



**INSTITUTO POLITÉCNICO NACIONAL**



**UNIDAD PROFECIONAL INTERDISCIPLINARIA**

**DE INGENIERÍA CAMPUS ZACATECAS IPN**

**PRACTICA I**

**ALUMNOS: Martha Dalila Cardona Serna**

**José Refugio Salinas Uribe**

**José Angel Montoya Zúñiga**

**Yavé Emmanuel Vargas Márquez**

**Giovanna Inosuli Campos Flores**

**Rodrigo Olmos Gómez**


**REPORTE DE PRUEBAS Y RESULTADOS**

**OBTENIDOS**

**SISTEMAS DISTRIBUIDOS**

**DOCENTE: Erika Paloma Sánchez-Femat**

Este reporte presenta las pruebas realizadas y los resultados obtenidos en esta práctica, cuyo objetivo fue diseñar y crear colecciones en MongoDB para almacenar los datos de una tienda en línea. Además, se desarrolló una API RESTful para interactuar con dichas colecciones, implementando las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) para cada una, y se probó su funcionalidad mediante el consumo de la API desde un cliente HTTP (Figura 1).

PUT	/users/{producto_id}	Update Producto	▼
DELETE	/users/{producto_id}	Delete Producto	▼
GET	/categorias/	Get Categorías	▼
POST	/categorias/	Create Categorías	▼
GET	/categorias/{categoria_id}	Get Categoría	▼
PUT	/categorias/{categoria_id}	Update Categoría	▼
DELETE	/categorias/{categoria_id}	Delete User	 ▼
GET	/pedidos/	Get Pedidos	▼
POST	/pedidos/	Create Pedido	▼
GET	/pedidos/{pedido_id}	Get Pedido	▼
PUT	/pedidos/{pedido_id}	Update Pedido	▼
DELETE	/pedidos/{pedido_id}	Delete Pedido	▼
GET	/clientes/	Get Clientes	▼
POST	/clientes/	Create Cliente	▼
GET	/clientes/{cliente_id}	Get Cliente	▼
PUT	/clientes/{cliente_id}	Update Cliente	▼

*Figura 1*

Durante el desarrollo se realizaron pruebas de todas las operaciones en cada una de las colecciones. A continuación, solo se pondrá a prueba una operación de cada colección para analizar los resultados obtenidos.

Primero se puso a prueba la operación GET (leer) en la colección producto, con la cual a través de el id del producto podemos obtener su información como nombre, descripción, precio y stock.

En la Figura 2 se muestra la información en la base de datos del producto que queremos obtener.

```
1  _id: ObjectId('66e05dc6eedb1da4acf367dd')
2  nombre : "Paleta Arcoíris"
3  descripcion : "Una colorida paleta de sabores frutales."
4  precio : 2.99
5  stock : 150
```

Figura 2

Después se hace la operación en la API desde un cliente HTTP (Figura 3).

The screenshot shows an HTTP client interface with the following components:

- Method and URL:** A blue button labeled "GET" and a text field containing the URL `/productos/{producto_id}` with the label "Get Producto".
- Parameters Section:**
  - A "Cancel" button in a red box.
  - A table with two columns: "Name" and "Description".
  - A parameter entry for `producto_id` with the type `string (path)`. It is marked as "required" with a red asterisk. The value `66e05dc6eedb1da4acf367dd` is entered in the adjacent text field.
- Servers Section:**
  - A note: "These operation-level options override the global server options."
  - A dropdown menu showing the value `/`.
- Buttons:** At the bottom, there are two buttons: "Execute" (blue) and "Clear" (light blue).

Figura 3

Aquí mismo nos muestra la información obtenida del producto “Paleta Arcoíris” (Figura 4).

```
Response body
{
  "nombre": "Paleta Arcoíris",
  "descripcion": "Una colorida paleta de sabores frutales.",
  "precio": 2.99,
  "stock": 150
}
```

Figura 4

Para poner a prueba la operación PUT (actualizar) se actualizó una categoría.

Observamos en la base de datos que la primera categoría es “Chilitos” (Figura 5).

```
1  _id: ObjectId('66e05faceedb1da4acf36802')      ObjectId
2  nombre : "Chilitos"                             String
3  descripcion : "golosina que combina sabores dulces, ácidos y picantes"      String

CANCEL  UPDAT

_id: ObjectId('66e05faceedb1da4acf36803')
nombre : "Gomitas"
descripcion : "Dulces masticables con diferentes formas y sabores, incluyendo version..."
```

Figura 5

En esta parte se colocó el id de la categoría que se quería actualizar, el nombre y descripción del nuevo dulce (Figura 6).

PUT /categorias/{categoria\_id} Update Categoria

Parameters

Cancel

Reset

Name

Description

categoria\_id

\*

required

string

(path)

66e05faceedb1da4acf36802

Request body

required

application/json

▼

```
{
  "nombre": "Caramelos",
  "descripcion": "dulces duros o blandos que se elaboran a partir de azúcar fundida que, al enfriarse, adquiere una textura firme"
}
```

Figura 6

Se puede observar que en la base de todos ya aparece la categoría actualizada (Figura 7).

```
_id: ObjectId('66e05faceedb1da4acf36802')
nombre : "Caramelos"
descripcion : "dulces duros o blandos que se elaboran a partir de azúcar fundida que,..."

_id: ObjectId('66e05faceedb1da4acf36803')
nombre : "Gomitas"
descripcion : "Dulces masticables con diferentes formas y sabores, incluyendo version..."
```

Figura 7

Ahora se pondrá a prueba la operación DELETE (eliminar) en la colección pedido. El pedido que se eliminará de la base de datos será el que termina con f8 su id (Figura 8).

```
_id: ObjectId('66e05f35eedb1da4acf367f7')
fecha : 2024-09-01T00:00:00.000+00:00
total : 59.99
▶ productos : Array (2)
```

1	_id: ObjectId('66e05f35eedb1da4acf367f8')	ObjectId
2	fecha : 2024-09-02T00:00:00.000+00:00	Date
3	total : 30.5	Double
4	▶ productos : Array (2)	Array

```
_id: ObjectId('66e05f35eedb1da4acf367f9')
fecha : Code('2024-09-03', {})
total : 80
▶ productos : Array (3)
```

Figura 8

Y para eliminarlo se coloca el id (Figura 9), y al ejecutarlo ya no debe aparecer en la base de datos (Figura 10).

**DELETE** /pedidos/{pedido\_id} Delete Pedido ^

**Parameters** Cancel

Name	Description
<b>pedido_id</b> * required string (path)	<input type="text" value="66e05f35eedb1da4acf367f8"/>

**Servers**

These operation-level options override the global server options.

Execute Clear

Figura 9

```
_id: ObjectId('66e05f35eedb1da4acf367f7')
fecha : 2024-09-01T00:00:00.000+00:00
total : 59.99
▸ productos : Array (2)
```

---

```
_id: ObjectId('66e05f35eedb1da4acf367f9')
fecha : Code('2024-09-03', {})
total : 80
▸ productos : Array (3)
```

---

```
_id: ObjectId('66e05f35eedb1da4acf367fa')
fecha : 2024-09-04T00:00:00.000+00:00
total : 12.99
▸ productos : Array (1)
```

---

Figura 10

Finalmente se probó la operación POST (crear o agregar) en la colección cliente.

Se observa que en la base de datos el ultimo cliente es “Elena” (Figura 11).

```
_id: ObjectId('66e05e9ceedb1da4acf367f4')  
nombre : "Pedro"  
apellido : "Morales"  
correo : "pedro.morales@email.com"
```

```
_id: ObjectId('66e05e9ceedb1da4acf367f5')  
nombre : "Elena"  
apellido : "Díaz"  
correo : "elena.diaz@email.com"
```

Figura 11

Aquí agregamos el nuevo cliente que es “José” (Figura 12).

The screenshot shows a REST client interface with a green header bar. The method is POST and the URL is /clientes/. The action is labeled 'Create Cliente'. Below the header, there is a 'Parameters' section with 'No parameters' listed. To the right of this section are 'Cancel' and 'Reset' buttons. Below the parameters is the 'Request body' section, which is marked as 'required'. A dropdown menu shows 'application/json'. The request body is a JSON object: { "nombre": "José", "apellido": "Salinas", "correo": "jose@gmail.com" }. Below the request body is the 'Response body' section, which shows the response JSON: { "id": "66e1241a4a7b1ba01d014937", "nombre": "José", "apellido": "Salinas", "correo": "jose@gmail.com" }. There are 'Download' and 'Copy' buttons next to the response body.

POST /clientes/ Create Cliente

Parameters

No parameters

Request body required application/json

```
{  
  "nombre": "José",  
  "apellido": "Salinas",  
  "correo": "jose@gmail.com"  
}
```

Response body

```
{  
  "id": "66e1241a4a7b1ba01d014937",  
  "nombre": "José",  
  "apellido": "Salinas",  
  "correo": "jose@gmail.com"  
}
```

Download

Figura 12

En la base de datos ya aparece el nuevo cliente (Figura 13).

<pre><b>_id:</b> ObjectId('66e05e9ceedb1da4acf367f4') <b>nombre :</b> "Pedro" <b>apellido :</b> "Morales" <b>correo :</b> "pedro.morales@email.com"</pre>
<pre><b>_id:</b> ObjectId('66e05e9ceedb1da4acf367f5') <b>nombre :</b> "Elena" <b>apellido :</b> "Díaz" <b>correo :</b> "elena.diaz@email.com"</pre>
<pre><b>_id:</b> ObjectId('66e1241a4a7b1ba01d014937') <b>nombre :</b> "José" <b>apellido :</b> "Salinas" <b>correo :</b> "jose@gmail.com"</pre>

Figura 13