

PROGRAMACION EN ANDROID CON JAVA PRACTICA No. 8

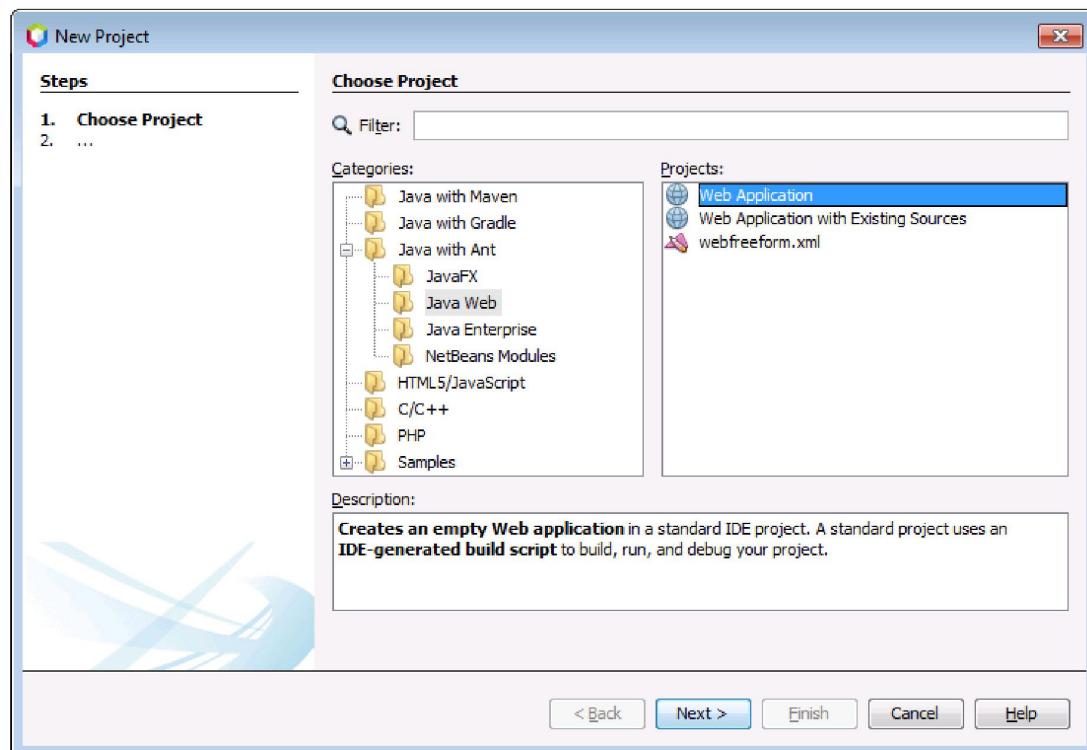
Consumir servicios SOAP desde Android

PARTE 1.

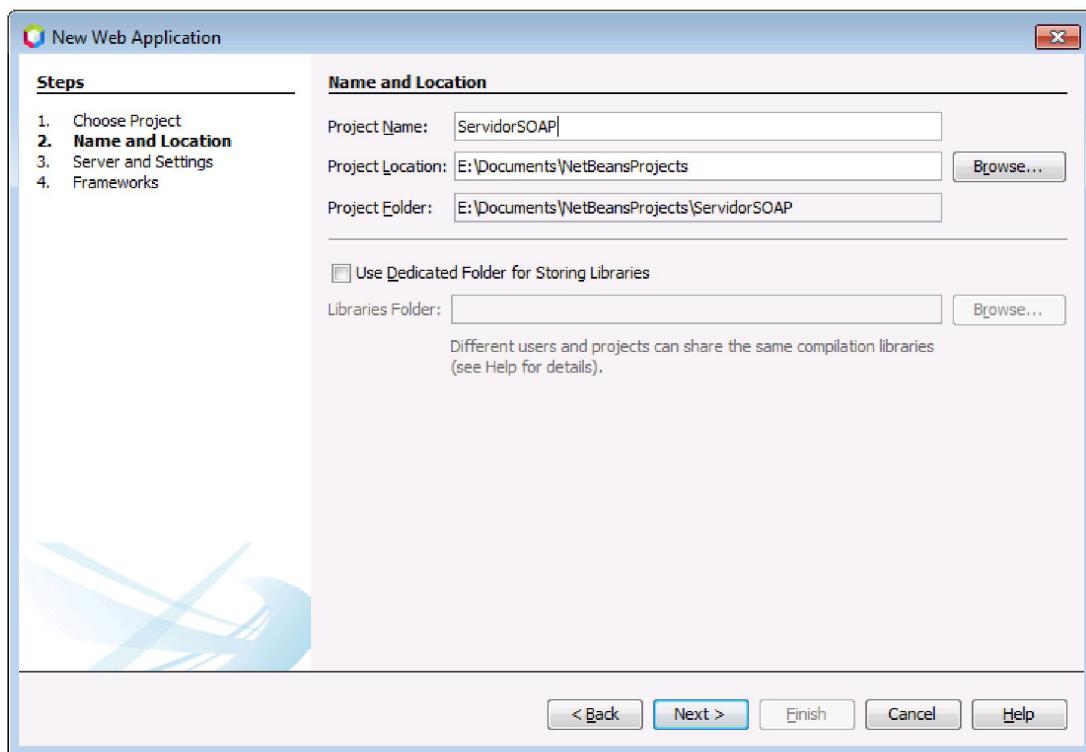
Creación del Servidor SOAP

Abra la IDE de desarrollo Netbeans

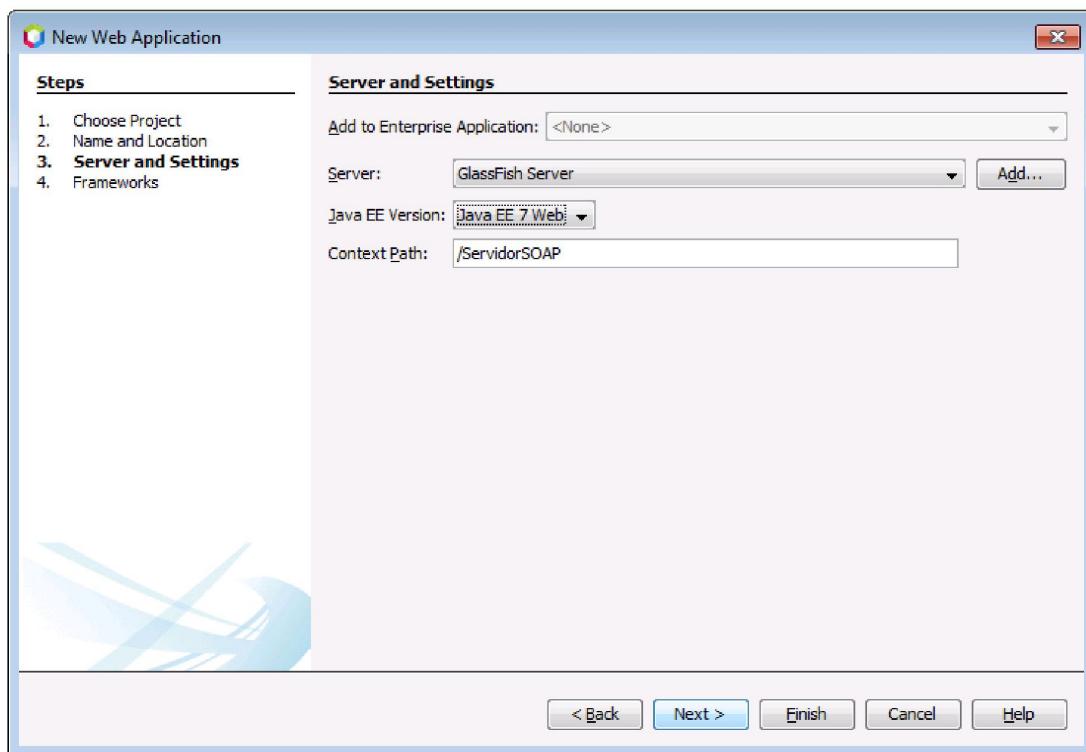
Cree un nuevo proyecto haciendo clic sobre la opción *New Project* del menú *File*. Seleccionamos como tipo de proyecto Web Application de la categoría Java Web y hacemos clic en el botón Next.



Cambiamos el nombre al proyecto a ServidorSOAP y hacemos clic en el botón Next.

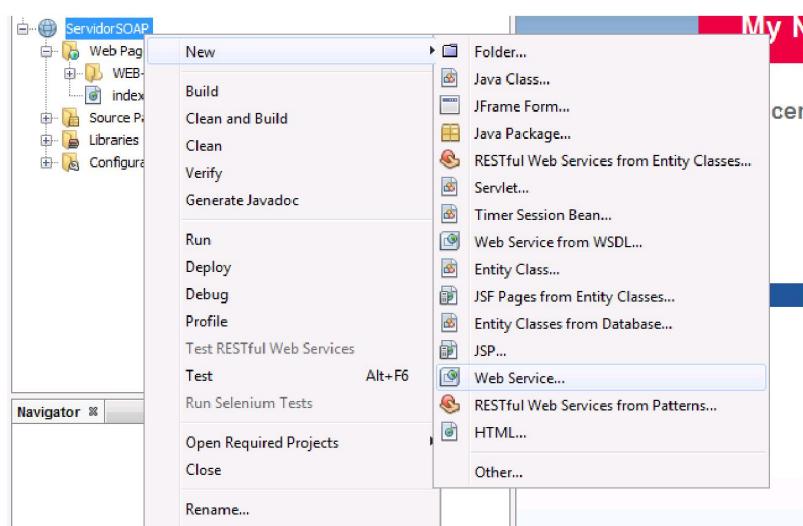


Seleccionamos como servidor GlassFish Server, Java EE 7 Web y como Context Path /ServidorSOAP.

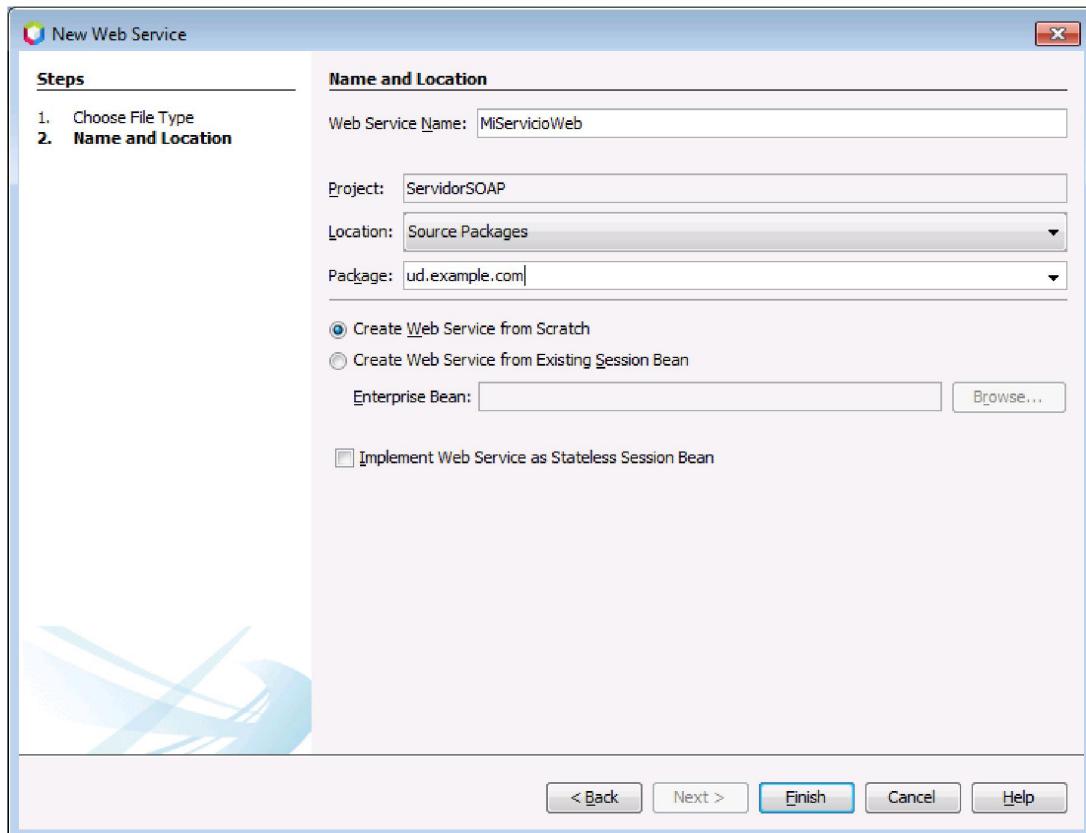


Hacemos clic en el botón Finish.

Adicionamos al proyecto ServidorSOAP un nuevo Servicio Web, para ello hacemos clic derecho del mouse sobre el proyecto y seleccionamos la opción Web service del submenú New.



En la nueva ventana colocaremos como nombre del servicio MiServicioWeb y como paquete ud.example.com, cambiando estos parámetros hacemos clic en el botón Finish.



Agregando Código:

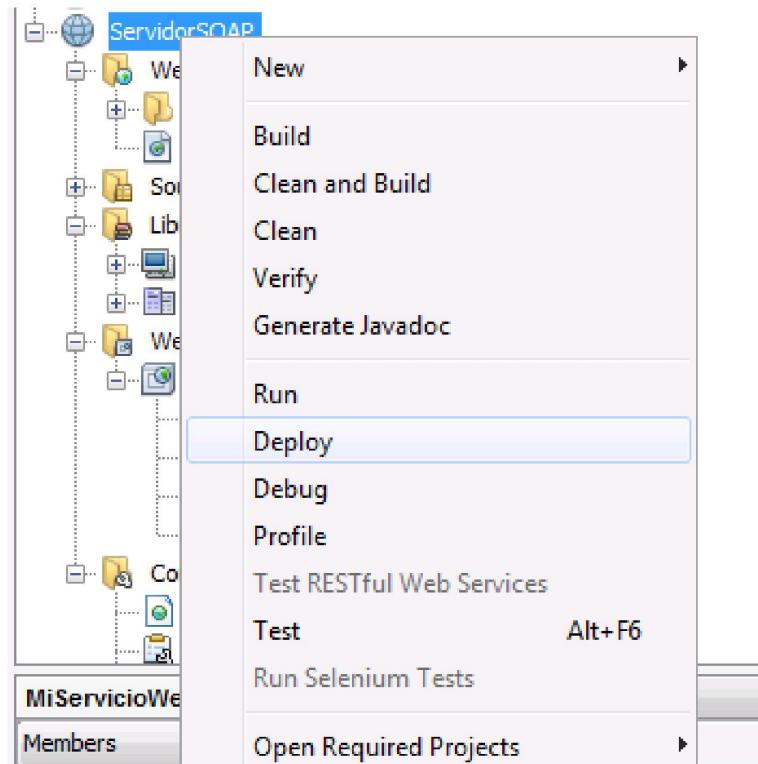
Sobre la clase MiServicioWeb.java agregamos el siguiente código:

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ud.example.com;

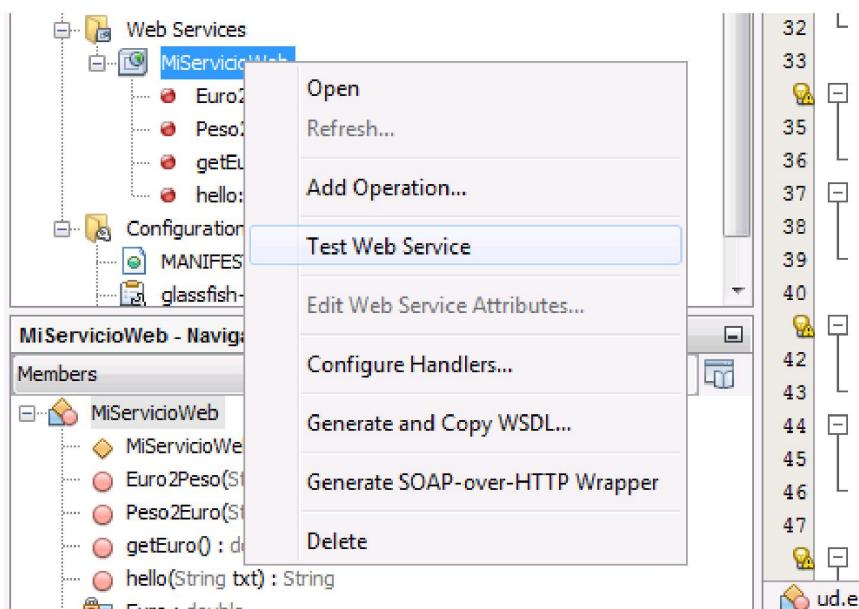
import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;
```

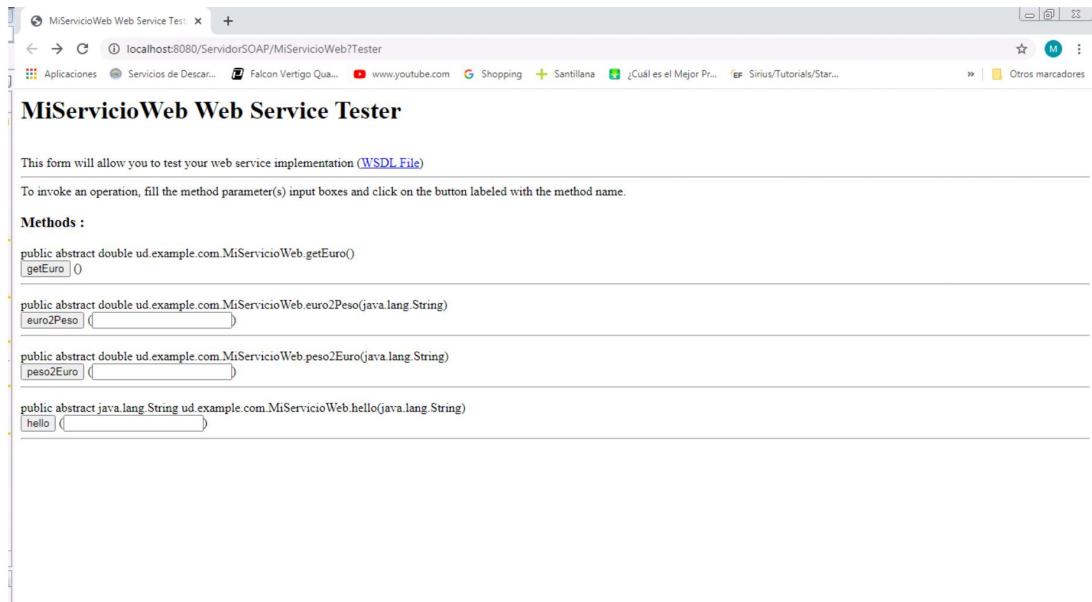
```
/**  
 *  
 * @author felipe  
 */  
@WebService(serviceName = "MiServicioWeb")  
public class MiServicioWeb {  
    private double Euro;  
    public MiServicioWeb(){  
        Euro = 2645.33;  
    }  
    /**  
     * This is a sample web service operation  
     */  
    @WebMethod(operationName = "hello")  
    public String hello(@WebParam(name = "name") String txt) {  
        return "Hello " + txt + " !";  
    }  
    /**  
     * Web service operation  
     */  
    @WebMethod(operationName = "getEuro")  
    public double getEuro() {  
        return Euro;  
    }  
    /**  
     * Web service operation  
     */  
    @WebMethod(operationName = "Peso2Euro")  
    public double Peso2Euro(@WebParam(name = "valor") String valor1) {  
        return Double.parseDouble(valor1)/Euro;  
    }  
    /**  
     * Web service operation  
     */  
    @WebMethod(operationName = "Euro2Peso")  
    public double Euro2Peso(@WebParam(name = "valor") String valor1) {  
        return Double.parseDouble(valor1)*Euro;  
    }  
}
```

Para terminar, publicamos nuestro proyecto (Deploy), esto lo logramos haciendo clic derecho, sobre el proyecto, y ejecutar la opción deploy



Por último, para probar el servicio SOAP podemos hacer clic derecho sobre MiServicioWeb y ejecute la opción Test Web Service, el cual desplegará en un browser el servicio SOAP.





This form will allow you to test your web service implementation ([WSDL File](#))

To invoke an operation, fill the method parameter(s) input boxes and click on the button labeled with the method name.

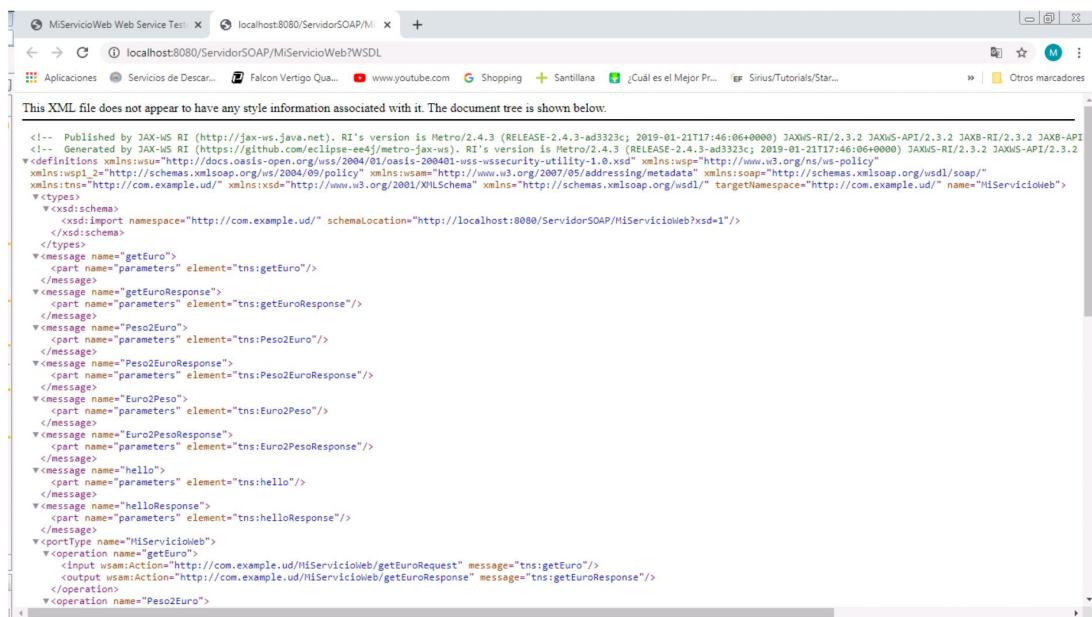
Methods :

```
public abstract double ud.example.com.MiServicioWeb.getEuro()
getEuro()
```

```
public abstract double ud.example.com.MiServicioWeb.euro2Peso(java.lang.String)
euro2Peso()
```

```
public abstract double ud.example.com.MiServicioWeb.peso2Euro(java.lang.String)
peso2Euro()
```

```
public abstract java.lang.String ud.example.com.MiServicioWeb.hello(java.lang.String)
hello()
```

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<!-- Published by JAX-WS RI (http://jax-ws.java.net). RI's version is Metro/2.4.3 (RELEASE-2.4.3-403823c; 2019-01-21T17:46:06+0000) JAXWS-RI/2.3.3 JAXWS-API/2.3.2 JAXB-RI/2.3.2 JAXB-API/2.3.2 definition generated on 2019-01-21T17:46:06+0000 from wsdl: <a href="https://github.com/eclipse-ee4j/jax-ws-javasdk">https://github.com/eclipse-ee4j/jax-ws-javasdk; RI's version is Metro/2.4.3 (RELEASE-2.4.3-403823c; 2019-01-21T17:46:06+0000) JAXWS-RI/2.3.3 JAXWS-API/2.3.3 definition generated on 2019-01-21T17:46:06+0000 from wsdl: -->
<definitions xmlns="http://docs.oasis-open.org/ws-2004/01/oasis-200401-ws-sxip-security-utility-1.0.xsd" xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://com.example.ud/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://com.example.ud/" name="MiServicioWeb">
  <types>
    <xsd:schema>
      <xsd:import namespace="http://com.example.ud/" schemaLocation="http://localhost:8080/ServidorSOAP/MiServicioWeb?xsd=1"/>
    </xsd:schema>
  </types>
  <message name="getEuro">
    <part name="parameters" element="tns:getEuro"/>
  </message>
  <message name="getEuroResponse">
    <part name="parameters" element="tns:getEuroResponse"/>
  </message>
  <message name="Peso2Euro">
    <part name="parameters" element="tns:Peso2Euro"/>
  </message>
  <message name="Peso2EuroResponse">
    <part name="parameters" element="tns:Peso2EuroResponse"/>
  </message>
  <message name="Euro2Peso">
    <part name="parameters" element="tns:Euro2Peso"/>
  </message>
  <message name="Euro2PesoResponse">
    <part name="parameters" element="tns:Euro2PesoResponse"/>
  </message>
  <message name="hello">
    <part name="parameters" element="tns:hello"/>
  </message>
  <message name="helloResponse">
    <part name="parameters" element="tns:helloResponse"/>
  </message>
</definitions>
<operation name="getEuro">
  <input wsam:Action="http://com.example.ud/MiServicioWeb/getEuroRequest" message="tns:getEuro"/>
  <output wsam:Action="http://com.example.ud/MiServicioWeb/getEuroResponse" message="tns:getEuroResponse"/>
</operation>
<operation name="Peso2Euro">
```

De esta manera se comprueba que está corriendo el servicio, los parámetros y métodos que este nos permite ejecutar y la posible respuesta que este da. Recuerde que los servicios SOAP y REST al final son similares sin importar en que servidores web y programas se realicen.

APLICACION ANDROID

PASO1

Creación de la Aplicación

Ejecute el programa *Android Studio*. En la ventana que se despliega haga clic sobre el menú *File* y la Opción *New Project*.

En la nueva ventana seleccione la pestaña *Phone and Table*, elija la opción *Empty Activity* y haga clic en el botón *Next*.

En la siguiente ventana escriba como nombre del proyecto **SOAPapp** y como dominio escriba **ud.example.soapapp**, seleccione la mínima versión de android de ejecución para su nueva aplicación (se recomienda la API 23 por defecto), seleccione como lenguaje Java y haga clic sobre el botón *Finish*.

PASO 2:

Agregar Dependencias al Proyecto

Copie o descargue la librería ksoap2 adjunta en la práctica o bájela de la siguiente URL:

<https://code.google.com/p/ksoap2-android/source/browse/m2-repo/com/google/code/ksoap2-android/ksoap2-android-assembly/>

utilizada en la práctica:

<https://code.google.com/p/ksoap2-android/source/browse/m2-repo/com/google/code/ksoap2-android/ksoap2-android-assembly/2.6.5/ksoap2-android-assembly-2.6.5-jar-with-dependencies.jar>

Nota: La librería .jar debe ser copiada dentro del proyecto en la subcarpeta libs de la carpeta app, por medio de un explorador del sistema.

Ahora, tendrá que utilizar el Gradle para ayudar a que la librería pueda funcionar correctamente, esto se logra, agregando las dependencias a *Build.gradle Module: App* (*build.grade* (*Module: SOAPapp.app*)).

Copie el siguiente código:

```
dependencies {
```

```
implementation 'androidx.appcompat:appcompat:1.1.0'  
implementation 'com.google.android.material:material:1.1.0'  
implementation  
'androidx.constraintlayout:constraintlayout:1.1.3'  
testImplementation 'junit:junit:4.+'  
androidTestImplementation 'androidx.test.ext:junit:1.1.1'  
androidTestImplementation 'androidx.test.espresso:espresso-  
core:3.2.0'  
compile files('libs/ksoap2-android-assembly-2.6.5-jar-with-  
dependencies.jar')  
}
```

PASO 3. Adición de Código

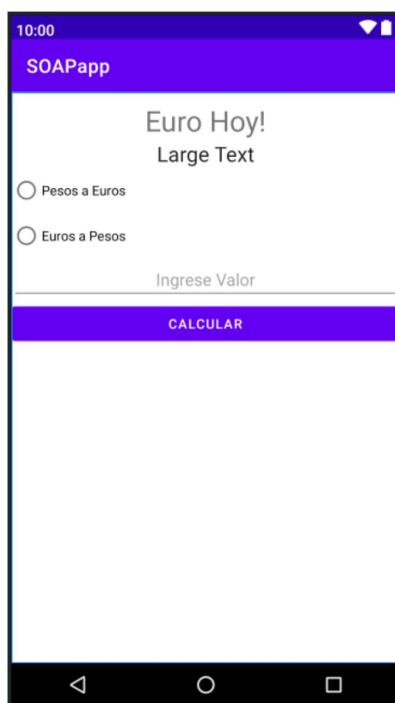
Modificación del archivo **AndroidManifest.xml**

Haga doble clic sobre el archivo AndroidManifest.xml del paquete manifests, y adicione el siguiente código al archivo:

```
<uses-permission android:name="android.permission.INTERNET" />
```

Modificación del archivo **activity_main.xml**

Nuestra actividad al final debería lucir de la siguiente manera:



Haga doble clic sobre el archivo activity_main.xml del paquete layout, y cambie el modo de visualización a texto, por medio de la pestaña Text. Modifique el código de la siguiente manera:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <TextView
            android:id="@+id/textView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="10dp"
            android:gravity="center"
            android:text="Euro Hoy!"
            android:textSize="30dp" />

        <TextView
            android:id="@+id/textView2"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:text="Large Text"
            android:gravity="center_horizontal" />

        <RadioGroup
            android:layout_width="match_parent"
            android:layout_height="wrap_content">

            <RadioButton
                android:id="@+id radioButton"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Pesos a Euros" />

            <RadioButton
                android:id="@+id radioButton2"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Euros a Pesos" />

        </RadioGroup>
```

```
<EditText
    android:id="@+id/editTextNumber"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:ems="10"
    android:inputType="number"
    android:gravity="center_horizontal"
    android:hint="Ingrese Valor"/>

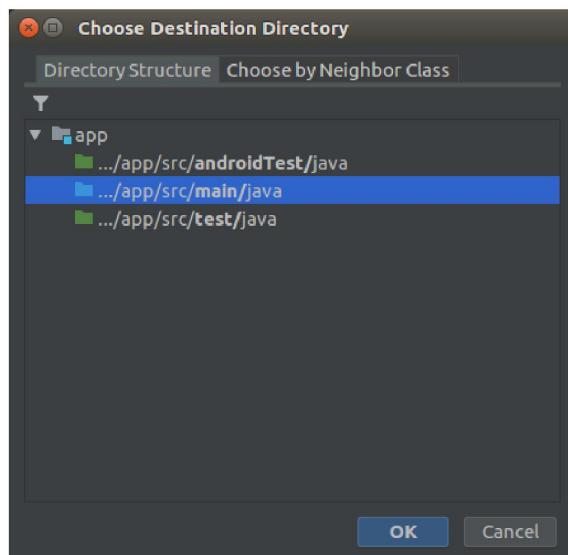
<Button
    android:id="@+id/button"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Calcular" />

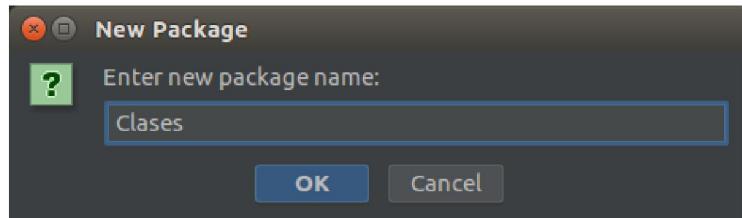
<TextView
    android:id="@+id/textView4"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="TextView"
    android:textSize="20dp"
    android:visibility="invisible" />

</LinearLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Creación de clases

Sobre el paquete *java*, haga clic derecho y agregue un nuevo paquete al proyecto, este se llamará **Clases**.





Agregue una nueva clase Java al proyecto, llamada WebService.

Código de la clase WebService

```
package Clases;

import org.ksoap2.SoapEnvelope;
import org.ksoap2.serialization.SoapObject;
import org.ksoap2.serialization.SoapPrimitive;
import org.ksoap2.serialization.SoapSerializationEnvelope;
import org.ksoap2.transport.HttpTransportSE;

public class WebService {
    //Namespace of the Webservice - can be found in WSDL
    private static String NAMESPACE = "http://com.example.ud/";
    //Webservice URL - WSDL File location
    private static String URL =
"http://192.168.0.4:8080/ServidorSOAP/MiServicioWeb?WSDL";
    //SOAP Action URI again Namespace + Web method name
    private static String SOAP_ACTION = "http://com.example.ud/";

    public static double CapturaEuroWS(String webMethName) {
        double resTxt = 0;
        // Create request
        SoapObject request = new SoapObject(NAMESPACE, webMethName);
        // Create envelope
        SoapSerializationEnvelope envelope = new
SoapSerializationEnvelope(SoapEnvelope.VER11);
        // Set output SOAP object
        envelope.setOutputSoapObject(request);
        //envelope.dotNet = true //Utilizar para servicios realizados en .Net
        // Create HTTP call object
        HttpTransportSE androidHttpTransport = new HttpTransportSE(URL);
        try {
            // Invoke web service
            androidHttpTransport.call(SOAP_ACTION+webMethName, envelope);
            // Get the response
            SoapPrimitive response = (SoapPrimitive) envelope.getResponse();
            // Assign it to resTxt variable static variable
            resTxt = Double.parseDouble(response.toString());
        } catch (Exception e) {
            //Print error
            e.printStackTrace();
        }
    }
}
```

```
//Return resTxt to calling object
    return resTxt;
}

public static double CambioEurows(String webMethName, String valord) {
    double resTxt = 0;
    // Create request
    SoapObject request = new SoapObject(NAMESPACE, webMethName);
    // Property which holds input parameters
    request.addProperty("valor", valord);
    // Create envelope
    SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(
        SoapEnvelope.VER11);
    // Set output SOAP object
    envelope.setOutputSoapObject(request);
    // Create HTTP call object
    HttpTransportSE androidHttpTransport = new HttpTransportSE(URL);
    try {
        // Invoke web service
        androidHttpTransport.call(SOAP_ACTION+webMethName, envelope);
        // Get the response
        SoapPrimitive response = (SoapPrimitive) envelope.getResponse();
        // Assign it to resTxt variable static variable
        resTxt = Double.parseDouble(response.toString());
    } catch (Exception e) {
        //Print error
        e.printStackTrace();
    }
    //Return resTxt to calling object
    return resTxt;
}

public static String HolaMundoWS(String name, String webMethName) {
    String resTxt = null;
    // Create request
    SoapObject request = new SoapObject(NAMESPACE, webMethName);
    // Property which holds input parameters
    //PropertyInfo sayHelloPI = new PropertyInfo();
    //sayHelloPI.setName("name");
    //sayHelloPI.setValue(name);
    //sayHelloPI.setType(String.class);
    //request.addProperty(sayHelloPI);
    // Add the property to request object
    request.addProperty("name",name);
    // Create envelope
    SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(
        SoapEnvelope.VER11);
    // Set output SOAP object
    envelope.setOutputSoapObject(request);
    // Create HTTP call object
    HttpTransportSE androidHttpTransport = new HttpTransportSE(URL);
    try {
        // Invoke web service
        androidHttpTransport.call(SOAP_ACTION+webMethName, envelope);
```

```
// Get the response
SoapPrimitive response = (SoapPrimitive) envelope.getResponse();
// Assign it to resTxt variable static variable
resTxt = response.toString();
} catch (Exception e) {
    //Print error
    e.printStackTrace();
    //Assign error message to resTxt
    resTxt = e.getMessage();
}
//Return resTxt to calling object
return resTxt;
}
```

Nota: La URL del servicio WSDL debe ser cambiada de acuerdo a la ruta de la maquina y del servicio utilizado

Modificación del archivo **MainActivity.java**

Haga doble clic sobre el archivo *MainActivity.java* del paquete *java*, y modifique el código de la siguiente manera:

Agregue los siguientes atributos privados a la clase

```
private Button miboton;
private TextView salida;
private TextView valoreuro;
private EditText entrada;
private RadioButton cambioP;
private String editText;
private String displayText;
private String displayEuro;
```

Agregue el siguiente método a la clase:

```
private class LLamadoInicialWS extends AsyncTask<String, Void, Void> {
    @Override
    protected Void doInBackground(String... strings) {
        double temp = WebService.CapturaEuroWS("getEuro");
        displayEuro = String.valueOf(temp);
        return null;
    }
    @Override
    protected void onPostExecute(Void result) {
        valoreuro.setText(displayEuro);
    }
    @Override
    protected void onPreExecute() {
        valoreuro.setText("Comenzó");
    }
}
```

```
private class CambioEuroWS extends AsyncTask<String, Void, Void> {
    @Override
    protected Void doInBackground(String... strings) {
        double temp;
        if(cambioP.isChecked()){
            temp = WebService.CambioEuroWS("Peso2Euro", editText);
        } else {
            temp = WebService.CambioEuroWS("Euro2Peso", editText);
        }
        displayText = String.valueOf(temp);
        return null;
    }
    @Override
    protected void onPostExecute(Void result) {
        salida.setText(displayText);
    }
    @Override
    protected void onPreExecute() {
        salida.setText("Comenzó");
    }
}
private class AsyncCallWS extends AsyncTask<String, Void, Void> {
    @Override
    protected Void doInBackground(String... params) {
        displayText = WebService.HolaMundoWS(editText, "hello");
        return null;
    }
    @Override
    protected void onPostExecute(Void result) {
        salida.setText(displayText);
    }
    @Override
    protected void onPreExecute() {
    }
    @Override
    protected void onProgressUpdate(Void... values) {
    }
}
```

En el método OnCreate de la clase agregue el siguiente código:

```
cambioP = findViewById(R.id.radioButton);
cambioP.setChecked(true);
entrada = findViewById(R.id.editTextNumber);
valoreuro = findViewById(R.id.textView2);
salida = findViewById(R.id.textView4);
miboton = findViewById(R.id.button);
miboton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (entrada.getText().length() != 0 && entrada.getText().toString() != "") {
```

```
//Get the text control value
editText = entrada.getText().toString();
//Create instance for AsyncCallWS
//AsyncCallWS task = new AsyncCallWS();
CambioEuroWS task = new CambioEuroWS();
//Call execute
task.execute();
salida.setVisibility(View.VISIBLE);
}
}
});

LLamadoInicialWS capini = new LLamadoInicialWS();
capini.execute();
```

Por último, **Ejecute y Evalúe la aplicación**

TRABAJO ADICIONAL

1. Adicionar el código necesario para arregla el resultado de la transformación de la moneda (solo mostrar dos decimales).
2. Agregar la palabra Euros o Pesos en la transformación de la moneda.