


Crear un simple Web Server

 gmoralesc.gitbooks.io/creando-apis-con-node-js-express-y-mongodb/content/configurando-el-proyecto-con-express/crear-un-simple-web-server.html

Creando un simple Web Server

Antes de comenzar el proyecto vamos a crear un servidor Web sencillo, ya que nuestra aplicación será una REST API que se podrá consumir por Web mediante el protocolo HTTP, será un REST API para administrador de publicaciones de usuarios con comentarios y con perfiles por usuario con persistencia de datos con MongoDB.

En el libro de Introducción a Node JS hay un capítulo dedicado a inicialización y configuración de proyectos en Node JS.

Seleccionamos un directorio de trabajo y creamos el directorio de la aplicación:

```
mkdir photobook-api && cd photobook-api
```

Inicializamos el proyecto de Node JS:

```
npm init -y
```

El comando anterior genera el archivo `package.json` que luego se puede editar para cambiar los valores de la descripción, autor y demás.

Inicializamos el repositorio de git:

```
git init
```

Creamos el archivo `.gitignore` en la raíz del proyecto:

```
touch .gitignore
```

Con el siguiente contenido y lo guardamos:

```
node_modules/  
.DS_Store  
Thumbs.db
```

Creamos el archivo principal (o de entrada) de la aplicación:

```
touch index.js
```

Finalmente para terminar con la configuración de la aplicación hacemos el commit inicial:

```
git add --all  
git commit -m "Initial commit"
```

Ya nuestro proyecto está listo y configurado para empezar a trabajar, pero siendo fiel a las buenas prácticas creamos una rama antes de empezar a trabajar en el nuevo *feature*:

```
git checkout -b feature/01-simple-web-server
```

Tomaremos el [ejemplo del Web Server](#) que se encuentra en la documentación de Node JS y reemplazamos el contenido del archivo `index.js` por el siguiente código:

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

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

En el ejemplo anterior Node JS utiliza el módulo de `http` para crear un servidor Web que escuchará todas las peticiones en la dirección IP y puerto establecidos, como es un ejemplo responderá **a cualquier solicitud** con la cadena `Hello World` en texto plano sin formato, solo para mostrar que el servidor Web esta funcionando.

Para ejecutarlo utilizamos la siguiente línea de comando:

```
node index
```

Para ver la respuesta abrimos el navegador en la siguiente dirección:

```
http://localhost:3000/
```

Normalmente en un REST API se consume mediante el protocolo HTTP y la respuesta son objetos en formato JSON (el cual introduciremos más adelante), organizado en una serie de **rutas** que adicionalmente reciben parámetros, el contrato para establecer la relación entre las acciones del REST API y los verbos HTTP es bastante estandarizada, pero no la estructura de la respuesta y la organización de la rutas, en esta aplicación comenzaremos con una estructura sugerida que irá cambiando dependiendo los diferentes casos y necesidades que se presenten.

Antes de finalizar hacemos commit de nuestro trabajo:

```
git add index.js
git commit -m "Create a simple Web server"
```

Más información:

HTTP