CHRISTOPHER ARIAS
MASTER

ESTRUCTURACION ARQUITECTURA BACKEND

para una aplicación de comercio electrónico, es fundamental tener en cuenta aspectos como escalabilidad, seguridad, mantenibilidad y rendimiento.

TECNOLOGIAS

- Lenguaje de Programación: JavaScript/TypeScript con Node.js.
- Framework: Express.js para la creación de API REST.
- Base de Datos: MongoDB para una base de datos NoSQL o PostgreSQL para una base de datos SQL.
- Autenticación: JSON Web Tokens (JWT) para autenticación y autorización.
- ORM/ODM: Mongoose para MongoDB o Sequelize para PostgreSQL.
- Testing: Jest para pruebas unitarias y de integración.
- Docker: Para contenerización y facilitar la implementación y el desarrollo.
- CI/CD: GitHub Actions para automatizar despliegues y pruebas.

PARA LA FORMA EN QUE SE ESTRUCTURARÍA PODRÍA QUEDAR ASÍ:

```
/ecommerce-backend
|
├── /src
|   ├── /config          # Configuraciones generales del proyecto
|   ├── /controllers     # Controladores para manejar la lógica de negocio
|   ├── /models          # Modelos de la base de datos
|   ├── /routes          # Definición de las rutas de la API
|   ├── /middlewares     # Middlewares para el manejo de peticiones
|   ├── /services        # Servicios para encapsular la lógica de negocio
|   ├── /utils           # Utilidades y funciones auxiliares
|   ├── /tests           # Pruebas unitarias y de integración
|   └── index.js         # Punto de entrada de la aplicación
|
├── .env                 # Variables de entorno
├── .gitignore           # Archivos y directorios a ignorar por Git
├── docker-compose.yml   # Configuración de Docker Compose
├── Dockerfile           # Configuración de Docker
├── package.json         # Dependencias y scripts del proyecto
└── README.md            # Documentación del proyecto
```

ENDPOINTS Y FUNCIONALIDADES BÁSICAS

- Usuarios: Registro, inicio de sesión, perfil, actualización de datos.
- Productos: CRUD (Crear, Leer, Actualizar, Eliminar) de productos, búsqueda y filtrado.
- Pagos: Integración con pasarelas de pago como Stripe o PayPal.
- Carrito de Compras: Añadir, eliminar productos, actualizar cantidades.
- Pedidos: Crear pedidos, listar pedidos de un usuario, detalle de un pedido.

PATRONES DE DISEÑO

en este caso podríamos usar los siguientes:

MVC

controlador/modelo/vista

CAPA DE SERVICIO

lógica de negocio de los controladores

PATRÓN DE REPOSITORIO

lógica de consulta de la base de datos en una capa independiente

TEMAS DE SEGURIDAD

AUTENTICACIÓN Y AUTORIZACIÓN:

Uso de JWT para asegurar las rutas.

VALIDACIÓN DE DATOS:

Validación de entradas de usuarios usando librerías como Joi o express-validator.

ENCRIPTACIÓN

Uso de bcrypt para encriptar contraseñas.

POLÍTICAS DE CORS:

Configurar CORS para permitir el acceso solo a dominios confiables.

RATE LIMITING

Implementar limitación de tasas para prevenir ataques de fuerza bruta.

www.chrisaa.co
+57 3196810701