15-17-(11)Crear una API Rest con Express y MySQL

-Utilizaremos Sequelize para conectar la base de datos.

Para trabajar Express, se debe conocer Html, Css y Java Script Basic

## Qué es Node.js?

Ejecute JavaScript en todas partes.

Node.js es un entorno de ejecución de JavaScript multiplataforma, de código abierto y gratuito que permite a los desarrolladores escribir herramientas de línea de comandos y scripts del lado del servidor fuera de un navegador.

## Qué es Express?

Es un Framework de Node.js o Java Script, que nos permite crear aplicaciones web del lado del servidor (Backend).

Es el Framework más popular de Node.js

Express es una herramienta de terceros, es código de otras personas. Muchas aplicaciones se complementan de módulos de terceros

Aquí en Express se utiliza Java Script puro.

npm es un manejador de módulos.

! Express es un módulo de npm !

node -v

npm -v

1-Creamos una carpeta que va a hacer el nombre del proyecto Ejemplo Apirestnodejsmysql

2-Entramos a Visual Studio Code y nos ubicamos en la ruta del proyecto ó vamos a la ruta del proyecto y abrimos una ventana de Windows PowerShell y allí digitamos code . y me abre automáticamente mi proyecto con Visual Studio Code

3-Abrimos la terminal de comandos con Ctrl + ñ ó por el menú de opciones Terminal - New Terminal

4- Ejecutamos el sgte comando npm init -y ó npm init , [Nos inicializa el proyecto para poder empezar a instalar las librerías, cualquiera de los dos es válido pero con el segundo te va a pedir por consola que le Agregues una Description y el Author.

Después de ejecutar el anterior comando se nos crea un archivo llamado package.json [Este archivo describe el proyecto] que contiene lo siguiente:

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

Por defecto se crea que la aplicación se inicia o ejecuta en el archivo index.js, pero se puede cambiar por el que se desee, preferiblemente dejarlo como index.js, que hay que crearlo para colocarle código.

5- Ahora instalamos las librerías que vamos a utilizar

Estas librerías se pueden ejecutar juntas simultáneamente separadas por espacio. Ejemplo: npm install express body-parser

El install se puede reemplazar por i

npm install express [Me crea una carpeta node_modules, la cual me permite que express funcione, también me crea una carpeta package-lock.json]

npm install body-parser [Para trabajar con peticiones post]

npm install sequelize

npm install mysql ó npm install mysql2 [Cuando se utiliza mysql2 es para utilizar la última versión, conecta el servidor con el banco de datos]

express-myconnection [Para la conexión de la base de datos]

npm install bcryptjs [Encripta el password de la base de datos]

npm install cors

npm install express-promise-router

npm install express-validator

npm moment

npm jwt-simple

npm install jsonwebtoken

npm install mongoose

npm install morgan

Cuando se crean o instalan las dependencias, automáticamente se actualiza el package.json, ejemplo instalé las siguientes dependencias y quedó así:


```
{
  "name": "apirestnodejsmysql",
  "version": "1.0.0",
  "description": "",
  "main": "server.js",
  "scripts": {
    "start": "nodemon server.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.17.1",
```

```
    "express-myconnection": "^1.0.4",
    "mysql": "^2.18.1"
  },
  "devDependencies": {
    "nodemon": "^2.0.7"
  }
}
```

6- También hay unas dependencias de desarrollo

@babel/cli

@babel/core

@babel/node

@babel/preset-env

nodemon [Esta es la más importante]

Al momento de terminar con estas líneas de las librerías hay que colocar -D [esto indica que son dependencias de desarrollo]

npm i nodemon --save-dev [Esta línea de comando también es válida para instalar nodemon]

[ --save-dev esta línea me permite decirle que es una librería de desarrollo, es decir que sólo la necesito mientras programo]

[Esta dependencia me permite que el servidor se reinicie automáticamente cada vez que yo hago un cambio en mi aplicación]

Cuando se instala nodemon, ir a package.json y cambiar por index.js ó por el nombre de inicio que se hay decidido.

```
"scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
```

Por

```
"scripts": {
   "start": "nodemon index.js"
 },
```

7- Luego que ya hayamos instalado las librerías o por lo menos las iniciales que necesitamos, porque podemos ir instalándolas a medida que las vamos requiriendo.

Creamos un archivo index.js, si no queremos que se llame así nuestro archivo de inicio, lo podemos cambiar, pero también hay que cambiarlo en package.json, que por defecto viene index.js. Ejemplo "main": "index.js"

Con nodemon para reiniciar el servidor se utiliza el siguiente comando: [ node index.js ]

npm start [Vuelve a reiniciar cada vez que se guarde algún archivo del proyecto]

npm run dev []

Escribimos el siguiente código en nuestro index.js

---

https://www.youtube.com/watch?v=T6rGUZGAWBk

```
const express =  require('express');
const bodyParser = require ('body-parser'); [La que me permite enviar objetos asociados a una petición post ]
const app =  express();
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true})); [Me codifica la url]
app.get('/', (req,res) =>{
   res.send('Hola Mundo');
});
app.listen(3000, () => {
   console.log('Servidor arrancado');
```

```
});
```

---

Ahora me voy a la terminal de comandos y ejecuto: node index.js

me debe mostrar en pantalla: [Servidor arrancado ] si no hay inconvenientes o errores

Luego verifico en el navegador, escribo localhost:3000 y me debe mostrar [Hola Mundo]

Otras formas de comenzar con el index.js seria:

---

https://www.youtube.com/watch?v=794Q71KVw1k

Este código es de la creación de un servidor con Node.js

```
const http = require('httpp');
const server =  http.createServer((req, res) =>{
    res.startus = 200;
    res.setHeader('Content-Type', 'text/plain');
    res.end('Hello world');
});

server.listen(3000, () => {
  console.log('Serve on Port 3000');
});
```

Me voy a la terminal de comandos y ejecuto:

node index.js

Y me debería mostrar CASO 2: Hello world

En el navegador localhost:3000

---

https://www.youtube.com/watch?v=yaMcqIeO9BU&list=PL-yog8huBN1jkfGnJ82HMTzDx9g7tOU0N

```
// const express =  require('express');
import express from 'express';
import morgan from 'morgan';

const app = express();
app.set('port', process.env.PORT || 3000);

app.listen (app.get('port'), ()=>{
    console.log('server on port: ', app.get('port'))
});
```

Me voy a la terminal de comandos y ejecuto:

```
node index.js
```

Y me debería mostrar CASO 3:  server on port: 3000

En el navegador localhost:3000

---

https://www.youtube.com/watch?v=OWukxSRtr-A

```
const express =  require('express');
const app =  express();
app.set('port', process.env.PORT || 9000)

app.get('/', (req,res) =>{
    res.send('Welcome to my API');
});
```

```
app.listen(app.get('port'), => {
    console.log('Server running on port', app.get('port'))
});
```

Me voy a la terminal de comandos y ejecuto:

```
node index.js
```

Y me debería mostrar CASO 3:  Server running on port: 9000

En el navegador localhost:9000

---