

CREACIÓN DE API

Memoria de la Práctica

1. Protocolo y Métodos:

- Utiliza HTTP/HTTPS para la comunicación, garantizando la transferencia segura de datos.
- Se apoya en métodos estándar como GET para lecturas, POST para creación, PUT para actualización, PATCH para modificaciones parciales y DELETE para eliminaciones.

2. Formato de Datos y Flexibilidad:

- Utiliza formatos ligeros y comprensibles, como JSON o XML, favoreciendo la legibilidad y la interoperabilidad.
- Facilita la comunicación y entendimiento entre sistemas diferentes.
- Altamente adaptable y escalable, permitiendo la introducción de nuevas funcionalidades sin alterar la interfaz existente.

3. Estilo Arquitectónico y Seguridad:

- Basada en el estilo Representational State Transfer (REST), lo que implica una arquitectura sin estado para favorecer la escalabilidad y la tolerancia a fallos.
- No guarda información del estado entre solicitudes.
- 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.

API SOAP:

1. Protocolo y Capa de Abstracción:

- Utiliza diversos protocolos como HTTP, SMTP, JMS, siendo independiente del transporte subyacente.
- Proporciona una capa adicional de abstracción para entornos empresariales complejos, permitiendo la interoperabilidad en sistemas heterogéneos.

2. Formato de Datos y Flexibilidad:

- Principalmente basada en XML para intercambio de datos, ofreciendo una estructura extensible y estructurada.
- Menos adaptable que REST. Cambios en la interfaz pueden impactar a clientes existentes debido a la rigidez de WSDL.

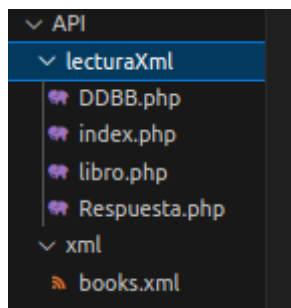
3. Estilo Arquitectónico y Seguridad:

- Basada en el Protocolo Simple de Acceso a Objetos (SOAP), proporcionando una estructura formal y rigurosa.
- Orientada a objetos, lo que implica una descripción formal y estructurada de las operaciones mediante Web Services Description Language (WSDL).

- 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 la API



BBDD comprimida

```
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.
     */
    private function cargarDatosDesdeXML()
    {
        // ...
    }

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

CARGA DE DATOS

```

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

```

/**
 *
 * @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, $valor) {
                    return strpos($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

```

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
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
/**
 * 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);

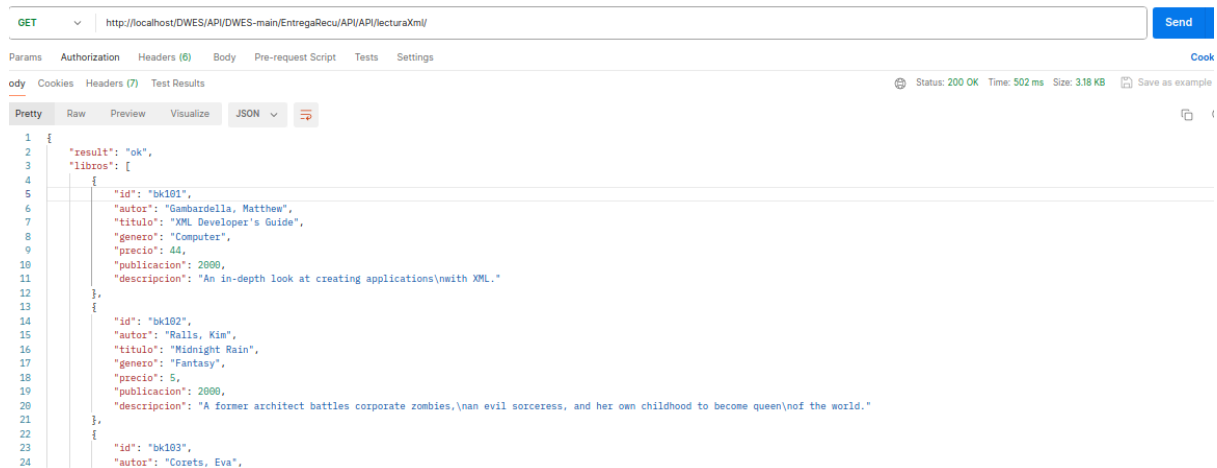
        echo json_encode($listaLibros);
    }
}
?>

```

RESULTADOS POSTMAN

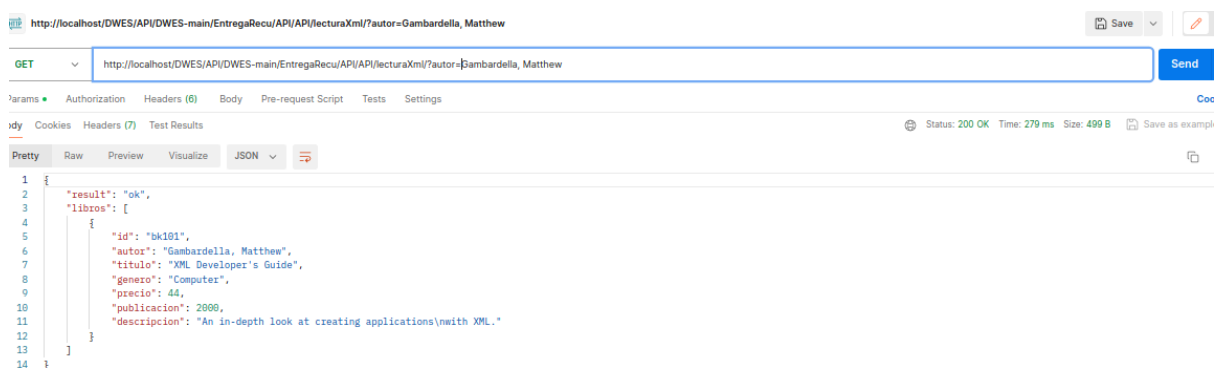
get sin parámetros

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



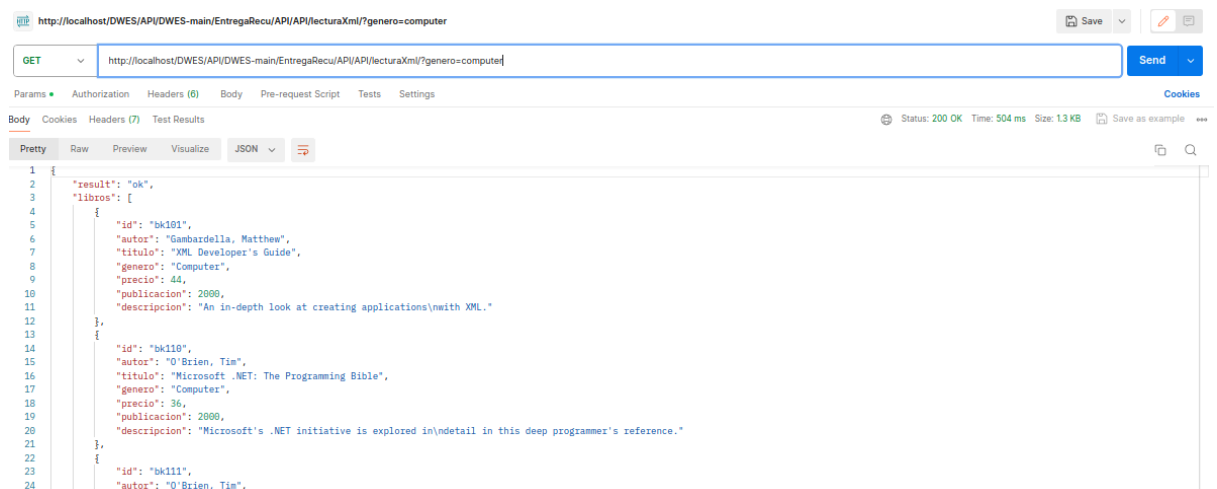
```
1 {
2   "result": "ok",
3   "libros": [
4     {
5       "id": "bk101",
6       "autor": "Gambardella, Matthew",
7       "titulo": "XML Developer's Guide",
8       "genero": "Computer",
9       "precio": 44,
10      "publicacion": 2000,
11      "descripcion": "An in-depth look at creating applications\\nwith XML."
12    },
13    {
14      "id": "bk102",
15      "autor": "Ralls, Kim",
16      "titulo": "Midnight Rain",
17      "genero": "Fantasy",
18      "precio": 5,
19      "publicacion": 2000,
20      "descripcion": "A former architect battles corporate zombies,\\nan evil sorceress, and her own childhood to become queen\\nof the world."
21    },
22    {
23      "id": "bk103",
24      "autor": "Corets, Eva",
```

GET AUTOR



```
1 {
2   "result": "ok",
3   "libros": [
4     {
5       "id": "bk101",
6       "autor": "Gambardella, Matthew",
7       "titulo": "XML Developer's Guide",
8       "genero": "Computer",
9       "precio": 44,
10      "publicacion": 2000,
11      "descripcion": "An in-depth look at creating applications\\nwith XML."
12    }
13  ]
14 }
```

GET GÉNERO



```
1 {
2   "result": "ok",
3   "libros": [
4     {
5       "id": "bk101",
6       "autor": "Gambardella, Matthew",
7       "titulo": "XML Developer's Guide",
8       "genero": "Computer",
9       "precio": 44,
10      "publicacion": 2000,
11      "descripcion": "An in-depth look at creating applications\\nwith XML."
12    },
13    {
14      "id": "bk110",
15      "autor": "O'Brien, Tim",
16      "titulo": "Microsoft .NET: The Programming Bible",
17      "genero": "Computer",
18      "precio": 36,
19      "publicacion": 2000,
20      "descripcion": "Microsoft's .NET initiative is explored in\\ndetail in this deep programmer's reference."
21    },
22    {
23      "id": "bk111",
24      "autor": "O'Brien, Tim",
```

GET ID

GET ▼ http://localhost/DWES/API/DWES-main/EntregaRecu/API/API/lecturaXml/?id=bk101

Params • Authorization Headers (6) Body Pre-request Script Tests Settings

ody Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON ▼ ≡

```

1 {
2   "result": "ok",
3   "libros": [
4     {
5       "id": "bk101",
6       "autor": "Gambardella, Matthew",
7       "titulo": "XML Developer's Guide",
8       "genero": "Computer",
9       "precio": 44,
10      "publicacion": 2000,
11      "descripcion": "An in-depth look at creating applications\nwith XML."
12    }
13  ]
14 }

```

GET PÁGINA

🔗 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

Params • Authorization Headers (6) Body Pre-request Script Tests Settings Cooki

dy Cookies Headers (7) Test Results Status: 200 OK Time: 112 ms Size: 743 B Save as example

Pretty Raw Preview Visualize JSON ▼ ≡

```

1 {
2   "result": "ok",
3   "libros": [
4     {
5       "id": "bk101",
6       "autor": "Gambardella, Matthew",
7       "titulo": "XML Developer's Guide",
8       "genero": "Computer",
9       "precio": 44,
10      "publicacion": 2000,
11      "descripcion": "An in-depth look at creating applications\nwith XML."
12    },
13    {
14      "id": "bk102",
15      "autor": "Ralls, Kim",
16      "titulo": "Midnight Rain",
17      "genero": "Fantasy",
18      "precio": 5,
19      "publicacion": 2000,
20      "descripcion": "A former architect battles corporate zombies,\nan evil sorceress, and her own childhood to become queen\nof the world."
21    }
22  ]
23 }

```