Carga de contenido estático

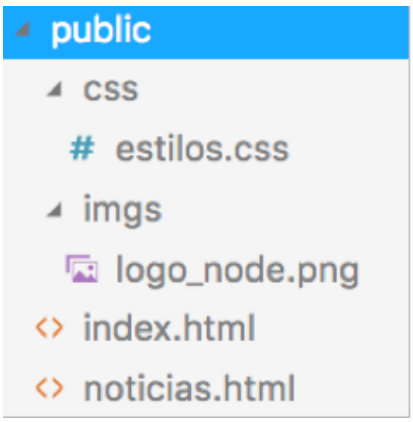
En sesiones anteriores hemos visto cómo estructurar una aplicación express para desarrollar un proveedor de servicios REST. Hemos almacenado los esquemas y modelos de datos en una carpeta `models`, y los enrutadores que dan respuesta a las distintas URLS en una carpeta `routes`, dejando la aplicación principal en la raíz del proyecto.

Esta estructura es la que normalmente se sigue para desarrollar una API REST (recuerda, Express es un framework *unopinionated*, lo que significa que podemos elegir cualquier otra estructura que nos parezca adecuada). Sin embargo, en este puzzle faltan algunas piezas para poder desarrollar aplicaciones web más completas, que permitan servir contenido HTML, más allá de los simples datos JSON. Para empezar, veremos en este documento cómo incorporar contenido estático (páginas HTML, estilos CSS y archivos JavaScript), y cómo estructurarlos en nuestra aplicación Express.

Veremos que con Express la gestión del contenido estático es muy sencilla, a través del correspondiente *middleware* que incorporaremos a nuestros proyectos que lo requieran. Para ello, haremos una prueba en un proyecto llamado "*PruebaExpressEstatico*" en nuestra carpeta de "*ProyectosNode/Pruebas*".

1. Ubicación del contenido estático

A la hora de ubicar el contenido estático de nuestra aplicación, es habitual dejarlo todo en una única subcarpeta, que típicamente se suele llamar "public". Imaginemos que esta subcarpeta tiene esta estructura:



La imagen "logo_node.png" puede ser cualquiera que queráis. En este ejemplo usaremos un logo de Node buscado en Internet:

La hoja de estilos tiene algunos estilos básicos (tipo de letra y color de fondo):

```
body
{
    background-color:rgb(245, 244, 201);
    font-family: Arial;
}
```

La página `index.html` es muy simple, incorporando la hoja de estilos, la imagen, un encabezado de primer nivel y un enlace a la otra página:

```
<!doctype html>
<html>
    <head>
        <link href="css/estilos.css" rel="stylesheet">
    </head>
    <body>
        <div align="center">
            <img src="imgs/logo_node.png" />
        </div>
        <h1>Bienvenido a Node.js</h1>
        <p><a href="noticias.html">Noticias de node</a></p>
    </body>
</html>
```

La página de `noticias.html` es similar a la anterior, mostrando un enlace a la página principal y un listado de ejemplo:

```
<!doctype html>
<html>
    <head>
        <link href="css/estilos.css" rel="stylesheet">
    </head>
    <body>
        <div align="center">
            <img src="imgs/logo_node.png" />
        </div>
        <h1>Bienvenido a Node.js</h1>
        <p><a href="index.html">Página de inicio</a></p>
        <ul>
            <li>Concurso Javascript para adictos a Node.js</li>
            <li>LinkedIn migra desde Rails hacia Node</li>
            <li>Conferencia Node.js en Italia</li>
        </ul>
    </body>
</html>
```

2. Procesamiento del contenido estático

Si queremos que se sirvan automáticamente estos contenidos al acceder a una URI concreta (que puede ser la propia uri "/public" u otra), empleamos el *middleware* **static**, integrado en Express. Como primer parámetro del comando `use` indicamos qué URI queremos utilizar para servir contenido estático (la que nosotros queramos), y en segundo lugar, cargamos el *middleware* `static` indicando en qué carpeta están realmente dichos contenidos (carpeta `/public` en nuestro caso):

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

let app = express();
app.use('/public', express.static(__dirname + '/public'));

app.listen(8080);
```

Opcionalmente, también podemos definir una ruta que redirija a la página de inicio ("index.html") si se intenta acceder a la raíz de la aplicación:

```
app.get('/', (req, res) => {
    res.redirect('/public/index.html');
});
```

Esta forma de servir contenido estático, como podemos deducir, se queda muy corta. No hay casi ninguna web que ofrezca un contenido sin cierto dinamismo; el propio listado de noticias podría extraerse de una base de datos y volcarse en la página... pero para eso necesitamos hacer uso de plantillas, que veremos en documentos posteriores.

3. Incluir estilos predefinidos: Bootstrap

Es posible también incluir los estilos de frameworks de diseño web, como por ejemplo *Bootstrap*, directamente en nuestra aplicación Express, siguiendo estos sencillos pasos:

1. En primer lugar, descargamos Bootstrap como un módulo más de nuestro proyecto:

```
npm install bootstrap
```

2. Después, aplicamos el *middleware* de contenido estático para que cargue por defecto los contenidos de la carpeta `dist` de instalación de Bootstrap, donde se encuentran, entre otras cosas, los estilos.

```
app.use(express.static(__dirname + '/node_modules/bootstrap/dist'));
```

En este caso, los contenidos de Bootstrap se cargarán asociados a la ruta raíz del proyecto (no hemos indicado ningún prefijo como primer parámetro, cosa que sí hemos hecho en el ejemplo anterior `/public`). Por lo tanto, si queremos incluir estilos Bootstrap en cualquier página HTML o vista de nuestro proyecto, podemos hacerlo así:

```
<link rel="stylesheet" href="/css/bootstrap.min.css"/>
```

Y si queremos cargar estilos propios de nuestra carpeta `public/css`, lo haremos de este otro modo, y así no interferirán (cada ruta estática tiene un prefijo diferente):

```
<link rel="stylesheet" href="/public/css/estilos.css"/>
```