EXERCISES QUE TRABAJAREMOS EN EL CUE

- EXERCISE 1: IMPLEMENTANDO UNA API REST.

EXERCISE 1: IMPLEMENTANDO UNA API REST

El objetivo de este ejercicio es plantear una guía paso a paso donde se creará una API para la gestión de libros, consulta, creación, eliminación y actualización.

Principales pasos que se realizarán al crear la API de Libros (To-Do):

1. Configurar el proyecto.
2. Dar inicio a la API.
3. Crear la primera ruta.
4. Crear la ruta de listar los libros.

1. CONFIGURAR EL PROYECTO

Para iniciar la configuración del proyecto, debemos entrar a la terminal y ubicarnos en una carpeta a elección. Seguidamente, realizamos los siguientes pasos:

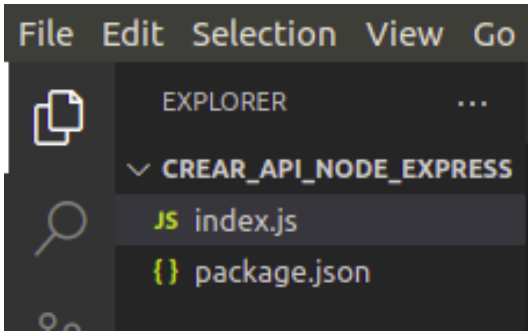
```
# Crear el proyecto
$ mkdir crear_api_node_express
# Accedemos a la carpeta
$ mkdir crear_api_node_express
# Inicializar el proyecto con los parámetros por defecto
$ npm init -y
Wrote ../crear_api_node_express/package.json:
{
  "name": "crear_api_node_express",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

Observamos que se creó el archivo package.json, y el comando `npm init -y` nos indicó que tenemos el index.js como punto de partida a la aplicación, debemos crearlo.

Inicializamos el Visual Studio Code:

```
$ code .
```

Procedemos a crear el archivo index.js:



A continuación, instalamos Express.js, que es la dependencia que se utilizará para crear la API, y es framework de aplicación web de back-end para Node.js. Para instalarlo en el proyecto, se ejecuta:

```
# Instalando Express.js
$ npm install --save express
```

Podemos observar el archivo package.json:

2. DAR INICIO A LA API

Una vez configurado el entorno, editaremos el archivo index.js, y escribiremos el siguiente código:

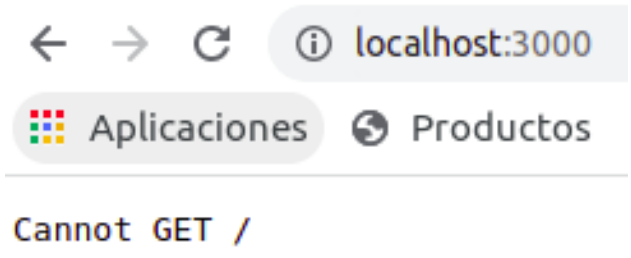
```
// Importando Express.js
const express = require("express");

// Este variable define el puerto del computador donde la API esta
disponible
const PORT = 3000;

// Definimos la variable que inicializa la libreria Express.js
const app = express();

// El siguiente código contiene:
// 1 - El puerto donde esta disponible la API
// 2 - Una función de llamada (callback) cuando la API esta lista
app.listen(PORT, () =>
    console.log(`La API de LIBROS se esta ejecuntando en: http:
//localhost: ${PORT}.`)
);
```

Y se ejecuta en la consola o terminal el siguiente comando:

```
$ node index.js
```

```
Debugger attached.
La API de LIBROS se esta ejecuntando en: http://localhost:3000.
```

Al ir a la URL http://localhost:3000/, observamos en nuestro navegador lo siguiente:



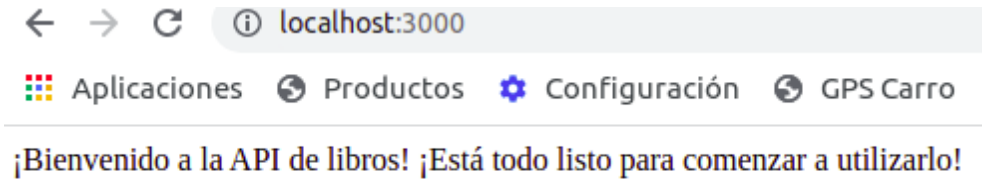
El error reflejado es que no se crearon rutas. Por lo tanto, crearemos una `GET` que nos muestre un mensaje sobre la API que estamos construyendo.

3. CREAR LA PRIMERA RUTA

Procedemos a crear nuestra primera ruta `GET`, agregando el siguiente código al *index.js* de la sección a continuación:

```
const app = express();

// El siguiente código crea una ruta GET con estos parámetros:
// 1 - La ruta donde se ejecutará el código
// 2 - La función que contiene el código a ejecutar
app.get('/', (request, response) => {
    // Se despliega una cadena de caracteres en http://localhost:3000
    response.send('¡Bienvenido a la API de libros! ¡Está todo listo para comenzar a utilizarlo!')
})
```

Procedemos a detener nuestro script con `CTRL + X` y ejecutando nuevamente *node index.js* y al consultar en el navegador observamos:



4. CREAR LA RUTA DE LISTAR LOS LIBROS

Al crear la ruta anterior, se generará una que devolverá el listado de libros que tenemos. Se debe tener en cuenta que el conjunto de libros se encuentra en una lista de libros construido de la siguiente manera:

```
let listaLibros = [{
        isbn: '978-1-61729-242-2',
        titulo: 'Express in Action',
        autor: 'EVAN M. HAHN',
    },
```

```
    {
        isbn: '978-1-78398-586-9',
        titulo: 'RESTful Web API Design with Node.js',
        autor: 'Valentin Bojinov',
    },
];
```

Nombre de la ruta: `"/libros"`

Variable: lista de Array.

Retorno: JSON (es el formato de respuesta utilizado por casi todas las API)

Método API: `GET` (porque queremos OBTENER todos los libros)

Agregamos el siguiente código:

```
app.get("/libros", (request, response) => {
    // La función devolverá su lista de libros en un formato JSON
    return response.json({
        todosLibros: listaLibros
    });
});
```

Observamos en el navegador el retorno de la misma. Para ello, se reinicia el servidor. Si el servidor ya se está ejecutando, presione `CTRL + C` para detenerlo primero, lo inicia, y en el navegador http://localhost:3000/libros se debería ver una respuesta JSON con todos los libros que ha agregado.

Para visualizar mejor los datos en formato JSON, se puede instalar la versión para el navegador Chrome JSON Viewer, en el cual tendremos el siguiente resultado:

NOTA:

Para una mejor resolución del proyecto, es recomendable tener actualizada la versión de Node.Js a la más reciente, y codificar con ES6.