Desarrollo de Servicios Web 1

Tema Nº3 y N°4:ARQUITECTURA SOAP

Indicador de logro Nº3 y N°4:Aplica la arquitectura SOAP a través del Web Service ClienteWS en Database con pruebas unitarias SoapUI/Postman.

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº3 y N°4:**

ARQUITECTURA SOAP

**Subtema 1:**

GESTIÓN DE BASES DE DATOS EN APLICACIONES WEB CON JAVA

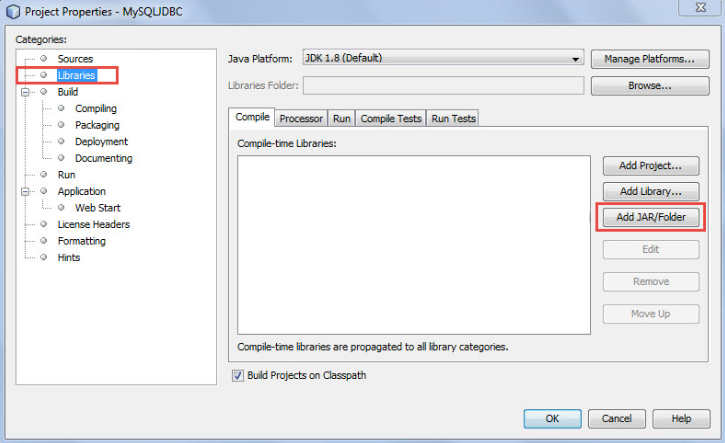
**MARCO TEÓRICO**

La gestión de bases de datos es un procedimiento importante en una aplicación web. Permite mantener y consultar información relevante para empresarios, clientes y usuarios en general.

Resulta entonces de suma importancia conocer cómo realizar una gestión adecuada de datos, principalmente en un ambiente web donde el acceso y la seguridad son aspectos muy importantes a considerar.

Para realizar una conexión a una base de datos mediante JSP o un Servlet, se utiliza un driver. Este driver es suministrado generalmente por la proveedora del DBMS (Data Base Management System). Este archivo constituye un JAR que contiene una serie de clases que implementan la funcionalidad necesaria para la gestión de datos.

Asimismo, se sabe la arquitectura de gestión de datos con el JDBC, los objetos que la componen considerando MySQL, entre otros factores.



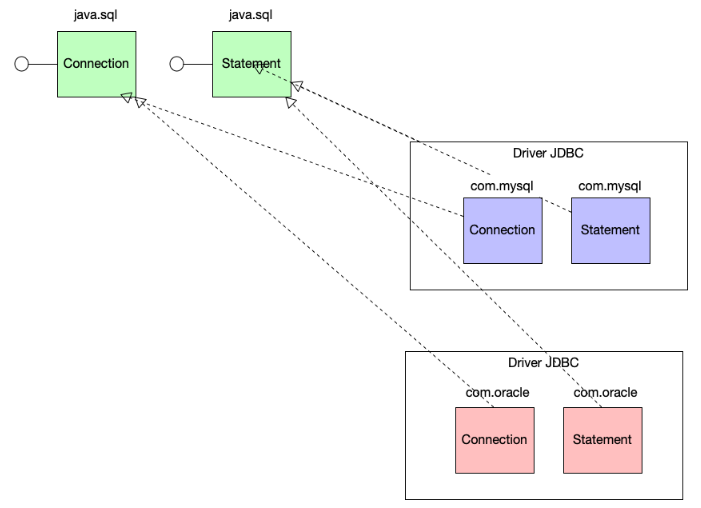


**JDBC**

Es el API para la ejecución de sentencias SQL. (Como punto de interés JDBC es una marca registrada y no un acrónimo, no obstante, a menudo es conocido como “Java Database Connectivity”). Consiste en un conjunto de clases e interfases escritas en el lenguaje de programación Java. JDBC suministra un API estándar para los desarrolladores y hace posible escribir aplicaciones de base de datos usando un API puro Java.

Usando JDBC es fácil enviar sentencias SQL virtualmente a cualquier sistema de base de datos. En otras palabras, con el API JDBC, no es necesario escribir un programa que acceda a una base de datos Sybase, otro para acceder a Oracle y otro para acceder a Informix. Un único programa escrito usando el API JDBC y el programa será capaz de enviar sentencias SQL a la base de datos apropiada. Y, con una aplicación escrita en el lenguaje de programación Java, tampoco es necesario escribir diferentes aplicaciones para ejecutar en diferentes plataformas.

La combinación de Java y JDBC permite al programador escribir una sola vez y ejecutarlo en cualquier entorno.



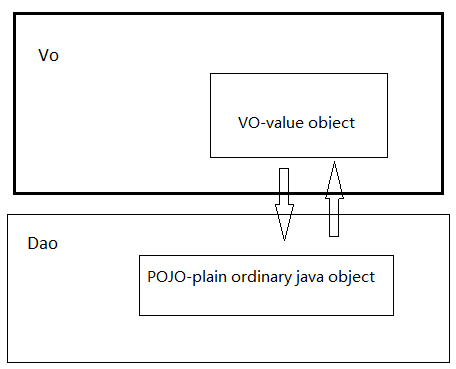
**Value Object (VO)**

La capa de negocio necesita intercambiar datos con la de presentación. Estos datos se modelan con un Value Object, que no será más que un bean de Java (no un EJB) que encapsula la información. La capa de presentación siempre utiliza estas clases, independientemente de que "por debajo" hayan o no EJBs. Aun en el caso de que la capa de negocio utilice EJBs, acceder directamente a ellos desde la capa de presentación es muy costoso si se accede individualmente a los campos del objeto.

**Data Access Object (DAO)**

Prácticamente todas las aplicaciones de hoy en día, requiere acceso al menos a una fuente de datos, dichas fuentes son por lo general base de datos relacionales, por lo que muchas veces no tenemos problema en acceder a los datos, sin embargo, hay ocasiones en las que necesitamos tener más de una fuente de datos o la fuente de datos que tenemos puede variar, lo que nos obligaría a refactorizar gran parte del código.

Para esto, tenemos el patrón Arquitectónico Data Access Object (DAO), el cual permite separar la lógica de acceso a datos de los Value Object (VO), de tal forma que el DAO encapsula toda la lógica de acceso de datos al resto de la aplicación.



**1. MATERIALES**

Para la experiencia a realizar se requiere lo siguiente:

1. EQUIPO

Se requiere tener un equipo de cómputo (PC o laptop).

1. HERRAMIENTAS

* Java Developer Kit (JDK) version 8.x
* NetBeans 8.2
* Apache Tomcat 9.0.12
* MySQL 5.x.x
* Mysql-connection.java
* Jax-ws/metro
* SoapUI 5.5

**2. PROCEDIMIENTO**

Aplicación Web JDBC con Web Service. Query de la BD con data:

use bd\_example;

create table categorias (

id\_categoria int primary key auto\_increment,

descripcion varchar(100)

);

create table productos (

id\_producto int primary key auto\_increment,

nombre varchar(100),

marca varchar(100),

precio double,

stock int,

id\_categoria int,

foreign key(id\_categoria) references categorias(id\_categoria)

);

insert into categorias(descripcion) values

('Televisores'), ('Computadoras'), ('Impresoras'), ('Cine en casa'), ('Radio');

insert into productos(nombre,marca,precio,stock,id\_categoria) values

('Tv LCD 19 Plg','Modelo Sony PFL3403',2199.0,40,1),

('Tv LCD 22 Plg','Modelo Lg LN22A450',2399.0,30,1),

('Tv LCD 32 Plg','Modelo Samsung LC813H',2999.0,25,1),

('Tv LCD 26 Plg','Modelo Panasonic 26LG30R',2999.0,25,1),

('CPU LE125','Lg',2199.0,15,2),

('CPU A64X2','Compaq',2499.0,8,2),

('CPU E5200','Hp',2699.0,5,2),

('CPU SG330','Dell',3099.0,5,2),

('Impresora T23','Xerox',219.0,18,3),

('Impresora DJ-6940','Dell',349.0,20,3),

('Impresora T33','Hp',399.0,12,3),

('Impresora K5400','Epson',499.0,8,3),

('Home cinema SPH70','Lg',319.0,4,4),

('Home cinema AMX115','Sony',419.0,4,4),

('Home cinema HT E 860','Samsung',599.0,2,4),

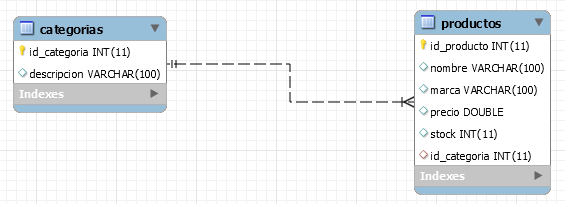
('Home cinema DC-T990','Panasonic',759.0,5,4),

('Home cinema HT-1105U','Sony',829.0,14,4),

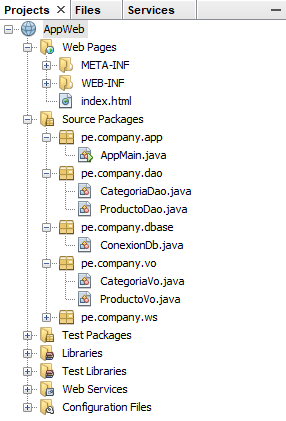
('Radio ICF-S10MK2/SCE','Panasonic',79.0,15,5),

('Radio ICF-18','Lg',99.0,15,5),

('Radio ICF-303/304 AM-FM','Sony',129.0,8,5);



Proyecto Web:



Clases VO:

package pe.company.vo;

import java.util.ArrayList;

import java.util.Collection;

public class CategoriaVo

{

private int id\_categoria;

private String descripcion;

private Collection<ProductoVo> itemsProducto=new ArrayList();

public CategoriaVo() {

}

public CategoriaVo(int id\_categoria, String descripcion) {

this.id\_categoria = id\_categoria;

this.descripcion = descripcion;

}

public void addProducto(ProductoVo producto) {

itemsProducto.add(producto);

}

//Getters and Setters

}

package pe.company.vo;

public class ProductoVo

{

private int id\_producto;

private String nombre;

private String marca;

private double precio;

private int stock;

private CategoriaVo categoria;

public ProductoVo() {}

public ProductoVo(int id\_producto, String nombre, String marca,

double precio, int stock, CategoriaVo categoria) {

this.id\_producto = id\_producto;

this.nombre = nombre;

this.marca = marca;

this.precio = precio;

this.stock = stock;

this.categoria = categoria;

}

@Override

public String toString() {

return "ProductoVo{" + "id\_producto=" + id\_producto + ", nombre=" + nombre +

", marca=" + marca + ", precio=" + precio +

", stock=" + stock + ", categoria=" + categoria.getDescripcion() + '}';

}

//Getters and Setters

}

Clases DAO:

package pe.company.dao;

import java.sql.\*;

import java.util.\*;

import pe.company.dbase.ConexionDb;

import pe.company.vo.CategoriaVo;

public class CategoriaDao

{

private Connection conn;

private PreparedStatement ps;

private ResultSet rs;

public CategoriaDao() {}

public CategoriaVo findById(int id\_categoria) {

CategoriaVo categoria=null;

try {

conn=ConexionDb.MySQL();

ps=conn.prepareStatement("select \* from categorias where id\_categoria=?");

ps.setInt(1,id\_categoria);

rs=ps.executeQuery();

if(rs.next()) {

categoria=new CategoriaVo();

categoria.setId\_categoria(rs.getInt("id\_categoria"));

categoria.setDescripcion(rs.getString("descripcion"));

}

}

catch(Exception ex){

ex.printStackTrace();

}

return categoria;

}

public Collection<CategoriaVo> findAll() {

List<CategoriaVo> list=new ArrayList<>();

try {

conn=ConexionDb.MySQL();

ps=conn.prepareStatement("select \* from categorias");

rs=ps.executeQuery();

while(rs.next()) {

CategoriaVo categoria=new CategoriaVo();

categoria.setId\_categoria(rs.getInt("id\_categoria"));

categoria.setDescripcion(rs.getString("descripcion"));

list.add(categoria);

}

}

catch(Exception ex){

ex.printStackTrace();

}

return list;

}

}

package pe.company.dao;

import java.sql.\*;

import java.util.\*;

import pe.company.dbase.ConexionDb;

import pe.company.vo.CategoriaVo;

import pe.company.vo.ProductoVo;

public class ProductoDao

{

private Connection conn;

private PreparedStatement ps;

private ResultSet rs;

private CategoriaDao categoriaDao=new CategoriaDao();

public ProductoDao() {}

public Collection<ProductoVo> findAll()

{

List<ProductoVo> list=new ArrayList<>();

try {

conn=ConexionDb.MySQL();

ps=conn.prepareStatement("select \* from productos");

rs=ps.executeQuery();

while(rs.next()) {

ProductoVo producto=new ProductoVo();

producto.setId\_producto(rs.getInt("id\_producto"));

producto.setNombre(rs.getString("nombre"));

producto.setMarca(rs.getString("marca"));

producto.setPrecio(rs.getDouble("precio"));

producto.setStock(rs.getInt("stock"));

CategoriaVo categoria=categoriaDao.findById(rs.getInt("id\_categoria"));

producto.setCategoria(categoria);

list.add(producto);

}

}

catch(Exception ex){

ex.printStackTrace();

}

return list;

}

}

Clase Conexión:

package pe.company.dbase;

import java.sql.\*;

public class ConexionDb

{

public static Connection MySQL() throws ClassNotFoundException, SQLException {

Class.*forName*("com.mysql.jdbc.Driver");

Connection c=DriverManager.*getConnection*

("jdbc:mysql://localhost:3306/bd\_example","root","mysql");

return c;

}

}

Creación del Servicio Web:



package pe.company.ws;

import java.util.Collection;

import javax.jws.\*;

import pe.company.dao.ProductoDao;

import pe.company.vo.ProductoVo;

@WebService(serviceName = "ProductoWS")

public class ProductoWS

{

//se instancia dao del producto

private ProductoDao productoDao=new ProductoDao();

@WebMethod(operationName="listarProductos")

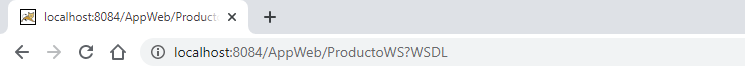
public Collection<ProductoVo> listarProducto(@WebParam(name="codigo") String codigo) {

return productoDao.findAll();

}

}

Ejecución del Proyecto Web, y se obtiene el WSDL:



<definitions

xmlns:wsu=*"http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"*

xmlns:wsp=*"http://www.w3.org/ns/ws-policy"*

xmlns:wsp1\_2=*"http://schemas.xmlsoap.org/ws/2004/09/policy"*

xmlns:wsam=*"http://www.w3.org/2007/05/addressing/metadata"*

xmlns:soap=*"http://schemas.xmlsoap.org/wsdl/soap/"*

xmlns:tns=*"http://ws.company.pe/"*

xmlns:xsd=*"http://www.w3.org/2001/XMLSchema"*

xmlns=*"http://schemas.xmlsoap.org/wsdl/"*

targetNamespace=*"http://ws.company.pe/"* name=*"ProductoWS"*>

<types>

<xsd:schema>

<xsd:import namespace=*"http://ws.company.pe/"*

schemaLocation=*"http://localhost:8084/AppWeb/ProductoWS?xsd=1"* />

</xsd:schema>

</types>

<message name=*"listarProductos"*>

<part name=*"parameters"* element=*"tns:listarProductos"* />

</message>

<message name=*"listarProductosResponse"*>

<part name=*"parameters"* element=*"tns:listarProductosResponse"* />

</message>

<portType name=*"ProductoWS"*>

<operation name=*"listarProductos"*>

<input

wsam:Action=*"http://ws.company.pe/ProductoWS/listarProductosRequest"*

message=*"tns:listarProductos"* />

<output

wsam:Action=*"http://ws.company.pe/ProductoWS/listarProductosResponse"*

message=*"tns:listarProductosResponse"* />

</operation>

</portType>

<binding name=*"ProductoWSPortBinding"* type=*"tns:ProductoWS"*>

<soap:binding

transport=*"http://schemas.xmlsoap.org/soap/http"* style=*"document"* />

<operation name=*"listarProductos"*>

<soap:operation soapAction=*""* />

<input> <soap:body use=*"literal"* /> </input>

<output> <soap:body use=*"literal"* /> </output>

</operation>

</binding>

<service name=*"ProductoWS"*>

<port name=*"ProductoWSPort"* binding=*"tns:ProductoWSPortBinding"*>

<soap:address

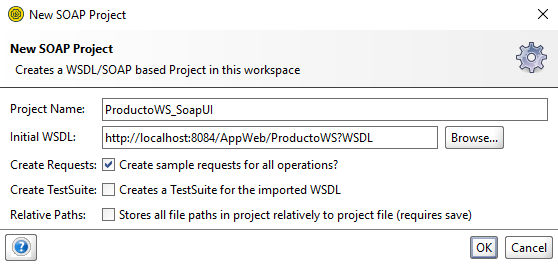
location=*"http://localhost:8084/AppWeb/ProductoWS"* />

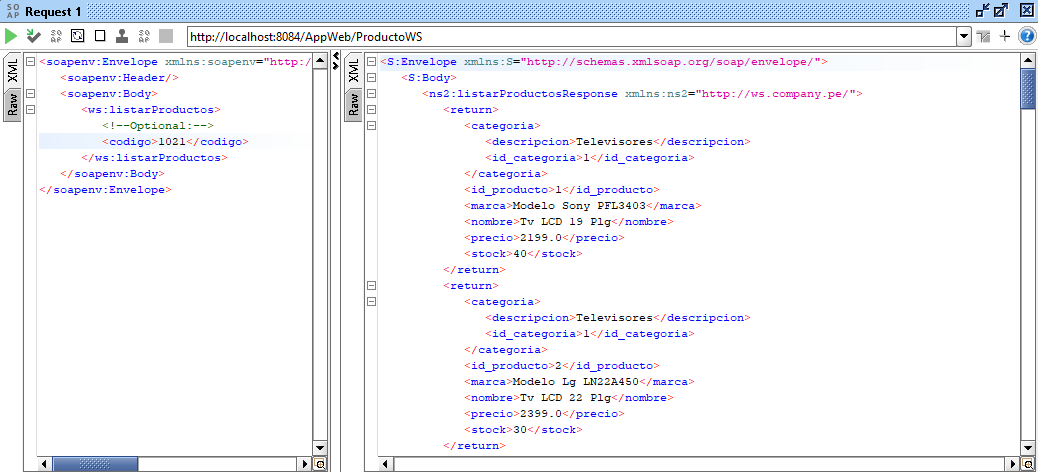
</port>

</service>

</definitions>

Prueba unitaria con SoapUI:





**ACTIVIDAD VIRTUAL:**

1. **CUESTIONARIO TÉCNICO**

Revisa y analiza el tema desarrollado en la presente sesión, luego responde las siguientes preguntas propuestas:

* ¿Qué es JDBC y cuál es su importancia en la actualidad?
* ¿Qué patrones de diseño conoce aparte del VO y DAO?
* ¿JDBC o JPA Hibernate cuáles son sus ventajas y desventajas para los Web Service?
* ¿Qué se entiende cuando se nos pide “consumir el Web Service”?
* Realice un Web Service mediante un código y mostrar sus datos personales.

1. **ENLACES DE REFERENCIA**

* http://profesores.fi-b.unam.mx/sun/Downloads/Java/jdbc.pdf
* https://www.arquitecturajava.com/jdbc-driver-un-concepto-clave/
* http://www.jtech.ua.es/j2ee/2002-2003/modulos/aplic-j2ee/apuntes/apuntes3.htm
* https://reactiveprogramming.io/blog/es/patrones-arquitectonicos/dao

1. **CONCLUSIONES DE LA EXPERIENCIA**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_