Prueba Técnica – Explicación de herramientas, configuraciones y módulos desarrollados

Repositorio del Proyecto

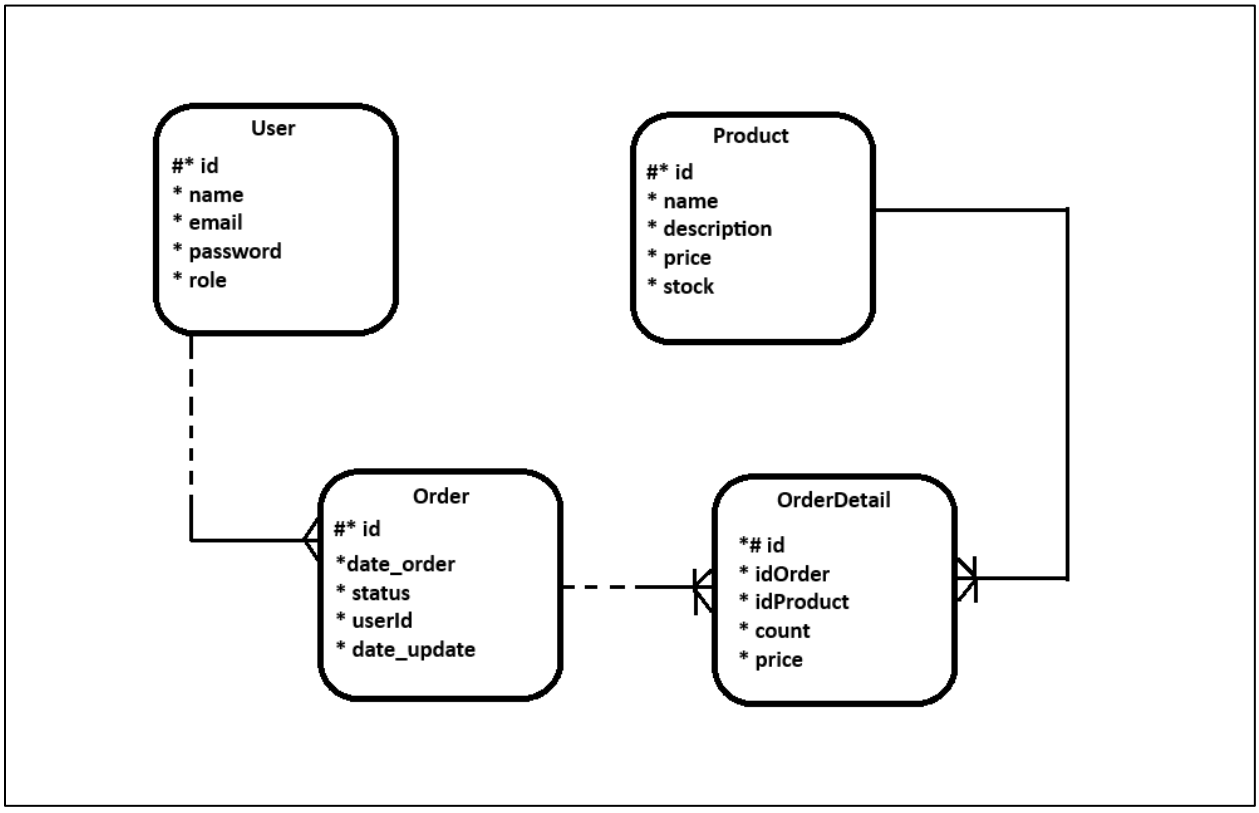
Para el proyecto desarrollado fue generado un repositorio público en *Github* donde se puede apreciar el código fuente, y una descripción de los pasos para la instalación del mismo.

En el siguiente enlace se puede acceder al repositorio: https://github.com/Joseeli54/products-express-ts/

Documentación del API

Se utilizó la herramienta **Swagger UI** para el proceso de documentación del API. Con esta interfaz se puede observar los *endpoints* generados para la gestión de los usuarios, productos y los pedidos realizados por los usuarios autenticados del sistema y facilita el entendimiento de los datos de solicitud y respuesta. Una vez que el proyecto sea ejecutado en la maquina local, se podrá ingresar a la interfaz ingresando a la ruta: *http://localhost:3000/docs/*

Diseño de la base de datos



Se aplicó un diseño simple, donde los usuarios tuvieran un nombre, un correo electrónico y una contraseña para autenticarse en el API.

User: Los usuarios podrán realizar peticiones al API, una vez que se encuentren autenticados, y tengan su *token*.

Product: Los productos que serán obtenidos por los usuarios, de modo que se pueda saber su nombre, descripción, precio y stock. De esta forma, también se añadió un parámetro adicional que indica cuando el producto está disponible o no, dependiendo de si el stock se encuentra en 0, o hay productos disponibles.

Order: Son aquellos pedidos que un usuario realizar a una cantidad de productos existentes, de modo que se pueda pasar por un proceso de envió, procesamiento y completación del encargo. Un usuario pueda realizar muchos pedidos, y un pedido solamente estará relacionado con un usuario.

OrderDetail: Esta es una entidad proveniente de una relación de muchos a muchos de las ordenes con los productos. Esto se debe a que un producto puede pertenecer a varias órdenes, y una orden puede involucrar a varios productos.

Manejador de base de datos

Se seleccionó el manejador de base de datos ***PostgreSQL***, en conjunto con **Prisma ORM**, debido a su capacidad de manejar las relaciones especificadas en las instrucciones, y adicional a eso, cumple con los requisitos de garantizar la integridad de los datos.

El proyecto no requiere de momento un manejador para datos no estructurados, de escalabilidad horizontal o mayor flexibilidad al momento de generar esquemas. Adicional a eso, una base de datos como ***MongoDB*** puede requerir más recursos en comparación con ***PostgreSQL***, debido a su naturaleza y complejidad, y, por lo tanto, la opción adecuada para el proyecto actual, sería una base de datos relacional.

El diseño de las tablas y sus relaciones es adecuado para una base de datos relacional con escalabilidad vertical, y **PostgreSQL** se integra muy bien con **Prisma ORM**, para facilitar la gestión de la base de datos en ***Express*** y ***Node.js***.

Herramientas o librerías utilizadas

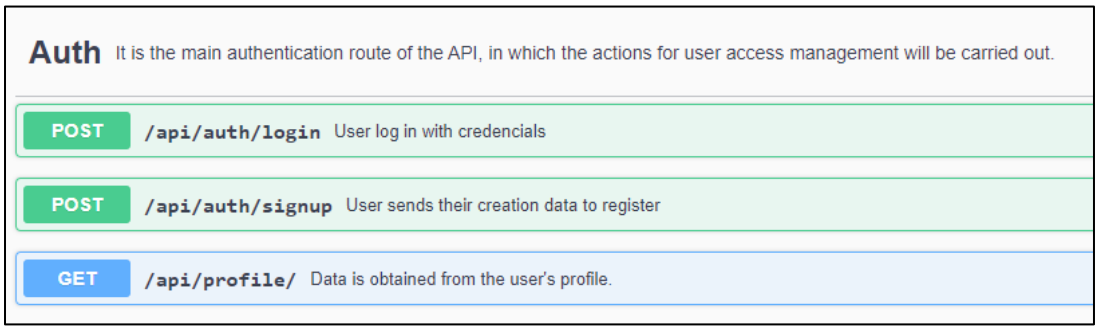
Herramienta	Razón
Prisma ORM	Gestión y buen manejo de la base de datos.
Jest	Realización de pruebas en el código fuente, como las pruebas unitarias.
Zod	Librería utilizada para la validación de los datos. Esta devuelve un error cuando algún formato no es correcto.
Nodemailer	Aplicado para el envío de correos electrónicos.
Swagger	Documentación del API desarrollado
jsonwebtoken	Generación del token JWT para la autenticación de usuario.

Estructura del Proyecto

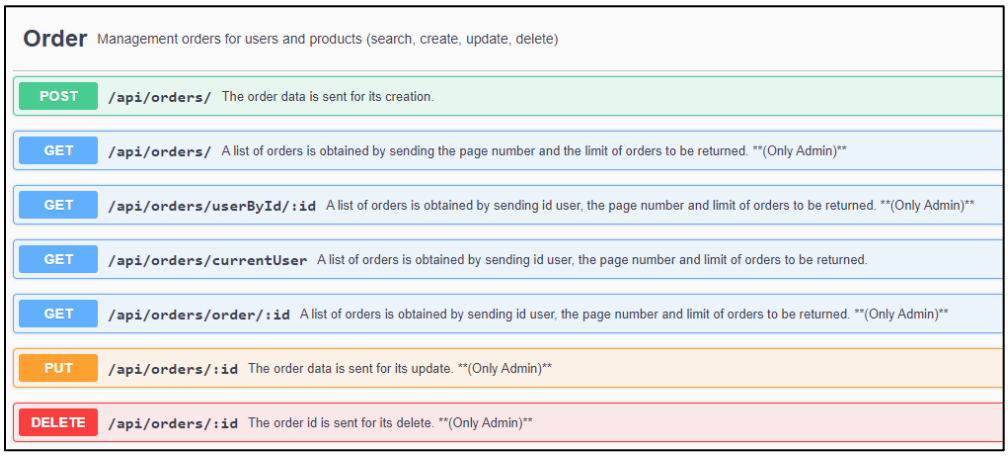
El proyecto es desarrollado bajo una estructura dividida en módulos, para separar los controladores, servicios, enrutamiento, así como los modelos y los módulos para la gestión de la información, que serán descritos a continuación:

- **Controllers:** El módulo de los controladores se agrega a la estructura para llevar a cabo la gestión de las peticiones realizadas al API, segregándolo en cuatro módulos diferentes de acuerdo a las necesidades del negocio y las entidades existentes en el sistema.
    - **Auth:** Autenticación de usuario.
    - **Order:** Gestión de pedidos, junto con su detalle. (Solo los administradores podrán modificar o eliminar la información)
    - **Product:** Gestión de los productos disponibles. (Solo los administradores podrán modificar o eliminar la información)
    - **User:** Gestión de usuario (Exclusivamente para administradores)
- **Exception:** El módulo de excepciones se estableció para llevar a cabo la gestión de los errores en el sistema, como, por ejemplo, el acceso no autorizado de usuarios en alguno de los *endpoints*.
    - **Bad-Request:** Datos enviados de forma incorrecta.
    - **Root:** Configuración generar de la clase de errores, con sus códigos de error.
    - **Internal-Exception:** Error disparado cuando hay un problema del lado del servidor.
    - **Unauthorized:** Error disparado cuando un usuario intenta realizar una petición sin estar autenticado o sin tener permiso.
    - **Validation:** Validación de los datos enviados para cualquier entidad. Si hay algún error, esta excepción es disparada.
    - **Not-Found:** No se encuentra el registro indicado en las peticiones.
- **Interfaces:** El módulo fue realizado para establecer una estructura en los datos que se deben cumplir, así como las respuestas del servidor a una petición *HTTP*.
    - **Results:** Este es una interface, que servirá como modelo para las respuestas del servidor. De este modo se configuró un formato, donde se indiquen los siguientes parámetros: satisfactorio (verdadero o falso), mensaje, data. Si hay algún error al momento de ejecutar las funciones, se indicará el tipo de error, la descripción del mismo y el código del error.
    - **DTO:** Los **Dto**, se configuraron igualmente para establecer los atributos de los datos enviados, de modo que se cumpla el formato de las tablas.
- **Mailer:** Módulo generado para llevar a cabo la gestión de correos electrónicos, de modo que sea enviado algún correo cuando se ejecute alguna acción. Actualmente solamente se encuentra configurado cuando un usuario es creado en el servidor.
- **Middlewares**: Este módulo era necesario para el API, debido a que actúa como un puente donde se verifican los accesos que puede tener una petición al momento de ser enviada. Para este proyecto solamente se configuraron 3 middlewares, que son los siguientes:
    - **Auth:** Este archivo se encarga de verificar que los usuarios que envían las peticiones estén autenticados en el servidor. Si el token enviado expiró o no es correcto, entonces se regresa un mensaje de error 401 (Unauthorized).
    - **Admin:** Este archivo se encarga de verificar si los usuarios autenticados son administradores. La función se utiliza únicamente para los *endpoints* con información sensible en el servidor.

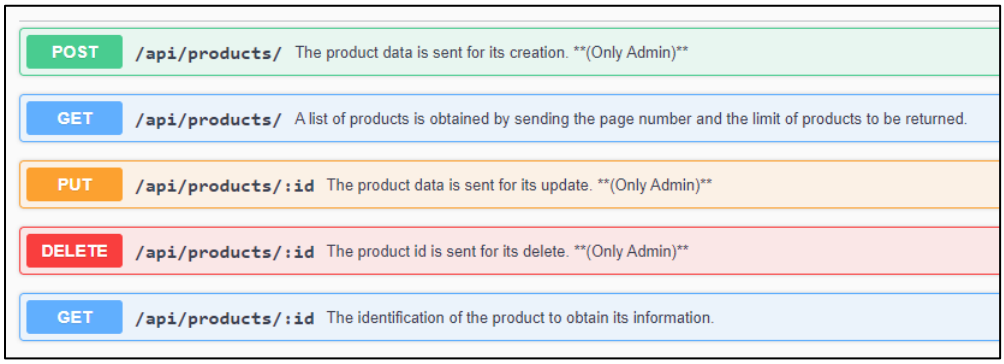
    - **Errors:** Este archivo se encarga de enviar los mensajes de error disparados por las excepciones en el sistema.
- **Models:** Otro módulo necesario para el código fuente, debido a que es utilizado en los repositorios para realizar las acciones de inserción, actualización, búsqueda o eliminación de los datos.
    - **Order-detail:** Modelo con los atributos de la entidad OrderDetail.
    - **Order:** Modelo con los atributos de la entidad Order.
    - **Product:** Modelo con los atributos de la entidad Product.
    - **User:** Modelo con los atributos de la entidad User.
- **Repositories:** Este módulo funciona para conectar con la base de datos del sistema, utilizando la herramienta Prisma, para hacer los procesos de búsqueda, inserción, modificación y eliminación. Estos son utilizados por los servicios para hacer la gestión de los datos. Contienen funciones que permiten facilitar los procesos de consultas a la base de datos.
    - **Orders:** Contiene un conjunto de funciones para consultas a la tabla Order u OrderDetail.
    - **Products:** Contiene un conjunto de funciones para consultas a la tabla Product.
    - **Users:** Contiene un conjunto de funciones para consultas a la tabla User.
- **Routes:** Este módulo también fue necesario para llevar el control de las peticiones HTTP, de modo que exista un orden de *endpoints* dividida por la tabla o entidad que se va a gestionar.
    - **Index:** Este es el archivo principal de las rutas
    - **Auth:** Rutas para el proceso de autenticación de usuario y registro del mismo.



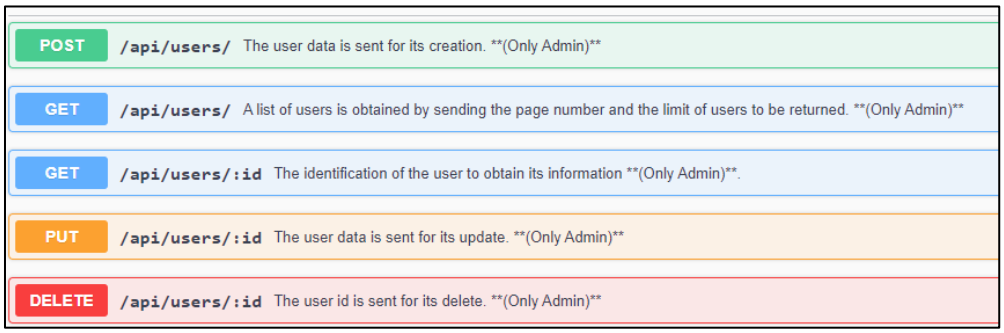
    - **Orders:** Rutas para el proceso de gestión de los pedidos a productos.



- **Products:** Rutas para el proceso de gestión de los productos.



- **Users:** Rutas para el proceso de gestión de los usuarios.



- **Schema:** Módulo encargado de llevar a cabo los esquemas que deben tener los atributos de cada entidad. Este es utilizado para validar el correcto formato de los datos enviados y su completitud.
  - **Login:** Esquemas para inicio de sesión y creación de cuenta.
  - **Orders:** Esquemas para los pedidos.
  - **Products:** Esquemas para los productos.
  - **Users:** Esquemas para los usuarios.
- **Services:** Módulo de los servicios que son llamados por los controladores para el procesamiento de los datos recibidos, y adicional a eso, también devolver una respuesta. Estos utilizan tanto repositorios, como esquemas para la gestión y validación de los datos. Una vez que se finalice el proceso, devuelven una respuesta a los controladores.
  - **Auth:** Servicios para autenticación de usuario.
    - **Login.service:** Inicio de sesión del usuario
    - **Signup.service:** Registro del usuario.
  - **Orders.service:** Gestión de los pedidos de usuario, así como el proceso de paginación de los mismos. (CRUD)
  - **Products.service:** Gestión de los productos, así como el proceso de paginación de los mismos. (CRUD)
  - **Users.service:** Gestión de los usuarios, así como el proceso de paginación de los mismos. (CRUD)