Universidad de las Ciencias Informáticas
Facultad 3

Componente para la integración del marco de trabajo XEGFORT al Sistema de Gobierno Electrónico

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor:

Luis Javier Rodríguez Castro

Tutores:

MSc. Yordanis García Leiva

Ing. Juan David Gómez Amador

Cotutor:

Ing. Reinier Silverio Figueroa

La Habana, 2017

"Año 59 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los _____ días del mes de _____ del año _____.

_____

Luis Javier Rodríguez Castro

_____ _____

Ing. Juan David Gómez Amador MSc. Yordanis García Leiva

_____

Ing. Reinier Silverio Figueroa

AGRADECIMIENTOS

Agradecerle en primer lugar a mi mamá sin ella hoy no sería la persona que soy, por apoyarme siempre e incondicionalmente con todas las situaciones y momentos de mi vida. A ella que siempre estaba ahí con sus consejos y ese amor incondicional que me ha dado la fuerza para avanzar en la vida. A ti mamá que has confiado en mí y te has sacrificado por mi causa. Sin ti mamá nada de esto hubiese sido posible.

También agradecer a mi hermana que desde siempre ha estado presente y me ha apoyado, ha sido mi fiel confidente, a ti tata que con tu voz y preocupación por mí durante todo este tiempo me ha enseñado que la distancia es poca cuando dos hermanos se aman. A ti mima por ser esa voz de la experiencia en mi conciencia para demostrarme el camino y las decisiones correctas, a ti pipo por estar en nuestras vidas cuando más lo necesitábamos, a mis primas y mis tíos que siempre ha estado al tanto de mí.

Agradecer inmensamente a la familia de mi esposa Roxana, a Leonel, MIchi y Leo que durante todo ese tiempo me han apoyado con mi carrera, a ti Qk por saber comprenderme y amarme de esa manera tan única y hermosa, por estar presente durante momento difíciles y agotadores, por darme ese hijo que ha significado tanto para mí y que me ha despertado ese amor de padre oculto aquí en mi corazón.

Agradecerles a Julio y Juniel por ser esa guía durante todo este tiempo, por demostrarme que a pesar de no estar en el mismo lugar aun así somos capaces de ayudarnos y mantener esa amistad a pesar de la distancia, a mis tutores y cotutor Yordanis, Reinier y Juan por ayudarme y apoyarme durante toda la tesis, simplemente y con mucho amor gracias por todo.

DEDICATORIA

Esta tesis va dedicada especialmente a mi mamá, a mi hijito del alma y a ti Qk, ustedes son mi fuerza y mi conciencia; a todas esas personas que de una forma u otra me ayudaron y apoyaron, a mis compañeros del 58 que realmente muchachos somos una familia y eso es algo que ni el tiempo y la distancia podrán cambiar.

RESUMEN

XEGFORT es un marco de trabajo orientado a la programación en Java, desarrollado en la Universidad de las Ciencias Informáticas. Este tiene un diseño condicionado para estructuras jurídicas y de gobierno, delegadas o divididas jerárquicamente. Además, cuenta con mecanismos de seguridad, activación por puesto de trabajo y gestión de sesiones de trabajo. La variada gama de tecnologías utilizadas en el desarrollo de soluciones informáticas dirigidas al Gobierno Electrónico, impide que XEGFORT pueda integrarse con otras soluciones de este sistema y que se convierta en el marco de trabajo base para el mismo, a pesar de sus características.

La presente investigación tiene como objetivo desarrollar un componente de servicio que permita la integración del marco de trabajo XEGFORT con soluciones informáticas del Sistema de Gobierno Electrónico. La propuesta de solución se centra en la aplicación de servicios web como mecanismos de integración, teniendo en cuenta el uso de la arquitectura orientado a servicios y la utilización de tecnologías de código abierto.

El proceso de desarrollo está guiado por el uso de la metodología de desarrollo de software AUP variación UCI. El resultado de la investigación es validado a través del desarrollo de pruebas funcionales y de rendimiento, utilizando la herramienta automatizada *SoapUI*. Por otra parte, se seleccionan un conjunto de indicadores con el propósito de verificar la relación causa efecto de la variable independiente sobre la dependiente.

PALABRAS CLAVES: Arquitectura Orientada a Servicios, Gobierno Electrónico, marco de trabajo, servicios web, XEGFORT

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

El término Gobierno Electrónico representa el uso de las tecnologías de la información y las comunicaciones (TIC) en los órganos de la administración pública para mejorar la información y los servicios ofrecidos a los ciudadanos, orientar la eficacia y eficiencia de la gestión pública e incrementar sustantivamente la transparencia del sector público y la participación de los ciudadanos (AGESIC, 2017).

En Cuba una de las entidades dedicadas a la aplicación de las tecnologías de la información y las comunicaciones en los órganos de la administración pública es el Centro de Gobierno Electrónico (CEGEL), perteneciente a la Facultad 3 de la Universidad de las Ciencias Informáticas (UCI), que tiene como misión satisfacer necesidades de clientes gubernamentales mediante el desarrollo de productos, servicios y soluciones integrales de alta confiabilidad, calidad, competitividad, fidelidad y eficiencia, a partir de un personal altamente calificado. Todos los servicios y soluciones que ofrece este centro forman parte de un sistema que responde a las necesidades y la gobernanza de las instituciones jurídicas y de gobierno para las cuales están destinados, por lo que el intercambio de información entre ellas constituye un aspecto fundamental para el logro de la interoperabilidad.

La interoperabilidad constituye la habilidad de más de un sistema o componente para intercambiar información y utilizar la misma (IEEE, 1999).

CEGEL ha desarrollado soluciones informáticas para los distintos órganos de gobiernos nacionales e internacionales, tales como el Tribunal Supremo Popular y la Fiscalía General de la República de Cuba, así como la División de Antecedentes Penales y el Servicio Autónomo de Registros y Notarías de la República Bolivariana de Venezuela.

La experiencia del centro en los proyectos antes mencionados ha demostrado que el Gobierno Electrónico es un sistema heterogéneo en cuanto a las tecnologías utilizadas. Lo anterior se justifica teniendo en cuenta que en CEGEL se utiliza más de un lenguaje de programación (Java y PHP) para el desarrollo de sus aplicaciones y componentes informáticos, sin existir hasta el momento una solución que permita integrar los mismos, a pesar de sus semejanzas en la arquitectura. Es importante destacar que, en el caso de las aplicaciones informáticas, estas presentan elementos comunes como la administración

(gestión de usuarios, roles, trazas y excepciones) y la seguridad, sin embargo, a partir de la dificultad que tienen para integrarse con otras soluciones desarrolladas en CEGEL debido a la incompatibilidad tecnológica que estas presentan, se hace necesario cada vez que se necesite desarrollar una nueva solución, volver a implementar estos elementos, sin tener la posibilidad de reutilizarlos.

CEGEL cuenta con un marco de trabajo denominado XEGFORT dirigido al desarrollo de soluciones informáticas para el Gobierno Electrónico. Este marco de trabajo ha sido utilizado con resultados satisfactorios en el desarrollo de soluciones informáticas para instituciones como la División de Antecedentes Penales de la República Bolivariana de Venezuela y la Cámara de Comercio de la República de Cuba. Además, se encuentra registrado en el Centro Nacional de Derechos de Autor (CENDA) en el año 2010. Este fue implementado sobre la plataforma J2EE y entre sus características se encuentran:

- Soporta varias instituciones: está diseñado para estructuras jurídicas y de gobierno delegadas o divididas jerárquicamente.
- Cuenta con mecanismos de seguridad, además de otros como activación por puesto de trabajo, gestión de sesiones de trabajo.
- Cuenta con componentes configurables de administración como roles, permisos y accesibilidad que pueden ser reconfigurados en cualquier momento del desarrollo o dentro del ambiente de producción.
- Debido a que está desarrollado sobre J2EE se integra con cualquiera de las tecnologías de esta plataforma.
- Reduce el tiempo necesario en fases de desarrollo como implementación ya que la abstracción de la plataforma J2EE permite disminuir la competencia de desarrollo con Java a niveles académicos de la universidad.

Atendiendo a estas características, XEGFORT puede comportarse como marco de trabajo base para la integración de las soluciones informáticas que conforman el Sistema de Gobierno Electrónico. Este cuenta con elementos comunes que poseen las soluciones informáticas de dicho sistema y se comporta como un componente independiente, que a su vez se integra dentro de estas soluciones informáticas.

Teniendo en cuenta las diferencias que pueden existir entre las tecnologías presentes en las soluciones informáticas desarrolladas para el Gobierno Electrónico, con las cuales

puede interactuar el marco de trabajo, XEGFORT no cuenta con los elementos necesarios para lograr la integración entre estas soluciones .Los elementos anteriores no permiten que este pueda constituir el marco de trabajo base para el sistema de Gobierno Electrónico, ni contribuir al logro de la interoperabilidad técnica entre soluciones informáticas desarrolladas para este sistema, a pesar de sus características.

A partir de la situación problemática antes descrita se define el siguiente **problema de investigación:**

¿Cómo lograr la integración del marco de trabajo XEGFORT con soluciones informáticas del Sistema de Gobierno Electrónico, contribuyendo a la interoperabilidad técnica entre los sistemas desarrollados en CEGEL?

Teniendo en cuenta el problema antes propuesto se define como **objeto de estudio:** mecanismos de integración entre sistemas.

Se identifica como **campo de acción:** arquitectura orientada a servicio aplicada al marco de trabajo XEGFORT.

Definiéndose como **objetivo general:** desarrollar un componente de servicios web para la integración del marco de trabajo XEGFORT con soluciones informáticas del Sistema de Gobierno Electrónico, contribuyendo a la interoperabilidad técnica entre los sistemas desarrollados en CEGEL.

Del objetivo general se desglosan los siguientes **objetivos específicos:**

- ✓ Elaborar el marco teórico referencial de la investigación, relacionado con los mecanismos de integración entre sistemas. `
- ✓ Desarrollar un componente de servicios web para la integración del marco de trabajo XEGFORT con soluciones informáticas del Gobierno Electrónico.
- ✓ Validar la solución propuesta utilizando métricas y herramientas automatizadas.
- ✓ Validar a través de indicadores, la relación causa efecto entre las variables definidas en la idea a defender de la presente investigación.

Posibles resultados:

- ✓ Obtener un componente de servicios web para la integración del marco de trabajo XEGFORT con soluciones informáticas del Sistema de Gobierno Electrónico, que contribuya a la interoperabilidad técnica entre los sistemas desarrollados en CEGEL.

Se define como **idea a defender:** el desarrollo del componente servicios web para la integración del marco de trabajo XEGFORT con soluciones informáticas del Sistema de Gobierno Electrónico, contribuye a la interoperabilidad técnica entre los sistemas desarrollados en CEGEL.

Métodos teóricos:

Para dar cumplimiento a las tareas de investigación se utilizaron los siguientes métodos teóricos:

- ✓ **Analítico-sintético**: este método permitió, luego del análisis de las características del marco de trabajo XEGFORT y los mecanismos de integración existentes, identificar los servicios web como solución a utilizar para el desarrollo del componente propuesto.
- ✓ **Análisis histórico-lógico**: se utilizó en el estudio de los mecanismos de integración, permitiendo analizar las características de cada uno de estos y definir cuál es el más apropiado para ser utilizado en el desarrollo del componente de servicios web propuesto.

Métodos Empíricos:

Para el cumplimiento de estas tareas se utilizó el siguiente método empírico:

- ✓ **Entrevista:** permitió obtener información concerniente a las experiencias de la utilización del marco de trabajo XEGFORT, así como las principales deficiencias que este presenta para integrarse con otras soluciones del Sistema de Gobierno Electrónico, aplicando una entrevista individual de tipo no estructurada al arquitecto de XEGFORT.

El contenido de este trabajo consta de tres capítulos, definidos de la siguiente forma:

Capítulo 1: Fundamentación teórica

En el presente capítulo se abordan los aspectos teóricos relacionados con la investigación, las herramientas, lenguajes y tecnologías que se emplean en la construcción de la solución, así como aspectos esenciales que sirvan de soporte a la misma. Además, se realiza una descripción de los principales mecanismos de integración y patrones de diseño existentes.

Capítulo 2: Descripción del componente de servicios web

En este capítulo se describe la solución propuesta. Se hace referencia a la técnica empleada para la captura de los requisitos. Se presentan los requisitos funcionales y no funcionales con los que debe cumplir el componente de servicios web propuesto en la presente investigación. Los requisitos funcionales son agrupados en historias de usuario. En el capítulo también se muestran los diagramas de componentes referentes a la arquitectura del marco de trabajo XEGFORT y del componente de servicios web. Por otra parte, se expone la utilización de los patrones de diseño y estándares de codificación utilizados durante la implementación.

Capítulo 3: Validación del componente de servicios web

El capítulo contiene los elementos que forman parte de la validación de la solución. Se exponen los resultados de la aplicación de métricas relaciones entre clases y tamaño operacional de clases para evaluar el diseño de la solución. Por otra parte, se describe cómo fue validada la calidad de la implementación, haciendo uso de herramientas automatizadas. De igual forma, se muestran los resultados obtenidos en la comprobación de la relación causa-efecto entre las variables definidas en el diseño metodológico de la presente investigación.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se abordan los aspectos teóricos relacionados con la investigación, las herramientas, lenguajes y tecnologías que se emplean en la construcción de la solución, así como aspectos esenciales que sirvan de soporte a la misma. Además, se realiza una descripción de los principales mecanismos de integración y patrones de diseño existentes.

1.1. Conceptos fundamentales asociados al dominio del problema

A continuación, se abordan los conceptos fundamentales asociados al dominio del problema.

Interoperabilidad

La interoperabilidad es la habilidad de los sistemas de tecnologías de la información y las comunicaciones (TIC) y de los procesos de negocio que ellas soportan, de intercambiar datos y posibilitar compartir información y conocimiento (RTA, 2017).

La interoperabilidad es un concepto básico para un correcto funcionamiento en el marco de la administración electrónica. Se define también como la habilidad necesaria para que organizaciones y sistemas diversos puedan interactuar con objetivos consensuados y comunes, además de tener como finalidad común la persecución de beneficios comunes. La interoperabilidad se estructura en tres dimensiones: técnica, semántica y organizativa (RTA, 2017).

Interoperabilidad técnica

La interoperabilidad técnica es aquella dimensión relativa a la relación entre sistemas y servicios de tecnología de la información, donde tienen cabida aspectos como las interfaces, la interconexión, la integración de datos y servicios, la presentación de la información, la accesibilidad o la seguridad (RTA, 2017).

Interoperabilidad semántica

La interoperabilidad semántica es aquella dimensión relativa a que la información intercambiada debe ser interpretable de forma automática y reutilizable por aplicaciones que ni siquiera intervienen en su creación (RTA, 2017).

Interoperabilidad organizativa

La interoperabilidad organizativa es aquella dimensión relativa a la capacidad de las entidades, y de los procesos a través de los cuales llevan a cabo sus actividades, para colaborar con el objetivo de alcanzar logros mutuamente acordados relativos a los servicios que prestan (RTA, 2017).

Una vez analizadas cada una de las dimensiones de la interoperabilidad, el autor de la presente investigación decide acotar el alcance de la solución desarrollada solo a la dimensión técnica, teniendo en cuenta que el componente de servicios web propuesto no está dirigido a un dominio en específico.

Arquitectura Orientada a Servicios (SOA)

El concepto de Arquitectura Orientada a Servicios (SOA, por sus siglas en inglés *Service Oriented Architecture*) hace referencia a un enfoque de arquitectura cuyo objetivo es la creación de sistemas a partir de servicios autónomos. Con SOA, la integración pasa a ser una reflexión previa más que una idea posterior. Probablemente, la solución final esté formada por servicios desarrollados en distintos lenguajes de programación y se aloje en plataformas diferentes con numerosos modelos de seguridad y procesos empresariales. Aunque este concepto pueda parecer altamente complejo, en realidad no se trata de una idea nueva. Ciertos sectores afirman que SOA fue el resultado de las experiencias asociadas al diseño y el desarrollo de sistemas distribuidos basados en tecnologías previamente disponibles. Gran parte de los conceptos asociados a SOA, como los servicios, el descubrimiento y el enlace en tiempo de ejecución, ya estaban asociados a las tecnologías CORBA[1] y DCOM[2]. Buena parte de los principios de diseño de servicios cuentan con numerosos aspectos en común con las técnicas basadas en la encapsulación, la abstracción y las interfaces claramente definidas (Microsoft, 2005).

Lenguaje de descripción de servicios web (WSDL):

[1] Por sus siglas en inglés Common Object Request Broker Architecture. Es un estándar que permite que diversos componentes de software distribuidos en entornos heterogéneos puedan trabajar en conjunto.
[2] Modelo de Objetos de Componentes Distribuidos.

El lenguaje de descripción de servicios web (WSDL, por sus siglas en inglés *Web Servicie Description Language*) es un documento XML que describe un conjunto de mensajes SOAP y la forma en que estos son enviados y recibidos. Es un documento que especifica, sin ambigüedad, lo que debe contener un mensaje de petición y lo que debe contener un mensaje de respuesta. Parte de las especificaciones en un WSDL consiste en determinar dónde está disponible un servicio y qué protocolo de comunicación utiliza. Todo lo anterior significa que el archivo WSDL, define todo lo necesario para escribir un programa que trabaja con un servicio web (W3C, 2013).

Lenguaje de definición de esquemas *XML* (*XSD*):

El propósito del lenguaje de definición de esquemas XML (XSD, por sus siglas en inglés *XML Schema Definition Language*) es definir y describir una clase de documentos de XML usando los componentes del esquema para limitar y documentar el significado, uso y relaciones de sus partes constitutivas: el tipo de dato, elementos y volumen, atributos y valores (W3C, 2012).

Bus de servicio empresarial (ESB):

El bus de servicio empresarial (ESB, por sus siglas en inglés *Enterprise Servicie Bus*) es un elemento de integración (multiprotocolo y multipropósito) en las arquitecturas orientadas a servicios o *Service Oriented Architecture* (SOA). Combina servicios web, mensajería, transformación, encaminamiento y enriquecimiento de datos, políticas de seguridad, entre otros. Permite la interacción entre aplicaciones heterogéneas desde las más modernas hasta las más convencionales (IBM, 2006).

1.2. Mecanismos de integración

Desde mediado de la década de los 90, con la aparición y extensión de internet, ha existido la necesidad de integración entre sistemas heterogéneos, tanto de software como de hardware. En este sub-epígrafe se abordan conceptos y puntos de vista que definen la manera más factible para proceder a una integración entre soluciones informáticas. Además, se describen los principales mecanismos existentes para lograr la integración entre estas soluciones, los cuales se describen a continuación:

Base de datos

La integración de datos es el proceso de transformación y conciliación de datos que permite una mayor agilidad en la gestión, proporcionando datos conectados, seguros y de calidad (powerdata, 2015).

El uso de marcos de trabajo que implementan el patrón arquitectónico mapeo objeto-relación (ORM por sus siglas en inglés *Object Relationship Mapping*) asegura en gran medida la información a partir de ataques como inyección de código SQL. Sin embargo, se expone el acceso a las bases de datos de cada solución informática a integrar, aumentando las probabilidades a que ocurran ataques que pudieran proveer acceso pleno a la información. Por otra parte, la estandarización es otro aspecto que se afecta, debido a la diversidad de gestores de bases de datos que se pueden utilizar.

Réplica de datos

El proceso de replicación consiste en replicar las consultas de actualización de un nodo maestro (en idioma inglés *master*) sobre uno o varios nodos esclavos (en idioma inglés *slave*), de manera que se obtiene una copia de las mismas a lo largo del tiempo (adictosaltrabajo, 2009).

La réplica de datos es uno de los mecanismos de integración más utilizados en sistemas distribuidos. Este mecanismo puede generar un crecimiento de las bases de datos debido a que los datos se copian en cada uno de los nodos que intervienen. Mayormente las herramientas de réplicas están desarrolladas para intercambiar datos de bases de datos de un mismo gestor. La réplica de datos también expone las bases de datos como el caso de la sección anterior con la salvedad que el sistema informático solo accede a la base de datos propia. Puede ser síncrona o asíncrona, pero en cualquiera de los casos puede provocar que los datos no estén disponibles en el momento que se desee o que exista un gran tráfico en la red (adictosaltrabajo, 2009).

Servicios Web

El término "servicios web" designa una tecnología que permite que las aplicaciones se comuniquen en una forma que no depende de la plataforma ni del lenguaje de programación. Un servicio web es una interfaz de software que describe un conjunto de operaciones a las cuales se puede acceder por la red a través de mensajería XML[3]

[3] Lenguaje de Marcas Extensible

estandarizada. Usa protocolos basados en el lenguaje XML con el objetivo de describir una operación a ejecutar o datos para intercambiar con otro servicio web. Un grupo de servicios web que interactúan de esa forma define la aplicación de un servicio web específico en una arquitectura orientada a servicios (SOA) (IBM, 2017).

Este mecanismo utiliza estándares como XML, HTTP[4] y HTTPS[5] que provee una ventaja considerable. De no asegurarse bien la información que se intercambia, los servicios web están propensos a ataques, aunque cuenta con mecanismos de seguridad que permiten disminuir las vulnerabilidades de este tipo. El acceso a los servicios web puede protegerse de varias formas y restringirlas a nivel de datos dependiendo de los privilegios. En el caso de los servicios que utilizan SOAP[6] son más seguros, pero la mantenibilidad de estos puede afectar a los sistemas consumidores. En cambio los servicios que utilizan REST[7] son más vulnerables, pero son más escalables.

Estos mecanismos se utilizan en diferentes entornos donde los beneficios que brindan son aprovechados al máximo, logrando así una solución idónea para el medio en que se desarrollan; pero su mal empleo puede afectar el proceso de desarrollo. Teniendo en cuenta los mecanismos de integración analizados previamente, se decide utilizar en la presente investigación los servicios web, pues estos permiten que las aplicaciones se comuniquen independientemente de su plataforma, el lenguaje de programación o la base de datos. Además, estos utilizan una interfaz de software que describe un conjunto de operaciones a las cuales se puede acceder por la red a través de mensajería estandarizada.

1.3. Metodología de Desarrollo de Software

Las Metodologías de Desarrollo de Software son el conjunto de procedimientos, técnicas y soporte documental utilizados para el diseño de sistemas de información. Su objetivo principal es exponer un conjunto de técnicas clásicas y modernas de modelado de sistemas que hagan posible desarrollar un software de calidad, incluyendo heurísticas de construcción y criterios de comparación de modelos de sistema (conceptodefinicion, 2017).

[4] Protocolo de Transferencia de Hipertexto
[5] Protocolo de Transferencia de Hipertexto Seguro
[6] Protocolo de Acceso Simple a Objetos
[7] Transferencia de Estado Representacional

Para el desarrollo del marco de trabajo XEGFORT se utilizó la metodología Variación de AUP para la UCI. Teniendo en cuenta que para el desarrollo del componente de servicios web propuesto en la presente investigación se reutilizan con conjunto de artefactos generados para XEGFORT, el autor de este trabajo de diploma decide continuar trabajando con esta metodología de desarrollo de software. Esta decisión se adopta con el propósito de realizar la menor cantidad de cambios posibles en los artefactos reutilizados.

Variación de AUP para la UCI

AUP UCI es una variación de la metodología Proceso Unificado Ágil (AUP, por sus siglas en inglés *Agile Unified Process*) versión simplificada del Proceso Unificado de Rational (RUP),. la cual se adapta al ciclo de vida definido para la actividad productiva de la UCI. Se apoya en el Modelo Integrado de Capacidad y Madurez para Desarrollo versión 1.3 (CMMI-DEV v1.3, por sus siglas en inglés *Capability Maturity Model Integration for Development versión 1.3*), el cual constituye una guía para aplicar las mejores prácticas en una entidad desarrolladora. Estas prácticas se centran en el desarrollo de productos y servicios de calidad (Sánchez, 2015).

Fases de AUP-UCI

De las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición) se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes 3 fases de AUP en una sola denominada Ejecución y se agrega la fase de Cierre. A continuación, una descripción de cada una de ellas (Sánchez, 2015).

Inicio

Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto (Sánchez, 2015).

Ejecución

En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto (Sánchez, 2015).

Cierre

En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto (Sánchez, 2015).

Disciplinas de AUP-UCI

AUP propone 7 disciplinas (Modelo, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyecto y Entorno), se decide para el ciclo de vida de los proyectos de la UCI tener 7 disciplinas también, pero a un nivel más atómico que el definido en AUP. Los flujos de trabajos: Modelado de negocio, Requisitos y Análisis y diseño en AUP están unidos en la disciplina Modelo, en la variación para la UCI se consideran a cada uno de ellos disciplinas. Se mantiene la disciplina Implementación, en el caso de Prueba se desagrega en 3 disciplinas: Pruebas Internas, de Liberación y Aceptación. Las restantes 3 disciplinas de AUP asociadas a la parte de gestión para la variación UCI se cubren con las áreas de procesos que define CMMI- DEV v1.3 para el nivel 2, serían CM (Gestión de la configuración), PP (Planeación de proyecto) y PMC (Monitoreo y control de proyecto) (Sánchez, 2015).

Escenarios para la disciplina Requisitos AUP-UCI

Escenario No 1: Proyectos que modelen el negocio con CUN (Casos de uso del negocio) solo pueden modelar el sistema con CUS (Casos de uso de sistema) (Sánchez, 2015).

Escenario No 2: Proyectos que modelen el negocio con MC (Modelo conceptual) solo pueden modelar el sistema con CUS (Casos de uso de sistema) (Sánchez, 2015).

Escenario No 3: Proyectos que modelen el negocio con DPN (Descripción de Proceso de Negocio) solo pueden modelar el sistema con DRP (Descripción de requisitos por proceso) (Sánchez, 2015).

Escenario No 4: Proyectos que no modelen negocio solo pueden modelar el sistema con Historias de usuario (HU) (Sánchez, 2015).

El autor de la presente investigación decide utilizar el escenario 4 (modelar el sistema con HU) teniendo en cuenta que para el desarrollo de la solución propuesta no se modela el negocio.

1.4. Lenguajes

El término lenguaje "puede ser entendido como un recurso que hace posible la comunicación" (IEEE, 1999). En el mundo de la informática existen varios tipos de lenguajes que no son usados precisamente para establecer la comunicación entre los hombres. A continuación, se mencionan y describen los principales lenguajes empleados en la investigación.

1.4.1. Lenguaje de modelado

El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener y controlar la información sobre tales sistemas. UML está diseñado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios (Jacobson, 2000).

1.4.2. Lenguaje de programación

Un lenguaje de programación es aquella estructura que, con una base sintáctica y semántica, imparte distintas instrucciones a un programa de computadora (IEEE, 1999). Existe una gran gama de lenguajes de programación utilizados en el proceso de desarrollo de software. Para el desarrollo de la presente investigación se decide utilizar el lenguaje Java, teniendo en cuenta la experiencia del desarrollador en el uso de este lenguaje, además el marco de trabajo al cuál responde la solución propuesta está implementado en Java.

Java es un lenguaje de alto nivel basado en el paradigma Orientado a Objetos cumpliendo con características como encapsulación, herencia y polimorfismo. Es un lenguaje de alto nivel que junto a la Máquina Virtual de Java o JVM (del inglés *Java Virtual Machine*),

permite que los proyectos sean ejecutados en diferentes entornos de hardware o software. Sus usos se extienden desde dispositivos móviles, aplicaciones de servidor y clientes, entre otras. Para el desarrollo de proyectos con un grado de complejidad técnica según las necesidades de estos, se puede hacer uso de multihilos, que garantiza la realización de actividades simultáneas, logrando un resultado en cuanto a rendimiento y respuestas del sistema. Permite la búsqueda y tratamiento de errores en tiempo de ejecución por lo que se considera un lenguaje robusto (Oracle, 2013).

1.4.3. Lenguaje de etiquetado extensible (XML)

El Lenguaje de Etiquetado Extensible (XML, por sus siglas en inglés *Extensible Markup Language*) es un meta-lenguaje que permite definir lenguajes de marcas desarrollado por el *World Wide Web Consortium*, utilizado para almacenar datos en forma legible. Proviene del lenguaje SGML8 y permite definir la gramática de lenguajes específicos (de la misma manera que HTML9 es a su vez un lenguaje definido por SGML) para estructurar documentos grandes. A diferencia de otros lenguajes, XML da soporte a bases de datos, siendo útil cuando varias aplicaciones deben comunicarse entre sí o integrar información (Atwood, 2009).

Teniendo en cuenta la definición antes expuesta y además que XML es un lenguaje estándar para el intercambio de mensajes entre servicios web, el autor de la presente investigación decide utilizar XML como lenguaje de marcado para el desarrollo de la solución propuesta.

1.5. Herramientas

Para apoyar y facilitar el desarrollo de la solución se utilizan una serie de herramientas. En el presente epígrafe se describen y explican cada una de estas, teniendo en cuenta sus características y versión.

Entorno de desarrollo integrado (IDE):

Un entorno de desarrollo integrado (IDE por sus siglas en inglés *Integrated Development Environment*) es un programa compuesto por un conjunto de herramientas para programar en un lenguaje de programación. Consiste en un editor de código, un

[8] Lenguaje de marcado generalizado estándar
[9] Lenguaje de marcas de hipertexto

compilador, un depurador y un constructor de interfaz gráfica (Fergarciac, 2013). Existe una gran gama de IDE utilizados en el proceso de desarrollo de software. En la presente investigación se decide utilizar la plataforma de desarrollo integrado *Eclipse Mars* en su versión 2.0, haciendo uso a través de esta del *WSO2 Developer Studio* como herramienta para desarrollar los servicios web. La decisión de utilizar estas tecnologías se fundamenta a partir de la bibliografía consultada, en la cual se plantea que estas son las más recomendadas para desarrollar aplicaciones sobre la plataforma de WS02 (ESB, BPS[10], DSS[11]), además permiten desarrollar aplicaciones web JAX-RS[12], JAX-WS[13], servicios Axis2 [14] y aplicaciones web tradicionales (emmersonmiranda, 2015).

Eclipse Mars.2 Release:

Eclipse es una plataforma de desarrollo, diseñada para ser extendida de forma indefinida a través de *plugins*. Fue concebida desde sus orígenes para convertirse en una plataforma de integración de herramientas de desarrollo. No se especifica un lenguaje de programación, sino que es un IDE genérico, aunque goza de mucha popularidad entre la comunidad de desarrolladores del lenguaje Java usando el *plugin* herramientas de desarrollo Java (JDT en sus siglas en inglés *Java Development Tools*) que viene incluido en la distribución estándar del IDE.

Proporciona herramientas para la gestión de espacios de trabajo, escribir, desplegar, ejecutar y depurar aplicaciones (ecplise, 2017).

WSO2 Developer Studio

DevStudio es el IDE recomendado para desarrollar aplicaciones sobre la plataforma de WS02 (ESB, BPS[15], DSS[16]), también permite desarrollar aplicaciones web JAX-RS, JAX-WS, servicios Axis2 y aplicaciones web. Este IDE es un *plugin* de eclipse que se puede instalar o bien se puede descargar una versión binaria pre-instalada de la web de WS02 (emmersonmiranda, 2015).

[10] Servidor de procesos empresariales
[11] Servidor de servicios de datos
[12] API de Java para Servicios Restful
[13] API de Java para Servicios Web XML
[14] Motor nuclear para servicios web
[15] Servidor de procesos empresariales
[16] Servidor de servicios de datos

DevStudio simplifica el proceso de creación de artefactos de forma gráfica y la administración de sus dependencias, además no solo soporta el proceso de desarrollo-*testing-debugging*, sino también el despliegue de artefactos en los servidores (emmersonmiranda, 2015). Este soporta la creación de proyectos y artefactos para los siguientes servidores:

- WSO2 Enterprise Service Bus
- WSO2 Data Services Server
- WSO2 Application Server
- WSO2 Business Process Server
- WSO2 Business Rules Server
- WSO2 Governance Registry (emmersonmiranda, 2015)

1.6. Patrones de diseño

Un patrón es una descripción de un problema y la solución, a la que se da un nombre, y que se puede aplicar a nuevos contextos; idealmente, proporciona consejos sobre el modo de aplicarlo en varias circunstancias, y considera los puntos fuertes y compromisos. Muchos patrones proporcionan guías sobre el modo en el que deberían asignarse las responsabilidades a los objetos, dada una categoría específica del problema (Larman, 2005).

Patrón de servicio fundacional

Este patrón hace referencia a las capacidades y contexto agnósticos de un servicio, que simplemente se refiere a la capacidad de un servicio para que pueda ser reutilizado en varios sitios en las aplicaciones de la empresa.

Para que esto sea posible, la lógica dentro del servicio tiene que estar encapsulada y no depender de nada externo al propio servicio (andreshevia, 2015).

Patrones sobre el inventario de servicios

Este patrón se refiere a que es preferible diseñar los servicios para evitar el cambio de protocolo y el cambio del modelo de datos. Es decir, que no haga falta cambiar esto cada vez que un servicio llama a otro, simplemente porque es un trabajo ímprobo mapear los campos de un servicio y otro en tiempo de construcción y también pasa factura en tiempo

de ejecución, ya que es necesario realizar miles o millones de transformaciones que podrían ser evitables.

Otro de los objetivos es el evitar los servicios duplicados o repetidos. Es decir, que para hacer una cosa concreta sólo haya un servicio y no dos, tres, o n veces, como suele ocurrir en algunas empresas (andreshevia, 2015).

Transformación

En una arquitectura SOA es necesario la transformación del formato, del modelo de datos y de los protocolos. Se debe procurar que en los nuevos servicios esta necesidad sea mínima, evidentemente se debe acceder a antiguos programas y aplicaciones legadas en la que es imprescindible hacer estas transformaciones (andreshevia, 2015).

Patrones de mensajería

En este apartado se recogen los patrones que en ocasiones se pueden encontrar como características, que están presentes en una implementación de servicios:

- Colas asíncronas de mensajes.
- Mensajes orientados a eventos.
- Envío fiable, es decir, garantizar que el mensaje va a llegar a su destinatario, aunque no haya red.
- *Callback* o cómo el destino puede devolver la llamada al cliente.
- Enrutado de mensajes.
- Intermediario de mensajes para su transformación, *logs* y gestión de errores. (Hevia, 2015).

1.7. Pruebas automatizadas

En las pruebas de software, la automatización de pruebas consiste en el uso de un software especial (casi siempre separado del software que se prueba) para controlar la ejecución de pruebas y la comparación entre los resultados obtenidos y los resultados esperados. La automatización de pruebas permite incluir pruebas repetitivas y necesarias dentro de un proceso formal de pruebas ya existente o bien adicionar pruebas cuya ejecución manual resultaría difícil (Prince, 2014).

Para el desarrollo de pruebas automatizadas se utilizan disímiles herramientas en correspondencia con el tipo de prueba a realizar, tal es el caso de la herramienta *SoapUI*.

SoapUI es una herramienta, desarrollada en java, para la realización de pruebas a aplicaciones con arquitectura orientada a servicio (SOA) y transferencia de estado representacional (REST). Soporta múltiples protocolos como SOAP, REST, HTTP, JMS[17] y JDBC[18]. Posee una versión de Código abierto y otra versión de pago realizada por la compañía *SmartBear*. Fue lanzada en septiembre de 2005 en *SourceForge* (SmartBear, 2017).

Teniendo en cuenta que las características de la herramienta anterior están en correspondencia con la solución obtenida, se decide utilizar SoapUI para el desarrollo de pruebas automatizadas al componente de servicios web propuesto.

1.8. Conclusiones parciales

Con el desarrollo del capítulo se concluye que el análisis del marco teórico referencial relacionado con los términos arquitectura orientada a servicio, lenguaje de descripción de servicios web, lenguaje de definición de esquemas XML, bus de servicio empresarial y servicios web, facilita una mejor comprensión del objeto de estudio de la presente investigación. Además, el análisis de réplica de datos, integración entre base de datos y servicios web constituye la base para la definición del mecanismo de integración a utilizar. Por otra parte, el estudio de las características de la metodología AUP variación UCI, las distintas herramientas utilizadas en la solución propuesta y en el desarrollo de pruebas automatizadas, fundamenta la correspondencia de la selección de estos para la implementación y validación del componente de servicios web propuesto.

[17] Servicio de mensajes Java
[18] Conectividad de bases de datos Java

CAPÍTULO 2: DESCRIPCIÓN DEL COMPONENTE DE SERVICIOS WEB

En el presente capítulo se realiza una descripción de la solución propuesta, a partir de las disciplinas requisitos, análisis y diseño e implementación definidas por la metodología seleccionada. En la disciplina de requisitos se describen los requisitos funcionales (RF) y no funcionales (RnF) que responden al desarrollo de la solución, y la forma en que los RF son encapsulados a través de HU. Por otra parte, en la disciplina análisis y diseño se describe la arquitectura empleada, así como los patrones de diseño de servicios web utilizados en la definición del componente propuesto. Por último, en la descripción de la implementación se especifican los estándares de codificación aplicados y se realiza una explicación de las funcionalidades de la solución propuesta.

2.1. Disciplina Requisitos

El esfuerzo principal en la disciplina Requisitos es desarrollar un modelo del sistema que se va a construir y comprende la administración de los requisitos funcionales y no funcionales del producto (Sánchez, 2015). A partir de que en el Capítulo 1 se decidió utilizar el escenario de HU para la disciplina de Requisitos a partir de las características de la solución propuesta, en el presente epígrafe se presentan los requisitos obtenidos, encapsulados en HU.

2.1.1. Técnica para la obtención de requisitos

Para la captura de los requisitos que responden a la solución propuesta se utilizó la siguiente técnica:

- ✓ Entrevista: es de gran utilidad para obtener información cualitativa, requiere seleccionar bien a los entrevistados para obtener la mayor cantidad de información en el menor tiempo posible. Es muy aceptada y permite acercarse al problema de una manera natural (Soulary, 2010). Esta técnica fue aplicada al arquitecto principal de XEGFORT utilizando una entrevista de tipo individual no estructurada.

- ✓ Tormenta de ideas: es una técnica de reuniones en grupo cuyo objetivo es la generación de ideas en un ambiente libre de críticas o juicios (Jacobson, 2000). En el caso de la presente investigación esta técnica se aplicó a través de encuentros con los desarrolladores del marco de trabajo XEGFORT, los cuales aportaron

criterios sobre las principales funcionalidades de este marco de trabajo que podían ser expuestas a través de servicios web, así como los elementos de seguridad y eficiencia a tener en cuenta en el desarrollo de la solución propuesta.

Como resultado de aplicar las técnicas antes descritas se obtuvieron los RF y RnF que se listan en el siguiente sub-epígrafe.

2.1.2. Especificación de requisitos

Los requisitos constituyen un punto clave en el desarrollo de aplicaciones informáticas. Un gran número de proyectos de software fracasan debido a una mala definición, especificación o administración de estos. Los requisitos se enfocan en las especificaciones de lo que se desea desarrollar y tienen dos clasificaciones: requisitos funcionales y no funcionales. Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a escenarios específicos y los requisitos no funcionales son restricciones de los servicios o funciones ofrecidos por el sistema (Sommerville, 2005). A continuación, se presentan los requisitos definidos para el desarrollo del componente de servicio propuesto.

Requisitos funcionales

Tabla 1. Requisitos funcionales. Fuente: elaboración propia

No	Nombre	Descripción
RF1	Autenticar usuario	Se realiza la autenticación de usuario en el sistema.
RF2	Adicionar usuario	Se adiciona un nuevo usuario en el sistema.
RF3	Buscar usuario	Realizar la búsqueda de usuarios a través del criterio Nombre de usuario.
RF4	Adicionar rol	Se adiciona un nuevo rol.
RF5	Modificar rol	Modificar un rol en el sistema.
RF6	Eliminar rol	Eliminar un rol en el sistema.
RF7	Buscar rol	Realizar búsquedas de roles a través del criterio Identificador de oficina.
RF8	Adicionar traza	Se adicionan las trazas de los usuarios con su respectiva descripción en el sistema.
RF9	Buscar trazas	Realizar búsquedas de las trazas en el sistema en

		dependencias de los criterios de búsquedas especificados por parámetro (Usuario, IP de inicio, IP de fin, Fecha de inicio, Fecha de fin, Si es histórico, Mínimo, Máximo, Sistema, Descripción).
RF10	Adicionar sesión	Se adiciona una sesión activa en el sistema.
RF11	Deshabilitar sesión activa del sistema	Se deshabilita una sesión activa del sistema.
RF12	Buscar sesiones activas del sistema	Buscar sesiones activas en el sistema según los parámetros de búsquedas especificados (Nombre, Primer apellido, Segundo apellido, Nombre de usuario, IP de inicio, IP de fin, Usuario autenticado).

Requisitos no funcionales

- ✓ **Seguridad**

  RnF 1: los usuarios deben estar autenticados en el sistema para poder ejecutar los servicios correspondientes a sus roles.

  RnF 2: restringir el acceso al sistema a través de la gestión de usuarios y delimitar el acceso de cada usuario con permisos específicos.

- ✓ **Eficiencia**

  RnF 3: el sistema debe ser capaz de ofrecer tiempos de respuesta mínimos iguales a 4 segundos.

Es importante señalar que los 12 RF listados anteriormente fueron obtenidos en correspondencia con los RF que responden al marco de trabajo XEGFORT, los cuales ya fueron validados durante la implementación del marco de trabajo. Teniendo en cuenta lo anterior en la presente tesis no se realiza la validación de los mismos.

2.1.3. Historias de usuario

Las historias de usuario son una forma rápida de administrar los requisitos de los usuarios sin tener que elaborar gran cantidad de documentos formales y sin requerir de tiempo para administrarlos. Las historias de usuario permiten responder rápidamente a los requisitos cambiantes, por lo que una historia de usuario puede tener varios cambios a lo largo de un desarrollo sin afectarse el tiempo (Beck, 2002). A continuación, en la Tabla 2

se muestra la HU correspondiente al RF 2, teniendo en cuenta que este RF representa una de las principales funcionalidades dentro del marco de trabajo XEGFORT. Es necesario señalar que cada una de las HU utilizadas en la presente investigación corresponde a las HU definidas durante el desarrollo del marco de trabajo XEGFORT, las cuales son usadas por el autor de esta tesis con el propósito de describir cada una de las funcionalidades a exponer a través del componente de servicio web propuesto, (ver Anexo 1**:** Historias de usuario).

Tabla 2. HU: Adicionar usuario. Fuente: elaboración propia.

Número: 2	Nombre del requisito: Adicionar usuario
Programador: : Luis Javier Rodríguez Castro	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 15 horas
	Tiempo Real: 20 horas
Descripción: Una vez autenticado en el sistema y en dependencia del rol al que pertenece el usuario procederá adicionar un nuevo usuario con la información requerida para la operación.	

2.2. Disciplina Análisis y Diseño

En esta disciplina los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de estos, y una descripción que sea fácil de mantener y ayude a la estructuración del sistema (incluyendo la arquitectura). Además, en esta disciplina se modela el sistema y su forma para que soporte todos los requisitos, incluyendo los requisitos no funcionales. Los modelos desarrollados son más formales y específicos que el de análisis (Sánchez, 2015).

2.2.1. Diseño arquitectónico

Para dar solución al problema planteado se propone agregar al marco de trabajo XEGFORT un componente de servicios que exponga las funcionalidades de administración y configuración como servicio, de forma tal que permita utilizarlo como marco de trabajo base dentro del Sistema de Gobierno Electrónico independientemente

de la distribución y tecnologías a utilizar. Además de sentar las bases para que cada proyecto que lo utilice como marco de trabajo base se pueda integrar con otros sistemas.

El componente *Xegfort_Services* se desarrolla teniendo en cuenta una Arquitectura Orientada a Servicios. La solución propuesta requiere extender arquitectónicamente el marco de trabajo XEGFORT. En la Figura 1 se representa la arquitectura de XEGFORT, así como las relaciones de uso entre cada uno de los componentes una vez desarrollado e integrado al marco de trabajo el componente de servicios web junto al ESB.

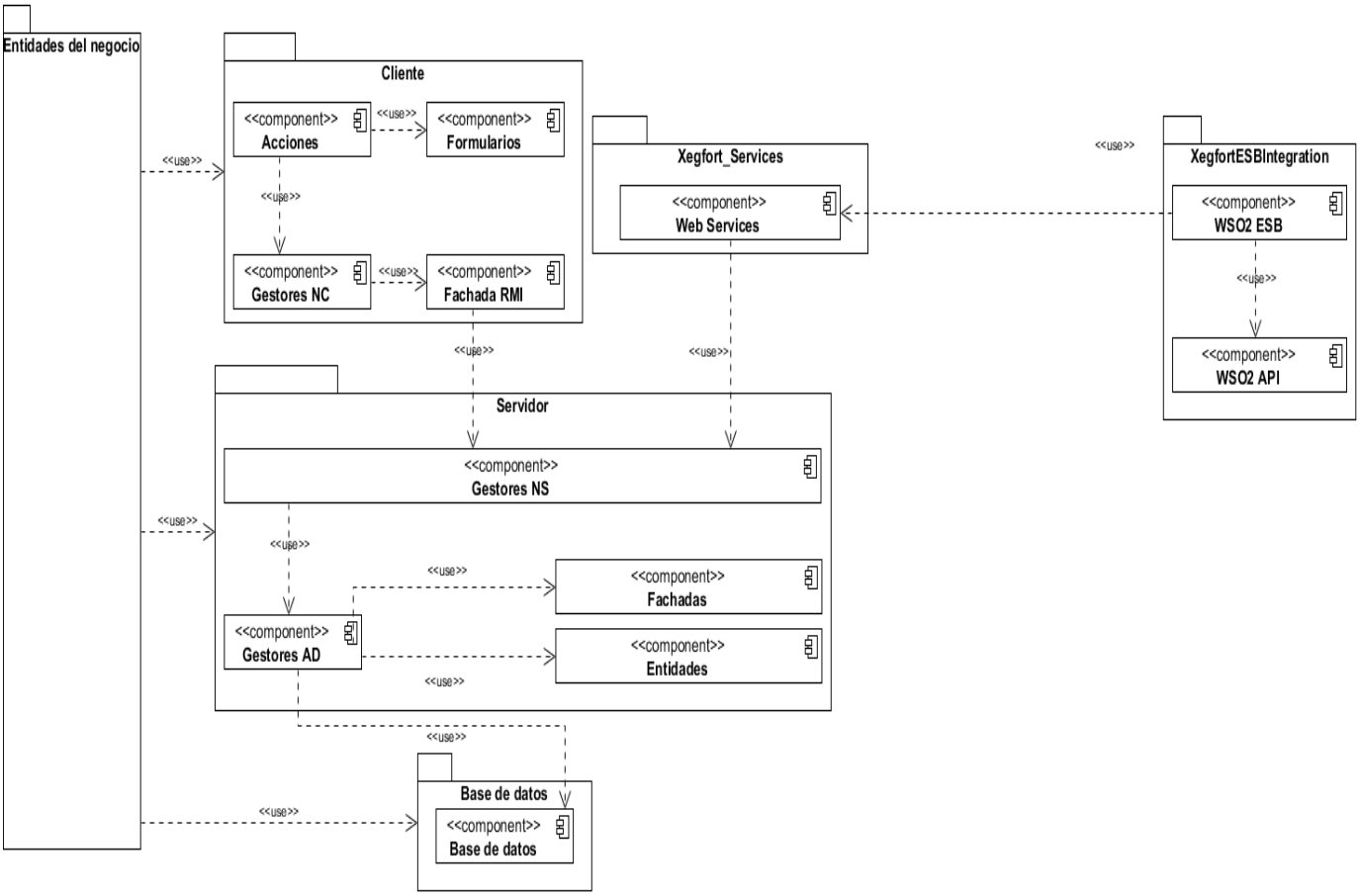

Figura 1. Arquitectura del Marco de trabajo. Fuente: elaboración propia

Teniendo en cuenta la figura anterior en la Tabla 3 se describen los elementos que componen la arquitectura de XEGFORT, con el propósito de demostrar la necesidad de extender arquitectónicamente el marco de trabajo, para lograr su integración al Sistema de Gobierno Electrónico.

Tabla 3. Descripción de los elementos de la arquitectura. Fuente: elaboración propia.

Elemento	Descripción

Entidades de negocio	Contiene las clases que representan el modelo de negocio.
Acciones	Clases encargadas de gestionar las interfaces de usuario. Realiza las validaciones de los componentes de interfaz de usuario.
Formularios	Tienen las interfaces de usuario basadas en swing.
GestoresNC	Clases que implementan la lógica de negocio asociadas al cliente, básicamente validaciones del lado del cliente vinculadas con restricciones de negocio.
Fachada RMI[19]	Clase que se encarga de la comunicación con los *beans* [20] de sesión desplegados en el contenedor de aplicaciones.
GestoresNS	*Beans* de sesión remotos sin estado que implementan la lógica de negocio
GestoresAD	*Beans* de sesión locales sin estado que implementan la lógica de datos

Es importante señalar que la arquitectura básica del marco de trabajo XEGFORT no cuenta con un mecanismo que facilite su integración al Sistema de Gobierno Electrónico, teniendo en cuenta que el método de invocación de los *beans* de sesión desplegados en el servidor, utiliza el protocolo RMI. Este es el un mecanismo ofrecido por Java para invocar un método de manera remota. RMI forma parte del entorno estándar de ejecución de Java y proporciona un mecanismo simple para la comunicación de servidores en aplicaciones distribuidas basadas exclusivamente en Java. Si se requiere comunicación entre otras tecnologías debe utilizarse CORBA o SOAP en lugar de RMI (Troelsen, 2010).

[19] Método de invocación remota para Java.
[20] Componente de software que tiene la particularidad de ser reutilizable.

A partir de la solución propuesta y su integración al marco de trabajo XEGFORT, a continuación, en la Figura 2 se representan los elementos que forman parte del componente *Xegfort_Services* representado en la Figura 1.

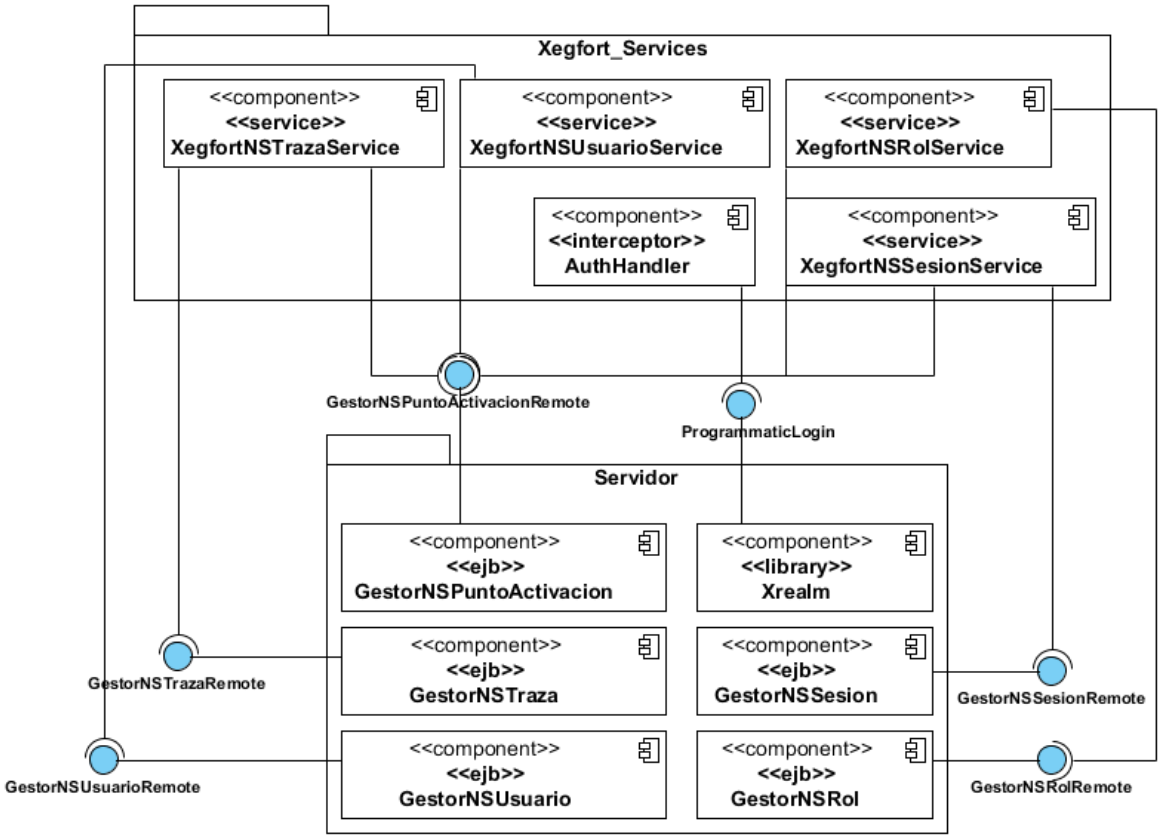

Figura 2. *Xegfort_Services.* Fuente: elaboración propia

Los elementos representados en la Figura 2 muestran que la solución propuesta cuenta con 4 servicios web, los cuales están compuestos por operaciones, cuantificándose un total de 14 operaciones. Cada uno de estos servicios consume los respectivos EJB[21] contenidos en la capa servidor de XEGFORT. A continuación, se realiza una descripción de estos 4 servicios:

***XegfortGestorNSRolService*:** este servicio agrupa las operaciones necesarias para la gestión de roles del marco de trabajo. A continuación, en la Tabla 4 se describen las operaciones contenidas en el mismo.

[21] Componente de software que tiene la particularidad de ser reutilizable en el entorno de una aplicación Java empresarial.

Tabla 4. Descripción de las operaciones del servicio *XegfortGestorNSRolService.* Fuente: elaboración propia.

Operación	**Descripción**
registrarRol	Operación mediante la cual se pueden adicionar roles al marco de trabajo. Para invocar esta operación se requiere: • El rol a insertar. La invocación retorna: • Esta operación no devuelve resultados.
modificarRol	Operación mediante la cual se pueden modificar los roles del marco de trabajo. Para invocar esta operación se requiere: • El rol a modificar. La invocación retorna • Esta operación no devuelve resultados.
eliminarRol	Operación mediante la cual se pueden eliminar los roles del marco de trabajo. Para invocar esta operación se requiere: • El rol a eliminar La invocación retorna: • Un valor verdadero en caso de realizarse la operación con éxito y falso en caso contrario.
obtnerRolPorIdOficina	Operación mediante la cual se obtienen los roles asociados a una oficina específica. Para invocar esta operación se requiere: • Identificador de la oficina La invocación retorna: • Un listado con los roles de la oficina especificada por parámetro.

***XegfortGestorNSUsuarioService*:** este servicio se encarga de la autenticación y registro de cuentas. A continuación, en la Tabla 5 se describen las operaciones contenidas en el mismo.

Tabla 5. Descripción de las operaciones del servicio *XegfortGestorNSUsuarioService*. Fuente: elaboración propia.

Operación	**Descripción**
Autenticar	Autentica a un usuario a nivel de aplicación Para invocar esta operación se requiere: • Nombre de usuario • Contraseña La invocación retorna: • El usuario autenticado
Autenticar	Autentica a un usuario a nivel de aplicación Para invocar esta operación se requiere: • Nombre de usuario • Contraseña • Dirección IP La invocación retorna: • El usuario autenticado
obtenerUsuarioPorUsuario	Operación mediante la cual se obtiene una entidad usuario a partir del usuario introducido como parámetro. Para invocar esta operación se requiere: • Nombre de usuario La invocación retorna: • El usuario correspondiente al valor introducido como parámetro
registrarCuentaUsuario	Esta operación permite crear una nueva cuenta de usuario en el marco de trabajo.

	Para invocar esta operación se requiere: • Entidad persona • Nombre de cuenta • Contraseña • Estado del usuario • Cambio de contraseña • Identificador de oficina • Funcionario La invocación retorna: • El usuario que ha sido registrado

XegfortGestorNSSesionService: este servicio se encarga de hacer todo lo referente a la gestión de las sesiones en el sistema. A continuación, en la Tabla 6 se describen las operaciones contenidas en el mismo.

Tabla 6. Descripción de las operaciones del servicio *XegfortGestorNSSesionService*. Fuente: elaboración propia.

Operación	**Descripción**
buscarSesion	Devuelve un listado de sesiones que cumplan con los parámetros de búsqueda. Para invocar esta operación se requiere: • Nombre • Primer apellido • Segundo apellido • Nombre de usuario • IP de inicio • IP de fin • Usuario autenticado

	La invocación retorna: • El listado de sesiones que cumplan con los parámetros de búsqueda.
adicionarSesionActiva	Adiciona una sesión activa al sistema. Para invocar esta operación se requiere: • Sesión de usuario. La invocación retorna: • La sesión registrada.
eliminarSesionActiva	Operación mediante la cual se elimina una sesión de usuario en el sistema. Para invocar esta operación se requiere: • Id del usuario La invocación retorna: • La operación no devuelve resultados.

***XegfortGestorNSTrazaService*:** este servicio se encarga de realizar todos los temas referentes a la gestión de las trazas en el sistema. A continuación, en la Tabla 7 se describen las operaciones contenidas en el mismo.

Tabla 7. Descripción de las operaciones del servicio *XegfortGestorNSTrazaService*. Fuente: elaboración propia.

Operación	**Descripción**
insertarTraza	Esta operación registra una traza en el sistema. Para invocar esta operación se requiere: • Descripción de a traza. • Si es una traza de sistema o no. La invocación retorna: • La traza registrada.
insertarTraza	Esta operación registra una traza en el sistema.

	Para invocar esta operación se requiere: • Descripción de a traza • Si es una traza de sistema o no • Sesión de dónde se realizó la operación La invocación retorna: • La traza registrada
devolverTrazas	Operación mediante la cual se realiza una búsqueda parametrizada de las trazas en el sistema. Para invocar esta operación se requiere: • Usuario • IP de inicio • IP de fin • Fecha de inicio • Fecha de fin • Si es histórico • Mínimo • Máximo • Sistema • Descripción La invocación retorna: • Un listado con las trazas que cumplan con los parámetros de búsqueda.

2.2.2. Patrones de diseño

En este sub-epígrafe se evidencia como los patrones de diseño se utilizaron en el desarrollo del componente de servicios web de la presente investigación.

Patrón de servicio fundacional

A continuación, se muestra en la Figura 3 como este patrón se utiliza en el componente de servicios web.

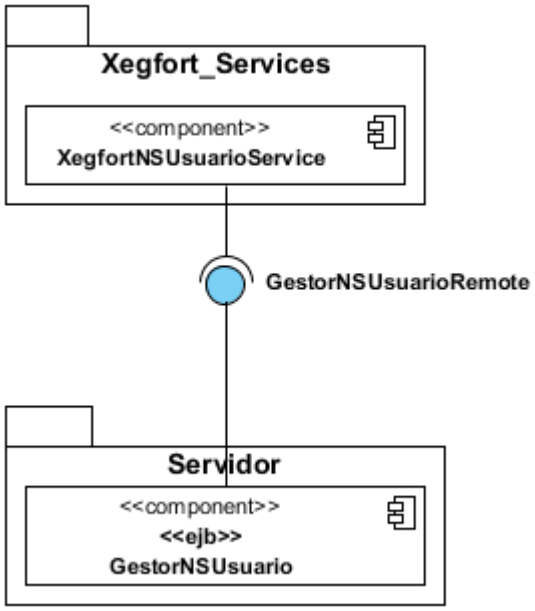

Figura 3. Patrón de servicio fundacional. Fuente: elaboración propia.

Se aprecia en la figura anterior como el componente de servicios web *Xegfort_Services* no depende de elementos externos a la arquitectura del marco de trabajo XEGFORT, sólo se relaciona con los EJB de la capa del Servidor, así queda evidenciado el uso del patrón fundacional. Para más detalles referentes a los componentes que conforman la arquitectura ver la Figura 1.

Transformación

En la Figura 4 se muestra el XSD correspondiente a los atributos que pertenecen a la operación *registrarCuentaUsuario* del servicio *XegofrtNSUsuarioServicie*.

```
▼<xs:complexType name="registrarCuentaUsuario_1">
  ▼<xs:sequence>
     <xs:element name="persona" type="tns:edPersona"/>
     <xs:element name="nombreCuenta" type="xs:string"/>
     <xs:element name="contrasena" type="xs:string"/>
     <xs:element name="estado" type="tns:edEstadoUsuario"/>
     <xs:element name="cambiarContrasena" type="xs:boolean"/>
     <xs:element name="idOficina" type="xs:int"/>
     <xs:element name="funcionario" type="xs:boolean"/>
   </xs:sequence>
 </xs:complexType>
▼<xs:complexType name="edPersona">
  ▼<xs:sequence>
     <xs:element name="fechaNacimiento" type="xs:dateTime" minOccurs="0"/>
     <xs:element name="idPersona" type="xs:int"/>
     <xs:element name="numeroCedula" type="xs:string" minOccurs="0"/>
     <xs:element name="numeroIdentificacion" type="xs:string" minOccurs="0'
     <xs:element name="primerApellido" type="xs:string"/>
     <xs:element name="primerNombre" type="xs:string"/>
     <xs:element name="segundoApellido" type="xs:string" minOccurs="0"/>
     <xs:element name="segundoNombre" type="xs:string" minOccurs="0"/>
     <xs:element name="sexo" type="xs:unsignedShort" minOccurs="0"/>
     <xs:element name="tipoDocumento" type="xs:string" minOccurs="0"/>
   </xs:sequence>
```

Figura 4. *XSD* correspondiente al servicio *XegfortNSUsuarioServicie*. Fuente: elaboración propia.

El tipo de dato complejo denominado *edPersona* en el XSD se transforma en el tipo de dato *EDPersona* una vez que se ejecuta el servicio con los atributos necesarios que conforman la entidad, en la Figura 7 se evidencia esta trasformación.

Inventario de servicios

Los XSD definen el modelo de datos a utilizar para cada invocación de los servicios, esto se puede evidenciar en la Figura 4. Además, este patrón también define que no deben existir servicios duplicados, evitando así que las funcionalidades en los servicios sean redundantes ver Figura 2. Para más detalles sobre las funcionalidades de los servicios ver las HU.

2.3. Disciplina de implementación

En esta disciplina a partir de los resultados del análisis y el diseño se construye la solución que da lugar a la presente investigación.

2.3.1. Estándares de codificación

Para el desarrollo de la solución se decide utilizar el estándar de codificación para Java definido por la Dirección Técnica de la Producción basado en Convenciones de Código para el lenguaje de programación JAVA™ (Hommel, 2007). Dentro de las normativas de este estándar se utilizan las siguientes.

- Los nombres de las clases serán con mayúscula, en caso de ser un nombre compuesto las siguientes palabras se escribirán de igual forma, en la Figura 5 se evidencia el uso de este estándar en la clase *XegfortNSUsuarioServicie*.
- Los nombres de los métodos serán con minúscula, en caso de ser un nombre compuesto las siguientes palabras se escribirán con mayúscula, ver Figura 5.
- Los identificadores de los parámetros serán con letras minúsculas y en caso de ser un nombre compuesto las siguientes palabras se escribirán con mayúscula, en la Figura 5 se evidencia el uso de este estándar de codificación.
- Las clases no deben de exceder de más de 1000 líneas de código.
- El código de las funcionalidades es descrito a través de comentarios de una línea que especifica el RF al cual corresponde.

2.3.2. Descripción de la solución propuesta

La solución propuesta se centra en el desarrollo de un componente de servicios web. Este componente consta de 4 servicios según lo mostrado en la Figura 2. A continuación se detallan los pasos para ejecutar la operación *registrarCuentaUsuario* perteneciente al servicio *XegfortNSUsuarioServicie*. La descripción de este ejemplo constituye una muestra de cómo son descritos cada uno de los flujos de las operaciones que conforman el componente de servicios web implementado. Los pasos a seguir son los siguientes:

Primeramente, se expone el contrato o documento WSDL, (ver Anexo 2: **Contrato o documento WSDL.**).

Una vez consultado el documento WSDL del servicio, es necesario conocer el esquema o XSD del servicio, (ver Anexo 3: **Esquema o XSD del servicio.**).

Luego de haber analizado el contrato WSDL y el XSD se procede a detallar el flujo de operaciones.

El flujo de operaciones comienza cuando una petición (*request*) es realizada por un cliente. Esta es interceptada por un manejador (*handler*), denominado *AuthHandler*, cuyo objetivo es verificar si la petición ha sido autenticada previamente. En caso negativo, *AuthHandler* procede a autenticar la petición y la ejecuta. Para una mejor comprensión, en la Figura 5 se representa el rol del método *handleMessage* encargado de verificar si la petición ha sido autenticada. En esta figura la línea 39 expone la forma en que se realiza la autenticación contra el servidor de aplicaciones y la línea 40 la autenticación contra el sistema.

```
32
33      @Override
34      public boolean handleMessage(SOAPMessageContext messageContext) {
35          HttpServletRequest req = (HttpServletRequest) messageContext.get(MessageContext.SERVLET_REQUEST);
36          if (req.getUserPrincipal() == null) {
37
38              try {
39                  req.login("administrador", "administrador");
40                  ejbRef.autenticar("administrador",
41                          "b20b0f63ce2ed361e8845d6bf2e59811aaa06ec96bcdb92f9bc0c5a25e83c9a6", obtenerIP());
42              } catch (Exception ex) {
43                  Logger.getLogger(AuthHandler.class.getName()).log(Level.SEVERE, null, ex);
44              }
45          }
46          return true;
47      }
```

Figura 5. Método *handleMessage*. Fuente: elaboración propia.

A partir de la ejecución de la línea 39 representada en la Figura 5 se deriva una serie de procedimientos ejecutados en el *Realm* de XEGFORT, haciendo referencia al componente *XRealm* representado en la Figura 2. Se denomina *Realm* al dominio de autenticación del servidor de aplicaciones. El método Autenticar representado en la Figura 6 constituye el núcleo del componente *XRealm*.

```
145     public String[] autenticar(String nombreusuario, char[] pass) throws LoginException, NoSuchAlgorithmException {
146         String[] grupos = null;
147 
148         if (esUsuarioValido(nombreusuario, pass)) {
149             grupos = buscarGrupos(nombreusuario);
150             grupos = addAssignGroups(grupos);
151         }
152         return grupos;
153     }
```

Figura 6. Método autenticar. Fuente: elaboración propia

A partir de la representación de la Figura 6, la línea 148 hace referencia a una condicional donde se comprueba si el usuario y la contraseña son válidos. Este método comprueba las credenciales a través de una verificación contra base de datos. En caso de ser exitosa se procede a buscar los grupos pertenecientes a ese usuario (línea 149) y asignarlos como roles de autorización. Luego de haber realizado estas operaciones el usuario autenticado despliega sus credenciales en el servidor (roles pertenecientes al usuario) necesarias para la ejecución de las funcionalidades que están encapsuladas en los EJB.

Una vez que el usuario se haya autenticado y tenga desplegadas sus credenciales en el servidor, la petición pasa a ejecutar la funcionalidad que ha sido solicitada. En el caso de la Figura 7 se representa el método *registrarCuentaUsuario* correspondiente al servicio *XegfortNSUsuarioServicie*.

```
24  @WebService(serviceName = "XegfortNSUsuarioServicie")
25  @HandlerChain(file = "XegfortNSTraza_handler.xml")
26  public class XegfortNSUsuarioServicie {
27
28      @EJB
29      private GestorNSUsuarioRemote ejbRef;
30
31      @WebMethod(operationName = "registrarCuentaUsuario_1")
32      @RequestWrapper(className = "registrarCuentaUsuario_1")
33      @ResponseWrapper(className = "registrarCuentaUsuario_1Response")
34      public EDUsuario registrarCuentaUsuario(
35              @WebParam(name = "persona") EDPersona persona,
36              @WebParam(name = "nombreCuenta") String nombreCuenta,
37              @WebParam(name = "contrasena") String contrasena,
38              @WebParam(name = "estado") EDEstadoUsuario estado,
39              @WebParam(name = "cambiarContrasena") boolean cambiarContrasena,
40              @WebParam(name = "idOficina") int idOficina,
41              @WebParam(name = "funcionario") boolean funcionario) throws Exception {
42
43          return ejbRef.registrarCuentaUsuario(persona, nombreCuenta, contrasena,
44                  estado, cambiarContrasena,
45                  idOficina, funcionario);
46      }
```

Figura 7. Método *registrarCuentaUsuario* del Servicio XegfortNSUsuarioServicie. Fuente: elaboración propia

Para poder exponer la clase *XegfortNSUsuarioServicie* como servicio web se hace uso de la anotación[22] *@WebService*, perteneciente a la especificación *JAX-WS* de java para la creación de servicios web. Luego para poder tener acceso a las funcionalidades de los EJB, se hace uso de la inyección de dependencias (representado en la línea 29 de la Figura 7). Posteriormente se especifica a través de la anotación *@WebMethod* que la funcionalidad *registrarCuentaUsuario* es accesible y se identifica con el nombre de *registrarCuentaUsuario*_1 en el contrato o WSDL del servicio. Finalmente, los parámetros del método son anotados con *@WebParam*, especificando el nombre del parámetro y el tipo de dato.

En la línea 43 de la Figura 7 se ejecuta el método de *registrarCuentaUsuario* mediante la variable ejbref que hace referencia al EJB *GestorNSUsuarioRemote* y retorna como resultado la entidad del usuario registrado.

[22] En el caso de la presente investigación las anotaciones representan interfaces que definen atributos y marcan la clase con un propósito.

2.4. Conclusiones parciales

El empleo de la metodología AUP variación UCI en función de describir el proceso de desarrollo del componente de servicios web propuesto, permitió organizar el desarrollo de la solución y generar los artefactos necesarios. Por otra parte, el uso del estilo arquitectónico SOA y el empleo de patrones de diseño garantizaron la obtención de una solución con poca dependencia entre clases, flexible al mantenimiento y a la introducción de cambios. Por tanto, el componente de servicios obtenido extiende arquitectónicamente el marco de trabajo XEGFORT, haciendo posible la integración de este al Sistema de Gobierno Electrónico.

CAPÍTULO 3: VALIDACIÓN DEL COMPONENTE DE SERVICIOS WEB

En el presente capítulo se describe el proceso de validación del componente de servicios propuesto. En la primera parte se muestran los resultados obtenidos en la validación de las clases del diseño aplicando las métricas tamaño operacional de clases y relaciones entre clases. Luego, se describe cómo fue validada la implementación, utilizando la herramienta automatizada SoapUI. Por último, se valida el cumplimiento del objetivo general de la investigación demostrando la relación causa efecto de la variable independiente sobre las dependientes.

3.1. Validación del diseño

Con el objetivo de comprobar el diseño de las clases se aplicaron las métricas de diseño relaciones entre clases (RC) y tamaño operacional de clases (TOC).

3.1.1. Relación entre clases

La métrica RC está dada por el número de relaciones de uso de una clase con otra. El primer paso es evaluar un conjunto de atributos de calidad entre los que se encuentran el acoplamiento, complejidad de mantenimiento y reutilización de cada clase (Pressman, 2002).

A continuación, se exponen los pasos ejecutados al aplicar la métrica a todas las clases del componente de servicios web propuesto en la presente investigación:

1. Determinar la Cantidad de Relaciones de Uso (CRU) que poseen las clases a medir.

2. Calcular el promedio de las CRU.

3. Teniendo en cuenta los valores antes obtenidos se determina la incidencia de los atributos de calidad en cada una de las clases, según los criterios expuestos en la Tabla 8.

Tabla 8. Rango de valores para medir la afectación de los atributos de calidad (RC). Fuente: (Lorenz, y otros, 1994)

Atributos de calidad	Clasificación	Criterio
Acoplamiento	Ninguna	CRU = 0
	Baja	CRU = 1
	Media	CRU = 2
	Alta	CRU > 2
Complejidad de mantenimiento	Baja	CRU <= Promedio
	Media	Promedio < CRU < = 2* promedio
	Alta	CRU > 2* promedio
Reutilización	Baja	CRU > 2* promedio
	Media	Promedio < CRU < = 2* promedio
	Alta	CRU <= Promedio
Cantidad de pruebas	Baja	CRU <= Promedio
	Media	Promedio < CRU < = 2* promedio
	Alta	CRU > 2* promedio

En la Figura 8 se muestra el resultado de aplicar la métrica RC:

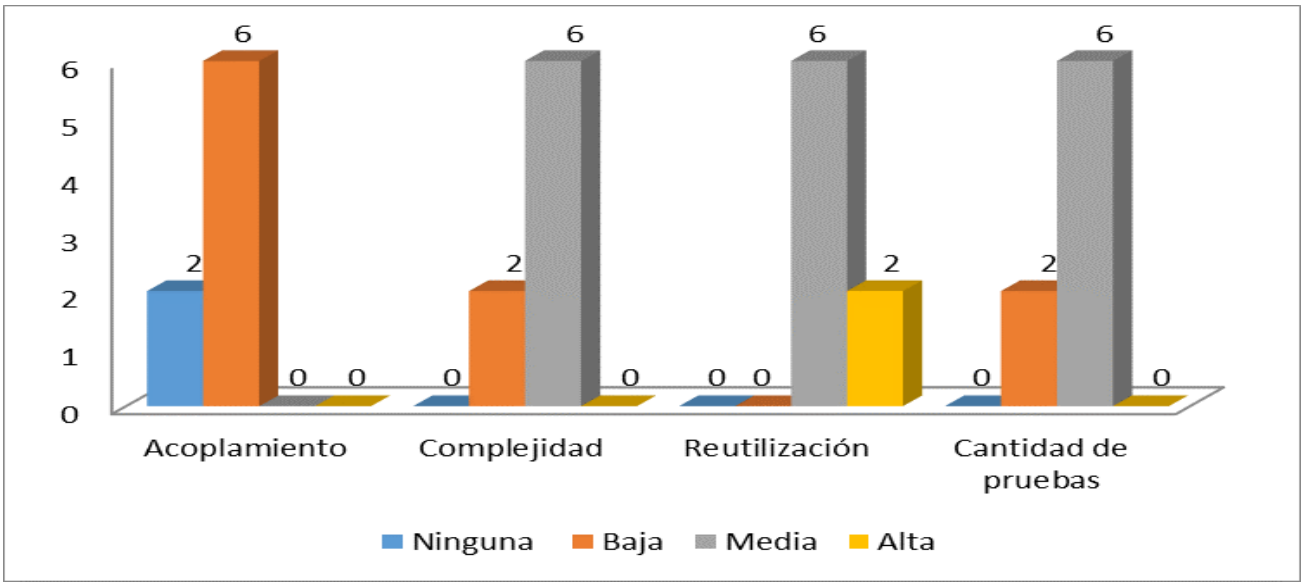

Figura 8. Representación en (%) de los resultados de la aplicación de la métrica RC.
Fuente: elaboración propia

Acoplamiento: según los resultados mostrados en la Figura 8, el 25% de las clases no posee relaciones de uso y el 75 % posee un acoplamiento bajo. Estos datos demuestran que ante futuros cambios que puedan ocurrir en el componente de servicios web las afectaciones a las clases del diseño serían mínimas.

Complejidad de mantenimiento: según los resultados que se muestran en la Figura 8, el 25% de las clases presentan una complejidad de mantenimiento baja y el 75% posee una complejidad media. Estos datos demuestran que, ante la necesidad de soporte al componente de servicios web, el desarrollador puede comprender y ejecutar con facilidad los cambios en las clases de diseño.

Reutilización: según los resultados que se muestran en la Figura 8, el 25% de las clases tiene un grado alto de reutilización y el 75% poseen un grado medio de reutilización. Estos datos demuestran que las clases del diseño son fáciles de utilizar en otro contexto solución.

Cantidad de pruebas: luego de aplicar la métrica se obtuvo que el 25% de las clases poseen un grado bajo de esfuerzo a la hora de realizar cambios, rectificaciones y pruebas de software y un 75% tienen un grado medio. Estos datos demuestran que las clases del diseño presentan dependencias explicitas, están desacopladas y son cohesivas, es decir

existe un control sobre sus dependencias. En lenguaje técnico significa que las clases son *testeables*.

Teniendo en cuenta lo analizado anteriormente, los valores de la métrica RC se comportan satisfactoriamente. Estos resultados demuestran que las clases del diseño del componente de servicios web presentan un acoplamiento bajo, una complejidad de mantenimiento media, la cantidad de pruebas posee una complejidad media y en efecto el grado de reutilización es medio.

3.1.2. Tamaño operacional de clases

La métrica TOC fue aplicada a cada una de las clases del diseño con el objetivo de medir la calidad de las mismas con respecto a su grado de responsabilidad, complejidad de implementación y reutilización (Pressman, 2002).

A continuación, se explican los pasos que se llevaron a cabo para aplicar la métrica:

1. Cálculo del umbral. El umbral se toma del tamaño general de una clase que se determina sumando todas las operaciones que posee.

2. Calcular el promedio de los umbrales.

3. Teniendo en cuenta los valores antes obtenidos se determina la incidencia de los atributos de calidad en cada una de las clases, según los criterios expuestos en la Tabla 9.

Tabla 9. Rango de valores para medir la afectación de los atributos de calidad (TOC). Fuente: (Lorenz, y otros, 1994)

Atributos de calidad	Clasificación	Criterio
Responsabilidad	Baja	Umbral<= Promedio
	Media	Promedio < Umbral < = 2* promedio

	Alta	Umbral > 2* promedio
Complejidad de implementación	Baja	Umbral <= Promedio
	Media	Promedio < Umbral < = 2* promedio
	Alta	Umbral > 2* promedio
Reutilización	Baja	Umbral > 2* promedio
	Media	Promedio < Umbral < = 2* promedio
	Alta	Umbral <= Promedio

En la Figura 9 se muestra el resultado de aplicar la métrica TOC:

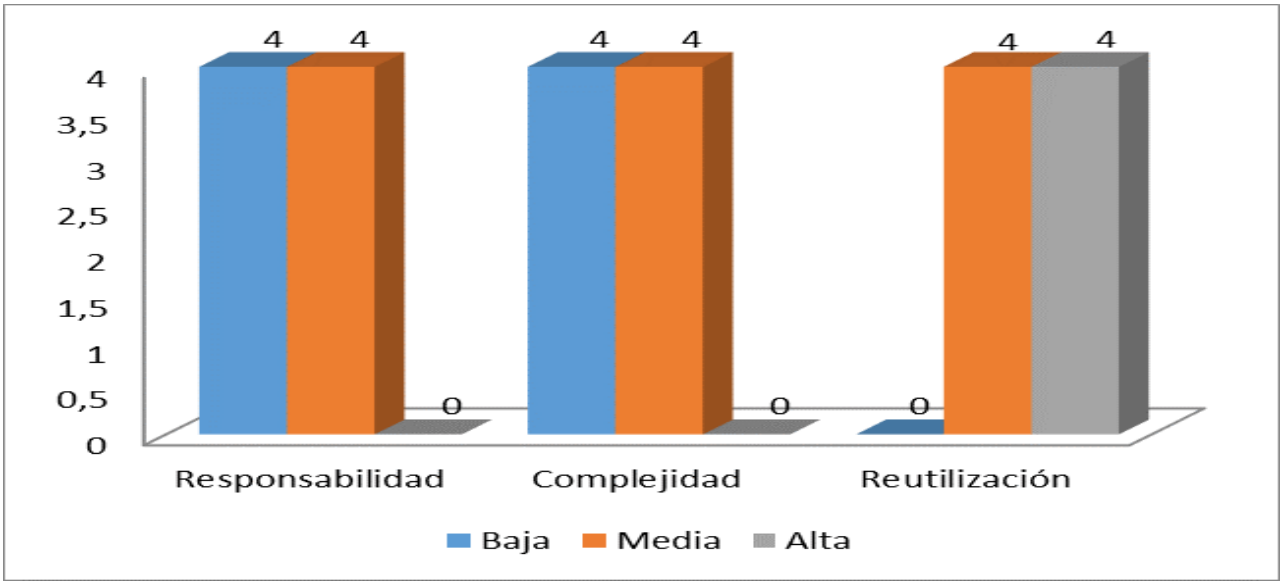

Figura 9. Representación en (%) de los resultados de la aplicación de la métrica TOC. Fuente: elaboración propia.

Responsabilidad: después de aplicar la métrica, se obtuvieron resultados satisfactorios que reflejan que la responsabilidad entre las clases no es alta, al obtenerse un resultado de 50% media y un 50% baja. Los datos anteriores demuestran que para cada clase está asignada la responsabilidad necesaria de acuerdo al objetivo para que fue creada.

Complejidad de implementación: luego de haber realizado la medición de la métrica, se obtuvo que la complejidad de implementación de las clases no es alta, al obtenerse un resultado de 50% de complejidad media y un 50% de complejidad baja. Estos datos demuestran que las clases del diseño presentan una estructura fácil de comprender, lo que implica que como resultado de su implementación, se obtenga un código con bajo acoplamiento y alta cohesión.

Reutilización: se obtuvieron valores que según muestra la gráfica de la Figura 7 la reutilización de las clases no es baja, al obtenerse un resultado de 50% de reutilización media y un 50% de reutilización alta. Estos datos demuestran que las clases del diseño son fáciles de utilizar en otro contexto solución.

Los resultados de aplicar la métrica TOC demuestran que las clases del diseño del componente de servicios web presentan una reutilización, complejidad y responsabilidad de las clases aceptable, teniendo en cuenta el análisis realizado para cada atributo.

3.2. Pruebas

Las pruebas de software comprenden el conjunto de actividades que se realizan para identificar posibles fallos de funcionamiento, configuración o usabilidad de un programa o aplicación, por medio de pruebas sobre el comportamiento del mismo (Sánchez, 2015).

A continuación, se describen las pruebas realizadas para la validación del componente de servicios web propuesto en la presente investigación.

3.2.1. Pruebas de rendimiento

Las pruebas de rendimiento permiten comprobar la capacidad del producto de software para proporcionar apropiados tiempos de respuesta y procesamiento, así como tasas de producción de resultados, al realizar su función bajo condiciones establecidas (ISO/IEC, 2005).

En la actualidad no existen estándares que definan los tiempos de respuesta de los servicios web, sin embargo, se han definido límites de tiempos que hace referencia a las habilidades de la percepción humana para la optimización y rendimiento de aplicaciones. El autor Jakob Nielsen (1993) define en el libro *Usability Engineering* que los tiempos límites de respuesta son los siguientes:

- ✓ 0,1 segundo es aproximadamente el límite para que el usuario sienta que el sistema está reaccionando instantáneamente, lo que significa que no es necesaria ninguna retroalimentación especial excepto para mostrar el resultado.

- ✓ 1,0 segundo es sobre el límite para que el flujo de pensamiento del usuario permanezca ininterrumpido, aunque el usuario notará el retraso. Normalmente, no se necesita retroalimentación especial durante los retrasos de más de 0,1 pero menos de 1,0 segundo, pero el usuario pierde la sensación de operar directamente en los datos.

- ✓ 10 segundos es sobre el límite para mantener la atención del usuario. Para retrasos más largos, los usuarios desearán realizar otras tareas mientras esperan a que finalice la computadora, por lo que deben recibir retroalimentación indicando cuándo la computadora espera que se haga. La retroalimentación durante el retardo es especialmente importante si es probable que el tiempo de respuesta sea muy variable, ya que los usuarios no sabrán cuanto esperar.

Por otra parte, en un estudio realizado por la universidad de Cambridge, MA en noviembre de 2006 se define que cuatro segundos es el tiempo máximo que un usuario promedio debe esperar por la respuesta de un sistema (Cambridge, 2006).

En el caso de la validación del componente de servicios web propuesto en la presente investigación, el desarrollo de las pruebas de rendimiento se realiza utilizando la herramienta automatizada SoapUI. A continuación, se muestran los resultados de la ejecución de esta prueba.

Al realizar una petición el tiempo de respuesta obtenido es de 2260 milisegundos (ms) como se muestra en la Figura 10.

Figura 10. Resultado de ejecución de las pruebas de rendimiento para una petición.
Fuente: elaboración propia.

Al realizar varias peticiones simulando un total de 5 computadoras durante 10 segundos, existiendo una diferencia de un segundo entre cada petición, el promedio de tiempo de respuesta obtenido es de 2545 ms como se muestra en la Figura 11.

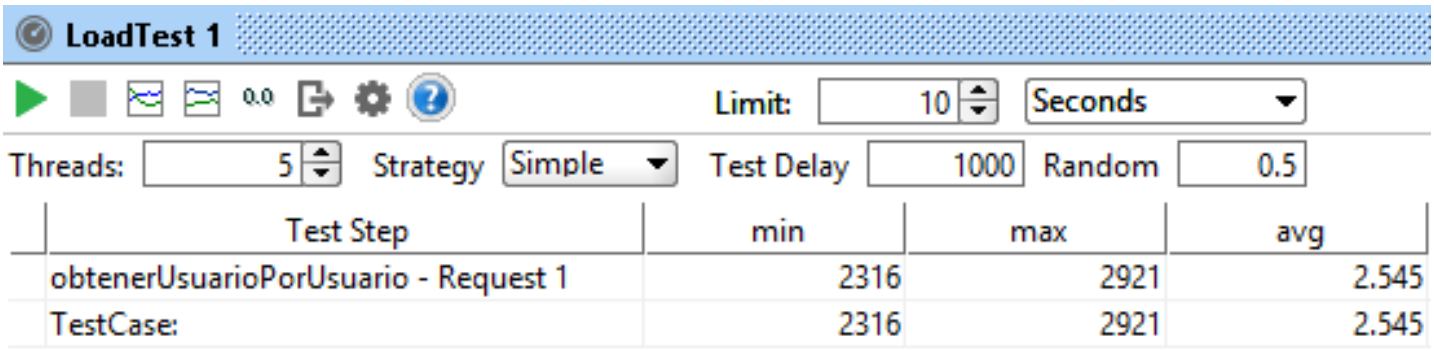

Figura 11. Resultado de ejecución de las pruebas de rendimiento para varias peticiones.
Fuente: elaboración propia.

Los resultados obtenidos con el desarrollo de las pruebas de rendimiento demuestran que los tiempos de respuesta promedio de las operaciones de los servicios pertenecientes al componente desarrollado son aceptables, teniendo en cuenta los límites de tiempo definidos por Jakob Nielsen (1993) y el estudio realizado en la universidad de Cambridge, MA.

3.2.2. Pruebas funcionales

Las pruebas funcionales son pruebas diseñadas tomando como referencia las especificaciones funcionales de un componente o sistema. Se realizan para comprobar si el software cumple las funciones esperadas (Pressman, 2007).

En el caso de la validación del componente de servicios web propuesto en la presente investigación, el desarrollo de las pruebas funcionales se realiza utilizando la herramienta automatizada *SoapUI*, se ejecutaron en un total de dos iteraciones. Las no conformidades (NC) identificadas en la primera iteración fueron de tipo validación, el número de estas se muestra en la Figura 12, las cuales fueron resultas en el tiempo establecido. En una segunda iteración no se obtuvieron NC, concluyendo así el desarrollo de la prueba. Estas pruebas fueron supervisadas por los especialistas del grupo de calidad de CEGEL, generándose así el Acta de liberación interna de productos de software (ver Anexo 4**:** Acta de liberación interna de productos de software.).

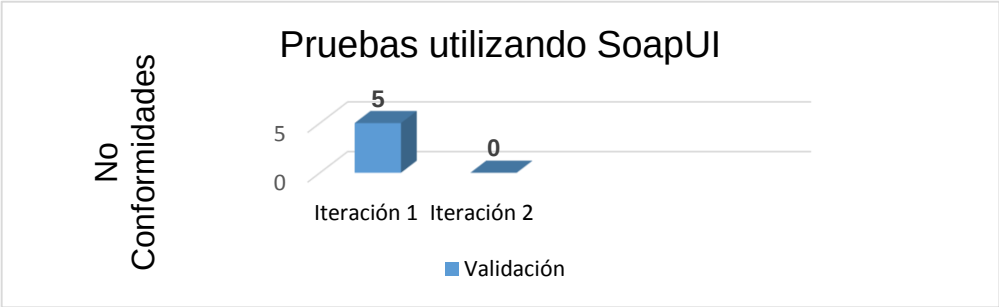

Figura 12. Relación de no conformidades por cada iteración. Fuente: elaboración propia.

A continuación, en las figuras 13 a la 16 se muestra un ejemplo donde se evidencia el uso de la herramienta automatizada sobre la operación autenticar perteneciente al servicio *XegfortNSUsuarioServicie*, mostrando los resultados que el servicio devuelve para el escenario inválido.

Servicio *XegfortNSUsuarioServicie*

Funcionalidad: autenticar

Escenario: Inválido

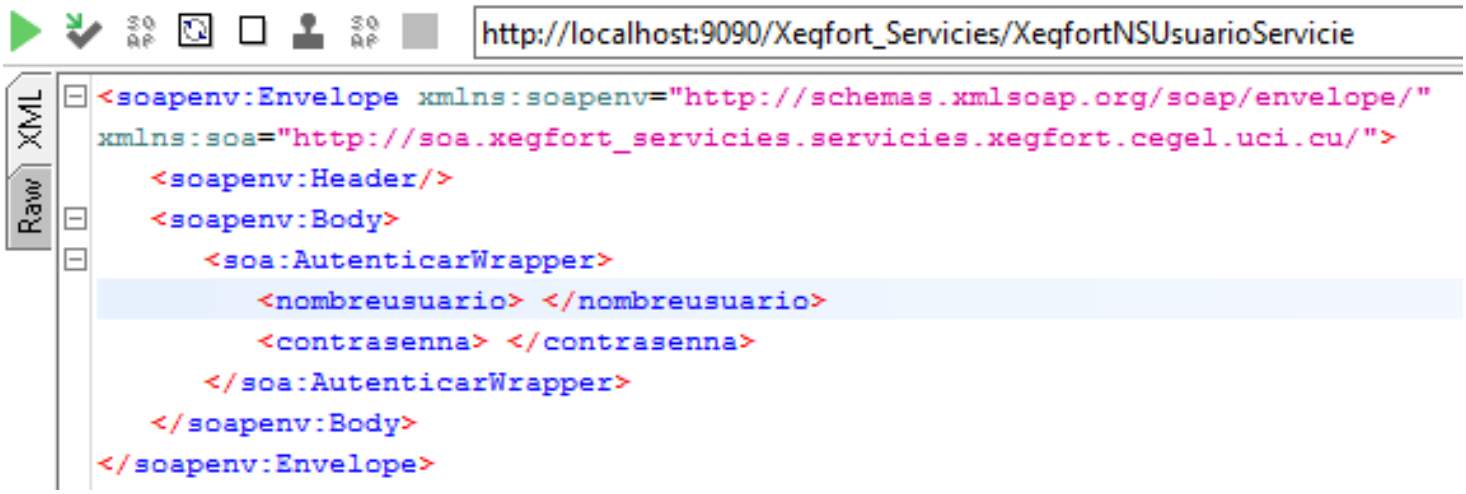

Figura 13. Funcionalidad autenticar usuario con parámetros vacíos. Fuente: elaboración propia

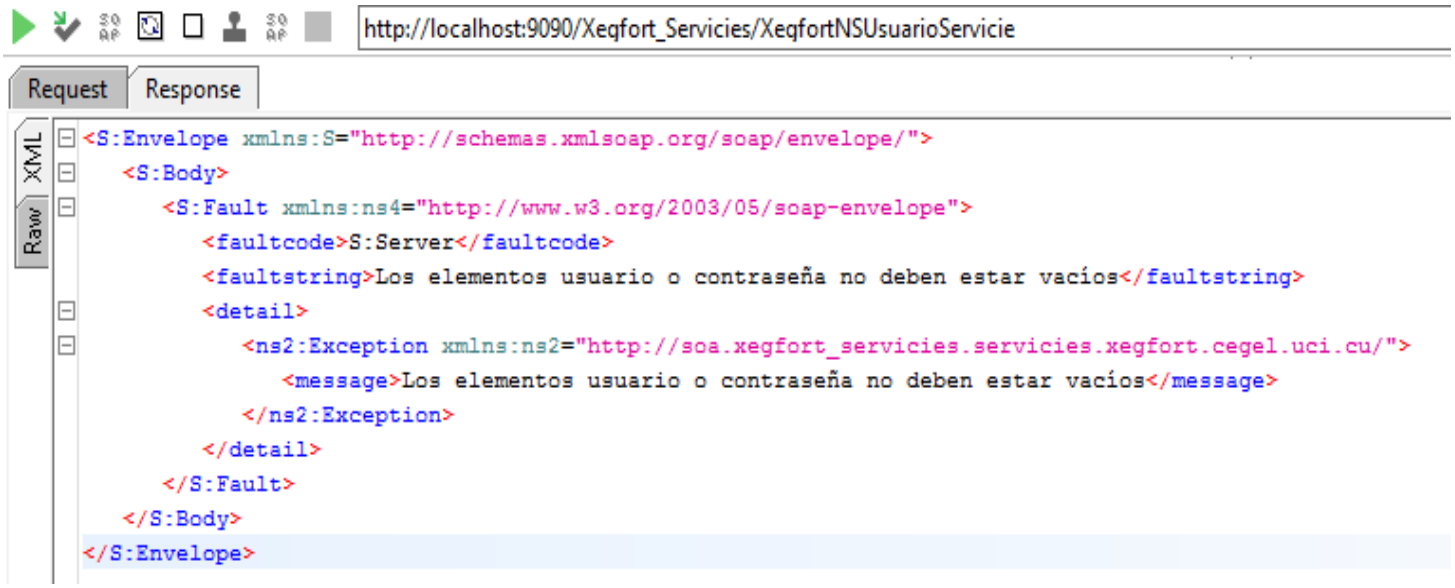

Figura 14. Resultado de la funcionalidad autenticar con parámetros vacíos. Fuente: elaboración propia

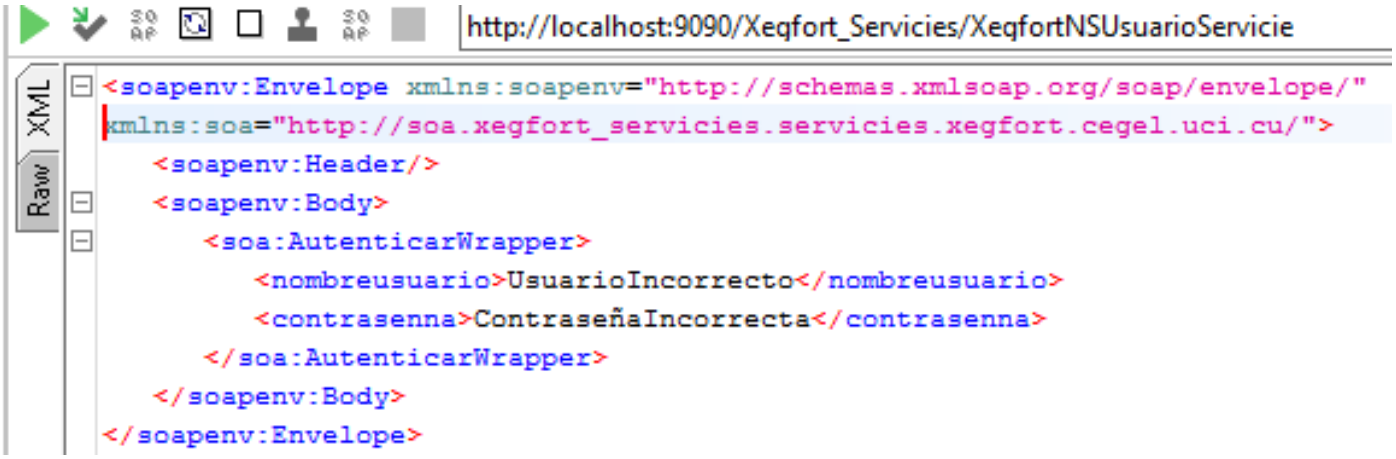

Figura 15. Funcionalidad autenticar con parámetros incorrectos. Fuente: elaboración propia.

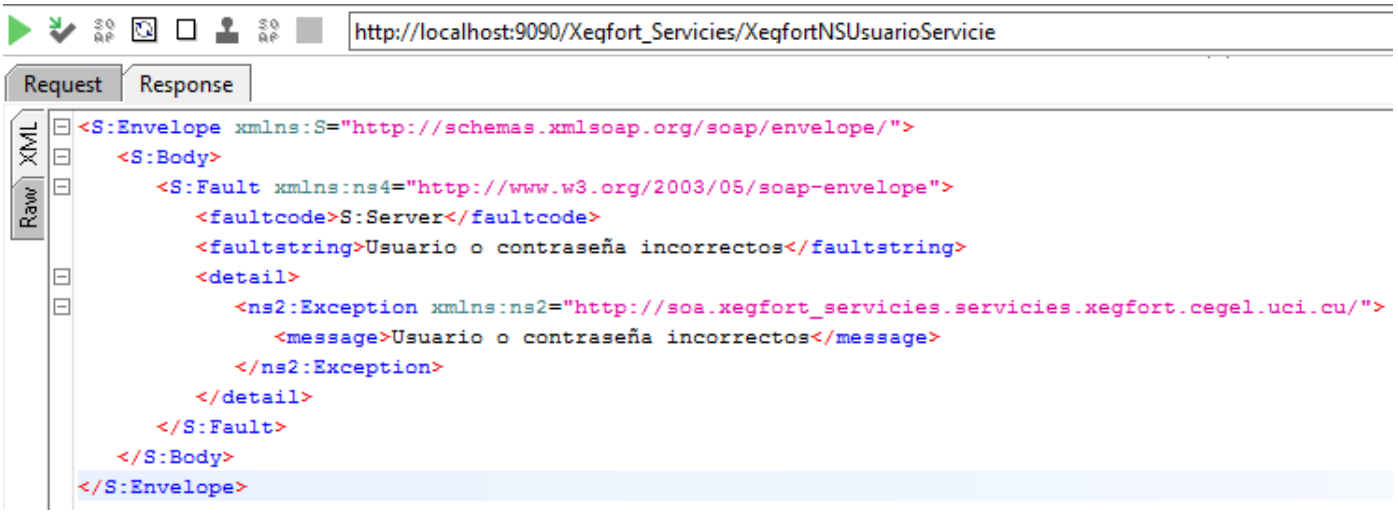

Figura 16. Resultado de la funcionalidad autenticar con parámetros incorrectos. Fuente: elaboración propia

3.3. Validación de los resultados de la solución

Teniendo en cuenta que en la investigación realizada se define como idea a defender: “el desarrollo de un componente de servicio para la integración del marco de trabajo XEGFORT con soluciones informáticas del Sistema de Gobierno Electrónico, contribuye a la interoperabilidad técnica entre los sistemas desarrollados en CEGEL”. A continuación, se analiza la relación causa efecto entre la variable independiente “integración del marco de trabajo XEGFORT con soluciones informáticas del Sistema de Gobierno Electrónico” y la variable dependiente “la interoperabilidad técnica de los sistemas desarrollados en CEGEL”.

El autor de la presente investigación realiza la comprobación de las variables utilizando un conjunto de indicadores relacionado con la dimensión técnica de la interoperabilidad definida por Comisión Europea en el marco del programa *Interoperable Delivery of European eGovernment Services to Public Administrations, Business and Citizens* (IDABC) en el sitio web Red de Transparencia y Acceso a la Información (2004). La decisión de utilizar estos indicadores se realiza teniendo en cuenta que el componente de servicios web desarrollado no está dirigido a un dominio en específico, por tanto, no es necesario aplicar el resto de las dimensiones que conforman la interoperabilidad (semántica y organizativa), reconocidas también por la IDABC. Los indicadores seleccionados son los siguientes:

- **Estándares abiertos:** garantizan y facilitan la interoperabilidad. Se desarrollan en un proceso transparente y de colaboración, están disponibles de forma gratuita, o con un costo razonable. Además, según estudios recientes, se recomienda su uso para alcanzar buenas prácticas en el ámbito de la administración electrónica (Friedrich y Undheim, 2008). Algunos ejemplos de estándares abiertos son: XML, TCP/IP23, HTTP, HTML, POP3 24 u ODF25.

- **Arquitecturas acordes a las soluciones:** hace referencia a la importancia de diseñar una estrategia que se dirija a la obtención de arquitecturas flexibles que permitan intercambiar la información e integren servicios de origen distinto, para permitir el desarrollo de servicios completos (RTA, 2017).

- **Integración:** debe dirimir sobre circunstancias técnicas como los procesos de trabajo, las hipotéticas incompatibilidades tecnológicas o la falta de definición en el uso de los datos (RTA, 2017).

A continuación, en la Tabla 10, se evalúan cada uno de los indicadores seleccionados. La evaluación se realiza teniendo en cuenta la definición de un cuasi experimento de tipo pre prueba-post prueba, a través del cual, estableciendo un antes (cuando al marco de trabajo no contaba con el componente de servicios web propuesto) y un después (cuando al marco de trabajo se le incluye el componente de servicios web propuesto) se demuestra cómo a través del desarrollo del componente de servicios web propuesto, el marco de

[23] Protocolo de Control de Transmisión/Protocolo de Internet
[24] Protocolo de Oficina de Correo o Protocolo de Oficina Postal
[25] Archivo de definición de objeto

trabajo XEGFORT se integra al Sistema de Gobierno Electrónico, contribuyendo a la interoperabilidad técnica entre los sistemas desarrollados en CEGEL.

Tabla 10. Validación de las variables de investigación. Fuente: elaboración propia

Indicadores seleccionados	Antes	Después
Estándares abiertos	Teniendo en cuenta que solo se utilizaba el protocolo RMI no se hacía uso de estándares abiertos para el intercambio de datos que posibilitara la interoperabilidad técnica entre sistemas.	A partir del desarrollo del componente de servicios web se utiliza el estándar abierto XML para el intercambio de mensajes SOAP, permitiendo unificar el formato de los mensajes contribuyendo a la interoperabilidad técnica entre sistemas.
Arquitecturas acordes a las soluciones	El uso del protocolo RMI no permite el intercambio de información entre servicios de distintos orígenes, afectando la flexibilidad de la arquitectura del marco de trabajo XEGFORT.	Con el desarrollo del componente de servicios web propuesto se logra el intercambio de información entre servicios de distintos orígenes, contribuyendo a la mejora en la flexibilidad de la arquitectura del marco de trabajo XEGFORT.
Integración	El uso de RMI no permite la compatibilidad tecnológica entre las soluciones informáticas que forman parte del Sistema de Gobierno Electrónico, de	La aplicación del componente de servicios web propuesto hace posible la compatibilidad tecnológica entre las soluciones informáticas

	forma simple.	que forman parte del Sistema de Gobierno Electrónico, permitiendo que estas interoperen de forma simple.

La comparación realizada en la tabla anterior, a través de los indicadores seleccionados, valida cómo utilizando el componente de servicios web propuesto en la presente investigación, se contribuye a la interoperabilidad técnica entre los sistemas desarrollados en CEGEL. Lo anterior se demuestra teniendo en cuenta que el marco de trabajo XEGFORT, al contar con el componente desarrollado en la presente investigación, incluye el estándar abierto XML para el intercambio de datos entre aplicaciones, flexibiliza su arquitectura permitiendo integrarse con sistemas de distintas tecnologías y por tanto incorpora los elementos necesarios para convertirse en el marco de trabajo base para el sistema de Gobierno Electrónico.

3.4. Conclusiones parciales

La aplicación de métricas de validación del diseño y el desarrollo de pruebas automatizadas a la solución propuesta, corroboran la obtención de un componente de servicios web funcional, avalada por el acta de liberación emitida por el Grupo de Calidad CEGEL. Por otra parte, la validación del resultado de la investigación, demuestra el cumplimiento de la relación causa efecto de la variable independiente “integración del marco de trabajo XEGFORT con soluciones informáticas del Sistema de Gobierno Electrónico” sobre la variable dependiente “la interoperabilidad técnica entre los sistemas desarrollados en CEGEL.

CONCLUSIONES GENERALES

El análisis de los mecanismos de integración existentes, permitió definir los servicios web como alternativa viable para el logro de la integración entre las soluciones que forman parte del Sistema de Gobierno Electrónico.

La aplicación de las disciplinas requisitos, análisis y diseño e implementación, hicieron posible definir los RF a exponer como servicios web; utilizar patrones de diseño que garantizan una solución con poca dependencia entre clases, flexible al mantenimiento y a la introducción de cambios; así como la obtención de un componente de servicios que extiende arquitectónicamente el marco de trabajo XEGFORT.

El uso la herramienta automatizada SOAPUI en el desarrollo de pruebas funcionales, corrobora la validez y utilidad del componente de servicios web propuesto, mientras que la ejecución de la prueba de rendimiento utilizando esta misma herramienta, demuestra el cumplimiento del RnF de eficiencia definido en la disciplina de requisitos.

Con la validación de las variables de la investigación, se demuestra que la integración del marco de trabajo XEGFORT con soluciones informáticas del Sistema de Gobierno Electrónico contribuye a la interoperabilidad técnica entre los sistemas desarrollados en CEGEL.

RECOMENDACIONES

Implementar un sistema de *log* que permita que el componente de servicios web sea auditable.

Integrar un componente orientado a eventos para responder a peticiones asíncronas.

Exponer en el componente de servicios web otras funcionalidades del marco de trabajo XEGFORT.

BIBLIOGRAFÍA REFERENCIADA

adictosaltrabajo. 2009. adictosaltrabajo. *adictosaltrabajo.* [En línea] 2009. https://www.adictosaltrabajo.com/tutoriales/mysql-replicacion/.

AGESIC, Agencia de Gobierno Electrónico y Sociedad de la Información y del Conocimiento. 2017. agesic. *agesic.* [En línea] 2017. https://www.agesic.gub.uy/innovaportal/v/163/1/agesic/gobierno-electronico-.html.

andreshevia. 2015. andreshevia. *andreshevia.* [En línea] 2015. https://andreshevia.com/2015/02/01/patrones-de-diseno-en-soa/.

Atwood, Jeff. 2009. XML: The Angle Bracket Tax. [En línea] 2009. https://blog.codinghorror.com/xml-the-angle-bracket-tax.

Beck, Kent. 1999. *Extreme Programming Explained.* 1999.

Cambridge, MA. 2006. akamai. [En línea] 2006. http://www.akamai.com/html/about/press/releases/2006/press_110606.html.

Carrasco Puebla, y otros. 2009. GestioPolis. [En línea] 2009. http://www.gestiopolis.com/administracion-estrategia/erp- arquitectura-orientada-a-servicios.htm..

conceptodefinicion. 2017. conceptodefinicion. *conceptodefinicion.* [En línea] 2017. http://conceptodefinicion.de/metodologia-de-desarrollo-de-software/.

eclipse. eclipse. *eclipse.* [En línea] https://eclipse.org/mars/.

ecplise. 2017. ecplise. *ecplise.* [En línea] 2017. https://eclipse.org/mars/.

emmersonmiranda. 2015. emmersonmiranda. *emmersonmiranda.* [En línea] 2015. http://www.emmersonmiranda.net.

Fergarciac. 2013. Fergarciac. *Fergarciac.* [En línea] 2013. http://fergarciac.wordpress.com/2013/01/25/entorno-de-desarrollointegrado-ide/.

Friedrich y Undheim. **Interoperability., The Momentum of Open Standards: a Pragmatic Approach to Software. 2008.** 2008, European Journal of ePractice, págs. pp. 1-13.

Hommel, Scott. 2007. *Convenciones de código para el lenguaje de programación Java.* s.l. : Sun Microsystems Inc., 2007.

IBM. 2017. IBM. *IBM.* [En línea] 2017. https://www.ibm.com/developerworks/ssa/webservices/newto/service.html.

—. 2006. IBM. *IBM.* [En línea] 2006. www.ibm.com/developerworks/webservices/library/ws-soa-eda-esb/index.html.

IEEE. 1999. IEEE Advancing Technology for Humanity. *IEEE Advancing Technology for Humanity.* [En línea] 1999. http://www.ieee.org.

ISO/IEC, 9126-1. 2005. *Parte 1: Modelo de calidad, Ingeniería de Software-Calidad del producto.* 2005.

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000. *El Proceso Unificado de Desarrollo de Software.* 2000.

Larman, Craig. 2005. *UML y Patrones 2da Edición.* 2005.

Lorenz, Mark y Kidd, Jeff. 1994. *Object-oriented software metrics: a practical guide. .* Nueva Jersey : Prentice-Hall, 1994.

Oracle. 2013. Oracle. *Oracle.* [En línea] 2013. http://www.oracle.com/technetwork/java.

powerdata. 2015. powerdata. *powerdata.* [En línea] 2015. http://blog.powerdata.es/el-valor-de-la-gestion-de-datos/bid/405060/qu-significa-la-integraci-n-de-datos.

Pressman, Roger. 2007. *Ingenniería del software. Un enfoque práctico.* 2007.

Prince, Steven. 2014. Pruebas de software. [En línea] 2014. http://pruebadesoftwareiut.blogspot.com/2014/04/normal-0-21-false-false-false-es-ve-x.html.

Producción, Dirección Técnica. 2011. Estándar de codificación para Java. *Estándar de codificación para Java.* La Habana,Cuba : s.n., 2011.

QODE. 2013. QODE. *QODE.* [En línea] 2013. http://qode.pro/blog/tecnologia-apps/.

RTA. 2017. Red de Transparencia y Acceso a la información. [En línea] 2017. http://mgd.redrta.org/directrices-interoperabilidad/mgd/2015-01-23/112234.html.

Sánchez, Tamara Rodríguez. 2015. Metodología de desarrollo para la Actividad productiva de la UCI. [aut. libro] UCI. *Metodología de desarrollo para la Actividad productiva de la UCI.* 2015.

SmartBear. 2017. SoapUI. [En línea] 2017. https://www.soapui.org/.

Sommerville, Ian. 2005. *Ingeniería del software. Séptima Edición.* Madrid. España : Pearson Educación. S. A., 2005. 84-7829-074-5.

Soulary, Beyris y Liliam, Celia. 2010. *Proceso de medición y análisis para el polo de hardware y automática.* s.l. : Vol 3, 2010.

tcpsi. 2017. tcpsi. [En línea] 2017. http://www.tcpsi.com/vermas/integracion_esb.htm.

Troelsen, Andrew. 2010. *Pro C# 2010 and the.NET 4 Platform.* 5. 2010. pág. pág. 1014.

W3C. 2012. W3C. *W3C.* [En línea] 2012. https://www.w3.org/TR/xmlschema11-1/#intro-purpose.

—. 2013. W3C. *W3C.* [En línea] 2013. (http://www.w3.org/TR/REC-xml/.

ANEXOS

Anexo 1: Historias de usuario

Número: 3	Nombre del requisito: Buscar cuenta de usuario
Programador: : Luis Javier Rodríguez Castro	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 13 horas
	Tiempo Real: 18 horas
Descripción: Una vez autenticado en el sistema y en dependencia del rol al que pertenece el usuario pues procederá a realizar búsquedas de cuentas de usuarios en dependencia de los criterios de búsquedas que necesite la operación.	

Número: 4	Nombre del requisito: Adicionar rol
Programador: Luis Javier Rodríguez Castro	Iteración Asignada: 2
Prioridad: Alta	Tiempo Estimado: 20 horas
	Tiempo Real: 23 horas
Descripción: Una vez autenticado en el sistema y en dependencia del rol al que pertenece el usuario pues procederá adicionar un nuevo rol en el sistema el cuál formará parte de los roles de sistema que se podrá asignar a un usuario posteriormente.	

Número: 5	Nombre del requisito: Modificar rol
Programador: Luis Javier Rodríguez Castro	Iteración Asignada: 2

Prioridad: Alta	Tiempo Estimado: 20 horas
	Tiempo Real: 23 horas
Descripción: Una vez autenticado en el sistema y en dependencia del rol al que pertenece el usuario pues procederá a la modificación de un rol especificado por parámetro en el sistema.	

Número: 6	Nombre del requisito: Eliminar rol
Programador: Luis Javier Rodríguez Castro	Iteración Asignada: 2
Prioridad: Alta	Tiempo Estimado: 20 horas
	Tiempo Real: 21 horas
Descripción: Una vez autenticado en el sistema y en dependencia del rol al que pertenece el usuario pues procederá a eliminar un rol pasado por parámetro a la operación.	

Número: 7	Nombre del requisito: Buscar roles
Programador: Luis Javier Rodríguez Castro	Iteración Asignada: 2
Prioridad: Alta	Tiempo Estimado: 15 horas
	Tiempo Real: 18
Descripción: Una vez autenticado en el sistema y en dependencia del rol al que pertenece el usuario pues procederá a la búsqueda de roles en el sistema en dependencia del parámetro de búsqueda.	

Número: 8	Nombre del requisito: Adicionar traza
Programador: Luis Javier Rodríguez Castro	Iteración Asignada: 3
Prioridad: Alta	Tiempo Estimado: 15 horas
	Tiempo Real: 18
Descripción: Una vez autenticado en el sistema y en dependencia del rol al que pertenece el usuario pues procederá adicionar las trazas de los usuarios con su respectiva descripción en el sistema.	

Número: 9	Nombre del requisito: Buscar trazas
Programador: Luis Javier Rodríguez Castro	Iteración Asignada: 3
Prioridad: Alta	Tiempo Estimado: 20 horas
	Tiempo Real: 22 horas
Descripción: Una vez autenticado en el sistema y en dependencia del rol al que pertenece el usuario pues procederá a realizar búsquedas de las trazas en el sistema en dependencias de los criterios de búsquedas especificados por parámetro.	

Número: 10	Nombre del requisito: Deshabilitar sesión activa del sistema
Programador: Luis Javier Rodríguez Castro	Iteración Asignada: 4
Prioridad: Alta	Tiempo Estimado: 15
	Tiempo Real: 20

Descripción: Se deshabilita una sesión activa del sistema esto trae consigo que para realizar una acción el sistema tendrá que autenticarse de nuevo para crear una nueva sesión en el sistema.

Número: 11	Nombre del requisito: Buscar sesiones activas del sistema
Programador: Luis Javier Rodríguez Castro	Iteración Asignada: 4
Prioridad: Alta	Tiempo Estimado: 14
	Tiempo Real: 18
Descripción: Una vez autenticado en el sistema y en dependencia del rol al que pertenece el usuario pues procederá a buscar las sesiones activas en el sistema según los parámetros de búsquedas especificados.	

Anexo 2: Contrato o documento WSDL.

```
<?xml version='1.0' encoding='UTF-8'?><!-- Published by JAX-WS RI
(http://jax-ws.java.net). RI's version is Metro/2.3.1-b419
(branches/2.3.1.x-7937; 2014-08-04T08:11:03+0000) JAXWS-RI/2.2.10-
b140803.1500 JAXWS-API/2.2.11 JAXB-RI/2.2.10-b140802.1033 JAXB-
API/2.2.12-b140109.1041 svn-revision#unknown. --><!-- Generated by JAX-WS
RI (http://jax-ws.java.net). RI's version is Metro/2.3.1-b419
(branches/2.3.1.x-7937; 2014-08-04T08:11:03+0000) JAXWS-RI/2.2.10-
b140803.1500 JAXWS-API/2.2.11 JAXB-RI/2.2.10-b140802.1033 JAXB-
API/2.2.12-b140109.1041 svn-revision#unknown. --><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://soa.xegfort_servicies.servicies.xegfort.cegel.uci.cu/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://soa.xegfort_servicies.servicies.xegfort.cegel.uci
.cu/" name="XegfortNSUsuarioServicie">
    <types>
        <xsd:schema>
            <xsd:import
namespace="http://soa.xegfort_servicies.servicies.xegfort.cegel.uci.cu/"
schemaLocation="XegfortNSUsuarioServicie.xsd_1.xsd"/>
        </xsd:schema>
    </types>
    <message name="autenticar_1">
        <part name="parameters" element="tns:Autenticar_1Wrapper"/>
    </message>
    <message name="autenticar_1Response">
        <part name="parameters" element="tns:autenticar_1Response"/>
    </message>
    <message name="Exception">
        <part name="fault" element="tns:Exception"/>
    </message>
    <message name="autenticar">
        <part name="parameters" element="tns:AutenticarWrapper"/>
    </message>
    <message name="autenticarResponse">
        <part name="parameters" element="tns:autenticarResponse"/>
    </message>
    <message name="registrarCuentaUsuario_1">
        <part name="parameters" element="tns:registrarCuentaUsuario_1"/>
    </message>
    <message name="registrarCuentaUsuario_1Response">
        <part name="parameters"
element="tns:registrarCuentaUsuario_1Response"/>
    </message>
    <message name="obtenerUsuarioPorUsuario">
        <part name="parameters"
element="tns:ObtenerUsuarioPorUsuarioWrapper"/>
    </message>
    <message name="obtenerUsuarioPorUsuarioResponse">
```

```
        <part name="parameters"
element="tns:obtenerUsuarioPorUsuarioResponse"/>
    </message>
    <message name="SOAPException">
        <part name="fault" element="tns:SOAPException"/>
    </message>
    <portType name="XegfortNSUsuarioServicie">
        <operation name="autenticar_1">
            <input wsam:Action="autenticar_1Action"
message="tns:autenticar_1"/>
            <output
wsam:Action="http://soa.xegfort_servicies.servicies.xegfort.cegel.uci.cu/
XegfortNSUsuarioServicie/autenticar_1Response"
message="tns:autenticar_1Response"/>
            <fault message="tns:Exception" name="Exception"
wsam:Action="http://soa.xegfort_servicies.servicies.xegfort.cegel.uci.cu/
XegfortNSUsuarioServicie/autenticar_1/Fault/Exception"/>
        </operation>
        <operation name="autenticar">
            <input wsam:Action="autenticarAction"
message="tns:autenticar"/>
            <output
wsam:Action="http://soa.xegfort_servicies.servicies.xegfort.cegel.uci.cu/
XegfortNSUsuarioServicie/autenticarResponse"
message="tns:autenticarResponse"/>
            <fault message="tns:Exception" name="Exception"
wsam:Action="http://soa.xegfort_servicies.servicies.xegfort.cegel.uci.cu/
XegfortNSUsuarioServicie/autenticar/Fault/Exception"/>
        </operation>
        <operation name="registrarCuentaUsuario_1">
            <input wsam:Action="registrarCuentaUsuario_1Action"
message="tns:registrarCuentaUsuario_1"/>
            <output
wsam:Action="http://soa.xegfort_servicies.servicies.xegfort.cegel.uci.cu/
XegfortNSUsuarioServicie/registrarCuentaUsuario_1Response"
message="tns:registrarCuentaUsuario_1Response"/>
            <fault message="tns:Exception" name="Exception"
wsam:Action="http://soa.xegfort_servicies.servicies.xegfort.cegel.uci.cu/
XegfortNSUsuarioServicie/registrarCuentaUsuario_1/Fault/Exception"/>
        </operation>
        <operation name="obtenerUsuarioPorUsuario">
            <input wsam:Action="obtenerUsuarioPorUsuarioAction"
message="tns:obtenerUsuarioPorUsuario"/>
            <output
wsam:Action="http://soa.xegfort_servicies.servicies.xegfort.cegel.uci.cu/
XegfortNSUsuarioServicie/obtenerUsuarioPorUsuarioResponse"
message="tns:obtenerUsuarioPorUsuarioResponse"/>
            <fault message="tns:SOAPException" name="SOAPException"
wsam:Action="http://soa.xegfort_servicies.servicies.xegfort.cegel.uci.cu/
XegfortNSUsuarioServicie/obtenerUsuarioPorUsuario/Fault/SOAPException"/>
            <fault message="tns:Exception" name="Exception"
wsam:Action="http://soa.xegfort_servicies.servicies.xegfort.cegel.uci.cu/
XegfortNSUsuarioServicie/obtenerUsuarioPorUsuario/Fault/Exception"/>
        </operation>
    </portType>
    <binding name="XegfortNSUsuarioServiciePortBinding"
type="tns:XegfortNSUsuarioServicie">
```

```
        <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
style="document"/>
        <operation name="autenticar_1">
            <soap:operation soapAction="autenticar_1Action"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
            <fault name="Exception">
                <soap:fault name="Exception" use="literal"/>
            </fault>
        </operation>
        <operation name="autenticar">
            <soap:operation soapAction="autenticarAction"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
            <fault name="Exception">
                <soap:fault name="Exception" use="literal"/>
            </fault>
        </operation>
        <operation name="registrarCuentaUsuario_1">
            <soap:operation soapAction="registrarCuentaUsuario_1Action"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
            <fault name="Exception">
                <soap:fault name="Exception" use="literal"/>
            </fault>
        </operation>
        <operation name="obtenerUsuarioPorUsuario">
            <soap:operation soapAction="obtenerUsuarioPorUsuarioAction"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
            <fault name="SOAPException">
                <soap:fault name="SOAPException" use="literal"/>
            </fault>
            <fault name="Exception">
                <soap:fault name="Exception" use="literal"/>
            </fault>
        </operation>
    </binding>
    <service name="XegfortNSUsuarioServicie">
        <port name="XegfortNSUsuarioServiciePort"
binding="tns:XegfortNSUsuarioServiciePortBinding">
```

```
            <soap:address
location="http://localhost:9090/Xegfort_Servicies/XegfortNSUsuarioServici
e"/>
        </port>
    </service>
</definitions>
```

Anexo 3: Esquema o XSD del servicio.

```
<?xml version='1.0' encoding='UTF-8'?><!-- Published by JAX-WS RI
(http://jax-ws.java.net). RI's version is Metro/2.3.1-b419
(branches/2.3.1.x-7937; 2014-08-04T08:11:03+0000) JAXWS-RI/2.2.10-
b140803.1500 JAXWS-API/2.2.11 JAXB-RI/2.2.10-b140802.1033 JAXB-
API/2.2.12-b140109.1041 svn-revision#unknown. --><xs:schema
xmlns:tns="http://soa.xegfort_servicies.servicies.xegfort.cegel.uci.cu/"
xmlns:xs="http://www.w3.org/2001/XMLSchema" version="1.0"
targetNamespace="http://soa.xegfort_servicies.servicies.xegfort.cegel.uci
.cu/">

    <xs:element name="AutenticarWrapper" nillable="true"
type="tns:autenticarWrapper"/>

    <xs:element name="Autenticar_1Wrapper" nillable="true"
type="tns:autenticar1Wrapper"/>

    <xs:element name="Exception" type="tns:Exception"/>

    <xs:element name="ObtenerUsuarioPorUsuarioWrapper" nillable="true"
type="tns:obtenerUsuarioPorUsuarioWrapper"/>

    <xs:element name="SOAPException" type="tns:SOAPException"/>

    <xs:element name="autenticarResponse" nillable="true"
type="tns:autenticar_1Response"/>

    <xs:element name="autenticar_1Response"
type="tns:autenticar_1Response"/>

    <xs:element name="obtenerUsuarioPorUsuarioResponse"
type="tns:obtenerUsuarioPorUsuarioResponse"/>

    <xs:element name="registrarCuentaUsuario_1"
type="tns:registrarCuentaUsuario_1"/>

    <xs:element name="registrarCuentaUsuario_1Response"
type="tns:registrarCuentaUsuario_1Response"/>

    <xs:complexType name="registrarCuentaUsuario_1">
        <xs:sequence>
            <xs:element name="persona" type="tns:edPersona" />
            <xs:element name="nombreCuenta" type="xs:string"/>
            <xs:element name="contrasena" type="xs:string" />
            <xs:element name="estado" type="tns:edEstadoUsuario" />
            <xs:element name="cambiarContrasena" type="xs:boolean" />
            <xs:element name="idOficina" type="xs:int"/>
            <xs:element name="funcionario" type="xs:boolean"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="edPersona">
        <xs:sequence>
            <xs:element name="fechaNacimiento" type="xs:dateTime"
minOccurs="0"/>
```

```
            <xs:element name="idPersona" type="xs:int"/>
            <xs:element name="numeroCedula" type="xs:string"
minOccurs="0"/>
            <xs:element name="numeroIdentificacion" type="xs:string"
minOccurs="0"/>
            <xs:element name="primerApellido" type="xs:string"/>
            <xs:element name="primerNombre" type="xs:string"/>
            <xs:element name="segundoApellido" type="xs:string"
minOccurs="0"/>
            <xs:element name="segundoNombre" type="xs:string"
minOccurs="0"/>
            <xs:element name="sexo" type="xs:unsignedShort"
minOccurs="0"/>
            <xs:element name="tipoDocumento" type="xs:string"
minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="edEstadoUsuario">
        <xs:sequence>
            <xs:element name="activo" type="xs:boolean"/>
            <xs:element name="estado" type="xs:boolean"/>
            <xs:element name="idEstadoUsuario" type="xs:int"/>
            <xs:element name="nombre" type="xs:string" minOccurs="0"/>
            <xs:element name="usuarios" type="tns:edUsuario"
nillable="true" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="edUsuario">
        <xs:complexContent>
            <xs:extension base="tns:personaOficina">
                <xs:sequence>
                    <xs:element name="cambiarContrasena"
type="xs:boolean"/>
                    <xs:element name="cargo"
type="tns:edNomencladorSimple" minOccurs="0"/>
                    <xs:element name="estado" type="tns:edEstadoUsuario"
minOccurs="0"/>
                    <xs:element name="estatus" type="xs:string"
minOccurs="0"/>
                    <xs:element name="fechaRegistro" type="xs:dateTime"
minOccurs="0"/>
                    <xs:element name="funcionario" type="xs:boolean"/>
                    <xs:element name="idUsuario" type="xs:double"/>
                    <xs:element name="oficina" type="tns:edOficina"
minOccurs="0"/>
                    <xs:element name="rolesBase" type="tns:edRolBase"
nillable="true" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element name="sesion" type="tns:edSesion"
minOccurs="0"/>
                    <xs:element name="trazas" type="tns:edTraza"
nillable="true" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element name="ultimoCambioContrasena"
type="xs:dateTime" minOccurs="0"/>
                    <xs:element name="usuario" type="xs:string"
minOccurs="0"/>
```

```
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="personaOficina">
        <xs:complexContent>
            <xs:extension base="tns:edPersona">
                <xs:sequence/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="edNomencladorSimple">
        <xs:sequence>
            <xs:element name="entidadNomenclador" type="xs:string"
minOccurs="0"/>
            <xs:element name="estado" type="xs:boolean" minOccurs="0"/>
            <xs:element name="id" type="xs:anyType" minOccurs="0"/>
            <xs:element name="nombre" type="xs:string" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="edOficina">
        <xs:sequence>
            <xs:element name="idOficina" type="xs:int" minOccurs="0"/>
            <xs:element name="nombre" type="xs:string" minOccurs="0"/>
            <xs:element name="telefono" type="xs:string" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="edRolBase">
        <xs:complexContent>
            <xs:extension base="tns:edRol">
                <xs:sequence>
                    <xs:element name="accion" type="xs:string"
minOccurs="0"/>
                    <xs:element name="administracion" type="xs:boolean"/>
                    <xs:element name="area" type="tns:edAreaF"
minOccurs="0"/>
                    <xs:element name="modulo" type="tns:edModulo"
minOccurs="0"/>
                    <xs:element name="titulo" type="xs:string"
minOccurs="0"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <xs:complexType name="edRol">
        <xs:sequence>
            <xs:element name="activo" type="xs:boolean"/>
            <xs:element name="idPadre" type="xs:double"/>
            <xs:element name="idRol" type="xs:double"/>
            <xs:element name="nombre" type="xs:string" minOccurs="0"/>
            <xs:element name="prioridad" type="xs:int" minOccurs="0"/>
            <xs:element name="rolFisico" type="xs:string" minOccurs="0"/>
```

```
            <xs:element name="rolesHijos" type="tns:edRol"
nillable="true" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="edAreaF">
        <xs:sequence>
            <xs:element name="activo" type="xs:boolean"/>
            <xs:element name="id" type="xs:int"/>
            <xs:element name="nombre" type="xs:string" minOccurs="0"/>
            <xs:element name="roles" type="tns:edRolBase" nillable="true"
minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="edModulo">
        <xs:sequence>
            <xs:element name="activo" type="xs:boolean"/>
            <xs:element name="idModulo" type="xs:long"/>
            <xs:element name="nombre" type="xs:string" minOccurs="0"/>
            <xs:element name="rolesBase" type="tns:edRolBase"
nillable="true" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="edSesion">
        <xs:sequence>
            <xs:element name="estado" type="xs:boolean" minOccurs="0"/>
            <xs:element name="fecha" type="xs:dateTime" minOccurs="0"/>
            <xs:element name="idSesion" type="xs:double" minOccurs="0"/>
            <xs:element name="ip" type="xs:string" minOccurs="0"/>
            <xs:element name="usuario" type="tns:edUsuario"
minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="edTraza">
        <xs:sequence>
            <xs:element name="descripcion" type="xs:string"
minOccurs="0"/>
            <xs:element name="fecha" type="xs:dateTime" minOccurs="0"/>
            <xs:element name="idTraza" type="xs:double"/>
            <xs:element name="ip" type="xs:string" minOccurs="0"/>
            <xs:element name="ipDouble" type="xs:double"/>
            <xs:element name="sistema" type="xs:boolean"/>
            <xs:element name="usuario" type="tns:edUsuario"
minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="registrarCuentaUsuario_1Response">
        <xs:sequence>
            <xs:element name="return" type="tns:edUsuario"
minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
```

```
    <xs:complexType name="Exception">
        <xs:sequence>
            <xs:element name="message" type="xs:string" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="obtenerUsuarioPorUsuarioWrapper">
        <xs:sequence>
            <xs:element name="usuario" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="obtenerUsuarioPorUsuarioResponse">
        <xs:sequence>
            <xs:element name="return" type="tns:edUsuario"
minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="SOAPException">
        <xs:sequence>
            <xs:element name="message" type="xs:string" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="autenticar1Wrapper">
        <xs:sequence>
            <xs:element name="nombreusuario" type="xs:string"/>
            <xs:element name="contrasenna" type="xs:string"/>
            <xs:element name="ip" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="autenticar_1Response">
        <xs:sequence>
            <xs:element name="return" type="tns:edUsuario"
minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="autenticarWrapper">
        <xs:sequence>
            <xs:element name="nombreusuario" type="xs:string"/>
            <xs:element name="contrasenna" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

Anexo 4: Acta de liberación interna de productos de software.

UCI Universidad de las Ciencias Informáticas | FACULTAD # 3
CENTRO DE GOBIERNO ELECTRÓNICO

CEGEL

Acta de Liberación Interna de Productos Software

Fecha de emisión del acta: 12/06/2017

Emitida a favor de: Tesis *"Componente para la integración del marco de trabajo XEGFORT al Sistema de Gobierno Electrónico".*

Datos del producto

Artefacto	Versión	Estado final	Cantidad Iteraciones	Tipos de pruebas realizadas	Fecha de liberación
App: Componente para la integración del marco de trabajo XEGFORT al Sistema de Gobierno Electrónico	*1.0*	*0*	*2*	*Pruebas de eficiencia* *Pruebas de Funcionalidad*	*12/06/2017*

MSc. Yordanis García Leiva
Asesor de Calidad CEGEL

CEGEL
UCI Universidad de las Ciencias Informáticas
CENTRO DE GOBIERNO ELECTRÓNICO

Luis Javier Rodríguez Castro
Autor

1