

**Tema**

Web Service

SOAP\_CONUI\_JAVA\_GR10

**Tutor**

Ing. Eduardo Mauricio Campaña Ortega

MIS. MDU. CCNA. CCIA.

PhD. (c) Ingeniería de Software

PhD. (c) Seguridad Información

**Fecha**

04/05/2025

**Tabla de contenido**

[INTRODUCCION A LA ARQUITECTURA DE SOFTWARE 4](#_Toc197257749)

[INTRODUCCIÓN 4](#_Toc197257750)

[OBJETIVO 4](#_Toc197257751)

[ARQUITECTURA DE SOFTWARE PARTE I 4](#_Toc197257752)

[ARQUITECTURA DE SOFTWARE PARTE II 9](#_Toc197257753)

[ARQUITECTURA DE SOFTWARE PARTE III 15](#_Toc197257754)

[CONCLUSIONES 20](#_Toc197257755)

[RECOMENDACIONES 21](#_Toc197257756)

[BIBLIOGRAFÍA 21](#_Toc197257757)

**ÍNDICE DE IMÁGENES**

[Figura 1. Analogía entre tipos de construcciones físicas y niveles de complejidad en la arquitectura de software 6](#_Toc197257553)

[Figura 2. La Mansión Winchester: una analogía con sistemas de software sin planeación arquitectónica clara 7](#_Toc197257554)

[Figura 3. La metáfora de las piedras: priorización de elementos clave en la arquitectura de software 9](#_Toc197257555)

[Figura 4. Modelo 4+1 vistas de la arquitectura de software según Kruchten 17](#_Toc197257556)

**ÍNDICE DE TABLAS**

[Tabla 1. Tipos de Requerimientos No Funcionales 13](#_Toc197257232)

# Servicio web SOAP para la conversión de unidades implementado en Netbeans

## INTRODUCCIÓN

La arquitectura de software constituye un componente esencial en el desarrollo de sistemas, ya que establece la estructura y el diseño general que guiará su construcción. Para comprender mejor su propósito y complejidad, puede compararse con la arquitectura tradicional: ambas disciplinas requieren planificación rigurosa, toma de decisiones estructurales y una orientación constante hacia la satisfacción de necesidades reales. Así como un arquitecto de edificaciones debe dominar aspectos técnicos y contextuales, el arquitecto de software debe comprender los requisitos del sistema, anticipar cambios y diseñar soluciones funcionales y sostenibles.

Dentro de esta disciplina, se destacan elementos clave como los componentes, las conexiones, los patrones arquitectónicos y la documentación, los cuales permiten definir el comportamiento y la adaptabilidad del sistema. Además, el modelado y diseño arquitectónico, mediante enfoques como el modelo 4+1 vistas y el uso de UML, resulta fundamental para representar, analizar y comunicar la estructura del software desde distintas perspectivas. Estos recursos no solo facilitan la colaboración entre equipos, sino que aseguran una base sólida para el desarrollo de sistemas escalables y coherentes con su entorno operativo.

## OBJETIVO

Explorar los principios fundamentales de la arquitectura de software mediante su analogía con la arquitectura tradicional, con el fin de comprender su propósito estructural, sus elementos esenciales y la relevancia del modelado y diseño arquitectónico como herramientas clave para construir sistemas robustos, escalables y alineados con las necesidades del entorno y los usuarios.

## ARQUITECTURA DE SOFTWARE PARTE I

## ARQUITECTURA DE SOFTWARE PARTE II

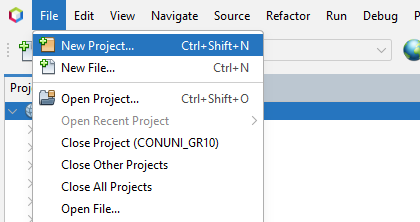
## ARQUITECTURA DE SOFTWARE PARTE III

## DESARROLLO

### APLICACIÓN SERVIDOR

#### Creación del proyecto servidor

Iniciar el IDE Apache NetBeans y, en la pantalla principal, seleccionar la opción “Archivo” en la barra superior, seguida de “Nuevo proyecto”.



1. Creación nuevo proyecto para el servidor en Netbeans

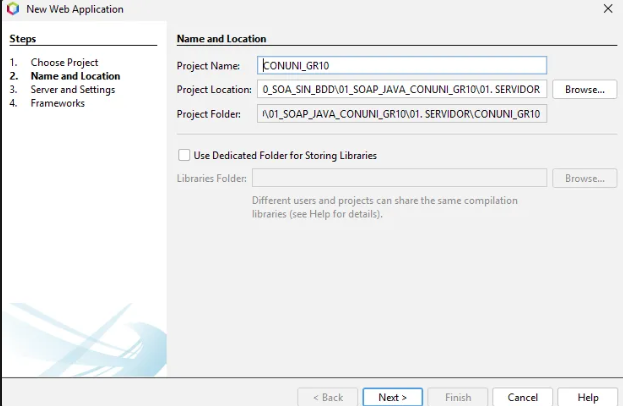
En la ventana emergente, ubicar en la categoría “Java with Ant” el tipo de proyecto “Java Web” y seleccionar “Aplicación Web”.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

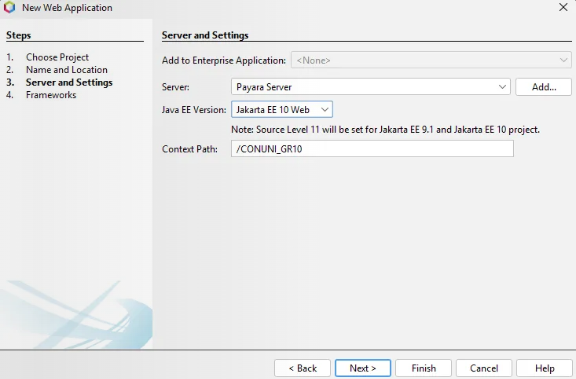
1. Selección de tipo de proyecto en Apache NetBeans

A continuación, asignar el nombre del proyecto, en este caso “CONUNI\_GR10”, y especificar su ubicación en el directorio correspondiente al servidor de SOAP en Java.



1. Configuración del nombre y ubicación del proyecto

Posteriormente, configurar las opciones del proyecto y del servidor donde se ejecutará una vez desplegado. Seleccionar “Payara Server” como servidor y, para la versión de Java, elegir preferentemente la más reciente.



1. Configuración del servidor y versión de Java

Seguidamente, elegir el Framework de desarrollo a utilizar. En este proyecto, no se requerirá ninguno, por lo que no se debe marcar ninguna casilla.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Una vez creado el proyecto, hacer clic derecho sobre el proyecto, seleccionar “Propiedades” en el menú, dirigirse a la pestaña “Librerías” y agregar la dependencia jakarta.jws-api.rar mediante la opción “Add JAR/Folder”, ubicada en la carpeta donde se instaló Payara Server en la máquina local.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

1. Agregar dependencia jakarta.jws-api.rar en las propiedades del proyecto

Para garantizar una estructura adecuada del proyecto, se implementará el patrón arquitectónico MVC. Para ello, crear en “Source Packages” los siguientes paquetes: ec.edu.monster.controlador, ec.edu.monster.vista, ec.edu.monster.modelo, ec.edu.monster.prueba, ec.edu.monster.servicio

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

1. Creación de paquetes para la estructura MVC

#### Creación del servicio web

Crear el servicio web en el paquete “ec.edu.monster.controlador” haciendo clic en “Other…”, seleccionando la carpeta “Web Services” y luego “Web Service”. El Web Service, al facilitar la comunicación entre dos aplicaciones, funciona de manera similar a un controlador, ya que recibe solicitudes de una aplicación externa y las transmite a otro componente para su ejecución. Por esta razón, se crea dentro del paquete “controlador”.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

1. Creación del Web Service en el paquete controlador

Asignar el nombre “CONUNI” al Web Service y verificar que se cree dentro del paquete “ec.edu.monster.controlador”.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

1. Configuración del nombre y paquete del Web Service CONUNI

Después de crear el Web Service, modificar las importaciones de javax.jws a jakarta.jws. Este cambio es necesario porque las especificaciones de Java EE han evolucionado hacia Jakarta EE, y jakarta.jws es el paquete actualizado para las anotaciones de servicios web en entornos modernos como Payara Server.

1. Actualización de importaciones a jakarta.jws en el Web Service CONUNI.java

|  |
| --- |
| package ec.edu.monster.controlador;  import jakarta.jws.WebService; // Cambia javax.jws por jakarta.jws  import jakarta.jws.WebMethod;  import jakarta.jws.WebParam; |

Con el servicio web creado, proceder a desarrollar el servicio para la conversión de unidades en el paquete “ec.edu.monster.servicio”. Hacer clic derecho sobre el paquete, seleccionar “New”, luego “Java Class” y nombrar la clase como “ConversionService”.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

1. Creación de la clase ConversionService en el paquete servicio

En el archivo “ConversionService”, implementar la lógica para la conversión de unidades, incluyendo los siguientes métodos.

1. Implementación de los métodos de conversión en ConversionService

|  |
| --- |
| Aw  package ec.edu.monster.servicio;  /\*\*  \* Service class for unit conversions.  \* @author MATIAS  \*/  public class ConversionService {  public double pulgadasACentimetros(double pulgadas) {  return pulgadas \* 2.54;  }  public double centimetrosAPulgadas(double centimetros) {  return centimetros / 2.54;  }  public double metrosAPies(double metros) {  return metros \* 3.28084;  }  public double piesAMetros(double pies) {  return pies / 3.28084;  }  public double metrosAYardas(double metros) {  return metros \* 1.09361;  }  public double yardasAMetros(double yardas) {  return yardas / 1.09361;  }  } |

Con los métodos desarrollados, integrarlos al Web Service “CONUNI” añadiendo lo siguiente:

1. Integración de los métodos de conversión en el Web Service CONUNI

|  |
| --- |
| package ec.edu.monster.controlador;  import jakarta.jws.WebService; // Cambia javax.jws por jakarta.jws  import jakarta.jws.WebMethod;  import jakarta.jws.WebParam;  import ec.edu.monster.servicio.ConversionService;  import ec.edu.monster.servicio.LoginService;  import ec.edu.monster.servicio.SumaService;  /\*\*  \*  \* @author MATIAS  \*/  @WebService(serviceName = "CONUNI")  public class CONUNI {  @WebMethod(operationName = "pulgadasACentimetros")  public double pulgadasACentimetros(@WebParam(name = "pulgadas") double pulgadas) {  ConversionService service = new ConversionService();  return service.pulgadasACentimetros(pulgadas);  }  @WebMethod(operationName = "centimetrosAPulgadas")  public double centimetrosAPulgadas(@WebParam(name = "centimetros") double centimetros) {  ConversionService service = new ConversionService();  return service.centimetrosAPulgadas(centimetros);  }  } |

Una vez implementado correctamente, ejecutar el proyecto para generar el archivo WSDL. Al hacerlo, se abrirá el navegador con la URL http://localhost:8080/CONUNI\_GR10/. A esta URL, añadir el nombre del servicio web “CONUNI”, resultando en http://localhost:8080/CONUNI\_GR10/CONUNI. Esto mostrará la página del servicio, desde donde se podrá copiar la URL del WSDL: http://localhost:8080/CONUNI\_GR10/CONUNI?wsdl.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

1. Ejecución del proyecto y obtención de la URL del WSDL

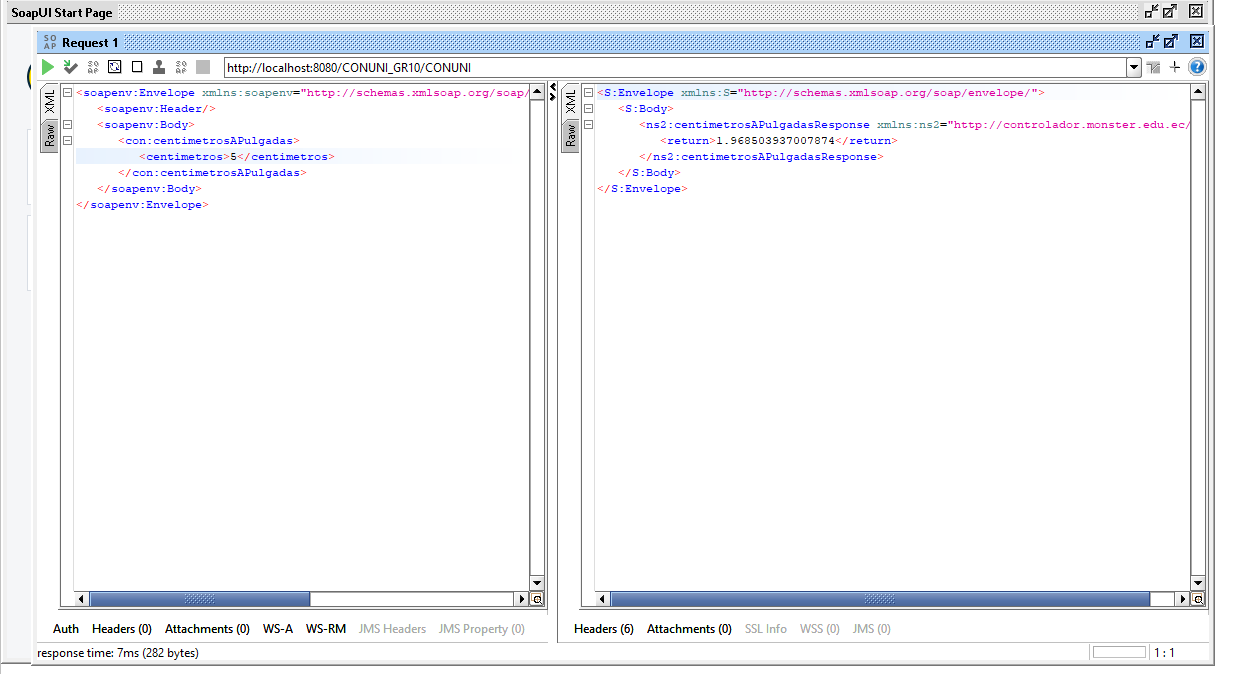
Para verificar el correcto funcionamiento del servicio web de conversiones, utilizar SOAP UI para probar la URL http://localhost:8080/CONUNI\_GR10/CONUNI?wsdl, evaluando las operaciones de conversión disponibles.

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

1. Confoguración del servicio web en SOAP UI

Probar la operación “centímetros a pulgadas” en SOAP UI, verificando el correcto funcionamiento del servicio web.



1. Prueba de la operación centímetros a pulgadas en SOAP UI

Para agregar el servicio de Login, hacer clic derecho en el paquete “ec.edu.monster.servicio”, seleccionar “New” y luego “Java Class” para crear la clase “LoginService”. Implementar en esta clase el método de autenticación con datos predefinidos:

1. Creación e implementación del método autenticar en LoginService

|  |
| --- |
| package ec.edu.monster.servicio;  /\*\*  \*  \* @author MATIAS  \*/  public class LoginService {  public boolean autenticar(String user, String password) {  return user.equals("monster") && password.equals("monster9");  }  } |

Incorporar el método de autenticación al servicio web “CONUNI” añadiendo lo siguiente:

1. Integración del método autenticar en el Web Service CONUNI

|  |
| --- |
| 1. @WebMethod(operationName = "login") 2. public boolean login(@WebParam(name = "usuario") String usuario, @WebParam(name = "contraseña") String contraseña) { 3. LoginService service = new LoginService(); 4. return service.autenticar(usuario,contraseña);   } |

Ejecutar nuevamente el servicio web y utilizar SOAP UI para verificar el correcto funcionamiento de la operación de autenticación.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

1. Ejecución y prueba de la operación autenticar en SOAP UI

## Cliente escritorio

### Creación del cliente para escritorio

Para iniciar el desarrollo del Cliente Escritorio, en la barra superior de NetBeans seleccionar “File” y luego “New Project”. Elegir la categoría “Java with Ant” y el tipo de proyecto “Java Application”.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

1. Creación de un nuevo proyecto para el cliente desktop como Java Application en NetBeans

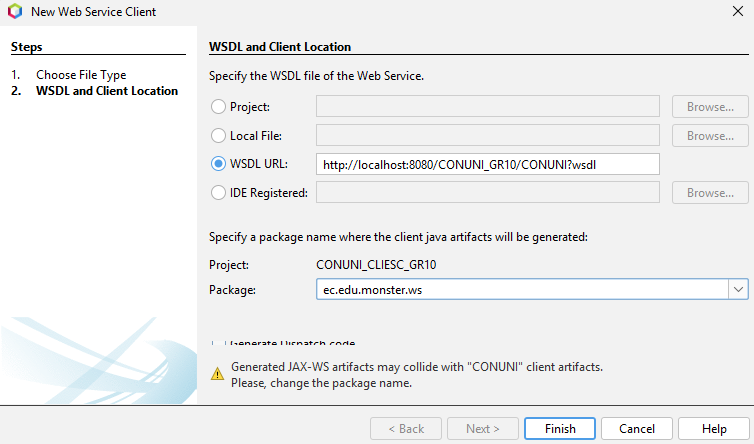
En la ventana emergente, asignar el nombre “CONUNI\_CLIESC\_GR10” al proyecto en el campo “Project Name”. Una vez creado, establecer una estructura adecuada implementando el patrón arquitectónico MVC. Para ello, crear en “Source Packages” los siguientes paquetes: ec.edu.monster.controlador ec.edu.monster.vista ec.edu.monster.modelo ec.edu.monster.prueba ec.edu.monster.servicio

Texto

El contenido generado por IA puede ser incorrecto.

1. Configuración del nombre y creación de paquetes para la estructura MVC del Cliente Escritorio

Para integrar el servicio web, hacer clic derecho en “Source Packages” y seleccionar “New” seguido de “Web Service Client”. En la ventana emergente, elegir la opción “WSDL URL” e ingresar la URL http://localhost:8080/CONUNI\_GR10/CONUNI?wsdl. En el campo de paquete, especificar ec.edu.monster.ws



1. Creación del Web Service Client con la URL del WSDL

En el paquete “ec.edu.monster.servicio”, crear una nueva clase llamada “CONUNIService”. Hacer clic derecho sobre la clase, seleccionar “Insert Code” y luego “Call Web Service Operation”. A continuación, agregar todas las operaciones correspondientes del servicio web.

1. Creación de CONUNIService y adición de operaciones del Web Service

|  |
| --- |
| package ec.edu.monster.servicio;  /\*\*  \*  \* @author MATIAS  \*/  public class CONUNIService {  private static double centimetrosAPulgadas(double centimetros) {  ec.edu.monster.ws.CONUNI\_Service service = new ec.edu.monster.ws.CONUNI\_Service();  ec.edu.monster.ws.CONUNI port = service.getCONUNIPort();  return port.centimetrosAPulgadas(centimetros);  }  private static double pulgadasACentimetros(double pulgadas) {  ec.edu.monster.ws.CONUNI\_Service service = new ec.edu.monster.ws.CONUNI\_Service();  ec.edu.monster.ws.CONUNI port = service.getCONUNIPort();  return port.pulgadasACentimetros(pulgadas);  }  private static double metrosAPies(double metros) {  ec.edu.monster.ws.CONUNI\_Service service = new ec.edu.monster.ws.CONUNI\_Service();  ec.edu.monster.ws.CONUNI port = service.getCONUNIPort();  return port.metrosAPies(metros);  }  private static double piesAMetros(double pies) {  ec.edu.monster.ws.CONUNI\_Service service = new ec.edu.monster.ws.CONUNI\_Service();  ec.edu.monster.ws.CONUNI port = service.getCONUNIPort();  return port.piesAMetros(pies);  }  private static double metrosAYardas(double metros) {  ec.edu.monster.ws.CONUNI\_Service service = new ec.edu.monster.ws.CONUNI\_Service();  ec.edu.monster.ws.CONUNI port = service.getCONUNIPort();  return port.metrosAYardas(metros);  }  private static double yardasAMetros(double yardas) {  ec.edu.monster.ws.CONUNI\_Service service = new ec.edu.monster.ws.CONUNI\_Service();  ec.edu.monster.ws.CONUNI port = service.getCONUNIPort();  return port.yardasAMetros(yardas);  }      } |

## CONCLUSIONES

La arquitectura de software comparte principios fundamentales con la arquitectura tradicional, como la planificación meticulosa, la toma de decisiones estructurales y la orientación a satisfacer necesidades reales.

A su vez, así como un arquitecto de edificaciones requiere conocimientos técnicos y visión estructural, un arquitecto de software debe dominar la programación y comprender profundamente los requerimientos del sistema para construir soluciones funcionales, sostenibles y adaptables, incluso con recursos limitados.

Por otro lado, la arquitectura de software se compone de elementos clave tales como: componentes, conexiones, patrones, requisitos y documentación, que, en conjunto, definen su estructura, comportamiento y capacidad de adaptación.

Es importante contemplar que un arquitecto de software no solo debe dominar estos elementos, sino también anticipar cambios tecnológicos y del negocio, brindando soluciones flexibles

Por último, el modelado y diseño de la arquitectura de software es fundamental para visualizar la estructura, el comportamiento y la interacción de los componentes dentro del entorno donde se desplegará el sistema.

Modelos clave como el 4+1 vistas de Philippe Kruchten, permiten analizar el sistema desde diferentes ángulos, facilitando la comunicación y comprensión entre los distintos grupos involucrados. Al integrar el modelo 4+1 con UML, se logra una visión integral que asegura el diseño y la implementación efectiva de soluciones flexibles y escalables.

## RECOMENDACIONES

Es fundamental considerar que un arquitecto de software no debe centrarse solo en las necesidades actuales del sistema, sino que anticiparse a posibles cambios futuros tanto tecnológicos como comerciales. Esto garantiza que el sistema se mantenga relevante y eficiente a lo largo del tiempo, adaptándose a nuevas demandas sin necesidad de rediseños costosos.

Por otro lado, la arquitectura de software debe facilitar la comunicación clara entre todos los grupos involucrados en el proyecto, por lo que, es recomendable optar por modelos como el 4+1 vistas y UML, lo que asegura que todos los actores, desde desarrolladores hasta clientes, comprendan cómo se estructura y opera el sistema, lo que optimiza la colaboración y la toma de decisiones.

## BIBLIOGRAFÍA

Bass, L., Clements, P., & Kazman, R. (2016). *Software Architecture in Practice* (3.ª edición ed.). Boston, MA: Addison-Wesley.

Booch, G., Rumbaugh, J., & Jacobson, I. (2005). *The Unified Modeling Language User Guide* (Segunda edición (2nd ed.) ed.). Boston: Addison-Wesley.

Elizondo, P. V., Cervantes Maceda, H., & Castro Careaga, L. (2015). *Arquitectura de Software: Conceptos y Ciclo de Desarrollo.* México, D.F.: Cengage Learning Editores, S.A. de C.V.

Ken, A. (26 de 11 de 2023). *Arquitectura de software: ¿Qué es y qué tipos hay?* Obtenido de Gluo: https://www.gluo.mx/blog/arquitectura-de-software-que-es-y-que-tipos-hay

Kruchten, P. (1995). Architectural Blueprints—The “4+1” View Model of Software Architecture. *IEEE Software, 12*, 42–50. doi:https://doi.org/10.1109/52.469759

Ojeda Montoya, J. F. (13 de 9 de 2023). *Arquitectura de Software: Fundamentos, Características, Usos y Ejemplos.* Obtenido de Linkedin: https://es.linkedin.com/pulse/arquitectura-de-software-fundamentos-caracter%C3%ADsticas-y-ojeda-montoya