PRIMERA ENTREGA

Fundamentos de Node.JS

Sebastián Gómez Jaramillo
sgomezja@tdea.edu.co

Tecnológico de Antioquia
Institución Universitaria

Contextualización

Durante el curso los estudiantes de forma individual o grupal van a realizar un proyecto de Software real, aplicando los conocimientos vistos en cada una de las semanas. Este software será aplicando historias de usuario, las cuales variarán en número dependiendo de la modalidad en la que se deseen hacer los trabajos.

Se recomienda realizar el curso con equipos de trabajo para desarrollar habilidades blandas (trabajo en equipo, resolución de problemas, aprendizaje colaborativo, entre otras) y además para poder aplicar adecuadamente los conceptos de metodologías ágiles en los que se soportan las actividades del curso.

Para la primera semana, debido a que solamente se han visto los fundamentos y los conceptos más básicos de Node.js todos los trabajos se harán de forma individual.

El desarrollo que se hará durante el curso será para la dependencia de Educación Continua del Tecnológico de Antioquia, la cual es la encargada de la creación, difusión, desarrollo y certificación de cursos de extensión, que son ofrecidos a comunidad académica en general y no hacen parte de la oferta formal del Tecnológico de Antioquia. Un ejemplo de curso de educación continua es el actual de Node.js. Este curso, tuvo un proceso inicial de difusión, posteriormente los interesados se inscribieron (en caso de ser necesario, se hace un proceso de pago), luego fueron matriculados en el curso, actualmente lo están desarrollando y finalmente, el docente del curso notifica cuales son los estudiantes que lo han aprobado para pasar al proceso final de evaluación del curso y de certificación.

A continuación, se presenta la primera actividad del curso “Introducción a Node.JS mediante el desarrollo de un proyecto ágil”.

Las historias de usuario de las semanas siguientes serán más completas y estarán enfocadas en todo el proceso completo (front y back), en este caso al ser sobre las bases de Node.JS se hace una aproximación de historias de usuario.

Historias de Usuario

Como interesado necesito obtener la información de cursos ofertados por educación continua para así tener diferentes opciones para elegir posteriormente un curso al cual matricularme.
Criterios de aceptación • Deben mostrar al menos tres cursos • Al mostrar cada curso debe indicar el id, el nombre, la duración y el valor • Al momento de listar los cursos, debe transcurrir un intervalo de tiempo de 2 segundos entre cada curso

Como interesado necesito seleccionar un curso para ingresar mis datos y quedar como prematriculado.
Criterios de aceptación • Debe obligatoriamente ingresar el id del curso, nombre del interesado y cedula • Debe buscar la información del curso y retonarme la información del mismo • Debe generar en un archivo de texto con la información del curso en conjunto con la mía • Generar un mensaje de alerta en caso de no encontrar un id. • En caso de no estar interesado debe mostrarme la lista de cursos, en caso de escribir “inscribir” debe hacer el proceso de inscripción

Recomendaciones

- Utilizar un arreglo de objetos.
  - Ejemplo a partir de unas notas de estudiantes
  - Con esto facilitará la búsqueda posterior
  - Se recomienda darle un id a los cursos.

```
let notas = [{
        nombre: 'Juan',
        matematicas: 4,
        ingles: 3,
        programacion: 2
},
{
        nombre: 'Maria',
        matematicas: 5,
              ingles: 1,
              programacion: 3
      }];
```

- Utilizar el find para buscar un elemento dado
  - A continuación, se presenta un ejemplo de la función find que se utiliza para encontrar un elemento dentro de un arreglo.
  - Se declara una variable que va a dar como resultado el elemento encontrado.
  - Se pone el nombre del arreglo.find y crea una función que hace la comparación y que recibe como argumento un arreglo y retorna según la comparación buscada en el arreglo.
  - Se muestra el ejemplo como función tradicional y el ejemplo como función flecha, donde no es necesario escribir function, el argumento no va entre paréntesis porque es uno solo y como se ejecuta todo en el mismo renglón no es necesario utilizar un return.

```
nombre = 'Juan';

let estudianteNota = notas.find(function(notaEst ) {
return notaEst.nombre == nombre})

//o el equivalente

let
estudianteNota = notas.find( notaEst =>  notaEst.nombre == nombre)

      // puede hacerse desde una función que tenga como argumento
el nombre a buscar
```