

Desarrollo Web

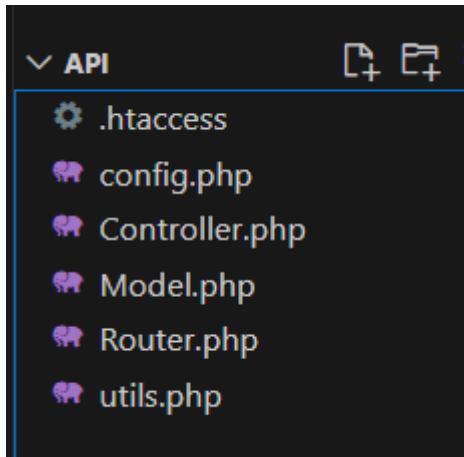
Entorno Sevidor

Tema 5

Tarea Presencial/Examen

by: Christian Martín Infantes

Sistema de archivos : MVC



.htaccess: para establecer y configurar las directivas del xampp.

config.php: Conexión a la base de datos.

Controller.php: El controlador del proyecto API.

Model.php: El modelo del proyecto API.

Router.php: El enrutador o index del proyecto.

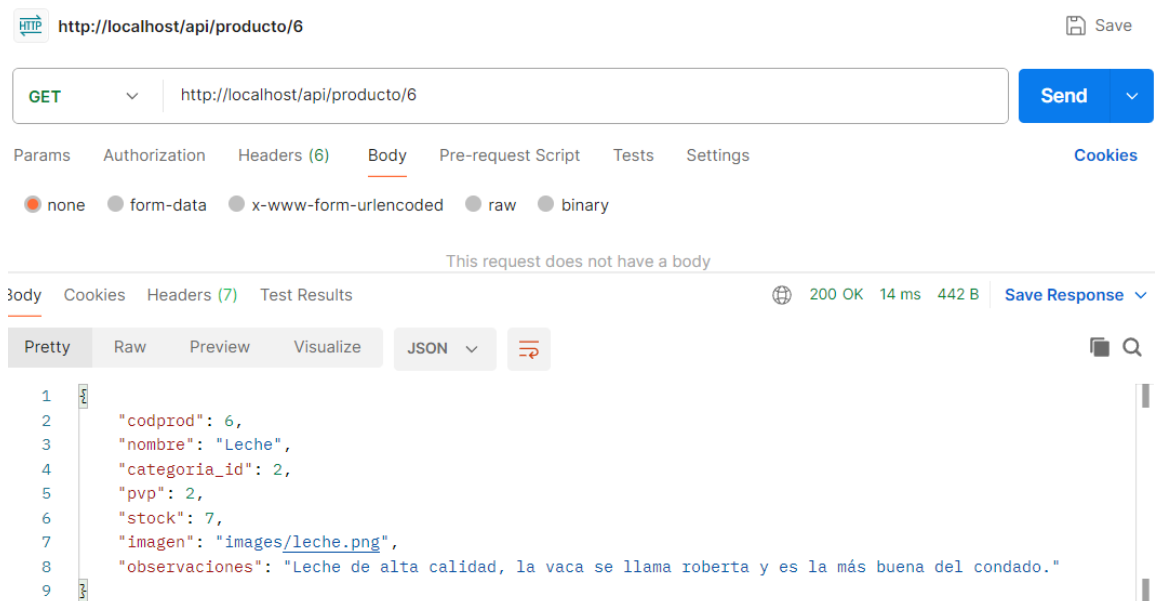
utils.php: Diversas utilidades.

Estos archivos se encuentran en “C:\xampp\htdocs\api” para poder funcionar correctamente.

POSTMAN

GET 1

Buscar por id: <http://localhost/api/producto/1>



Pasamos el id por la URL, lo separamos por “/” e identificamos el id.
Al saber que es un método GET con un id, llamaremos al controlador que a su vez llama al modelo para realizar la consulta sql, luego devuelve los datos al controlador y este muestra el json.

GET 2

Mostrar todo: <http://localhost/api/producto>

GET

▼

http://localhost/api/producto

Send

ParamsAuthorizationHeaders (6)BodyPre-request ScriptTestsSettingsCoc

none

form-data

x-www-form-urlencoded

raw

binary

This request does not have a body

bodyCookiesHeaders (7)Test Results200 OK34 ms1.03 KBSave Respor

PrettyRawPreviewVisualizeJSON▼

16"stock": 10,

17"imagen": "images/queso.png",

18"observaciones": "Producto duradero. 1.5 meses"

19},

20{

21"codprod": 3,

22"nombre": "CocaCola",

23"categoria_id": 3,

24"pvp": 1.5,

25"stock": 20,

26"imagen": "images/cola.png",

27"observaciones": "Bebida duradera. 5 meses"

28},

29{

30"codprod": 4,

31"nombre": "Guisantes",

32"categoria_id": 4,

33"pvp": 0.85,

34"stock": 5,

Pasamos el id por la URL, lo separamos por "/" y al no detectar nada especial, y saber que es un método GET, por defecto mostraremos todo el listado de producto, de la misma forma, llamaremos al controlador que a su vez llama al modelo para realizar la consulta sql, luego devuelve los datos al controlador y este muestra el json.

GET 3

Listado paginado y ordenado por nombre de categoría en orden creciente

<http://localhost/api/producto?page=2&perPage=2>

The screenshot shows a REST client interface with a GET request to `http://localhost/api/producto?page=2&perPage=2`. The request parameters are `page=2` and `perPage=2`. The response is a JSON object with the following structure:

```
1 {
2   "pagina": 2,
3   "por_pagina": 2,
4   "total_paginas": 3,
5   "total_productos": 5,
6   "productos": [
7     {
8       "codprod": 2,
9       "nombre": "Queso",
10      "categoria_id": 2,
11      "pvp": 12,
12      "stock": 10,
13      "imagen": "images/queso.png",
14      "observaciones": "Producto duradero. 1.5 meses",
15      "nombre_categoria": "Lácteos"
16    },
17    ...
18  ]
19 }
```

Este es un poco más especial que el resto, El funcionamiento es el mismo, pero este recibe un parámetros por la url, estos parámetros son: La pagina en la que se quiere ubicar y cuántos resultados habrá por páginas.

Además está ordenado con por el nombre de la categoría de forma ascendente, esto logrado gracias al JOIN y al ORDER BY en la consulta sql.

Este método del controlador tiene dos métodos asociados en el modelo, uno para hacer la consulta y otro para identificar la cantidad total de registros/productos

POST

Crear producto: <http://localhost/api/producto>

POST

http://localhost/api/producto

Send

ParamsAuthorizationHeaders (8)BodyPre-request ScriptTestsSettings

none

form-data

x-www-form-urlencoded

raw

binary

Text

```
1 {
2   "nombre": "Leche",
3   "categoria_id": 2,
4   "pvp": 2,
5   "stock": 7,
6   "imagen": "images/leche.png",
7   "observaciones": "Leche de alta calidad, la vaca se llama roberta y es la más buena del condado."
8 }
```

BodyCookiesHeaders (7)Test Results

201 Created10 ms260 BSave Res

PrettyRawPreviewVisualizeJSON

```
1 {
2   "id": "7"
3 }
```

Para realizar la llamada con el método post, debemos ir a Body, y seleccionamos en raw, y escribimos el nuevo producto en formato json.

Ahora al detectar que el método es un POST con el switch del router.php. Esto iniciará / llamará a los métodos para crear un nuevo producto, que es resumidas cuentas es una consulta sql con un insert into

PUT

Editar un producto: <http://localhost/api/producto/7>

PUT

▼

http://localhost/api/producto/7

Params

Authorization

Headers (8)

Body ●

Pre-request Script

Tests

Settings

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

Text ▼

```
1 {
2   .... "nombre": "pepsiman",
3   .... "categoria_id": 3,
4   .... "pvp": 1.8,
5   .... "stock": 30,
6   .... "imagen": "images/pepsiman.png",
7   .... "observaciones": "El hombre bebida, el noble defensor de la pepsi."
8 }
```

Body

Cookies

Headers (7)

Test Results

🌐

Pretty

Raw

Preview

Visualize

JSON ▼

↻

```
1 {
2   "message": "Producto editado"
3 }
```

Para realizar la llamada con el método PUT, debemos ir a Body, y seleccionamos en raw, y escribimos el nuevo producto en formato json, Además en la URL debemos añadir el id que queremos editar

Ahora al detectar que el método es un PUT con el switch del router.php. Esto iniciará / llamará a los métodos para crear un editar el producto, que es resumidas cuentas es una consulta sql con un UPDATE into

DELETE

Eliminar un producto: <http://localhost/api/producto>

DELETE

<http://localhost/api/producto/7>

Send

ParamsAuthorizationHeaders (8)Body ●Pre-request ScriptTestsSettingsCookies

Query Params

	Key	Value	Bulk Edit
	Key	Value	

BodyCookiesHeaders (7)Test Results200 OK39 ms266 BSave Response

PrettyRawPreviewVisualizeJSON

1

2

3

"message": "Borrado"

Para realizar la llamada con el método DELETE en la URL debemos añadir el id que queremos eliminar

Ahora al detectar que el método es un DELETE con el switch del router.php. Esto iniciará / llamará a los métodos para crear un editar el producto, que es resumidas cuentas es una consulta sql con un DELETE

GIT HUB

<https://github.com/NeoChrisMIN/apiservidorexamentema5.git>

BASE DE DATOS

```
-- phpMyAdmin SQL Dump
-- version 5.2.1
-- https://www.phpmyadmin.net/
--
-- Servidor: 127.0.0.1
-- Tiempo de generación: 18-04-2024 a las 19:36:25
-- Versión del servidor: 10.4.28-MariaDB
```

```

-- Versión de PHP: 8.2.4

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
START TRANSACTION;
SET time_zone = "+00:00";

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- Base de datos: `bdproductos`
--

--
-- Estructura de tabla para la tabla `categoria`
--

CREATE TABLE `categoria` (
  `id` int(11) NOT NULL,
  `nombre` varchar(50) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

--
-- Volcado de datos para la tabla `categoria`
--

INSERT INTO `categoria` (`id`, `nombre`) VALUES
(1, 'Fruta'),
(2, 'Lácteos'),
(3, 'Bebidas'),
(4, 'Legumbres');

--
-- Estructura de tabla para la tabla `productos`
--

```



```

CREATE TABLE `productos` (
  `codprod` int(11) NOT NULL,
  `nombre` varchar(100) NOT NULL,
  `categoria_id` int(11) NOT NULL,
  `pvp` float NOT NULL DEFAULT 0,
  `stock` int(11) NOT NULL DEFAULT 0,
  `imagen` varchar(200) DEFAULT NULL,
  `observaciones` text DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

--
-- Volcado de datos para la tabla `productos`
--

INSERT INTO `productos` (`codprod`, `nombre`, `categoria_id`, `pvp`,
`stock`, `imagen`, `observaciones`) VALUES
(1, 'Manzana', 1, 7.5, 10, 'images/manzana.png', 'Fruta verde.
Duradera. 3-5 días'),
(2, 'Queso', 2, 12, 10, 'images/queso.png', 'Producto duradero. 1.5
meses'),
(3, 'CocaCola', 3, 1.5, 20, 'images/cola.png', 'Bebida duradera. 5
meses'),
(4, 'Guisantes', 4, 0.85, 5, 'images/guisantes.png', 'Legumbre verde.
Poco duradera. 1-2 días'),
(6, 'Leche', 2, 2, 7, 'images/leche.png', 'Leche de alta calidad, la
vaca se llama roberta y es la más buena del condado.');
```

```

--
-- Índices para tablas volcadas
--

--
-- Indices de la tabla `categoria`
--
ALTER TABLE `categoria`
  ADD PRIMARY KEY (`id`);

--
-- Indices de la tabla `productos`
--
ALTER TABLE `productos`

```

```
ADD PRIMARY KEY (`codprod`),
ADD KEY `categoria_id` (`categoria_id`);

--
-- AUTO_INCREMENT de las tablas volcadas
--
--
-- AUTO_INCREMENT de la tabla `categoria`
--
ALTER TABLE `categoria`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=5;

--
-- AUTO_INCREMENT de la tabla `productos`
--
ALTER TABLE `productos`
  MODIFY `codprod` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=8;

--
-- Restricciones para tablas volcadas
--
--
-- Filtros para la tabla `productos`
--
ALTER TABLE `productos`
  ADD CONSTRAINT `productos_ibfk_1` FOREIGN KEY (`categoria_id`)
REFERENCES `categoria` (`id`);
COMMIT;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

