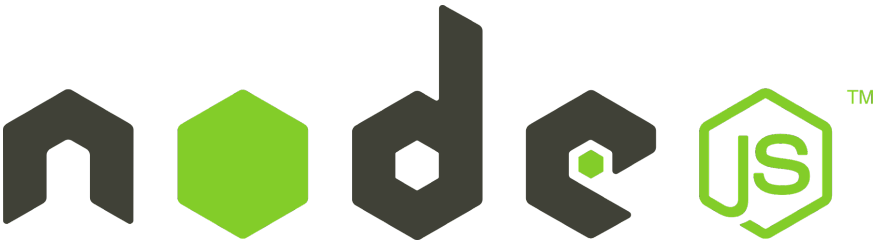
DESARROLLO DE SERVICIOS WEB CON NODEJS



Email: *julietalanciotti@gmail.com*

Módulo 3 - Estructura de un Proyecto

Estructura de archivos

La siguiente estructura de archivos presentada a continuación no es la única que se utiliza, pero sí la más recomendada. La estructura inicial del proyecto contiene:

- **package.json:** se encuentra en la raíz del proyecto y es donde se especificará la información de nuestro paquete de dependencias.
- **index.js:** para la conexión a la base de datos, para configuración de entornos, entre otras cosas.
- **middlewares:** es un directorio para agrupar todos los que se utilicen en la API.
- **models:** directorio para agrupar todos los modelos y esquemas a utilizar por la BD.
- **controllers:** directorio donde se agruparán todas acciones y operaciones sobre nuestra BD.
- **routes:** estarán definidas todas las rutas a las que responderá la aplicación.
- **node_modules:** se genera automáticamente en nuestro proyectos cuando instalamos paquetes o dependencias mediante npm. Cualquier paquete instalado está en este directorio en una carpeta con el nombre del paquete, junto a todos sus ficheros necesarios y dependencias respectivas. Suele ser de gran tamaño.
- **const:** utilizado para crear archivos con variables globales.
- **migrations:** para agrupar todas las migraciones realizadas en el sistema.
- **seeders:** agrupa todos los archivos que se ejecutarán una vez levantado el servidor.

Nodemon

En NodeJS se debe reiniciar el proceso para que los cambios surtan efecto. Y por cada cambio que se realiza deberemos esperar y solucionar los problemas que se presenten.

nodemon es un módulo que cuando detecta algún cambio de archivo en los directorios, actúa. No requiere que se lleve a cabo ningún cambio en nuestra aplicación para ejecutarlo.

Ventajas

1. Es fácil de usar.
2. No afecta al código original y ninguna instancia requiere llamarlo.
3. Ayuda a reducir el tiempo de escritura de código y ejecución.

Instalación

1. Instale el módulo de la siguiente manera de forma global:
   `npm install -g nodemon`

2. Luego de la instalación, puede verificar la versión con el siguiente comando:
   `nodemon version`

3. En caso de tener Windows, es probable que le aparezca un error, para eso recomiendo ver el siguiente video donde explico cómo resolverlo: https://youtu.be/-sZEFhwbuBU

4. Recordar que para que nuestro proyecto funcione con nodemon y sea más sencillo su uso, deberá modificar en la línea del archivo **package.json** la siguiente instrucción:
   `"start": "nodemon src/index.js"` →
   anteriormente decía: `"node src/index.js"`

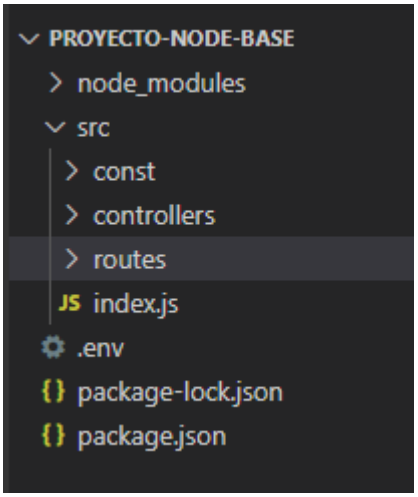
Crear un proyecto base

A continuación vamos a resumir los pasos que necesitamos realizar para lograr tener la base de un proyecto con NodeJS. Tener en cuenta que hasta donde sabemos, sólo podremos verificar el funcionamiento de algún endpoint mostrando algún mensaje.

ℹ️ **Recomiendo ver el video donde hacemos desde cero un proyecto con lo aprendido hasta ahora: https://youtu.be/iRJ0RUUTGLE**

1. Crear un directorio o repositorio en GitHub.
2. Acceder al directorio.
3. `npm init` → creo el proyecto.
4. `npm i express` → instalo express.
5. `npm install dotenv` → nos permite utilizar variables de entorno.
   a. crear archivo .env
   b. cada vez que se quiere hacer referencia a una variable de entorno, se debe utilizar `require('dotenv').config()`
6. Crear directorio **src**
   1. Crear archivo **index.js** → tiene toda la configuración de la api.
   2. Crear directorio **routes** → agrupará todas las rutas existentes de la api.

3. Crear directorio **controllers** → agrupa todas las funcionalidades que se ejecutan cuando se llama a algún endpoint.
4. Crear directorio **const** → agrupa todas las variables que no cambian su valor.



Hasta el momento esta sería la estructura básica de nuestro proyecto.

Constantes

Como un dato extra, dejaré cómo podríamos utilizar constantes (variables que no cambian su valor durante la ejecución del programa) a lo largo de nuestro proyecto.

Para que todo quede más organizado, vamos a crear una carpeta llamada **const** y ahí crearemos un archivo llamado **globalConstants.js.** En él escribiremos todas nuestras constantes.

Estarán separadas por coma, y se les asignará su valor utilizando dos puntos.

Uso de las constantes

Para utilizarlas en el archivo que sea, como primera instancia deberemos importarlas. Para esto escribiremos:

```
const globalConstants = require('../const/globalConstants')
```

De esta manera, cuando queramos usar una constante, solo tendremos que hacer:

```
var a = globalConstants.NOMBRE_DEFAULT;
```

Extensiones útiles de Visual Studio Code

Les dejo algunas extensiones que utilizo para desarrollar apis con NodeJS en Visual Studio Code. A mi parecer son bastantes útiles. Muchas son conocidas, pero para aquellos que no utilizan, tal vez les sirva.

1. **Material Icon Theme:** agrega iconos para diferentes tipos de archivos y ayuda a distinguir rápidamente diferentes archivos en un proyecto.
2. **DotENV: y**a que utilizamos variables de entorno, existe una extensión para resaltar la información de este archivo y que sea más sencillo su seguimiento.
3. **REST Client:** para realizar solicitudes HTTP directamente desde el editor y ver las respuestas desde una ventana separada. De esta manera nos ahorramos tiempo.
4. **Tabnine:** utilizando inteligencia artificial, ayuda a autocompletar el código a medida que vamos programando.
5. **Copilot:** ahora es paga, pero es más completa que la anterior y trabaja de manera similar.