Qué es Node.JS

Es un entorno de ejecución de JavaScript que permite a los desarrolladores ejecutar código JavaScript en el servidor, fuera del navegador. Es conocido por su alta eficiencia y capacidad para manejar aplicaciones en tiempo real y de alta concurrencia.
Para utilizar tu propio servidor, lo normal, se trae por un get una web, pero esto se puede modificar colocando un PORT.

```
const http = require("http");

const server = http.createServer((req, res) => {
res.writeHead(200, {"content-Type":"text/plain"});
res.end('<h1>Hola Node JS</h1><p>Daniela Berrios...</p>')

})

const PORT:3000;
 server.listen(PORT, () =>
console.log(`http://localhost:${PORT}`));
```

buscar el ejemplo completo en clase 11 index.js, cuando se hace un cambio hay que cerrar y volver a levantar el servidor, tener cuidado con el content-Type para cambiar los formatos de contenido.

Tipos de Módulos en Node js

- Los predeterminados o nativos (del propio JS o Node)
- Los externos (librerías)
- Los internos (los que creamos nosotros mismos)

rutas

- Rutas Absolutas, normalmente se utilizaran mas estas rutas en Node JS.
- Rutas (cortitas), se utilizan poco y se emplean colocando ./carpeta-Usar

Solicitudes HTTP (CRUD)

Las solicitudes HTTP son métodos que permiten a los clientes (como navegadores web o aplicaciones) comunicarse con los servidores para realizar diversas operaciones.

Métodos HTTP:

- GET: Recupera información del servidor. Es el método más común y se utiliza para solicitar datos sin modificar el estado del servidor. Básicamente busca y trae información

```
app.get('/ruta', (req, res) => {
  res.send('Datos recuperados');
});
```

- POST: Envía datos al servidor para crear o actualizar un recurso. Es comúnmente utilizado para enviar formularios o datos a la base de datos.

```
app.post('/ruta', (req, res) => {
  // Procesa los datos enviados
  res.send('Datos enviados');
});
```

- PUT: Actualiza un recurso existente en el servidor. A diferencia de POST, PUT es idempotente, lo que significa que realizar la misma solicitud varias veces producirá el mismo resultado.

```
app.put('/ruta/:id', (req, res) => {
  // Actualiza el recurso con el ID especificado
  res.send('Recurso actualizado');
});
```

- DELETE: Elimina un recurso del servidor.

```
app.delete('/ruta/:id', (req, res) => {
  // Elimina el recurso con el ID especificado
  res.send('Recurso eliminado');
});
```

- Listen: No es un método HTTP, sino una función de Express.js que inicia el servidor y lo pone a la escucha de solicitudes en un puerto específico.

```
app.listen(3000, () => {
  console.log('Servidor escuchando en el puerto 3000');
});
```

- path: La ruta o endpoint en la URL que define dónde se debe dirigir la solicitud. Por ejemplo, en app.get('/ruta', ...), /ruta es el path.

NODE Express

Framework

Un framework es un conjunto de herramientas y bibliotecas que proporciona una estructura base para desarrollar software de manera más rápida y eficiente. En lugar de escribir todo el código desde cero, los desarrolladores pueden utilizar un framework para manejar tareas comunes y repetitivas, como la gestión de bases de datos, la seguridad, la autenticación, y la manipulación de datos.

Ventajas de usar un framework:

- Ahorro de tiempo: Permite reutilizar código y soluciones existentes, acelerando el proceso de desarrollo.
- Consistencia: Establece estándares de codificación que aseguran que el código sea coherente y fácil de mantener.
- Colaboración: Facilita el trabajo en equipo al proporcionar una estructura común y bien definida.
- Seguridad: Muchos frameworks incluyen actualizaciones y mejoras continuas en seguridad.

Ejemplos de frameworks populares:

- Django: Para desarrollo web en Python.
- Ruby on Rails: Para desarrollo web en Ruby.
- React: Para construir interfaces de usuario en JavaScript.
- Spring: Para aplicaciones empresariales en Java

Que es Node Express

Es un framework Unopinionated, lo que quiere decir que no nos dice como tenemos que crear la estructura de nuestro proyecto, ni como escribir el código, solo nos da funcionalidades ya hechas, con las cuales nosotros debemos que crear la estructura y decidir como utilizar esas funcionalidades, facilitando así el proceso de desarrollo. Express.js es un framework minimalista y flexible para Node.js que facilita la creación de aplicaciones web y APIs. Proporciona un conjunto robusto de características para manejar peticiones y respuestas HTTP, enrutamiento, middleware, y mucho más.

Características de Express.js:

- Enrutamiento: Define rutas para manejar diferentes tipos de solicitudes HTTP (GET, POST, PUT, DELETE).
- Middleware: Funciones que se ejecutan antes de llegar a la ruta final, permitiendo manipular las solicitudes y respuestas.
- Gestión de peticiones y respuestas: Simplifica la creación de servidores web y la gestión de datos enviados y recibidos

Ejemplo de código js empleando Node express:

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

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

```
app.listen(PORT, () => {
  console.log(`Servidor Express escuchando en el puerto
${PORT}`);
});
```

Este es un ejemplo básico de servidor que accede a la ruta raíz.