CHALLENGE N1U

Sprint

Aa Name	Assign	Status
Registrar un nuevo producto	Ⓐ Abigail Justiniano	Done
Actualizar los datos de un producto	Ⓐ Abigail Justiniano	Done
Eliminar Producto	Ⓐ Abigail Justiniano	Done
Listar los datos de un producto específico	Ⓐ Abigail Justiniano	Done
Listar todos los productos	Ⓐ Abigail Justiniano	Done
Listar todos los restaurantes	Ⓐ Abigail Justiniano	Done
Registrar un nuevo restaurante	Ⓐ Abigail Justiniano	Done
Listar los datos de un restaurante específico	Ⓐ Abigail Justiniano	Done
Eliminar un restaurante	Ⓐ Abigail Justiniano	Done
Actualizar los datos de un restaurante	Ⓐ Abigail Justiniano	Done

Documentación

Modelo: Se utilizó el patrón de diseño MVC (Modelos, Vistas, Controladores) para separar la lógica de la aplicación en tres capas:

- Modelo: Representa la estructura de las entidades (restaurantes y productos) y su lógica de negocio.
- Servicio: Se encarga de la lógica de negocio específica para cada endpoint (CRUD para restaurantes y productos).
- Controller: Intermedia entre el cliente y el modelo, procesando las solicitudes del cliente y enviando los datos al modelo. Define las rutas y procesa las solicitudes HTTP.

Tecnologías:

- NodeJS: Entorno de ejecución en tiempo real para JavaScript.
- Mongo: Base de datos NoSQL document-oriented.
- Nodemon: Herramienta que reinicia automáticamente el servidor cada vez que se modifica un archivo.
- Multer: Middleware para la gestión de archivos subidos.
- Express: Framework web minimalista y flexible.
- BodyParser: Middleware para analizar el cuerpo de las solicitudes HTTP.

Infraestructura

- **Docker:** Plataforma para la creación, gestión y distribución de aplicaciones.
- **Docker-compose:** Herramienta para definir y ejecutar aplicaciones compuestas por varios contenedores.
- **Dockerfile:** Archivo que define la configuración de un contenedor.

Configuración:

- Se configuró la aplicación para que se ejecute en dos contenedores: uno para la web y otro para la base de datos.
- La web se hostea de manera local en el puerto 8080.
- La base de datos se hostea de manera local en el puerto 27017.

Beneficios de usar Docker:

- Evitar la descarga de NodeJS: Docker te permite definir la versión específica de NodeJS que se necesita para ejecutar la aplicación. Esto evita la necesidad de descargar e instalar NodeJS manualmente en cada máquina.
- Eficiencia en las pruebas: Puedes crear diferentes contenedores para diferentes escenarios de prueba, lo que te permite probar la aplicación de forma más eficiente.

Imagen de Restaurante y Producto

Formato:

- Se admiten formatos PNG y JPG.
- Se recomienda un tamaño máximo de 2 MB.
- La imagen debe ser de alta calidad y representar al restaurante o producto de forma atractiva.

Subida:

- La imagen se ingresa como un archivo mediante el campo `photo` en el body de la solicitud.
- Se utiliza la biblioteca Multer para la gestión de archivos subidos.

Almacenamiento:

- La imagen se almacena en el servidor en la ruta /uploads.
- Se utiliza un nombre de archivo único para evitar conflictos.

Seguridad:
Se implementan medidas de seguridad para evitar la subida de archivos maliciosos.

Tecnologías adicionales:

- **Postman:** Herramienta para la gestión de APIs.
- **Mongoose:** ODM (Object Document Mapper) para MongoDB.

Recursos:

- Repositorio del proyecto: `https://github.com/debora2004/challenge-justiniano.git`
- Documentación de NodeJS: `https://nodejs.org/en/docs/`
- Documentación de Mongo: `https://docs.mongodb.com/`
- Documentación de Docker: `https://docs.docker.com/`
- Documentación de Multer: `https://github.com/expressjs/multer`

Mejoras para la Aplicación

- **Implementar un sistema de compresión de imágenes** para reducir el tamaño de los archivos.
- **Implementar Tests automáticos.**
- **Permitir la subida de imágenes** desde una URL.
- **Implementar un sistema de eliminación de imágenes** para evitar el almacenamiento innecesario de archivos.
- **Gestión de usuarios:** Implementar un sistema de gestión de usuarios para permitir el registro, la autenticación y la autorización de usuarios. Esto permitirá un mejor control de acceso a la aplicación y sus funcionalidades.
- **Gestión de pedidos:** Implementar un sistema de gestión de pedidos para permitir a los usuarios realizar pedidos de productos. Esto permitirá una experiencia de compra completa dentro de la aplicación.
- **Paginación:** Implementar la paginación para la visualización de datos, especialmente para listados extensos como la lista de productos. Esto facilitará la navegación y el rendimiento.
- **Filtros y búsquedas:** Implementar filtros y búsquedas para que los usuarios puedan encontrar fácilmente la información que necesitan. Esto mejorará la usabilidad de la aplicación.
- **Validación de datos:** Implementar una validación robusta de datos para asegurar la integridad de la información ingresada por los usuarios. Esto evitará errores y problemas en la aplicación.
- **AWS:** Implementar AWS para poder guardar los datos en la nube de la aplicación.
- **Firebase:** Implementar Firebase para la autenticación de usuarios, almacenamiento de datos en la nube y notificaciones push.