tu proyecto, como el nombre, versión, dependencias, entre otros.

   Puedes hacerlo de dos maneras:

   - Con preguntas interactivas:

```
npm init
```

   - **Con valores predeterminados:** Si deseas crear el package.json rápidamente con valores por defecto, puedes usar la opción -y:

```
npm init -y
```

3. **Instalar dependencias iniciales (opcional):** Si ya sabes qué dependencias quieres usar, puedes instalarlas desde el principio. Por ejemplo:

   - Instalar express:

```
npm install express
```

Instalar **nodemon** (para reiniciar el servidor automáticamente al detectar cambios en los archivos):

```
npm install --save-dev nodemon
```

4. **Crear un archivo de entrada (por ejemplo, index.js):** Este archivo será el punto de inicio de tu aplicación.

Alternativas en Windows:

1. **Usar el comando type para crear un archivo vacío:** En el CMD de Windows, puedes crear un archivo vacío usando el comando type:

```
type nul > index.js
```

2. **Usar el comando echo para crear un archivo vacío:** Puedes usar el comando echo para crear un archivo vacío de esta manera:

```
echo. > index.js
```

3. **Usar PowerShell:** Si prefieres usar PowerShell, que está incluido en Windows 11, puedes crear un archivo vacío con el siguiente comando:

```
New-Item -Path . -Name "index.js" -ItemType "File"
```

5. **Abrir el archivo con Visual Studio Code:**

Si tienes Visual Studio Code instalado y agregado a tu PATH, puedes abrir el archivo con este editor utilizando:

```
code index.js
```

Otras opciones:

WebStorm index.js

subl index.js

atom index.js

6. **Configurar el script de inicio en package.json:** Abre el archivo package.json y asegúrate de tener un script de inicio configurado. Por ejemplo, si usas nodemon para desarrollo:

```
"scripts": {
  "start": "node index.js",
  "dev": "nodemon index.js"
}
```

Con esto, puedes iniciar tu aplicación con:

```
npm start
```

Y en modo desarrollo (con reinicio automático):

```
npm run dev
```

Ahora tu proyecto Node.js está creado e inicializado y puedes comenzar a desarrollar tu aplicación.

ANEXO 1 – EXPRESS

Express es un framework web minimalista y flexible para Node.js que facilita la creación de aplicaciones web y API. Fue diseñado para simplificar y acelerar el desarrollo de aplicaciones web robustas y escalables, proporcionando un conjunto de características y herramientas para manejar solicitudes HTTP, enrutamiento, middleware, y mucho más.

Características principales de Express:

1. **Enrutamiento:** Express permite definir rutas para diferentes solicitudes HTTP (GET, POST, PUT, DELETE, etc.). Puedes crear rutas simples para manejar solicitudes específicas o rutas más complejas que aceptan parámetros.
2. **Middleware:** El middleware en Express son funciones que tienen acceso al objeto de solicitud (req), el objeto de respuesta (res), y una función que indica cuándo pasar al siguiente middleware. Puedes usar middleware para diversas tareas, como autenticar usuarios, manejar errores, y más.
3. **Manejo de solicitudes y respuestas:** Express facilita el procesamiento de solicitudes HTTP y la gestión de respuestas. Puedes acceder fácilmente a los datos enviados en las solicitudes, configurar cabeceras de respuesta, enviar respuestas JSON, archivos, etc.
4. **Renderizado de vistas:** Aunque Express no obliga a usar un motor de plantillas específico, puedes integrarlo con motores de plantillas como EJS, Pug, Handlebars, entre otros, para renderizar páginas HTML dinámicas.
5. **Extensibilidad:** Express tiene una arquitectura basada en middleware que permite añadir fácilmente nuevas funcionalidades a tu aplicación. Además, hay una gran cantidad de middleware y complementos disponibles para extender las capacidades de Express.

Ejemplo básico de una aplicación con Express

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

// Ruta principal que responde con un mensaje simple
app.get('/', (req, res) => {
  res.send('¡Hola, mundo!');
});

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

¿Para qué sirve Express?

Express es útil para una variedad de aplicaciones, incluyendo:

- **Aplicaciones web:** Puedes crear sitios web y aplicaciones web con funcionalidades completas.
- **API RESTful:** Express es muy popular para desarrollar APIs RESTful que proporcionan datos y servicios a clientes como aplicaciones móviles o aplicaciones web.
- **Aplicaciones de una sola página (SPA):** Puedes servir archivos estáticos, manejar enrutamiento, y gestionar otras tareas necesarias para aplicaciones SPA.

En resumen, Express es una herramienta poderosa y versátil para el desarrollo web en Node.js, que te permite enfocarte en la lógica de tu aplicación sin preocuparte por las complejidades de manejar servidores HTTP y enrutamiento manualmente.

ANEXO 2 – NODEMON

Nodemon es una herramienta que facilita el desarrollo de aplicaciones Node.js al monitorear automáticamente los cambios en los archivos del proyecto y reiniciar el servidor cuando se detectan modificaciones. Esto elimina la necesidad de detener y reiniciar manualmente el servidor cada vez que realizas un cambio en tu código, lo que mejora significativamente la eficiencia del desarrollo.

Características principales de Nodemon:

1. **Monitoreo de archivos:** Nodemon supervisa todos los archivos en tu proyecto, y cuando detecta cambios en archivos con extensiones específicas (por defecto .js, .json, .mjs, .coffee, entre otros), reinicia automáticamente la aplicación.
2. **Reinicio automático:** Cada vez que haces un cambio en tu código y guardas el archivo, Nodemon reinicia el servidor de Node.js para que puedas ver los cambios inmediatamente sin necesidad de intervención manual.
3. **Compatibilidad con configuraciones personalizadas:** Puedes configurar Nodemon para que supervise archivos con extensiones específicas, ignorar ciertos archivos o directorios, y establecer diferentes comportamientos según tus necesidades de desarrollo.
4. **Facilidad de uso:** Nodemon es fácil de instalar y usar. Solo necesitas reemplazar node con nodemon en tu línea de comandos, y Nodemon se encargará del resto.
5. **Integración con scripts de npm:** Puedes integrar Nodemon en scripts de npm, lo que te permite iniciar tu aplicación en modo de desarrollo con comandos personalizados.

Instalación y uso:

1. **Instalación global (recomendada):** Puedes instalar Nodemon globalmente en tu sistema para que esté disponible en cualquier proyecto:

   bash

   npm install -g nodemon

2. **Instalación como dependencia de desarrollo:** Si prefieres instalar Nodemon solo para un proyecto específico, puedes hacerlo como una dependencia de desarrollo:

   npm install --save-dev nodemon

3. **Uso básico:** Una vez instalado, puedes usar Nodemon en lugar de node para iniciar tu aplicación. Por ejemplo, si normalmente inicias tu aplicación con:

   node app.js

   Ahora puedes usar Nodemon:

   nodemon app.js

4. **Uso con scripts de npm:** Puedes definir un script en tu package.json para usar Nodemon:

```
"scripts": {
  "start": "node app.js",
  "dev": "nodemon app.js"
```

}

Luego puedes iniciar tu aplicación en modo de desarrollo con:

npm run dev

ANEXO 3 – LIVE SERVER

Live Server es una extensión popular para editores de código como Visual Studio Code que permite lanzar un servidor local de desarrollo con una funcionalidad de "live reload" o recarga en vivo. Esto significa que cuando editas y guardas tus archivos HTML, CSS, o JavaScript, la página web se recarga automáticamente en tu navegador para reflejar los cambios realizados.

Características principales de Live Server:

1. **Recarga automática (Live Reload):** La característica más destacada de Live Server es la capacidad de recargar automáticamente la página web en el navegador cada vez que guardas cambios en tus archivos. Esto mejora significativamente la eficiencia del desarrollo web, ya que no necesitas recargar manualmente el navegador cada vez que haces un cambio.
2. **Servidor local:** Live Server crea un servidor HTTP local, lo que te permite ver tu proyecto web en un entorno similar a como se serviría en un servidor real. Esto es útil para probar características que requieren un servidor, como la carga de recursos externos o la simulación de rutas.
3. **Soporte para múltiples formatos:** Aunque es principalmente utilizado para archivos HTML, Live Server también admite la recarga en vivo para CSS y JavaScript, lo que lo hace adecuado para proyectos front-end.
4. **Configuración sencilla:** Live Server se puede configurar fácilmente y permite opciones personalizadas como la elección del puerto, la carpeta raíz, la configuración de proxy, entre otras.
5. **Compatible con otros frameworks:** Si estás trabajando con frameworks o bibliotecas front-end como Angular, React, o Vue.js, puedes usar Live Server para una experiencia de desarrollo más rápida y eficiente.

ANEXO 4 – NODE JS