El futuro digital es de todos

MinTIC

UNIVERSIDAD DEL NORTE

Vigilada Mineducación

CICLO IV:

Desarrollo de Aplicaciones Web

MisionTIC2022

El futuro digital es de todos | MinTIC

UNIVERSIDAD DEL NORTE
Vigilada Mineducación

Sesión 10:
Desarrollo de Aplicaciones Web

Desarrollo de Back-End web con Node.js

MisiónTIC2022

Objetivos de la sesión

Al finalizar esta sesión estarás en capacidad de:

1. Configurar el framework de Node.js en un ambiente de trabajo.
2. Implementar una aplicación web básica con Node.js.

Node.js

- Node.js es un entorno de ejecución para aplicaciones de JS de lado del servidor.
- Fue creado por Ryan Dahl, el mismo creador de deno, escrito en C, C++ y JS.
- Node.js cuenta con un repositorio de paquetes open source en npm, adicionalmente npm (Node Package Manager) también es un CLI que se instala en conjunto con node.
- Actualmente Node.js es utilizado para el desarrollo de todo tipo de aplicaciones:
  - De escritorio con Electron.
  - Aplicaciones web con React, Angular, entre otros.
  - De servidor con Express, Nest, Next, entre otros.
  - Móviles con React Native o Ionic.

Node.js - Instalación

- Para instalar Node.js dirigirse a este enlace y descargar la version LTS. Esto nos descargara un archivo binario, el cual deberemos de ejecutar como administrador.
- Siguiendo con la instalación de de Node.js este nos instalará npm, una vez finalizada su instalación podemos revisar su estado mediante una línea de comandos.
- Con los comandos **`node --version`** y **`npm --version`**:

```
PS C:\Users> node --version
v14.16.0
PS C:\Users>
```

```
PS C:\Users> npm --version
6.14.11
PS C:\Users>
```

Node.js - Package.json

- Una vez instalado Node.js, procedemos con ejecutar el comando **`npm init -y`** sobre la carpeta donde queramos ubicar nuestro proyecto.
- Esto nos generará un archivo llamado **`package.json`** donde encontraremos toda la configuración y descripción de nuestro proyecto:

```
{
  "name": "demo",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

Node.js - Interfaz de Comandos

- Nuestro archivo **`package.json`** tiene configurado un archivo de entrada llamado **`index.js`**, inicialmente este archivo no existe y uno tiene que crearlo por su cuenta, por lo que creamos **`index.js`** con la siguiente estructura:

```
console.log('Hola mundo!');
```

- Nuestro archivo **`package.json`** tiene un apartado de scripts el consta de un json donde solo se define el key **“`test`”**.
- Nosotros agregaremos una key que se llamara **“`start`”** y tendrá el siguiente valor:

```
"start": "node index.js",
```

- De esta forma podemos configurar scripts dentro de nuestro proyecto.

Node.js - Interfaz de Comandos

- Luego en nuestra terminal de comandos podemos ejecutar `npm start`:

```
D:\Users\Demar\Code\JS\Node Js\demo  demo@1.0.0
λ npm start

> demo@1.0.0 start D:\Users\Demar\Code\JS\Node Js\demo
> node index.js

Hola mundo!
```

- Como podemos ver esto resulta en lo mismo que ejecutar node `index.js` desde la terminal de comandos:

```
D:\Users\Demar\Code\JS\Node Js\demo  demo@1.0.0
λ node index.js
Hola mundo!
```

Node.js - Buenas Prácticas

- Para tener una buena experiencia de desarrollo con Node.js, por lo general es recomendable hacer lo siguiente:
  - Configurar Prettier para darle formato al código.
  - Configurar Eslint para tener una mejor experiencia al desarrollar
  - Configurar Eslint y Prettier para que estén de acuerdo con el formato y reglas a utilizar.
  - Configurar Babel para utilizar características de siguiente generación como import/export.

Node.js - Prettier

- Primero será necesario que configuremos prettier en nuestro editor de texto, para este caso optamos por usar Visual Studio Code (VSCode).
- Luego instalaremos prettier con el comando **`npm install prettier --save-dev`**.
- Por último debemos agregar el archivo **`.prettierrc`** en nuestro directorio raíz.
- A continuación usaremos el siguiente archivo a modo de ejemplo:

```
{
  "trailingComma": "es5",
  "tabWidth": 2,
  "semi": true,
  "singleQuote": true
}
```

- Para más detalle sobre las reglas ver este enlace

Node.js - Eslint

- Primero será necesario que configuremos eslint en nuestro editor de texto, para este caso optamos por usar Visual Studio Code (VSCode).
- Luego instalaremos **`eslint`** con el comando **`npm install eslint eslint-config-node eslint-config-prettier eslint-config-standard eslint-plugin-import eslint-plugin-node eslint-plugin-prettier eslint-plugin-promise --save-dev`**.
- Por último debemos agregar el archivo **`.eslintrc.json`** en nuestro directorio raíz.
- A continuación usaremos el siguiente archivo a modo de ejemplo:

```
{
  "trailingComma": "es5",
  "tabWidth": 2,
  "semi": true,
  "singleQuote": true
}
```

Node.js - Eslint

- A continuación usaremos el siguiente archivo a modo de ejemplo:

```
{
  "env": { "node": true, "es2021": true },
  "extends": ["standard", "plugin:node/recommended"],
  "parserOptions": { "ecmaVersion": "latest", "sourceType": "module" },
  "rules": {
    "no-unused-vars": "warn",
    "no-console": "off",
    "func-names": "off",
    "no-plusplus": "off",
    "no-process-exit": "off",
    "class-methods-use-this": "off",
    "node/no-unsupported-features/es-syntax": "off"
  }
}
```

- Para más información sobre los archivos de configuración de Eslint ver este enlace.

Node.js - Eslint + Prettier

- Para poder realizar esta configuración debemos instalar los siguientes paquetes: **`eslint-config-prettier eslint-plugin-prettier`**
- Los cuales instalaremos con el comando **`npm install eslint-config-prettier eslint-plugin-prettier --save-dev`**.
- Esto nos permite agregar configuraciones adicionales a **`eslint`** para que trabaje de la mano con prettier.

Node.js - Eslint + Prettier

- Para eso modificaremos nuestra configuración de la siguiente manera:

```
{
  "env": { "node": true, "es2021": true },
  "extends": ["standard", "prettier", "plugin:node/recommended"],
  "parserOptions": { "ecmaVersion": "latest", "sourceType": "module" },
  "plugins": ["prettier"],
  "rules": {
    "prettier/prettier": "error",
    "no-unused-vars": "warn",
    "no-console": "off",
    "func-names": "off",
    "no-plusplus": "off",
    "no-process-exit": "off",
    "class-methods-use-this": "off",
    "node/no-unsupported-features/es-syntax": "off"
  }
}
```

Node.js - Babel

- Por último agregaremos Babel a nuestro proyecto para poder usar funcionales de las versiones más recientes de ECMAScript.
- Para agregar babel ejecutaremos el comando **`npm install @babel/cli @babel/core @babel/node @babel/preset-env --save-dev`**
- Por utlimo agregaremos la siguiente configuración en el archivo **`.babelrc`** en nuestro directorio raíz:

```
{
  "presets": [["@babel/preset-env", { "targets": { "node": true } }]]
}
```

Node.js - Scripts

- Finalmente agregaremos los siguientes scripts o comandos a nuestro proyecto:

```
"start": "npm run build && node ./build/index.js",
"start:dev": "nodemon --exec babel-node src/index .js",
"build:babel": "babel -d ./build ./src -s",
"build": "npm run build:babel",
"prettier": "prettier src/**/*.js",
"prettier:fix": "prettier --write src/**/*.js"
```

- Inicialmente usaremos los comandos `start:dev`, `build` y `start`.

- Para más información ver el repositorio de demostración.

El futuro digital es de todos

MinTIC

UNIVERSIDAD DEL NORTE

Vigilada Mineducación

Ejercicios de práctica

Mision TIC 2022

El futuro digital es de todos MinTIC

UNIVERSIDAD DEL NORTE
Vigilada Mineducación

Referencias

- https://www.npmjs.com/
- https://reactjs.org/
- https://nodejs.org/en/download/

MisiónTIC2022

El futuro digital es de todos

MinTIC

UNIVERSIDAD DEL NORTE

Vigilada Mineducación

¡GRACIAS

POR SER PARTE DE ESTA EXPERIENCIA DE APRENDIZAJE!

Mision TIC 2022