Metodologías y Tecnologías de la Integración de Sistemas

Ingeniería Informática

Arancha Ferrero Ortiz de Zárate
72850555X

Practica 1
Interoperabilidad WSDL

17 de febrero del 2018

Descripción de la práctica

El objetivo de esta práctica es crear un WSDL desde cero (top-down), sin necesidad de crear un servicio web.

Para ello, se define primero el WSDL. A continuación, creamos el servicio web que utiliza el WSDL previamente creado. Además, para almacenar algunos datos necesarios en el servicio web, hemos de crear una base de datos MySql.

Para la comprobación del correcto funcionamiento de los servicios web, emplearemos la herramienta: SoapUI.

Finalmente, para poder comprobar visual y fácilmente dichos servicios, crearé un cliente web en PHP empleando el popular framework Laravel.

Requisitos del sistema

- ❏ Java 8
- ❏ PHP 7
- ❏ Composer
- ❏ XAMPP 7.1.14

Archivos de la práctica

Junto a esta memoria, adjunto el proyecto dinámico de Eclipse *'practica1'*, con el WSDL y el servicio web, el proyecto cliente en Laravel llamado *'cliente-web-laravel'* y el archivo SQL con la base de datos inicializada, *schema.sql*.

Recomiendo ubicar el proyecto de Eclipse dentro de la carpeta C:\MTIS\workspace.

La ubicación del proyecto del cliente web es indiferente, pero es importante abrir una terminal en su ubicación para poder ponerla en marcha.

Instalación del software necesario

1. **Descargar e instalar XAMPP 7.1.14:** Para ello,dentro de su página oficial, podemos obtener links a sus instaladores para diferentes SOs:
    a. Windows
    b. Linux
    c. OS X

    Con la instalación de XAMPP, se nos instalará PHP 7, así que no hay que preocuparse por ello.

2. **Descargar e instalar Composer**: En su web oficial nos ofrecen una amplia variedad de formas de instalarlo. Para Windows, sugiero el instalador que nos proporcionan: Composer-Setup.exe. El cuál, además de instalarnos composer, nos configura la variable de entorno *composer* desde cualquier directorio.

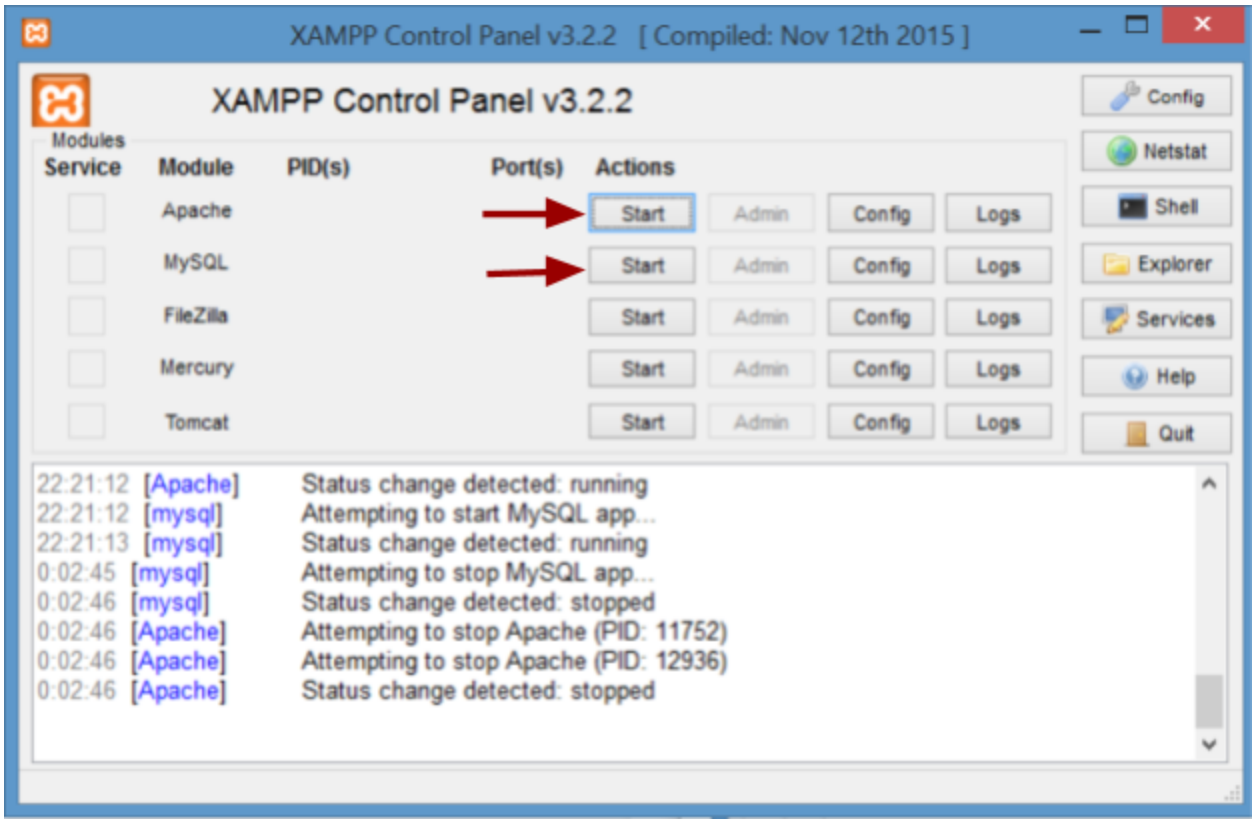
3. **Instalar Laravel:** Una vez instalado Composer, podemos instalar Laravel fácilmente desde una terminal ejecutando:

```
composer global require "laravel/installer"
```

Puesta en marcha

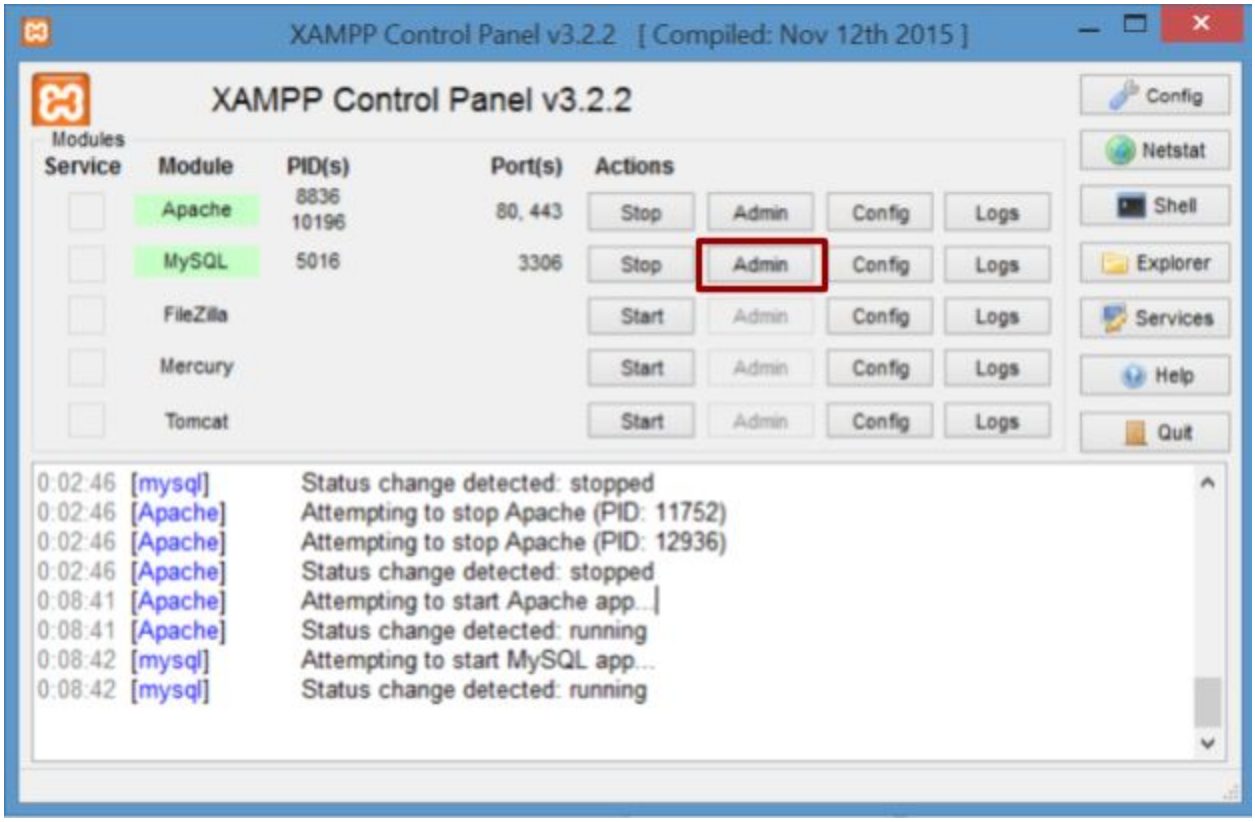
1. Iniciar servidores Apache y MySql en XAMPP

   Una vez abierto XAMPP, iniciamos los servidores Apache y MySQL, pulsamos en Start:

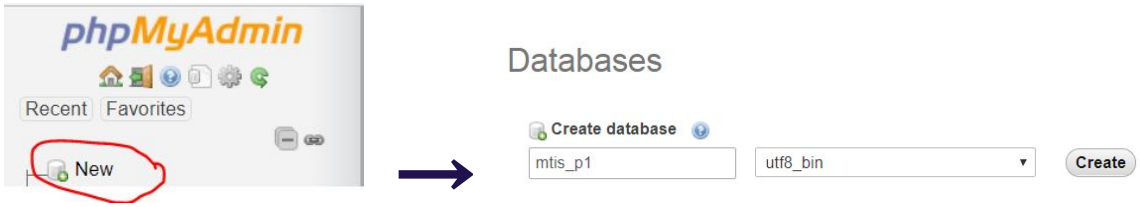


2. Crear base de datos MySql

Crearemos la base de datos con ayuda de phpMyAdmin. Para ello, en XAMPP, pulsaremos sobre el botón de Admin habilitado al iniciar el servidor MySql:

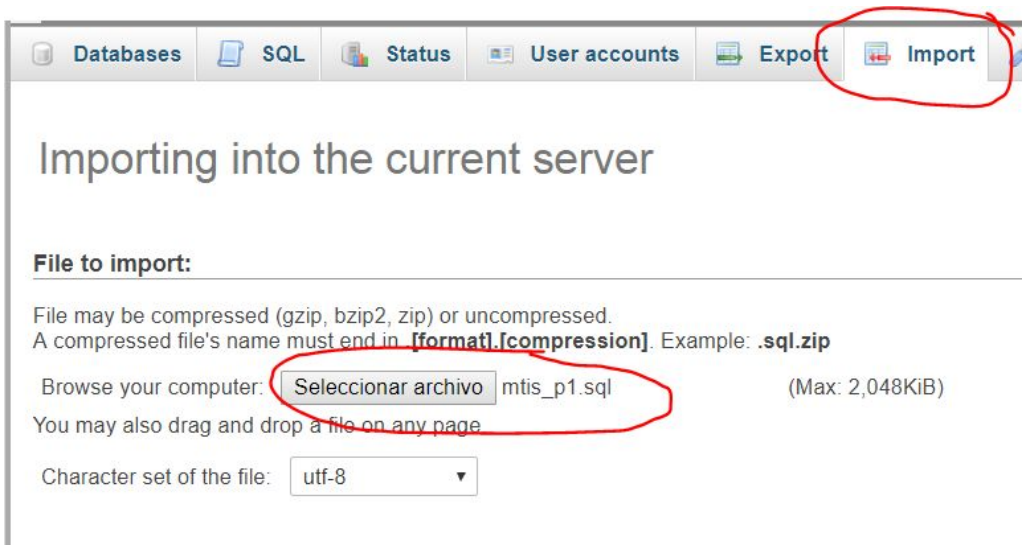


Ahora, en el panel de phpMyAdmin, crearemos una nueva base de datos con el nombre *'mtis_p1'* y codificación *utf8_bin*:

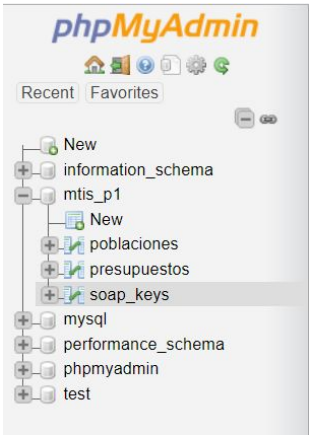


Para inicializar la Base de Datos emplearemos el archivo schema.sql adjuntado, con la base de datos exportada.

En el panel de phpMyAdmin, dentro de la pestaña de Import, podemos seleccionar nuestro archivo schema.sql, y la codificación utf-8, e importarlo, pulsando (más abajo) sobre el botón Go:



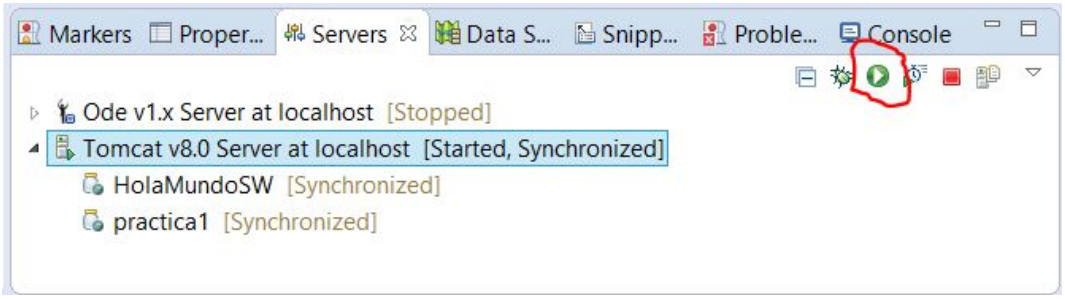
Finalmente, obtendremos el siguiente resultado con nuestras tablas creadas y con datos:



P.D.: En la tabla 'soap_keys', se pueden consultar cuáles existen actualmente.

3. Iniciar servidor Tomcat

Para iniciar el servidor Tomcat con el servicio web, necesitamos abrir el proyecto *'practica1'* con Eclipse. Una vez abierto, en el menú de abajo, en la pestaña Servers, seleccionamos Tomcat y pulsamos sobre el botón de "play":



4. Iniciar servidor Laravel

A la hora de iniciar el servidor en Laravel con el cliente, necesitamos abrir una terminal en la ruta donde se halla el proyecto *cliente-web-laravel* adjuntado en la práctica. En él ejecutamos (sólo la primera vez):

```
composer update
```

Y a continuación, iniciamos el servidor con:

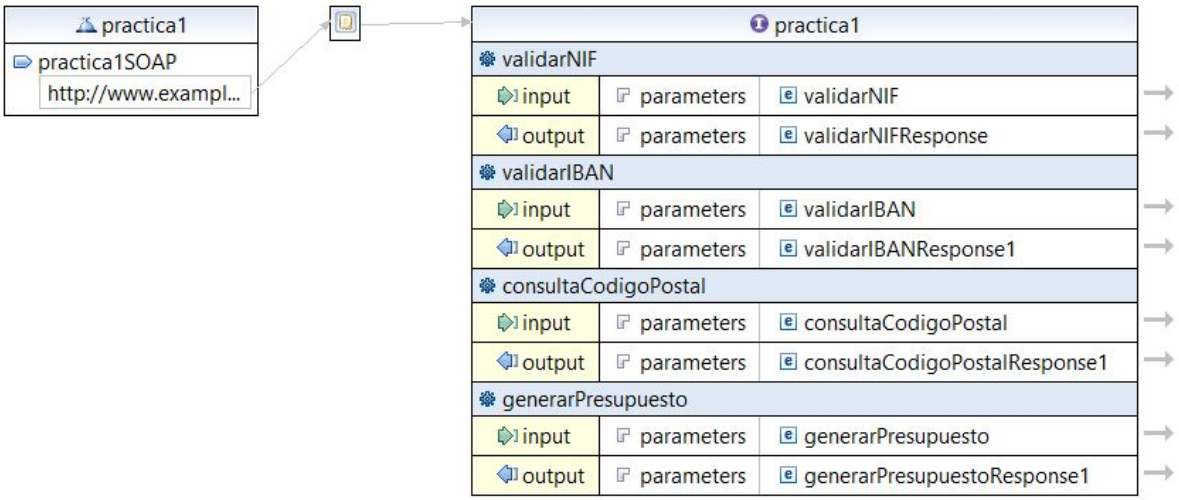
```
php artisan serve
```

5. Abrir URL del cliente en el Navegador

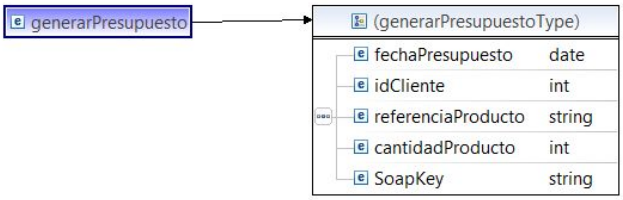
Finalmente, podemos comprobar el funcionamiento de nuestra práctica desde la URL: localhost:8000

Pasos seguidos para las elaboración de la práctica

El primer paso ha sido definir en el contrato WSDL las cuatro operaciones especificadas en el enunciado de la práctica con sus entradas y salidas: *validarNIF, validarIBAN, consultaCódigoPostal y generaPresupuesto*.



Ejemplo de salidas en la operación generaPresupuesto:



Seguidamente, he creado el servicio web a partir del contrato WSDL creado. Con ello, se me crean las distintas clases de entrada y respuesta asociadas a cada operación.

Además, también se crea el archivo Practica1Skeleton.java, en el cuál estará la lógica de cada operación. Por ejemplo, la operación *validarNIF*:

```
public org.example.www.practica1.ValidarNIFResponse validarNIF(org.example.www.practica1.ValidarNIF validarNIF) {
    ValidarNIFResponse respuesta = new ValidarNIFResponse();
    // Check soapKey
    String soapKey = validarNIF.localSoapKey;
    boolean isValidSoapKey = checkSoapKey(soapKey);
    if(!isValidSoapKey) {
        throw new  java.lang.UnsupportedOperationException("Soap Key introduced is invalid");
    }

    String nif = validarNIF.localNif;

    // Validando NIF
    boolean correcto = false;
    Pattern pattern = Pattern.compile("(\\d{1,8})([TRWAGMYFPDXBNJZSQVHLCKEtrwagmyfpdxbnjzsqvhlcke])");
    Matcher matcher = pattern.matcher(nif);
    if (matcher.matches()) {
        String letra = matcher.group(2);
        String letras = "TRWAGMYFPDXBNJZSQVHLCKE";
        int index = Integer.parseInt(matcher.group(1));
        index = index % 23;
        String reference = letras.substring(index, index + 1);

        if (reference.equalsIgnoreCase(letra)) {
            correcto = true;
        } else {
            correcto = false;
        }
    } else {
        correcto = false;
    }

    respuesta.localIsValid = correcto;
    return respuesta;
}
```

A continuación, con ayuda de SOAP UI, se comprueba el correcto funcionamiento de cada operación (previamente a esto, se crea e inicializa la base de datos):

Finalmente, cree el proyecto cliente web con Laravel:

```
laravel new cliente-web-laravel
```

Me serví de la librería de Php, que ya viene instalada con Laravel: SoapClient, para realizar la conexión del cliente con el servicio web cada vez que se realizaba una operación:

```
$client = new SoapClient($this->wsdl, array('exceptions' => 0));
$response = $client->validarNIF(array('nif' => $nif, 'SoapKey' => $soapKey));
```

Lo cuál me devolvía un objeto $response, que en el caso de la validación del NIF, para obtener el elemento respuesta *isValid,* devuelto por SOAP, accedemos: `$response->isValid`

P.D.: Para la conexión con la base de datos MySql, me serví de una clase, *MySqlConnect.java*, proporcionada en una respuesta en StackOverflow.