UD9 – Servicios web SOAP y RESTful

2º CFGS
Desarrollo de Aplicaciones Web

2022-23

1.- Arquitectura orientada a servicios

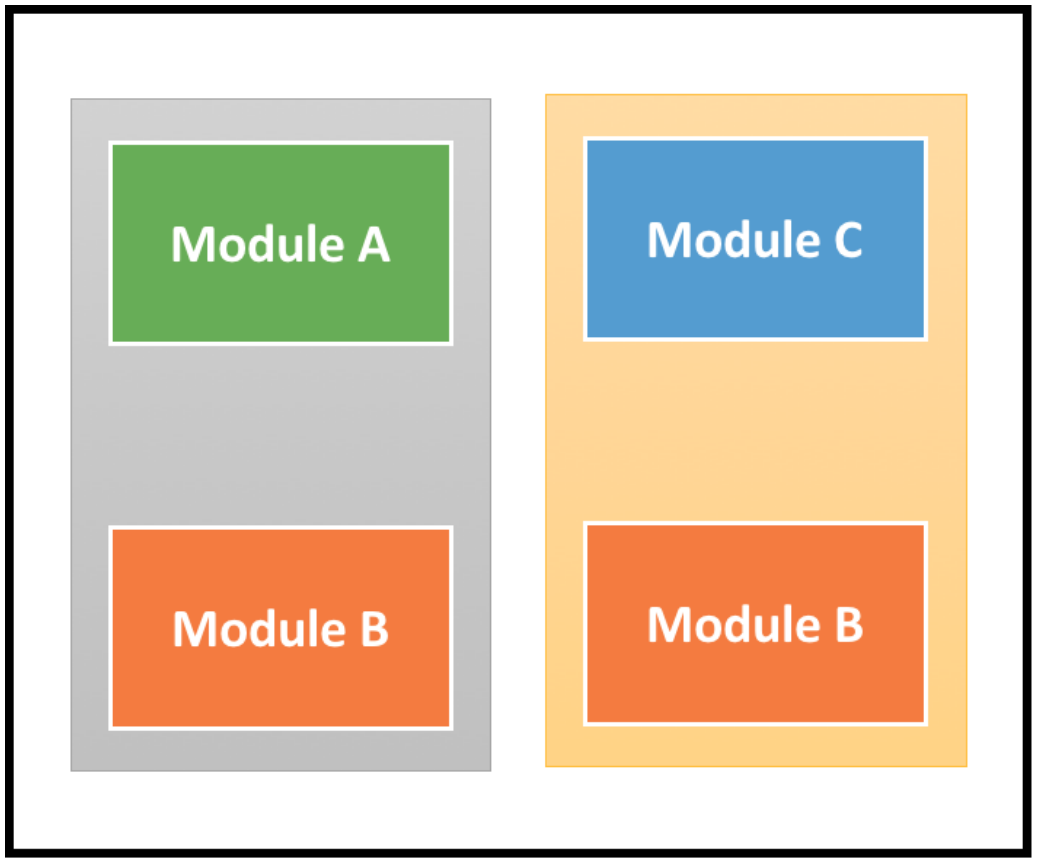
SOA → Service Oriented Architecture

Los **servicios web** (web services) son un conjunto de protocolos y estándares que permiten que diferentes aplicaciones intercambien datos.

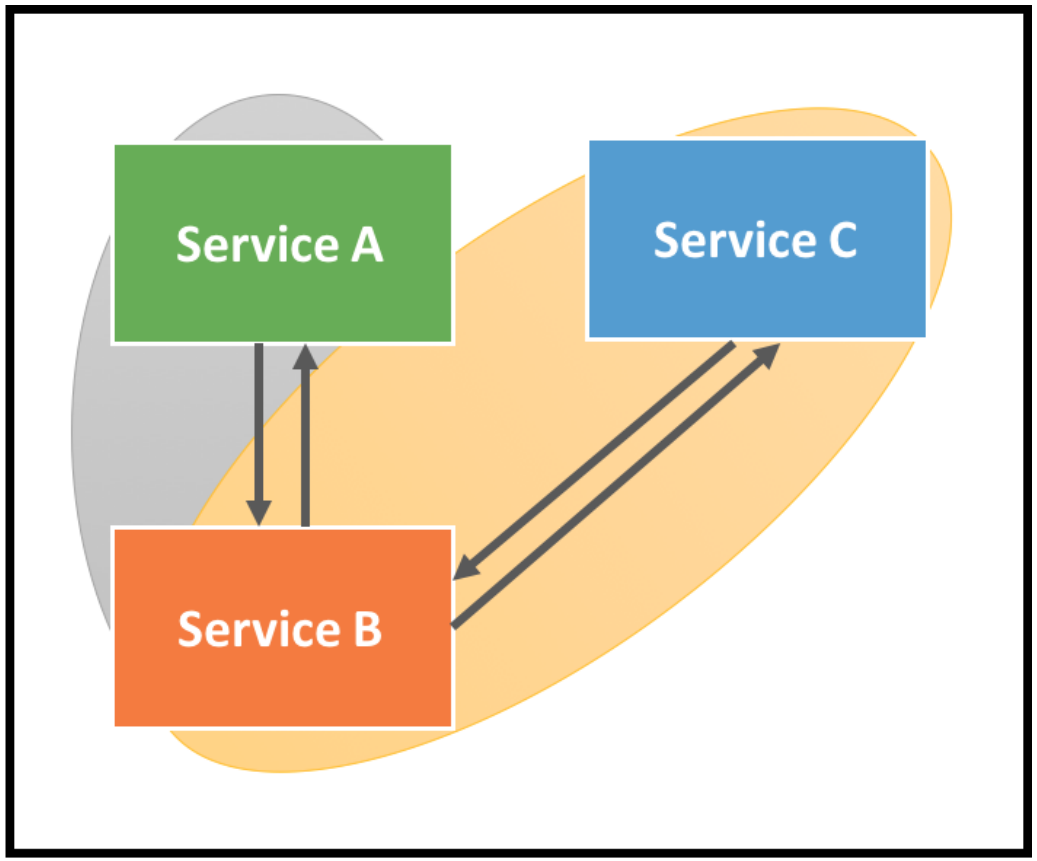
Una de las características principales es que estas aplicaciones pueden estar desarrolladas con diferentes lenguajes de programación y pueden estar ejecutándose en plataformas diferentes.

1.- Arquitectura orientada a servicios

SOA → Service Oriented Architecture



Arquitectura monolítica



Arquitectura orientada a servicios

1.- Arquitectura orientada a servicios

Cuando se utiliza una arquitectura orientada a servicios, el desarrollo de aplicaciones **no se basa en generar aplicaciones completas**.

Con esta arquitectura se desarrollan servicios (aplicaciones) que generan resultados.

Estos servicios se pueden **utilizar** tanto en la **propia aplicación** como en **aplicaciones de terceros**.

Gracias a los servicios se puede **reutilizar el código** de una manera muy sencilla.

1.- Arquitectura orientada a servicios

Los servicios web son una herramienta muy potente para el uso interno del desarrollo de una aplicación web gracias a la facilidad de reutilización de funcionalidades.

Pero además también es una herramienta muy potente para **incluir funcionalidades propias en otras aplicaciones web y viceversa**, de hecho, hoy en día muchísimas aplicaciones web ofrecen parte de su funcionalidad a terceros mediante servicios web.

- Publicaciones de redes sociales en aplicaciones web externas.
- Comentarios de una red social en aplicaciones web externas.
- Registro/login con aplicaciones web externas a la propia.
- Ficha de un producto de una tienda en una aplicación web externa.
- Compartir contenido de una aplicación externa en una red social.

2.- Principios en los que se basa SOA

- **Contrato de servicio estandarizado**: un servicio ofrece siempre los mismos estándares.
- **Bajo acoplamiento**: los servicios realizan funciones muy concretas sin necesidad de otros componentes.
- **Abstracción**: los servicios no muestran cómo realizan las funcionalidades.
- **Reusabilidad**: los servicios ofrecen funcionalidades independientes al consumidor y su entorno por lo que se pueden reutilizar en cualquier proyecto.
- **Autonomía**: los servicios son independientes de la tecnología del consumidor.
- **Sin estado**: reduce el consumo de recursos al delegar el manejo de estados (sesiones).
- **Garantizar su descubrimiento**: los servicios ofrecen metadata para poder descubrirlos e interpretarlos para su uso.
- **Se puede usar en composiciones**: los servicios se pueden usar de manera conjunta sin importar el tamaño ni la complejidad.

3.- Estándares usados en SOA

- **HTML**: HyperText Markup Language
- **XML**: eXtensible Markup Language
- **JSON**: JavaScript Object Notation
- **SOAP**: Simple Object Access Protocol
- **WSDL**: Web Servides Description Language
- **REST**: Representational State Transfer
- **UDDI**: Universal Description, Discovery and Integration

3.- Estándares usados en SOA

XML: e**X**tensible **M**arkup **L**anguage

Metalenguaje que permite definir lenguajes de marcas.

HTML se define en base a XML.

Se utiliza para intercambiar datos entre servidor y cliente web.

```xml
<?xml version="1.0"?>
<Catalog>
    <Book id="bk101">
        <Author>Garghentini, Davide</Author>
        <Title>XML Developer's Guide</Title>
        <Price>44.95</Price>
        <Description>Creating applications with XML.</Description>
    </Book>
    <Book id="bk102">
        <Author>Garcia, Debra</Author>
        <Title>Midnight Rain</Title>
        <Price>5.95</Price>
        <Description>A former architect battles corporate zombies, an evil sorceress</Description>
    </Book>
</Catalog>
```

3.- Estándares usados en SOA

XML

Si un script php genera un XML no debe mostrar HTML por pantalla.

Primero se indicará con la cabecera correspondiente que se genera un archivo XML y posteriormente irá mostrando cada línea del XML:

```
header('Content-Type: application/xml; charset=utf-8');
echo '<?xml version="1.0" encoding="UTF-8"?>';
echo '<bliblioteca>';
   echo '<libro>';
   // continuar con toda la generación del XML
```

3.- Estándares usados en SOA

JSON: **J**ava**S**cript **O**bject **N**otation

Permite describir objetos con notación de texto.

Es mucho más ligero y sencillo que XML.

Se ha convertido en la alternativa a XML por el uso de AJAX.

```
{
    "libro": [
        {
            "id": "01",
            "lenguaje": "Java",
            "edición": "tercera",
            "autor": "Herbert Schildt"
        },
        {
            "id": "07",
            "lenguaje": "C++",
            "edición": "seguna",
            "autor": "E.Balagurusamy"
        }
    ]
}
```

3.- Estándares usados en SOA

JSON

En PHP existen las siguientes funciones para trabajar con JSON.

- Convertir un array en una cadena en notación JSON:

```
$cadenaJson = json_encode($array);
```

- Obtener un array a partir de una cadena en notación JSON:

```
$array = json_decode($cadenaJson, true);
```

El parámetro **true** permite que el array generado sea asociativo.

3.- Estándares usados en SOA

JSON

Si un script php genera un JSON no debe mostrar nada de HTML por pantalla y finalmente mostrará el JSON indicando previamente la cabecera correspondiente:

```php
// primero se debe generar el array normal con los datos
// $datos
header('Content-Type: application/json; charset=utf-8');
echo json_encode($datos);
```

Práctica

Actividad 1:

Creando XML

Actividad 2:

Creando JSON.

4.- SOAP

SOAP es un **protocolo** que indica cómo deben ser los mensajes entre el servidor y el cliente, y cómo deben procesarse dichos mensajes.

En PHP se puede hacer uso de SOAP de diferentes maneras, a continuación, se estudiará la manera nativa **PHP SOAP**.

Se utilizan **archivos WSDL** (Web Services Description Language) para **describir las funcionalidades** de los servicios SOAP.

Si se usa PHP SOAP se debe definir manualmente el archivo WSDL.

5.- WSDL

Web **S**ervices **D**escription **L**anguage

Lenguaje para la descripción de servicios web.

Indica cómo se tiene que conectar al servicio, los parámetros que acepta y qué devuelve.

Generalmente para obtener el fichero WSDL de un servicio web se tiene que añadir la cadena "?wsdl" a la URL del servicio web.

http://www.dneonline.com/calculator.asmx
http://www.dneonline.com/calculator.asmx?wsdl

No es necesario que un servicio web ofrezca el archivo WDSL pero es importante que lo haga para dotar de robustez al servicio web.

5.- WSDL

Un archivo WSDL tiene la siguiente estructura:

```
<definitions
    name="…"
    targetNamespace="http://…"
    xmlns:tns="http://…"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    … >

    <types> … </types>
    <message> … </message>
    <portType> … </portType>
    <binding> … </binding>
    <service> … </service>
</definitions>
```

5.- WSDL

- **types**: definición de los tipos de datos que usa el servicios.
- **messages**: definición del conjunto de datos que recibe o envía una función del servicio.
- **portType**: cada portTye es un conjunto de funciones que se implementan en el servicio web.
- **binding**: definición de cómo se va a transmitir la información de cada portType.
- **service**: contiene una lista de elementos de tipo port. Cada port indica en qué URL se puede acceder al servicio web.

6.- PHP SOAP

Para activar el uso de SOAP en PHP se debe **activar su extensión** en el archivo **php.ini**:

;extension=soap

Se debe eliminar el ; y reiniciar el servidor web.

Para desarrollar y/o usar servicios web con PHP se usan las siguientes clases:

SoapClient: https://php.net/manual/es/soapclient.soapclient.php

SoapServer: https://php.net/manual/es/soapserver.soapserver.php

6.- PHP SOAP

Analizando un servicio web:

calcularLetraDNI.php → servicio web SOAP con la función que ofrece.

index.php → script que consume el servicio web anterior

Uno de los problemas de SOAP es que la gestión de errores desde el servicio web no aporta información sobre cuál es el error que se produce. Si se produce un error se envía una respuesta HTTP 500 (el servidor no pudo completar la petición).

Al usar un servicio web SOAP antes de llamar a una función del servicio web hay que asegurarse de que los datos son correctos.

Práctica

Actividad 3:

Crear y usar un servicio web.

7.- API RESTful

REST → **RE**presentational **S**tate **T**ransfer

Es una arquitectura que se basa en **peticiones HTTP** para trabajar con los datos de la aplicación web (almacenados en la base de datos).

Fundamentos:

- Protocolo cliente/servidor sin estado.
- Operaciones bien definidas: **GET**, **POST**, **PUT**, **PATCH**, **DELETE**
- Sintaxis universal.

7.- API RESTful

Se conoce como **API Restful** a los servicios web desarrollados mediante la arquitectura REST (peticiones HTTP).

Se usan **peticiones HTTP**, generalmente mediante **URL amigables** para realizar acciones.

El servidor puede devolver la información en diferentes formatos: XML, **JSON**, JSON-LD (Google), KML, KMZ, CSV…

Con API RESTful la respuesta puede contener cualquier tipo de información por lo que se puede realizar una petición al servicio web y si hay un error este puede informar de lo ocurrido (al contrario que con SOAP).

7.- API RESTful

Los diferentes tipos de petición se usan para las siguientes tareas:

- **GET**: obtener uno o varios registro de una tabla.
- **POST**: almacenar un registro nuevo en una tabla.
- **PUT**: cambiar todos los campos de un registro de una tabla.
- **PATCH**: cambiar algún campo de un registro de una tabla.
- **DELETE**: eliminar un registro de una tabla.

7.- API RESTful

Para desarrollar servicios web API RESTful con PHP se debe:

- Definir las URL (amigables) aceptadas por la API RESTful.
- Definir el tipo de petición aceptado para cada URL.
- Crear los scripts que traten las peticiones a las URL que se definan.

A cada para "URL-tipo petición" se le conoce como **endopoint**.

Habitualmente las URL aceptadas suelen añadir el prefijo **api**.

7.- API RESTful

Ejemplo:

Para obtener todas las publicaciones de un usuario el endpoint de la API RESTful podría ser:

miaplicación.com/api/publicaciones/RickSanchez (tipo GET)

Y el script que trate la petición:

miaplicacion.com/api/publicaciones.php?usuario=***NombreUsuario***

7.- API RESTful

Ejemplo:
desarrollar con API RESTful el servicio web para operaciones realizado con SOAP

Los endpoint, todos tipo GET, podrían ser:

miaplicacion.com/api/suma/3/4
miaplicacion.com/api/resta/3/4
miaplicacion.com/api/multiplicacion/3/4
miaplicacion.com/api/division/3/4

Todas estas URL se redirigirán a:

miaplicacion.com/calculadora.php?operacion=$1&op1=$2&op2=$3

Dentro del script calculadora.php según la variable operación se realizará una acción un otra y se devolverá el resultado en formato JSON, por ejemplo:

{ "error": 0, "resultado": 127 }

7.- API RESTful

En ocasiones, la acción a realizar mediante la petición API RESTful conlleva la modificación de los datos de la base de datos.

Otras veces interesa limitar el uso externo de la API RESTful para evitar problemas de seguridad o demasiados accesos.

En estos casos lo más habitual es tener que registrarse para poder usar la API RESTful.

De esta manera se obtiene un **token** que se deberá añadir a la URL de la petición.

miaplicacion.com/api/publicaciones/RickSanchez/e613db7e8b4f50aaee0f

miaplicacion.com/api/publicaciones.php?usuario=***Usuario***&apiKey=***token***

7.- API RESTful

Los ejemplos anteriores solo muestran **peticiones GET**, pero se pueden desarrollar acciones para el resto de peticiones POST, PUT, PATCH, DELETE.

En el navegador solo se pueden probar aquellas peticiones GET o POST, por lo que si se quieren probar el resto de peticiones se puede usar algún software como Postman.

En el siguiente enlace se puede consultar un tutorial de cómo crear un servicio web Restful con PHP:

https://www.codigonaranja.com/2018/crear-restful-web-service-php

En la siguientes unidades (framework Laravel) se estudiarán todos los tipos de peticiones HTTP y se trabajará con ellas.

7.- API RESTful

Cómo publicar un servidor web API RESTful:

Los desarrolladores de servicios web API Restful publican cómo utilizar dicho servicio mediante su API RESTful.

Consultando la documentación se sabe:

- Cómo funcionan
- Cuáles son los **endpoint** disponibles.
- Formato de los datos devueltos (generalmente JSON).
- Si necesitan registro previo a su uso (generalmente facilitan un Token).

7.- API RESTful

Ejemplos de API públicas:

https://github.com/public-apis/public-apis

https://rickandmortyapi.com/documentation

7.- API RESTful

Ejemplos de API públicas:
https://valencia.opendatasoft.com/explore/?sort=modified

Ejemplo de uso:
https://valencia.opendatasoft.com/explore/dataset/valenbisi-disponibilitat-valenbisi-dsiponibilidad/api/

Documentación: https://valencia.opendatasoft.com/api/v2/console

7.- API RESTful

https://valencia.opendatasoft.com/api/records/1.0/search/?dataset=valenbisi-disponibilitat-valenbisi-dsiponibilidad&q=cadiz


```
{
    "nhits": 1,
    "parameters": {
        "dataset": "valenbisi-disponibilitat-valenbisi-dsiponibilidad",
        "q": "cadiz",
        "rows": 10,
        "start": 0,
        "format": "json",
        "timezone": "UTC"
    },
    "records": [
        {
            "datasetid": "valenbisi-disponibilitat-valenbisi-dsiponibilidad",
            "recordid": "8b8bb8a3c09a7dbbf8838775ff83c3f796d3f6b1",
            "fields": {
                "geo_point_2d": [
                    39.460925286769694,
                    -0.3727053849231208
                ],
                "ticket": "T",
                "total": 15,
                "name": "152_C/ LITERATO AZORÍN",
                "geo_shape": {
                    "coordinates": [
                        -0.3727053849231208,
                        39.460925286769694
                    ],
                        "type": "Point"
                    },
                    "updated_at": "07/12/2022 22:01:30",
                    "open": "T",
                    "number": 152,
                    "address": "Reina Doña María - Cádiz",
                    "available": 14,
                    "free": 1,
                    "gid": 901732,
                    "globalid": "{285026C8-649F-469E-B3D3-5258E4B4D551}"
                },
                "geometry": {
                    "type": "Point",
                    "coordinates": [
                        -0.3727053849231208,
                        39.460925286769694
                    ]
                },
                "record_timestamp": "2022-12-07T21:02:16.195Z"
            }
        ]
}
```

7.- API RESTful

Para realizar una petición API RESTful desde PHP se usa la función **file_get_contents** a la que habrá que pasar como parámetro un endpoint:

```php
$endpoint = 'https://miaplicacion.com/api/publicaciones/RickSanchez';
$data = file_get_contents($endpoint);
```

Como una petición API RESTful devuelve un texto en JSON, a continuación se deberá decodificar para usarla en PHP:

```php
$data = json_decode($data);
// recorrer $data para mostrar los datos recibidos
```

Práctica

Actividad 4:

Pokemon.

Actividad 5 – opcional:

Rick & Morty.

Actividad 6 – opcional:

Usando otra API RESTFul.