



ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA



FACULTAD DE CIENCIAS DE LA COMPUTACIÓN,
UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE

INFORME TÉCNICO
IMPLEMENTACIÓN DE UN SERVICIO WEB SOAP CON
JAVA Y APACHE TOMCAT
(TrackingSOAPService)

DOCENTE:

Ing. ANGEL CUDCO

NOMBRE:

Santiago David Vega Insuasti
Jessica Jahaira Barberan Castro

MATERIA:

Arquitectura de Software

NRC:

23441

31 de mayo del 2025

INFORME TÉCNICO

IMPLEMENTACIÓN DE UN SERVICIO WEB SOAP CON JAVA Y APACHE TOMCAT

(TrackingSOAPService)

1. INTRODUCCION

En el ámbito de la ingeniería de software y las arquitecturas empresariales actuales, los servicios web desempeñan una función esencial al facilitar la interoperabilidad entre aplicaciones diversas. Gracias a ellos, es posible que sistemas desarrollados en distintos lenguajes de programación y ejecutados en plataformas variadas se comuniquen eficazmente mediante el uso de protocolos estandarizados. Particularmente, los servicios web que utilizan SOAP (Simple Object Access Protocol) proporcionan un marco estructurado para el intercambio de mensajes en formato XML a través de redes, lo cual habilita la creación de soluciones distribuidas, seguras y bien definidas. Este informe técnico tiene como finalidad describir detalladamente el diseño, la implementación y la puesta en marcha de un servicio web SOAP denominado TrackingSOAPService, diseñado para simular el monitoreo de paquetes en tránsito. Este tipo de servicio resulta especialmente relevante para sectores como la logística, el transporte y el comercio electrónico, en los que el rastreo de envíos en tiempo real constituye una parte clave del servicio al cliente y del control operativo interno.

El desarrollo se realizó utilizando el lenguaje de programación Java, aplicando la especificación JAX-WS (Java API for XML Web Services) como base tecnológica, Apache Tomcat 9 como servidor de aplicaciones, y el sistema Maven para gestionar las dependencias y empaquetar el proyecto. El entorno de desarrollo elegido fue IntelliJ IDEA, debido a sus amplias funcionalidades para proyectos web y basados en servicios.

Durante el desarrollo del proyecto, se llevaron a cabo diversas fases que incluyeron la configuración inicial del entorno, la creación del servicio, su definición mediante anotaciones y archivos de configuración (como *web.xml* y *sun-jaxws.xml*), así como la generación del archivo .war y su posterior despliegue en el contenedor web. Para validar el correcto funcionamiento del servicio, se emplearon herramientas como SoapUI, que permitieron simular escenarios de uso real y verificar la respuesta del sistema.

Este proyecto no solo constituye una experiencia práctica en el desarrollo de servicios web, sino que también proporciona una comprensión amplia sobre la integración de distintas tecnologías dentro del enfoque de Arquitectura Orientada a Servicios (SOA), estableciendo así una base sólida para futuras implementaciones más complejas y escalables en entornos empresariales.

2. Objetivo General

Desarrollar e implementar un servicio web basado en SOAP, completamente funcional y accesible, que facilite la consulta del estado de paquetes en tránsito a través de un número de seguimiento único. Para ello, se emplearán

tecnologías estándar como Java, JAX-WS, Apache Tomcat y Maven, con el propósito de consolidar los conocimientos teóricos y prácticos relacionados con la Arquitectura Orientada a Servicios (SOA) dentro de un entorno de desarrollo debidamente configurado y controlado.

3. Objetivos específicos

- **Establecer un entorno de desarrollo compatible y funcional para servicios web SOAP.**

Preparar un entorno de trabajo robusto mediante la instalación y correcta configuración de herramientas esenciales como IntelliJ IDEA, Java Development Kit (JDK) versión 11, Apache Tomcat 9 y el gestor de dependencias Maven. Se busca asegurar la compatibilidad entre las versiones empleadas, así como la integración fluida entre todos los componentes necesarios para el desarrollo y ejecución de servicios web en Java.

- **Desarrollar la lógica interna del servicio web empleando la especificación JAX-WS.**

Crear las clases Java que conforman la estructura lógica del servicio, incluyendo aquellas que representan entidades de datos como *TrackingEvent* y *TrackingStatusResponse*, junto con la clase principal del servicio (*TrackingService*). Se aplicarán las anotaciones correspondientes de JAX-WS para definir y exponer los métodos que serán consumidos como operaciones web por clientes externos.

- **Configurar los archivos de despliegue requeridos para el funcionamiento del servicio.**

Implementar los archivos de configuración necesarios para que el contenedor web Java reconozca y ejecute correctamente el servicio. Esto incluye la elaboración del archivo *web.xml* con las declaraciones pertinentes del servlet, y el archivo *sun-jaxws.xml*, donde se define el punto de acceso (endpoint) al servicio SOAP.

- **Generar el paquete de la aplicación en formato .war mediante Maven.**

Emplear Maven como herramienta de construcción para automatizar tareas como la descarga de dependencias, compilación del código fuente, organización del proyecto en la estructura adecuada, y la creación del archivo *.war*, que contiene la aplicación lista para ser desplegada en un servidor de aplicaciones como Tomcat.

- **Realizar el despliegue operativo del servicio en el servidor Apache Tomcat.**

Ejecutar el proceso de despliegue trasladando el archivo *.war* a la carpeta *webapps* de Apache Tomcat. A continuación, iniciar el servidor utilizando el script *startup.bat* y comprobar que el servicio esté activo, accesible desde el

navegador, y que el archivo WSDL se genere y muestre correctamente como comprobación de su disponibilidad.

- **Ejecutar pruebas de funcionamiento y validación del servicio SOAP.**
Evaluar el rendimiento y la interoperabilidad del servicio mediante pruebas funcionales con herramientas como SoapUI. Se verificará que los mensajes SOAP enviados desde el cliente sean correctamente procesados por el servidor, y que las respuestas contengan los datos esperados. Estas pruebas permitirán validar tanto la lógica implementada como la conformidad del servicio con los estándares de interoperabilidad definidos en el archivo WSDL.

4. Metodología

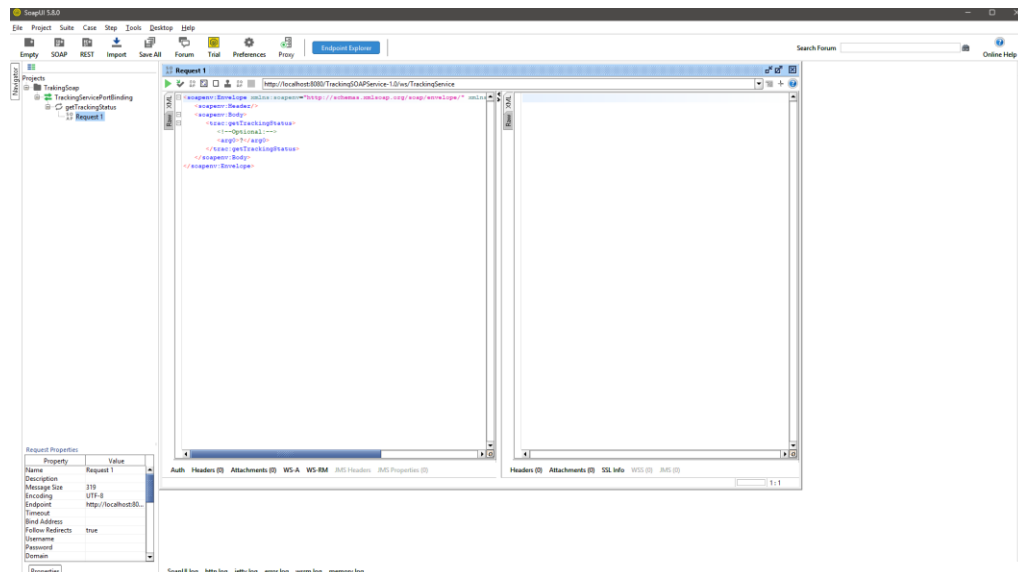
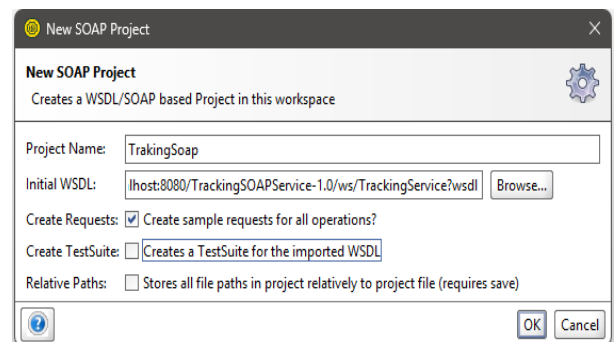
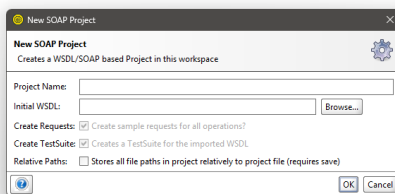
La metodología adoptada para el desarrollo del proyecto TrackingSOAPService fue de carácter práctico-experimental, implementada con un enfoque incremental, iterativo y orientado a la obtención de resultados concretos. Este enfoque permitió avanzar de forma gradual en la construcción de cada componente necesario para la implementación de un servicio web basado en los principios de la Arquitectura Orientada a Servicios (SOA), utilizando tecnologías y herramientas ampliamente reconocidas en la industria del desarrollo de software.

El proyecto se estructuró en cinco etapas fundamentales, detalladas a continuación:

- **Fase 1: Configuración del entorno de desarrollo**
El proceso inició con la preparación del entorno técnico necesario. Se instaló y configuró IntelliJ IDEA como entorno de desarrollo integrado (IDE), debido a su robustez para manejar proyectos Java EE y su compatibilidad con Maven. Se utilizó JDK 11 para asegurar la compatibilidad con las librerías requeridas, y se implementó Apache Tomcat 9 como contenedor de servlets para el despliegue del servicio. Durante esta fase, se verificó la correcta integración entre el IDE, el JDK y el servidor, y se creó un proyecto tipo *Web Application* con soporte para Maven.
- **Fase 2: Organización del proyecto y desarrollo del modelo de datos**
En esta etapa se definió la estructura lógica del proyecto conforme a buenas prácticas de desarrollo web. Se organizaron los paquetes dentro del espacio de nombres com.example.trackingsoapervice y se desarrollaron las clases del modelo, incluyendo *TrackingEvent*, *TrackingStatusResponse* y *TrackingService*. Esta última fue enriquecida con anotaciones de JAX-WS como *@WebService* y *@WebMethod*, con el objetivo de publicar sus métodos como operaciones web disponibles para clientes externos.
- **Fase 3: Configuración de los archivos de despliegue y del endpoint**
Una vez finalizada la implementación del modelo, se configuraron los archivos necesarios para permitir el despliegue del servicio. Se generó el archivo *web.xml* dentro del directorio *WEB-INF*, donde se declararon el servlet

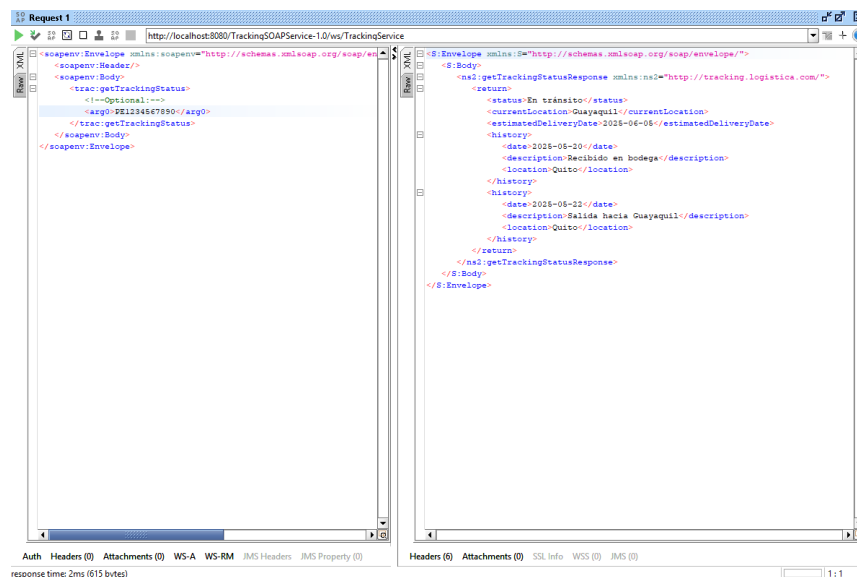
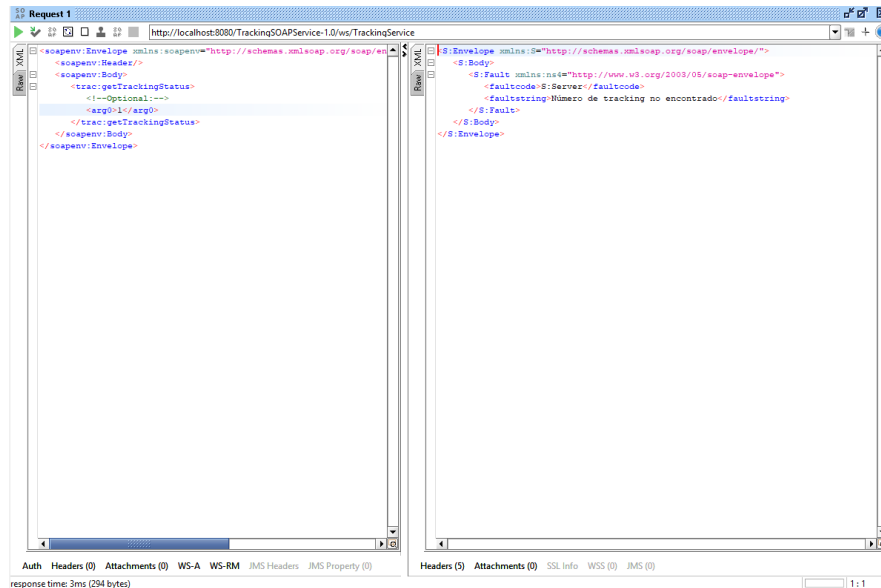
principal (*WSServlet*) y el listener de contexto. Paralelamente, se creó el archivo *sun-jaxws.xml*, que define el endpoint del servicio en la ruta */ws/TrackingService* y lo vincula con la clase correspondiente. Estos archivos de configuración resultan esenciales para que el servidor Tomcat pueda reconocer y ejecutar correctamente el servicio web.

- **Fase 4:** Empaquetado del proyecto y despliegue en el servidor Tomcat
Una vez completadas las configuraciones, se procedió a empaquetar la aplicación utilizando Maven, generando un archivo *.war* (Web Application Archive) que incluye todos los recursos necesarios: clases compiladas, bibliotecas y archivos de configuración. Dicho archivo fue transferido manualmente al directorio *webapps* de la instalación de Tomcat. Posteriormente, se inició el servidor mediante el script *startup.bat* y se verificó el despliegue accediendo al descriptor WSDL del servicio en la dirección: <http://localhost:8080/TrackingSOAPService-1.0/ws/TrackingService?wsdl>.



- **Fase 5:** Validación funcional del servicio con SoapUI

Finalmente, se llevó a cabo la fase de pruebas utilizando SoapUI como herramienta cliente. Se cargó el WSDL del servicio a través de la URL pública proporcionada por Tomcat, lo que permitió a SoapUI generar automáticamente las estructuras de solicitud. Se realizaron múltiples pruebas enviando mensajes SOAP con diferentes códigos de seguimiento, evaluando la respuesta del sistema para validar la funcionalidad del servicio, la precisión del modelo implementado y la correcta interpretación de los mensajes bajo el protocolo SOAP.



5. Resultados

Como producto de la aplicación sistemática del proceso de desarrollo y despliegue del servicio web SOAP, se consiguió implementar exitosamente un sistema operativo que permite realizar consultas sobre el estado de un paquete mediante un identificador único de seguimiento. La construcción del servicio TrackingSOAPService no solo satisfizo los requerimientos funcionales previstos, sino que también contribuyó significativamente a afianzar los conocimientos prácticos sobre la Arquitectura Orientada a Servicios (SOA) y la tecnología JAX-WS aplicada en entornos Java.

Los principales resultados obtenidos se resumen en los siguientes puntos destacados:

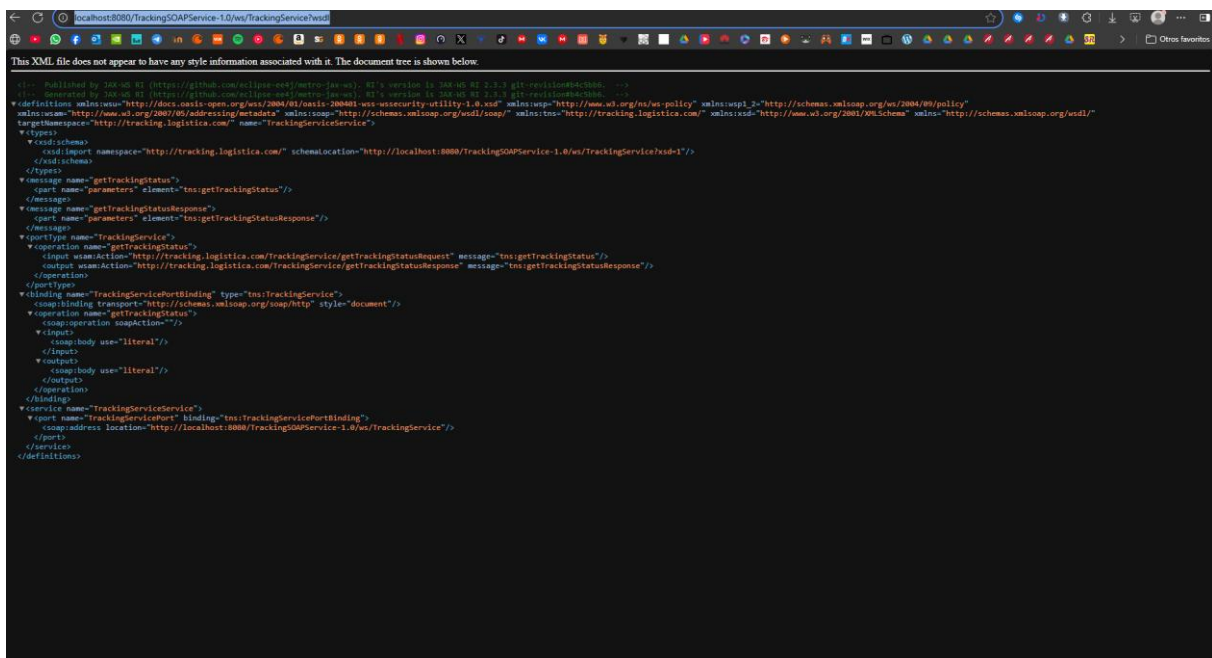
- **Implementación técnica**

Se diseñó e implementó correctamente la clase principal *TrackingService*, incorporando las anotaciones *@WebService* y *@WebMethod*, las cuales permitieron exponer métodos como *getTrackingStatus(String trackingCode)* como operaciones disponibles para ser consumidas externamente. La lógica del servicio fue desarrollada para retornar datos simulados que incluyeran información sobre la ubicación del paquete, su estado actual y una fecha estimada de entrega, validando así su funcionalidad como sistema de rastreo básico en aplicaciones logísticas.

- **Generación y disponibilidad del archivo WSDL**

Al desplegar el archivo *.war* en el servidor Apache Tomcat, el servicio quedó accesible públicamente a través del navegador web desde la siguiente dirección:

<http://localhost:8080/TrackingSOAPService-1.0/ws/TrackingService?wsdl>



```
<?xml version='1.0' encoding='UTF-8'>
<definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http://schemas.xmlsoap.org/ws/2003/05/policy"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing" xmlns:wsu="http://schemas.xmlsoap.org/ws/2004/08/identity"
  xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy" targetNamespace="http://tracking.logistica.com" name="TrackingService">
  <types>
    <xsd:schema
      xmlns="http://tracking.logistica.com" schemaLocation="http://localhost:8080/TrackingSOAPService-1.0/ws/TrackingService?wsdl">
    </xsd:schema>
    </types>
    <message name="getTrackingStatus">
      <part name="parameters" element="tns:getTrackingStatus"/>
    </message>
    <message name="getTrackingStatusResponse">
      <part name="parameters" element="tns:getTrackingStatusResponse"/>
    </message>
    <portType name="TrackingService">
      <operation name="getTrackingStatus">
        <input wsam:action="http://tracking.logistica.com/TrackingService/getTrackingStatusRequest" message="tns:getTrackingStatus"/>
        <output wsam:action="http://tracking.logistica.com/TrackingService/getTrackingStatusResponse" message="tns:getTrackingStatusResponse"/>
      </operation>
    </portType>
    <binding name="TrackingServicePortBinding" type="tns:TrackingService">
      <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
      <operation name="getTrackingStatus">
        <soap:operation soapAction=""/>
        <input>
          <soap:body use="literal"/>
        </input>
        <output>
          <soap:body use="literal"/>
        </output>
      </operation>
    </binding>
    <service name="TrackingServiceService">
      <port name="TrackingServicePort" binding="tns:TrackingServicePortBinding">
        <soap:address location="http://localhost:8080/TrackingSOAPService-1.0/ws/TrackingService"/>
      </port>
    </service>
  </definitions>
```

- Desde esta URL se obtuvo el archivo WSDL (Web Services Description Language), lo cual confirmó que el servicio fue correctamente procesado por el contenedor web y que su interfaz estaba habilitada para su utilización por clientes externos bajo los estándares establecidos para servicios SOAP.
- Estructura del proyecto y empaquetado adecuado
El uso de Maven facilitó una administración ordenada y eficiente de las dependencias necesarias, tales como *javax.servlet*, *jaxws-rt* y *javax.xml.bind*. Asimismo, el proceso de empaquetado generó correctamente el archivo *.war*, el cual fue ubicado en la carpeta *webapps* del servidor Tomcat, permitiendo su despliegue automático. Este resultado evidenció que el proyecto se encuentra alineado con las exigencias técnicas propias de aplicaciones Java EE.
- **Pruebas funcionales exitosas con SoapUI**
Se llevaron a cabo pruebas utilizando SoapUI, herramienta con la que se simularon llamadas desde clientes SOAP. Tras importar el archivo WSDL, se generaron automáticamente las estructuras de solicitud para invocar el método *getTrackingStatus*, usando distintos códigos de seguimiento. Las respuestas generadas por el servicio fueron correctas, presentando mensajes XML con las etiquetas esperadas: *<status>*, *<location>*, *<estimatedDelivery>* y *<trackingCode>*. Estos resultados confirmaron la interoperabilidad del servicio con aplicaciones cliente externas.
- **Corrección de errores y alineación de versiones**
Durante el desarrollo se identificaron varios errores relacionados con incompatibilidades entre versiones de Java, bibliotecas de *javax.servlet* y configuraciones del entorno. Estos inconvenientes fueron resueltos de manera progresiva, permitiendo comprender la importancia de mantener una configuración coherente entre todas las herramientas empleadas: IDE, JDK, servidor de aplicaciones y librerías. Finalmente, se logró estabilizar el proyecto en un entorno basado en Java 11 y Tomcat 9, asegurando así una ejecución estable y sin conflictos.
- **Verificación del entorno mediante pruebas complementarias en el navegador**
Además de la verificación del servicio principal, se desplegó y probó el servlet auxiliar *HelloServlet*, accediendo a través de la dirección:
<http://localhost:8080/TrackingSOAPService/hello-servlet>

Esta prueba adicional permitió comprobar la correcta respuesta del contenedor Tomcat ante solicitudes HTTP del tipo GET y validar el despliegue completo del archivo *.war*, evidenciando la operatividad integral del entorno web.

6. Bibliografía

Oracle. (2024). *Jakarta EE 9 Specification*. <https://jakarta.ee/specifications/>

Apache Software Foundation. (2025). *Apache Tomcat 9 Documentation*.
<https://tomcat.apache.org/tomcat-9.0-doc/>

Java API for XML Web Services (JAX-WS). (2024). *JAX-WS Reference Implementation*. <https://eclipse-ee4j.github.io/metro-jax-ws/>

IntelliJ IDEA. (2025). *Developing a Java Web Application*.
<https://www.jetbrains.com/idea/>

SOAPUI. (2025). *SOAP Web Service Testing Tool*. <https://www.soapui.org/>