PRÁCTICA 2: NODEJS y ARCHIVOS JSON

Duración estimada: 100 min**.**

Objetivos:

- Manejar archivos con NodeJs
- Trabajar con formularios y enviar a archivos json para almacenar datos
- Desarrollar aplicaciones en la web
- Manejar modelo Vista-Controlador (MVC)

Introducción

En NodeJS todas las operaciones de acceso al sistema de archivos están englobadas dentro del módulo "fs" (File System).

Importar el módulo fs

Igual que hacemos con otros módulos de Node, hay que procesar el correspondiente require para tener disponibles las funciones de acceso al sistema de ficheros. Se encuentran en el module llamado 'fs' y lo vamos a importar con el siguiente código:

```
let fs = require('fs');
```

- **Método readFile() asíncrono.** Este método accede a un fichero para su lectura y nos entrega el contenido en forma de buffer o en forma de cadena.
  fs.readFile(archivo [, options], callback)
- **Método readFileSync síncrono:** Como alternativa tenemos el método readFileSync() que hace básicamente lo mismo, pero bloquea la ejecución de las siguientes líneas de código hasta que la lectura haya concluído y se tenga el contenido del archivo completo.
  fs.readFileSync(file[, options])
- **Método writeFileSync síncrono:** se comporta de manera similar a fs.writeFile , pero no recibe una devolución de llamada ya que se completa de forma síncrona y, por lo tanto, bloquea el subproceso principal
  fs.writeFileSync

Consultar: https://nodejs.org/api/fs.html

MODELO VISTA CONTROLADOR (MVC)

- **MODEL**: Es el encargado de manejar la información de nuestra aplicación (por ejemplo se encarga de guardar información en un archivo).
- **VIEW:** Es lo que el usuario ve y con lo que interactúa.
- **CONTROLLER:** Es el encargado de unir el model y el view, y también posee la lógica para transformar los datos para que los entienda tanto el Model como la View.

PLANTILLAS HBS:

Para que Express pueda representar archivos de plantilla, deben establecerse los siguientes valores de aplicación:

- views, el directorio donde se encuentran los archivos de plantilla.
  Ejemplo: ***app.set('views', './views')***
- view engine, el motor de plantilla que se utiliza.
  Ejemplo: ***app.set('view engine', 'hbs')***

Los motores de plantilla compatibles con Express como, por ejemplo, **hbs** exportan una función denominada ***__express(filePath, options, callback)***, que es invocada por la función **res.render()** para representar el código de plantilla

Pasar parámetros a plantillas

```
app.get("/", function (req, res) {
    res.render('home', {
        post: {
            author: 'Janith Kasun',
            image: 'https://picsum.photos/500/500',
            comments: []
        }
    });
});
```

los post contiene campos como author, imagey comments. Podemos hacer referencia al post en nuestra plantilla Manillares {{post}}:

```
<div class="posts">
    <div class="row justify-content-center">
        <div class="col-lg-7" style="margin-top: 50px;">
            <div class="card">
                <img src="{{post.image}}"
                    class="card-img-top" alt="...">
                <div class="card-body">
                    <h5 class="card-title">Posted by {{post.author}}</h5>
                    <ul class="list-group">
                        <li class="list-group-item">This is suppose to be a comment</li>
                        <li class="list-group-item">This is suppose to be a comment</li>
                    </ul>
                </div>
            </div>
        </div>
    </div>
</div>
```

Condiciones de uso: Como tenemos lógica condicional, es decir, mostrar los comentarios si están presentes y un mensaje si no lo están, veamos cómo podemos usar condicionales en las plantillas de Handlebars:

```
<div class="posts">
    <div class="row justify-content-center">
```

```
        <div class="col-lg-7" style="margin-top: 50px;">
          <div class="card">

            <img src="{{post.image}}" class="card-img-top" alt="...">
            <div class="card-body">
              <h5 class="card-title">Posted by {{post.author}}</h5>

              {{#if post.comments}}
              <ul class="list-group">
                <!-- Display comment logic -->

              </ul>
              {{else}}
              <ul class="list-group">
                <li class="list-group-item">Be first to comment on this post!</li>
              </ul>
              {{/if}}
            </div>
          </div>
        </div>
      </div>
    </div>
```

Si la sentencia if regresa true, el bloque dentro del #if se renderizará el bloque. Si false, undefined, null, "", 0o [] se devuelven, el bloque no se procesará.

#if solo acepta una condición y no puede usar la sintaxis de comparación de JavaScript (===).

Usar bucles:

Y ahora, en nuestra plantilla, usaremos el #each bucle para revisarlos todos:

```
<div class="posts">
  <div class="row justify-content-center">
    <div class="col-lg-7" style="margin-top: 50px;">
      <div class="card">

        <img src="{{post.image}}" class="card-img-top" alt="...">
        <div class="card-body">
          <h5 class="card-title">Posted by {{post.author}}</h5>

          {{#if post.comments}}
          <ul class="list-group">
            {{#each post.comments}}
            <li class="list-group-item">{{this}}</li>
            {{/each}}
          </ul>
          {{else}}
          <ul class="list-group">
            <li class="list-group-item">Be first to comment on this post</li>
          </ul>
          {{/if}}
        </div>
      </div>
    </div>
  </div>
</div>
```

Dentro de #each bucle, puedes usar this para hacer referencia al elemento que está en la iteración actual. Si tiene una matriz de objetos, también puede acceder a cualquier atributo de ese objeto.

METODOS GET y POST:

- **POST:** Los parámetros POST son aquellos que se envían desde la página web al servidor sin que sean visibles en la URL.

  Lo primero que deberemos de conocer es que los parámetros POST no se envían en la URL si no que se envían en el cuerpo de la petición. En el caso de Express se cuenta con un middleware llamado body-parser el cual nos ayudará a acceder al contenido del cuerpo de los mensajes.

  La estructura para acceder a las variables será: ***req.body.nombrevariable.***

- **GET:** Los parámetros GET son aquellos que se envían mediante la url en un formato de pares clave/valor y después de la URL. A la hora de acceder al parámetro GET utilizamos el objeto req.query seguido del nombre: ***req.query.nombre***

Secuencia/Desarrollo:

1. **Ejercicio 1**: Realizar una web que nos permita realizar el mantenimiento básico de una tabla (Clientes), teniendo en cuenta que los datos se van a almacenar en un archivo json. Una vez almacenados los datos, debes de mostrar los datos de todos los clientes en una tabla. Para ello debes utilizar una plantilla HBS, que recorre el archivo y lo visualiza.

   Para introducir los datos, debes utilizar un fichero html con formato.