Clase 1

Webservices con express

Temario

1. **Qué es Express**
2. **Instalación**
3. **Creación de servidores**
4. Middlewares
5. Enrutamiento
6. Knex/Sequelize
7. Sesiones, Cookies y JWT

¿Qué es Express?

Es el **framework** web más popular de Node, es la librería subyacente para un gran número de otros frameworks web de Node populares. Proporciona mecanismos para:

- Escritura de manejadores de peticiones con diferentes verbos HTTP en diferentes direcciones URL (rutas).
- Integración con motores de renderización de "vistas" para generar respuestas mediante la introducción de datos en plantillas.
- Establecer ajustes de aplicaciones web como qué puerto usar para conectar, y la localización de las plantillas que se utilizan para renderizar la respuesta.
- Añadir procesamiento de peticiones "middleware" adicional en cualquier punto dentro de la tubería de manejo de la petición.

Framework: Conjunto de herramientas y módulos que pueden ser reutilizados para varios proyectos.

Utilizar un framework permite agilizar los procesos de desarrollo ya que evita tener que escribir código de forma repetitiva, asegura unas buenas prácticas y la consistencia del código.

Instalando Express

1. Creamos un directorio y accedemos a él
   `mkdir class1`
   `cd class1`
2. Inicializamos un proyecto de node para generar el archivo **package.json**
   `npm init`

   Este comando nos solicitará información del proyecto como nombre, versión, autor, punto de entrada, etc.
3. Instalamos express como dependencia del proyecto
   `npm install express --save`
4. La opción –save añade el módulo como dependencia en el archivo package.json. Esto permite instalar todas las dependencias de un proyecto con el comando
   `npm install`
5. Abrimos con VSCode el archivo package.json

Package.json:
Archivo json que contiene información relevante del proyecto, nombre, versión, links, dependencias, etc

```
{
  "name": "class1",
  "version": "1.0.0",
  "description": "My first node app",
  "main": "app.js",
  ▷ Debug
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "app",
    "express"
  ],
  "author": "flopez@veridas.com",
  "license": "ISC",
  "dependencies": {
    "express": "^4.18.2"
  }
}
```

Creando Servidores

¿Qué es un servidor web?

Un servidor web es un software que tiene como misión principal devolver información (páginas o datos) cuando recibe peticiones por parte de los usuarios.

En otras palabras, es el software que permite que los usuarios que quieren ver una página web en su navegador puedan hacerlo.

Otros servidores web:

- Apache
- Nginx
- LiteSpeed
- Microsoft IIS
- Sun Java System Web Server

Creando Servidores: Sin express

```
// Se carga el módulo de HTTP
let http = require("http");
// Declaro el puerto de escucha
const port = 3000;
// Creación del servidor HTTP, y se define la escucha
// de peticiones en el puerto ${port}
http.createServer(function(request, response) {
  const url = request.url;
  const method = request.method;
  if (method === 'GET' && url === '/') {
     // Se define la cabecera HTTP, con el estado HTTP (OK: 200) y el tipo de
contenido
     response.writeHead(200, {'Content-Type': 'text/plain'});
     // Se responde la solicitud con el mensaje "Hello World!!"
     response.end('Hello World!!\n');
  } else {
     response.writeHead(404);
     response.end('Not found');
  }
}).listen(port);
// Se escribe la URL para el acceso al servidor
console.log(`Example server listening on http://localhost:${port}`);
```

Creando Servidores: Con express

```
// Se carga el módulo de Express
const express = require('express');
// Creo la aplicación Express
const app = express();
// Declaro el puerto de escucha
const port = 3000;

// Defino la ruta que se llamará cuando se reciba una petición HTTP GET
// en la dirección '/'
// La función callback recibe una petición y una respuesta como argumentos
app.get('/', (req, res) => {
   // Se define la cabecera HTTP con el tipo de contenido
   res.set('Content-Type', 'text/plain');
   // Se responde la solicitud con el mensaje "Hello World!!"
   res.status(200).send('Hello World!!');
});
// Creo el servidor en el puerto ${port}
app.listen(port, () => {
   // Se escribe la URL para el acceso al servidor
   console.log(`Example server listening on http://localhost:${port}`);
});
```

Vistas:
Mustache

```
npm install mustache-express --save
```

```
const express = require("express");
const mustacheExpress = require('mustache-express');
const app = express();
const port = 3000;

app.engine('html', mustacheExpress());

app.set('view engine', 'html');
app.set('views', __dirname + '/views');

app.get('/user/:name', (req, res) => {
   res.render('user', { name: req.params.name })
})
// Creo el servidor en el puerto ${port}
app.listen(port, () => {
 // Se escribe la URL para el acceso al servidor
 console.log(`Example server listening on http://localhost:${port}`);
});
```

```
<!-- views/user.html -->
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <title>Vistas</title>
</head>
<body>
   <p>Hola, {{name}}</p>
</body>
</html>
```

Ejercicios

1. Instalar el paquete **nodemon** y comprobar qué ocurre al arrancar la aplicación con él.
2. Crear el script **start-dev** en package.json que inicie la app con nodemon.
3. Crear una nueva vista y un endpoint **GET /student/:id** que pinte el id en la vista y como “title” ponga “Student”.
4. Refactorizar plantilla con partials (header y footer).

ALBAÑILESDIGITALES

Developing your new career

servicio navarro de empleo

das·nano

VeriDas