CREACIÓN DE API

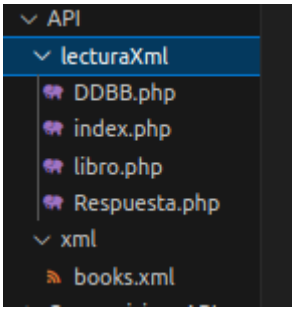
Memoria de la Práctica

1. API REST
    a. Protocolo y Métodos:
        i. Utiliza HTTP/HTTPS para la comunicación, garantizando la transferencia segura de datos.
        ii. Se apoya en métodos estándar como GET para lecturas, POST para creación, PUT para actualización, PATCH para modificaciones parciales y DELETE para eliminaciones.
    b. Formato de Datos y Flexibilidad:
        i. Utiliza formatos ligeros y comprensibles, como JSON o XML, favoreciendo la legibilidad y la interoperabilidad.
        ii. Facilita la comunicación y entendimiento entre sistemas diferentes.
        iii. Altamente adaptable y escalable, permitiendo la introducción de nuevas funcionalidades sin alterar la interfaz existente.
    c. Estilo Arquitectónico y Seguridad:
        i. Basada en el estilo Representational State Transfer (REST), lo que implica una arquitectura sin estado para favorecer la escalabilidad y la tolerancia a fallos.
        ii. No guarda información del estado entre solicitudes.
        iii. Utiliza HTTPS para asegurar la comunicación e implementa medidas de seguridad en el transporte y la aplicación para garantizar la integridad y confidencialidad de los datos.
2. API SOAP:
    a. Protocolo y Capa de Abstracción:
        i. Utiliza diversos protocolos como HTTP, SMTP, JMS, siendo independiente del transporte subyacente.
        ii. Proporciona una capa adicional de abstracción para entornos empresariales complejos, permitiendo la interoperabilidad en sistemas heterogéneos.
    b. Formato de Datos y Flexibilidad:
        i. Principalmente basada en XML para intercambio de datos, ofreciendo una estructura extensible y estructurada.
        ii. Menos adaptable que REST. Cambios en la interfaz pueden impactar a clientes existentes debido a la rigidez de WSDL
    c. Estilo Arquitectónico y Seguridad:
        i. Basada en el Protocolo Simple de Acceso a Objetos (SOAP), proporcionando una estructura formal y rigurosa.
        ii. Orientada a objetos, lo que implica una descripción formal y estructurada de las operaciones mediante Web Services Description Language (WSDL).

iii. Ofrece estándares de seguridad como WS-Security para protección a nivel de mensaje, con opciones más robustas y detalladas, adecuadas para entornos empresariales que requieren un nivel superior de seguridad.

CÓDIGO DE LA API

Estructura de los archivos de la API



BBDD comprimida

```php
<?php
2 references | 2 implementations
class Libreria
{
    /**
     * Lee los datos del archivo XML y los convierte en un arreglo de libros.
     *
     * @return array Un array de libros, donde cada libro es un array asociativo.
     */


    1 reference
    private function cargarDatosDesdeXML()
ocs/DWES/API/DWES-main/EntregaRecu/API/Consumicion_API
    }

    /**
     *
     * @param array $filtros es un array asociativo con los filtros de búsqueda incluye pagina .
     * @return array Una matriz de libros con los libros que cumplen el filtrado.
     */
    1 reference | 0 overrides
    public function obtenerListaLibros($filtros = [])
    { ⋯
    }
}
```

CARGA DE DATOS

```php
private function cargarDatosDesdeXML()
{
    $archivoXML = '../xml/books.xml';
    $datosXML = simplexml_load_file($archivoXML);
    $coleccionLibros = [];

    foreach ($datosXML->book as $libro) {
        $coleccionLibros[] = [
            'id' => (string)$libro['id'],
            'autor' => (string)$libro->author,
            'titulo' => (string)$libro->title,
            'genero' => (string)$libro->genre,
            'precio' => (int)$libro->price,
            'publicacion' => (int)$libro->publish_date,
            'descripcion' => (string)$libro->description
        ];
    }
    return $coleccionLibros;
}
```

LISTA DE LIBROS

```php
/**
 *
 * @param array $filtros es un array asociativo con los filtros de búsqueda incluye pagina .
 * @return array Una matriz de libros con los libros que cumplen el filtrado.
 */
1 reference | 0 overrides
public function obtenerListaLibros($filtros = [])
{
    $coleccionLibros = $this->cargarDatosDesdeXML();
    if (!is_array($filtros)) {
        return $coleccionLibros;
    } else {
        foreach ($filtros as $campo => $valor) {
            if ($campo !== 'pagina') {
                $coleccionLibros = array_filter($coleccionLibros, function ($libro) use ($campo, $valor) {
                    return stripos($libro[$campo], $valor) !== false;
                });
            } elseif (isset($filtros['pagina'])) {
                $pagina = (int)$filtros['pagina'];
                $librosPaginados = array_slice($coleccionLibros, 0, $pagina);
                $coleccionLibros = $librosPaginados;
            }
        }

        return array_values($coleccionLibros);
    }
}
```

LIBROS

```php
class Libro extends Libreria
{
    /**
     * array con los filtros permitidos para su posterior uso para filtrar los libros segun la
     * peticion realizada
     */
    1 reference
    private $filtrosPermitidos_get = array('id', 'autor', 'genero', 'pagina');

    /**
     *
     * @param array $filtros Array asociativo de la peticion get
     * @return array Un array que coinciden con los criterios de filtrado.
     */
    1 reference | 0 overrides
    public function obtenerLibrosConFiltros($filtros)
    {
        foreach ($filtros as $clave => $filtro) {
            if (!in_array($clave, $this->filtrosPermitidos_get)) {
                unset($filtros[$clave]);
                $respuesta = array(
                    'result' => 'error',
                );
                //hace llamada a una respuesta del servidor que informara al cliente de que hubo un error y devuelve el codigo 404
                RespuestaServicio::enviarRespuesta(404, $respuesta);
                exit;
            }
        }
        $datosLibros = parent::obtenerListaLibros($filtros);
        return $datosLibros;
    }
```

INDEX

```php
<?php
include_once 'libro.php';
include_once 'Respuesta.php';

$libro = new Libro();

if (($_SERVER['REQUEST_METHOD'])) {
    $resultado = array('result' => 'ok','libros' => $libro->obtenerLibrosConFiltros($_GET)
    );
    RespuestaServicio::enviarRespuesta(200, $resultado);
}
```

RESPUESTA

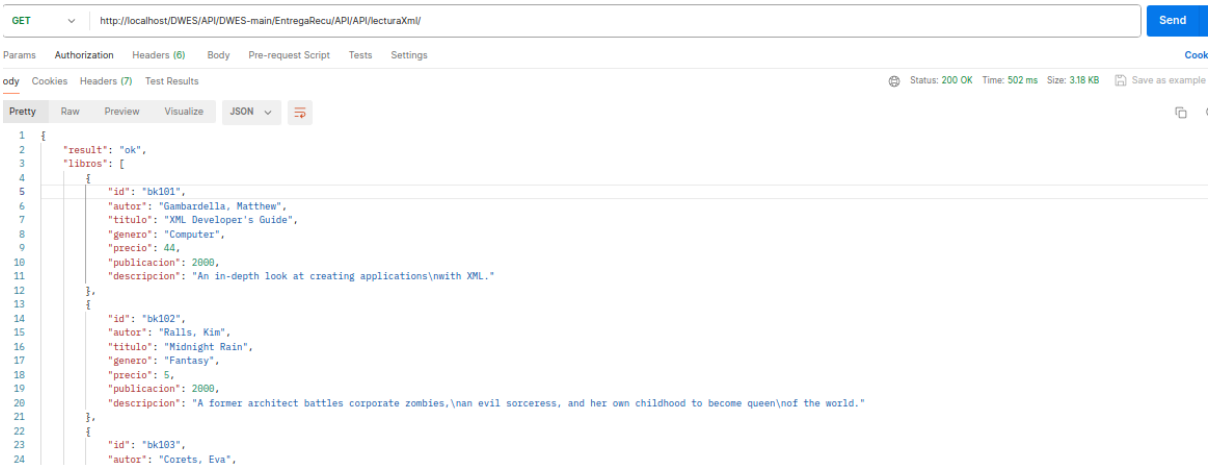
```php
<?php
/**
 * Clase Respuesta:
 * gestiona la creación y
 * envío de respuestas JSON
 */
2 references | 0 implementations
class RespuestaServicio{
    2 references | 0 overrides
    public static function enviarRespuesta($codigo, $listaLibros)
    {
        header('Content-type: application/json');
        http_response_code($codigo);

        echo json_encode($listaLibros);
    }
}
?>
```

RESULTADOS POSTMAN

En todos los casos de filtrado por un campo excepto en paginas se filtra por el campo definido y si el valor por el que se realiza la búsqueda está contenido en los resultados.

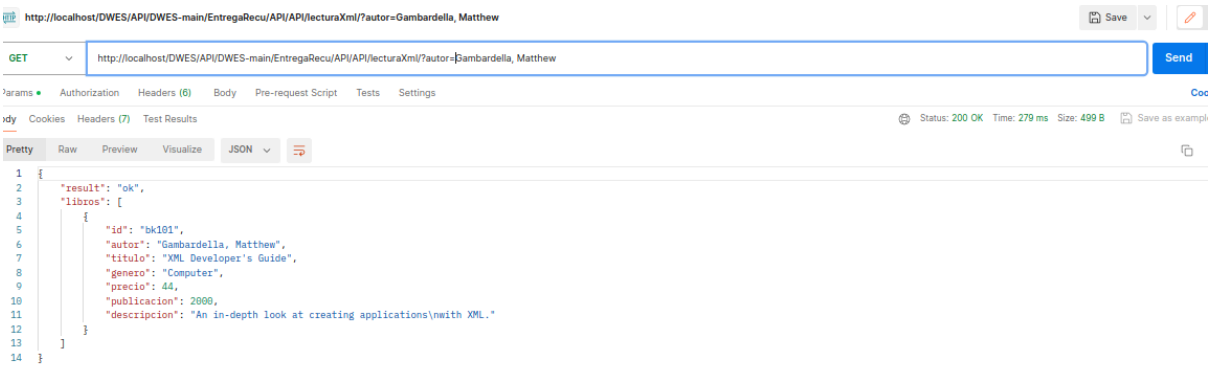
get sin parámetros

- No se le pasa ningún valor por el cual comparar y devuelve la lista de todos los libros que hay en el xml



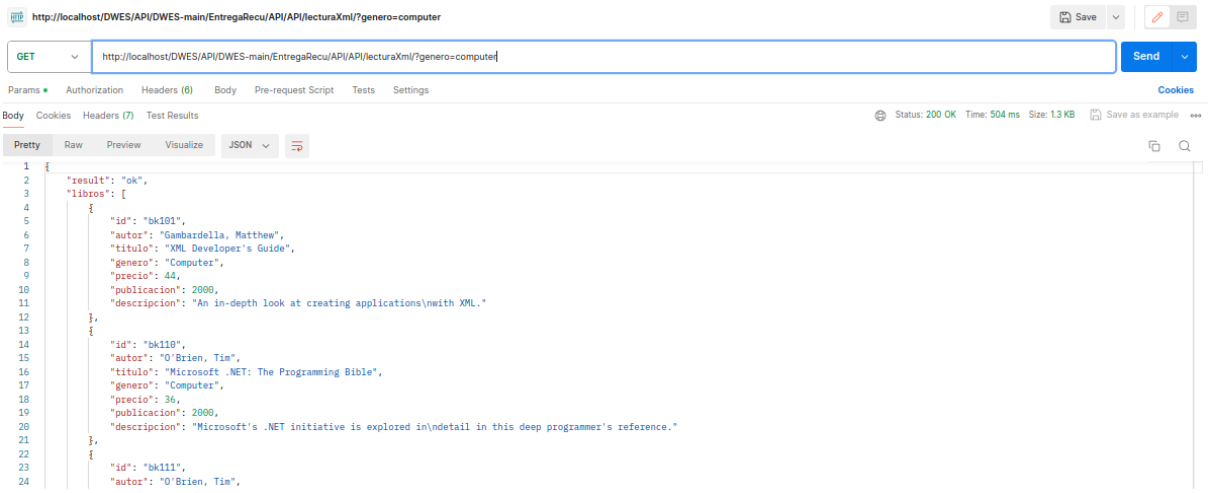
GET AUTOR:

Espera recibir una cadena por la cual comprueba si esta contenida en el campo autor, esto quiere decir si le pasamos ,c, devuelve los autores que contienen la letra c.



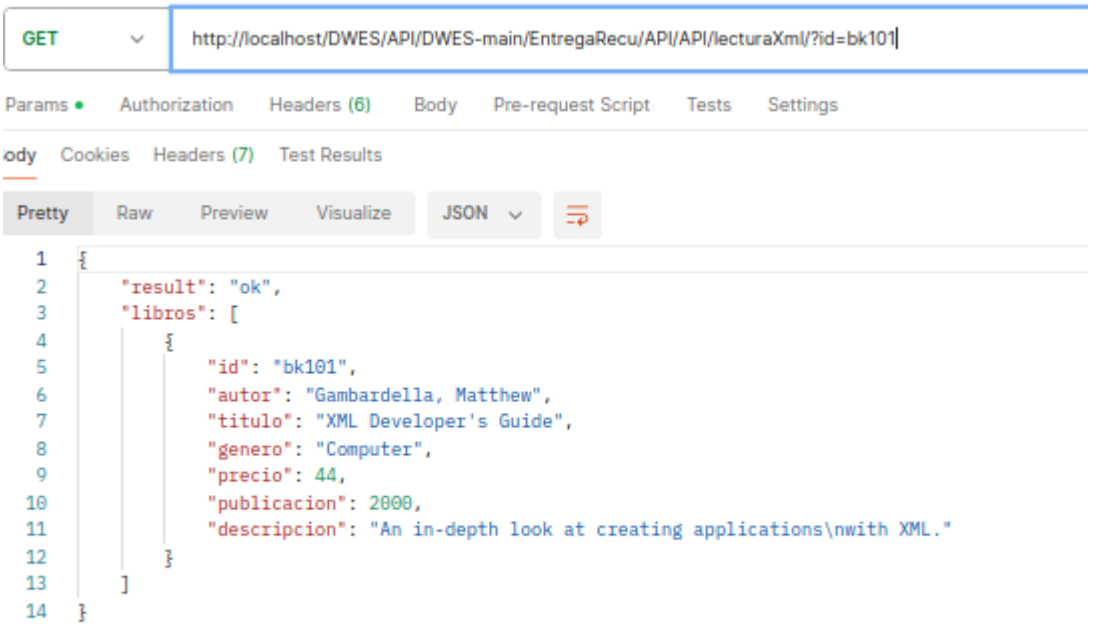
GET GÉNERO

Espera recibir una cadena por la cual comprueba si está contenida en el campo género de cada libro , esto quiere decir si le pasamos ,c, devuelve los libros con género que contienen la letra c.



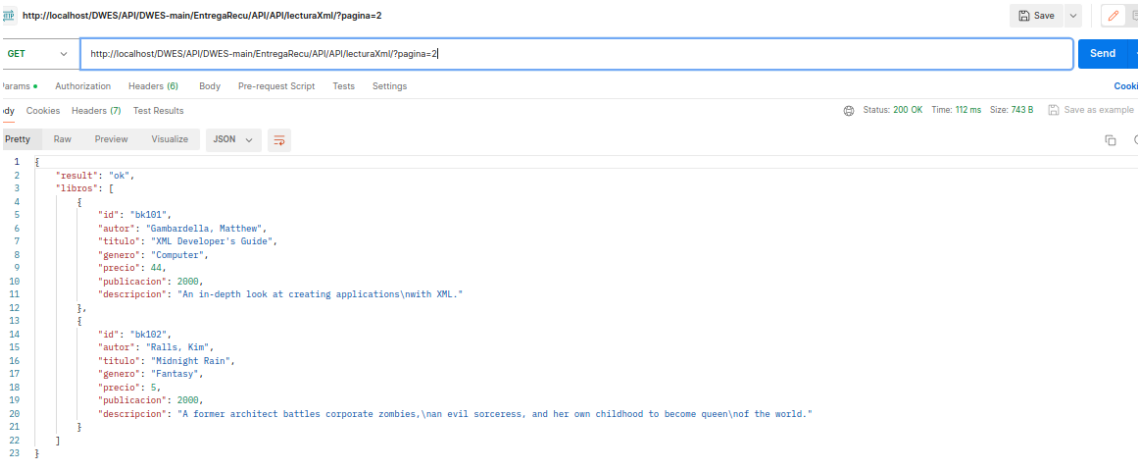
GET ID

Espera un id en y devuelve el libro con ese id.



GET PÁGINA

Espera un entero y devuelve ese numero de libros .


http://localhost/DWES/API/DWES-main/EntregaRecu/API/API/lecturaXml/?pagina=2
Save
GET
http://localhost/DWES/API/DWES-main/EntregaRecu/API/API/lecturaXml/?pagina=2
Send
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
Headers (7)
Test Results
Status: 200 OK
Time: 112 ms
Size: 743 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
{
"result": "ok",
"libros": [
{
"id": "bk101",
"autor": "Gambardella, Matthew",
"titulo": "XML Developer's Guide",
"genero": "Computer",
"precio": 44,
"publicacion": 2000,
"descripcion": "An in-depth look at creating applications\nwith XML."
},
{
"id": "bk102",
"autor": "Ralls, Kim",
"titulo": "Midnight Rain",
"genero": "Fantasy",
"precio": 5,
"publicacion": 2000,
"descripcion": "A former architect battles corporate zombies,\nan evil sorceress, and her own childhood to become queen\nof the world."
}
]
}