**Nodejs, PostgreSQL & Typescript, REST API CRUD**

Pasos para Crear este Proyecto

M3-L1

1. Crear la carpeta del proyecto
2. Crear un README.md con las userstories en la raíz del proyecto
3. Crear el diagrama E/R, guardarlo como imagen y agregarlo al README.md

M3-L2

1. Crear una carpeta back y una front donde se guardarán todos los archivos del proyecto como dos aplicaciones independientes
2. Crear un archivo .gitignore que guarda carpetas y archivos que no se subirán a github  
    node\_modules/ para ignorar todas las carpetas node\_modules sin importar donde se encuentre  
    \*.env para ignorar todos los archivos que terminen en .env sin importar su nombre  
    dist/ para ignorar todas las carpetas dist sin importar donde se encuentre

BACK

1. Crear una carpeta src
2. Crear el archivo .env para guardar todas las variables de entorno del backend
3. Crear el archivo index.ts que va a contener la conexión a express  
   Por ahora contiene un console.log(“Hola Mundo!!! ”)
4. Ir a la carpeta back en la consola usando cd back
5. Incializar el back con npm init para crear el package.json
6. Instalar TS con npm install typescript -D o npm install --save-dev typescript las dos opciones crean dependencias de desarrollo en el package.json
7. Crear el archivo tsconfig.json con npx tsc --init
8. Configurar tsconfig.json

{

"compilerOptions": {

/\* Visit https://aka.ms/tsconfig to read more about this file \*/

/\* Language and Environment \*/

"target": "ES6" /\* Set the JavaScript language version for emitted JavaScript and include compatible library declarations. \*/,

/\* Modules \*/

"module": "commonjs" /\* Specify what module code is generated. \*/,

"rootDir": "./src" /\* Specify the root folder within your source files. \*/,

/\* Emit \*/

"outDir": "./dist" /\* Specify an output folder for all emitted files. \*/,

"removeComments": true,                           /\* Disable emitting comments. \*/

/\* Type Checking \*/

"strict": true /\* Enable all strict type-checking options. \*/,

},

"include": ["./src/\*\*/\*.ts"],

"exclude": ["node\_modules", ".env"]

}

1. Agregar al package.json  
   "scripts": {

"build": "tsc" para compilar ts a js

},

1. Agregar al package.json otro script

"start": "node ./dist/index.js" para ejecutar el index.js

1. Agregar al package.json otro script  
   "dev": "tsc && node ./dist/index.js" para compilar y ejecutar con un solo comando

M3-L3

1. Instalar dependencias

npm install --save-dev @types/express @types/node nodemon ts-node

npm install express morgan dotenv cors  
Para buscar documentación escribir npm librería en el buscador y nos lleva a la documentación correspondiente

1. Crear archivo nodemon.json

{

"watch": ["src"],

"ext": "ts",

"exec": "ts-node ./src/index.ts"

}

1. Ajustar package.json  
   "scripts": {

"build": "tsc",

"start": "node ./dist/index.js",

"build:start": "tsc && node ./dist/index.js",

"dev": "nodemon"

},

1. Crear las carpetas necesarias para el backend del proyecto  
   config, controllers, dtos, entities/schemas, interfaces, middlwares, routes, services
2. Agregar las variables de entorno en el archivo .env  
   PORT=3000
3. Crear un archivo de configuración  
   envs.ts que contiene la llamada a dotenv para exportar las variabes de entorno en cualquier lugar donde las necesite  
   import "dotenv/config";

export const PORT = process.env.PORT;

1. Crear archive server.ts  
   import cors from "cors";

import express from "express";

import morgan from "morgan";  
const server = express();

server.use(cors()); // desde que métodos voy a dejar que se ejecuten, desde qué url, qué header, así es para todo tipo de peticiones

server.use(morgan("dev")); // dev = desarrollo

server.use(express.json()); // para que el servidor entienda los json que le llegan por body

server.get("/", (req, res) => {

res.send("Hola Mundo!")

})

export default server;

Instalar por consola  
npm install --save-dev @types/cors @types/Morgan

1. Llamar al servidor desde index.ts  
   import { PORT } from "./config/envs";

import server from "./server";

server.listen(PORT, () => {

console.log(`Server listening on http://localhost:${PORT}`);

});

1. Crear los controladores para users y appointments
2. Crear los routers para users y appointments y el router principal o indexRouter.ts
3. Actualizar el archivo server.ts
4. Crear las interfaces IUser, IAppointment, ICredential
5. Crear los servicios usersService, appointmentsService, credentialsService
6. Crear los dtos userDto, appointmentDto, credentialDto