JavaScript 2022

Práctica 4

Objetivos

- node.js, npm.

Ejercicio 1

Siguiendo los pasos de este tutorial construya una aplicación para servir **archivos estáticos** como por ejemplo `html` , `css`, `js` y, `png`.

A medida que sigue el tutorial, analice qué hace cada uno de estos comandos.

- `mkdir express-static-file-tutorial`
- `cd express-static-file-tutorial`
- `npm init -y`
- `touch index.js`
- `npm install express --save`

Nota: en Windows `mkdir` es `md` y el comando `touch` de Linux crea un archivo vacío.

1- ¿Qué función cumple el archivo `package.json`?

2- ¿Qué versión de la biblioteca `express` está usando?

3- ¿Por qué se le indica a la app que escuche el puerto 3000?

4- ¿Cómo se le indica a la biblioteca `express` en qué carpeta van a estar los archivos estáticos?

5- ¿Para qué sirve la función `require` que se usa en index.js?

```
const express = require('express');`
```

6- ¿Qué significa este bloque de código en `index.js`?

```
app.get('/', (req, res) => {
    res.send('Hello World!');
});
```

7- Por qué la URL de la imagen `shark.png` es `http://localhost:3000/shark.png` y no `http://localhost:3000/static/shark.png`?

Nota: la documentación de express.

Ejercicio 2

Implemente un formulario de login como el que sigue.

Agregue a la aplicación del **ejercicio 1** un controller para recibir los datos del formulario e imprimirlos **en la consola del servidor**.

¿Con qué método HTTP recibe los datos?

¿Tuvo que realizar alguna configuración en **express** para recibir los datos de un formulario?

Ejercicio 3

Modifique formulario del ejercicio 2 para validar **en el cliente** (es decir en el navegador) que los campos ingresados no sean vacíos y que el nombre de usuario sea una dirección de e-mail válida. En caso de que algún dato no sea válido, no se debe enviar el formulario al servidor y se debe mostrar un mensaje al usuario indicando qué datos son inválidos.

¿Cuál es **la mejor manera** de validar el formato de una dirección de e-mail?

Ejercicio 4

Modifique formulario del ejercicio 3 para que al recibir en el servidor los datos del formulario se verifiquen el usuario y la contraseña comparándolos con 2 constantes predefinidas (por ejemplo `'admin@mail.com'` y `'1234'`).

En caso de ser correctos, se debe **redirigir** a una página que informe el éxito y en caso de no serlo, a la misma página de login.

¿Qué es una redirección en el contexto del protocolo HTTP?

¿Cómo se realiza una redirección utilizando **express**?

Ejercicio 5

Modifique el ejercicio 4 de manera que en lugar de tener 2 constantes con el login válido, haya un archivo llamado `usuarios.json`. Al recibir los datos del formulario se debe buscar en ese archivo para determinar si el usuario y la contraseña son correctos.

¿En qué carpeta del servidor guardó el archivo `usuarios.json` y por qué?

Ejercicio 6

Agregue al archivo `usuarios.json` del ejercicio 5 (que ya tiene el e-mail y la password), el nombre completo de la persona. Por ejemplo el contenido podría ser el que sigue.

```
[
    {
        "username": "mimail@email.com",
        "password": "xasadsaflkjaskjd",
        "name": "Jason"
    },
    {
        "username": "otro@email.com",
        "password": "serewrewrw",
        "name": "Gary"
    }
]
```

Luego implemente un controller para consultar el nombre de un usuario a partir de su `username`. Al acceder a la URL `http://localhost:3000/names?username=mimail@email.com` el resultado debe ser

```
{
    "name": "Jason"
}
```

¿Con qué método HTTP se recibe la solicitud? ¿Cómo se devuelve una respuesta en formato JSON con **express**? ¿Qué devuelve si se recibe como parámetro un username que no existe en el archivo? ¿Qué devuelve si no se envía el parámetro `username`? ¿Qué sucede si se accede a esta URL `http://localhost:3000/names?username=mimail@email.com&username=otro@email.com`?

Ejercicio 7

Basándose en el **ejercicio 13 de la práctica 2**, guarde los datos de las personas en un archivo JSON llamado `people.json`. Luego implemente una API REST como se indica a continuación.

1. `/overweight_people`: devuelve en formato JSON un arreglo con los nombres de las personas con un IMC mayor a 25.
2. `/people_by_age`: devuelve en formato JSON un arreglo de las edades de las personas indexado por el nombre de cada una. (Por ejemplo algo de la forma `["Bobby": 22, "Mark": 36]`).
3. `/imc_over_40`: devuelve en formato JSON un arreglo con el IMC de los mayores de 40.
4. `/average_imc`: devuelve en formato JSON el IMC promedio de todas las personas. Por ejemplo `{"avg": 23}`.
5. `/youngest`: devuelve en formato JSON la persona más joven.
6. `/people_by_height`: devuelve en formato JSON un arreglo de personas ordenadas por estatura.

Por ejemplo, acceder a la URL `http://localhost:3000/average_imc` devuelve `{"avg": 23}`.

Ejercicio 8

Implemente una página HTML que consuma la API del ejercicio 7 y muestre los datos de las personas en el navegador.

Ejercicio 9

Implemente una calculadora del poder de compra del dólar estadounidense como la que está disponible en www.usinflationcalculator.com. Esta calculadora tiene dos componentes.

- Del lado del servidor, utilice la serie del índice de precios al consumidor (CPI) BLS CPI para implementar un servicio que reciba los parámetros necesarios para realizar el cálculo y devuelva el resultado. Debe tener en cuenta las validaciones necesarias y tanto la respuesta correcta como las posibles respuestas ante errores.
- Del lado del cliente, implemente la interfaz con el usuario y utilice AJAX para consumir el servicio del ítem anterior.

Opcionalmente puede incorporar en la respuesta el porcentaje de inflación que hubo en el período solicitado o calcularlo en el cliente.

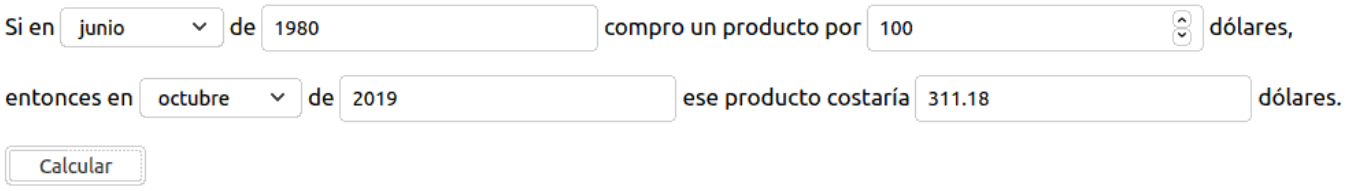
Analice qué parámetros debe enviar en la solicitud y con qué método (GET, POST u otro).

Tenga en cuenta que debe validar los datos tanto en el cliente (navegador) antes de enviarlos como en el servidor al recibirlos. Por ejemplo en el cliente puede validar que el año esté completado y sea un número, pero sólo en el servidor puede validar que exista el dato de CPI para ese año.

Analice qué respuesta dar en caso de detectar errores al recibir la solicitud en el servidor.

En el cliente debe analizar la respuesta obtenida para determinar si fue correcta o un error. En cada caso se debe informar al usuario.

Este es un ejemplo de cómo debería funcionar la calculadora.



La fórmula que se debe aplicar está explicada en US Inflation Calculator. Tenga presente que esa calculadora no tiene en cuenta el mes del año, pero sí se debe tener en cuenta en esta versión.

Por ejemplo, dados

- el CPI de enero de 1984: `101,9`
- el CPI de enero de 2022: `281,148`

si en enero de 1984 una computadora Apple Macintosh costaba `USD 2.495,00`, en enero de 2022 el precio ajustado por inflación sería `6.883,85`.

```
6.883,85 = 2495 / 101,9 * 281,148
```

Consideraciones sobre el ejercicio

- Se debe realizar en grupo.
- Se debe subir al repositorio git asignado al grupo en gitlab.
- Será evaluado.
- Cada integrante del grupo debe hacer sus *commits* con su aporte a la solución.