UD8 – Servicios web RESTful

2º CFGS
Desarrollo de Aplicaciones Web

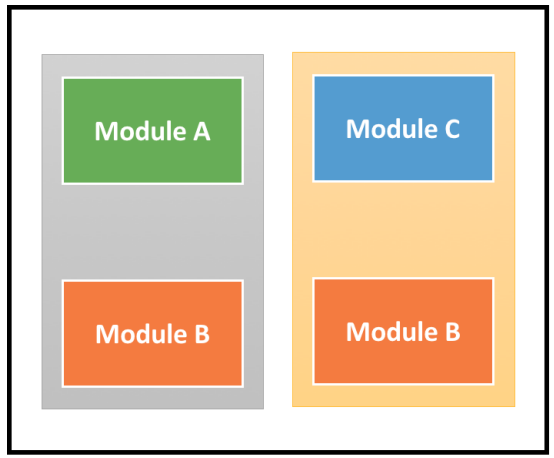
2024-25

1.- SOA: Arquitectura orientada a servicios

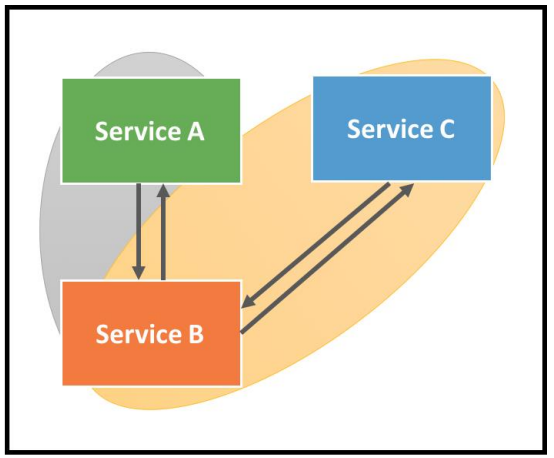
SOA → Service Oriented Architecture

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.



Arquitectura monolítica



Arquitectura orientada a servicios

1.- SOA: 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 (pequeñas aplicaciones) que generan resultados.

Estos resultados 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.- SOA: 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).
- **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

JSON: Java**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:

```php
$jsonString = json_encode($array);
```

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

```php
$array = json_decode($jsonString, 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 HTML por pantalla.

Mediante la cabecera siguiente se indica que se envía un JSON:

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

Práctica

Actividad 1:

Generando JSON desde PHP.

4.- REST

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.

5.- API RESTful

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

Se usan **peticiones HTTP**, generalmente **URL amigables** para:

- Obtener todos los datos de una tabla.
- Obtener todos los datos de un registro de una tabla.
- Obtener todos los datos de un registro de una tabla y sus tablas relacionadas.
- Añadir datos a una tabla.
- Modificar los datos de un registro de una tabla.
- Eliminar un registro de una tabla.
- …

5.- API RESTful

El servidor resuelve las peticiones devolviendo la información en alguno de los siguientes formatos estandarizados:

- XML
- **JSON**
- JSON-LD (Google)
- KML
- KMZ
- CSV
- …

Con API RESTful la respuesta puede contener cualquier tipo de información incluyendo los errores si fuera necesario.

5.- API RESTful

Generalmente cuando un servicio web del tipo RESTful se compone de diferentes recursos en el servidor, en nuestro caso scripts PHP.

Estos recursos en el servidor pueden:

- No recibir datos.
- Recibir datos mediante la URL (GET).
- Recibir datos mediante formularios (POST).

Y la respuesta que generarán estos scripts PHP será en JSON o XML.

En la actualidad la mayoría de servicios web del tipo RESTful generan el resultado en JSON.

5.- API RESTful

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

- **GET**: obtener uno o varios registros 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.

6.- API RESTful - Desarrollo

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 par "URL-tipo petición" se le conoce como **endopoint**.

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

6.- API RESTful - Desarrollo

Ejemplo

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

miaplicación.com/api/publications/RickSanchez (GET)

Y el script que trate la petición:

miaplicacion.com/api/publications.php?user=***NombreUsuario***

Este script devolverá un JSON con todas las publicaciones del usuario.

6.- API RESTful - Desarrollo

Ejemplo

Desarrollo de una calculadora con un servicio web API RESTful:

Las peticiones serían todas tipo GET, y los endpoint podrían ser:

miaplicacion.com/api/add/3/4
miaplicacion.com/api/subtract/3/4
miaplicacion.com/api/multiplication/3/4
miaplicacion.com/api/division/3/4

Todas estas URL se redirigirán a:

miaplicacion.com/calculator.php?operation=$1&number1=$2&number2=$3

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

{ "error": 0, "resultado": 127 } o { "error": "No se puede dividir entre cero."}

6.- API RESTful - Desarrollo

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

Las peticiones serán POST, PUT, PATCH y 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 la siguientes unidades (framework Laravel) se estudiarán todos los tipos de peticiones HTTP y se trabajará con ellas.

7.- API RESTful - Seguridad mediante token

Si el servicio API RESTful se va a usar desde la propia aplicación no será necesario implementar ningún sistema de seguridad.

Si el servicio API RESTful va a ser usado por aplicaciones de terceros se suele implementar un nivel de seguridad mediante un token que se añade a los endpoint.

Se puede implementar la seguridad mediante token para:

- Cualquier tipo de petición.
- Solo para las operaciones que modifican los datos.
- Si se quiere limitar la cantidad de accesos a la API por parte de aplicaciones de terceros.

7.- API RESTful - Seguridad mediante token

Para implementar la seguridad mediante token es necesario registrarse en la aplicación:

- Al registrarse se genera el token y se almacena en la base de datos.
- Desde la sección **account** se puede consultar el token.
- Se añade el token a las peticiones según indique en la documentación.

Endpoint:

miaplicacion.com/api/publications/RickSanchez/e613db7e8b4f50aaee0f

Redirección:

miaplicacion.com/api/publications.php?user=***usuario***&apiKey=***token***

8.- API RESTful - Publicación

Cuando se ha desarrollado un servicio API RESTful se debe publicar la documentación donde se indique cómo usar el servicio:

En la documentación se debe indicar:

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

9.- API RESTful - Ejemplos

Ejemplos de API públicas:

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

https://rickandmortyapi.com/documentation

9.- API RESTful - Ejemplos

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

9.- API RESTful - Ejemplos

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"
            }
        ]
}
```

Práctica

Actividad 2:

API Pokemon.

10.- API RESTful - Consumir

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:

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

```php
$endpoint = 'https://miaplicacion.com/api/publications/RickSanchez';
$data = file_get_contents($endpoint);
$data = json_decode($data);
// recorrer $data para mostrar los datos recibidos
```

10.- API RESTful - Consumir

También se pueden realizar peticiones a servicios web API RESTful desde el cliente mediante JavaScript.

Hoy en día mediante AJAX y las peticiones a API RESTFul se puede cambiar la información que se muestra al usuario sin cambiar de página web.

Por ejemplo:

- Hacer like en una publicación.
- Añadir un comentario a una publicación.
- Cargar más elementos al llegar al final de la página.
- …

Práctica

Actividad 3:

Rick & Morty.