Desarrollo de Servicios Web 1

Tema Nº7 y N°8:Documentos, despliegue REST y REST con JSON con JEE

Indicador de logro Nº7 y N°8:Aplica documentos y despliegue REST a través del WADL y XSD en Web Services SOAP además de REST con JSON en JEE.

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº7 y N°8:**

DOCUMENTOS, DESPLIEGUE REST y REST con JSON con JEE

**Subtema 1:**

DESPLIEGUE REST

**MARCO TEÓRICO**

Las aplicaciones web tienen las siguientes características:

* Se ejecutan en un servidor (físico o virtual), no en el dispositivo del usuario
* Pueden atender a miles de usuarios simultáneos, no a uno sólo
* Es muy habitual que se necesiten varios servidores para una única aplicación web (por escalabilidad y tolerancia a fallos)
* Están formadas por código y por recursos (imágenes, documentos html, css, js, ficheros estáticos…)
* Utilizan servicios adicionales: base de datos, servidor de correo, servidor de vídeo, etc.
* Requieren de un proceso de instalación y configuración (despliegue) en el servidor o servidores.

Para desplegar una aplicación web de JAX-RS, necesita un paquete WAR o un paquete EAR que esté configurado y habilitado para los servicios RESTful.

Toda aplicación web debe tener una raíz de contexto para que la aplicación web se despliegue correctamente. Una raíz de contexto para cada módulo web se define en el descriptor de despliegue de aplicaciones durante el ensamblaje de aplicaciones o durante el despliegue de aplicaciones.

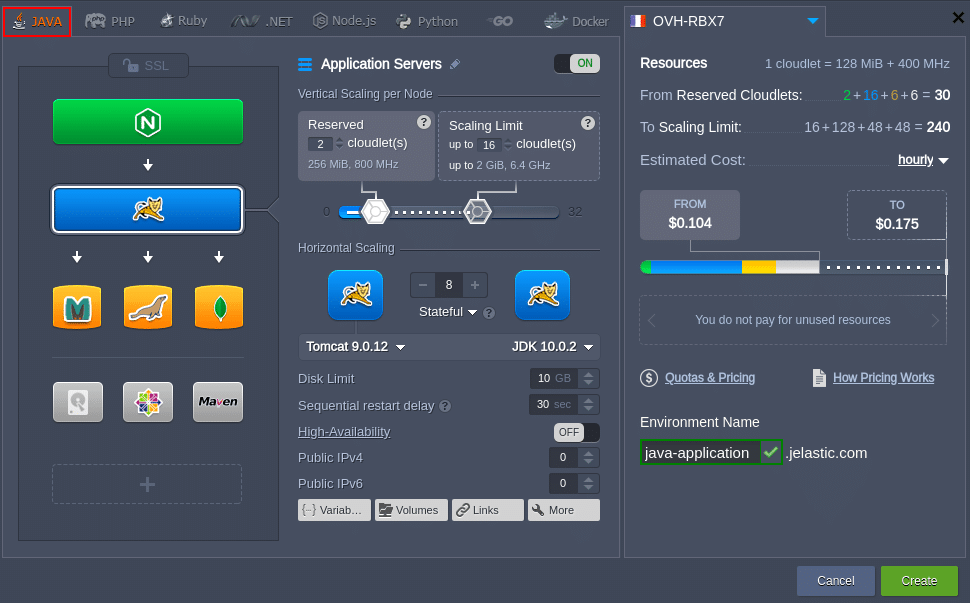
El directorio raíz de contexto se combina con la correlación de servlet definida del archivo WAR para componer el URL completo que los usuarios escriben para acceder al servlet. La raíz de contexto de cada aplicación web desplegada debe ser exclusiva en el servidor. La raíz de contexto también puede estar vacía.

Por ejemplo, si una aplicación web ha utilizado una raíz de contexto de sample/application/, el URL de solicitud de la aplicación web comenzará con http://<nombre\_host>:<puerto>/sample/application/.

El patrón URL de un servlet se añade a la raíz de contexto de la aplicación web. Por ejemplo, si la raíz de contexto es sample/application/ y la correlación URL de servlet es rest/api/\*, el URI base para la aplicación web de JAX-RS será http://<nombre\_host>:<puerto>/sample/application/rest/api.

Existen muchos proveedores con diferentes servicios, algunos de ellos como:





**Subtema 2:**

REST CON JSON y CONTENEDOR JEE DATABASE

**MARCO TEÓRICO**

La clase que va a ser nuestro servicio Web no es más que una clase normal de java, con métodos. Únicamente hay que poner las anotaciones específicas de JAX-RS para que el contenedor de aplicaciones lo convierta en servicio web cuando arranque. Las anotaciones son las siguientes:

* **@Path** para indicar el path en la url donde estará nuestro servicio web.
* **@Produces** para indicar el formato en el que el servicio web nos devuelve el resultado. Son habituales "application/xml" y "application/json".
* **@Consumes** para indicar el formato en el que nuestro servicio web admite los datos. Igual que @Produces, son habituales "application/xml" y "application/json".

El protocolo http define [varios métodos](https://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html) para las peticiones http. En los web services son habituales las cuatro siguientes:

* **@GET** Cuando queremos pedir datos al servicio web.
* **@POST** Cuando enviamos datos al servicio web para que el servicio web haga con ellos lo que considere oportuno.
* **@PUT** Cuando queremos guardar datos en una url específica. Si esos datos no existen en esa URL, se crean. Si ya existen, deben modificarse para que sean lo que nosotros hemos pasado.
* **@DELETE** Para borrar los datos de una url específica.

**Advanced REST Client**

Advanced REST Client es una herramienta que puede encontrarse como plugin para el navegador Google Chrome y, que como su propio nombre indica hace las veces de un cliente REST. Además, ayuda a cualquier desarrollador web a crear y testar peticiones HTTP de prueba.

La iniciativa de esta herramienta nace con la idea de poder ir comprobando gran parte de la implementación del servicio web, sin necesidad de tener programada la parte de la aplicación cliente como por ejemplo Android, que tiene que encargarse del consumo del servicio web.

Presenta un interfaz muy simple, incluso para usuarios que no hayan usado ningún cliente REST anteriormente.

Además, tiene un uso básico, introduciendo, por un lado y por el otro:

* El método HTTP a utilizar: GET, POST, PUT, DELETE
* La dirección URL de nuestro servicio, o URI

**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-connector.java
* Jersey 2.5 (JAX-RS RI)
* Advanced REST Client

**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),

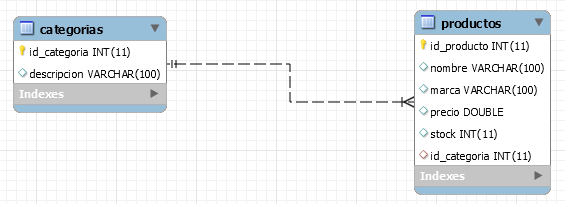
('Home cinema DC-T990','Panasonic',759.0,5,4),

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

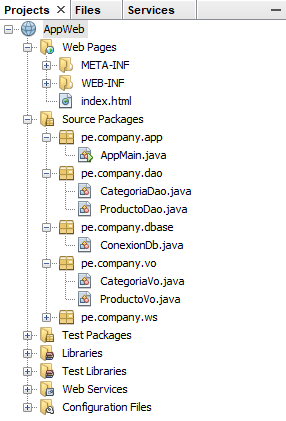
('Radio ICF-S10MK2/SCE','Panasonic',79.0,15,5),

('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 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;

}

}

package pe.company.dao;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.ArrayList;

import java.util.Collection;

import java.util.List;

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 void insert(ProductoVo producto)

{

try

{

conn=ConexionDb.MySQL();

ps=conn.prepareStatement("insert into

productos(nombre,marca,precio,stock,id\_categoria) values(?,?,?,?,?)");

ps.setString(1,producto.getNombre());

ps.setString(2,producto.getMarca());

ps.setDouble(3,producto.getPrecio());

ps.setInt(4,producto.getStock());

ps.setInt(5,producto.getCategoria().getId\_categoria());

int rows=ps.executeUpdate();

if(rows!=1)

throw new Exception("Error!");

}

catch(Exception ex){

ex.printStackTrace();

}

}

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*

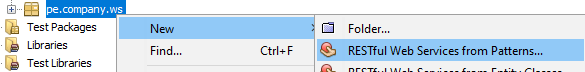
("jdbc:mysql://localhost:3306/bd\_example","root","mysql");

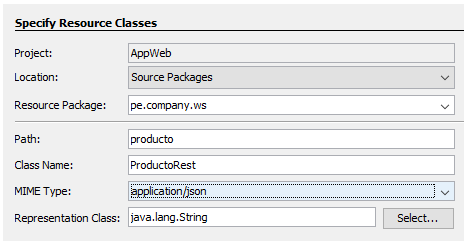
return c;

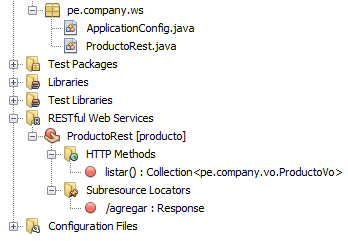
}

}

Creación de Servicios Web:







package pe.company.ws;

import java.util.Collection;

import javax.ws.rs.core.Context;

import javax.ws.rs.core.UriInfo;

import javax.ws.rs.Produces;

import javax.ws.rs.Consumes;

import javax.ws.rs.GET;

import javax.ws.rs.PUT;

import javax.ws.rs.Path;

import javax.ws.rs.core.MediaType;

import javax.ws.rs.core.Response;

import pe.company.dao.ProductoDao;

import pe.company.vo.ProductoVo;

@Path("producto")

public class ProductoRest

{

@Context

private UriInfo context;

//instancia al dao

private ProductoDao productoDao=new ProductoDao();

public ProductoRest() {}

@Path("/agregar")

@PUT

@Consumes(MediaType.APPLICATION\_JSON)

@Produces(MediaType.APPLICATION\_JSON)

public Response agregar(ProductoVo producto)

{

productoDao.insert(producto);

return Response.ok().entity(producto).build();

}

@Path("/listar")

@GET

@Produces(MediaType.APPLICATION\_JSON)

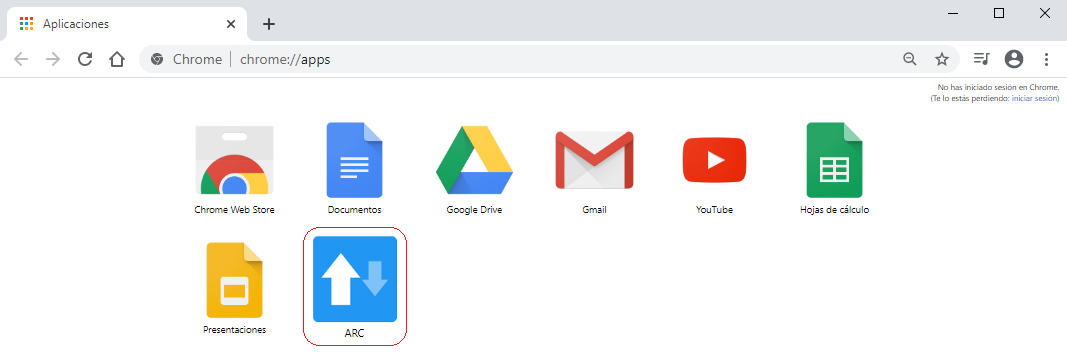
public Collection<ProductoVo> listar() {

return productoDao.findAll();

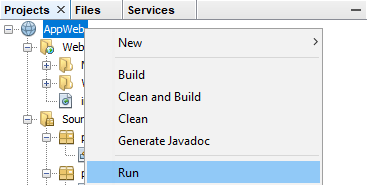
}

}

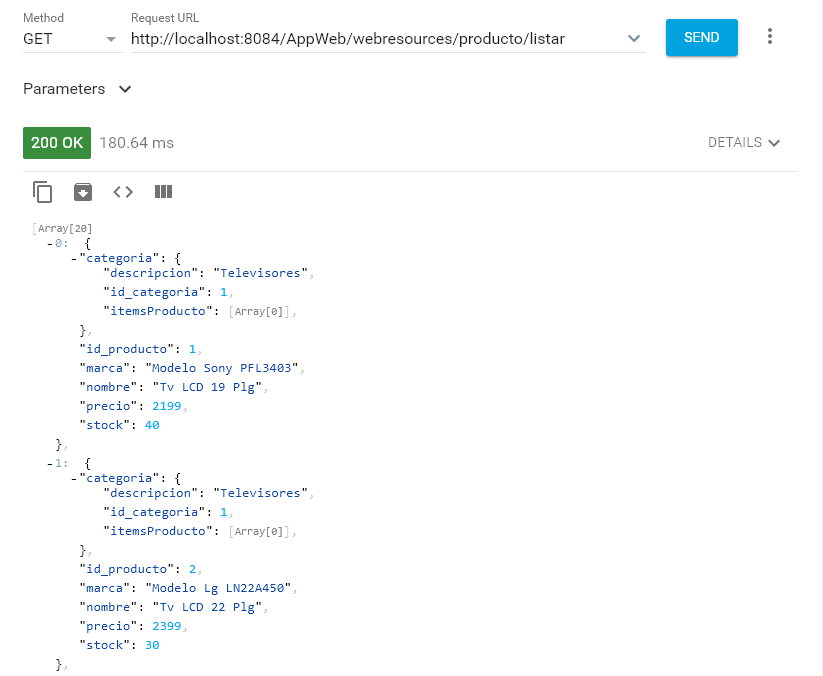
Se utilizará como prueba unitaria el programa Advanced REST Client:



Ejecutamos el proyecto web:

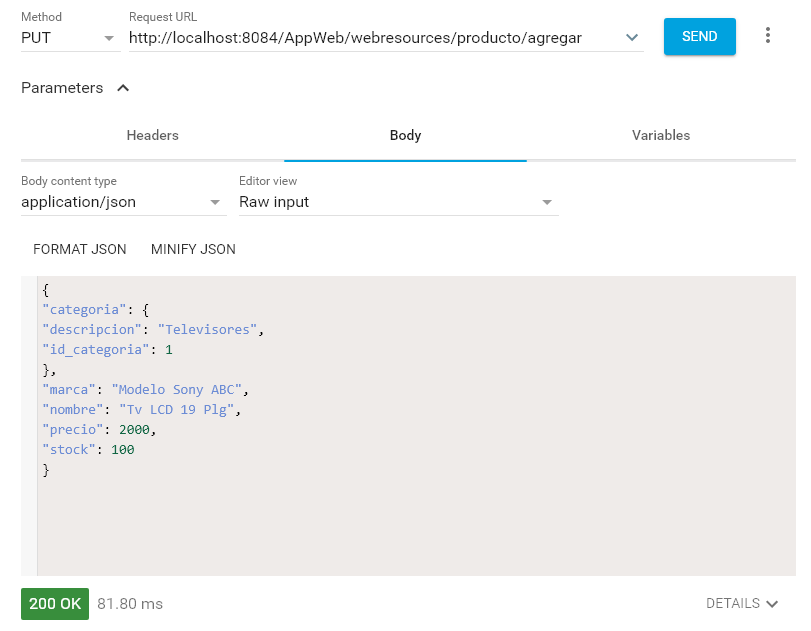


Haciendo la prueba unitaria listar productos:



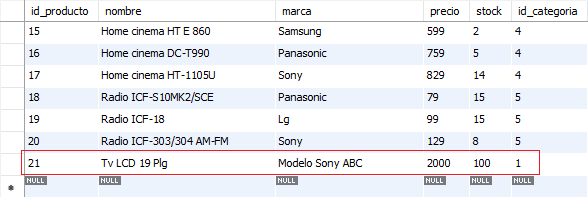
Se observa el listado de productos en JSON

Haciendo la prueba unitaria agregar producto:



Se acaba de insertar un nuevo producto por JSON a la base de datos

En la base de datos se observa el nuevo producto agregado por JSON:



**ACTIVIDAD VIRTUAL:**

1. **CUESTIONARIO TÉCNICO**

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

* ¿Qué es Jelastic?
* ¿Qué se necesita para desplegar una aplicación web JAX-RS?
* ¿Las anotaciones @Consumes y @Produces en qué se diferencian?
* ¿A parte del testeador Advanced REST Client qué otros se pueden usar?
* Mediante NetBeans, Base de datos, Tomcat y Jersey crear un servicio REST para el resto de operaciones de esta guía. Probar accesos de clientes.

1. **ENLACES DE REFERENCIA**

* https://www.ibm.com/docs/es/was/9.0.5?topic=services-deploying-jax-rs-11-web-applications
* https://www.infomaniak.com/es/asistencia/faq/2260/que-es-jelastic
* https://mejorsoftware.info/tools/advanced-rest-client

1. **CONCLUSIONES DE LA EXPERIENCIA**

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

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