PRESENTACIÓN

WEB SERVICES

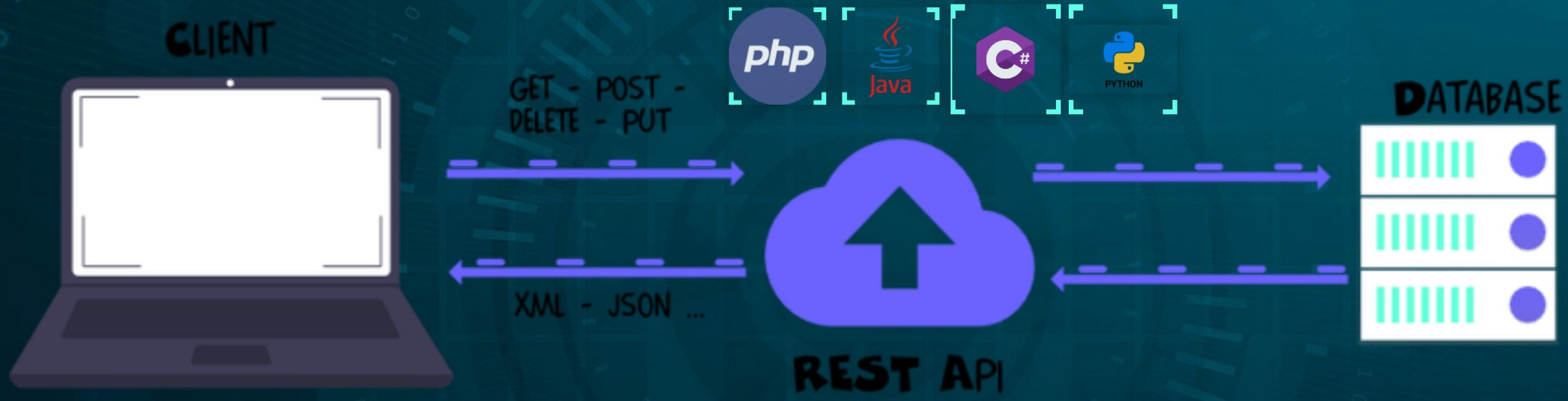
EMPEZAR

ÍNDICE

DEFINICIÓN

Es una tecnología multiplataforma y distribuida que permite la comunicación y compartir datos o información entre dos aplicaciones en la internet, utilizando un conjunto de protocolos y estandares de programación.

¿CÓMO FUNCIONAN?



WebApp or Desktop or Mobile

Aplicación con la que interactua el usuario.

Web Server

Servidor local o remoto donde se alojan los webservice

Base de datos

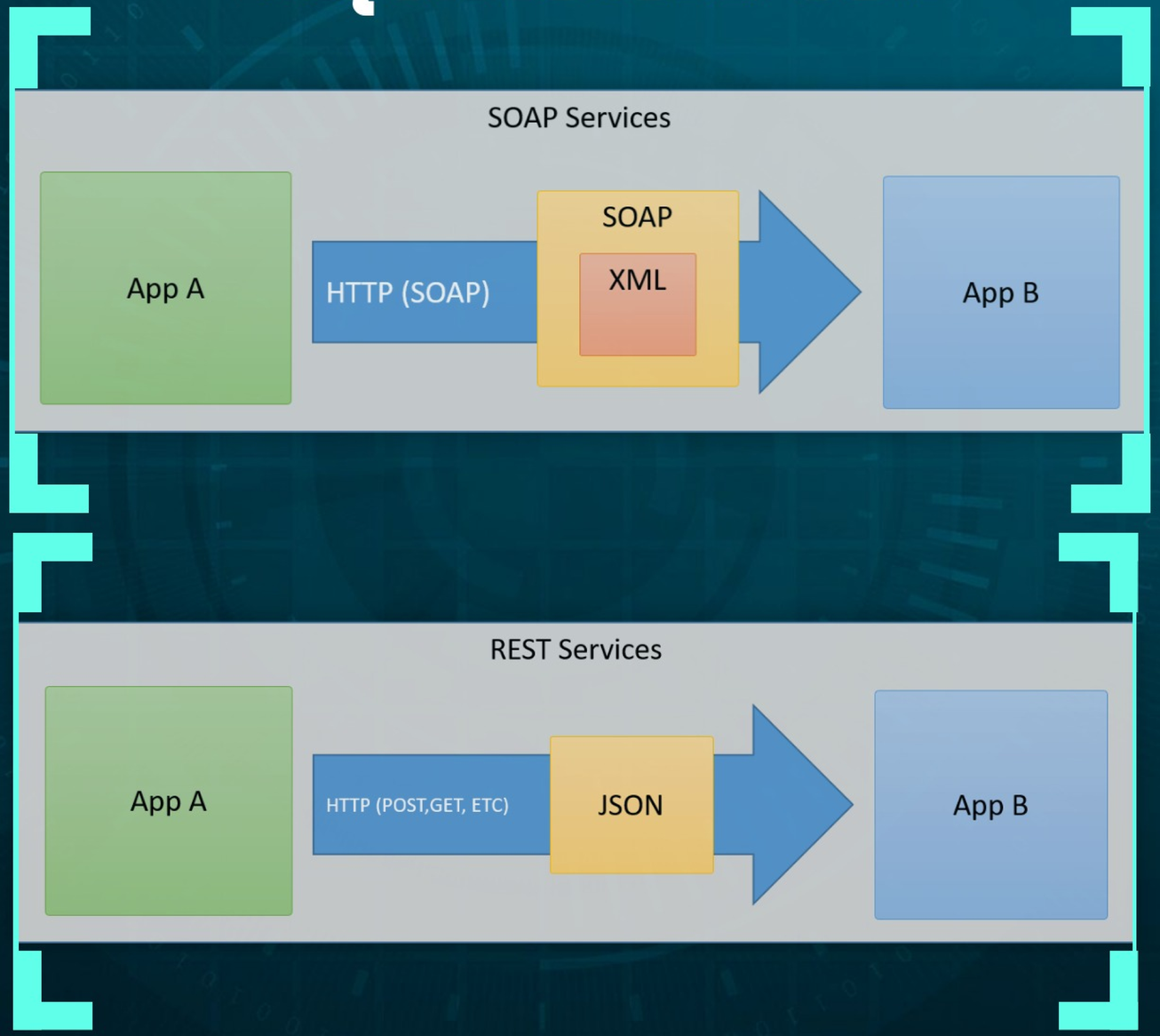
Base de datos local o remota

PROTOCOLOS

- Soporta XML.
- HTTP, SMTP o JMS.
- Cliente/Servidor en distintos lenguajes de programación.
- Ampliamente Estandarizado.

- Soporta XML y JSON
- HTTP, POST, GET, PUT, DELETE
- Cliente/Servidor en distintos lenguajes de programación
- Buen rendimiento y performance

ARQUITECTURA SOA



JSON

Es un formato ligero de intercambio de datos.

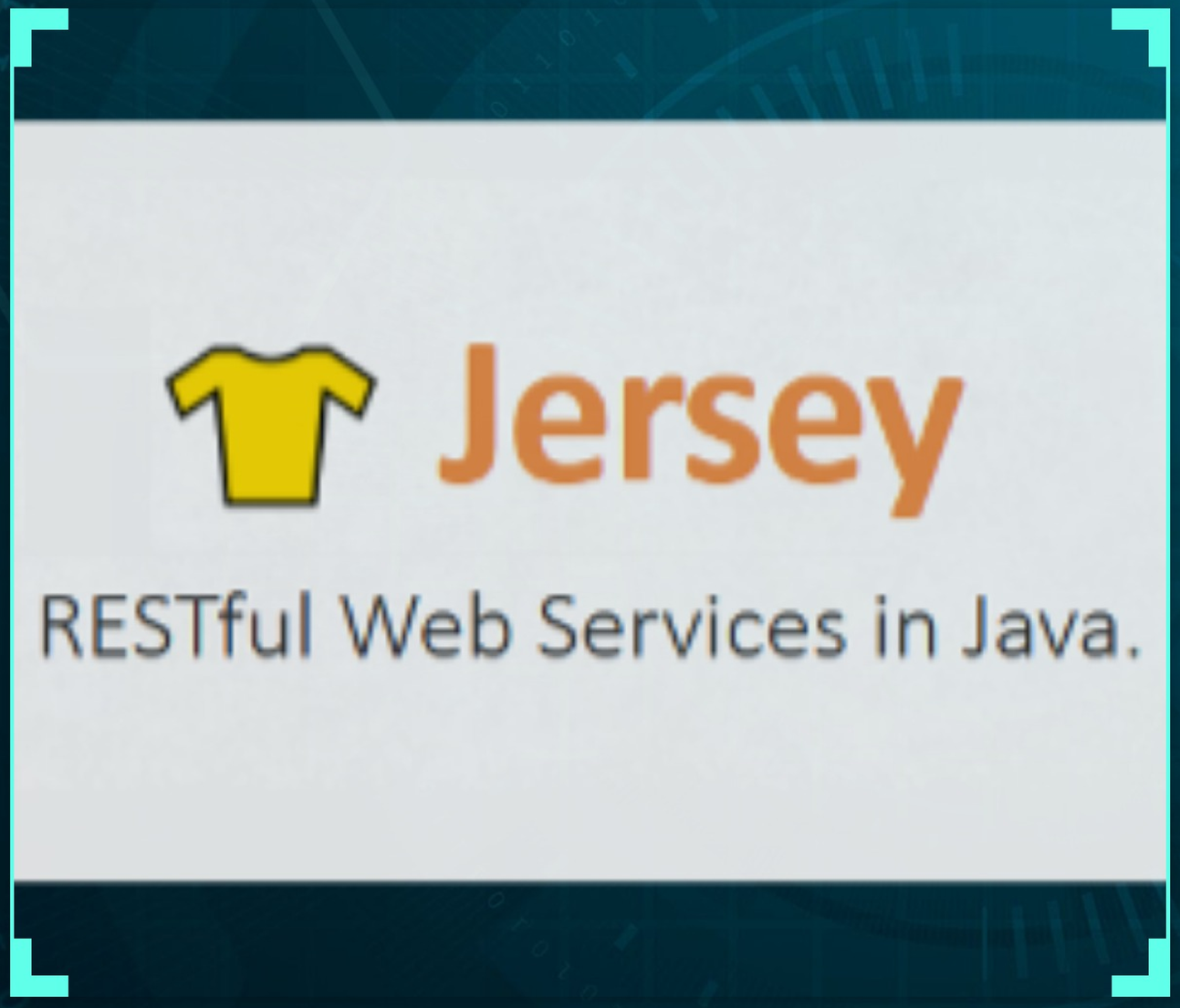
Está formado por dos estructuras:

- Una colección de pares de nombre/valor (Objeto).
- Una lista ordenada de valores (Arreglos).

Model | Example Value


```
[
  {
    "Id": 0,
    "FirstName": "string",
    "LastName": "string",
    "Name": "string",
    "EmailAddress": "string",
    "TerritoryId": 0
  }
]
```


Response Content Type application/json

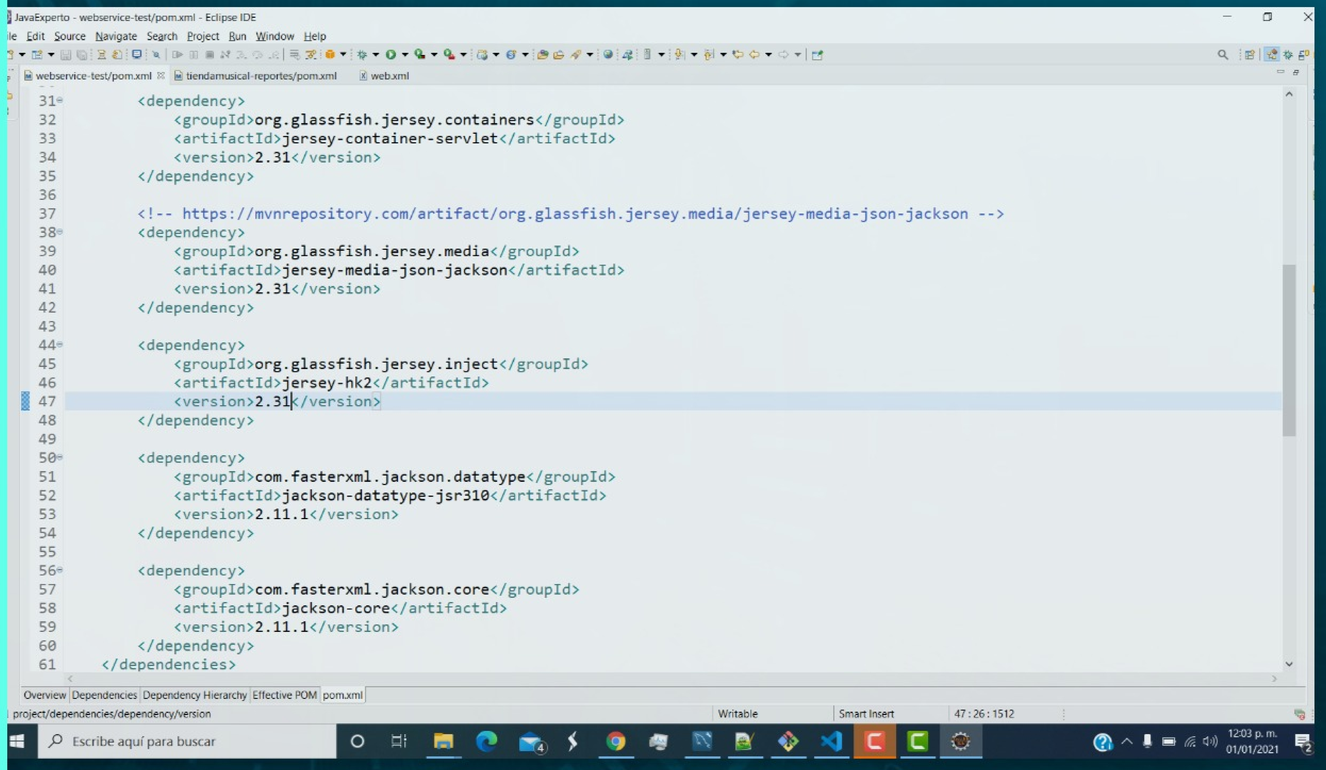


JERSEY

Es un cliente RESTful que Facilita a los desarrolladores crear servicios web RESTful con Java.

CONFIGURACIÓN

Paso 1: Integrar la dependencia



CONFIGURACIÓN

Paso 2:

Configurar el mappeo en web.xml

JavaExperto - webservice-test/src/main/webapp/WEB-INF/web.xml - Eclipse IDE

File Edit Source Navigate Search Project Run Window Help

webservice-test/pom.xml | tiendamusical-reportes/pom.xml | web.xml

```xml
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
         http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
         version="3.1">
    <display-name>Archetype Created Web Application</display-name>

    <servlet>
        <servlet-name>jersey-servlet</servlet-name>
        <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>
        <init-param>
            <param-name>jersey.config.server.provider.packages</param-name>
            <param-value>com.devpredator.webservicetest.ws</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>jersey-servlet</servlet-name>
        <url-pattern>/devpredator/*</url-pattern>
    </servlet-mapping>
</web-app>
```

CONFIGURACIÓN

Paso 3:

Crear una clase java anotada con @PATH y un método con @PATH y @GET

webservice-test/pom.xml | tiendamusical-reportes/pom.xml | web.xml | TestWS.java

```java
/**
 */
package com.devpredator.webservicetest.ws;

import javax.ws.rs.GET;

/**
 * @author DevPredator
 *
 */
@Path("testWS")
public class TestWS {

    @GET
    @Path("mostrarMensaje")
    public String mostrarMensaje() {
        return "Probando webservice";
    }
}

```