**NestJs Fundamentals**

**Habilidades y competencias que desarrollarás...**

* Inicialización de proyectos en node.
* Configuración básica de un proyecto.
* **ACTIVIDAD 01**
* Crear un proyecto en Nest JS bajo el nombre ecommerce-<usuario de github>.
* **ACTIVIDAD 02**
* Crear los módulos Products, Users y Auth.
* **ACTIVIDAD 03**
* Crear sus respectivos controllers y services.
* **ACTIVIDAD 04**
* Crear los endpoints GET /products, GET /users y Get /auth.
* **ACTIVIDAD 05**
* Crear un middleware global que loguee la ruta, método y la fecha-hora en que se llamó al endpoint.

**¡Bien hecho!**

**TIPS**

* Utiliza Nest CLI para inicializar el proyecto.
* Recuerda “modularizar” el código para trabajar de forma ordenada.

**NestJS Fundamentals 2**

**INSTRUCCIONES**

**Habilidades y competencias que desarrollarás...**

* Creación y uso de repositorios de NestJS.
* **ACTIVIDAD 01**
* Crear los repositorios para Users y Products.
* **ACTIVIDAD 02**
* Guardar las entidades en un array en memoria.
* **ACTIVIDAD 03**
* Cargar algunas entidades de prueba *hardcodeadas*, con las siguientes propiedades...
* **Users**
* id:number
* email: string
* name: string
* password: string
* address: string
* phone: string
* country?: string | undefined
* city?: string | undefined
* **Products**
* id:number
* name: string
* description: string
* price: number
* stock: boolean
* imgUrl: string
* **ACTIVIDAD 04**
* Modificar los endpoints GET /products y GET /users para que devuelvan el array de entidades.

**¡Bien hecho!**

**TIPS**

* Recuerda que los controllers sólo pueden comunicarse con el repositorio a través de los servicios.
* No olvides actualizar el array de providers .

**NestJS Routing**

**INSTRUCCIONES**

**Habilidades y competencias que desarrollarás...**

* Creación de rutas para el procesamiento de solicitudes correspondientes al CRUD de cada entidad.
* Validación del cuerpo de las solicitudes.
* Implementar la paginación de resultados de acuerdo a la información recibida en la ruta.
* Uso de guardias para la protección de rutas.
* **ACTIVIDAD 01**
* Crear todos los endpoints CRUD para Products y Users (GET, GET{id}, POST, PUT{id}, DELETE{id}).

**ACTIVIDAD 02**

Desarrollar la lógica de creación, listado y eliminación desde el repository.

* GET debe devolver la lista de elementos , y httpStatus = 200.
* GET{id} debe devolver el elemento con id pedido, y httpStatus = 200.
* En el caso de Users, no devolver el password.
* POST debe devolver el id de la entidad creada, y httpStatus = 201.
* PUT{id} y DELETE{id} pueden devolver el id de la entidad editada/eliminada y httpStatus 200.
* **ACTIVIDAD 03**
* Validar en POST y PUT que la estructura de la entidad corresponda a la estructura de cada entidad.

**ACTIVIDAD 04**

El método GET puede recibir como query params los valores page y limit.

* Si no recibe el parámetro page, el valor por defecto es 1.
* Si no recibe el parámetro limit, el valor por defecto es 5.
* Bonus: Implementar la lógica desde el repositorio, para paginar las entradas devuelvas.

**ACTIVIDAD 05**

Crear el endpoint POST /auth/signin, que reciba email y password.

* Para el login se utilizarán las credenciales email / password.
* Inyectar el usersRepository para poder hacer consultas.
* No se procederá al login si faltan alguna de las dos credenciales.
* No se procederá con el login en caso de que no exista un usuario registrado con la dirección de email proporcionada.
* En caso de que el usuario no exista o la contraseña proporcionada no coincida con la registrada, se deberá enviar una única respuesta para cualquiera de los casos. Ej: “Email o password incorrectos”. NOTA: Por seguridad es preferible no especificar cuál de los dos datos ha fallado en su verificación.

**ACTIVIDAD 06**

Dentro de la carpeta Auth, crear una guarda AuthGuard, que debe verificar lo siguiente...

* Debe existir un header Authorization.
* Dicho header, tiene que tener una estructura como la siguiente: Basic: <email>:<password>.
* NO validaremos por ahora que sea un email y un password válido, únicamente verificar si el header es enviado y continente un email y un password.
* Todos los endpoints de Users, salvo el POST, deben utilizar esta guarda.
* Todos los endpoints de Products, salvo el GET y el GET{id} deben utilizar esta guarda.

**¡Bien hecho!**

**TIPS**

* No te preocupes por ahora por el manejo de errores, la ruta de autenticación puede devolver strings únicamente.