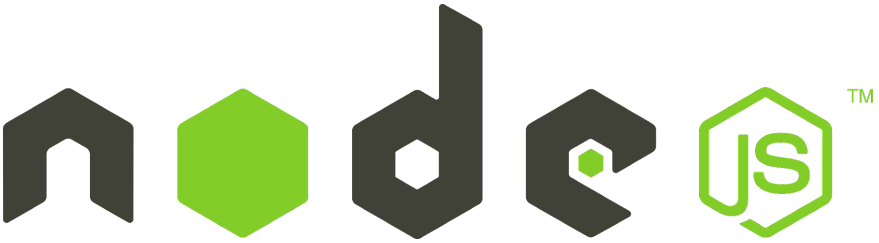
DESARROLLO DE SERVICIOS WEB CON NODEJS



Email: *julietalanciotti@gmail.com*

Módulo 1 - Introducción

JavaScript

Según Mozilla (MDN 2020), **JavaScript**, o simplemente JS, es un lenguaje de scripting multiplataforma y orientado a objetos. Se trata de un lenguaje de programación tipo script, basado en objetos y guiado por eventos, diseñado específicamente para el desarrollo de aplicaciones cliente-servidor dentro del ámbito de Internet. Los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos.

Está estandarizado en Ecma International (asociación europea para la creación de estándares para la comunicación y la información) con el fin de ofrecer un lenguaje de programación estandarizado e internacional basado en Javascript y llamado ECMAScript para que todos los navegadores puedan realizar un intérprete que acepte la misma sintaxis.

ECMAScript 6

ECMAScript v6 (Abreviado como ES6 o ES2015) es el estándar que sigue JavaScript desde junio de 2015. Hasta ese momento la versión de JS que estábamos usando en nuestros navegadores y Node.js, era la v5.

- Actualizaciones ECMAScript 6
    - *Función Arrow:* definir funciones mucho más claras y limpias a nivel sintaxis.

- *Clases:* parecidas a los constructores o funciones utilizadas anteriormente, pero esta vez bajo el paradigma de programación orientada a objetos. Incluye herencia. (No deja de ser Sugar Syntax ya que en JS sólo existen prototipos).
- *let:* declara una variable local en un bloque de ámbito (scope), inicializándola opcionalmente a un valor.
- *const:* declara una constante de sólo lectura en un bloque de ámbito. Una constante funciona como una variable pero no cambia su valor, es decir, representa un lugar de almacenamiento de tipos de datos en la memoria pero una vez asignado el valor inicial este no puede ser modificado. La sintaxis de la definición del identificador es la misma que la de las variables.

ℹ️ **Más información:** http://es6-features.org

Node JS

Durante sus primeros 20 años, JavaScript se utilizó principalmente para la creación de scripts del lado del cliente. Dado que sólo podía utilizarse dentro de la etiqueta *<script>*, los desarrolladores tenían que trabajar en múltiples lenguajes y frameworks entre los componentes del front-end y del back-end. Por lo tanto surgió **NodeJS** como una solución a esto.

Es un *entorno de ejecución* de un solo hilo, de código abierto y multiplataforma para crear aplicaciones de red y del lado del servidor rápidas y escalables. Se ejecuta en el

motor de ejecución de JavaScript V8, y utiliza una arquitectura de E/S basada en eventos y sin bloqueos, lo que la hace eficiente y adecuada para aplicaciones en tiempo real.

ℹ️ **Más información: https://nodejs.org/es/about/**

Para instalar NodeJS en la computadora, te recomiendo mirar el siguiente video: https://youtu.be/CqjXtmUyww4

NPM

npm es el Manejador de Paquetes de Node.js (Node Package Manager) que viene incluido y ayuda a cada desarrollo asociado a Node. Es ampliamente utilizado por los desarrolladores de JavaScript para compartir herramientas, instalar módulos y administrar dependencias.

Aunque crea parte de la estructura/organización de su directorio, este no es el propósito principal.

El objetivo principal, es la dependencia automatizada y la administración de paquetes. Esto significa que puede especificar todas las dependencias de su proyecto dentro de su archivo *package.json*. Luego, en cualquier momento que usted (o cualquier otra persona) necesite comenzar con su proyecto, pueden ejecutar `npm`

`install` para tener inmediatamente todas las dependencias instaladas. Además de esto, también es posible especificar de qué versiones depende su proyecto para evitar que las actualizaciones lo rompan.

Como se mencionó anteriormente, npm ayuda a la gestión de paquetes en el proyecto. Definitivamente es posible descargar manualmente sus bibliotecas, copiarlas en los directorios correctos y usarlas de esa manera. Sin embargo, a medida que su proyecto (y la lista de dependencias) crezca, esto rápidamente se volverá lento y complicado, también hará que colaborar y compartir su proyecto sea mucho más difícil.

Protocolo HTTP

Hypertext Transfer Protocol (HTTP) (*Protocolo de Transferencia de Hipertexto*) es un protocolo utilizado para la transmisión de documentos hipermedia, como HTML. Fue diseñado para la comunicación entre los navegadores y servidores web, aunque puede ser utilizado para otros propósitos también.

HTTP define un conjunto de métodos de petición para indicar la acción que desea realizar para un recurso determinado.
Está orientado a transacciones y sigue el esquema de petición/respuesta entre un cliente y un servidor.

El cliente realiza una petición enviando un mensaje con cierto formato al servidor. El servidor le envía un mensaje de respuesta. Para hacerlo más concreto, un cliente podría ser un navegador web y un servidor podría ser una aplicación en un servidor web corriendo en Internet.

Para las solicitudes se comienza con una acción requerida por el servidor, a esto se le denomina método de petición, seguido de la URL del recurso y la versión http que soporte al cliente. Lo importante es el método de petición y la URL. También los mensajes tienen una cabecera que son metadatos con información diversa.

Método GET

El método GET es por excelencia utilizado en el desarrollo de aplicaciones web para enviar una solicitud de *"obtención de información"* a un servidor desde un cliente. Su forma más común de uso es mediante formularios HTML, donde a partir de una serie de datos que se brindan, por ejemplo, un id, un DNI, un nombre, etc, se pretende lograr como respuesta una serie de registros o datos.

> **¿Cuándo es recomendado utilizar el método GET?**
>
> Cuando estamos desarrollando una aplicación y el cliente necesita obtener información por parte del servidor. GET nos puede servir para enviar parámetros, como ser filtros para una tabla, formas de ordenación (ascendente, descendente, etc), parámetros de búsqueda, entre muchos otros.

Método POST

El método post es utilizado en el protocolo para poder proporcionar información al servidor desde un cliente. A diferencia del método GET, en POST la información es enviada a través de las cabeceras de los paquetes o, así también, dentro del cuerpo del mensaje, haciendo de esta manera que la información sea invisible para el

usuario. Es sin dudas un método recomendado para manejar gran cantidad de datos, como por ejemplo un formulario que sea enviado desde un cliente, como así también, para datos que necesiten mayor seguridad, como el caso de las contraseñas.

Otros métodos:

MÉTODO	FUNCIONALIDAD
GET	El método **GET** solicita una representación de un recurso específico. Las peticiones que usan el método GET sólo deben recuperar datos.
POST	El método **POST** se utiliza para enviar una entidad a un recurso en específico, causando a menudo un cambio en el estado o efectos secundarios en el servidor.
HEAD	El método **HEAD** pide una respuesta idéntica a la de una petición GET, pero sin el cuerpo de la respuesta.
PUT	El modo **PUT** reemplaza todas las representaciones actuales del recurso de destino con la carga útil de la petición.
DELETE	El método **DELETE** borra un recurso en específico.
PATCH	El método **PATCH** es utilizado para aplicar modificaciones parciales a un recurso.

Arquitectura de Servicios

La Arquitectura Orientada a Servicios (**SOA**) es un tipo de arquitectura de IT que se apoya en la orientación a servicios. La *orientación a servicios* es una forma de pensar en servicios, su construcción y sus resultados.

Un servicio es una representación lógica de una actividad de negocio que tiene un resultado de negocio específico (ejemplo: comprobar el crédito de un cliente, obtener datos de clima, consolidar reportes de perforación).

Arquitectura y Servicios REST

REST (Transferencia de Estado Representacional) o mejor conocido en inglés como *representational state transfer*, es un estilo de arquitectura de software que define o establece un conjunto de estándares, propiedades y buenas prácticas que se pueden implementar sobre HTTP. Su principal función es la de permitir que un desarrollo web pueda operar con otros mediante sus estándares a través de internet o de una red. La característica que destaca a REST es el hecho de que es "*stateless*", es decir, un protocolo que no posee estado.

Esta arquitectura fue definida por Roy Fielding en el año 2000, que además es uno de los principales artífices de la especificación del protocolo HTTP. La principal ventaja de esta arquitectura es que ha aportado a la web una mayor escalabilidad, es decir, dan soporte a un mayor número de componentes y las interacciones entre ellos. Esta ventaja es gracias a una serie de características que presenta la arquitectura REST.

En la actualidad no existe proyecto o aplicación que no disponga de una API REST para la creación de servicios profesionales a partir de ese software. Twitter, YouTube, etc. Hay cientos de empresas que generan negocio gracias a REST y las APIs REST. Sin ellas, todo el crecimiento horizontal sería prácticamente imposible. Esto es así

porque REST es el estándar más lógico, eficiente y habitual en la creación de APIs para servicios de Internet.

Características

- Es un protocolo sin estado, debido a que no se guarda la información en el servidor. Es decir, toda la información será enviada por el cliente en cada mensaje HTTP, consiguiendo un ahorro en variables de sesión y almacenamiento interno del servidor.
- Presenta un conjunto de operaciones bien definidas, siendo las más importantes GET, POST, PUT y DELETE, que se emplean en todos los recursos.
- Utiliza URIs únicas siguiendo una sintaxis universal.
- Emplea hipermedios para representar la información, que suelen ser HTML, XML o JSON.