

**UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA**

**FACULTAD DE PRODUCCIÓN Y SERVICIOS**

**PROGRAMA PROFESIONAL DE INGENIERÍA DE SISTEMAS**



**ASIGNATURA:** Tópicos Avanzados En Ingeniería De Software

**DOCENTE:** ROBERT EDISON ARISACA MAMANI

***“Examen práctico 2: Microservicios y Serverless”***

**INTEGRANTES:**

- RODOLFO ROBERT QUISPE HUACHO
- FATIMA GIGI ROJAS CARHUAS
- ALFREDO ALEJANDRO PERALTILLA MENDOZA
- JULIO GERARDO PAUCAR CASTILLO
- DEYNER JUNIOR PATIÑO MENDOZA

**Noviembre - 2024**

## Documentación técnica

<b>1. Introducción.....</b>	<b>2</b>
1.1. Propósito.....	2
1.2. Alcance.....	3
<b>2. Resumen del sistema.....</b>	<b>3</b>
2.1. Diagrama de Arquitectura.....	3
2.2. Tecnologías utilizadas.....	4
2.3. Despliegue del sistema.....	4
<b>3. Guía de instalación.....</b>	<b>5</b>
3.1. Requisitos previos.....	5
3.2. Pasos de instalación.....	5
<b>4. Guía de configuración.....</b>	<b>6</b>
<b>5. Guía de uso.....</b>	<b>6</b>
5.1. Descripción general de la interfaz de usuario.....	6
5.2. Funcionalidad principal.....	9
5.3. Funciones avanzadas.....	9
<b>7. Esquema de base de datos.....</b>	<b>11</b>
7.1. Definiciones de tablas.....	11
7.2. Relaciones.....	11
<b>8. Pruebas.....</b>	<b>12</b>
8.1. Plan de prueba.....	12
8.2. Casos de prueba.....	12
8.3. Resultados de la prueba.....	15
<b>9. Anexos.....</b>	<b>17</b>

## **1. Introducción**

### **1.1. Propósito**

El propósito de este proyecto es desarrollar una aplicación web que permita a una pequeña empresa minorista automatizar su sistema de gestión de inventarios. Mediante el uso de una arquitectura basada en microservicios y tecnologías serverless, se busca proporcionar una solución escalable, flexible y eficiente que facilite el registro de productos, la actualización de cantidades existentes, y la consulta del inventario en tiempo real. Además, el sistema debe optimizar las operaciones de reabastecimiento mediante la generación y registro de notas de entrada y salida, contribuyendo a una mejor toma de decisiones y al ahorro de tiempo en las operaciones diarias.

### **1.2. Alcance**

El sistema actual permite funcionalidades básicas como la creación y listado de productos, así como la aplicación de filtros por categoría, precio y cantidad, lo que permite a los empleados generar reportes e imprimirlos según sus necesidades. Asimismo, el sistema es capaz de gestionar el reabastecimiento del inventario mediante el registro de notas de entrada y salida, asegurando un control adecuado de las existencias.

Link de la pagina: <https://d1wpmifej8u2k4.cloudfront.net/product-list>

## **2. Resumen del sistema**

### **2.1. Diagrama de Arquitectura**

A continuación, se describen sus principales componentes y cómo interactúan:

- **Frontend:**  
La interfaz web de la aplicación permite realizar tareas como registrar productos, actualizar inventarios, etc..
- **API Gateway:**  
Actúa como intermediario entre el frontend y los microservicios del backend.
- **Funciones Lambda:**  
Cada microservicio está implementado como una función en AWS Lambda, lo que elimina la necesidad de administrar servidores.
- **DynamoDB:**  
Se utiliza como base de datos NoSQL para almacenar la información de productos y movimientos de inventario.
- **S3:**  
Almacena recursos estáticos.
- **CloudFront:**  
Distribuye contenido estático de manera eficiente.

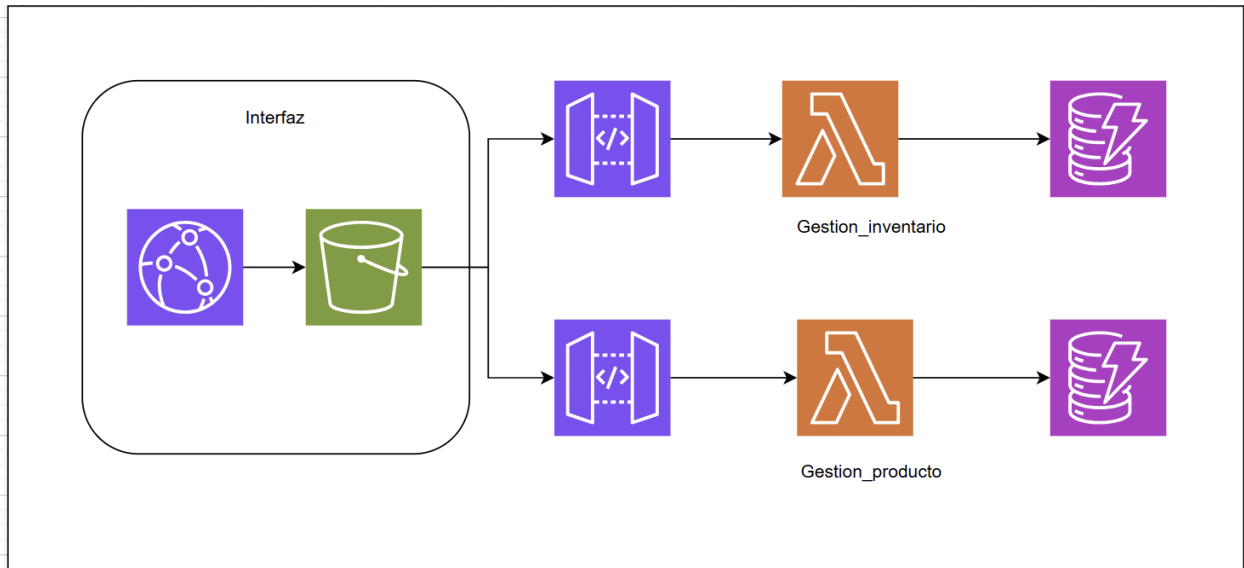


Figura 1: Diagrama de arquitectura. Fuente: Propia

## 2.2. Tecnologías utilizadas

- Frontend: Angular + Bootstrap
- Backend: Python con Django REST Framework
- Base de datos: NoSQL (Amazon DynamoDB ).

## 2.3. Despliegue del sistema

El despliegue de la aplicación utiliza una combinación de servicios de AWS para, siguiendo los principios de una arquitectura serverless.

A continuación, se detalla el proceso:

- **Despliegue del Frontend:**
  - El código del frontend fue compilado en Angular utilizando el comando `ng build`, luego los archivos generados se cargaron en un bucket de **Amazon S3**, configurado para funcionar como un hosting estático.
  - Para mejorar la distribución del contenido, se utilizó **Amazon CloudFront**, un servicio de CDN que proporciona baja latencia y mayor velocidad al entregar los archivos a los usuarios finales.
- **Despliegue del Backend**
  - Se configuró un entorno virtual y se instalaron las dependencias necesarias, incluido Zappa ya que en el archivo `zappa_settings.json` se definió la configuración de despliegue, como: el nombre del proyecto, región de AWS donde se desplegará, permisos requeridos para Lambda y API Gateway.

- Se ejecutó el comando `zappa deploy` para subir el código a **AWS Lambda**, también nos ayudó a crear el **API Gateway** para exponer los endpoints del backend a través de HTTP.

### 3. Guía de instalación

#### 3.1. Requisitos previos

- **Cuenta en AWS:** Necesaria para desplegar los servicios (API Gateway, Lambda, DynamoDB).
- **Node.js:** Para gestionar las dependencias del frontend (Angular).
- **Python 3.8+:** Para ejecutar el backend basado en Django REST Framework.
- **AWS CLI y Zappa:** Para gestionar y desplegar los servicios serverless en AWS.
- **Git:** Para clonar el repositorio del proyecto.

#### 3.2. Pasos de instalación

- **Clonar el repositorio del proyecto:**

```
git clone https://github.com/julio-1610/TAIS\_EXAM2.git  
cd TAIS_EXAM2/TAIS_EXAMEN/
```

- **Configurar el Frontend:**

- Instalar dependencias:

```
cd frontend  
npm install
```

- Ejecutar el servidor de desarrollo

```
ng serve
```

- **Configurar el Backend para gestión de productos y gestión de inventario:**

- Crear un entorno virtual y activarlo

```
cd backend  
cd gestion_inventarios o cd gestion_productos  
python -m venv venv  
venv\Scripts\activate
```

- Instalar dependencias

```
pip install -r requirements.txt
```

## 4. Guía de configuración

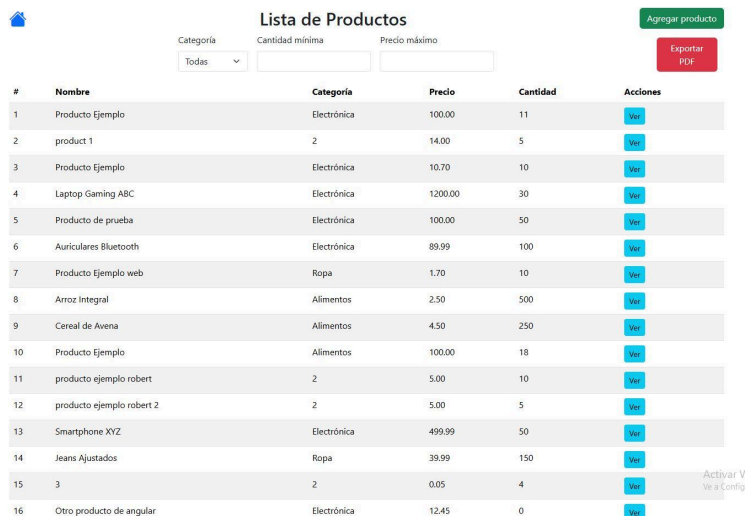
### 4.1. Parámetros de configuración

- **Credenciales AWS:**
  - AWS Acces Key ID: AKIASE5KQ2\*\*\*\*\*437L
  - AWS Secret Acces Key: slg8GJA7WLkU9bUThonigmOsk1J\*\*\*\*\*TZFkK
  - Default region name:us-east-2
- **API Gateway**
  - Productos:<https://ohxkv3ewre.execute-api.us-east-2.amazonaws.com/dev/api/productos/>
  - Movimiento del inventario:  
<https://pzxkp74tnk.execute-api.us-east-2.amazonaws.com/dev/api/movimientos/>

## 5. Guía de uso

### 5.1. Descripción general de la interfaz de usuario

#### ■ Lista de Productos



**Lista de Productos**

Categoría:  Cantidad mínima:  Precio máximo:   
 Todas

#	Nombre	Categoría	Precio	Cantidad	Acciones
1	Producto Ejemplo	Electrónica	100.00	11	<input type="button" value="Ver"/>
2	product 1	2	14.00	5	<input type="button" value="Ver"/>
3	Producto Ejemplo	Electrónica	10.70	10	<input type="button" value="Ver"/>
4	Laptop Gaming ABC	Electrónica	1200.00	30	<input type="button" value="Ver"/>
5	Producto de prueba	Electrónica	100.00	50	<input type="button" value="Ver"/>
6	Auriculares Bluetooth	Electrónica	89.99	100	<input type="button" value="Ver"/>
7	Producto Ejemplo web	Ropa	1.70	10	<input type="button" value="Ver"/>
8	Arroz Integral	Alimentos	2.50	500	<input type="button" value="Ver"/>
9	Cereal de Avena	Alimentos	4.50	250	<input type="button" value="Ver"/>
10	Producto Ejemplo	Alimentos	100.00	18	<input type="button" value="Ver"/>
11	producto ejemplo robert	2	5.00	10	<input type="button" value="Ver"/>
12	producto ejemplo robert 2	2	5.00	5	<input type="button" value="Ver"/>
13	Smartphone XYZ	Electrónica	499.99	50	<input type="button" value="Ver"/>
14	Jeans Ajustados	Ropa	39.99	150	<input type="button" value="Ver"/>
15	3	2	0.05	4	<input type="button" value="Ver"/>
16	Otro producto de angular	Electrónica	12.45	0	<input type="button" value="Ver"/>

Activar V

Descripción: Esta es la página principal y funciona como un dashboard, tenemos todos los productos, y la cantidad que están disponibles.

Herramientas:

#### ● Filtros:

Categoría:  Cantidad mínima:  Precio máximo:   
 Todas

- Categoría: Nos permite buscar el producto según su categoría, Actualmente tenemos 3 categorías, electrónica, ropa y alimentos.
- Cantidad mínima: Nos permite buscar el producto según la cantidad mínima que tengan
- Precio máximo: Nos permite buscar el producto según su precio

Ejemplo de uso de filtros:



### Lista de Productos

Agregar producto

Categoría

Cantidad mínima

Precio máximo

Exportar PDF

Electrónica

2

#	Nombre	Categoría	Precio	Cantidad	Acciones
1	Producto Ejemplo	Electrónica	100.00	11	<span>Ver</span>
2	Producto Ejemplo	Electrónica	10.70	10	<span>Ver</span>
3	Laptop Gaming ABC	Electrónica	1200.00	30	<span>Ver</span>
4	Producto de prueba	Electrónica	100.00	50	<span>Ver</span>
5	Auriculares Bluetooth	Electrónica	89.99	100	<span>Ver</span>
6	Smartphone XYZ	Electrónica	499.99	50	<span>Ver</span>

- Exportar pdf: Es un botón que permite la descarga de un pdf con todos los datos de la lista que encontramos.



- Agregar Producto: El botón nos llevará a un formulario que nos servirá para agregar un producto nuevo

Agregar producto

- Ver Detalles: El botón nos llevará hacia la ventana detalles de producto

Ver

- **Formulario Agregar Producto:** Nos permite crear un nuevo producto, actualmente tiene 5 campos, y un botón que sirve para guardar un nuevo producto

Agregar Producto

×

Nombre del producto:

Descripción (Opcional)

Cantidad

Precio Unitario

Categoría

Guardar

- **Detalle de Producto:** Es la página que nos da todos los detalles de un producto, debajo de ello tenemos todos los movimientos de salida y entrada que tuvo aquel producto

### Detalle del Producto

Producto: Producto Ejemplo

Descripción: Descripción del producto ejemplo

Categoría: Electrónica

Precio Unitario: 100.00

Cantidad: 11

Volver

### Movimientos del Producto

Agregar Movimiento

#	Tipo	Cantidad	Descripción
1	SALIDA	7	nosedg
2	SALIDA	1	compra producto ejemplo
3	SALIDA	1	compra sd
4	ENTRADA	5	12345
5	ENTRADA	5	nosedg



- **Agregar movimiento:** Es el formulario que nos permite agregar un nuevo movimiento, ya sea entrada o salida

#### Agregar Movimiento



Tipo de Movimiento\*

Selecciona un tipo



Descripción\*

Cantidad\*

0

Guardar Movimiento

### 5.2. Funcionalidad principal

- **Ver Lista de Productos:** Podemos visualizar una lista con todos los productos.
- **Filtrar:** Podemos filtrar para tener una búsqueda de un producto más fácil, hay varias opciones de filtrado:
  - Categoría
  - Cantidad
  - Precio
- **Agregar Producto:** Podemos agregar un nuevo producto.
- **Ver Detalles de producto:** Podemos ver todos los detalles del producto, esos detalles son: Nombre, descripción, categoría, precio unitario y la cantidad.
- **Lista de Movimientos:** Podemos visualizar la lista de movimientos de un producto en específico, podemos ver datos del movimiento como si este fue de salida, la cantidad y una pequeña descripción.
- **Agregar Movimiento:** Es un formulario que nos permite agregar un movimiento nuevo. este puede ser de salida o entrada, agregar el movimiento afecta a la cantidad de productos que tenemos

### 5.3. Funciones avanzadas

- **Generar archivo pdf con los productos:** Podemos generar un pdf que se nos descarga con una lista de productos, estos pueden ser filtrados o no, eh aquí un ejemplo.



#### Lista de Productos Filtrados

#	Nombre	Categoría	Precio	Cantidad
1	Producto Ejemplo	Electrónica	100.00	11
2	product 1	2	14.00	5
3	Producto Ejemplo	Electrónica	10.70	10
4	Laptop Gaming ABC	Electrónica	1200.00	30
5	Producto de prueba	Electrónica	100.00	50
6	Auriculares Bluetooth	Electrónica	89.99	100
7	Producto Ejemplo web	Ropa	1.70	10
8	Arroz Integral	Alimentos	2.50	500
9	Cereal de Avena	Alimentos	4.50	250
10	Producto Ejemplo	Alimentos	100.00	18
11	producto ejemplo robert	2	5.00	10
12	producto ejemplo robert 2	2	5.00	5
13	Smartphone XYZ	Electrónica	499.99	50
14	Jeans Ajustados	Ropa	39.99	150
15	3	2	0.05	4
16	Otro producto de angular	Electrónica	12.45	0
17	nose 2	2	2.00	5
18	Camiseta Casual	Ropa	19.99	200
19	Aceite de Oliva	Alimentos	5.99	300
20	Chaqueta de Invierno	Ropa	89.99	80

## 6. Documentación API

### 6.1. Puntos finales

- **Gestión Productos**

Método	URL	Descripción
GET	/dev/api/productos/	Obtiene una lista de todos los productos en el inventario.
POST	/dev/api/productos/	Crea un nuevo producto en el inventario.
DELETE	/dev/api/productos/{id_producto}/	Elimina un producto del inventario.
POST	/dev/api/productos/<str:id_producto>/actualizar_inventario/	Este endpoint permite a los sistemas de gestión de inventarios actualizar la cantidad de un producto específico en el inventario.

- **Gestión Movimientos de inventario**

Método	URL	Descripción
GET	/dev/api/movimientos/	Obtiene una lista de todos los productos en el inventario.

POST	/dev/api/movimientos/	Crea un nuevo producto en el inventario.
DELETE	/dev/api/movimientos/{id_movimiento}/	Elimina un producto del inventario.

## 7. Esquema de base de datos

### 7.1. Definiciones de tablas

Las tablas necesarias (productos y movimientos) fueron configuradas directamente en DynamoDB.

- **Tabla: Productos**

Atributo	Tipo de Dato	Descripción
id_producto	String (PK)	Identificador único del producto
nombre	String	Nombre del producto
descripcion	String	Descripción del producto
categoria	String	Categoría del producto
precio	Number	Precio del producto
cantidad	Number	Cantidad disponible en inventario

- **Tabla: MovimientosInventario**

Atributo	Tipo de Dato	Descripción
id_movimiento	String (PK)	Identificador único del movimiento
id_producto	String (SK)	ID del producto relacionado al movimiento
tipo_movimiento	String	Tipo de movimiento (entrada o salida)
descripcion	String	Descripción del movimiento
cantidad	Number	Cantidad movida (negativa para salidas)

### 7.2. Relaciones

La relación entre las tablas Productos y MovimientosInventario está establecida mediante el atributo id\_producto en la tabla MovimientosInventario. Esto permite realizar consultas para obtener los movimientos de inventario asociados con un producto en específico.

## **8. Pruebas**

### **8.1. Plan de prueba**

El objetivo de las pruebas es validar que el sistema de gestión de inventarios funcione correctamente, garantizando la correcta interacción entre el frontend, el backend y la base de datos, así como asegurar que todas las funcionalidades sean confiables y escalables.

Alcance de las Pruebas:

- Verificación de la creación, actualización y eliminación de productos.
- Verificación de la gestión de movimientos de inventario (entradas y salidas).
- Validación de las respuestas de la API.
- Evaluación de la integridad de los datos en la base de datos (DynamoDB).

### **8.2. Casos de prueba**

#### **Prueba de actualización de producto o inventario**

POST flujo: Actualizar inventario

**Método:** POST

**URL:**

`https://<tu-api-endpoint>/dev/api/productos/<str:id_producto>/actualizar_inventario/`

**Body:** raw y JSON.

#### **Prueba de Creación de Producto**

Descripción: Verificar que el sistema permite agregar un nuevo producto al inventario.

**Método:** POST

**URL:** `/dev/api/productos/`

**Body :**

json

```
1  {  
2    "nombre": "Producto de prueba",  
3    "descripcion": "Descripción del producto de prueba",  
4    "categoria": "Electrónica",  
5    "precio": 100.00,  
6    "cantidad": 50  
7  }
```

**Resultado esperado:** El producto debe ser creado correctamente en la base de datos, y la respuesta de la API debe contener los datos del producto recién creado.

#### Prueba de Manejo de Error: Intento de Eliminar un Producto Inexistente

**Descripción:** Verificar que la API maneje correctamente el intento de eliminar un producto que no existe.

**Método:** DELETE

<https://ohxkv3ewre.execute-api.us-east-2.amazonaws.com/dev/api/productos/999999999/>

#### Prueba de Manejo de Error: Intento de Eliminar un Producto existente

**Crear el producto (POST):**

**Método:** POST

**URL:**

<https://ohxkv3ewre.execute-api.us-east-2.amazonaws.com/dev/api/productos/>

**Cuerpo:**



```
1  {  
2    "nombre": "Producto a eliminar",  
3    "descripcion": "Este producto será eliminado después de la prueba",  
4    "categoria": "Electrónica",  
5    "precio": 150.00,  
6    "cantidad": 30  
7  }
```

### **Eliminar el producto (DELETE):**

Método: DELETE

URL:

[https://ohxkv3ewre.execute-api.us-east-2.amazonaws.com/dev/api/products/{id\\_producto}/](https://ohxkv3ewre.execute-api.us-east-2.amazonaws.com/dev/api/products/{id_producto}/)

ID: "id\_producto": "690a6266-e1c1-4a0b-8e98-fda752664139"

### **Prueba Consulta a todos los Productos (GET)**

**Descripción:** Verificar que la API pueda devolver una lista de todos los productos en el inventario.

**Método:** GET

URL:

<https://ohxkv3ewre.execute-api.us-east-2.amazonaws.com/dev/api/products/>



### 8.3. Resultados de la prueba

#### Pruebas unitarias de Frontend:

28 specs, 1 failure, randomized with seed 12945

Spec List | [Failures](#)

```
ProductService
  • should retrieve products via GET
  • should retrieve a product by ID via GET
  • should save a product via POST

AppComponent
  • should have the 'frontend' title
  • should create the app
  x should render title

ProductStateService
  • should set and get a product
  • should clear a product

MovementFormComponent
  • should create the component
  • should update cantidadDisponible from ProductService
  • should call obtenerCantidadDisponible when idProducto changes

ConfirmationComponent
  • should create
  • should return the correct icon class for "success"
  • should display the message and hide it after 5 seconds
  • should use the default icon class for an unknown type
  • should return the correct icon class for "danger"

ProductListComponent
  • should filter products correctly
  • should navigate to product details on verDetalle
  • should load products on initialization
  • should create the component

ProductFormComponent
  • should reset the form after saving
  • should create the component
  • should validate unique product name
  • should load existing product names on initialization

MovementService
  • should retrieve movements via GET
  • should save a movement via POST

ProductViewComponent
  • should create the component
  • should navigate to product-list if no product is found
```

#### Prueba de Creación de Producto



**UNSA**  
UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA

## Documento Técnico de Gestión de Inventarios

Body Cookies Headers (19) Test Results

201 Created • 291 ms • 963 B

Pretty Raw Preview Visualize JSON

```
1 {
2   "id_producto": "cd8e1983-1cc4-4b79-8b15-3f358be993e5",
3   "nombre": "Producto de prueba",
4   "descripcion": "Descripción del producto de prueba",
5   "categoria": "Electrónica",
6   "precio": 100.0,
7   "cantidad": 50
8 }
```

✓ 201 Created  
A new resource was created successfully.



### Lista de Productos

Agregar producto

Exportar PDF

Categoría

Cantidad mínima

Precio máximo

Todas

#	Nombre	Categoría	Precio	Cantidad	Acciones
1	Producto Ejemplo	Electrónica	100.00	11	Ver
2	product 1	2	14.00	5	Ver
3	Producto Ejemplo	Electrónica	10.70	10	Ver
4	Laptop Gaming ABC	Electrónica	1200.00	30	Ver

### Prueba de Manejo de Error: Intento de Eliminar un Producto Inexistente

DELETE https://ohxkv3ewre.execute-api.us-east-2.amazonaws.com/dev/api/productos/999999999/

Params Authorization Headers (5) Body Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON

1

Body Cookies Headers (19) Test Results

404 Not Found

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "Producto no encontrado"
3 }
```

### Prueba de Manejo de Error: Intento de Eliminar un Producto Existente

POST https://ohxkv3ewre.execute-api.us-east-2.amazonaws.com/dev/api/productos/

Params Authorization Headers (7) Body Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON

1 {
2 "nombre": "Producto a eliminar",
3 "descripcion": "Este producto será eliminado después de la prueba",
4 "categoria": "Electrónica",
5 "precio": 150.00,
6 "cantidad": 30
7 }

Body Cookies Headers (19) Test Results

201 Created • 421 ms • 983 B

Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "id_producto": "698a6266-e1c1-4a8b-8e98-fda762664139",
3   "nombre": "Producto a eliminar",
4   "descripcion": "Este producto será eliminado después de la prueba",
5   "categoria": "Electrónica",
6   "precio": 150.0,
7   "cantidad": 30
8 }
```





**UNSA**  
UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA

## Documento Técnico de Gestión de Inventarios

DELETE <https://ohxkv3ewre.execute-api.us-east-2.amazonaws.com/dev/api/productos/690a6266-e1c1-4a0b-8e98-fda752664139/>

Params Authorization Headers (7) **Body** Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▾

```
1 {
2   "nombre": "Producto a eliminar",
3   "descripcion": "Este producto será eliminado después de la prueba",
4   "categoria": "Electrónica",
5   "precio": 150.00,
6   "cantidad": 30
7 }
```

Body Cookies Headers (19) Test Results

200 OK • 868 ms • 862 B

Pretty Raw Preview Visualize **JSON** ▾

```
1 {
2   "message": "Producto eliminado correctamente"
3 }
```

### Prueba Consulta a todos los Productos (GET)

<https://ohxkv3ewre.execute-api.us-east-2.amazonaws.com/dev/api/productos/>

GET <https://ohxkv3ewre.execute-api.us-east-2.amazonaws.com/dev/api/productos/>

Params Authorization Headers (5) **Body** Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▾

1

Body Cookies Headers (19) Test Results

200 OK

Pretty Raw Preview Visualize **JSON** ▾

```
1 [
2   {
3     "id_producto": "12345",
4     "nombre": "Producto Ejemplo",
5     "descripcion": "Descripción del producto ejemplo",
6     "categoria": "Electrónica",
7     "precio": "100.00",
8     "cantidad": 11
9   },
10  {
11    "id_producto": "123e99e4-21e6-4c76-9e4c-378c514c63c6",
12    "nombre": "product 1",
13    "descripcion": "4",
14    "categoria": "2",
15    "precio": "14.00",
16    "cantidad": 5
17  },
18  {
19    "id_producto": "1283d603-9351-4e99-8d11-0eb0063209ec",
20    "nombre": "Producto Ejemplo",
21    "descripcion": "Descripción del producto ejemplo",
22    "categoria": "Electrónica",
23    "precio": "10.70",
24    "cantidad": 10
25  },
26  {
27    "id_producto": "e7d8f9g0h112",
28    "nombre": "Laptop Gaming ABC",
29    "descripcion": "Laptop para juegos con tarjeta gráfica GTX 1660 y 16GB de RAM.",
30    "categoria": "Electrónica",
31    "precio": "1200.00",
32    "cantidad": 30
33  },
34  {
35    "id_producto": "cd8e1903-1cc4-4b79-8b15-3f350be993e5",
```

## 9. Anexos

- Video

url: <https://drive.google.com/drive/folders/1kqEn8pWKh5xTcJaPIFFe1xBOvhnMw1zw?usp=sharing>

- Link del github: [https://github.com/julio-1610/TAIS\\_EXAM2.git](https://github.com/julio-1610/TAIS_EXAM2.git)
- Link de la pagina: <https://d1wpmifei8u2k4.cloudfront.net/product-list>