Node

Node.js es un entorno de ejecución de programas **JavaScript** del lado del servidor. A través de Node.js podemos construir **aplicaciones Web** utilizando JavaScript aprovechando la experiencia, a nivel mundial, de miles de desarrolladores en este lenguaje de programación, pero ahora en entornos Back End.

Comenzamos nuestro viaje explorando los conceptos, principios y herramientas de las **tecnologías Back End**, para luego profundizar en los detalles de Node.js. Seguidamente avanzamos estudiando algunas herramientas que se emplean en la preparación del ambiente de programación, lo cual es vital para poder ejecutar nuestras tareas de programación. Adicionalmente, exploramos la depuración de nuestros códigos en Node.js. y terminamos esta parte del recorrido creando una aplicación Web usando **Express**.

A continuación, revisamos una serie de conceptos relacionados con **Node Express** que son necesarios para entender cómo funciona una aplicación construida usando esta herramienta. Durante esta parte, empleamos herramientas, tales como, las **plantillas, middlewares, parámetros en el URL y sesiones Flash**, en el desarrollo de aplicaciones Web.

En la siguiente estación en nuestro recorrido, interactuamos con una **base de datos MySQL** para realizar las operaciones propias de la misma, tales como, actualizar, eliminar, agregar y borrar a través de los **queries SQL**. Construimos así una aplicación Web que obtiene datos de una base de datos, para insertarlos en una **plantilla HTML** y que se dibujen finalmente en el navegador. En esta sección también estudiamos **REST**, como una tecnología que proporciona una arquitectura para interactuar con aplicaciones Web, a través del uso de **APIs** (Interfaz de Programación de Aplicaciones o Application Programming Interface, en inglés). Durante esta parte profundizamos en el uso de los **métodos HTTP** y el formato de intercambio de datos **JSON**.

Nuestra última parada en nuestro camino a través de Node.js fue el desarrollo de una serie de requerimientos, para construir códigos que ahora forman parte de nuestro **portafolio** de desarrollos en Node.js.

- Aplicar los fundamentos y principios de Node.js en la construcción de aplicaciones Web.
- Usar las herramientas de Node.js para la preparación del ambiente de programación.
- Desarrollar APIs REST para el intercambio y administración de datos en Internet desde nuestra aplicación Web.
- Desarrollar aplicaciones Web empleando Node Express. Desarrollar consultas a una base de datos MySQL usando Node Express.

Ahora estás listo para desarrollar tus propias soluciones Node.js en ambientes Back End. Pero primero, te proporcionamos un resumen de lo que hemos estudiado sobre este entorno de programación.

UNIDAD 1: Node Back End

Una **aplicación Web** se puede definir como una herramienta que se accede desde el cliente, el cual realiza peticiones a un servidor, donde el cliente es usualmente un navegador. En el marco de esta arquitectura se encuentran las **tecnologías del lado del cliente o tecnologías Front End**, que son el conjunto de herramientas que el navegador entiende, tal como un recurso HTML. En este mismo contexto podemos definir los **contenidos estáticos**, donde los recursos que permiten generar dichos contenidos se crean antes de ser almacenados en el servidor, quien los despacha a medida que son solicitados por los clientes.

Por otra parte tenemos las **tecnologías del lado del servidor o Back End** que se ejecutan en el servidor. Y ligadas a estas tecnologías están los **contenidos dinámicos**, donde los recursos que soportan dichos contenidos se crean cuando el cliente los solicita. El servidor maneja lenguajes, tales como PHP, Node, Ruby, Python, Java, C++, Perl para crear scripts que generan estos contenidos.

Otro modelo es el denominado **Single Page Applications (SAP)**, el cual soluciona el problema del refrescamiento constante de todo el contenido cada vez que se hace una

transición de una página a otra. En el esquema SAP, solo se actualizan trozos de las páginas usando las **nuevas técnicas de JavaScript**, tal como **Ajax**.

Node.js es un entorno de ejecución de **JavaScript multiplataforma** considerado también como un interpretador que se sustenta en **ECMAScript 6 y posteriores**. El mismo facilita las tareas de instalación y desarrollo de los programas, a través de un conjunto de herramientas. Una de sus características es el trabajar con estructuras independientes que se incorporan a los programas según se necesiten, llamados **módulos o paquetes**. NPM es el manejador de paquetes de Node.js; donde muchos de estos paquetes están disponibles en el sitio de npmjs. Sin embargo, existe una alternativa a NPM, denominada **Yarn** que es mucho más rápida a la hora de gestionar los módulos.

Similarmente a otros lenguajes de programación, Node cuenta con un **depurador** que permite ejecutar un programa paso a paso, y así poder identificar y arreglar los errores de nuestro código.

Para construir una aplicación en Node.js necesitamos crear un proyecto usando el comando **npm init** que nos genera dos archivos, **package.json** que describe nuestro paquete y el **archivo de la aplicación**. Adicionalmente, con la finalidad de facilitar el proceso de construcción de la aplicación Web, empleamos **Express** que es un framework para construir aplicaciones que dispone de todas las funciones del servidor, tales como el manejo de cookies y de sesiones.

UNIDAD 2: Express

Existen ciertas nociones básicas que respaldan el desarrollo de **aplicaciones Web** usando **Node Express**. Una de ellas es la **vía de acceso de rutas**, el cual es un identificador usado por el sistema como referencia para determinar el recurso sobre el cual se ejecuta alguna operación. Por otro lado, tenemos las rutas que se definen como la forma en que una aplicación Web responde a una petición del cliente para un punto final o recurso particular, que se identifica a través de una vía de acceso de ruta. En la estructura básica

de una ruta están los **métodos HTTP**. Un **manejador de ruta**, por su parte, es la función que se ejecuta cuando la ruta coincide.

En Express también encontramos los **middlewares**. Un middleware es un conjunto de pasos por los que pasa una petición con la finalidad de ejecutar un código, invocar al próximo middleware usando la función siguiente, finalizar una cadena de solicitudes y respuestas, o realizar cambios en la solicitud. Igualmente, tenemos las **plantillas** o templates que permiten describir el comportamiento de una aplicación de forma modular, facilitando la evaluación de las partes dinámicas de nuestra aplicación Web. A través del uso de un motor como el **ejs** podemos emplear plantillas en nuestro programa.

Existen tres formas de **pasar parámetros** entre nuestra aplicación Web y el cliente. La primera es a través del uso de **query strings** que constituye una manera de diferenciar parámetros de la identificación de un recurso en el URL. La segunda es una forma más elegante y se denomina **params**, que se utiliza para recursos que sean identificables, tales como una persona o un producto. La tercera forma consiste en capturar valores ingresados en un formulario, tales como el login y la contraseña. Una vez capturados dichos valores, es posible usarlos para conectarnos a una base de datos o para realizar alguna otra operación que lo requiera a través de la librería **body-parser**.

Otro concepto importante en Express es el de **sesión**, que almacena información que será usada en la aplicación Web durante la visita de un usuario a varias páginas en un momento dado. La sesión se puede usar para muchas cosas, una de ellas es el **mecanismo de Flash o variables** Flash. Las variables Flash se utilizan una sola vez, por ejemplo, para desplegar un valor en el navegador. El mismo se destruirá una vez que refresquemos el browser.

Para conectarnos a una base de datos desde nuestra aplicación Express necesitamos instalar el **conector** que nos permite interactuar con la misma. Seguidamente necesitamos **crear una conexión** donde usamos los parámetros que permiten indicar ambos lados de dicha conexión, incluyendo el servidor donde está instalada la base de datos. Una forma más eficiente de conectarse con el servidor de la base de datos es utilizando un **pool de**

conexiones, el cual permite compartir un conjunto de conexiones entre varios clientes hasta un número máximo. Esto limita el número de clientes conectados concurrentemente a la base de datos resultando en un mejor uso de los recursos.

UNIDAD 3: Creando aplicaciones Back End

Cuando trabajamos con una **base de datos MySQL** desde nuestra aplicación Node Express debemos instalar la **librería de MySQL**, para luego inicializarla y configurarla. Luego configuramos la conexión usando los siguientes parámetros: **dirección del servidor**, donde está instalada la base de datos, el usuario con el cual nos vamos a conectar, **la clave del usuario y la base** de datos que utilizaremos. Podemos también elegir emplear un **pool de conexiones** para usar más eficientemente los recursos del servidor. Una vez configurada la conexión nos conectamos a la base de datos para enviar uno o más queries. Los resultados de las consultas son empleados por la aplicación para, por ejemplo, realizar una operación sobre ellos o enviarlos al cliente vía la plantilla HTML. A través de los queries se pueden crear, actualizar o eliminar registros de la base de datos.

REST es una arquitectura que usa **recursos**, tales como las noticias y los usuarios en un sitio de Noticias, y se emplea a través de un **API REST** que permite que nuestra aplicación interactúe con otras aplicaciones a través de mecanismos y formatos bien conocidos. Cuando se emplea REST, el cliente envía peticiones a través de los **métodos HTTP**, tales como **GET, POST, PUT y DELETE**. Las peticiones están dirigidas a un recurso específico que se define en el URL. El **resultado** de la petición, que viaja desde el servidor al cliente, usualmente viene representado en un **formato JSON**. El **API** específica entonces las operaciones del servidor de acuerdo con las peticiones siguiendo las reglas establecidas por REST.

Los **métodos HTTP** se emplean de la siguiente forma. El método **GET** permite obtener recursos, mientras que el **POST** se utiliza para crear recursos y el **PUT** para actualizar recursos. Finalmente, el **DELETE** elimina recursos.

Actualmente, todos los grandes sitios Web tienen **APIs REST** para ofrecer datos a aplicaciones móviles, ofrecer datos a programadores en un formato ampliamente aceptado, ofrecer datos a nuestra propia aplicación y obtener datos de otras aplicaciones. Sitios Web como Amazon, Youtube, Dropbox y Twitter tienen sus propias **APIs**.

UNIDAD 4: Caso de Estudio: Blog de Viajes

El desarrollo de una aplicación Web se puede tornar una tarea difícil si no tomamos en cuenta los pasos recomendados para la construcción de un software. Durante la última parte de nuestro recorrido por Node.js, te fuimos guiando a través del proceso de construcción de tu aplicación. A continuación, te enumeramos los pasos que puedes usar en tus futuros desarrollos. Debido a que el proceso de desarrollo de software, hoy en día, no es lineal sino más bien iterativo, las tareas relacionadas a los siguientes pasos las podrás ejecutar más de una vez durante el ciclo de desarrollo.

1. Definición de los requerimientos.
2. Análisis de la solución en función de los requerimientos, la cual comprende:
    a. Diseño de la base de datos
    b. Diseño de la interfaz (Bocetos)
    c. Diseño de las rutas de Express
3. Implementación de la solución:
    a. Creación de la base de datos
    b. Creación de los datos de prueba e incorporación de los mismos en la base de datos
    c. Creación de la carpeta base (crear el proyecto en Express) y agregar las librerías comunes (mysql, express-session, express-flash, body-parser, ejs, configuración base) y el archivo del programa principal
    d. Creación de la interfaz de usuario general de la aplicación usando CSS, HTML, JavaScript y considerando hacer plantillas parciales para reusar el código.
    e. Implementación de cada ruta de Express
    f. Implementación de los middlewares en caso de que se requieran
    g. Depuración del código en cada paso anterior de acuerdo a las necesidades del programador
4. Realización de las pruebas del programa:

a. Ejecución de las pruebas funcionales del sistema para validar que arroje los resultados esperados.