

Práctica 3: Crear servicio web REST

Objetivo:

En esta práctica creamos el back-end de la aplicación de e-commerce que hemos estado desarrollando en las demás prácticas. Apoyándonos con Node.js, creamos nuestra REST API que permitirá crear, modificar, editar, consultar y borrar los productos, para facilitar las cosas guardaremos la información en archivos JSON y realizaremos una autenticación muy básica. Para validar nuestro servicio web, lo haremos únicamente con solicitudes de HTTP para realizar operaciones de altas, bajas, cambios y consultas usando PostMan.

Instrucciones:

Crear una REST API para nuestra aplicación de e-commerce.

Esta práctica contiene los siguientes puntos:

1. Consultar productos
2. Consultar productos en específico
3. Consultar productos según una lista de IDs
4. Registro de nuevos productos (con validaciones)
5. Editar información de un producto
6. Borrar algún producto
7. Filtro de productos (obtener todos o algunos según los parámetros)
8. Manejo de errores de respuesta de HTTP
9. Validación de headers usando middlewares
10. Pruebas y ejemplos

Endpoint	Método	headers	body	Comentarios
/products/	GET			Si se pasa el parámetro "query" aplicar el filtro correspondiente.
/products/:id	GET			Regresar el producto correspondiente. (mandar status 200). Si el ID no coincide mandar status 404.
/products/cart	POST	content-type: application/json	Arreglo de objetos con los atributos: productUuid, amount	Regresar un arreglo con los productos correspondientes. (mandar status 200) Si body no es un arreglo mandar status 400. Si algún ID no coincide mandar status 404.
/admin/products/	POST	x-auth content-type: application/json	uuid, imageUrl, title, description, unit, category, pricePerUnit, stock	Crear el producto correspondiente, en caso de éxito mandar status 201, de lo contrario mandar 400.
/admin/products/:id	PUT	x-auth content-type: application/json	imageUrl, title, description, unit, category, pricePerUnit, stock	Actualizar el producto correspondiente, en caso de éxito mandar status 200, de lo contrario mandar 400. Si el ID no coincide mandar status 404.
/admin/products/:id	DELETE	x-auth		Borrar el producto correspondiente, en caso de éxito mandar status 200. Si el ID no coincide mandar status 404.

Evaluación:

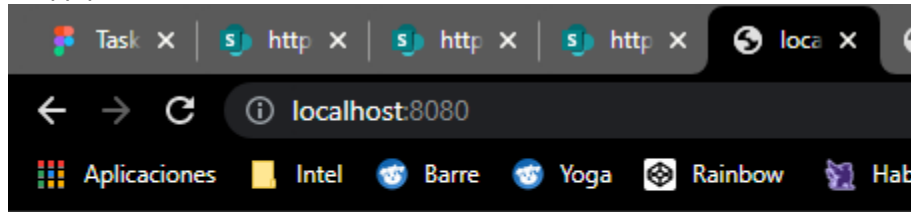
Máxima calificación de la práctica: 100 puntos (los puntos sobrantes no son acumulables)

Requerimiento	Valor	Realizado (SI / NO)
<i>GET /products</i> <i>- Filtro por parámetros de búsqueda.</i>	15	SI
<i>POST /products/cart</i>	15	SI
<i>GET /products/:id</i>	10	SI
<i>Middleware de autenticar</i>	10	SI
<i>POST /admin/products</i>	10	SI
<i>PUT /admin/products/:id</i>	15	SI
<i>DELETE /admin/products:id</i>	10	SI
<i>GET /, /home, /shopping_cart</i>	10	SI
<i>Código estructurado y modularizado</i> <i>- Archivo de router global</i> <i>- Archivo de router para productos</i> <i>- Archivo de router para endpoints de administrador de productos</i>	10	SI
<i>Manejo de archivos</i>	10	SI
TOTAL	115/ 100	

Evidencias:

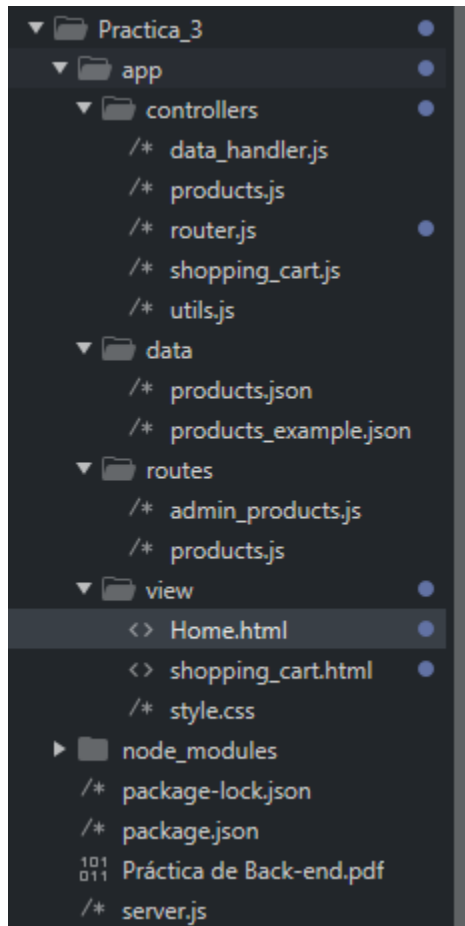
1. Configuración y estructura de archivos

Para comprobar y facilitar la práctica 3, configuramos node.js de forma que la ruta raíz regrese “e-commerce app práctica 3”



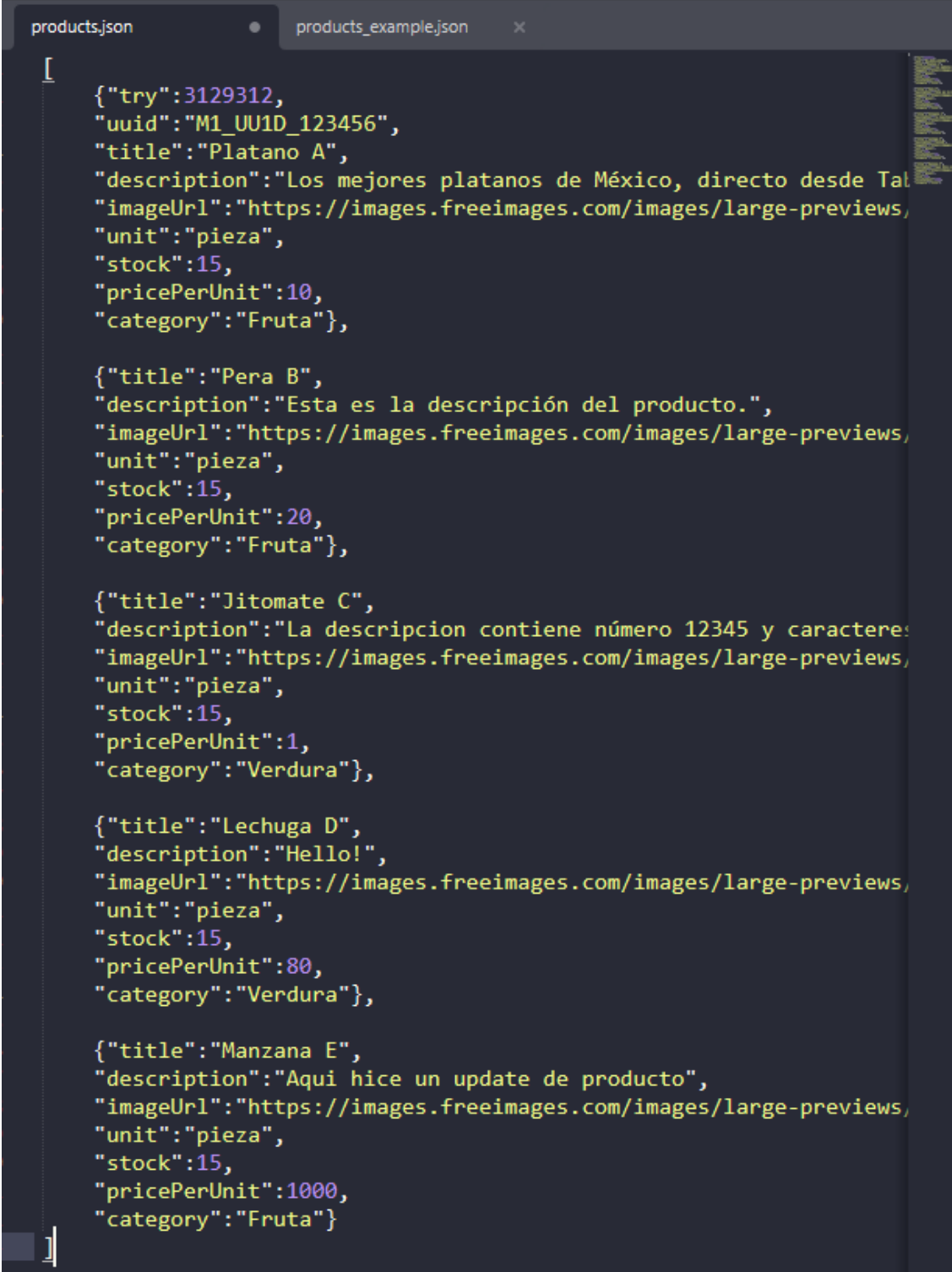
e-commerce app práctica 3

Adicional creamos una estructura de archivos:



2. GET /products

Esta solicitud regresa la lista de productos que se encuentren en nuestro documento/base.



```
products.json  products_example.json  x

[
  {
    "try": 3129312,
    "uuid": "M1_UU1D_123456",
    "title": "Platano A",
    "description": "Los mejores platanos de México, directo desde Tal",
    "imageUrl": "https://images.freeimages.com/images/large-previews",
    "unit": "pieza",
    "stock": 15,
    "pricePerUnit": 10,
    "category": "Fruta"
  },

  {
    "title": "Pera B",
    "description": "Esta es la descripción del producto.",
    "imageUrl": "https://images.freeimages.com/images/large-previews",
    "unit": "pieza",
    "stock": 15,
    "pricePerUnit": 20,
    "category": "Fruta"
  },

  {
    "title": "Jitomate C",
    "description": "La descripción contiene número 12345 y caracteres",
    "imageUrl": "https://images.freeimages.com/images/large-previews",
    "unit": "pieza",
    "stock": 15,
    "pricePerUnit": 1,
    "category": "Verdura"
  },

  {
    "title": "Lechuga D",
    "description": "Hello!",
    "imageUrl": "https://images.freeimages.com/images/large-previews",
    "unit": "pieza",
    "stock": 15,
    "pricePerUnit": 80,
    "category": "Verdura"
  },

  {
    "title": "Manzana E",
    "description": "Aquí hice un update de producto",
    "imageUrl": "https://images.freeimages.com/images/large-previews",
    "unit": "pieza",
    "stock": 15,
    "pricePerUnit": 1000,
    "category": "Fruta"
  }
]
```

products.json

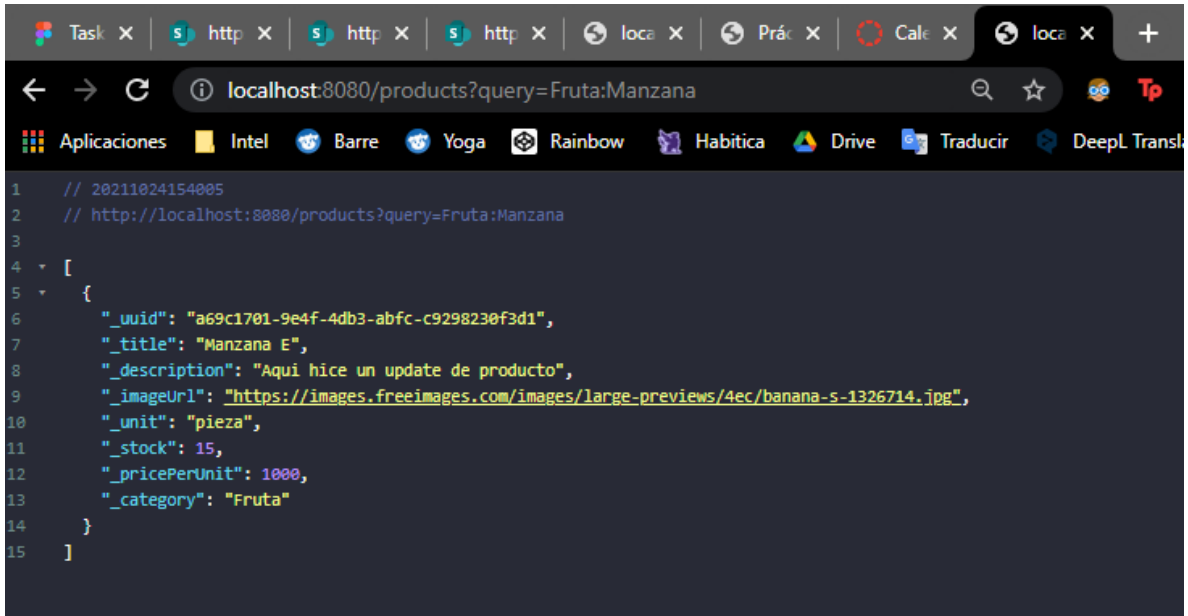
1. Regresa todos los productos: regresa los valores que ya estén cargados en nuestro arreglo de productos (products).



The screenshot shows a web browser window with the address bar displaying 'localhost:8080/products'. The browser's developer tools are open, showing a REST client interface. The first tab is selected, showing a GET request to 'localhost:8080/products'. The response is a JSON array of five product objects. The products are: 'Platano A', 'Pera B', 'Jitomate C', 'Lechuga D', and 'Manzana E'. Each product object contains fields for '_uuid', '_title', '_description', '_imageUrl', '_unit', '_stock', '_pricePerUnit', and '_category'.

```
1 // 20211024153901
2 // http://localhost:8080/products
3
4 [
5   {
6     "_uuid": "4537cc3b-7645-4330-842d-43704f96c9a4",
7     "_title": "Platano A",
8     "_description": "Los mejores platanos de México, directo desde Tabasco.",
9     "_imageUrl": "https://images.freeimages.com/images/large-previews/4ec/banana-s-1326714.jpg",
10    "_unit": "pieza",
11    "_stock": 15,
12    "_pricePerUnit": 10,
13    "_category": "Fruta"
14  },
15  {
16    "_uuid": "86e23484-698c-4b9b-ad3f-907aeb0b3cfb",
17    "_title": "Pera B",
18    "_description": "Esta es la descripción del producto.",
19    "_imageUrl": "https://images.freeimages.com/images/large-previews/4ec/banana-s-1326714.jpg",
20    "_unit": "pieza",
21    "_stock": 15,
22    "_pricePerUnit": 20,
23    "_category": "Fruta"
24  },
25  {
26    "_uuid": "a3570af5-ea8c-4f16-b454-e551b29e878c",
27    "_title": "Jitomate C",
28    "_description": "La descripción contiene número 12345 y caracteres especiales ñ.-/?%.",
29    "_imageUrl": "https://images.freeimages.com/images/large-previews/4ec/banana-s-1326714.jpg",
30    "_unit": "pieza",
31    "_stock": 15,
32    "_pricePerUnit": 1,
33    "_category": "Verdura"
34  },
35  {
36    "_uuid": "6042e48a-67b2-477c-90ce-e655c861258c",
37    "_title": "Lechuga D",
38    "_description": "Hello!",
39    "_imageUrl": "https://images.freeimages.com/images/large-previews/4ec/banana-s-1326714.jpg",
40    "_unit": "pieza",
41    "_stock": 15,
42    "_pricePerUnit": 80,
43    "_category": "Verdura"
44  },
45  {
46    "_uuid": "a69c1701-9e4f-4db3-abfc-c9298230f3d1",
47    "_title": "Manzana E",
48    "_description": "Aquí hice un update de producto",
49    "_imageUrl": "https://images.freeimages.com/images/large-previews/4ec/banana-s-1326714.jpg",
50    "_unit": "pieza",
51    "_stock": 15,
52    "_pricePerUnit": 1000,
53    "_category": "Fruta"
54  }
55 ]
```

2. Regresar lista filtrada de productos: verifica si la petición cuenta con parámetros de búsqueda, en cuyo caso filtramos la lista y regresamos solo los valores deseados.



The screenshot shows a web browser with multiple tabs. The active tab is titled 'localhost:8080/products?query=Fruta:Manzana'. The browser's address bar shows the URL. Below the address bar, there is a toolbar with various application icons. The main content area of the browser displays a REST client response. The response is a JSON array containing one object. The object has the following properties: '_uuid' (a69c1701-9e4f-4db3-abfc-c9298230f3d1), '_title' (Manzana E), '_description' (Aqui hice un update de producto), '_imageUrl' (https://images.freeimages.com/images/large-previews/4ec/banana-s-1326714.jpg), '_unit' (pieza), '_stock' (15), '_pricePerUnit' (1000), and '_category' (Fruta). The response is displayed in a dark-themed editor with line numbers on the left.

```
1 // 20211024154005
2 // http://localhost:8080/products?query=Fruta:Manzana
3
4 [
5   {
6     "_uuid": "a69c1701-9e4f-4db3-abfc-c9298230f3d1",
7     "_title": "Manzana E",
8     "_description": "Aqui hice un update de producto",
9     "_imageUrl": "https://images.freeimages.com/images/large-previews/4ec/banana-s-1326714.jpg",
10    "_unit": "pieza",
11    "_stock": 15,
12    "_pricePerUnit": 1000,
13    "_category": "Fruta"
14  }
15 ]
```

3. POST /products/cart

Para poder cargar los productos correspondientes en nuestro carrito de compras, debemos permitir la búsqueda de productos directamente por su ID.

1. Tendremos que validar que el body recibido sea de tipo arreglo, de lo contrario debemos regresar un estatus de error (400).

Practica 3 / Product - Cart

POST http://localhost:8080/products/cart

Params Auth Headers (9) Body Pre-req. Tests Settings

raw JSON

```
1
2 {
3   "productUUID": "61affc91-02a1-4434-89e2-de09981e62dd",
4   "amount": "1"
5 }
```

Body 400 Bad Request 6 ms 265 B Save Response

Pretty Raw Preview Visualize

Body debe de ser un arreglo!

2. Iteramos por el arreglo recibido e iremos agregando los elementos correspondientes a una lista temporal de productos, en caso de que no se encuentre algún producto, debemos regresar un estatus de error (404) y un mensaje explicando que no se encontró dicho producto.

Practica 3 / Product - Cart

POST http://localhost:8080/products/cart

Params Auth Headers (9) Body Pre-req. Tests Settings

raw JSON

```
1 [
2   {
3     "productUUID": "61affc91-02a1-4434-89e2-de09981e62dd",
4     "amount": "1"
5   },
6   {
7     "productUUID": "Y0-N0-3X15T0",
8     "amount": "1"
9   }
]
```

Body 404 Not Found 8 ms 277 B Save Response

Pretty Raw Preview Visualize

Producto con UUID: Y0-N0-3X15T0 no existe!

3. Al terminar si todos los productos fueron encontrados, regresamos la lista temporal con un estatus de éxito (200).

Practica 3 / Product - Cart Save ...

POST ▼ `http://localhost:8080/products/cart` Send

Params Auth Headers (9) **Body** ● Pre-req. Tests Settings Co

raw ▼ **JSON** ▼ Bea

```
1  [
2    {
3      "productUUID": "4537cc3b-7645-4330-842d-43704f96c9a4",
4      "amount": "1"
5    },
6    {
7      "productUUID": "86e23484-698c-4b9b-ad3f-907aeb0b3cfb",
8      "amount": "1"
9    }
10 ]
```

Body ▼ 200 OK 12 ms 819 B Save Respon

Pretty Raw Preview Visualize **JSON** ▼

```
1  [
2    {
3      "_uuid": "4537cc3b-7645-4330-842d-43704f96c9a4",
4      "_title": "Platano A",
5      "_description": "Los mejores platanos de México, directo desde Tabasco.",
6      "_imageUrl": "https://images.freeimages.com/images/large-previews/4ec/banana-s-1326714.jpg",
7      "_unit": "pieza",
8      "_stock": 15,
9      "_pricePerUnit": 10,
10     "_category": "Fruta"
11   },
12   {
13     "_uuid": "86e23484-698c-4b9b-ad3f-907aeb0b3cfb",
14     "_title": "Pera B",
15     "_description": "Esta es la descripción del producto.",
16     "_imageUrl": "https://images.freeimages.com/images/large-previews/4ec/banana-s-1326714.jpg",
17     "_unit": "pieza",
18     "_stock": 15,
19     "_pricePerUnit": 20,
20     "_category": "Fruta"
21   }
22 ]
```


4. GET /products/:id

Debemos permitir la búsqueda de productos directamente por su ID.

1. En caso de que el ID sea inválido, debemos regresar un estatus de error (404) con un mensaje explicando que no se encontró el producto.

Practica 3 / Products - ByID Save ...

GET ▼ http://localhost:8080/products/N0-3X15T0 Send

Params Auth Headers (6) Body Pre-req. Tests Settings Cc

none ▼

This request does not have a body

Body ▼ 404 Not Found 8 ms 274 B Save Response

Pretty Raw Preview Visualize HTML ▼

```
1 Producto con UUID: N0-3X15T0 no existe!
```

2. En caso de que todo esté correcto, regresa un estatus de éxito (200) y el producto correspondiente. Recuerda agregar a la respuesta el header correspondiente para JSON.

Practica 3 / Products - ByID Save ▼ ...

GET ▼ http://localhost:8080/products/61affc91-02a1-4434-89e2-de09981e62dd Send

Params Auth Headers (6) Body Pre-req. Tests Settings C

none ▼

This request does not have a body

Body ▼ 200 OK 8 ms 537 B Save Response

Pretty Raw Preview Visualize JSON ▼

```
1 { "_uuid": "61affc91-02a1-4434-89e2-de09981e62dd", "_title": "Platano A",
  "_description": "Los mejores platanos de México, directo desde Tabasco.",
  "_imageUrl": "https://images.freeimages.com/images/large-previews/4ec/banana-s-1326714.jpg", "_unit": "pieza", "_stock": 15, "_pricePerUnit": 10,
  "_category": "Fruta" }
```

5. Middleware de autenticación

Crear un middleware llamado validateAdmin que verifica que exista el header x-auth con el valor "admin".

Practica 3 / Admin - Try Auth

GET http://localhost:8080/admin **Send**

Params Auth **Headers (7)** Body Pre-req. Tests Settings

Headers 6 hidden

	KEY	VALUE	DES	...	Bulk Edit	Prese
<input checked="" type="checkbox"/>	x-auth	admin				
	Key	Value				Description

Body 200 OK 10 ms 250 B **Save Response**

Pretty Raw Preview Visualize HTML

```
1 ADMIN PRODUCTS WORKING
```

Mandar error 403 en caso de que no exista "Acceso no autorizado, no se cuenta con privilegios de administrador".

Practica 3 / Admin - Try Auth

GET http://localhost:8080/admin **Send**

Params Auth **Headers (7)** Body Pre-req. Tests Settings

Headers 6 hidden

	KEY	VALUE	DES	...	Bulk Edit	Prese
<input type="checkbox"/>	x-auth	admin				
	Key	Value				Description

Body 403 Forbidden 27 ms 256 B **Save Response**

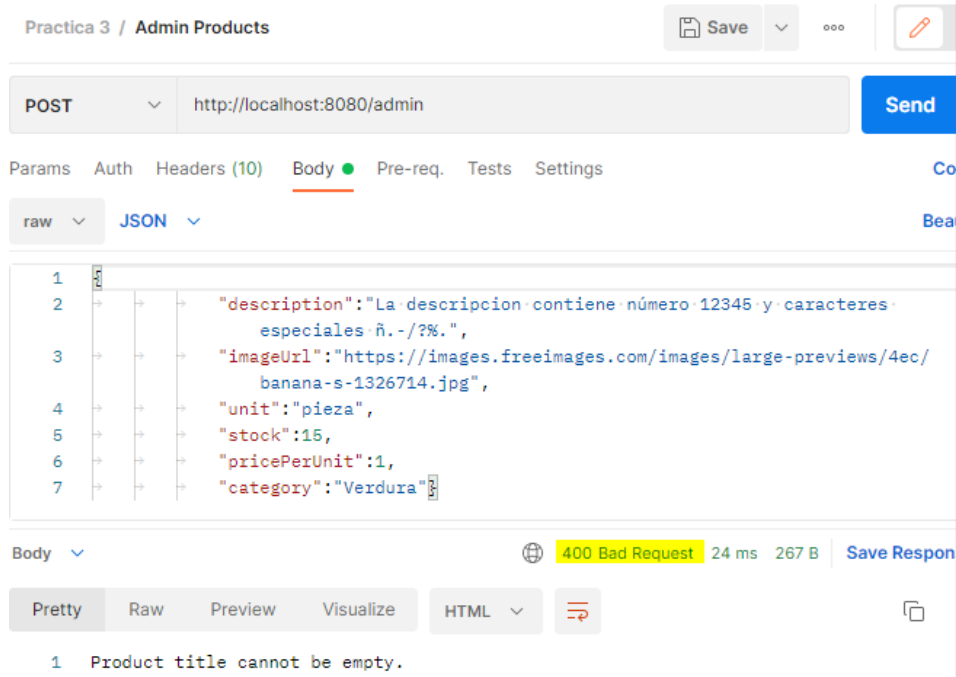
Pretty Raw Preview Visualize Text

```
1 Acceso no autorizado
```

6. POST /admin/products

Para la parte de administrador, lo primero que realizaremos será trabajar con el registro de un producto. Para hacer el registro debemos hacer varias cosas:

1. Validar que en el body se encuentren los atributos que nos interesan en caso contrario regresar bad request (400) e indicar en el mensaje de respuesta cuáles atributos faltan.



Practica 3 / Admin Products

POST http://localhost:8080/admin

Params Auth Headers (10) Body Pre-req. Tests Settings

raw JSON

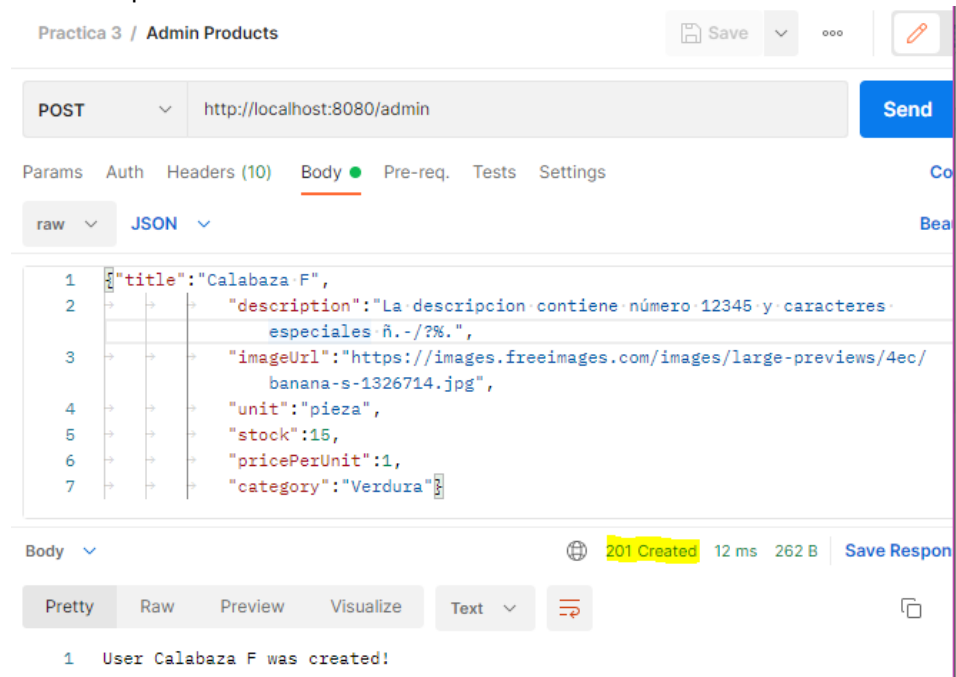
```
1 {}
2 {
3   "description": "La descripcion contiene número 12345 y caracteres especiales ñ.-/?%.",
4   "imageUrl": "https://images.freeimages.com/images/large-previews/4ec/banana-s-1326714.jpg",
5   "unit": "pieza",
6   "stock": 15,
7   "pricePerUnit": 1,
8   "category": "Verdura"
9 }
```

Body 400 Bad Request 24 ms 267 B Save Response

Pretty Raw Preview Visualize HTML

```
1 Product title cannot be empty.
```

2. Asegúrate que el ID del producto, se autogenera usando UUIDs.
3. Guardar el nuevo producto en el arreglo y regresar el status 201 y como mensaje el nombre del producto creado.



Practica 3 / Admin Products

POST http://localhost:8080/admin

Params Auth Headers (10) Body Pre-req. Tests Settings

raw JSON

```
1 {"title": "Calabaza F",
2   "description": "La descripcion contiene número 12345 y caracteres especiales ñ.-/?%.",
3   "imageUrl": "https://images.freeimages.com/images/large-previews/4ec/banana-s-1326714.jpg",
4   "unit": "pieza",
5   "stock": 15,
6   "pricePerUnit": 1,
7   "category": "Verdura"
8 }
```

Body 201 Created 12 ms 262 B Save Response

Pretty Raw Preview Visualize Text

```
1 User Calabaza F was created!
```

4. No te olvides de actualizar el archivo de productos.

```
22     "_pricePerUnit": 20,  
23     "_category": "Fruta"  
24   },  
25   {  
26     "_uuid": "95f1ce81-5251-4e41-a5bf-84f8982ccf22",  
27     "_title": "Jitomate C",  
28     "_description": "La descripcion contiene número 12345 y caracteres especiales ñ.-/?%.",  
29     "_imageUrl": "https://images.freeimages.com/images/large-previews/4ec/banana-s-1326714.jpg",  
30     "_unit": "pieza",  
31     "_stock": 15,  
32     "_pricePerUnit": 1,  
33     "_category": "Verdura"  
34   },  
35   {  
36     "_uuid": "ff4d3d90-c23d-4cd5-8b74-e77dd72e539c",  
37     "_title": "Lechuga D",  
38     "_description": "Hello!",  
39     "_imageUrl": "https://images.freeimages.com/images/large-previews/4ec/banana-s-1326714.jpg",  
40     "_unit": "pieza",  
41     "_stock": 15,  
42     "_pricePerUnit": 80,  
43     "_category": "Verdura"  
44   },  
45   {  
46     "_uuid": "49e03f2e-34af-4080-94ce-99f15c600ea5",  
47     "_title": "Manzana E",  
48     "_description": "Aqui hice un update de producto",  
49     "_imageUrl": "https://images.freeimages.com/images/large-previews/4ec/banana-s-1326714.jpg",  
50     "_unit": "pieza",  
51     "_stock": 15,  
52     "_pricePerUnit": 1000,  
53     "_category": "Fruta"  
54   },  
55   {  
56     "_uuid": "79239489-79b6-4ef8-a96e-0b5963266fcc",  
57     "_title": "Calabaza F",  
58     "_description": "La descripcion contiene número 12345 y caracteres especiales ñ.-/?%.",  
59     "_imageUrl": "https://images.freeimages.com/images/large-previews/4ec/banana-s-1326714.jpg",  
60     "_unit": "pieza",  
61     "_stock": 15,  
62     "_pricePerUnit": 1,  
63     "_category": "Verdura"  
64   }  
65 ]
```

7. PUT /admin/products/:id

Permitiremos al administrador modificar los atributos de algún producto.

1. Verifica que el ID corresponde a un producto existente, de lo contrario regresa un estatus de error (404).

Practica 3 / Admin - Product ByID

PUT http://localhost:8080/admin/Y0-N0-3x15T0 Send

Params Auth Headers (10) Body Pre-req. Tests Settings

raw JSON

```
1 {"title": "Patata E",
2   "description": "La descripcion contiene número 12345 y caracteres
3   especiales ñ.-/?%.",
4   "imageUrl": "https://images.freeimages.com/images/large-previews/4ec/
5   banana-s-1326714.jpg",
6   "unit": "pieza",
7   "stock": 15,
8   "pricePerUnit": 1,
9   "category": "Verdura"}
```

Body 404 Not Found 25 ms 277 B Save Response

Pretty Raw Preview Visualize HTML

1 Producto con UUID: Y0-N0-3x15T0 no existe!

2. Validar que en el body se encuentren los atributos que nos interesan similar al POST.
3. Guardar los cambios al producto en el arreglo y regresar el status 200, con un mensaje con el nombre del producto que fue actualizado.

Practica 3 / Admin - Product ByID

PUT http://localhost:8080/admin/c367ccf6-ded8-4389-bfcf-d45274f84cf8 Send

Params Auth Headers (10) Body Pre-req. Tests Settings

raw JSON

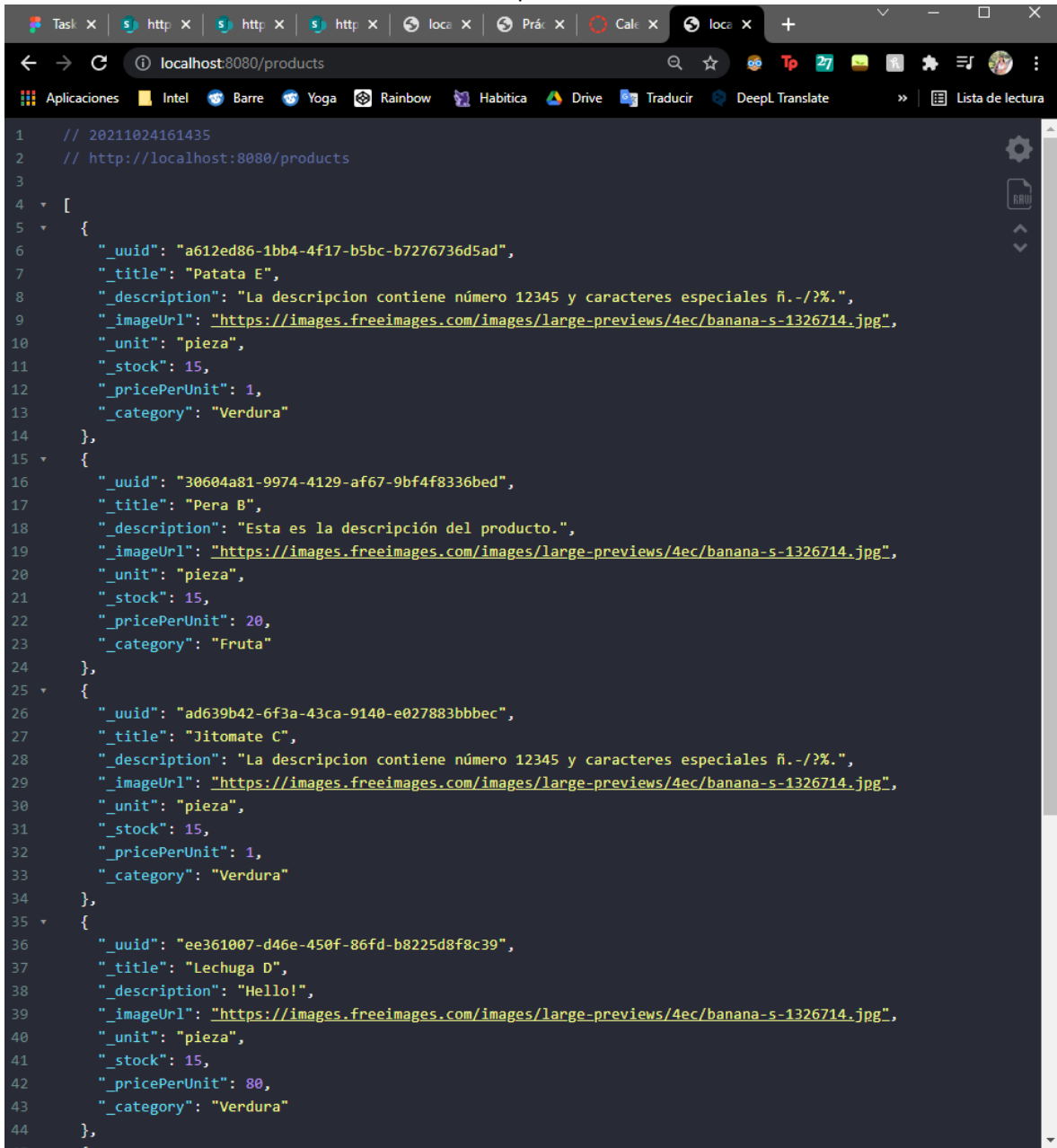
```
1 {"title": "Patata E",
2   "description": "La descripcion contiene número 12345 y caracteres
3   especiales ñ.-/?%.",
4   "imageUrl": "https://images.freeimages.com/images/large-previews/4ec/
5   banana-s-1326714.jpg",
6   "unit": "pieza",
7   "stock": 15,
8   "pricePerUnit": 1,
9   "category": "Verdura"}
```

Body 200 OK 9 ms 256 B Save Response

Pretty Raw Preview Visualize Text

1 User Platano A actualizado!

4. No te olvides de actualizar el archivo de productos.





The screenshot shows a web browser window with the address bar set to `localhost:8080/products`. The browser's developer tools are open, displaying a REST client interface. The request is a GET to `localhost:8080/products`, and the response is a JSON array of four product objects. The products are: 'Patata E' (category: Verdura), 'Pera B' (category: Fruta), 'Jitomate C' (category: Verdura), and 'Lechuga D' (category: Verdura). Each object contains fields for `_uuid`, `_title`, `_description`, `_imageUrl`, `_unit`, `_stock`, `_pricePerUnit`, and `_category`.




```
1 // 20211024161435
2 // http://localhost:8080/products
3
4 [
5   {
6     "_uuid": "a612ed86-1bb4-4f17-b5bc-b7276736d5ad",
7     "_title": "Patata E",
8     "_description": "La descripcion contiene número 12345 y caracteres especiales ñ.-/?%.",
9     "_imageUrl": "https://images.freeimages.com/images/large-previews/4ec/banana-s-1326714.jpg",
10    "_unit": "pieza",
11    "_stock": 15,
12    "_pricePerUnit": 1,
13    "_category": "Verdura"
14  },
15  {
16    "_uuid": "30604a81-9974-4129-af67-9bf4f8336bed",
17    "_title": "Pera B",
18    "_description": "Esta es la descripción del producto.",
19    "_imageUrl": "https://images.freeimages.com/images/large-previews/4ec/banana-s-1326714.jpg",
20    "_unit": "pieza",
21    "_stock": 15,
22    "_pricePerUnit": 20,
23    "_category": "Fruta"
24  },
25  {
26    "_uuid": "ad639b42-6f3a-43ca-9140-e027883bbbec",
27    "_title": "Jitomate C",
28    "_description": "La descripcion contiene número 12345 y caracteres especiales ñ.-/?%.",
29    "_imageUrl": "https://images.freeimages.com/images/large-previews/4ec/banana-s-1326714.jpg",
30    "_unit": "pieza",
31    "_stock": 15,
32    "_pricePerUnit": 1,
33    "_category": "Verdura"
34  },
35  {
36    "_uuid": "ee361007-d46e-450f-86fd-b8225d8f8c39",
37    "_title": "Lechuga D",
38    "_description": "Hello!",
39    "_imageUrl": "https://images.freeimages.com/images/large-previews/4ec/banana-s-1326714.jpg",
40    "_unit": "pieza",
41    "_stock": 15,
42    "_pricePerUnit": 80,
43    "_category": "Verdura"
44  },
45 ]
```


8. DELETE /admin/products/:id

Permitiremos al administrador borrar un producto específico.


1. Verifica que el ID corresponde a un producto existente, de lo contrario regresa un estatus de error (404).

Practica 3 / Admin - Delete  



Save   




DELETE  http://localhost:8080/admin/Y0-N0-3X15T0 Send

Params Auth Headers (7) Body Pre-req. Tests Settings Co

Headers  6 hidden







	KEY	VALUE	DES	...	Bulk Edit	Pres
<input checked="" type="checkbox"/>	x-auth	admin				
	Key	Value			Description	


Body   404 Not Found 20 ms 277 B Save Respon




Pretty Raw Preview Visualize HTML   


1 Producto con UUID: Y0-N0-3X15T0 no existe!

2. Guardar los cambios en el arreglo y regresar el status 200, con un mensaje con el nombre del producto que fue borrado.


← :  POST  PUT A.  GET P.  DEL A.  +  No Environment

Practica 3 / Admin - Delete 



Save   




DELETE  http://localhost:8080/admin/a612ed86-1bb4-4f17-b5bc-b7276736d5ad Send

Params Auth Headers (7) Body Pre-req. Tests Settings Co

Headers  6 hidden

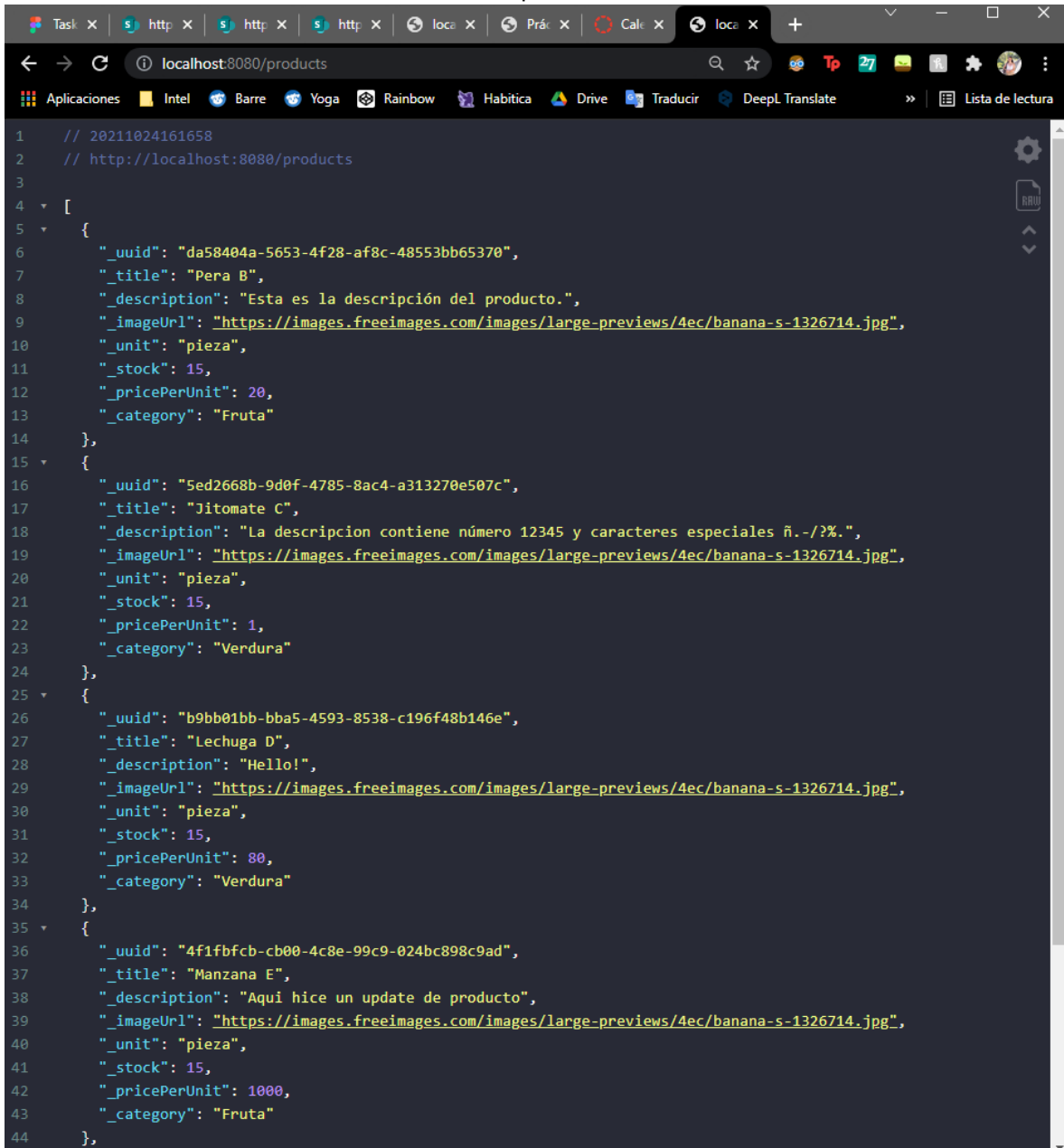
	KEY	VALUE	DES	...	Bulk Edit	Pres
<input checked="" type="checkbox"/>	x-auth	admin				
	Key	Value			Description	

Body   200 OK 13 ms 253 B Save Respor

Pretty Raw Preview Visualize Text   

1 User Patata E eliminado!

3. No te olvides de actualizar el archivo de productos.

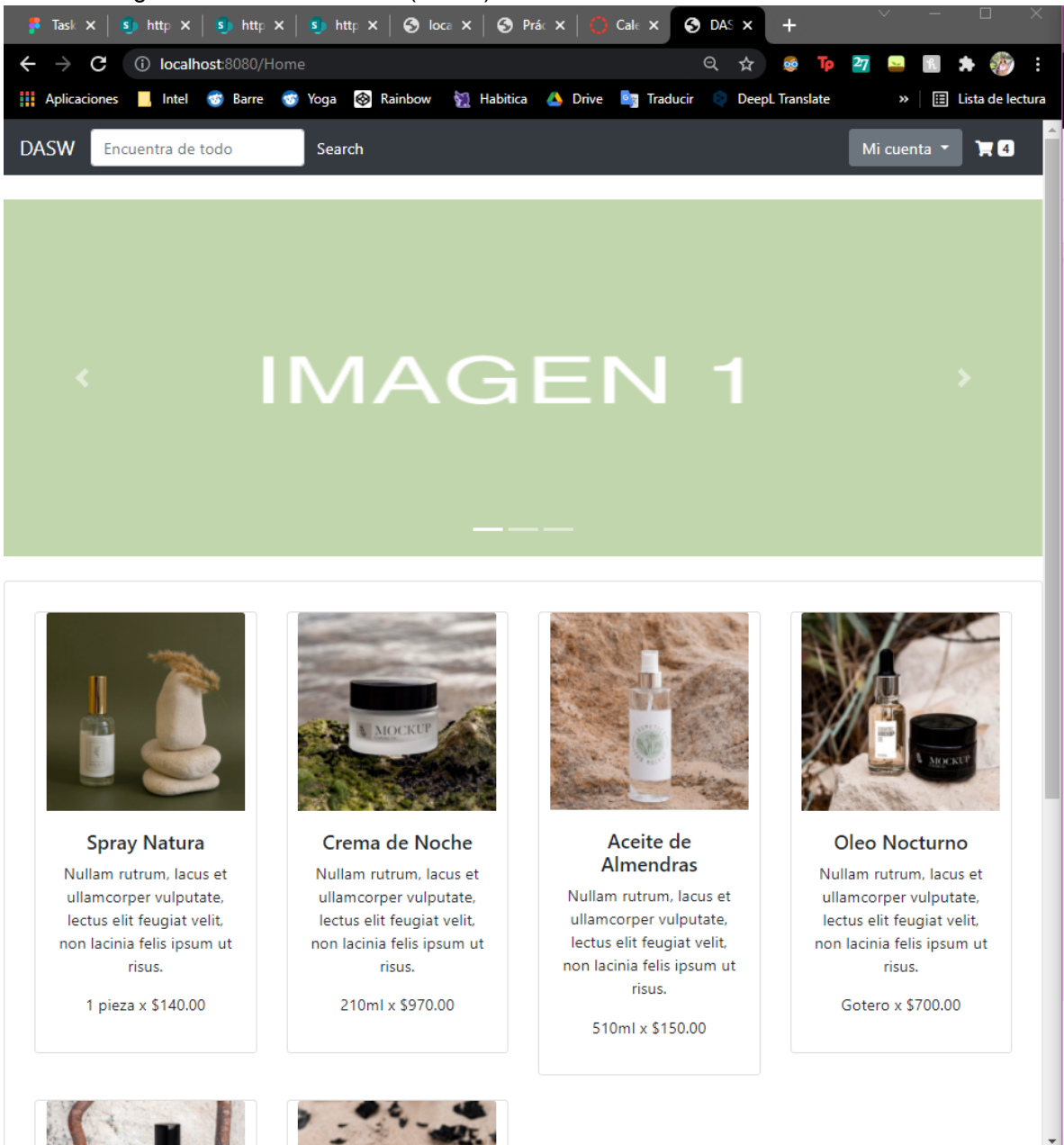


The screenshot shows a web browser window with the address bar set to `localhost:8080/products`. The browser's taskbar at the top includes icons for Task, http, http, http, loca, Prá, Cal, loca, and a plus sign. The browser's toolbar shows various extensions like Aplicaciones, Intel, Barre, Yoga, Rainbow, Habitica, Drive, Traducir, DeepL Translate, and a 'Lista de lectura' button. The main content area displays a REST client interface with a JSON array of product data. The JSON is as follows:

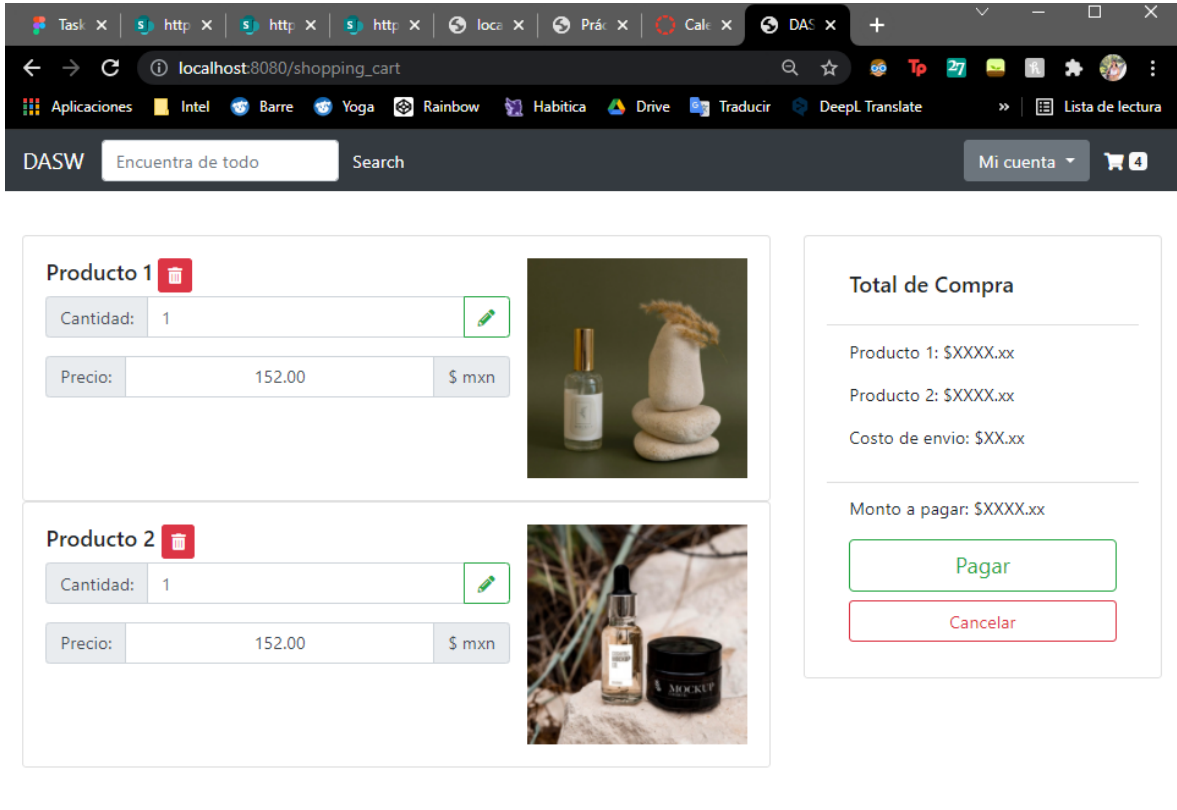
```
1 // 20211024161658
2 // http://localhost:8080/products
3
4 [
5   {
6     "_uuid": "da58404a-5653-4f28-af8c-48553bb65370",
7     "_title": "Pera B",
8     "_description": "Esta es la descripción del producto.",
9     "_imageUrl": "https://images.freeimages.com/images/large-previews/4ec/banana-s-1326714.jpg",
10    "_unit": "pieza",
11    "_stock": 15,
12    "_pricePerUnit": 20,
13    "_category": "Fruta"
14  },
15  {
16    "_uuid": "5ed2668b-9d0f-4785-8ac4-a313270e507c",
17    "_title": "Jitomate C",
18    "_description": "La descripcion contiene número 12345 y caracteres especiales ñ.-/?%.",
19    "_imageUrl": "https://images.freeimages.com/images/large-previews/4ec/banana-s-1326714.jpg",
20    "_unit": "pieza",
21    "_stock": 15,
22    "_pricePerUnit": 1,
23    "_category": "Verdura"
24  },
25  {
26    "_uuid": "b9bb01bb-bba5-4593-8538-c196f48b146e",
27    "_title": "Lechuga D",
28    "_description": "Hello!",
29    "_imageUrl": "https://images.freeimages.com/images/large-previews/4ec/banana-s-1326714.jpg",
30    "_unit": "pieza",
31    "_stock": 15,
32    "_pricePerUnit": 80,
33    "_category": "Verdura"
34  },
35  {
36    "_uuid": "4f1fbfcb-cb00-4c8e-99c9-024bc898c9ad",
37    "_title": "Manzana E",
38    "_description": "Aquí hice un update de producto",
39    "_imageUrl": "https://images.freeimages.com/images/large-previews/4ec/banana-s-1326714.jpg",
40    "_unit": "pieza",
41    "_stock": 15,
42    "_pricePerUnit": 1000,
43    "_category": "Fruta"
44  },
45 ]
```


9. GET /home, /shopping_cart

1. Al ingresar a la ruta de home (/home) se muestra home.html.



2. Al ingresar a la ruta del carrito de compras (/shopping_cart) se muestre shopping_cart.html.



Código:

1. Repositorio Github: https://github.com/LiliaLobato/DASW_Practicas/tree/main/Practica_3

Conclusiones:

La práctica no es complicada, básicamente ya habíamos hecho absolutamente todo lo que nos pidieron y la clase con la explicación fue equivalente a hacer la mitad. El problema que tuve es que me atrasé con las clases, me costó mucho trabajo hacer la descarga de programas + seteo de las herramientas + poner atención a la explicación. Del tiempo que le dediqué a la práctica, 90% fue volver a ver las clases, hacer con calma los ejercicios y asegurarme que mi configuración local estuviera funcionando.

Como recomendación, me gustaría que nos dijera con una clase de anticipación si tenemos que descargar algo.