UD 8. Creación de una API Rest

TEORÍA

Desarrollo web en entorno servidor

2do CFGS DAW

Autor: Vanessa Tarí Costa

vanessa.tari@ceedcv.es

2020/2021

Licencia

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

 Importante

 Atención

 Interesante

ÍNDICE

1. INTRODUCCIÓN

La mayoría de las aplicaciones web actuales, disponen de una interfaz de programación de aplicaciones (API) que los clientes pueden utilizar para interactuar con la aplicación. Una API debe ser independiente de la plataforma, es decir cualquier cliente puede usar la API con independencia de cómo esté implementada internamente, para ello es necesario el uso de protocolos y mecanismos por medio de los cuales el cliente y el servicio web puedan comunicarse. Además la API web debe poder agregar nuevas funcionalidades sin que afecte a las aplicaciones cliente.

2. API REST

En 2000, Roy Fielding propuso la transferencia de estado representacional (REST), como enfoque de arquitectura para el diseño de servicios web. En esta unidad nos vamos a centrar en el diseño de API REST para HTTP.

La principal ventaja que ofrece REST sobre HTTP es que al usar estándares abiertos, las aplicaciones cliente pueden usar cualquier lenguaje o herramientas que puedan generar solicitudes HTTP con independencia de la implementación del servicio web REST.

Características

Las principales características de REST son:

- **Es un protocolo cliente/servidor sin estado**: cada petición HTTP contiene toda la información necesaria sin necesidad de que el cliente o el servidor deban recordar el estado previo para satisfacerla.
- Las operaciones más importantes relacionadas con los datos en cualquier sistema REST y la especificación HTTP son cuatro: **POST** (crear), **GET** (leer y consultar), **PUT** (editar) y **DELETE** (eliminar).
- Los objetos en REST siempre se obtienen o manipulan a partir de una URI. La URI es el identificador único de cada recurso. Ejemplo de un URI de un pedido: **https://domino.com/pedidos/1.**
- Los clientes interactúan con un servicio mediante el intercambio de representaciones tipo JSON o XML. Ejemplo de JSON:

```
{"pedidoID":1,"pedidoValor":78.90,"productoID":1,"cantidad":1}
```

- Las API REST se controlan mediante vínculos de hipermedia contenidos en la representación. Por ejemplo, a continuación se muestra la representación de un pedido que contiene vínculos para obtener o actualizar el cliente asociado con el pedido.

```
{
      "pedidoID":3,
      "productoID":2,
      "cantidad":4,
      "pedidoValor":16.60,
      "enlaces": [
      {"rel":"producto","href":"https://dominio.com/clientes/3",
"action":"GET" },
      {"rel":"producto","href":"https://dominio.com/clientes/3",
"action":"PUT" }
      ]
}
```

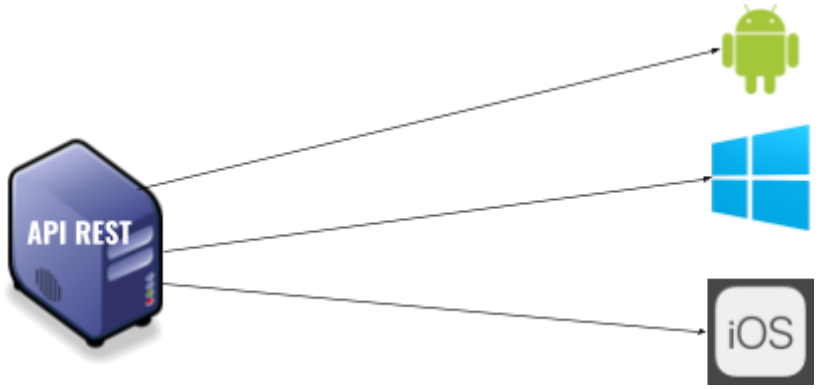
Para cualquier API REST es obligatorio disponer del principio de **HATEOAS** (Hypermedia As The Engine Of Application State – Hipermedia Como Motor del Estado de la Aplicación) para ser una verdadera API REST. Este principio es el que define que cada vez que se

hace una petición al servidor y éste devuelve una respuesta, parte de la información que contendrá serán los hipervínculos de navegación asociada a otros recursos del cliente.

Ventajas

Las principales ventajas de REST son las siguientes:

- **Separación entre el cliente y el servidor:** el protocolo REST separa completamente la interfaz de cliente del almacenamiento de datos. Esto tiene algunas ventajas, por ejemplo la portabilidad, la escalabilidad de proyectos y la evolución de los distintos componentes de los desarrollos de forma independiente.



- **Independencia de tecnologías/lenguajes**: puedes desarrollar en cualquier tipo de tecnología o lenguaje de programación el API REST, con indiferencia de que en un futuro se decida cambiar la implementación, siempre y cuando se respete el contrato, es decir que se sigan manteniendo las mismas operaciones en la API.
- **Experiencia de usuario:** aunque depende un poco de cómo esté hecha la parte del cliente, el desarrollo de sitios web basados en API pueden dar mejor rendimiento que uno tradicional. Cuando haces una solicitud al servidor lo que tienes como respuesta son datos planos, que requieren tiempos de transferencia menores que si esos mismos datos los recibieras mezclados con el HTML/CSS de la presentación.
- **REST requiere menos recursos en el servidor:** al no mantener el estado, no requiere memoria, se pueden atender más peticiones. Además, no requiere escribir el HTML, por lo tanto, tienes menos procesamiento en el servidor.

3. DEFINICIÓN DE OPERACIONES

El protocolo HTTP define una serie de métodos que asignan significado semántico a una solicitud. Los métodos HTTP comunes que usan la mayoría de las API web RESTful son:

- **GET** recupera una representación del recurso en el URI especificado. El cuerpo del mensaje de respuesta contiene los detalles del recurso solicitado.
- **POST** crea un nuevo recurso en el URI especificado. El cuerpo del mensaje de solicitud proporciona los detalles del nuevo recurso.
- **PUT** crea o sustituye el recurso en el URI especificado. El cuerpo del mensaje de solicitud especifica el recurso que se va a crear o actualizar.
- **PATCH** realiza una actualización parcial de un recurso. El cuerpo de la solicitud especifica el conjunto de cambios que se aplican al recurso.
- **DELETE** quita el recurso en el URI especificado.

4. CREACIÓN DE UNA API REST EN PHP

Vamos a crear una API muy sencilla, donde vamos a tener todos los métodos que necesitamos dentro del mismo archivo index.php, en el próximo punto veremos cómo se hace en Codeigniter de una forma más profesional.

Lo primero que vamos a hacer es crear la línea que mostrará la salida del servicio web en formato JSON:

```
header('Content-Type: application/json');
```

Ahora, vamos a crear una función para gestionar los errores, y cuya salida será también un JSON.

```
function error($mensaje){
    $respuesta = array(
        "tipo" => "error",
        "msj" => $mensaje
    );
    return $respuesta;
}
```

Vamos a crear una función para la conexión a la base de datos, aquí perfectamente podríamos usar la clase Database.php creada en la unidad 6, pero para simplificar, lo vamos a poner en el mismo archivo. En este ejemplo, nos vamos a conectar a la base de datos de nba del ejercicio no evaluable visto en la unidad 5..

```
function conectar(){
    $usuario = "root";
    $password = "";
    $db = new PDO('mysql:host=localhost;dbname=nba, $usuario,
$password);
    return $db;
}
```

La función devolverá un array con todos los datos que existen en la base de datos o si algo no va bien un array con el error formateado por la función error().

```
function listar(){
    $jugadores = array();
    try {
        $db = conectar();
        //Tenemos que ponerle un alias al campo Nombre de la tabla equipo
para que no se confunda con el campo Nombre del jugador
        $sql = "SELECT j.*, e.Ciudad, e.Conferencia, e.Division, e.Nombre AS
Nombre_equipo FROM `jugadores` AS j INNER JOIN equipos AS e ON
j.ID_equipo=e.id";
        $prepared_statement = $db->prepare($sql);
        $prepared_statement->execute();
        foreach ($prepared_statement->fetchAll(PDO::FETCH_ASSOC) as $row) {
            $data = array(
                "id" => $row['id'],
                "nombre" => $row['Nombre'],
                "anyo_inicio" => $row['Anyo_Inicio'],
                "anyo_fin" => $row['Anyo_Fin'],
                "posicion" => $row['Posicion'],
                "altura" => $row['Altura'],
                "peso" => $row['Peso'],
                "nacimiento" => $row['Nacimiento'],
                "procedencia" => $row['Procedencia'],
                "equipo" => array(
                    "id" => $row['ID_equipo'],
                    "nombre" => $row['Nombre_equipo'],
                    "ciudad" => $row['Ciudad'],
                    "conferencia" => $row['Conferencia'],
                    "division" => $row['Division'],
                ),
                "url" =>
url("/apirest/index.php?opcion=obtener&id=".$row['id'])
            );
            array_push($jugadores, $data);
        }
```

```
    } catch (PDOException $e) {
        return error($e->getMessage());
    }

    return $jugadores;
}
```

Hay que tener en cuenta, que para que el API Rest siga el principio de **HATEOAS** debemos añadir a cada item del array la referencia o enlace a los datos de ese elemento. Esto se haría añadiendo al JSON el campo url correspondiente. En este caso, lo estamos añadiendo al array de esta forma: `"url" => url("/apirest/index.php?opcion=obtener&id=".$row['id'])`

Para poder obtener una url correcta, vamos a hacer uso de la siguiente función que nos devolverá la base url donde se encuentra alojada nuestra página web.

```
function url($segmento){
    if(isset($_SERVER['HTTPS'])){
        $protocol = ($_SERVER['HTTPS'] && $_SERVER['HTTPS'] != "off") ?
"https" : "http";
    }
    else{
        $protocol = 'http';
    }
    return $protocol . "://" . $_SERVER['HTTP_HOST'] . $segmento;
}
```

Ahora nos falta crear la función que nos devuelve un único elemento pasándole el id del elemento a buscar.

```
function obtener($id){
    $jugador = array();
    try {
        $db = conectar();
        $sql = "SELECT j.*, e.Ciudad, e.Conferencia, e.Division,
e.Nombre AS Nombre_equipo FROM `jugadores` AS j INNER JOIN equipos AS e
ON j.ID_equipo=e.id WHERE j.id=:id";
        $prepared_statement = $db->prepare($sql);
        $prepared_statement->bindParam(':id', $id, PDO::PARAM_INT);
        $prepared_statement->execute();
        foreach ($prepared_statement->fetchAll(PDO::FETCH_ASSOC) as
$row) {
```

```
            $jugador = array(
                "id" => $row['id'],
                "nombre" => $row['Nombre'],
                "anyo_inicio" => $row['Anyo_Inicio'],
                "anyo_fin" => $row['Anyo_Fin'],
                "posicion" => $row['Posicion'],
                "altura" => $row['Altura'],
                "peso" => $row['Peso'],
                "nacimiento" => $row['Nacimiento'],
                "procedencia" => $row['Procedencia'],
                "equipo" => array(
                    "id" => $row['ID_equipo'],
                    "nombre" => $row['Nombre_equipo'],
                    "ciudad" => $row['Ciudad'],
                    "conferencia" => $row['Conferencia'],
                    "division" => $row['Division'],
                ),
                "url" =>
url("/apirest/index.php?opcion=obtener&id=".$row['id'])
            );
        }

    } catch (PDOException $e) {
        return error($e->getMessage());
    }
    return $jugador;
}
```

Finalmente, añadimos el código que nos redirecciona a cada una de las funciones:

```
if($_GET['opcion']=='listar'){
    echo json_encode(listar());
}elseif($_GET['opcion']=='obtener'){
    echo json_encode(obtener($_GET["id"]));
}
else{
    echo json_encode(error("No se ha especificado parametro"));
}
```

Para probarlo, nos podemos ir directamente al navegador o utilizar alguna herramienta externa que nos permita hacer este tipo de solicitudes, yo concretamente utilizo Postman que nos ayuda para el desarrollo de nuestras APIs y os lo recomiendo para vuestros futuros proyectos.

Si accedemos a la URL *http://localhost/apirest/index.php?opcion=listar*, el resultado sería el siguiente:


```
[
	{
	"id": "2",
	"nombre": "Zaid Abdul-Aziz",
	"anyo_inicio": "1969",
	"anyo_fin": "1978",
	"posicion": "C-F",
	"altura": "6",
	"peso": "235",
	"nacimiento": "April 7, 1946",
	"procedencia": "Iowa State University",
	"equipo": {
		"id": "2",
		"nombre": "Celtics",
		"ciudad": "Boston",
		"conferencia": "Este",
		"division": "Central"
	},
	"url": "http://localhost/apirest/index.php?opcion=obtener&id=2"
	},
	{
	"id": "3",
	"nombre": "Alaa Abdelnaby",
	"anyo_inicio": "1991",
	"anyo_fin": "1995",
	"posicion": "F-C",
	"altura": "6",
	"peso": "240",
	"nacimiento": "June 24, 1968",
	"procedencia": "Duke University",
	"equipo": {
		"id": "1",
		"nombre": "Hawks",
		"ciudad": "Atlanta",
		"conferencia": "Este",
		"division": "Atlantico"
	},
	"url": "http://localhost/apirest/index.php?opcion=obtener&id=3"
	},
	{
	"id": "7",
	"nombre": "Mahmoud Abdul-Rauf",
	"anyo_inicio": "1991",
	"anyo_fin": "2001",
	"posicion": "G",
	"altura": "6",
	"peso": "162",
	"nacimiento": "March 9, 1969",
	"procedencia": "Louisiana State University",
	"equipo": {
		"id": "4",
		"nombre": "Bulls",
		"ciudad": "Chicago",
```

```
            "conferencia": "Este",
            "division": "Sureste"
        },
        "url": "http://localhost/apirest/index.php?opcion=obtener&id=7"
        },
        {
        "id": "8",
        "nombre": "Tariq Abdul-Wahad",
        "anyo_inicio": "1998",
        "anyo_fin": "2003",
        "posicion": "F",
        "altura": "6",
        "peso": "223",
        "nacimiento": "November 3, 1974",
        "procedencia": "San Jose State University",
        "equipo": {
            "id": "4",
            "nombre": "Bulls",
            "ciudad": "Chicago",
            "conferencia": "Este",
            "division": "Sureste"
        },
        "url": "http://localhost/apirest/index.php?opcion=obtener&id=8"
        },
        {
        "id": "9",
        "nombre": "Shareef Abdur-Rahim",
        "anyo_inicio": "1997",
        "anyo_fin": "2008",
        "posicion": "F",
        "altura": "6",
        "peso": "225",
        "nacimiento": "December 11, 1976",
        "procedencia": "University of California",
        "equipo": {
            "id": "6",
            "nombre": "Mavericks",
            "ciudad": "Dallas",
            "conferencia": "Oeste",
            "division": "Medio Oest"
        },
        "url": "http://localhost/apirest/index.php?opcion=obtener&id=9"
        },
        {
        "id": "10",
        "nombre": "Tom Abernethy",
        "anyo_inicio": "1977",
        "anyo_fin": "1981",
        "posicion": "F",
        "altura": "6",
        "peso": "220",
        "nacimiento": "May 6, 1954",
        "procedencia": "Indiana University",
        "equipo": {
            "id": "6",
            "nombre": "Mavericks",
```

```
                "ciudad": "Dallas",
                "conferencia": "Oeste",
                "division": "Medio Oest"
        },
        "url": "http://localhost/apirest/index.php?opcion=obtener&id=10"
        },
        {
        "id": "11",
        "nombre": "Forest Able",
        "anyo_inicio": "1957",
        "anyo_fin": "1957",
        "posicion": "G",
        "altura": "6",
        "peso": "180",
        "nacimiento": "July 27, 1932",
        "procedencia": "Western Kentucky University",
        "equipo": {
                "id": "2",
                "nombre": "Celtics",
                "ciudad": "Boston",
                "conferencia": "Este",
                "division": "Central"
        },
        "url": "http://localhost/apirest/index.php?opcion=obtener&id=11"
        },
        {
        "id": "12",
        "nombre": "John Abramovic",
        "anyo_inicio": "1947",
        "anyo_fin": "1948",
        "posicion": "F",
        "altura": "6",
        "peso": "195",
        "nacimiento": "February 9, 1919",
        "procedencia": "Salem International University",
        "equipo": {
                "id": "3",
                "nombre": "Pelicans",
                "ciudad": "New Orleans",
                "conferencia": "Oeste",
                "division": "Medio Oeste"
        },
        "url": "http://localhost/apirest/index.php?opcion=obtener&id=12"
        }
]
```


Se puede observar que dentro de cada item se encuentra la URL a la que apunta ese ítem en concreto, por lo tanto, cumpliriamos así con el principio de **HATEOAS**.

Debemos tener en cuenta, que se podrían hacer tantas funciones como fueran necesarias, cuantos más servicios web tenga nuestra API mejor servicio daremos a los clientes.

Si accedemos a la URL de cada ítem, el resultado sería únicamente ese dato concreto:

```
{
    "id": "9",
    "nombre": "Shareef Abdur-Rahim",
    "anyo_inicio": "1997",
    "anyo_fin": "2008",
    "posicion": "F",
    "altura": "6",
    "peso": "225",
    "nacimiento": "December 11, 1976",
    "procedencia": "University of California",
    "equipo": {
        "id": "6",
        "nombre": "Mavericks",
        "ciudad": "Dallas",
        "conferencia": "Oeste",
        "division": "Medio Oest"
    },
    "url": "http://localhost/apirest/index.php?opcion=obtener&id=9"
}
```


5. APIS DE INTERÉS

- API de Pokemon: https://pokeapi.co/
- API de Flickr: https://www.flickr.com/services/api/
- API de Booking: https://developers.booking.com/api/commercial/index.html?version=2.7&page_url=getting-started
- API de Youtube: https://developers.google.com/youtube/v3/getting-started

6. WEBGRAFÍA

[1] Simple api rest con php
[2] API REST: qué es y cuáles son sus ventajas en el desarrollo de proyectos
[3] Guía de diseño de una API - Best practices for cloud applications
[4] RESTful Resource Handling — CodeIgniter 4.0.4 documentation