Vamos a crear una RestApi para **autos**.

**2122\_restapi\_autos**

# REST API Primeros pasos

// iniciar proyecto node.js

**npm init -y** // Se crea el package.json

// instalar typescript local si no se instaló global

*npm i typescript --save-dev* //Local

*npm i typescript -g* //Global

// Iniciar proyecto typescript

***tsc --init*** //Si global. Se crea el tsconfig.json

*npx tsc --init* //Si local. Se crea el tsconfig.json

// Cambiamos la configuración del ***tsconfig.json***

***"target": "es6",***

***"outDir": "./build",***

**Morgan** es una función de middleware para registrar información sobre la solicitud / respuesta http en una aplicación de servidor. Podemos ver las peticiones en consola.

// Un **middleware** es un bloque de código que se ejecuta entre

// la petición que hace el usuario (request) y la petición llega al servidor.

**CORS** significa Cross-Origin Resource Sharing, y es una política a nivel de navegador web que se aplica para prevenir que el dominio A evite acceder a recursos del dominio B usando solicitudes del tipo AJAX

**Nodemon** es una utilidad que monitorea los cambios en el código fuente que se esta desarrollando y automáticamente re inicia el servidor. Es una herramienta muy útil para desarrollo de aplicaciones en nodejs.

**Express** es una infraestructura de aplicaciones web Node.js mínima y flexible que proporciona un conjunto sólido de características para las aplicaciones web y móviles.

<https://expressjs.com/es/>

Instalamos los módulos como dependencias de producción:

***npm i express mongoose morgan cors***

Instalamos los módulos como dependencias de desarrollo:

***npm install @types/node @types/cors @types/express @types/morgan nodemon -D***

Después de estas instalaciones tendremos en el archivo package.json:

"dependencies": {

"cors": "^2.8.5",

"express": "^4.17.1",

"mongoose": "^6.0.12",

"morgan": "^1.10.0"

},

"devDependencies": {

"@types/cors": "^2.8.12",

"@types/express": "^4.17.13",

"@types/morgan": "^1.9.3",

"@types/node": "^16.11.7",

"nodemon": "^2.0.15"

}

// Cambiamos el *package.json* con:

"scripts": {

"ts": "tsc -w",

"dev": "nodemon ./build/server.js",

"start": "node ./build/server.js"

},

Creamos la carpeta src con el archivo **server.ts**

Crear la carpeta routes con el archivo **routes.ts**

Crear la carpeta model con el archivo **autos.ts**

Crear la carpeta database con el archivo **database.ts**

**Ver estos archivos en la aplicación**.

// Para compilar

*npm run ts* // que como tiene el -w incorporado se compilará si cambiamos el código

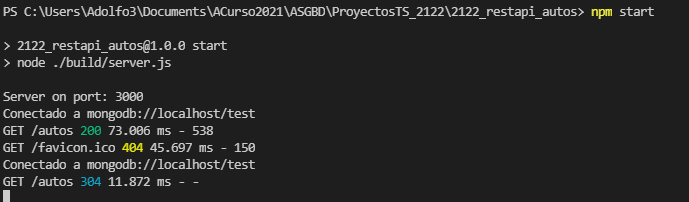
// Para ejecutar en desarrollo

*npm run dev* // que como tiene el nodemon se reiniciará el servidor si cambiamos

// además como tenemos

// Para ejecutar en producción

*npm start*

******

Y después de insertar autos con la aplicación:

**2122\_IntroduccionHerencia\_BDANDRES**

Tenemos:

