Desarrollo de Software VII

Laboratorio # 17 – Prof. Regis Rivera

Objetivo: Introducción a la programación orientada a objetos en PHP (segunda parte).

I. ¿Qué es la API REST?

Para definir "REST API", primero debemos saber qué es "REST" y qué es "API".

REST significa "Transferencia de estado representativa". Es un concepto o arquitectura para gestionar información a través de Internet. Los conceptos REST se denominan recursos. Una representación de un recurso debe ser apátrida. Generalmente está representado por JSON .

API significa "interfaz de programación de aplicaciones". Es un conjunto de reglas que permite que una aplicación de software se comunique con otra. Esas "reglas" pueden incluir operaciones de creación, lectura, actualización y eliminación.

¿Por qué necesitamos API REST?

En muchas aplicaciones, REST API es una necesidad porque esta es la forma más ligera de crear, leer, actualizar o eliminar información entre diferentes aplicaciones a través de Internet o el protocolo HTTP. Esta información se presenta al usuario en un instante, especialmente si utiliza JavaScript para representar los datos en una página web.

¿Dónde se utiliza la API REST?

REST API puede ser utilizado por cualquier aplicación que pueda conectarse a internet. Si los datos de una aplicación se pueden crear, leer, actualizar o eliminar utilizando otra aplicación, generalmente significa que se utiliza una API REST.

II. Herramientas para crear y probar este API Rest

Se sugiere contar por lo mínimo con lo siguiente:

2.1 Navegador Web
2.2 Editor de código
2.3 Xampp
- MySQL (MariaDB)
- Apache
- Php

2.4 Postman
- Herramienta para probar los API Rest creados
- Ver más detalles en el siguiente punto

III. Descargar y configurar PostMan

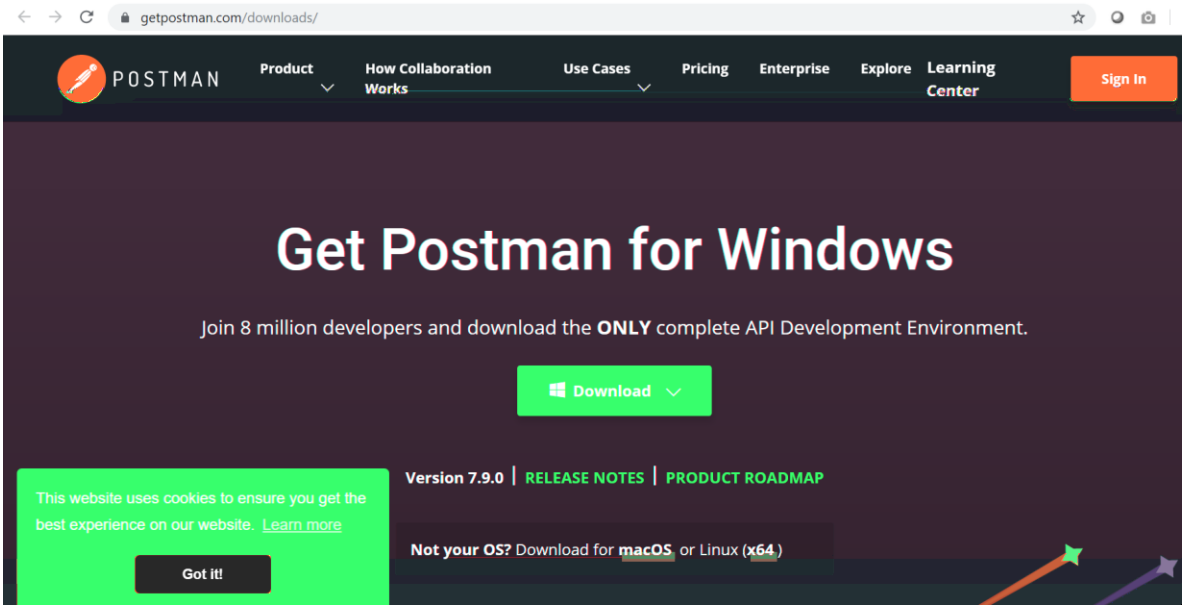
Postman es una herramienta que se utiliza, sobre todo, para el testing de API REST, aunque también admite otras funcionalidades que se salen de lo que engloba el testing de este tipo de sistemas.

Gracias a esta herramienta, además de probar, consumir y depurar API REST, podremos monitorizarlas, escribir pruebas automatizadas para ellas, documentarlas, mockearlas, simularlas, etc.

3.1 Descargar Postman

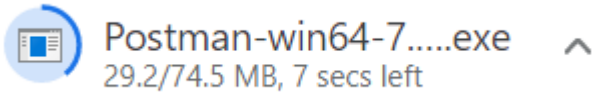
Ir a https://www.getpostman.com/downloads/

Descargar la última vesión de acuerdo a su SO:

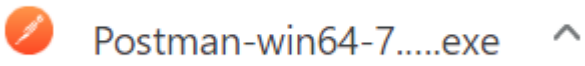


3.2 Iniciar instalador de Postman

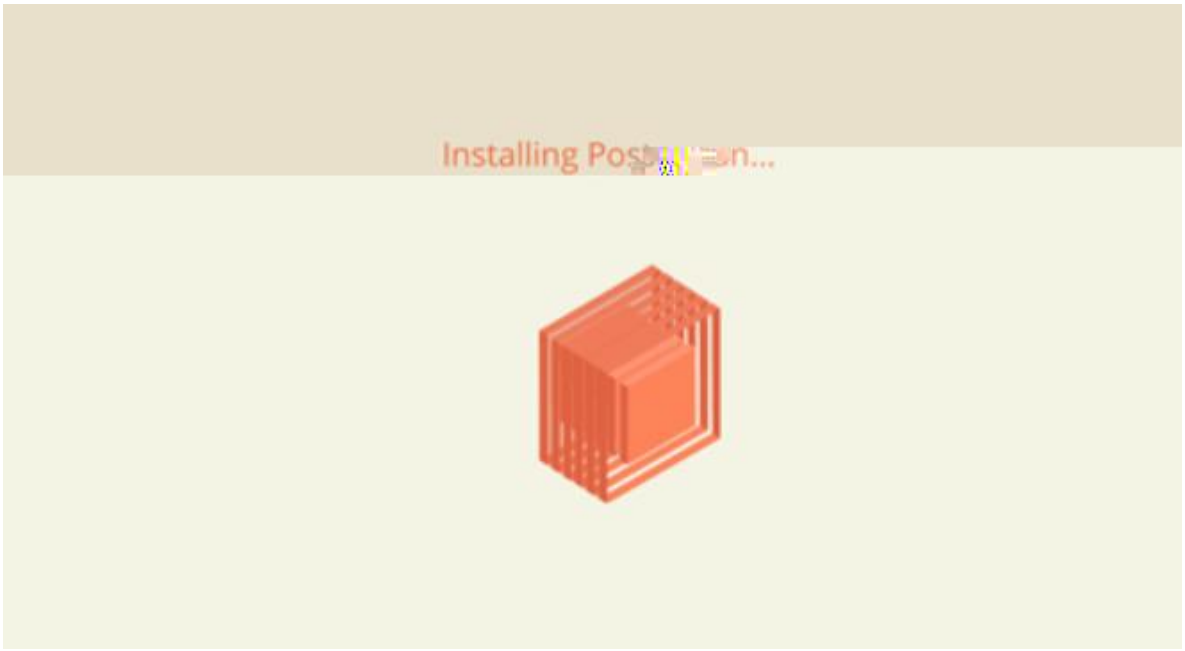
Ejemplo de instalación para Windows 10 en 64bits, de igual forma es de referencia para instalaciones en otras versiones u otros sistemas operativos:



Para ello, abrir el instalador de postman descargado:

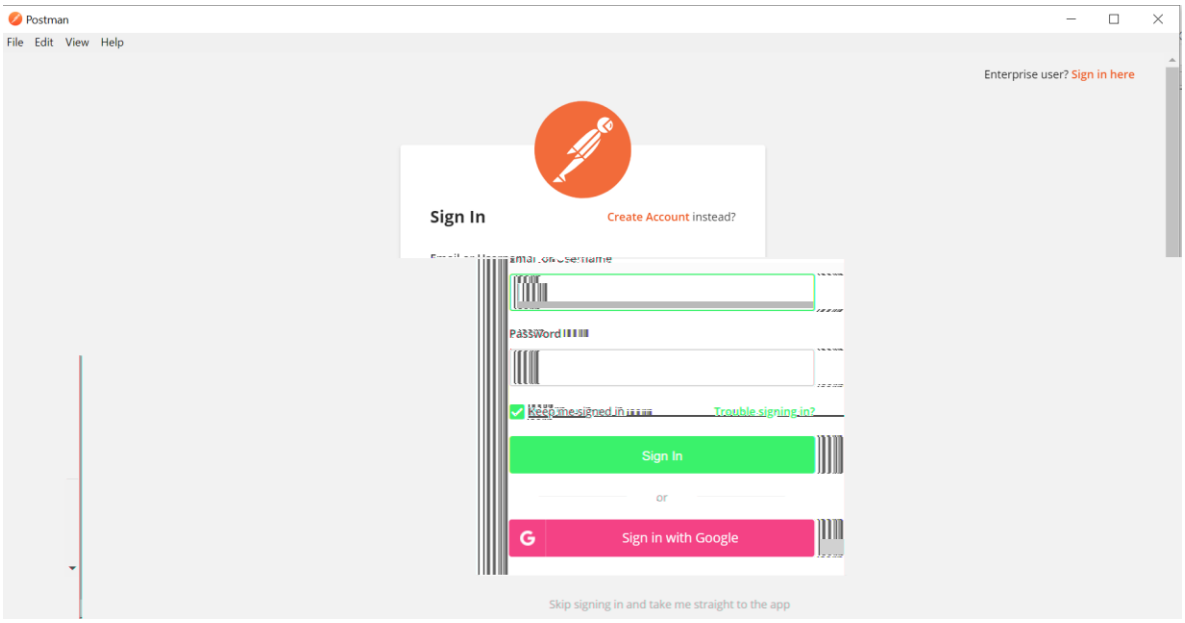


Pantalla de inicio al instalar



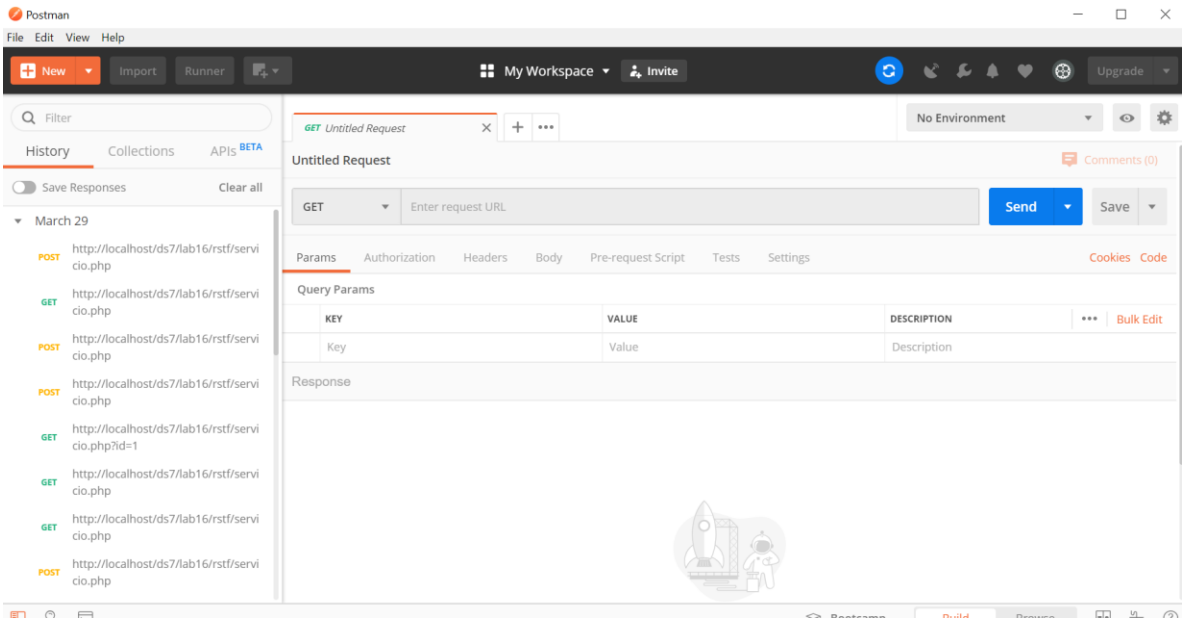
3.3 Ingreso por primera vez a Postman

Colocar credenciales de postman o crear credenciales nuevas. También está la opción de utilizar las redes sociales como credenciales de acceso



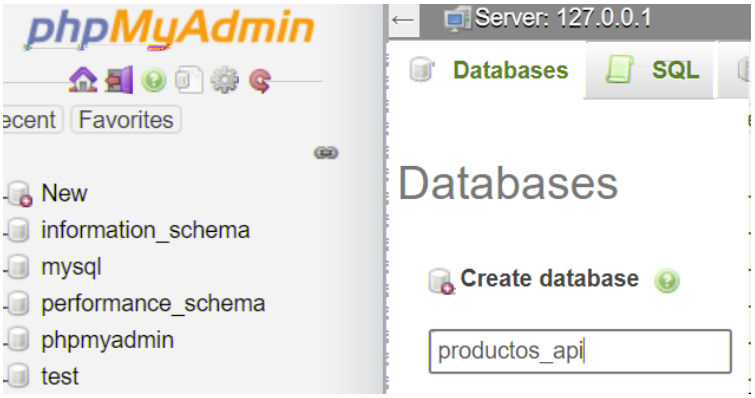
3.4 Vista inicial de Postman

Espacio de trabajo de postman

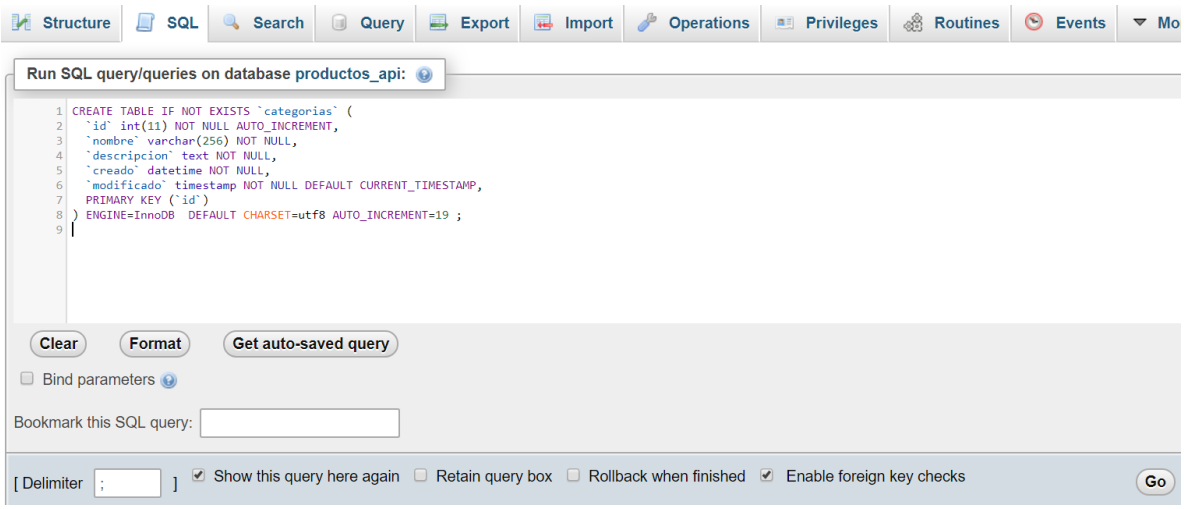


IV. Configurar la base de datos

Usando PhpMyAdmin, cree una nueva base de datos productos_api. Después de eso, ejecutar las siguientes consultas SQL para crear nuevas tablas con datos de muestra.



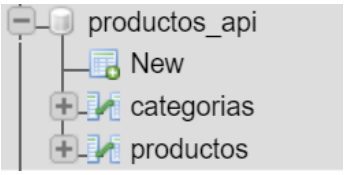
4.1 Crear tabla de categorías



4.2 Datos de volcado para la tabla de categorías

V. Tabla de productos

Hasta el momento, van 2 tablas:



VI. Datos de volcado para la tabla de productos

Volcar los siguientes registros en la tabla productos:

```
INSERT INTO `productos` (`id`, `nombre`, `descripcion`, `precio`, `categoria_id`, `creado
`, `modificado`) VALUES
(1, 'Samsung Galaxy Note 10 Plus', ' El mejor celular Samsung en este momento y el más co
mpleto', '1110', 2, '2019-06-01 01:12:26', '2019-05-31 17:12:26'),
(2, ' Huawei P30 Pro', ' Cámara que ofrece zoom óptico 5X', '719', 2, '2019-06-
01 01:12:26', '2019-05-31 17:12:26'),
(3, 'Samsung Galaxy S10', '¿Qué tal este smartphone?', '800', 3, '2019-06-
01 01:12:26', '2019-05-31 17:12:26'),
(6, 'Ropa para levantar Pesas', 'De tipo Bench Shirt', '29', 1, '2019-06-
01 01:12:26', '2019-05-31 02:12:21'),
(7, 'Laptop Lenovo', 'Mi socio comercial', '399', 2, '2019-06-01 01:13:45', '2019-05-
31 02:13:39'),
(8, 'Samsung Galaxy Tab 10.1', 'Buena Tableta', '259', 2, '2019-06-01 01:14:13', '2019-
05-31 02:14:08'),
(9, 'Reloj Spalding', 'Mi reloj deportivo', '199', 1, '2019-06-01 01:18:36', '2019-05-
31 02:18:31'),
(10, ' Smart Watch de Sony', '¡El reloj inteligente más genial!', '300', 2, '2019-06-
06 17:10:01', '2019-06-05 18:09:51'),
(11, 'Huawei Y9s', 'Para fines de prueba.', '350', 2, '2019-06-06 17:11:04', '2019-06-
05 18:10:54'),
(12, 'Camisa Abercrombie Lake Arnold', 'Perfecto como regalo!',
```

VII. Conectarse a la base de datos

El siguiente código muestra las credenciales de la base de datos y un método para obtener una conexión de base de datos mediante PDO.

1. Crear una carpeta llamada api (bajo htdocs). Abrir la carpeta de la API. Crear carpeta de configuración llamada configuracion. Abrir la carpeta de configuración. Crear archivo un llamado conexion.php. Colocar el siguiente código dentro de él.

```
<?php
class Conexion{

    // especificar las credenciales de base de datos
    private $host = "localhost";
    private $db_name = "productos_api";
    private $username = "root";
    private $password = "";
    public $conn;

    // obtener la conexion de la base de datos
    public function obtenerConexion(){

        $this->conn = null;

        try{
            $this->conn = new PDO("mysql:host=" . $this->host . ";dbname=" . $this-
>db_name, $this->username, $this->password);
            $this->conn->exec("set names utf8");
        }catch(PDOException $exception){
            echo "Error de conexion a base de datos: " . $exception->getMessage();
        }

        return $this->conn;
    }
}
?>
```

VIII. Leer productos

8.1 Objeto del producto

Crear código de una clase llamada Producto con varias de sus propiedades. Esta cuenta con un método de constructor que aceptará la conexión de la base de datos. Utilizaremos esta clase para leer datos de la base de datos. Abrir la carpeta API. Crear carpeta de objetos llamada objetos. Abrir carpeta de objetos. Crear archivo producto.php . Coloque el siguiente código dentro de él.

```php
<?php
class Producto{

    // conexion de base de datos y tabla productos
    private $conn;
    private $nombre_tabla = "productos";

    // atributos de la clase
    public $id;
    public $nombre;
    public $descripcion;
    public $precio;
    public $categoria_id;
    public $categoria_desc;
    public $creado;

    // constructor con $db como conexion a base de datos
    public function __construct($db){
        $this->conn = $db;
    }
}
?>
```

8.2 Crear archivo para leer productos

El siguiente código muestra encabezados sobre quién puede leer este archivo y qué tipo de contenido devolverá.

En este caso, nuestro archivo leer.php puede ser leído por cualquier persona (asterisco * significa todo) y devolverá datos en formato JSON.

Abrir la carpeta de la API. Crear carpeta de producto llamada producto. Abrir la carpeta del producto. Crear el archivo leer.php . Coloque el siguiente código dentro de él.

```php
<?php
//encabezados obligatorios
header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");

// la conexion a la base de datos estará aqui
```

8.3 Conectarse a la base de datos y tabla de productos

En el siguiente código, incluimos los archivos conexion.php y producto.php . Estos son los archivos que creamos anteriormente.

Necesitamos usar el método obtenerConexion() de la clase Conexion para obtener una conexión a la base de datos. Pasamos esta conexión a la clase de Producto .

Reemplazar el comentario //la conexión de la base de datos estará aquí del archivo leer.php con el siguiente código.

```php
<?php
//encabezados obligatorios
header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");

// incluir archivos de conexion y objetos
include_once '../configuracion/conexion.php';
include_once '../objetos/producto.php';

// inicializar base de datos y objeto producto
$conex = new Conexion();
$db = $conex->obtenerConexion();

// inicializar objeto
$producto = new Producto($db);

// lectura de productos estará aqui
```

8.4 Leer productos de la base de datos

En el siguiente código, utilizamos el método read() de la clase Producto para leer datos de la base de datos. A través de la variable $num , verificamos si se encontraron registros.

Si se encuentran registros, lo recorremos utilizando el ciclo while, agregamos cada registro a la matriz $products_arr , establecemos un código de respuesta 200 OK y se lo mostramos al usuario en formato JSON.

Reemplazar el comentario `//lectura de productos estará aquí` del archivo read.php con el siguiente código.

```php
// query productos
$stmt = $producto->read();
$num = $stmt->rowCount();

// verificar si hay mas de 0 registros encontrados
if($num>0){

    // arreglo de productos
    $products_arr=array();
    $products_arr["records"]=array();

    // obtiene todo el contenido de la tabla
    // fetch() es mas rapido que fetchAll()
    while ($row = $stmt->fetch(PDO::FETCH_ASSOC)){
        // extraer fila
        // esto creara de $row['nombre'] a
        // solamente $nombre
        extract($row);

        $product_item=array(
            "id" => $id,
            "nombre" => $nombre,
            "descripcion" => html_entity_decode($descripcion),
            "precio" => $precio,
            "categoria_id" => $categoria_id,
            "categoria_desc" => $categoria_desc
        );

        array_push($products_arr["records"], $product_item);
    }

    // asignar codigo de respuesta - 200 OK
    http_response_code(200);

    // mostrar productos en formato json
    echo json_encode($products_arr);
}

// no hay productos encontrados estará aqui
```

8.5 Agregar el método "read ()" del producto

Utilizamos el método read() en la sección anterior, pero aún no existe en la clase Producto . Necesitamos agregar este método read() . El siguiente código muestra la consulta para obtener registros de la base de datos.

Abrir carpeta de objetos . Abrir el archivo producto.php . Colocar el siguiente código dentro de la clase de Producto . Para asegurarse de haberlo agregado correctamente, coloque el código antes de la última llave de cierre.

```
    // leer productos
    function read(){

        // query para seleccionar todos
        $query = "SELECT
                c.nombre as categoria_desc, p.id, p.nombre, p.descripcion, p.precio
, p.categoria_id, p.creado
            FROM
                " . $this->nombre_tabla . " p
                LEFT JOIN
                    categorias c
                        ON p.categoria_id = c.id
            ORDER BY
                p.creado DESC";

        // sentencia para preparar query
        $stmt = $this->conn->prepare($query);

        // ejecutar query
        $stmt->execute();

        return $stmt;
    }
```

8.6 Dígale al usuario que no se encontraron productos

Si la variable $num tiene un valor de cero o negativo, significa que no hay registros devueltos desde la base de datos. Necesitamos decirle al usuario sobre esto.

En el siguiente código, configuramos el código de respuesta en 404: no encontrado y un mensaje que dice NO se encontraron productos.

Reemplazar el comentario `//no hay productos encontrados estará aquí` del archivo read.php con el siguiente código.

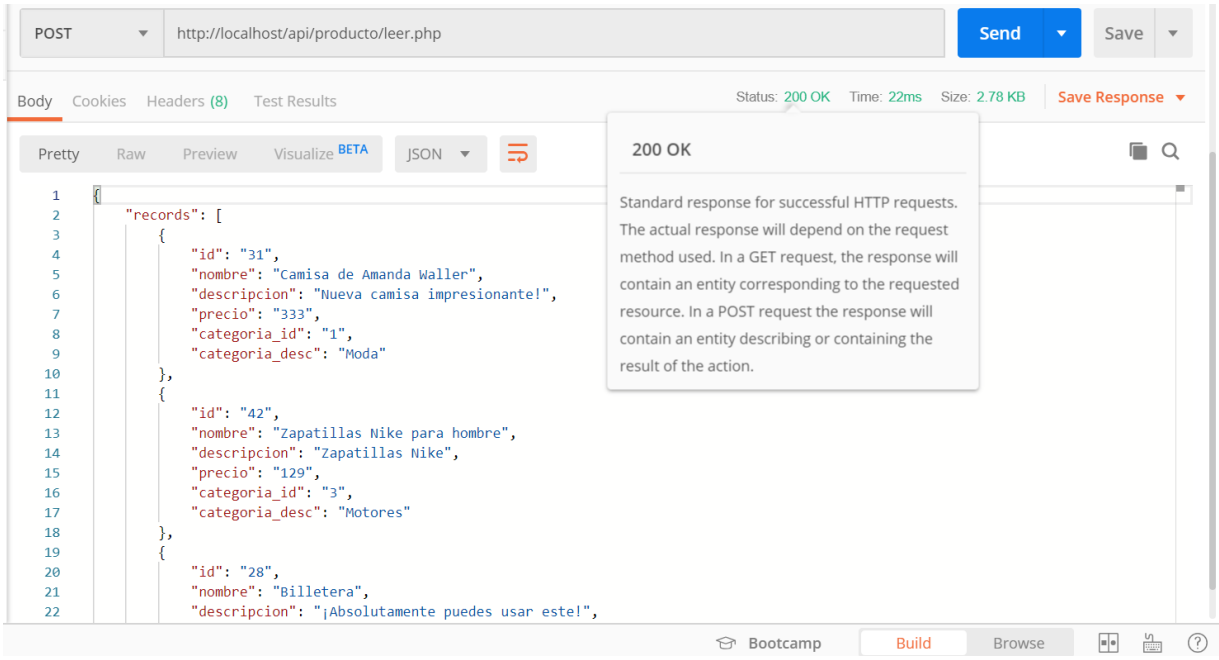
```php
else{

    // asignar codigo de respuesta - 404 No encontrado
    http_response_code(404);

    // informarle al usuario que no se encontraron productos
    echo json_encode(
        array("message" => "No se encontraron productos.")
    );
}
```

8.7 Salida

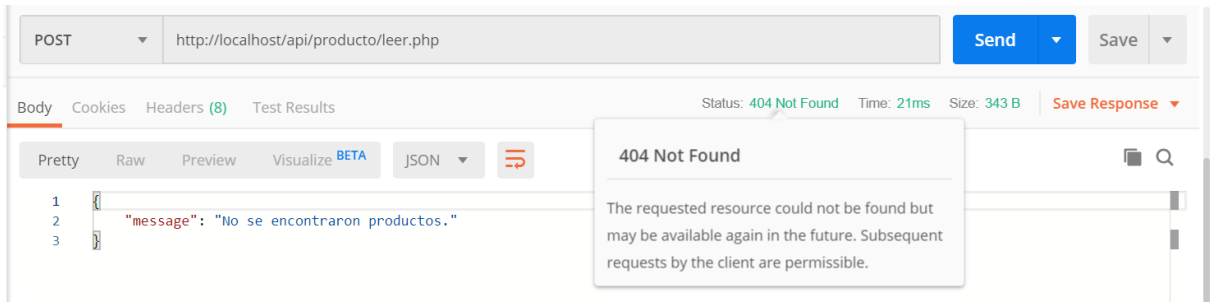
Debe usar POSTMAN para probar nuestra API. Ingresar la siguiente la URL de solicitud.

http://localhost/api/producto/leer.php Click en el botón azul "Send" bajo el tipo de consulta "POST"

Salida si hay datos de productos:



Salida si no hay datos de productos:



IX. Crear producto

9.1 Crear archivo create.php

Abrir la carpeta producto. Crear un nuevo archivo crear.php . Abrir ese archivo y colocar el siguiente código dentro de él.

```php
<?php
// encabezados obligatorios
header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");
header("Access-Control-Allow-Methods: POST");
header("Access-Control-Max-Age: 3600");
header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-
Headers, Authorization, X-Requested-With");

// obtener conexion de base de datos
include_once '../configuracion/conexion.php';

// instanciar el objeto producto
include_once '../objetos/producto.php';

$conex = new Conexion();
$db = $conex->obtenerConexion();

$producto = new Producto($db);

// obtener los datos
$data = json_decode(file_get_contents("php://input"));

// asegurar que los datos no esten vacios
if(
    !empty($data->nombre) &&
    !empty($data->precio) &&
    !empty($data->descripcion) &&
    !empty($data->categoria_id)
){
     // asignar valores de propiedad a producto
    $producto->nombre = $data->nombre;
    $producto->precio = $data->precio;
    $producto->descripcion = $data->descripcion;
    $producto->categoria_id = $data->categoria_id;
    $producto->creado = date('Y-m-d H:i:s');
```

```
    // crear el producto
    if($producto->crear()){

        // asignar codigo de respuesta - 201 creado
        http_response_code(201);

        // informar al usuario
        echo json_encode(array("message" => "El producto ha sido creado."));
    }

    // si no puede crear el producto, informar al usuario
    else{

        // asignar codigo de respuesta - 503 servicio no disponible
        http_response_code(503);

        // informar al usuario
        echo json_encode(array("message" => "No se puede crear el producto."));
    }
}

// informar al usuario que los datos estan incompletos
else{

    // asignar codigo de respuesta - 400 solicitud incorrecta
    http_response_code(400);

    // informar al usuario
    echo json_encode(array("message" => "No se puede crear el producto. Los datos
 están incompletos."));
}
?>
```

9.2 Método crear() del producto

Abrir carpeta objetos. Abrir el archivo producto.php . La sección anterior no funcionará sin el siguiente código dentro de la clase Producto (objectos / producto.php).

Este código corresponde al método crear() dentro de la clase Producto

```
    // crear producto
    function crear(){

    // query para insertar un registro
    $query = "INSERT INTO
                " . $this->nombre_tabla . "
            SET
            nombre=:nombre, precio=:precio, descripcion=:descripcion, categoria
_id=:categoria_id, creado=:creado";

    // preparar query
    $stmt = $this->conn->prepare($query);

    // sanitize
    $this->nombre=htmlspecialchars(strip_tags($this->nombre));
    $this->precio=htmlspecialchars(strip_tags($this->precio));
    $this->descripcion=htmlspecialchars(strip_tags($this->descripcion));
    $this->categoria_id=htmlspecialchars(strip_tags($this->categoria_id));
    $this->creado=htmlspecialchars(strip_tags($this->creado));

    // bind values
    $stmt->bindParam(":nombre", $this->nombre);
    $stmt->bindParam(":precio", $this->precio);
    $stmt->bindParam(":descripcion", $this->descripcion);
    $stmt->bindParam(":categoria_id", $this->categoria_id);
    $stmt->bindParam(":creado", $this->creado);

    // execute query
    if($stmt->execute()){
        return true;
    }

    return false;

}
```

9.3 Salida

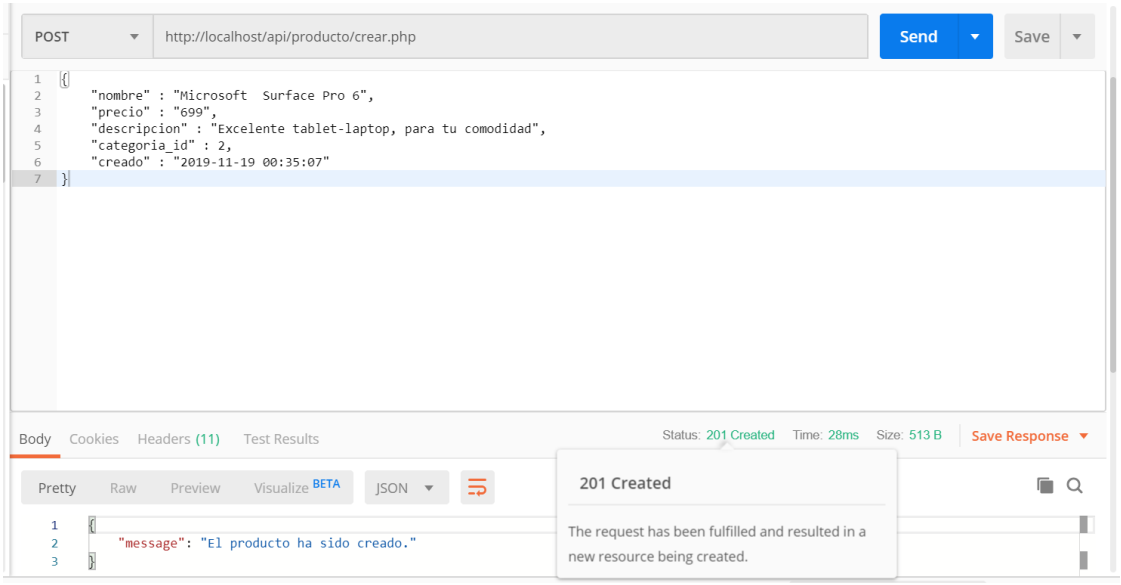
Para probar la creación exitosa de un producto, abrir POSTMAN. Ingrese lo siguiente como la URL de solicitud

http://localhost/api/producto/crear.php

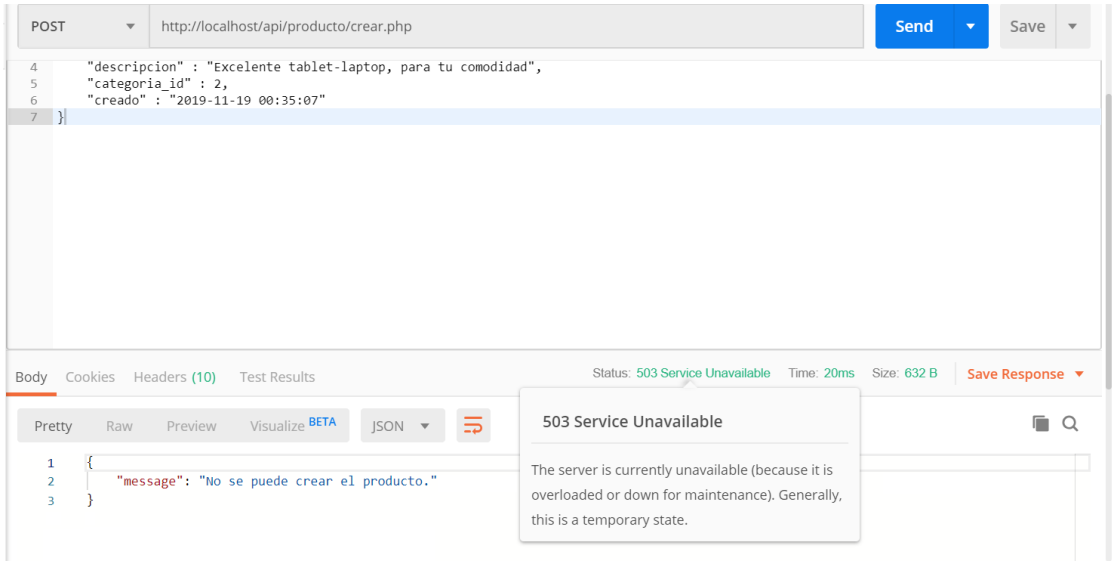
Haga clic en la pestaña "Cuerpo". Haga clic en "crudo". Ingrese este valor JSON:

```
{
    "nombre" : "Microsoft  Surface Pro 6",
    "precio" : "699",
    "descripcion" : "Excelente tablet-laptop, para tu comodidad",
    "categoria_id" : 2,
    "creado" : "2019-11-19 00:35:07"
}
```

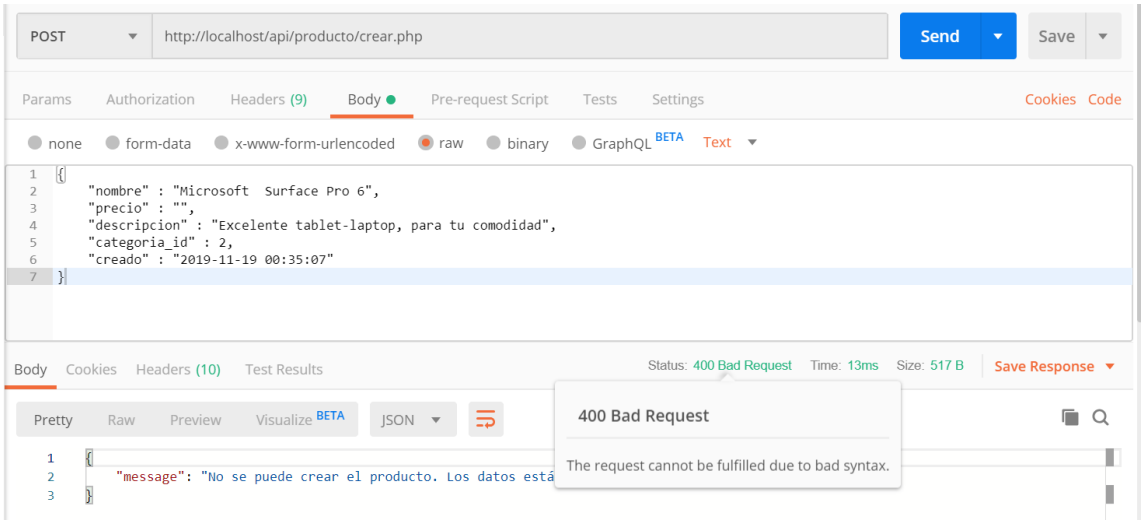
Debe tener un aspecto como este:



Si el sistema no puede crear el producto, debería verse así:



Si los datos enviados están incompletos, por ejemplo, faltan los datos del precio, la salida debería verse así:



X. Leer un producto

10.1 Crear archivo leer_uno.php

Abrir la carpeta del producto. Crear un nuevo archivo leer_uno.php . Abrir este archivo y colocar el siguiente código.

```php
<?php
// encabezados obligatorios
header("Access-Control-Allow-Origin: *");
header("Access-Control-Allow-Headers: access");
header("Access-Control-Allow-Methods: GET");
header("Access-Control-Allow-Credentials: true");
header('Content-Type: application/json');

// incluir archivos de conexion y objetos
include_once '../configuracion/conexion.php';
include_once '../objetos/producto.php';

// obtener conexion a la base de datos
$conex = new Conexion();
$db = $conex->obtenerConexion();

// preparar el objeto producto
$product = new Producto($db);

// set ID property of record to read
$product->id = isset($_GET['id']) ? $_GET['id'] : die();

// leer los detalles del producto a editar
$product->readOne();

if($product->nombre!=null){
    // creacion del arreglo
    $product_arr = array(
        "id" =>  $product->id,
        "nombre" => $product->nombre,
        "descripcion" => $product->descripcion,
        "precio" => $product->precio,
        "categoria_id" => $product->categoria_id,
        "categoria_desc" => $product->categoria_desc

    );

    // asignar codigo de respuesta - 200 OK
    http_response_code(200);

    // mostrarlo en formato json
    echo json_encode($product_arr);
}
?>
```

10.2 Método lecturaUnica() del producto

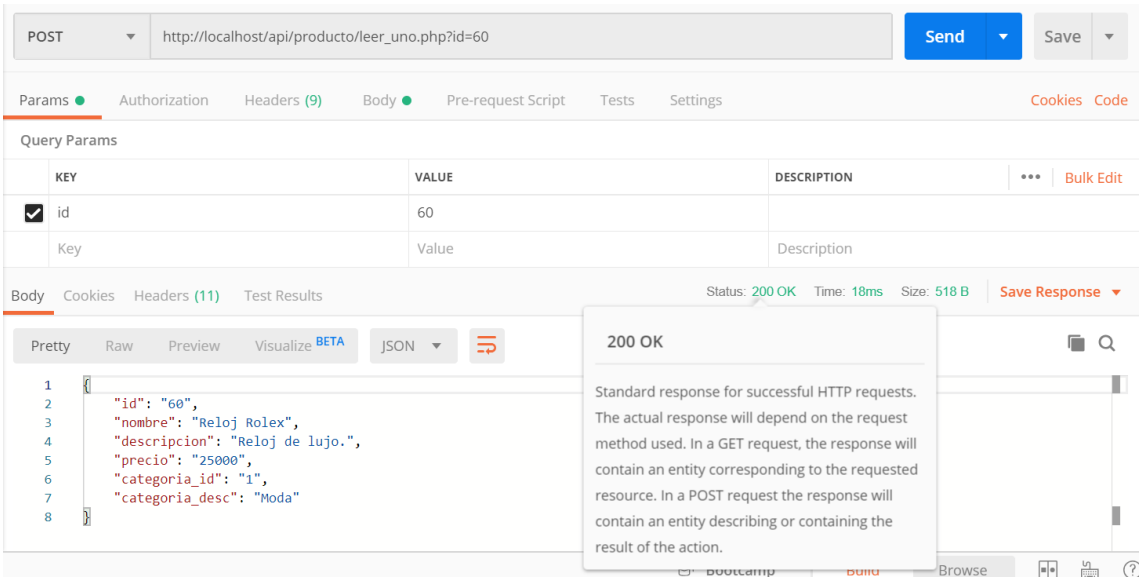
Abrir carpeta de objetos. Abrir el archivo producto.php . La sección anterior no funcionará sin el siguiente código dentro de la clase de Producto.

```
// utilizado al completar el formulario de actualización del producto
function readOne(){

    // consulta para leer un solo registro
    $query = "SELECT
                c.nombre as categoria_desc, p.id, p.nombre, p.descripcion, p.
precio, p.categoria_id, p.creado
            FROM
                " . $this->nombre_tabla . " p
                LEFT JOIN
                    categorias c
                        ON p.categoria_id = c.id
            WHERE
                p.id = ?
            LIMIT
                0,1";

    // preparar declaración de consulta
    $stmt = $this->conn->prepare( $query );

    // ID de enlace del producto a actualizar
    $stmt->bindParam(1, $this->id);

    // ejecutar consulta
    $stmt->execute();

    // obtener fila recuperada
    $row = $stmt->fetch(PDO::FETCH_ASSOC);

    // establecer valores a las propiedades del objeto
    $this->nombre = $row['nombre'];
    $this->precio = $row['precio'];
    $this->descripcion = $row['descripcion'];
    $this->categoria_id = $row['categoria_id'];
    $this->categoria_desc = $row['categoria_desc'];
}
```

10.3 Salida

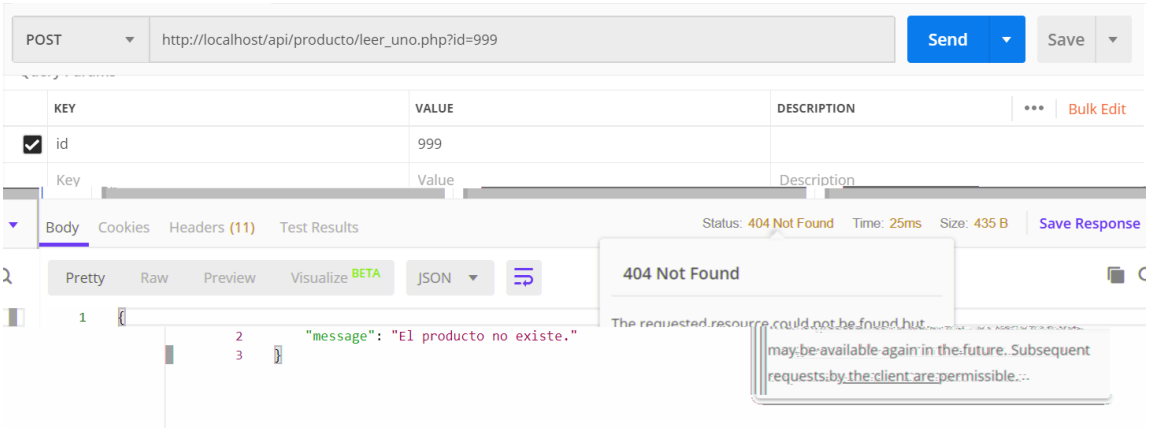
Primero, probaremos un producto que exista. Abrir POSTMAN. Ingresar lo siguiente como la URL de solicitud. Haga clic en el botón azul "Enviar".

http://localhost/api/producto/leer_uno.php?id=60



A continuación, probaremos un producto que no existe. Ingrese lo siguiente como la URL de solicitud. Haga clic en el botón azul "Enviar".

http://localhost/api/producto/leer_uno.php?id=999



Opcional: Elaborar métodos para,

- Actualizar producto
- Eliminar producto

En este servicio.