Para este proyecto vamos a necesitar tener instalado nodejs, typeorm,adicionalmente vamos a instalar express, tenemos que tener instalado typescrip, mysql, para manejar los roles vamos a utilizar jwt.

Primero vamos a instalar de manera global typeorm, para ello abrimos una terminal como administrador.

>>npm i typeorm -g

Con esta dependencia instalada se nos permite ejecutar un comando que nos crea la arquitectura inicial del proyecto node js.

>>typeorm init --name login\_roles --database mysql --express

Ahora vamos aingresar a la carpeta login\_roles

>>cd login\_roles

Vamos a abrir esta carpeta en visual studio code, podemos ver que nos crea los archivos necesarios para comenzar el proyecto, en tsconfig.json tenemos la configuración de typescrip, en el package.json tenemos las dependencias que vamos a necesitar instalar, el ormconfig es la configuración de nuestra base de datos.

Vamos a inicar el proyecto con las dependencias en el package.json

{

   "name": "login\_roles",

   "version": "0.0.1",

   "description": "Awesome project developed with TypeORM.",

   "devDependencies": {

      "ts-node": "3.3.0",

      "@types/node": "^8.0.29",

      "typescript": "3.3.3333"

   },

   "dependencies": {

      "typeorm": "0.2.37",

      "reflect-metadata": "^0.1.10",

      "mysql": "^2.14.1",

      "express": "^4.15.4",

      "body-parser": "^1.18.1"

   },

   "scripts": {

      "start": "ts-node src/index.ts"

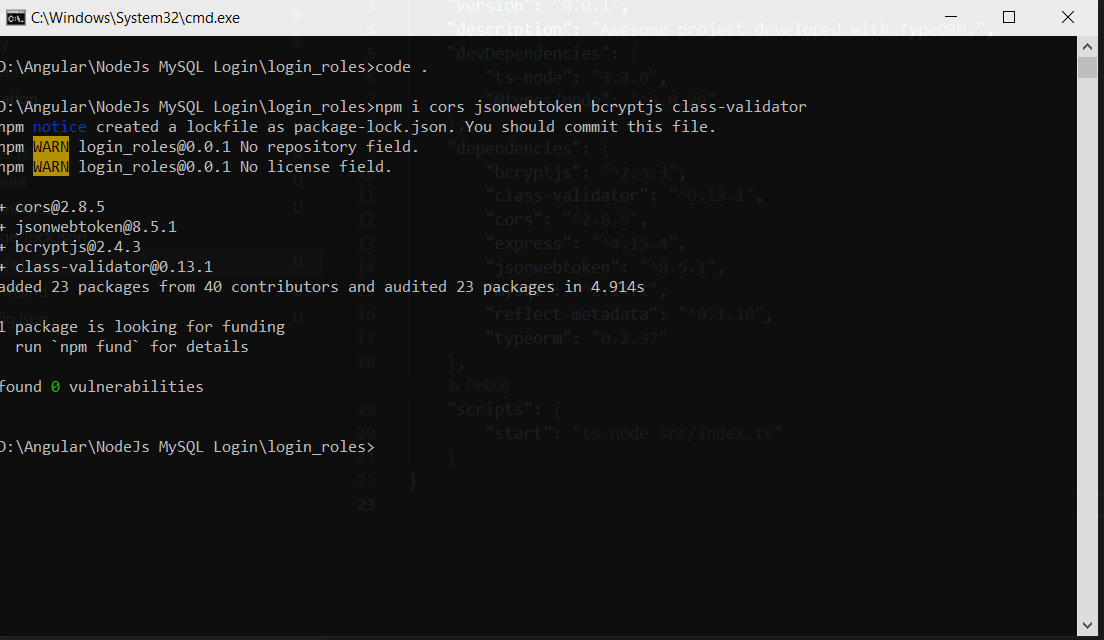
   }

}

Vamos a eliminar la dependencia de typescipt por que quiero utilizar una versión más reciente, también voy a eliminar body-parser esta dependencia lo que hace es que me permite comunicarme con el cliente, pero esto ya lo puedo hacer directamente con express.

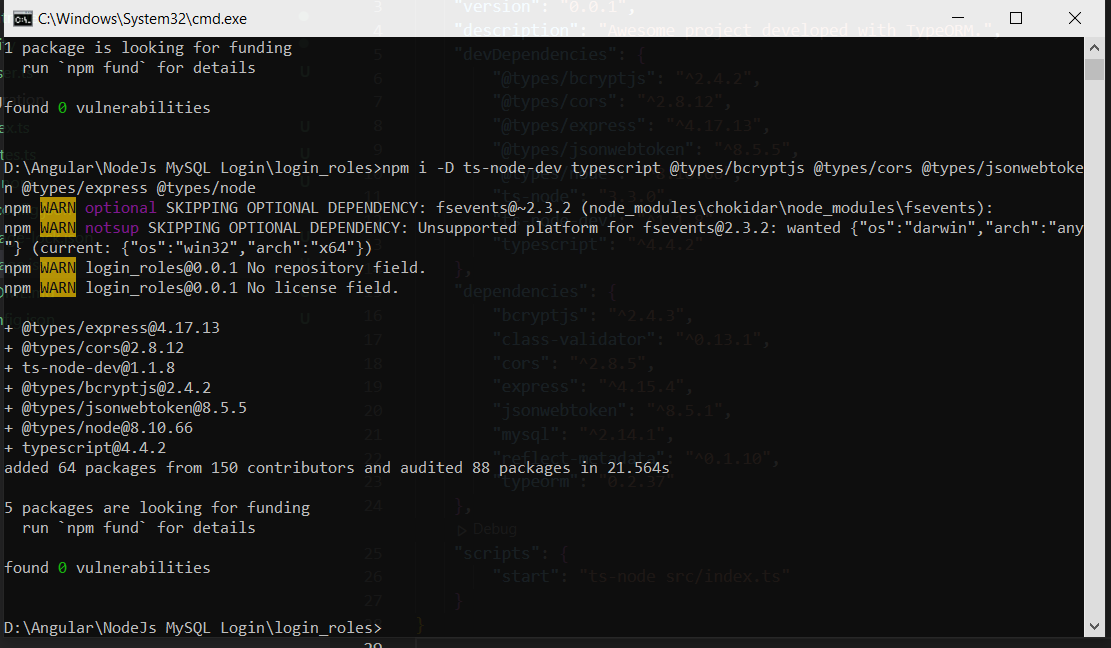
Ahora vamos a instalar las dependencias que nos hacen falta, vamos a instalar cors para poder aceptar todas las peticiones que nos hagan de un frontend; también necesitamos jsonwebtoken que nos va a permitir generar un token para nuestro usuario logeado, bcryptjs para poder encriptar la contraseña en la base de datos, una dependencia que se llama class-validator esta nos ayuda a hacer las validaciones de nuestros campos en la base de datos.

>>npm i cors jsonwebtoken bcryptjs class-validator



También vamos a necesitar otras dependencias pero estas estarán solo en modo de producción por que no queremos compilarlas con el resto del proyecto, vamos a instalar ts-node-dev que nos va a ayudar a reiniciar nuestro servidor, typescript, también vamos a necesitar instalar unos types, @types/bcryptjs, @types/cors, @types/jsonwebtoken, @types/express, @types/node

>>npm i -D ts-node-dev typescript @types/bcryptjs @types/cors @types/jsonwebtoken @types/express @types/node



En el archive de configuracion de typescript podemos ver las opciones, vamos a cambiar el nombre del directorio de compilación en vez de build a dist, vamos a agregar una nueva propiedad que se llama rootDir.

{

   "compilerOptions": {

      "lib": [

         "es5",

         "es6"

      ],

      "target": "es5",

      "module": "commonjs",

      "moduleResolution": "node",

      "outDir": "./dist",

      "rootDir": "./src",

      "emitDecoratorMetadata": true,

      "experimentalDecorators": true,

      "sourceMap": true

   }

}

Como podemos ver en el proyecto ya tenemos la arptea con el node\_modelu pero aun no tenemos todos los modulos instalados, para ello vamos a eliminar esta carptea y ejecutamos el comando npm i

>>npm i

Ahora vamos a configurar los scripts del package.json, primero voy a crear los scripts.

"scripts": {

        "tsc": "tsc",

        "dev": "set debug=\* && ts-node-dev --respawn --transpile-only ./src/index.ts"

    }

Ahora vamos a configurar el orm este nos permite tener incluso mas bases de datos configuradas, de tal manera de tener hasta microservicios corriendo en diferentes servidores.

{

    "type": "mysql",

    "host": "localhost",

    "port": 3306,

    "username": "root",

    "password": "admin",

    "database": "login\_node",

    "synchronize": true,

    "logging": false,

    "entities": [

        "src/entity/\*\*/\*.ts"

    ],

    "migrations": [

        "src/migration/\*\*/\*.ts"

    ],

    "subscribers": [

        "src/subscriber/\*\*/\*.ts"

    ],

    "cli": {

        "entitiesDir": "src/entity",

        "migrationsDir": "src/migration",

        "subscribersDir": "src/subscriber"

    }

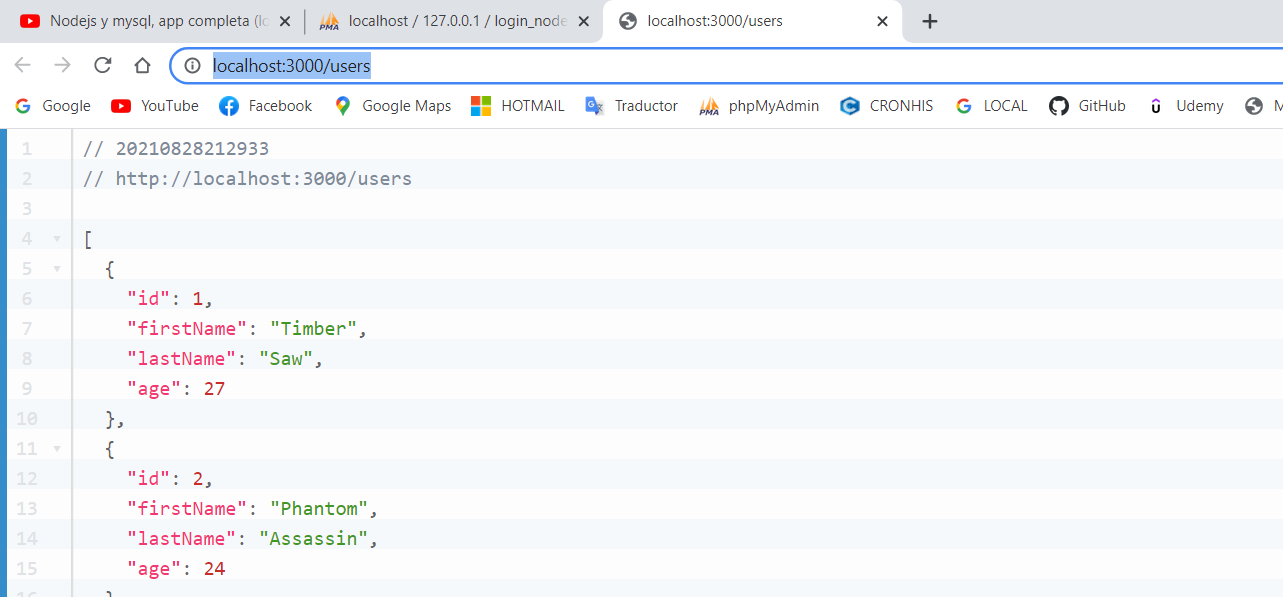
}

Una vez configurada nuestra conexión a la base de datos vamos a ejecutar el comando que creamos.

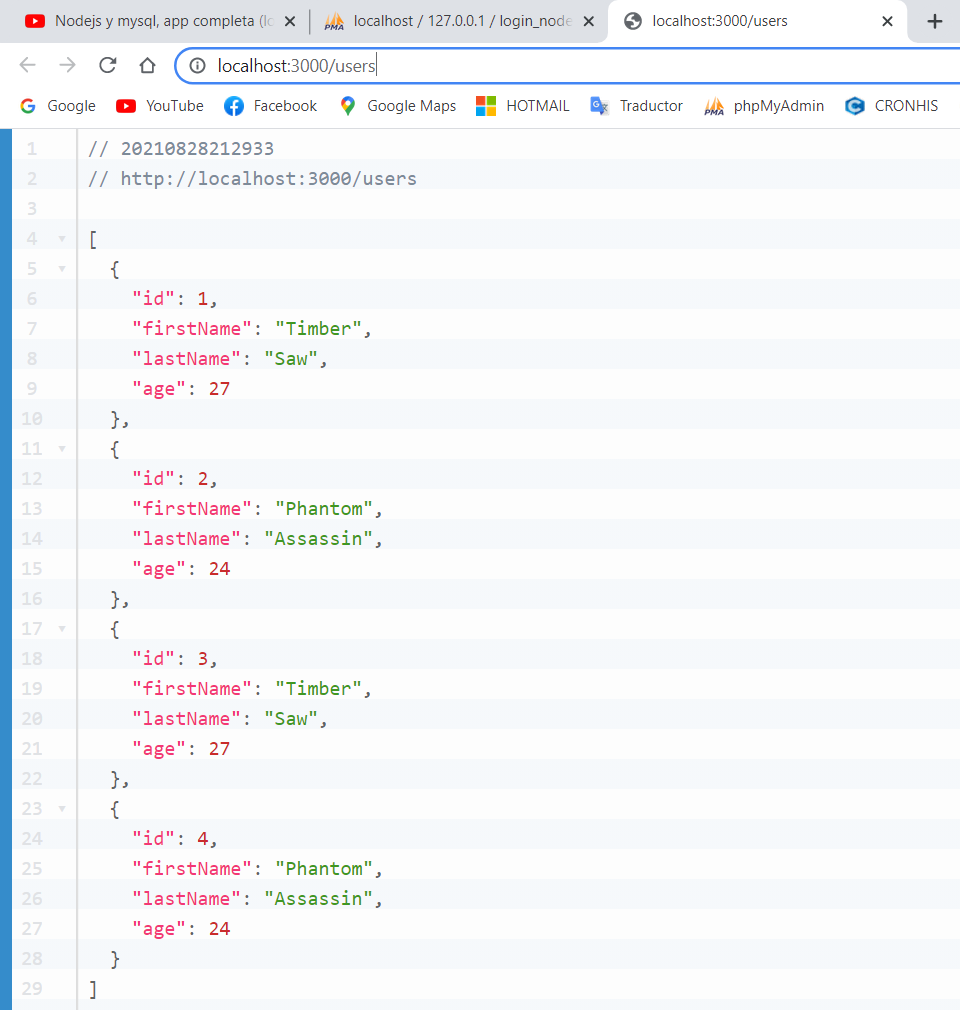
>>npm run dev

Ahora nos dice que si queremos ver el endpoint que ha creado vamos a la siguiente dirección,

<http://localhost:3000/users>



Podemos ver que nos crea los usuarios que está en nuestro código del index, si cancelamos el servicio y lo volvemos a arranar vemos que nos vuelve a crear los usuarios.



Hasta aquí vamos bien se han creado el servidor con las dependencias que necesitamos, vamos a configurar el archivo index.

Vamos a instalar un paquete qye hace que nuestra aplicación sea mas segura

>>npm i helmet

Los imports serian los siguientes

import "reflect-metadata";

import {createConnection} from "typeorm";

import \* as express from "express";

import {Request, Response} from "express";

import \* as cors from "cors";

import \* as helmet from "helmet";

creamos una constante para obtener el puerto si el proyecto esta en un servidor o lo dejamos en el puerto 3000 si estamos en desarrollo.

const PORT = process.env.PORT || 3000;

el método de conexión quedaría así.

createConnection().then(async () => {

    // create express app

    const app = express();

    // Middlewares

    app.use(cors());

    app.use(helmet());

    app.use(express.json());

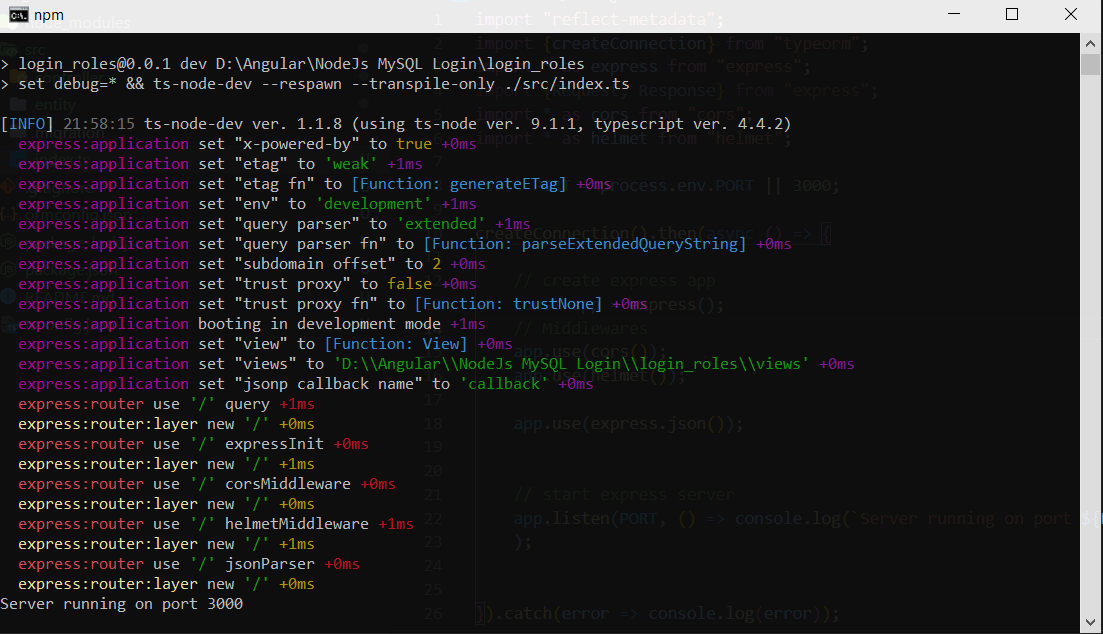
    // start express server

    app.listen(PORT, () => console.log(`Server running on port ${PORT}`)

    );

}).catch(error => console.log(error));

Si lo ejecutamos vemos que nos dice que el servidor esta corriendo en el puerto 3000



Vamos a crear una carpeta de rutas y dentro del tres archivos index.ts, auth.ts, user.ts

Ahora dentro de la carpeta controler vamos a crear el archivo AuthController.ts

Vamos a empezar a trabajar con el index.ts de las rutas.

1:16:00