**M2S17: Introducción a Backend y ExpressJS**

**Hola, nos vemos nuevamente. ¿cómo te fue en la entrega de tu 4to proyecto?, ya estamos finalizando el módulo 2 y estas muy cerca de iniciar el módulo 3. Para esta semana estarás aprendiendo sobre algunos conceptos de Backend.**

**Sigue adelante...**

**ÍNDICE**

* [Conceptos de Backend](https://github.com/U-Camp/BOOT-M2-SEM17#conceptos-de-backend)
* [ExpressJS](https://github.com/U-Camp/BOOT-M2-SEM17#expressjs)
  + [Instalación](https://github.com/U-Camp/BOOT-M2-SEM17#instalaci%C3%B3n)
  + [Instanciamiento](https://github.com/U-Camp/BOOT-M2-SEM17#instanciamiento)
  + [Creación y gestión de APIs](https://github.com/U-Camp/BOOT-M2-SEM17#creaci%C3%B3n-y-gesti%C3%B3n-de-apis)
  + [Ruteo básico](https://github.com/U-Camp/BOOT-M2-SEM17#ruteo-b%C3%A1sico)

**Conceptos de Backend**

En el momento en que nosotros comenzamos a trabajar una aplicación web, es necesario que tengamos claridad de dónde será su ejecución.

Existen dos partes fundamentales:

* Frontend. Desarrolla dentro de la parte del cliente.
* Backend. Desarrolla dentro del servidor.

El servidor implica una nueva área que permitirá realizar procesos síncronos y asíncronos. Asimismo, puede tener una relación directa y segura con una base de datos, el cual permitirá manipular elementos fluídamente.

Para poder trabajar un Backend con JavaScript, utilizaremos Node.js como el ambiente de desarrollo y ExpressJS como la herramienta principal de gestión de rutas, es decir, las diferentes formas en la cual el usuario o mejor dicho, un cliente, accederá.

**ExpressJS**

ExpressJS es un "framework" que te permite establecer una infraestructura web para Node.js.

A través del mismo, puedes desarrollar:

* Aplicaciones Web
* APIs
* Monitoreo de rendimiento

Como tal, puedes acceder a toda la documentación a través de este enlace: <https://expressjs.com/es/>

**Instalación**

Comenzaremos creando un proyecto, inicializando con Node.js:

npm init --yes

Una vez realizado esto, se creará un achivo llamado package.json y posteriormente instalaremos Express:

npm install express

**Instanciamiento**

Empezaremos creando una instancia a través de un archivo llamado index.js. Luego, crearemos una ruta de prueba, utilizando .get.

Después, observa que estamos levantando la instancia a través de un método llamado listen, el cual deriva de la instancia sucedida en app.

const express = require("express")

const app = express()

app.get("/", () => {

res.send("Hola mundo")

})

app.listen("3000", () => {

console.log("Nos hemos conectado correctamente")

})

Para ejecutar el archivo, primero instalaremos una librería adicional que nos permitirá levantar y monitorear los cambios que sucedan en nuestros archivos:

npm install -g nodemon

Luego, debido a que estaremos escuchando en el puerto "3000", es importante situarnos en la carpeta correcta con el archivo inmediato y estableceremos dentro de la terminal el siguiente comando:

nodemon index.js

Listo. Si accedes a localhost:3000 en tu navegador podrás ver un mensaje que diga "Hola mundo".

**Creación y gestión de APIs**

Una API Rest normalmente es código backend ejecutándose del lado del servidor usando algún lenguaje de programación. Como ya se mencionó, NodeJS es Javascript del lado del servidor, pero para poder cumplir cabalmente con la especificación Rest, tiene que ser RESTFul, esto se logra implementando un módulo llamado ExpressJS.

ExpressJS, es el framework web más popular de NodeJS, es la librería subyacente para un gran número de otros frameworks de NodeJS populares. Proporciona mecanismos para:

* Escritura de manejadores de peticiones con diferentes verbos HTTP (Métodos de petición y consulta, como GET, POST, PUT y DELETE) con diferentes rutas (URLs), lo que hace que su uso nativo de REST APIs sea fácil de implementar.
* Añadir procesamiento de peticiones middleware adicionales a cualquier punto dentro del pipeline del manejo de peticiones REST.

A pesar de que Express es en sí mismo bastante minimalista, los desarrolladores han creado paquetes middleware compatibles para abordar casi cualquier problema de desarrollo web, los cuales se pueden localizar en Express Middleware, existen librerías para trabajar con cookies, sesiones, parámetros URL, datos de REST API, cabeceras de seguridad y muchos más, por lo que con su uso facilita la implementación de esas librerías y permite una integración rápida y hecha en la mejor manera posible.

**Ruteo básico**

Para desarrollar el ruteo, empezaremos trabajando con 4 procesos:

C - Crear - POST R - Leer - GET U - Actualizar - PUT D - Borrado - DELETE

Para crear un elemento, usaremos GET. Es necesario que nosotros derivemos información desde nuestra interfaz u otra aplicación, y al momento de acceder al servidor se redirija a esta ruta estableciendo una inserción de datos, generalmente en una base de datos.

app.post('/', function (req, res) {

res.send('Recibido una petición POST');

});

Para leer, GET es el proceso correcto para pedirle a un servidor que necesitaremos obtener y recibir ciertos elementos que nosotros establezcamos en nuestra búsqueda.

app.get('/', function (req, res) {

res.send('Recibida una petición GET');

});

Para actualizar, el método será PUT, el cual permitirá indicar que nos queremos situar en un elemento y posteriormente, realizar un cambio en los valores de sus propiedades.

app.put('/', function (req, res) {

res.send('Recibida una petición PUT');

});

Finalmente, para borrar, el método indicado será DELETE, el cual contendrá una indicación de borrado de un elemento específico.

app.delete('/', function (req, res) {

res.send('Recibida una petición DELETE');

});

**¡Muy bien!, hasta aquí el contenido de esta semana. Recuerda leer mucho, investigar y hacer consultas en caso de ser necesario.**

**Hasta pronto, nos vemos en clase.**