DOCUMENTACIÓN TÉCNICA API CONNECT

Interfaz de uso del servicio web

Para acceder al servicio web realizado, deberemos utilizar una URL de base que nos llevará al punto de control del servicio. En este caso, nuestra URL base será la siguiente:

BASE URL: http://localhost/DWES/API-Connect/books

Al no estar desplegada en un servidor con hosting, lo desplegamos en nuestro propio equipo con XAMPP. Este acceso sería un acceso no parametrizado, sin embargo, el cliente podrá filtrar la información parametrizando la URL.

Peticiones esperadas:

Las peticiones esperadas al servicio web en este caso será únicamente la petición GET para recibir los datos de la API. En el propio código de la aplicación se podrían añadir funcionalidades para gestionar las peticiones esperadas, de esa manera, el cliente podría realizar peticiones POST y PUT.

Parámetros disponibles:

Los parámetros disponibles para filtrar la información recibida del servicio son:

Filtrar por ID:

Ejemplo de petición: http://localhost/DWES/API-Connect/books?id=bk112

Filtrar por Autor:

Ejemplo de petición: http://localhost/DWES/API-Connect/books?autor=Galos, Mike

Filtrar por Género:

Ejemplo de petición: http://localhost/DWES/API-Connect/books?genero=Computer

Paginación:

Respecto a la paginación, no encontraba el sentido a realizar una paginación para este tipo de peticiones, ya que, en nuestro caso, la estructura de datos es demasiado pequeña. Por tanto, si realizamos una paginación como su concepto indica que se debe realizar, al paginar por 5, seguiría mostrando resultados.

Mi planteamiento ha sido que muestre un limit a los resultados. Si introduces un 5, muestra 5 libros. Esta me ha parecido la solución más lógica en este proyecto.

Ejemplo de petición: http://localhost/DWES/API-Connect/books?pagina=2

Estos parámetros son los únicos soportados por el servicio web. Al introducir un parámetro que no sea el esperado por la aplicación, la aplicación devuelve un mensaje de error.

Respuesta JSON:

El servicio web, devuelve un conjunto de datos transformados de XML a formato JSON para que sea legible por un cliente que los consuma. En este caso, hemos elegido JSON porque es el más utilizado y tiene una tratabilidad de los datos muy fácil de utilizar.

Un ejemplo de la estructura de datos que devuelve el servicio web al realizar una petición GET sería el siguiente:


```
{
   "result": "ok",
   "libros": [
      {
         "id": "bk101",
         "autor": "Gambardella, Matthew",
```

```
        "titulo": "XML Developer's Guide",
        "genero": "Computer",
        "precio": 44.95,
        "lanzamiento": "2000-10-01",
        "descripcion": "An in-depth look at creating applications\nwith XML."
    },
    {
        "id": "bk102",
        "autor": "Ralls, Kim",
        "titulo": "Midnight Rain",
        "genero": "Fantasy",
        "precio": 5.95,
        "lanzamiento": "2000-12-16",
        "descripcion": "A former architect battles corporate zombies,\nan evil sorceress, and her own childhood to become queen\nof the world."
    },
....
  ]
}
```


Manejo de errores

En cuanto al manejo de errores, encontramos:

- Si el fichero no ha sido cargado: Si no ha encontrado el archivo o ha ocurrido algún error, el servicio web devolverá un error con el siguiente mensaje: "No se ha podido cargar el archivo".
- Si los parámetros no están permitidos, el servicio web devolverá el siguiente mensaje de error: “Error en la solicitud”.
- Si el método de solicitud no está permitido, devolverá un result “error”.
- Si no encuentra ningún resultado, devolverá un array vacío.