Consumiendo la Api

Índice

[Cambio de rutas 1](#_Toc125363203)

[Endpoint raiz 1](#_Toc125363204)

[Servir archivos de una carpeta 2](#_Toc125363205)

[plantilla inicial 2](#_Toc125363206)

[Main.js 2](#_Toc125363207)

[Primera función 3](#_Toc125363208)

[ejecución inicial 3](#_Toc125363209)

[Segunda función 3](#_Toc125363210)

[Crear usuarios 4](#_Toc125363211)

[Resultado 5](#_Toc125363212)

[Lista de usuarios 5](#_Toc125363213)

[Eliminar un usuario 6](#_Toc125363214)

Accedemos a api.js

# Cambio de rutas

*//para consumir la api se cambian las rutas*

app.get('/usuarios',user.list)

# Endpoint raiz

*//endpoint de la raiz*

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

    console.log(\_\_dirname);

*//método para enviar un archivo html*

*//dirname envia la ubicacion a sendFile*

*res*.sendFile(`${\_\_dirname}/index.html`)

})

\_\_dirname



Se crea un archivo index.html



Se muestra el archivo en la ruta localhost:3000/



<head>

        <title>Fomulario de nombres</title>

*<!-- importa el archivo main.js -->*

        <script src="main.js"></script>

    </head>

Dentro del index.html se va a importa un archivo main.js

Para eso en api.js se debe realizar un proceso

# Servir archivos de una carpeta

Después de la línea de conexión a la base de datos

*//servir archivos de una carpeta*

*//es una funcion de express*

*//static es la carpeta lo sive /archivo.js*

*//app es el nombre de la carptea contenedora*

app.use(express.static('app'))

importante se debe crear la carpeta

y el archivo main.js

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente 

En index.html

plantilla inicial

<!DOCTYPE html>

<html>

    <head>

        <title>Fomulario de nombres</title>

*<!-- importa el archivo main.js -->*

        <script src="main.js"></script>

    </head>

    <body>

    </body>

</html>

# Main.js

Luego esto se ejecutara en main.js

# Primera función

const loadInitialTemplate = () => {

*//constante donde se agregara el html*

    const template = `

    <h1>Usuarios</h1>

    <from id="user-form">

        <div>

            <label>Nombre</label>

            <input type="text" placeholder="Milton" name="nombre"/>

        </div>

        <div>

            <label>Apellidos</label>

            <input type="text" placeholder="Angamarca" name="apellidos"/>

        </div>

        <button type="submit">Enviar</butston>

    </form>

    <ul id="user-list"></ul>

    `

*//adjunta el texto html a la etiqueta body*

*//body que se encuentra en index.html*

    const body = document.getElementsByTagName('body')[0]

*//asignacion el html dentro de la etiuqueta body*

    body.innerHTML = template

}

*//cuando ha cargado todo el contendio de la ventada*

*//cuando se cargue todo el html*

# ejecución inicial

window.onload = () => {

    loadInitialTemplate();

}

En la web

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

# Segunda función

*//segunda funcion*

const addFormListener = () =>{

*//busca el fromulario por el id*

*//se llama user-form*

    const userForm=document.getElementById('user-form')

*//agregar event listener*

    userForm.onsubmit = async (*e*) => {

*//evitar que se refresque*

*e*.preventDefault()

*//obtiene todos los datos de un formulario*

*//se envia una referencia al html (userForm)*

        const formData = new FormData(userForm)

*//constante*

*//transforma un objeto iterable de entradas*

*//transforma en un objeto*

        const data = Object.fromEntries(formData.entries())

        console.log(data)

    }

}

Acceder a la web

Se ve que trae todos los datos del formulario y devuelve un objeto json

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente Interfaz de usuario gráfica

Descripción generada automáticamente

# Crear usuarios

Crear usuarios en la base de datos

En la segunda función se realizan modificaciones

*//llamado a el endpoint usuarios*

*await* fetch('/usuarios',{

*//metodo a usar post*

            method:'POST',

*//cuerpo del mensaje*

*//convertirlo como string*

            body: JSON.stringify(data),

*//cabecera*

            headers:{

                'content-Type':'application/json'

            }

        })

*//setear los campos del formulario*

        userForm.reset()

    }

}

# Resultado

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Correo electrónico

Descripción generada automáticamente

# Lista de usuarios

*//tercera funcion*

*//mostrar datos*

const getUsuarios= async ()=>{

*//llamado al endpoint usuarios*

*//fetch por defecto realiza usa el método get*

    const response=*await* fetch('/usuarios')

*//se guarda la respuesta*

*//se transforma en json*

    const usuarios=*await* response.json()

*//plantilla*

    const template = *usuario*  =>`

    <li>

        ${*usuario*.nombre} ${*usuario*.apellidos}<button data-id="${*usuario*.\_id}">Eliminar</button>

    </li>

    `

*//map texto html*

    const userList=document.getElementById('user-list')

*//reemplazar inner html*

    userList.innerHTML=usuarios.map(*usuario*=>template(*usuario*)).join('')

}

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

# Eliminar un usuario

Para el método eliminar uno se va a aplicar a continuación de la lista de usuarios

*//asignar comportamineto de los nodos*

*//no retorna nada*

    usuarios.forEach(*usuario* => {

*//buscar el boton*

*//(`[]`)se indica el selector*

        const usuarioNodo=document.querySelector(`[data-id="${*usuario*.\_id}"]`)

*//se le asigna un evento al boton*

        usuarioNodo.onclick=async *e* =>{

*//se relaiza la peticion*

*await* fetch(`/usuarios/${*usuario*.\_id}`,{

                method: 'DELETE',

            })

*//se debe subir un nodo para elminar el elemento*

            usuarioNodo.parentNode.remove()

*//comunicar que fue elminado el usuario*

            alert('Eliminado con éxito')

        }

    });

}