**Taller API 1**

**Cuestionario**

1. **¿Qué es Node.js?**

Node.js es una herramienta que permite ejecutar JavaScript en el servidor, en lugar de solo en el navegador. Esto significa que puedes usar JavaScript para crear aplicaciones web completas, no solo para hacer que las páginas se vean bien.

1. **¿Cómo se crea un proyecto en node?**

* Crea una carpeta para tu proyecto (por ejemplo, mi-proyecto).
* Abre la terminal y navega a esa carpeta usando cd mi-proyecto.
* Ejecuta npm init para inicializar un nuevo proyecto. Esto te guiará a través de varias preguntas para crear un archivo package.json, donde se almacenarán las configuraciones del proyecto.
* Puedes usar npm init -y para crear un package.json con valores predeterminados.

1. **¿Para qué es el package json?**

El package.json es un archivo que contiene metadatos sobre el proyecto, incluyendo dependencias, scripts, versiones y otra información relevante. Es fundamental para gestionar las bibliotecas que tu proyecto necesita.

1. **¿Qué es express y para qué se utiliza?**

Express es una herramienta que hace que sea más fácil crear aplicaciones web con Node.js. Te ayuda a manejar las rutas (dónde van las solicitudes), a gestionar datos y a organizar mejor tu código.

1. **Escriba el código de un servidor básico en express**

const express = require('express'); *// Importamos la biblioteca Express*

const app = express(); *// Creamos una instancia de Express*

const PORT = 3000; *// Definimos el puerto en el que escuchará el servidor*

const host = "127.0.0.1"; *// Definimos la dirección IP del host*

*// Definimos una ruta para la página principal*

app.get('/', (*req*, *res*) => {

*res*.send('¡Código!'); *// Cuando se accede a esta ruta, se envía el mensaje "¡Código!"*

});

*// Iniciamos el servidor y le decimos que escuche en el puerto definido*

app.listen(PORT, () => {

    console.log(`Servidor escuchando en http://${host}:${PORT}`); *// Mensaje en la consola para confirmar que el servidor está corriendo*

});

1. **¿Qué es una API?**

Una API (Interfaz de Programación de Aplicaciones) es un conjunto de reglas y protocolos que permite que diferentes aplicaciones se comuniquen entre sí. Facilita el intercambio de datos y funcionalidades.

1. **¿Cuáles son los métodos http?**

* GET: Recuperar datos.
* POST: Enviar datos.
* PUT: Actualizar datos.
* DELETE: Eliminar datos.
* PATCH: Modificar parcialmente un recurso.

1. **¿Qué es API REST?**

Una API REST es un tipo de API que sigue ciertas reglas para hacer que la comunicación entre programas sea más sencilla. Utiliza los métodos HTTP y se enfoca en recursos, como datos de usuarios o productos.

1. **¿Cómo se instala express en un proyecto Node.js?**

Ejecuta el siguiente comando en la terminal:

npm install express

1. **Le solicitan crear los métodos POST y PUT para una API REST. ¿Cómo testear su funcionamiento rápidamente?**

Se puede usar aplicaciones como Postman, que te permite enviar solicitudes al API y ver las respuestas fácilmente. Así se puede probar si tus métodos funcionan correctamente.

1. **Le solicitan que realice una api que devuelva el promedio de temperaturas de un array en el que se encuentran almacenadas las temperaturas por día ¿Cuál método http emplearía?**

El método GET, ya que estás recuperando datos (el promedio de temperaturas).

1. **Responda Falso o Verdadero:**

JSON es un lenguaje de programación para crear APIS **Falso**

Express es un Framework para Node.js **Verdadero**

JSON es un formato empleado para intercambiar datos entre lenguajes de programación. **Verdadero**

1. La siguiente API <https://www.datos.gov.co/resource/sdmr-tfmf.json> contiene datos de los Medicamentos Vitales No Disponibles en Colombia.

Tomado de <https://dev.socrata.com/foundry/www.datos.gov.co/sdmr-tfmf>

1. Teste el método GET de la API.¿Qué obtuvo, un array de simple o un array deobjetos?

Se obtiene un array de objetos, donde cada objeto tiene información sobre un medicamento.

1. Su jefe le solicita que realice inserciones a la api de medicamentos, si lo podría realizar. Justifique la respuesta.

No se podría hacer inserciones a esta API a menos que tenga un lugar específico para eso y que tenga permiso. La mayoría de las APIs solo permiten leer datos por razones de seguridad.

1. Le solicitan que analice si el formato del valor de la clave cantidad\_solicitada, requiere alguna conversión para realizar cálculos en el frontend ¿Qué respondería?

Depende del formato de cantidad\_solicitada. Si es un número en forma de cadena, necesitarás convertirlo a un número (usando parseInt o parseFloat) para realizar cálculos.

1. Le solicitan listar en un html los valores del nombre\_comercial pero en minúsculas. ¿Qué haría, agregar código al frontend que programará o solicitaría a los proveedores de la api que conviertan los valores del campo a minúsculas?

Lo mejor sería agregar código en el frontend que convierta los nombres a minúsculas cuando los muestres. Esto es más rápido y fácil que pedir cambios a la API.

**API:**

**Codifique en inglés.**

**API**

Dado el siguiente array:

const ventas = [

{

id: 1,

numeroFactura: 'FAC001',

numeroUnidades: 10,

nombreCliente: 'Juana de Arco',

valorTotal: 500

},

{

id: 2,

numeroFactura: 'FAC002',

numeroUnidades: 5,

nombreCliente: 'Policarpa Salavarrieta',

valorTotal: 250

},

{

id: 3,

numeroFactura: 'FAC003',

numeroUnidades: 20,

nombreCliente: 'Tutankamon',

valorTotal: 1000

},

{

id: 4,

numeroFactura: 'FAC004',

numeroUnidades: 7,

nombreCliente: 'Sócrates',

valorTotal: 350

},

{

id: 5,

numeroFactura: 'FAC005',

numeroUnidades: 12,

nombreCliente: 'Simón Bolivar’,

valorTotal: 600

}

];

const sales = [

    {

      id: 1,

      invoiceNumber: 'FAC001',

      unitsNumber: 10,

      customerName: 'Juana de Arco',

      totalValue: 500

    },

    {

      id: 2,

      invoiceNumber: 'FAC002',

      unitsNumber: 5,

      customerName: 'Policarpa Salavarrieta',

      totalValue: 250

    },

    {

      id: 3,

      invoiceNumber: 'FAC003',

      unitsNumber: 20,

      customerName: 'Tutankamon',

      totalValue: 1000

    },

    {

      id: 4,

      invoiceNumber: 'FAC004',

      unitsNumber: 7,

      customerName: 'Socrates',

      totalValue: 350

    },

    {

      id: 5,

      invoiceNumber: 'FAC005',

      unitsNumber: 12,

      customerName: 'Simón Bolivar',

      totalValue: 600

    }

  ];

1. **Crear un node un proyecto llamado facturación**
2. **Crear en node el método get que calcule y retorne en json la suma total de número de unidades facturadas.**
3. **Crear en node el método get que calcule y retorne en json el valor total facturado.**
4. **Crear en node el método get que retorne los datos de una factura de venta de acuerdo al id a consultar**
5. **Crear en node el método get que retorne en un json el id de la factura con el nombre del cliente de todas las facturas.**
6. **Crear el método para insertar facturas en el array siguiendo la misma estructura de los objetos del ejemplo.**
7. **Crear el método put que permita realizar al usuario un decremento entre un 1 y 10% al valor total de todas las facturas. El % lo debe enviar al testear la api.**
8. **Crear en node el método para eliminar una factura.**

const express = require('express');

const app = express();

const port = 3000;

const bodyParser = require('body-parser');

app.use(bodyParser.json()); *// body parse es un middleware que nos permite acceder al cuerpo de la petición*

const ventas = [

    {

        id: 1,

        numCheck: 'FAC001',

        numUnit: 10,

        clientName: 'Juana de Arco',

        value: 500

    },

    {

        id: 2,

        numCheck: 'FAC002',

        numUnit: 5,

        clientName: 'Policarpa Salavarrieta',

        value: 250

    },

    {

        id: 3,

        numCheck: 'FAC003',

        numUnit: 20,

        clientName: 'Tutankamon',

        value: 1000

    },

    {

        id: 4,

        numCheck: 'FAC004',

        numUnit: 7,

        clientName: 'Sócrates',

        value: 350

    },

    {

        id: 5,

        numCheck: 'FAC005',

        numUnit: 12,

        clientName: 'Simón Bolivar',

        value: 600

    }

];

*//2.    Crear en node el método get que calcule y retorne en json la suma total de número de unidades facturadas.*

app.get('/unidades', (*req*, *res*) => {

    let suma = 0;

    ventas.forEach(*element* => {

        suma += *element*.numUnit;

    });

    res.json({ suma }); *// retorna la suma de las unidades en json*

    console.log(suma);

}

);

*//3.    Crear en node el método get que calcule y retorne en json el valor total facturado.*

app.get('/total', (*req*, *res*) => {

    let suma = 0;

    ventas.forEach(*element* => {

        suma += element.value \* element.numUnit;

    });

    res.json({ suma });

    console.log(suma);

}

);

*//4.    Crear en node el método get que retorne los datos de una factura de venta de acuerdo al id a consultar*

app.get('/ventas/:id', (*req*, *res*) => {

    const id = req.params.id;

    const venta = ventas.find(*element* => element.id == id);

    res.json(venta);

}

);

*//5. Crear en node el método get que retorne en un json el id de la factura con el nombre del cliente de todas las facturas.*

app.post('/ventas', (*req*, *res*) => {

    const { id, numCheck, numUnit, clientName, value } = req.body; *// destructuración de objetos*

    const newVenta = {

        id,

        numCheck,

        numUnit,

        clientName,

        value

    };

    ventas.push(newVenta);

    res.json(newVenta);

})

*//6.    Crear el método para insertar facturas en el array siguiendo la misma estructura de los objetos del ejemplo.*

app.post('/ventas', (*req*, *res*) => {

    const { id, numCheck, numUnit, clientName, value } = *req*.body;

    const newVenta = { id, numCheck, numUnit, clientName, value };

    ventas.push(newVenta);

*res*.status(201).json(newVenta);

});

*//7.    Crear el método put que permita realizar al usuario un decremento entre un 1 y 10% al valor total de todas las facturas. El % lo debe enviar al testear la api.*

app.put('/ventas/decrementar', (*req*, *res*) => {

    const { percentage } = *req*.body;

    if (percentage < 1 || percentage > 10) {

        return *res*.status(400).json({ message: 'Percentage must be between 1 and 10' });

    }

    ventas.forEach(*venta* => {

*venta*.value \*= (1 - (percentage / 100));

    });

*res*.json({ message: 'Values updated', ventas });

});

*//8.    Crear en node el método para eliminar una factura.*

app.delete('/ventas/:id', (*req*, *res*) => {

    const id = parseInt(*req*.params.id);

    const initialLength = ventas.length;

    ventas = ventas.filter(*element* => *element*.id !== id);

    if (ventas.length < initialLength) {

*res*.json({ message: 'Invoice deleted' });

    } else {

*res*.status(404).json({ message: 'Invoice not found' });

    }

});

app.listen(PORT, () => {

    console.log(`Servidor escuchando en http://${host}:${PORT}`);

});