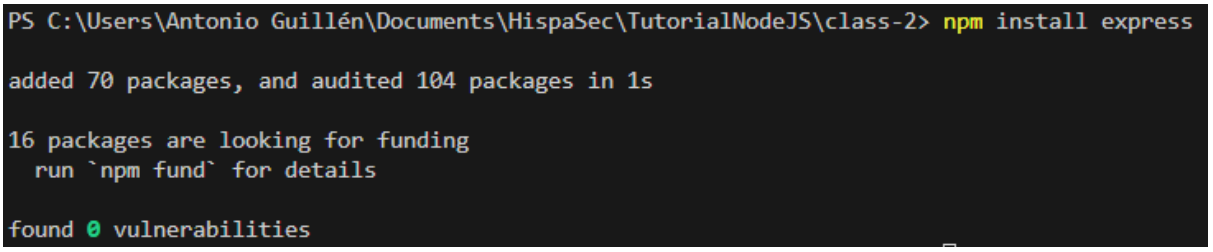
Manual ExpressJS
Antonio Guillén

Vamos a comenzar con nuestro manual de la instalación y creación de un servidor web con **ExpressJS**.
Para ello debemos de saber que es esta tecnología es un framework web para Node.js que simplifica el desarrollo de aplicaciones web y APIs proporcionando una capa de abstracción sobre el servidor HTTP nativo de Node.js. Es minimalista, flexible y popular debido a su simplicidad y la amplia gama de middleware disponibles. Permite a los desarrolladores crear aplicaciones web de manera eficiente y escalable.
Ahora sí comencemos.
Para empezar debemos instalar el paquete, en el caso de este manual vamos a usar **npm**.



Una vez instalado, debemos de crear una aplicación, para ello debemos importar el módulo **express** y seguidamente crear una instancia mediante: **express()**
Al tener la instancia podremos indicarle las rutas:

- **disable(value:** JSONProperty**):** Deshabilita propiedades a la hora de enviar respuestas, esto por ejemplo se puede utilizar para deshabilitar la propiedad que Express te crea por defecto llamada **x-powered-by** utilizada para indicar que se está utilizando dicha tecnología, esto puede ser un problema de seguridad, así que es recomendable desactivarla

- **listen(port:** int,**function:** () => {}**):** Le indica por que puerto está escuchando
- **get(route:** string**, function:** (**req:** request, **res:** response) => {}**):** Lanza una función para el método **GET** y la ruta especificada
- **post(route:** string**, function:** (**req:** request, **res:** response) => {}**):** Lanza una función para el método **POST** y la ruta especificada
  <TODO>
- **use(function:** (**req:** request, **res:** response) => {}**):** Lanza una función para todos los métodos

Cuando estemos manejando las solicitudes y enviemos las respuestas, lo harán con un código **200** por defecto, pero si queremos especificarlo, antes de lanzar la función, debemos llamar a la función **.status(code:** int**)** y seguidamente enviar la respuesta.
podremos hacer varias cosas:

- **res.send(value:** string**):** Sirve para enviar la respuesta en formato **string**
- **res.json(value:** json**):** Sirve para enviar una respuesta en formato **JSON**.

Es aconsejable usar **MiddleWares**, estos son en el contexto de Express.js una función que se ejecuta en el ciclo de vida de una solicitud HTTP. Estas funciones tienen acceso tanto al objeto de solicitud como al objeto de respuesta, y se utilizan para ejecutar código antes, después o durante el manejo de una solicitud. Los MiddleWares pueden realizar tareas como autenticación, autorización, validación de datos, compresión de respuestas, registro de solicitudes y manejo de errores. Son esenciales en Express.js y permiten modularizar y reutilizar la lógica de la aplicación de manera eficiente.

Entonces, para utilizar uno de estos puedes usar cualquier función para manejar solicitudes como get(), post() o use() pero añadiéndole un tercer parámetro llamado **next**, en este caso usaremos un **use()** y así funcionará con todos nuestros métodos.
Cuando lo tengamos metemos toda la lógica que queramos que se ejecute en el MiddleWare y muy importante, cuando terminemos ejecutar la función **next()** ya que si no quedará esperando infinitamente
Aquí sería un ejemplo de un servidor web con todo lo aprendido hasta el momento

```js
const ditto = require('./pokemons/ditto.json')
const express = require('express')
const app = express()

// Indica el puerto mediante variables de entorno o uno por defecto
const PORT = process.env.PORT ?? 1234

// Desactiva la propiedad expecificada
app.disable('x-powered-by')

// Creación de MiddleWare para transformar solicitudes POST JSON
// Tenemos varias opciones, obviamente usar una a la vez
// Opción - 1
app.use((req, res, next) => {
    if (req.method === 'POST' && req.headers['content-type'] === 'application/json') {
        let body = ''

        req.on('data', (chunk) => {
            body += chunk.toString()
        })

        req.on('end', () => {
            const data = JSON.parse(body)
            req.body = data

            next()
        })
    }else{
        next()
    }
})

// Opción - 2
app.use(express.json())

// Manejo de rutas por GET
app.get('/pokemon/ditto', (req, res) => {
    res.json(ditto)
})

// Manejo de rutas por POST
app.post('/pokemon', (req, res) => {
    console.log(req.body)
    res.status(201).json(req.body)
})

// Manejo de error 404 al no encontrar ruta
app.use((req, res) => {
    req.status(404).send('<h1>Error 404</h1>')
})

// Indicamos el puerto especificado arriba
app.listen(PORT, () => {
    console.log(`Server running on: http://localhost:${PORT}`)
})
```