

DAVIAN STIVEN ASCANIO | EJERCICIO DE LA API

1. Verificar estado del servicio

```
# --- API REST: Endpoint de estado del servicio ---
@app.route('/status', methods=['GET'])
def service_status():
    """
    Endpoint de prueba para verificar el estado del servicio.
    Devuelve un mensaje JSON indicando que el servicio funciona correctamente.
    """
    return jsonify(message="El servicio está funcionando correctamente.")
```

Descripción

Verifica que el servicio Flask está funcionando correctamente. Es útil para pruebas de salud del sistema.

Respuesta exitosa

Ejemplo de uso

- Método: GET
- URL: <https://tienda-arq.onrender.com/status>

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** https://tienda-arq.onrender.com/status
- Params:** Query Params table with 2 columns: Key, Value.
- Body:** JSON format showing the response:

```
{
  "message": "El servicio está funcionando correctamente."
}
```

2. Listar todos los productos

```
# --- API REST: Listar productos ---
@app.route('/api/productos', methods=['GET'])
def listar_productos():
    """
    Endpoint para obtener todos los productos en formato JSON.
    """
    productos = Producto.query.all()
    return jsonify([
        {
            "id": p.id,
            "nombre": p.nombre,
            "descripcion": p.descripcion,
            "precio": p.precio,
            "id_categoria": p.id_categoria
        }
        for p in productos
    ])
```

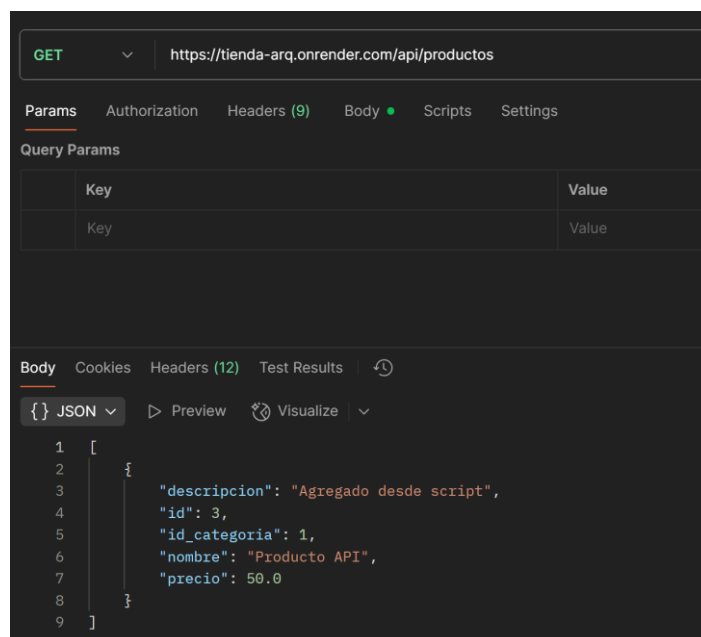
Descripción

Devuelve una lista de todos los productos registrados en la base de datos, en formato JSON.

Respuesta exitosa

Ejemplo de uso

- Método: GET
- URL: <https://tienda-arq.onrender.com/api/productos>



The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** <https://tienda-arq.onrender.com/api/productos>
- Params:** Authorization, Headers (9), Body, Scripts, Settings
- Query Params:** A table with 2 columns: Key, Value.
- Body:** Cookies, Headers (12), Test Results, Visualize
- Response:** JSON array containing one product object.

```
1 [
2   {
3     "descripcion": "Agregado desde script",
4     "id": 3,
5     "id_categoria": 1,
6     "nombre": "Producto API",
7     "precio": 50.0
8   }
9 ]
```

3. Agregar un nuevo producto

```
# --- API REST: Agregar producto ---
@app.route('/api/productos/agregar', methods=['POST'])
def agregar_producto():
    """
    Endpoint para agregar un nuevo producto vía JSON.
    Espera los campos: nombre, descripcion, precio, id_categoria.
    """
    data = request.get_json()
    if not data:
        return jsonify({"error": "No se proporcionaron datos JSON"}), 400
    try:
        nuevo = Producto(
            nombre=data["nombre"],
            descripcion=data.get("descripcion", ""),
            precio=data["precio"],
            id_categoria=data["id_categoria"]
        )
        db.session.add(nuevo)
        db.session.commit()
        return jsonify({"message": "Producto agregado correctamente."}), 201
    except Exception as e:
        db.session.rollback()
        return jsonify({"error": f"Error al agregar producto: {str(e)}"}), 500
```

Descripción

Permite agregar un nuevo producto enviando los datos en formato JSON. Los campos requeridos son:

- nombre (string), descripción, (string, opcional), precio (float), id_categoria (int)

Ejemplo de uso

- Método: GET
- URL: <https://tienda-arq.onrender.com/api/productos/agregar>

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** <https://tienda-arq.onrender.com/api/agregar>
- Body Type:** raw
- Request Body (JSON):**

```
1 {
2   "nombre": "Cargador",
3   "descripcion": "Cargador de PC",
4   "precio": 50.0,
5   "id_categoria": 1
6 }
```
- Response Body (JSON):**

```
1 {
2   "message": "Producto agregado correctamente."
3 }
```