Objetivo del ejercicio:

Crear una API RESTful que gestione una base de datos de productos y categorías usando MongoDB y Mongoose.

Requisitos:

- Node.js y Express.
- Mongoose para interactuar con MongoDB.
- Uso de rutas HTTP (GET, POST, PUT, DELETE).
- Validación de datos con Mongoose.
- Relación de uno a muchos entre productos y categorías.
- Manejo de errores adecuados (por ejemplo, 404 si no se encuentra un recurso, 400 si hay un error de validación).
- Middleware para validación de datos.
- Uso de variables de entorno para la configuración (puerto y URI de MongoDB).

Parte 1: Configuración inicial

1. **Instalación de dependencias:**
    a. Inicializa un proyecto de Node.js con npm init.
    b. Instala las dependencias necesarias:
2. **Estructura básica del proyecto:** Crea una estructura de carpetas como esta:

/src

├— /models

├— /routes

├— /controllers

├— /middlewares

├— app.ts

3. **Configuración de Express y conexión a MongoDB**

Parte 2: Definir los modelos

1. **Modelo de Categoría:**
   Crea un modelo de Mongoose para las categorías de productos en models/Category.js:

2. **Modelo de Producto:** Crea un modelo de Mongoose para los productos en models/Product.js, relacionando el producto con una categoría

Modelo Producto (Product)

1. **name** (Nombre del producto)
    a. **Tipo**: String
    b. **Requerido**: true
    c. **Descripción**: El nombre del producto. Debe ser único para evitar duplicados.
    d. **Ejemplo**: "Camiseta Roja"
2. **price** (Precio del producto)
    a. **Tipo**: Number
    b. **Requerido**: true
    c. **Descripción**: El precio del producto en formato numérico. Debe ser un número positivo.
    d. **Ejemplo**: 25.99
3. **category** (Categoría del producto)
    a. **Tipo**: ObjectId (Referencia)
    b. **Requerido**: true
    c. **Descripción**: Relación con la colección de Category. Este campo debería ser una referencia al modelo Category (relación de muchos a uno).
    d. **Ejemplo**: ObjectId("5f8d0d55b54764421b7156f3") (Referencia a una categoría de MongoDB)
4. **description** (Descripción del producto)
    a. **Tipo**: String
    b. **Requerido**: false
    c. **Descripción**: Descripción opcional del producto.
    d. **Ejemplo**: "Camiseta de algodón, color rojo, talla M."
5. **inStock** (Disponibilidad)
    a. **Tipo**: Boolean
    b. **Valor por defecto**: true
    c. **Descripción**: Indica si el producto está disponible en stock o no.
    d. **Ejemplo**: true (disponible), false (no disponible)
6. **createdAt** (Fecha de creación)
    a. **Tipo**: Date
    b. **Valor por defecto**: Date.now
    c. **Descripción**: Fecha en la que se creó el producto.
    d. **Ejemplo**: 2023-10-01T14:48:00.000Z

Modelo Categoría (Category)

1. **name** (Nombre de la categoría)
    a. **Tipo**: String
    b. **Requerido**: true
    c. **Descripción**: El nombre de la categoría. Debe ser único.
    d. **Ejemplo**: "Ropa"
2. **description** (Descripción de la categoría)
    a. **Tipo**: String
    b. **Requerido**: true
    c. **Descripción**: Una breve descripción de la categoría.
    d. **Ejemplo**: "Categoría de ropa para hombres y mujeres."
3. **createdAt** (Fecha de creación)
    a. **Tipo**: Date
    b. **Valor por defecto**: Date.now
    c. **Descripción**: Fecha en la que se creó la categoría.
    d. **Ejemplo**: 2023-10-01T14:48:00.000Z

Parte 3: Rutas y controladores

1. **Rutas para Categorías:** Crea las rutas para gestionar categorías
2. **Rutas para Producto**

Parte 4: Agregar Middleware y Validaciones (30 minutos)

1. **Middleware para Validación:**
   Crea un middleware para validar los datos de entrada (por ejemplo, validar que el precio sea positivo) en middlewares/validateProduct.ts

Parte 5: Pruebas y Despliegue

1. **Pruebas usando Postman o Insomnia:**
   Pide a los estudiantes que prueben la API usando herramientas como Postman o Insomnia. Los puntos a probar son:
    a. Crear y obtener categorías.
    b. Crear y obtener productos.
    c. Validar que los errores se gestionan correctamente (por ejemplo, no se puede crear un producto sin una categoría válida).