UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

TRABAJO DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS COMPUTACIONALES

TEMA:

"COMPARAR DOS ESB J2EE; MULEESB VS GLASSFISHESB/OPENESB, CON EL PROTOTIPO: DE FACTURACIÓN ELECTRÓNICA."

AUTOR: FABRICIO XAVIER HUERA VINUEZA

DIRECTOR: ING. MAURICIO REA

IBARRA – ECUADOR

2016

UNIVERSIDAD TÉCNICA DEL NORTE

BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN

A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

La UNIVERSIDAD TÉCNICA DEL NORTE dentro del proyecto Repositorio Digital institucional, determina la necesidad de disponer los textos completos de forma digital con la finalidad de apoyar los procesos de investigación, docencia y extensión de la universidad.

Por medio del presente documento dejo sentada mi voluntad de participar en este proyecto, para lo cual pongo a disposición la siguiente investigación:

DATOS DE CONTACTO	
CÉDULA DE IDENTIDAD	040121121-4
APELLIDOS Y NOMBRES	HUERA VINUEZA FABRICIO XAVIER
DIRECCIÓN	LA VICTORIA PASAJE L Y VICTOR BARAHONA
EMAIL	xavys10@yahoo.com
TELÉFONO FIJO	062616131
TELÉFONO MÓVIL	0999593954
DATOS DE LA OBRA	
TÍTULO	"COMPARAR DOS ESB J2EE; MULEESB VS GLASSFISHESB/OPENESB, CON EL PROTOTIPO: DE FACTURACIÓN ELECTRÓNICA."
AUTOR	MAURICIO ROLANDO CARRILLO PABÓN
FECHA	JUNIO DEL 2016
PROGRAMA	☒ PREGRADO ☐ POSTGRADO
TÍTULO POR EL QUE OPTA	INGENIERÍA EN SISTEMAS COMPUTACIONALES
DIRECTORA	ING. MAURICIO REA

2. AUTORIZACIÓN DE USO A FAVOR DE LA UNIVERSIDAD

Yo, Huera Vinueza Fabricio Xavier, con cédula de identidad Nro. 040121121-4, en calidad de autor y titular de los derechos patrimoniales del proyecto de grado descrito anteriormente, hago entrega del ejemplar respectivo en forma digital y autorizo a la Universidad Técnica del Norte, la publicación de la obra en el Repositorio Digital Institucional y el uso del archivo digital en la Biblioteca de la Universidad con fines académicos, para ampliar la disponibilidad del material y como apoyo a la educación, investigación y extensión, en concordancia con la Ley de Educación Superior Artículo 144.

3. CONSTANCIAS

El autor (es) manifiesta (n) que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto la obra es original y que es (son) el (los) titular (es) de los derechos patrimoniales, por lo que asume (n) la responsabilidad sobre el contenido de la misma y saldrá (n) en defensa de la Universidad en caso de reclamación por parte de terceros.

Firma

Nombre: Huera Vinueza Fabricio Xavier

Cédula: 040121121-4

Ibarra, Junio del 2016

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE INVESTIGACIÓN

A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

Yo, Huera Vinueza Fabricio Xavier, con cédula de identidad Nro. 040121121-4, manifiesto mi voluntad de ceder a la Universidad Técnica del Norte los derechos patrimoniales consagrados en la ley de propiedad intelectual del Ecuador, artículo 4, 5 y 6, en calidad de autor del proyecto de grado denominado: **"COMPARAR DOS ESB J2EE; MULEESB VS GLASSFISHESB/OPENESB, CON EL PROTOTIPO: DE FACTURACIÓN ELECTRÓNICA."** que ha sido desarrollado para optar por el título de Ingeniero en Sistemas Computacionales, en la Universidad Técnica del Norte, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En mi condición de autor me reservo los derechos morales de la obra antes citada. En concordancia suscribo este documento en el momento que hago entrega del trabajo final en formato impreso y digital a la Biblioteca de la Universidad Técnica del Norte.

.....

Firma

Nombre: Huera Vinueza Fabricio Xavier

Cédula: 040121121-4

Ibarra, Junio del 2016

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CERTIFICACIÓN DIRECTOR DE TESIS

Certifico que la tesis **"COMPARAR DOS ESB J2EE; MULEESB VS GLASSFISHESB/OPENESB, CON EL PROTOTIPO: DE FACTURACIÓN ELECTRÓNICA."**, ha sido realizada con interés profesional y responsabilidad por el señor: Huera Vinueza Fabricio Xavier, portador de la cédula de identidad Nro. 040121121-4; previo a la obtención del Título de Ingeniero en Sistemas Computacionales.

.....

Ing. Mauricio Rea

DIRECTOR DE GRADO

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

DECLARACIÒN

Yo, Huera Vinueza Fabricio Xavier, declaro bajo juramento que el trabajo aquí descrito es de mi autoría, que no ha sido previamente presentado para ningún grado, ni calificación profesional; y que he consultado las referencias bibliográficas que se incluyen en este documento.

.....

Firma

Nombre: Huera Vinueza Fabricio Xavier

Cédula: 040121121-4

Ibarra, Junio del 2016

DEDICATORIA

A:

Mis padres, por haber sido un gran apoyo constante y la fuente de mi fuerza y voluntad de salir adelante, que me guiaron en cada paso que di durante sus días de vida, y que fueron el pilar fundamental en toda mi educación, tanto en la académica, como en la vida diaria y en todo lo que me he convertido.

Mi esposa por estar ahí en las buenas y malas, por confiar en mí y darme aliento a seguir luchando, por su amor incondicional y su apoyo en mi vida.

Mi familia por su unión ante los problemas y el apoyo que dan a cada miembro de la familia.

Fabricio Xavier Huera Vinueza

AGRADECIMIENTO

A:

Dios, por haberme dado la vida, la salud, el apoyo constante de mis padres, por acompañarme en todas las etapas de la vida y enseñarme conocimientos de la vida.

Mi esposa Karlita, que la amo con todo mi corazón que con su amor valor y entrega es una persona incondicional en mi vida, es mi soporte, mi mejor amiga sus consejos, ánimo y compañía me ayuda en los momentos más difíciles de mi vida, me enseñó a no bajar los brazos.

Los docentes de la universidad, por compartirme los conocimientos, valores profesionales y éticos que me ayudaron a tomar las decisiones en toda mi vida estudiantil.

Al Ing. Mauricio Rea, por su paciencia, sus consejos y su acertada asesoría para poder culminar muy satisfactoriamente este proyecto de titulación.

Fabricio Xavier Huera Vinueza

RESUMEN

La presente tesis está conformada por siete capítulos en los cuales se describe el estudio comparativo de los ESB Mule ESB y OpenESB con el prototipo del sistema de facturación electrónica que fue desarrollado con el último esquema del SRI para la emisión de comprobantes electrónicos.

En el capítulo uno se describe el planteamiento del problema, la solución que se puede dar al correcto funcionamiento de las empresas con la utilización de la Arquitectura Orientada a Servicios.

El capítulo dos describe todo lo referente a SOA ya que las empresas requieren interactuar con procesos internos y externos sin afectar el tiempo, costos que esto representa por tanto, SOA nos ayuda a minimizar esos efectos ya que está basada en definiciones de los servicios reutilizables, donde los proveedores y los consumidores de servicios puedan interactuar de una forma desacoplada en los procesos de negocios.

El capítulo tres se basa en el estudio de bus de servicio Mule ESB el cual nos brinda un sin número de beneficios, con mensajería ligera, permite una rápida conexión e intercambio de los datos entre las aplicaciones, utiliza la Arquitectura Orientada a Servicios, permitiendo una rápida y fácil integración en la mayoría de sistemas que se encuentran desarrollados en java, con eclipse que es una herramienta utilizada en las mayoría de proyectos de software libre.

En el capítulo cuatro se analiza a OpenESB, es una implementación de ESB basado en las especificaciones JBI. Permite integrar fácil y rápidamente las aplicaciones empresariales y WS como las aplicaciones compuestas que están débilmente acopladas, esto le permite componer de manera rápida y fluida las aplicaciones compuestas, con todas y cada una de las ventajas de la verdadera Arquitectura Orientada a Servicios.

El capítulo cinco se realiza el estudio comparativo entre los dos Buses de Servicios Empresariales, que se basara en estudiar las características técnicas y

funcionales de cada Bus de Servicio Empresarial, comparando las funcionalidades de Mule ESB versus OpenESB para poder establecer cual ESB cumple con todas las especificaciones necesarias y se adapte mejor, preste las facilidades para la optimización de tiempo y recursos en el desarrollo del sistema de facturación electrónica.

El capítulo seis se desarrolla el prototipo de facturación electrónica, el cual fue realizado en la plataforma de java utilizando como framework de desarrollo a Eclipse Kepler y la base de datos PostgreSQL, todo el sistemas se lo desarrollo con la metodología ágil XP (Programación Extrema), el sistema cumple con el esquema de emisión de facturas electrónicas del SRI.

Finalmente el capítulo siete se establecerán las conclusiones y recomendaciones que se obtuvieron cuando se realizó el desarrollo del proyecto.

ABSTRACT

The present work has composed of seven chapters in which the comparative study of Mule ESB and OpenESB described with the prototype of electronic invoicing system that it was developed with the latest SRI scheme for issuing electronic vouchers.

In chapter one has described problem statement, the solution that it can be given to the proper functioning of enterprises with the use of service-oriented architecture.

Chapter 2 has described everything about this or since enterprises require to interact with internal and external processes without affecting the cost of time this represents therefore, this SOA helps us to minimize these effects since it is based on definitions of reusable services, where service providers and consumers can interact in a decoupled way in the business processes.

Chapter 3 has based on the study of Mule ESB bus service which gives us countless benefits with a slight messaging allows quick connection and exchange of data between applications, use the service-oriented architecture allowing for quick and easy integration into most systems that are found developed in java, with eclipse is a tool used in most free software projects.

Chapter 4 has analyzed OpenESB which is an implementation of ESB based on JBI specifications. It allows quickly and easily integrate business and WS applications such as composite applications that they are loosely coupled, this allows you to compose quickly and seamlessly composite applications, with each and every one of the advantages of true service-oriented architecture.

Chapter 5 has performed the comparative study between the two Buses Enterprise Services, which is based on studying the technical and functional characteristics of each Enterprise Service Bus, comparing the features of Mule ESB versus OpenESB to establish which ESB meets all specifications necessary and best suited pay facilities to optimize time and resources in the development of electronic invoicing.

Chapter 6 has developed the prototype of electronic invoicing, which it was done on the platform of java using as a development framework for Eclipse Kepler and PostgreSQL database, the whole system was developed with agile methodology XP (Extreme Programming) the system complies with the scheme of issuing electronic bills SRI.

Finally, Chapter 7 will be established the conclusions and recommendations were obtained when the project was performed.

ÌNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÌNDICE DE TABLAS

CAPÍTULO I

1 INTRODUCCIÓN.

1.1 ANTECEDENTES.

En la empresa MantizSoft se ha observado que los procesos de facturación se realizan mediante la utilización de documentos físicos en papel, la información esta almacenada en archiveros, esto crea un gran problema porque al momento de solicitar un reporte de las facturas, el proceso es muy lento por la minuciosa búsqueda de documentos en los archivos físicos que posee la empresa.

La inadecuada estructuración y flexibilidad en el intercambio de información entre los distintos módulos, los cuales son el de facturación así como el de inventarios, como también la falta total de integración entre la tecnología y los procesos del negocio, ya que estos proceso funciona en diferentes tecnologías que impiden la optimización del tiempo y calidad que se requiere en el negocio.

En algún punto en el crecimiento de las organizaciones o las empresas, con relación a las soluciones informáticas que proveen sistemas que soportan los procesos de negocios, y cuando el tamaño creciente de la empresa y la complejidad que estas tienen; hay que dar el salto al concepto de arquitectura empresarial.

Debido a estas necesidades se ha optado por implementar la factura electrónica el mismo que será desarrollado con herramientas de software libre, utilizando SOA, lenguaje de programación java.

El estudio propone variedad de ventajas económicas y de tecnología para la empresa porque la arquitectura orientada a servicios (SOA) nos brinda un sin fin de posibilidades para mejorar el funcionamiento de la empresa.

FIGURA 1.1: Sin SOA, con SOA.

Fuente: https://sistemas2009unl.wordpress.com/

El uso de los bus de servicios empresariales (ESB) es una solución que facilita la integración distribuida, que está basada en mensajes y en estándares.

La función que realiza un ESB es facilitar una comunicación confiable entre los diferentes recursos tecnológicos como plataformas, aplicaciones y servicios, que están distribuidos en diferentes sistemas de toda la empresa. La fortaleza de una solución de ESB es la posibilidad de la comunicación entre sistemas de cualquier protocolo, se encarga de traducir de un lenguaje a otros, el lenguaje que se utiliza se normaliza, empleando lenguaje XML. (Borja, Florez, & Torres Leon, 2011).

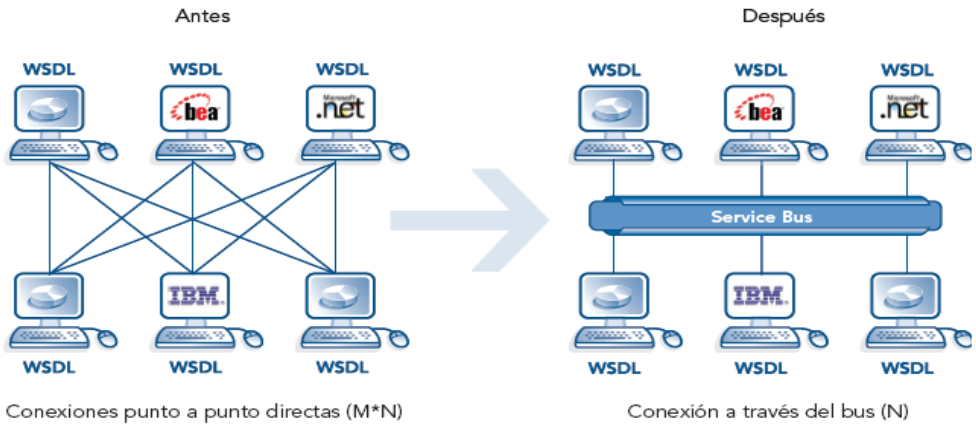


FIGURA 1.2: Antes y Después de ESB.

Fuente: http://temariotic.wikidot.com/

1.1.1 INTRODUCCIÓN.

Las empresas para ser competitivas requieren arquitecturas, procesos comerciales de TI flexibles, que soporten un alto nivel de desarrollo y un alto nivel de competencia. Las empresas están inmersas en un mundo que exige un desarrollo empresarial y tecnológico, apresurado y competitivo, lo que obliga a que sus procesos deban invariablemente optimizar y mejorar resultados.

La optimización y mejora que se espera requiere contar con una arquitectura que utilice la tecnología de integración entre aplicaciones y procesos eficientes para un buen desempeño de la empresa.

Cada vez es mucho más frecuente que las aplicaciones tienen que responder a variedad de necesidades que hace años atrás eran difíciles de realizar o no existían:

- Ser autónomas de los gestores de bases de datos y sistemas operativos.
- Ser accedidos desde lugares distantes geográficamente.
- Interactuar con diferentes sistemas que ya se encuentran en funcionamiento
- Atender gran variedad de interacciones, originadas por los usuarios que utilizan la aplicación simultáneamente.

Así, todas las aplicaciones desarrolladas han frecuentado un enfoque monolítico y centralizado hacia entornos heterogéneos y distribuidos.

1.1.2 PLANTEAMIENTO DEL PROBLEMA.

En la actualidad existe variedad complejidades en las organizaciones que tienen que integrar y coordinar sus sistemas operacionales ya que estos trabajan independientemente y solucionan determinadas tareas, además no existen estándares de desarrollo tecnológico, ocasionando una amplia variedad de aplicaciones que trabajan en diferentes plataformas lo cual hace que la integración entre ellas sea complicada.

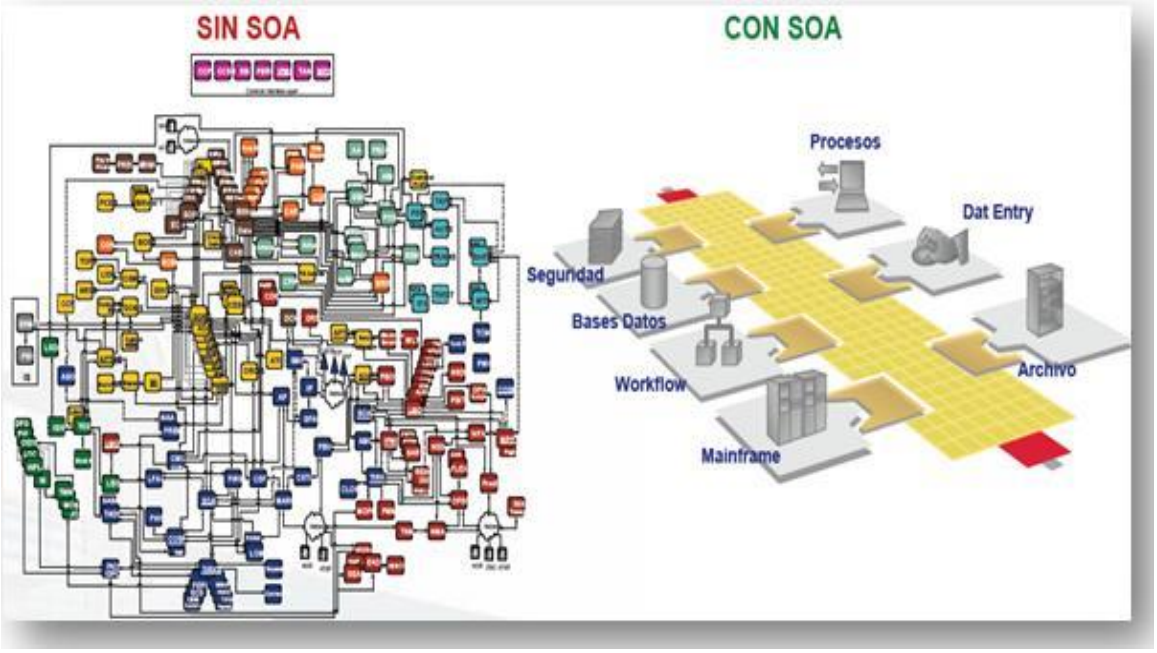


FIGURA 1.3: El antes y después de SOA.
Fuente: http://sistemas2009unl.wordpress.com/

Es evidente que SOA se ha convertido desde hace varios años en el término de moda del mundo de las TIC, pero existe escaza información sobre SOA especialmente de los ESB la mayoría de información se encuentra en inglés, pero a sí mismo es un término tan poco entendido y abstracto que pocas son las empresas o profesionales que han asumido a SOA como la estrategia para resolver los más serios y complejos casos de integración y producción de sistemas informáticos coherentes con la necesidad de comunicarse con los sistemas actuales.

Existen multitud de topologías que pueden ser implementadas en varias tecnológicas, eso quiere decir que al tener una variedad de plataformas tecnológicas que proponen diversos patrones de arquitecturas para implementar SOA, lo cual resulta un gran problema saber cuál de todos los ESB escoger, en este sentido en el mercado existen gran cantidad de ESB de gran calidad sin embargo decidir cuál de todos se puede adaptar de la mejor manera a nuestras aplicaciones se convierte en el problema inicial antes de asumir cualquier proyecto SOA.

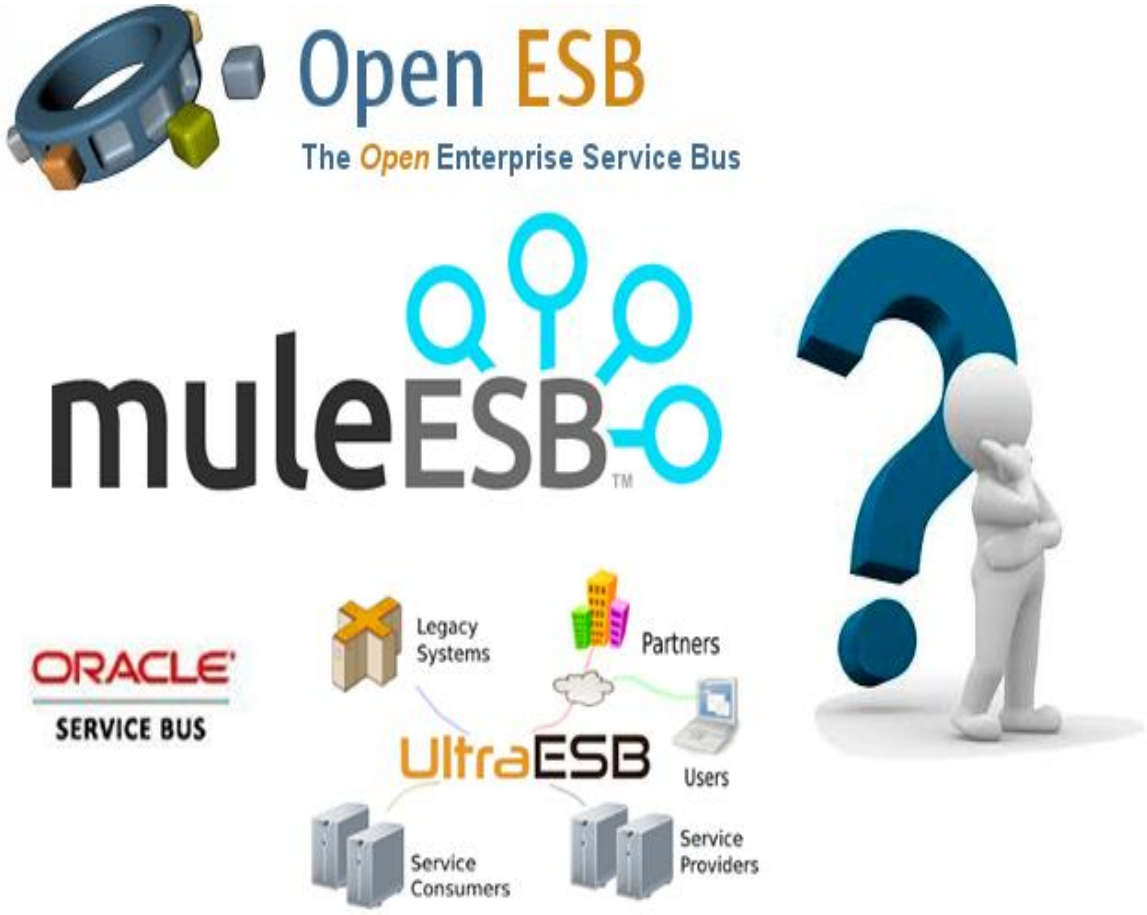


FIGURA 1.4: Variedad de ESB existentes.

Fuente: Propia

Cuando se acude a documentación es comprensible que un editor de software de un fabricante compare su producto con el de los competidores y se hagan todo lo posible para promover su producto y resaltar sus mejores partes, el problema es que la mayoría de información es escasa.

A esto se debe sumar que la mayoría de los proyectos SOA fracasan, porque no se aplican o entiende correctamente los principios de SOA, no se respetan las responsabilidades de cada capa o tecnología, o se escogen mal las tecnologías base para la solución SOA.

1.1.3 JUSTIFICACIÓN.

Cada fabricante que propone su Bus de servicios intenta convencer sobre las múltiples bondades que se obtienen al usar dicho ESB, en este sentido es imprescindible tener un marco de referencia adecuado para tomar esta decisión.

El estudio propuesto selecciona 2 de los Buses de Servicios identificados como referentes de SOA, estos son OpenESB/GlassFishESB y Mule ESB, provenientes de casas de software amigas que contraponen 2 estrategias completamente diferentes de integración, el primero usando JBI y el segundo su propio motor de integración basado en interfaces y conectores, ambos basados en J2EE pero en su estructura interna son distintos, allí radica la importancia de contraponer sus arquitecturas, estrategias, tecnologías e implementar un caso de integración sobre ambos ESB.

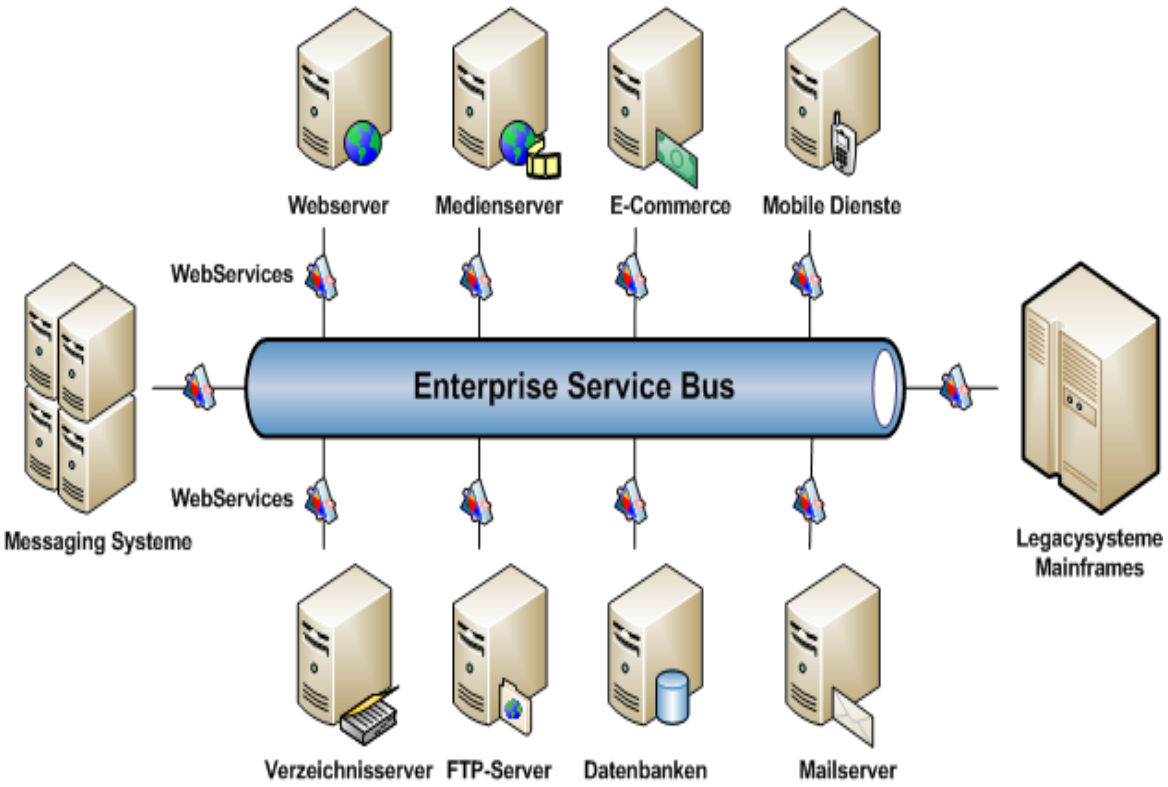


FIGURA 1.5: Diagrama de la arquitectura ESB.

Fuente: http://muleesb.blogspot.com/

Se ha seleccionado MULE ESB puesto que este se ha convertido en el ESB mayormente usado en el mundo y su accionar en la comunidad hace que gane terreno en relación a los ESB propietarios, este ESB es un gestor distribuible que maneja interacciones con aplicaciones y servicios que usan diferentes tecnologías de mensajería y transporte, Mule está diseñado para ser fácilmente embebible y liviano en aplicaciones Java y servidores de aplicación.

FIGURA 1.6: Representación Mule ESB.
Fuente: https://www.mulesoft.org

Respecto a Open ESB/GlassFishESB este es un bus de servicios empresarial basado en Java de código abierto, este él es ESB de referencia para la implementación de JBI (Java Business Integration), XML, XML Schema, WSDL, BPEL y aplicaciones compuestas que proporciona simplicidad, eficiencia, durabilidad a largo plazo.

FIGURA 1.7: Logo de OpenESB.
Fuente: http://www.open-esb.net/

Al finalizar este trabajo se espera tener los criterios suficientes para identificar las características que realmente se espera de las herramientas de desarrollo usadas en la integración de modo que fomenten sencillez, productividad, fiabilidad, eficiencia, escalabilidad, para el desarrollo de SOA.

1.1.4 OBJETIVOS.

- **Objetivo General.**

- Comparar dos ESB (Enterprise Service Bus); Mule ESB vs GlassFishESB/OpenESB, con el Prototipo: Facturación Electrónica.

- **Objetivos específicos.**

- Determinar el marco referencial.

- Examinar el funcionamiento y características del buses de servicios Mule ESB.

- Examinar el funcionamiento y características del buses de servicios GlassFishESB/OpenESB.

- Establecer parámetros de comparación de los ESB (Enterprise Service Bus).

- Desarrollar un prototipo de facturación para la implementación de los ESB.

1.1.5 ALCANCE.

En el presente proyecto se va a estudiar dos ESB (Enterprise Service Bus) los cuales son:

- MULE ESB.

- OPEN ESB/GLASSFISH ESB.

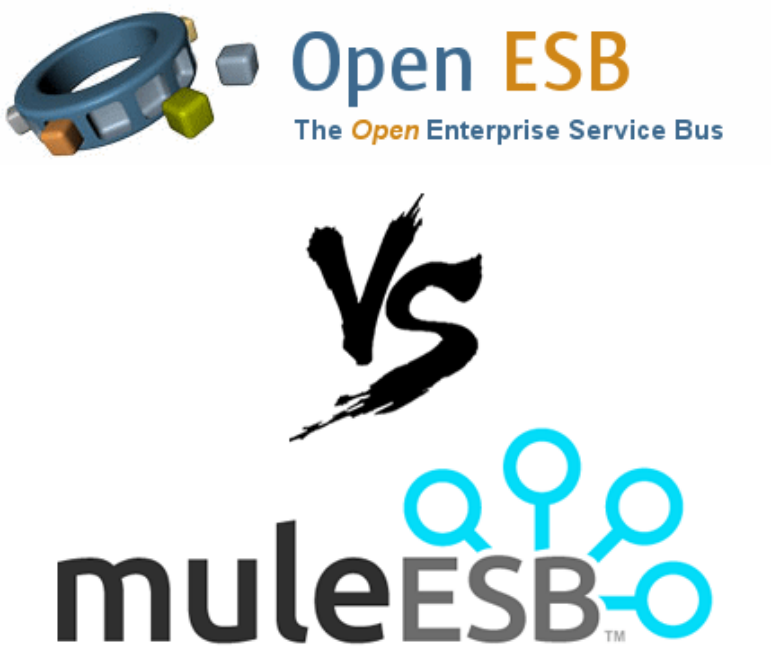


FIGURA 1.8: ESB a comparar.

Fuente: Propia

Puesto que el primero constituye el ESB de código abierto más popular en la actualidad, la mayoría de empresas están utilizando este ESB.

OPEN ESB/GLASFISH ESB constituye el ESB de referencia para JBI, la cual es el estándar de integración de servicios que se usa en J2EE.

Se incluye dentro del alcance del proyecto de tesis el análisis de los ESB y las definiciones necesarias de los servicios para conseguir su integración. La creación de un prototipo con el mejor ESB que se obtiene del resultado del presente estudio, para el prototipo de facturación electrónica utilizaremos la metodología XP (eXtreme Programming).

Arquitectura del prototipo de facturación electrónico.

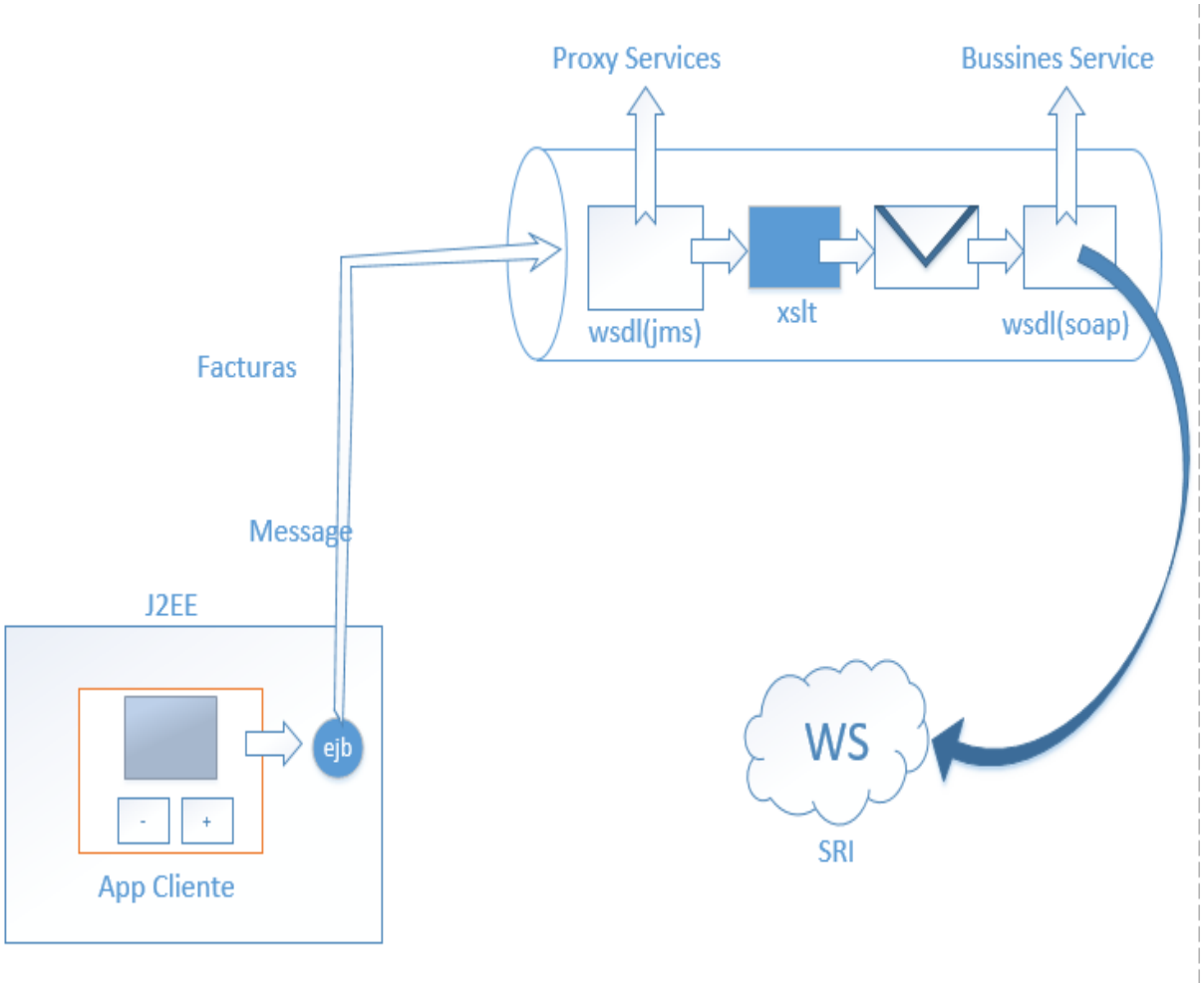


FIGURA 1.9: Arquitectura del prototipo.

Fuente: Propia

CAPÍTULO II

2 INTRODUCCIÓN SOA.

Cuando se introdujeron las tecnologías de información (TI) en todas las industrias, las empresas han construido y desarrollado sus sistemas de información en base a las necesidades que tienen las empresas de negocios, conformando infraestructuras de los datos que serían transformados en almacenes de información, pobremente cohesionados o mayormente aislados.

Esta situación llegó a un punto crítico cuando las empresas empezaron a enfrentar procesos complejos de fusión o expansión, obligando a ordenar la información y transferirla de una manera más consolidada y eficiente.

Cada vez hay aplicaciones más complejas, con mucho menos tiempo de desarrollo y un presupuesto limitado, y en la mayoría de los casos se requiere re implementar funcionalidades ya existentes.

Tratar de reutilizar todas las funcionalidades resulta una labor complicada debido a que no fueron diseñadas para poder integrarse entre sí, o bien fueron desarrolladas sobre tecnologías o plataformas desacordes entre sí.

El significado de “integración” ha llegado a convertirse en la llave maestra necesaria para la gestión táctica, operacional y estratégica de los distintos actores de la empresa, lo que permite reducir el tiempo y los esfuerzos de desarrollo y mantención de las TI, las cuales las ponen definitivamente al servicio del crecimiento de las empresas.

SOA es la evolución de la computación distribuida, basada en el modelo de pregunta/respuesta para las aplicaciones síncronas y asíncronas. En ellas las funciones individuales o la lógica de negocio son presentadas como un servicio de aplicaciones que consumen servicios (clientes). La clave de estos servicios es que son de naturaleza desacoplada. La interfaz de los servicios es indiferente de la implementación de los mismos.

Por ejemplo. Uno de los servicio puede estar implementado tanto en. JavaEE como en .NET y la aplicación que va a consume el servicio puede estar en un lenguaje y plataforma diferente a esto dos.

SOA se muestra en este escenario como una de las soluciones adecuadas, definiendo una de las arquitecturas donde todos los procesos o actividades están creados para ofrecer un servicio. Lo que se entiende por Servicio es que es un componente de software con las interfaces bien definidas e independientes de toda su implementación.

2.1 DEFINICIONES.

La arquitectura Orientada a Servicios (SOA) es un modelo de computación distribuida, que consiste en la provisión de la funcionalidad mínima y necesaria (a la carta), según los requisitos particulares de cada usuario, a partir de servicios independientes y disponibles públicamente en Internet. La orientación actual de las aplicaciones distribuidas tiene un origen en el estilo arquitectónico SOA, que proporciona a los diseñadores de aplicaciones guías de acuerdo a principios, y patrones de diseño relacionados con el concepto central de servicio. En SOA los servicios desempeñan un papel fundamental. Los servicios son elementos básicos de computación y actúan como un engranaje que permite la creación de servicios muchos más complejos en términos de interacción y funcionalidad, que a su vez pueden reutilizar para el desarrollo de aplicaciones distribuidas basadas en servicios. (López-Vázquez, 2012).

(Kendall & Kendall, 2011) Afirman lo siguiente:

El desarrollo modular fue el que originó un concepto que se denomina arquitectura orientada a servicios (SOA), el cual es diferente a los módulos en el diagrama de estructura. En vez de ser jerárquica como la metodología descendente de los diagramas de estructura, la metodología SOA se encarga de crear servicios individuales que no tengan ninguna asociación o estén débilmente acoplados entre sí.

Cada servicio ejecuta una acción. Un servicio puede devolver el número de días en este mes; otro indica si el presente es un año bisiesto; un tercer servicio puede reservar cinco noches en un cuarto de hotel, desde finales de febrero hasta principios de marzo. Aunque el tercer servicio necesita conocer los valores que se obtienen del primero y segundo, son independientes unos de otros. Podemos usar cada uno de los servicios en otras aplicaciones dentro de la organización, o incluso en otras organizaciones.

Podemos decir que la arquitectura orientada a servicios es sólo un conjunto de servicios a los se llaman para que provean funciones específicas. En vez de incluir llamadas a otros servicios, el servicio puede estar usando ciertos protocolos definidos para poder comunicarse con otros.

La figura 2.1 muestra cómo se hacen llamadas a los servicios en todo el sistema. Los servicios pueden ser generales en naturaleza y se pueden externalizar, o incluso pueden estar disponibles en Web. Otros servicios son más especializados y orientados hacia la empresa en sí. Estos servicios para empresas proveen reglas de negocios y también pueden diferenciar a una empresa de otra. Se pueden hacer llamadas a los servicios en un momento dado y en repetidas ocasiones en muchos módulos de aplicaciones.

La carga de conectar los servicios de una forma útil es un proceso denominado orquestación, y depende del diseñador de sistemas. Para lograr esto podemos incluso seleccionar servicios de un menú y supervisarlos mediante un tablero de control SOA.

Para poder establecer SOA, los servicios deben cumplir con los siguientes requisitos:

- Ser modulares.
- Ser reutilizables.
- Trabajar en conjunto con otros módulos (interoperabilidad).
- Ser capaces de clasificarse e identificarse.
- Debe ser posible supervisarlos.
- Cumplir con los estándares específicos. (pág. 522)

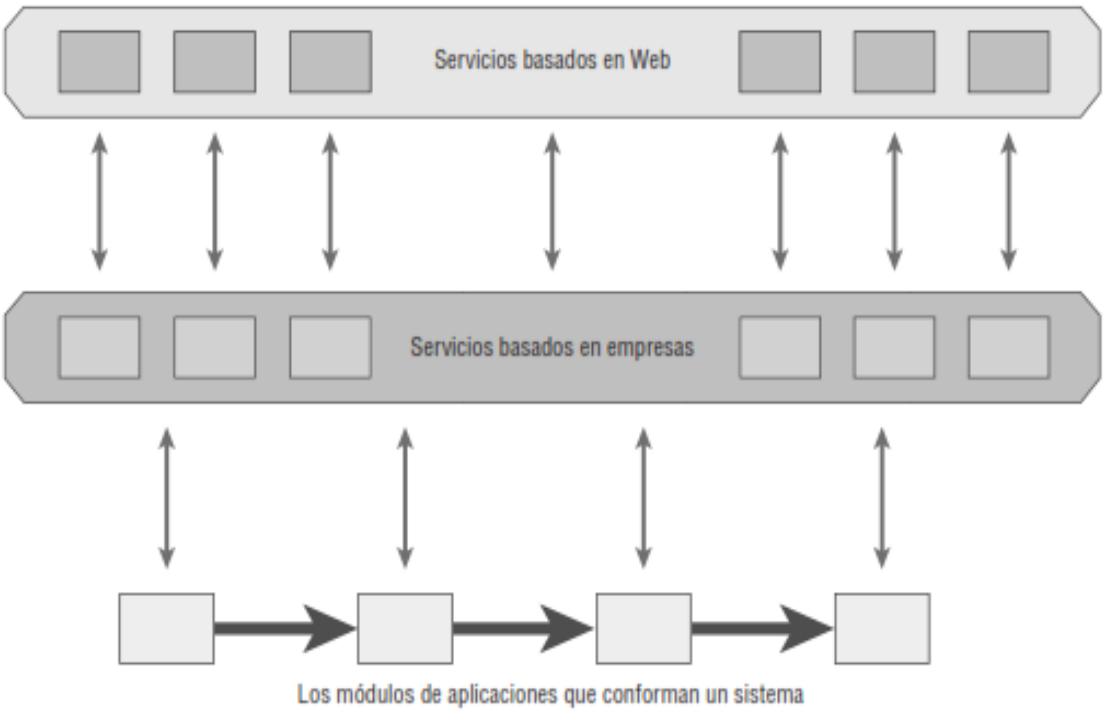


FIGURA 2.10: Los módulos de SOA son independientes y pueden ser omnipresentes.
Fuente: (Kendall & Kendall, 2011, pág. 522)

Aunque las ventajas de reutilización e interoperabilidad son obvias, SOA presenta sus retos. Primero que nada, hay que acordar los estándares industriales. Después hay que mantener una biblioteca donde los desarrolladores busquen los servicios que necesitan. Por último, la seguridad y la privacidad pueden presentar problemas al utilizar software desarrollado por alguien más. Los defensores de SOA afirman que la arquitectura orientada a servicios ha hecho posibles muchas de las características que se encuentran en la Web 2.0. (pág. 523)

2.2 PRINCIPIOS DE SOA.

En la publicación de Trumbulianos que se basa en (Erl, 2012) dice:

Servicios reusables: Los servicio deben ser diseñado y construido para poder ser reutilizados dentro de las mismas aplicaciones, dentro del dominio de aplicaciones de la empresa o incluso dentro de los dominios públicos para que sean utilizados masivamente.

Servicios con un contrato formal: Todos los servicios desarrollados, deben proporcionar un contrato donde debe figurar: el nombre del servicio, las funcionales que ofrece, su forma de acceso, los datos de entrada y de salida de cada funcionalidad. Todo consumidor del servicio, podrá acceder mediante el contrato, así logrando la independencia entre el consumidor y la implementación del propio servicio.

Servicios con bajo acoplamiento: Es decir, los servicios tienen que poder ser independientes unos de otros. Para llegar a lograr este acoplamiento bajo, lo que se hará es que cada vez que se vaya a ejecutar un servicio, se acceda a través del contrato, lo cual se logrará la independencia entre el servicio que se está ejecutan y el que llama. De esta forma serán en su totalidad reutilizables.

Servicios que permitan la composición: Los servicios deben ser construidos de manera que pueda ser utilizado en la construcción de servicios genéricos que sean del más alto nivel, el cual debe estar compuesto de servicios de bajo nivel. En los casos de Web Service, esto debe lograrse mediante el uso de protocolos para orquestación (WS-BPEL) y coreografía (WS-CDL).

Servicios tienen que ser autónomos: Los Servicios deben tener su entorno de ejecución. De tal manera que el servicio es totalmente independiente y así se puede asegurar que podrán ser reutilizable desde la plataforma de ejecución.

Servicios sin estado: Un servicio no debe guardar ningún tipo de información. Debe ser así porque la aplicación que está formada por el conjunto de aplicaciones no debe almacenar algún tipo de información, puede producir algunos problemas de inconsistencia de los datos. La solución debe ser que un servicio contenga únicamente lógica, y que la información se almacene en algún sistema de información.

Servicios tiene que poder ser descubiertos: Todo servicio tiene que ser descubierto de alguna forma para que sea utilizado, consiguiendo con esto evitar la creación de servicios que proporcionen funcionalidades iguales. En el caso de los Web Service, el descubrimiento se logra publicando las interfaces de los servicios en registros UDDI.

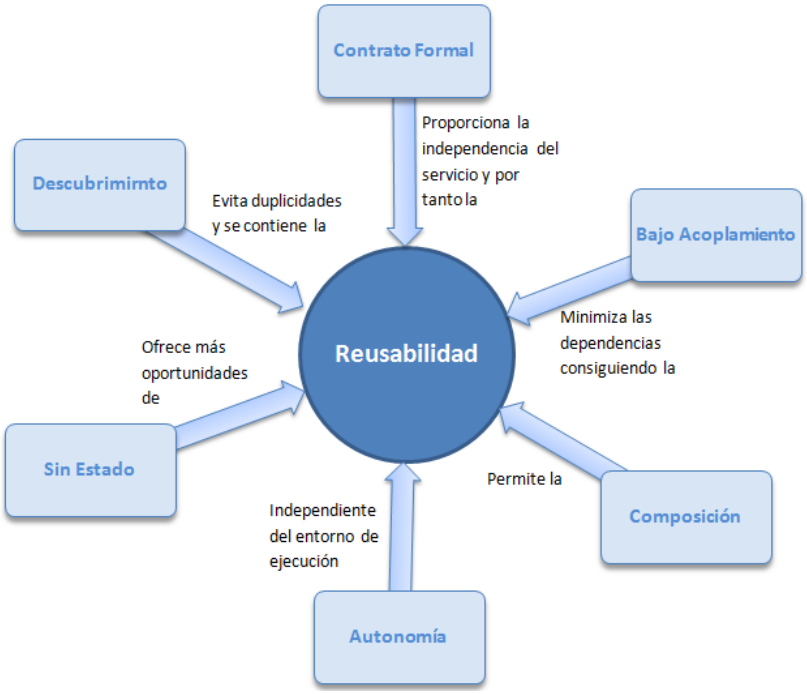


FIGURA 2.11: Interrelación entre los principios de SOA.

Fuente: http://arquiware-pc.blogspot.com/

2.3 VENTAJAS Y DESVENTAJAS.

El (Centro de Alto Rendimiento de Accenture, 2012) describe las siguientes ventajas de SOA:

2.3.1 VENTAJAS.

- ***Ayuda a mejorar la flexibilidad y agilidad de las organizaciones.***

Las empresas tienen que ser capaces de poder crear y producir productos y servicios nuevos para todos sus clientes que son más exigentes. El aumento que existe entre la colaboración de los clientes y proveedores, la mayor capacidad de interpretar los datos del cliente, proporcionan a las empresas los medios necesarios que puedan diagnosticar los cambios del mercado de una forma rápida y más precisa. Lo que necesitan en estos momentos es conseguir que sus procesos de negocio sean capaces de ser adaptados al mismo ritmo. Este dinamismo exige nuevas capacidades tecnológicas que permitan adaptar los sistemas informáticos.

Al intentar crear aplicaciones nuevas para los proceso, nunca ha podido ser capaz de generar agilidad. Las aplicaciones eran desarrolladas normalmente en momentos totalmente distintos, con diferentes intenciones, conjuntos de usuarios, plataformas y niveles de servicio que suponían los diferentes ciclos de mantenimiento. Haciendo un análisis no sorprende que todos los esfuerzos necesarios para integrar aplicaciones y sistemas de las organizaciones fueran a ser muy costosos y laboriosos de implantar y mantener.

La arquitectura SOA se concentra en las capacidades y no en las aplicaciones. SOA examina la arquitectura de la empresa, incluidos las tecnologías de la información y los procesos de negocio. También el alto nivel de interoperabilidad y desacoplamiento que proporciona la arquitectura SOA permitiendo un grado alto de reutilización (externo e interno) y de parametrización. Todo redunda en la mayor facilidad y flexibilidad para adaptar y mejorar todos los procesos que tienen las organizaciones según los cambios del negocio.

- ***SOA permite personalizar masivamente las tecnologías de la información.***

La personalización en forma masiva es un concepto que se ha escogido de procesos de fabricación, donde al mezclar de manera distinta los módulos estándar, se puede formar un producto específico dentro de la infraestructura masiva de producción. La arquitectura SOA puede aplicar el similar principio a la tecnología de las organizaciones, como consecuencia de los procesos de negocio capacitados por dicha tecnología. Por ejemplo, en una compañía de telecomunicaciones, SOA ha permitido apresurar el proceso de integración y creación de los nuevos servicios, y disminuir sus costos, lo que permitido desarrollar políticas de contratación y precios mejor ajustados a segmentos específicos de los clientes. Habitualmente, la información que se necesita para desarrollar nuevos productos o servicios, SOA crea una manera más fácil, rápida y barata de acceder a esa tecnología que en el pasado. Ésta es en efecto, la capacidad de individualizar servicios y productos a escala.

- ***SOA permite la simplicidad del desarrollo en las soluciones informáticas mediante el uso de estándares de la industria y capacidades en común de la industrialización.***

SOA desacopla a los tres componentes de las aplicaciones: presentación, lógica de negocio y orquestación de procesos, estandariza la comunicación entre las capas. Todo esto favorece a que los procesos de construcción se puedan fraccionar y por lo tanto poder industrializarse fácilmente. Además, las empresas pueden focalizarse en los componentes que tienen mayor valor como los procesos y comprar el resto de componentes. (pág. 11)

- ***SOA permite aislar a los sistemas frente a los cambios generados por otras partes de la organización (protección de las inversiones realizadas.***

Al ordenar los sistemas en módulos pequeños (servicios) se reduce valiosamente el impacto de los cambios. Durante las décadas pasadas, las empresas han realizado valientes inversiones en sus infraestructuras tecnológicas.

A través de una creación del modelo flexible que pueda ser reconfigurado en función de todas las necesidades del negocio, SOA reutiliza, de una manera efectiva, los distintos sistemas tecnológicos actuales, por ejemplo, identificando la funcionalidad de los sistemas y encapsulándolos en los servicios que se puedan utilizar en distintas aplicaciones y procesos.

Las principales compañías que ofertan distintas herramientas de “discovery” están reorientando los productos para poder ofrecer la caracterización de reglas de negocio y servicios de los sistemas, para facilitar su evolución hacia la arquitectura SOA. (pág. 12)

- ***La arquitectura SOA permite ordenar y aproximar las áreas de tecnología y negocio.***

SOA ayuda en la brecha entre el sistema y la visión del negocio constituyendo un diálogo con un lenguaje en común. Las áreas del negocio se concentran en la definición de los procesos con la estrategia y el modelo de negocio de la empresa.

El área de tecnología realiza los procesos a partir del uso de servicios ya existentes y la creación de unos nuevos, cuando sea necesaria su creación. Cuando el negocio solicita cambios en los procesos que ya existen, éstos se los realiza de manera más flexible y ágil, pues están desarrollados mediante tecnología estándar y servicios que se pueden reutilizar. Por vez primer, existe una definición frecuente de las aplicaciones: los procesos, tanto el área de tecnología como el área de negocio colaboran y entienden. (p. 12)

2.3.2 DESVENTAJAS.

SOA tiene las siguientes desventajas:

- SOA depende estrictamente de la utilización e implementación de estándares. Sin estándares, la comunicación entre las aplicaciones requiere de mucho código y tiempo.
- SOA no es para las aplicaciones con un nivel alto de transferencia de los datos.
- Implica conocer todos los procesos del negocio, clasificarlos, poder extraer las funciones que son más comunes, estandarizar y formar capas de servicios que pueden ser requeridos por otros procesos del negocio.
- SOA requiere que las organizaciones cambien y esto requiere un alto esfuerzo, este cambio no es tan sencillo, para la mayoría de las organizaciones.
- La respuesta de los servicios es directamente afectada por aspectos externos como configuración, problemas en la red, entre otros.

2.4 COMPONENTES SOA.

Según la página web (Oposiciones TIC, 2012) nos dice que:

Esta arquitectura se representa en un modelo de construcción de sistemas distribuidos en la cual la funcionalidad demandada debe ser entregada a las aplicaciones a través de los servicios. En la figura siguiente se observa el esquema de la arquitectura y los elementos que tiene.

Como se puede observar, el esquema se encuentra separado en 2 zonas; una que abarca el ámbito funcional y la otra es la calidad de servicio.

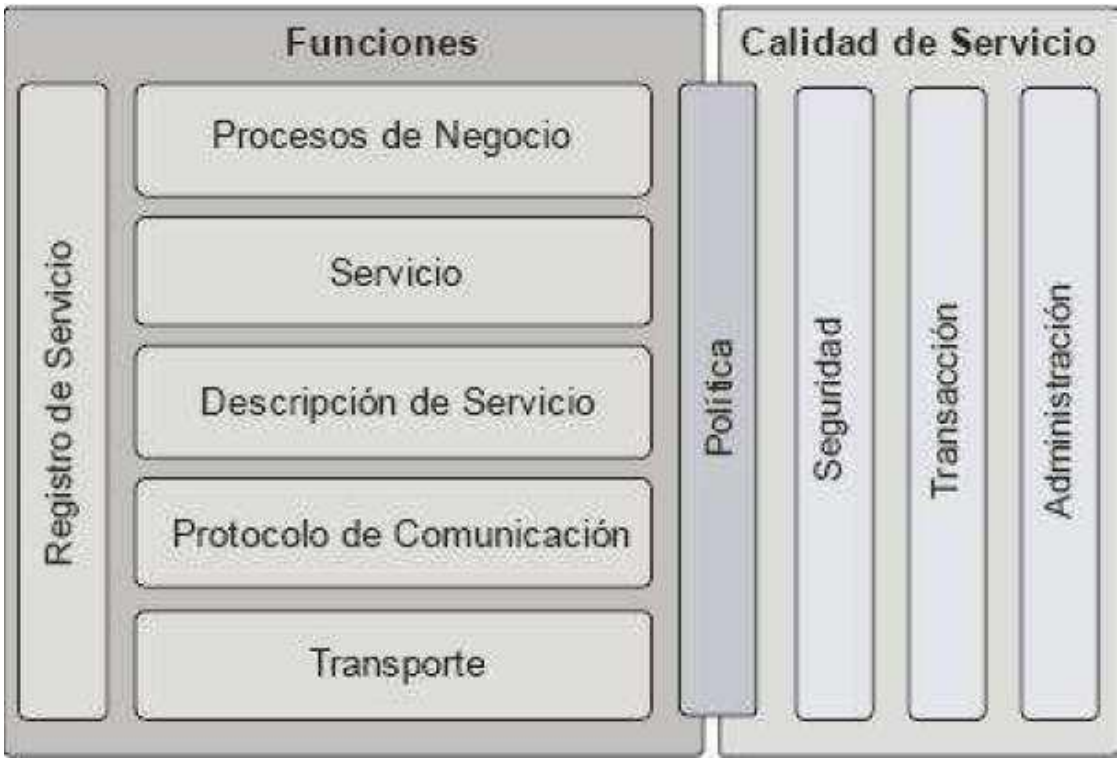


FIGURA 2.12: Elementos de SOA.
Fuente: (Oposiciones TIC, 2012)

A continuación se describen los elementos representados en la figura 2.3.

2.4.1 FUNCIONES:

- **Transporte:** es el mecanismo que sirve para llevar las demandas de los servicios desde el consumidor de servicio hacia el proveedor de servicio, y la respuesta desde el proveedor al consumidor.

- **Protocolo de comunicación de los servicios:** es un mecanismo a través del cual el proveedor de servicios y un consumidor de los servicios comunican qué está solicitando y qué está respondiendo.

- **Descripción del servicio:** es el esquema acordado para representar el servicio y cómo debe ser invocado, y qué datos son requeridos por este servicio para invocarlo con éxito.

- **Servicio:** describe el servicio que está disponible para su utilización.

- **Procesos de Negocio:** es la colección de los servicios, que son invocados en una secuencia muy particular en un conjunto específico pasos y de reglas, que satisfacen un requisito del negocio.

- **Registro de Servicios:** es el repositorio que describe los servicios y datos que pueden utilizar los proveedores para publicar los servicios, así igual que los consumidores de servicios para descubrir o hallar servicios disponibles.

2.4.2 CALIDAD DE SERVICIO:

- **Política:** es el conjunto de reglas o condiciones bajo las cuales el proveedor del servicio hace que esté disponible el servicio para consumirlo.

- **Seguridad:** es el conjunto de reglas que se aplican para la identificación, control de acceso y autorización a los consumidores de servicios.

- **Transacciones:** es un conjunto de atributos que pueden aplicarse a un conjunto de servicios para poder entregar un resultado consistente.

- **Administración:** es un conjunto de atributos que pueden aplicarse para operar los servicios que son consumidos o proporcionados.

Las contribuciones en SOA siguen el paradigma ligar, descubrir e invocar, donde el consumidor de servicios localizado dinámicamente el servicio consultando el registro de servicios para encontrar uno que pueda cumplir con un determinado criterio. Si el servicio llega a existir, el registro suministra al consumidor la interfaz de contrato y la dirección que tiene el servicio proveedor. En el diagrama se muestra las entidades (operaciones, artefactos y roles) en una arquitectura orientada a servicios donde colaboran.

FIGURA 2.13: Búsqueda de Servicios.
Fuente: (Oposiciones TIC, 2012)

Cada entidad puede escoger uno de los tres roles posibles convenientes a consumidor, proveedor y/o registro:

Un cliente consumidor de servicios es un módulo una aplicación de software u otro servicio que solicita la funcionalidad proporcionada por los servicio, y la ejecuta de acuerdo a un contrato de interfaz.

El proveedor de los servicios es la entidad asequible a través de una red que acepta y ejecuta consultas de consumidores, y publica los servicios y el contrato de interfaces en los registro de servicios para poder consumidor, descubrir y acceder al servicio.

Un registro de los servicios se encarga de posibilitar el descubrimiento de los servicios, conteniendo un repositorio de servicios que se encuentran en estado habilitado permitiendo visualizar la interfaz del proveedor de servicios a los consumidores que están interesados.

Las operaciones que se pueden realizar las entidades son:

- Publicar. Para acceder a un servicio primero se debe publicar su descripción para que un consumidor lo descubra e invoque.

- Descubrir. El consumidor de los servicios localiza el servicio que cumpla con el cierto criterio consultando los registros del servicio.

- Invocar. Una vez que se obtiene la descripción del servicio por parte del consumidor, éste lo invoca haciendo el uso de la información que presenta en la descripción del servicio.

Los artefactos que tiene la arquitectura orientada a servicios son:

- Servicio. El servicio que está en estado disponible para utilizarlo a través de una interfaz que es publicada y que permite ser invocado por el consumidor de servicios.

- Descripción de servicio. La descripción del servicio especifica que la forma en que el consumidor de servicio interactúe con el proveedor del servicio, especificando el formato de las consultas y las respuestas desde el servicio. Esta descripción especifica el conjunto de precondiciones, pos condiciones y los niveles de calidad de servicio (QoS).

2.5 DISEÑO Y DESARROLLO DE SOA.

El modelado y diseño de las aplicaciones SOA se lo conoce como el análisis y diseño orientado a los servicios. SOA es un marco de trabajo para poder desarrollar software como un marco de trabajo que se lo implementa. Para que el proyecto de SOA tenga mucho éxito los desarrolladores de software deben orientarse a una mentalidad de crear los servicios que son orquestados por el cliente o middleware que implementa los procesos de negocio.

Cuando se desarrolla unos sistemas que utilizan SOA se requiere de un gran compromiso con el modelo en los términos de la planificación, herramientas e infraestructura. (Sánchez, 2009, pág. 3)

Cuando se habla de la arquitectura orientada a servicios se refiere a servicios que pueden estar en Internet o también en la intranet, los cuales están usando web service. Existe una diversidad de estándares que se encuentran relacionados con los servicios web. Incluyen los siguientes:

- **HTTP** protocolo de transferencia de hipertextos.
- **WSDL** (Web Services Description Language) para describir la funcionalidad de un WS.
- **SOAP** (Simple Object Access Protocol) intercambio de datos.
- **XML** para la representación de datos.
- **UDDI** para la publicación o registro de los mismos.

Hay que saber que un sistema con SOA no necesariamente pueden necesitar la utilización de los estándares para ser "orientado a servicios" pero es recomendable usarlos.

En los ambiente SOA, los nodos de una red hacen que estén disponibles los recursos a otros que participantes en la red como unos servicios que son independientes a todos los que tienen acceso del modo estandarizado. Las definiciones de SOA identifican la utilización de WS (que empleando SOAP y WSDL) en la implementación, no obstante se puede utilizar SOA en cualquier tecnología basada en servicios. (Marquéz Solis, 2007, pág. 24)

2.6 CAPAS DE SOA.

A continuación vamos a describir el papel que juega cada capa en la arquitectura SOA.

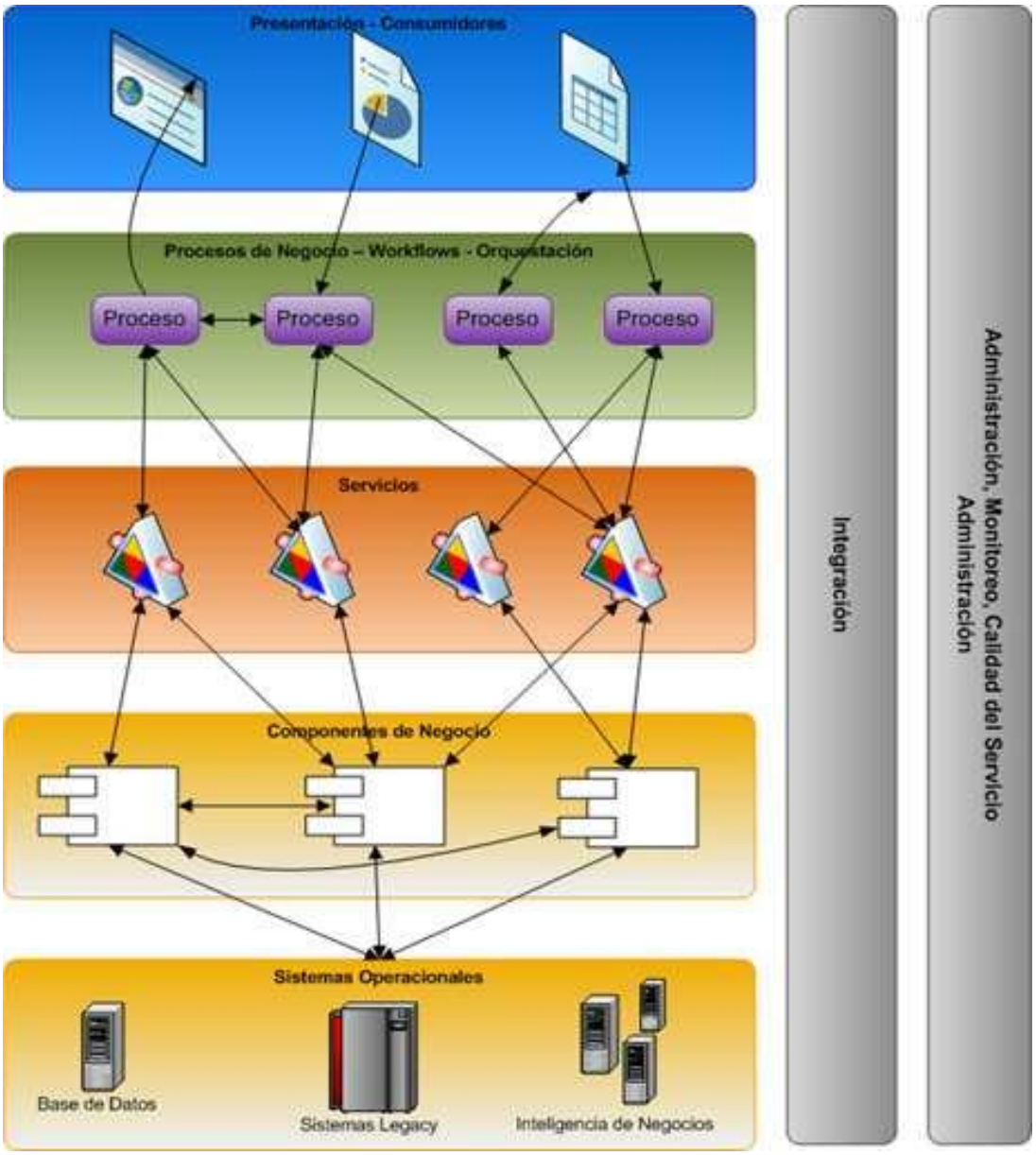


FIGURA 2.14: Capas de SOA.

Fuente: http://icomparable.blogspot.com/

En la página web (Rojas, 2009) Nos explica de las capas de SOA.

En esta figura agregamos la capa de Administración y la capa de integración, Calidad de Servicio, Monitoreo, y Administración.

Capa 1: Los Sistemas Operacionales es la capa donde las aplicaciones existen dentro de la empresa, que son conocidas como legacy systems. En esta capa podemos tener, ERP´s, CRM´s, aplicaciones de BI que están orientadas a objetos, etc. Todas las aplicaciones se integran con SOA.

Capa 2: La capa de Componentes contiene cada uno de los componentes que se encargan de ofrecer la funcionalidad que exponen los servicios. En esta capa típicamente se utilizan tecnologías para poder contener cada uno de los componentes que existen en esta capa, tales como los servidores de aplicaciones, los cuales ayudan a llevar a cabo las tareas de implementar componentes, también ayudan a manejar el balanceo de los componente y la disponibilidad.

Capa 3: La capa de Servicios es en la que residen los servicios que la organización ha decido exponer. Se los puede descubrir, también ser referenciados directamente, ser parte de la orquestación o de los servicio compuesto. Normalmente todos estos servicios exponen a la funcionalidad de negocio a través de los contratos que permiten la invocación de los componentes del negocio que se hallan en la capa de los componentes de la empresa. Estos contratos ayudan a cambiar la forma en que se realizan las tareas sin necesidad de hacer un redeploy de estos servicios expuestos.

Capa 4: Los Procesos de Negocio es la capa que expone la orquestación de los servicios. Los servicios están fuertemente ligados a los workflows por lo tanto ellos actúan como una sola aplicación. En esta capa se utilizan las herramientas visuales para construir más rápido los flujos del trabajo tales como el Biztalk Server que es un diseñador de orquestación o algunas otras herramientas de terceros que permiten crear los workflows en la notación BPEL.

Capa 5: La capa de Presentación esta capa no forma parte de SOA, pero cada día se ha vuelto más relevante. Los usuarios pueden acceder a las orquestaciones y los servicios invocando desde diferentes interfaces de usuario y pueden consumir la funcionalidad que desean.

Capa 6: La capa de Integración (ESB – Bus de Servicios Empresariales), facilita la rápida integración de los servicios a través de un conjunto de capacidades tales como mediación de protocolos, ruteo, mecanismos de transformación. Con el WSDL (Web Service Description Language) se puede especificar el binding, el cual nos ayuda a la localización del servicio que se está proveyendo. El ESB facilita la independencia de la ubicación del servicio para su integración, ya que el ESB es el que controla el ruteo de los mensajes que llegan para ser procesados.

Capa 7: La administración el Monitoreo y Calidad del Servicio en esta capa nos brinda las características requeridas para administrar, monitorear y mantener la alta calidad del servicio en áreas como desempeño, seguridad y disponibilidad. Se la llama como el SOA governancEstrategias para la implantación de SOA.

Se puede aplicar dos tipos de estrategia para la implementación de SOA: la que se denomina descendente (top-down) y la ascendente (botton-up). Ambas poseen sus propios puntos débiles que podrían poner en riesgo el éxito de todo el proyecto. Algunas organizaciones que han optado por intentar poner en marcha la infraestructura SOA aplicando el enfoque descendente han descubierto después que la infraestructura que por fin se la ha podido poner en servicio, se encuentra desconectada de todas las necesidades del negocio. Y a la inversa, el enfoque ascendente puede ser un fracaso porque puede originar una implementación caótica de los servicios creados sin tener en cuenta a los objetivos de la organización. (Microsoft Corporation, 2011, pág. 9)

2.7 ELEMENTOS DE SOA.

Los componentes que forman parte de una Arquitectura Orientada a Servicios (SOA) son:

- Consumidores.
- Repositorio de Servicios.
- Servidores.
- Bus de servicios.

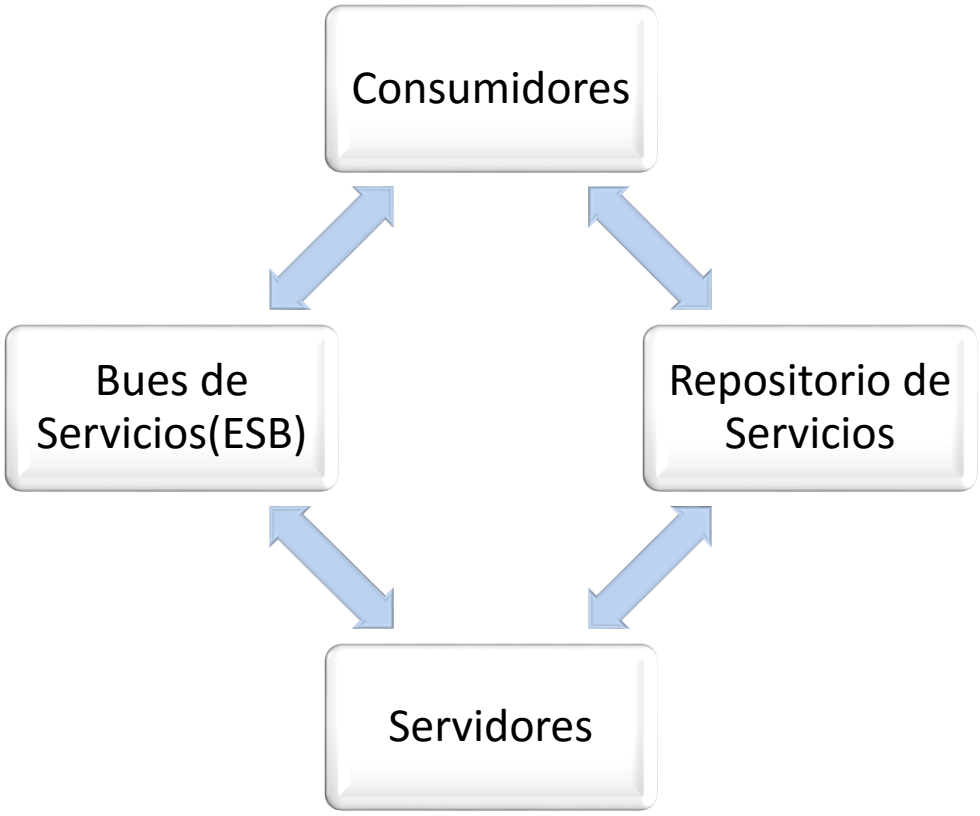


FIGURA 2.15: Elementos de SOA.

Fuente: Propia

2.7.1 SERVIDORES.

Un servicio de negocio es aquel componente reutilizable del software, con significado funcional muy completo, y que se compone por:

- **Contrato**: especificaciones de la funcionalidad, restricciones y forma de uso del servicio.
- **Interfaz**: mecanismo que expone los servicios a los usuarios.
- **Implementación**: contiene la lógica y los accesos a los datos.

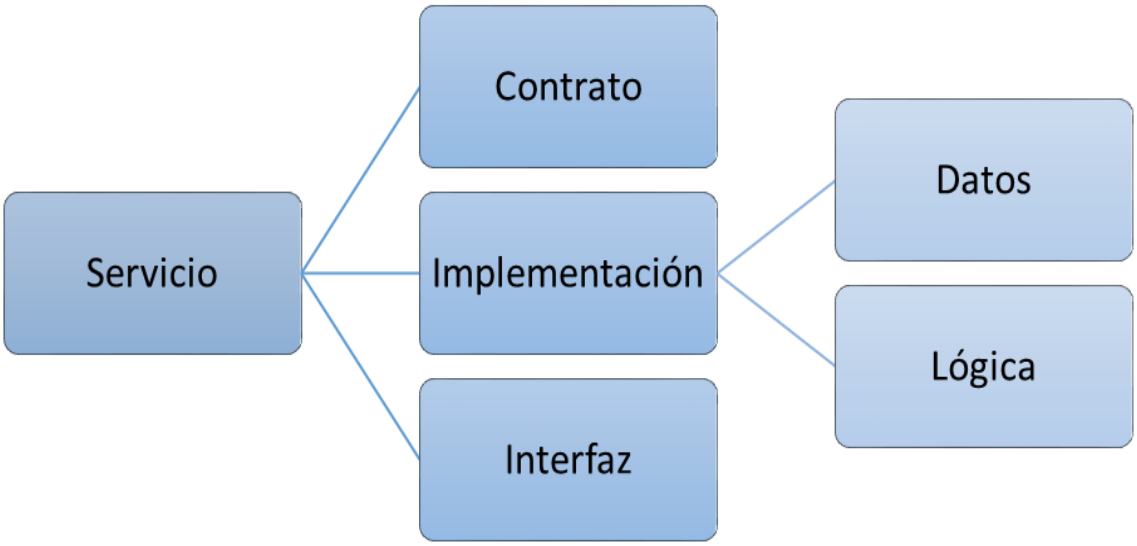


FIGURA 2.16: Elementos de SOA Servidores.

Fuente: Propia

2.7.2 REPOSITORIO DE SERVICIOS.

Un repositorio de servicios nos proporciona muchas facilidades para poder descubrir servicios y de una mejor manera adquirir la información que sea necesaria para su uso.

Además de tener la propia información de los contratos, los repositorios nos pueden proporcionar la información de:

- Personas de contacto.
- Restricciones técnicas.
- Acuerdos de nivel de servicios.

2.7.3 BUS DE SERVICIOS EMPRESARIAL (ESB).

2.7.3.1 DEFINICIONES.

El término ESB (Bus de Servicios Empresariales) se lo utiliza con mucha frecuencia en el contexto de la implementación de las funcionalidades de mensajería en la infraestructura orientada a los servicios.

Un ESB es uno de los numerosos mecanismos de construcción que sirve para poner en la marcha a una infraestructura orientada a servicios. Las capacidades necesarias que debe tener la mensajería en las infraestructuras orientadas a servicio exigen una ampliación de todas las funcione y son tradicionalmente conocidas por Enterprise Application Integration y la Message Oriented Middleware para que se los pueda incluir en el soporte de primera clase de los estándares en los Servicios Web e integración con los otros componentes de la infraestructura de servicios como la gestión de políticas, registro de los metadatos y el entorno de monitorización operativos y del negocio. Dado a la heterogeneidad del conjunto de las infraestructuras corporativas y los importantes esfuerzos de la inversión realizados ya en elementos Enterprise Application Integration y Message Oriented Middleware, que también tienen una gran importancia en el ESB y pueda aprovechar al máximo todas las posibilidades que estos entornos ofrecen en un mundo orientado a servicios. (Microsoft Corporation, 2008, pág. 1)

FIGURA 2.17: Bus de servicios Empresarial (ESB).
Fuente: http://www.tcpsi.com/

Un ESB (bus de servicios empresariales) es una de las soluciones de integración distribuida, que se basa en los mensajes y en los estándares abiertos. La función principal de un ESB es la de proporcionar una comunicación confiable entre los distintos recursos de tecnología así como las aplicaciones, plataformas y los servicios, que están distribuidos en los múltiples sistemas de la empresa.

A medida que cada departamento de TI se centra cada vez más en un diseño de SOA para reducir los costos del desarrollo y para ampliar la agilidad de los negocios, los ESB se han convirtiendo en el primer paso clave para poder establecer SOA en la empresa. Los ESB forman parte de los cimientos de SOA y se los puede complementar con la capacidad de productividad, como la orquestación de los servicios y los registros. (Tibco, 2008, pág. 2)

El gran desafío de la Arquitectura Orientada a Servicios es poder resolver la escalabilidad que tienen las conexiones punto a punto, donde el número de las conexiones tiende a crecer exponencialmente por cada una de las aplicaciones que se añadan. Con la utilización de un ESB (Bus de Servicio Empresarial) cada una de las aplicaciones se conecta por una sólo vez a la infraestructura troncal común. Esto ayuda a reducir al mínimo las conexiones existentes y suministra una ubicación centralizada para su fácil administración y para las gestiones de sistemas integrados y de arquitecturas (Gras, 2012)

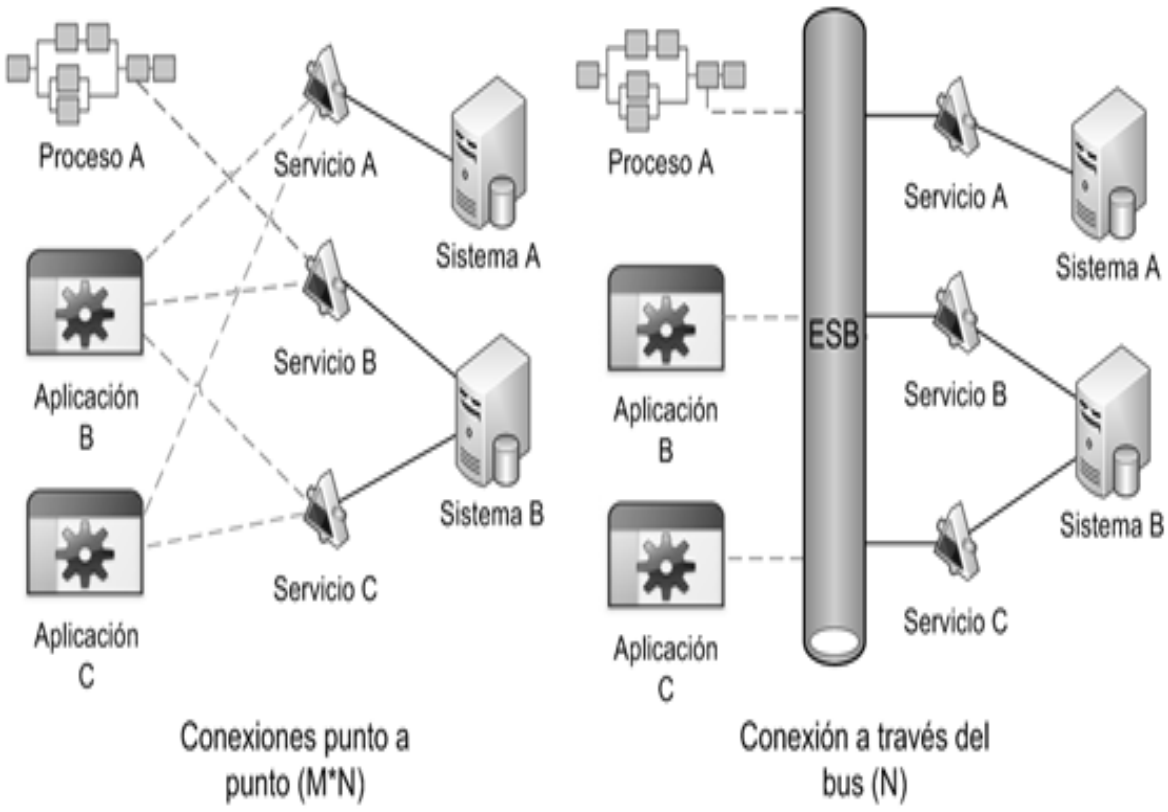


FIGURA 2.18: Conexión Punto a punto y Conexión a través de ESB.

Fuente: http://horizontesbpm.blog.com/?p=111#_ftn2

Un ESB brinda muchas características, el ESB proporciona la plataforma de integración basada en los estándares que combinan mensajería, transformación de datos servicios Web, y el enrutamiento inteligente. En el ESB las aplicaciones y los servicios están unidos en la Arquitectura Orientada a Servicios, permitiendo la operación de una manera independiente (Gras, 2012)

2.7.3.2 CARACTERÍSTICAS DE UN ESB.

Las siguientes características fueron descritas en la página web (TCP, 2014).

- Son independientes respecto a los sistemas operativos y a los lenguajes de programación.
- Utiliza XML como el lenguaje estándar de comunicación.
- Soporta los estándares de Servicios Web.
- Posee adaptadores para realizar una fácil integración con aplicaciones.
- Tiene un modelo de seguridad estándar que ayuda a autorizar, autentificar y auditar la utilización del ESB.
- La transformación de los mensajes.
- Enruta los mensajes aplicando reglas del negocio en función del contenido del mensaje.
- Manipulación de las excepciones.
- La validación de los mensajes.
- Soporta el encolado y mantenimiento de los mensajes, si las aplicaciones no se encuentran disponibles.
- Monitoriza el sistema y la actividad del negocio.

2.7.3.3 REQUISITOS A ESCALA EMPRESARIAL DE UN ESB.

En el documento (Tibco, 2008) describe las funciones que son los elementos esencial para una la integración satisfactoria de SOA.

- ***Mensajería distribuida.*** El núcleo que poseen los ESB lo constituye la aplicación de middleware que es orientada al mensaje. El núcleo proporciona una técnica de transporte confiable y distribuido que emplea el mecanismo de almacenamiento y el reenvío gracias al cual se puede garantizar la entrega de los mensajes incluso cuando ocurre una anomalía en la red.

- ***Transparencia en las ubicaciones.*** Con la mediación entre los servicios, uno de los servicios del cliente que invoca al proveedor solo debe saber que existe el servicio; el cliente no necesita conocer dónde y cómo se está ejecutando el servicio. Un ESB localiza al servicio cuando se lo invoca. Esto ayuda a proporcionar un cierto nivel de virtualización del servicio y de la transparencia de la ubicación, de forma que si el equipo falla, o si se la cambia a la ubicación del proveedor de servicio, no es necesario notificar el cambio a cada cliente. Esto puede contribuir a minimizar los riesgos y la reducción del costo de gestión de las TI.

- ***Transparencia del transporte.*** En el enfoque tradicional de la integración que es punto a punto los componentes y los objetos están estrechamente acoplados. En SOA, los servicios se encuentran repartidos por todo el entorno de las TI y su acoplamiento es estricto en menor grado, gracias a que hay la transparencia de ubicación. Además de que se apoya en las transparencias de las ubicaciones para poder conectar clientes y los proveedores de los servicios, el ESB es el encargado de proporcionar un protocolo de trasporte físico el cual facilita la comunicación entre los servicios utilizando los transportes diferentes.

- ***Soporte multiprotocolo.*** Debido a que este plantea cuestiones de confiabilidad y únicamente funciona con los patrones de intercambio de los mensajes (MEP) sincrónicos, el modelo del transporte HTTP no ayuda a

satisfacer los requisitos de los servicios y de las aplicaciones. Como por ejemplo, el servicio del mensaje de Java (JMS) además posee características asincrónicas, ofreciendo más fiabilidad en los transportes de HTTP. Para poder ver el comportamiento que tienen las aplicaciones individuales, algunos de los sistemas recurren a SOAP a través de los JMS. También se utiliza otro tipo de modelo de transporte, entre los que se encuentran los sistemas de transporte de propietarios de varios y de principales proveedores de los sistemas y soluciones de las planificaciones de los recursos empresariales. El ESB tiene que ser capaz de soportar varios tipos de sistemas de transporte para la integración de sistemas dispares y poder gestionar el transporte de las comunicaciones complejas de una manera eficaz.

- ***Calidad del servicio.*** En todas las aplicaciones empresariales, es necesario la calidad de servicio (QOS) ya que hace referencia fundamentalmente la confiabilidad. Las entregas que se hace de los mensajes y a la fiabilidad que tienen los servicios de invocación son las funciones de misión muy crítica en cualquier sistema, ya que los servicios Web por sí solos aun no ofrecen un servicio de entrega que sea garantizado. El ESB, por otra parte, puede proporcionar un servicio que sea garantizado y de muy alta fiabilidad en la entrega del mensaje de un principio a un fin que va más allá de la fiabilidad que estos pueden ofrecer en el transporte como JMS. Así también, los métodos que se emplean para alcanzar un nivel alto de QOS deben satisfacer a los estándares que existen, por ejemplo, tener compatibilidad con las especificaciones WS-Reliable Messaging.

- ***Patrones del intercambio de mensajes*.** La mayoría de los ESB son basados en el paradigma de solicitud/respuesta usando los SOAP sobre HTTP; esto se refiere a que el servicio cliente envía un mensaje de solicitud al cliente y espera recibir la respuesta. Esto se lo llega a conocer como un MEP sincrónico. Sin embargo, en los MEP de publicación/suscripción, el servicio del cliente puede enviar el mensaje y suscribirse a la respuesta, en vez de estar esperando a recibirla.

- ***MEP de la publicación/suscripción.*** Se pueden responder de forma eficaz en los eventos en el contexto empresarial, cuando el ciclo de vida de la acción del servicio tiende a durar un prolongado tiempo, el ESB debe ser capaz de poder manejar ambos paradigmas.

- ***Enrutamiento que se basa en el contenido.*** Hay dos tipos de enrutamiento dentro del ESB. El primer servicio de enrutamiento llega a producirse cuando la invocación del servicio entra en el ESB y éste orienta la respuesta al proveedor del servicio, sin la necesidad de que el servicio cliente sepa la ubicación del proveedor del servicio. El otro enrutamiento se basa en el contenido, se introduce una serie de reglas o una lógica del negocio que aplica al contenido del mensaje en esta etapa del enrutamiento y que hacen viable que el ESB encamine los mensajes a los proveedores de servicio basándose en los contenidos; dando la prioridad, por ejemplo, a los pedidos que realizan determinados clientes también marcando los pedidos que tienen gran tamaño para darles el tratamiento especial que necesiten. Esto hace que el servicio sea muy valioso para las empresas, ya que contribuye a reducir el costo de la gestión, garantizando que se respeten todos los acuerdos a nivel de cada servicio y permitiendo a las empresas centrarse en las mejorar de actividades con satisfacción de sus clientes.

- ***Transformación.*** Si la tarea del ESB es dirigir los mensajes de un servicio al siguiente, en ocasiones el formato de los datos del servicio no satisface los requisitos necesita el siguiente servicio. Por tal motivo, el ESB debe ser apto en la transformación de los datos de un formato a otro.

- ***Orquestación de servicios.*** La herramienta ESB permite la orquestación de servicios, de modo tal que se puedan desarrollar procesos que solo incorporen actividades automáticas y que se puedan constituir servicios de negocio.

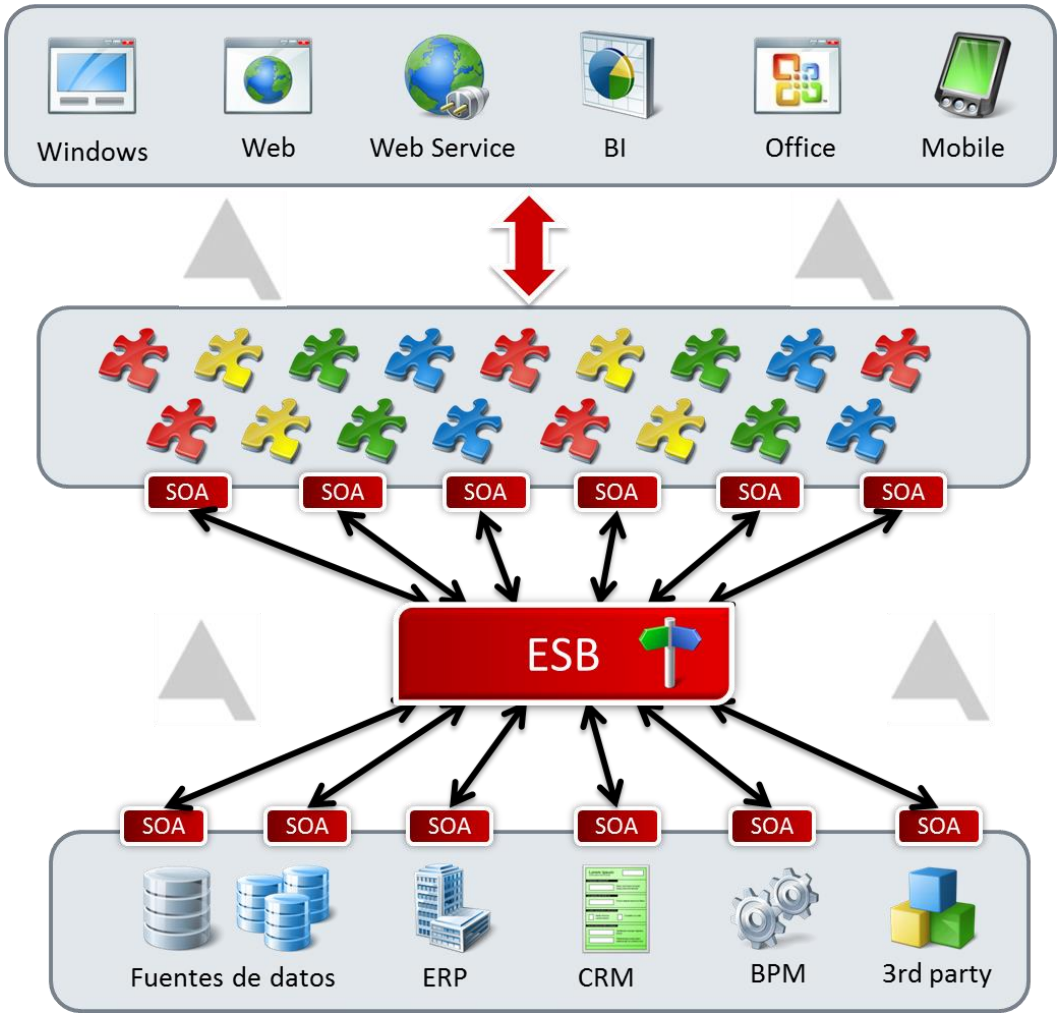


FIGURA 2.19: Negocio de integración de aplicaciones con ESB.
Fuente: http://innovacionsectoraseguradorblogspot.com/2012/04/la-integracion-de-la-movilidad.html

Mensajes XML.

Las aplicaciones interactúan con mensajes XML, que introducen y sacan los contenedores de servicios en puntos extremos. Las aplicaciones no necesariamente necesitan ser conscientes de que hay protocolos de comunicaciones subyacentes ni de localizaciones físicas; tan sólo pueden ver simples "cajas" que entran y salen. Así, los servicios podrían ser trasladados, reemplazados o actualizados sin tener que interrumpir a los sistemas del negocio ni tener que modificar a las aplicaciones. La utilización del XML en el ESB proporciona gran flexibilidad y hace que la infraestructura

sea más resistente a los cambios que existen en los negocios y las aplicaciones.

Por ejemplo, al usar hojas de estilo XML el ESB puede enviar los contenidos del mensaje de un formato a otro. Las aplicaciones no requieren adherirse a un formato que sea específico ni hay que enviar a los datos a un punto que sea central para su transformación. (IDG Communications, 2012)

El ESB trata a las aplicaciones como servicios, con independencia de cómo sea la conexión al bus, permitiendo que las empresas migren gradualmente a la arquitectura basada en servicios con un riesgo que sea mínimo y con una eficaz planificación de las inversiones. (IDG Communications, 2012)

2.7.4 CONSUMIDORES DE SERVICIOS.

Los consumidores de servicios son aquellos elementos de una arquitectura SOA que realizan las siguientes cosas:

- Descubren los servicios a través del repositorio.
- Realizan las llamadas a los mismos servicios de acuerdo al contrato que tienen y a través de una interfaz definida para tal efecto.

CAPÍTULO III

3 INTRODUCCIÓN MULE ESB.

Mule ESB es una de las primeros y exitosos ESB de código abierto. Tienen unas funcionalidades que son similares a los demás ESB de código abierto. Mule ESB incluye una instalación sencilla e intuitiva, tiene las herramientas basadas en Eclipse. Por lo general, los ESB de código abierto son soluciones muy ligeras y extensibles. Aparte de la versión de código abierto, existe una versión comercial que está disponible en su página web.

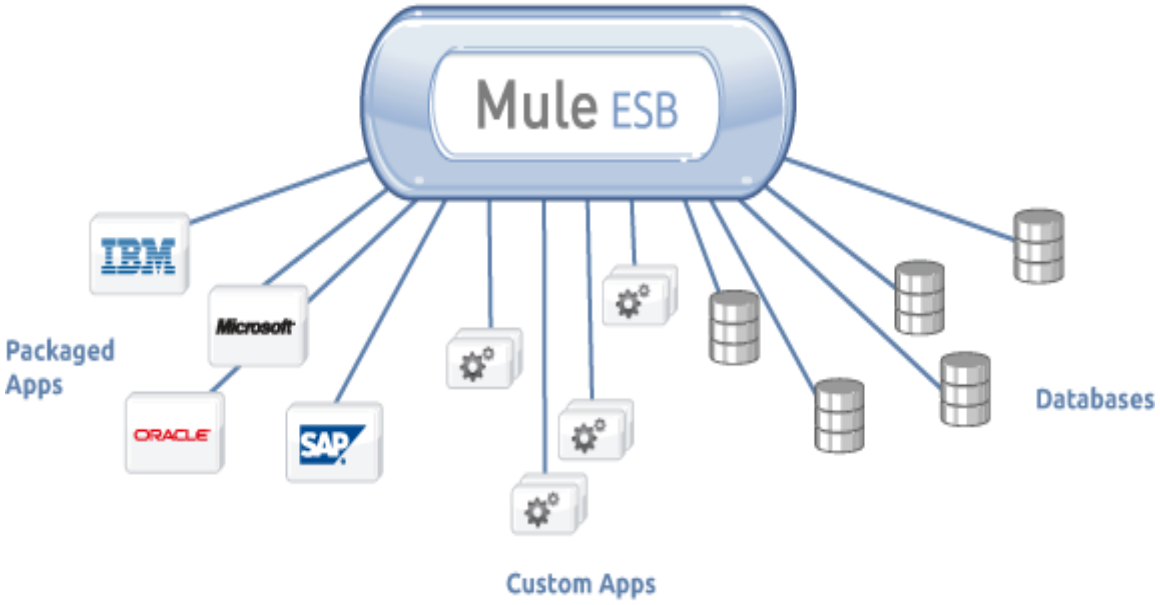


FIGURA 3.20: Estructura de un negocio con Mule ESB.
Fuente: http://blogs.mulesoft.org/welcome-to-mule-enterprise/

Para las personas que todavía no lo conocen, sobre lo que es código abierto, esto no quiere decir que sea gratis. Incluso los vendedores de software de código abierto tienen que ganar dinero y no puede desarrollar productos y ofrecer apoyo de forma gratuita, sin embargo, los precios son mucho más amigables con los clientes, la versión de código abierto puede ser utilizada (incluso en producción) sin ningún costo de licencia. A menudo, la versión de código abierto se la utiliza para realizar pruebas de concepto.

3.1 DEFINICIONES.

En la página web oficial (MuleSoft, 2013) define a Mule ESB de la siguiente manera.

Mule ESB es un bus de servicios basado en Java es ligero y tiene una plataforma de integración que ayuda a los desarrolladores a conectar las aplicaciones de forma rápida y fácil, lo que les permite utilizar el intercambiar de datos.

Mule ESB es de una fácil integración en los sistemas existentes, sin tener en cuenta la variedad de tecnologías que pueden utilizar las aplicación, tales como:

- JMS.
- Servicios Web.
- JDBC.
- HTTP.

3.2 ARQUITECTURA.

En la página web (Ramos, 2012) describe la siguiente arquitectura de Mule ESB.

Con Mule ESB podemos integrar nuestros procesos del negocio de una forma transparente entre ellos, por lo cual conseguimos tenerlos completamente desacoplados.

El bus actúa como mediador entre los diferentes servicios del negocio y se encarga de poder conectarlos entre sí mediante los mensajes de comunicación de estándares. La gran ventaja de un bus es que tenemos que definir la manera de poder integrar servicio con el bus seleccionado, por lo que nos aislamos de la configuración del resto de servicios.

La forma en la que se configura Mule se realiza mediante el único fichero llamado mule-config.xml. Este fichero se parece a la forma en que se configurar Spring porque Mule está montado sobre Spring.

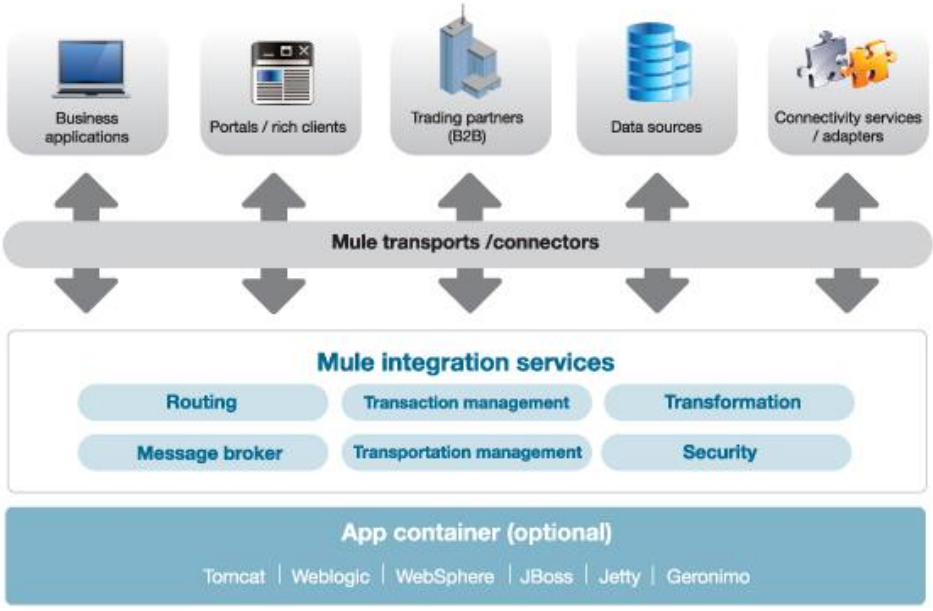


FIGURA 3.21: Servicios de MULE ESB.

Fuente: http://www.mulesoft.org/what-mule-esb

Los más principales módulos de Mule ESB son:

- **Componentes:** Cualquier objeto, POJO, servicio REST, EJBs, Bean de Spring son los que incluyen la lógica del negocio.
- **Inbound routers:** Es encargado de controlar al mensaje recibido para poder direccionarlo a un servicio.
- **Outbound routers:** Es encargado de redirigir la salida hacia otro servicio. Se lo puede utilizar como un balanceador de carga o enrutamiento que se basa en reglas.
- **Endpoints:** Son utilizados para la conexión de componentes con otros sistemas que son externos.
- **Transformadores:** Son los encargados de gestionar y transformar los datos de un formato a otro para redirigirlo a la entrada del componente.

- **Transporte:** Se encargan de poder manejar la forma en que se conectan entre si las diferentes tecnologías con utilización de diferentes protocolos de transporte: HTTP, JMS, FTP, etc.

3.3. VENTAJAS Y DESVENTAJAS

En la página web oficial (MuleSoft, 2013) se presentan las siguientes ventajas y desventajas de Mule ESB.

3.3.1 VENTAJAS:

La principal ventaja de Mule ESB es permitir la comunicación de diferentes aplicaciones entre sí, actuando como un sistema que transporta datos entre aplicaciones dentro de la empresa a través de Internet. Mule ESB incluye potentes capacidades que incluyen:

La creación de servicios y hospedaje: exponer y alojar servicios reutilizables, utilizando Mule ESB como un contenedor de servicio ligero.

Mediación de servicios: escudo de formatos de mensaje y protocolos, la lógica de negocio separada de mensajería y permitir las llamadas de servicio independientes de la ubicación.

El enrutamiento de mensajes: enrutar mensajes, filtrar, agregar, y volver a la secuencia en base al contenido.

La transformación de datos: intercambian datos a través de diferentes formatos y protocolos de transporte.

3.3.2 DESVENTAJAS.

Al igual que cualquier otro modelo de arquitectura que utiliza un motor central, es que puede llegar a ser un punto único de fallo de la red. Es responsable de toda la concurrencia entre las series y los estados de datos de la aplicación, todos los mensajes entre los solicitantes deben pasar a través de él.

Bajo carga pesada, el corredor puede convertirse en un cuello de botella para los mensajes. Un solo destino central para todos los mensajes también hace que sea difícil utilizar con éxito a través de grandes distancias geográficas.

Las implementaciones son a menudo sus productos propios de peso pesado, destinados a apoyar subconjunto de un proveedor específico de la tecnología. Esto puede presentar problemas si su escenario de integración se refiere a productos de varios proveedores, los sistemas desarrollados internamente, o productos heredados que ya no son compatibles con el vendedor.

Mule ESB fue desarrollado para llevar todo el proceso de integración. Si la arquitectura de integración es la siguiente:

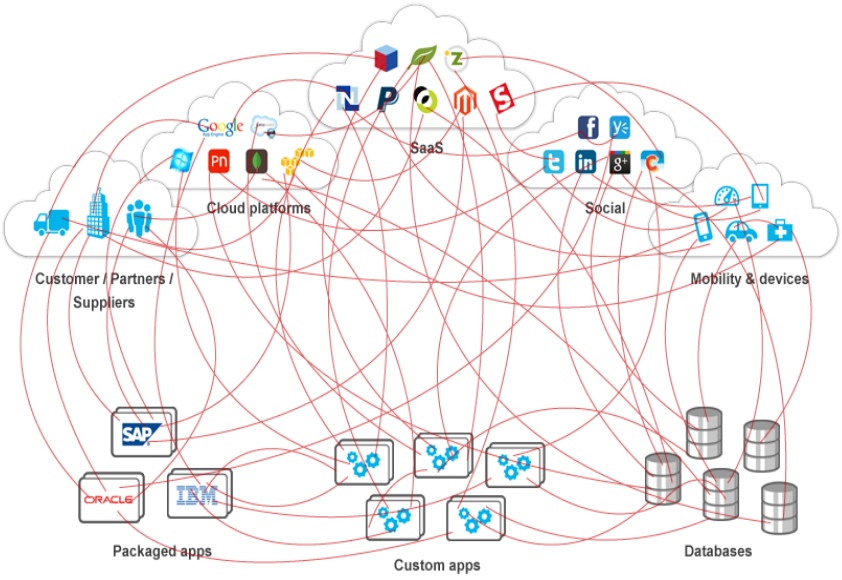


FIGURA 3.22: Arquitectura de integración de punto a punto.
Fuente: http://www.mulesoft.org/documentation/display/current/Meet+Mule

Con Mule ESB se puede conectar todo este sin número de conexiones que están de punto a punto y poder tener una arquitectura mejorada. La implementación de un bus de servicios empresarial como lo es Mule ESB elimina la dificultad de conectar todas las aplicaciones que utiliza en su negocio, ya sean en la intranet o en la nube. (MuleSoft, 2013)

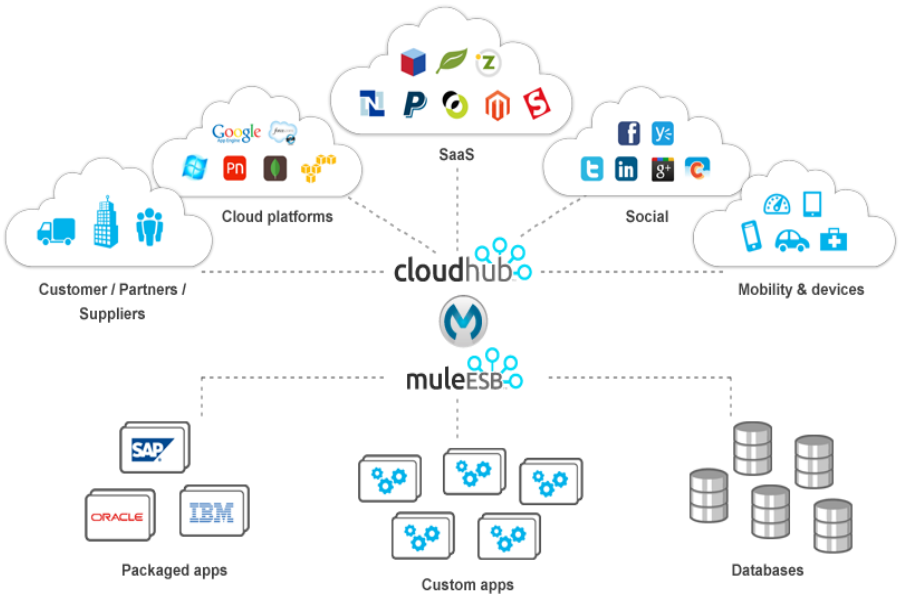


FIGURA 3.23: Arquitectura de integración utilizando Mule ESB.
Fuente: http://www.mulesoft.org/documentation/display/current/Meet+Mule

3.4 CARACTERÍSTICAS.

En la página web oficial (MuleSoft, 2013) se presentan las siguientes características de Mule ESB.

- Integrar aplicaciones o sistemas en las instalaciones o en la nube.
- Utilizar conectores out-of-the-box para crear integración SaaS aplicaciones.
- Construir y exponer las API.
- Consumir API.
- Crear servicios Web que orquestan las llamadas a otros servicios.
- Crear interfaces para exponer aplicaciones para el consumo móvil.
- Integrar B2B con soluciones que sean seguras, eficientes y rápidos de construir y desplegar.

- Cambiar aplicaciones en la nube.
- Soporte para más de 30 protocolos y tecnologías.
- Modelo de programación basado en POJO simplificado y aprovechando los desarrollos en conjuntos de habilidades existentes para el despliegue rápido.
- Soporte para múltiples puntos de acceso, tales como JMS, JDBC, y SOAP.
- No depende de los protocolos propietarios específicos del proveedor.
- Facilidad de uso de los servicios, se puede configurar fácilmente en un archivo de configuración.
- Tamaño reducido: la memoria y el disco, no requiere servidor de aplicaciones.

FIGURA 3.24: Características de Mule ESB.

Fuente: http://www.mulesoft.org/documentation/display/current/Mule+Fundamentals

3.5 BUS DE SERVICIOS EMPRESARIAL MULE ESB.

Mule ESB es ampliamente el ESB más utilizado de código abierto del mundo, con más de 1,5 millones de descargas y 2.500 despliegues de producción. Con el modelo de desarrollo simplificado de Mule ESB y arquitectura ligera, los desarrolladores pueden ser productivos en cuestión de minutos, creando con facilidad la integración de servicios de aplicación.

Mule ESB elimina la complejidad de la integración, lo que facilita a los desarrolladores a construir fácilmente el alto rendimiento, seguridad, interacciones multi-protocolo entre los sistemas y servicios heterogéneos. (MuleSoft, 2013)

3.5.1 INSTALACIÓN MULE ESB.

1. Ingresamos a la página web de www.mulesoft.com, vamos a “Products” y seleccionamos “for SOA”.

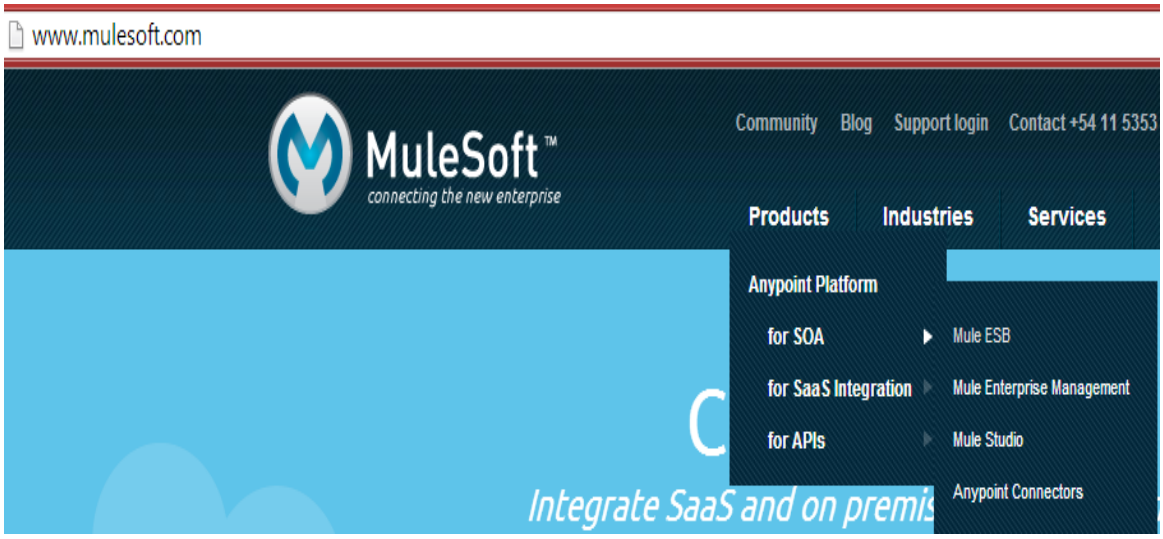


FIGURA 3.25: Pagina de www.mulesoft.com.

Fuente. Propia

2. En la página de Mule ESB hacemos clic en “Download” para descargar el software de Mule ESB.

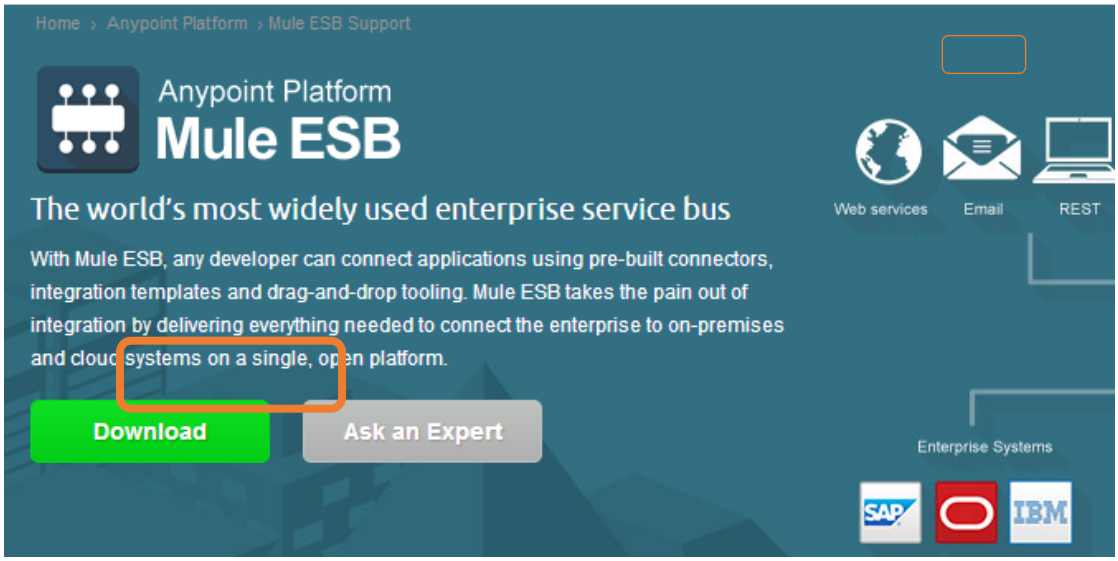


FIGURA 3.26: Enlace de descarga de Mule ESB.

Fuente. Propia

3. En la siguiente página escogemos la opción “Designtime” y la opción para Windows.

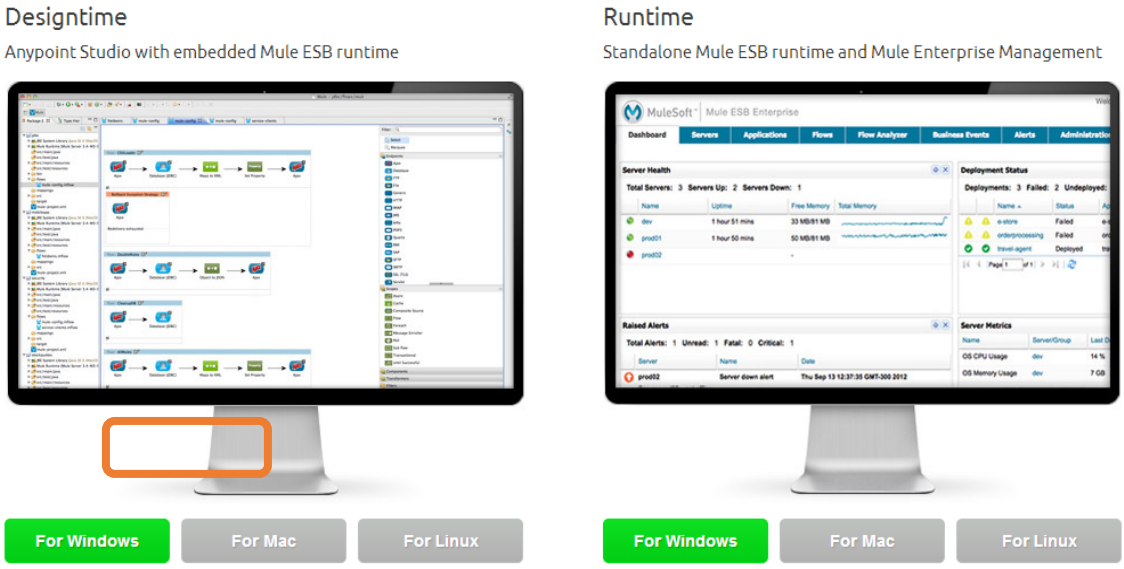


FIGURA 3.27: Opciones de descarga de Mule ESB.

Fuente. Propia

4. Una vez descargado el software lo descomprimimos en una carpeta donde lo vamos a utilizar, ejecutamos el archivo “AnypointStudio”.

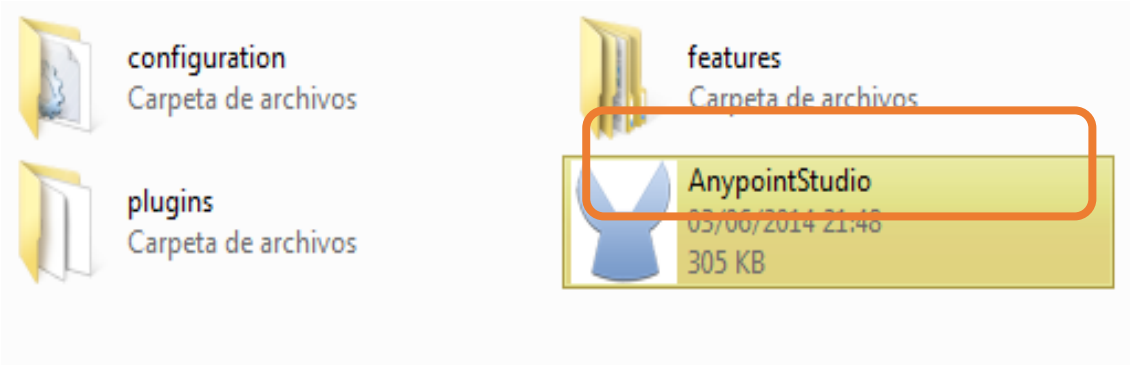


FIGURA 3.28: Carpeta del instalador de Mule ESB.

Fuente. Propia

3.5.1.1 EJEMPLO PRÁCTICO DE MULE ESB.

El objetivo de este tutorial es construido un ejemplo práctico, se construirá una aplicación muy simple que se nos haga fácil de entender el funcionamiento de Mule ESB.

Crear un nuevo proyecto.

1. En AnyPoint Studio, hacer clic en **file** y seleccionar **New > Mule Project. Studio** abre el nuevo proyecto Mule asistente. En el campo nombre, escribir *Mi Primer Proyecto*, a continuación, haga clic en Finalizar.

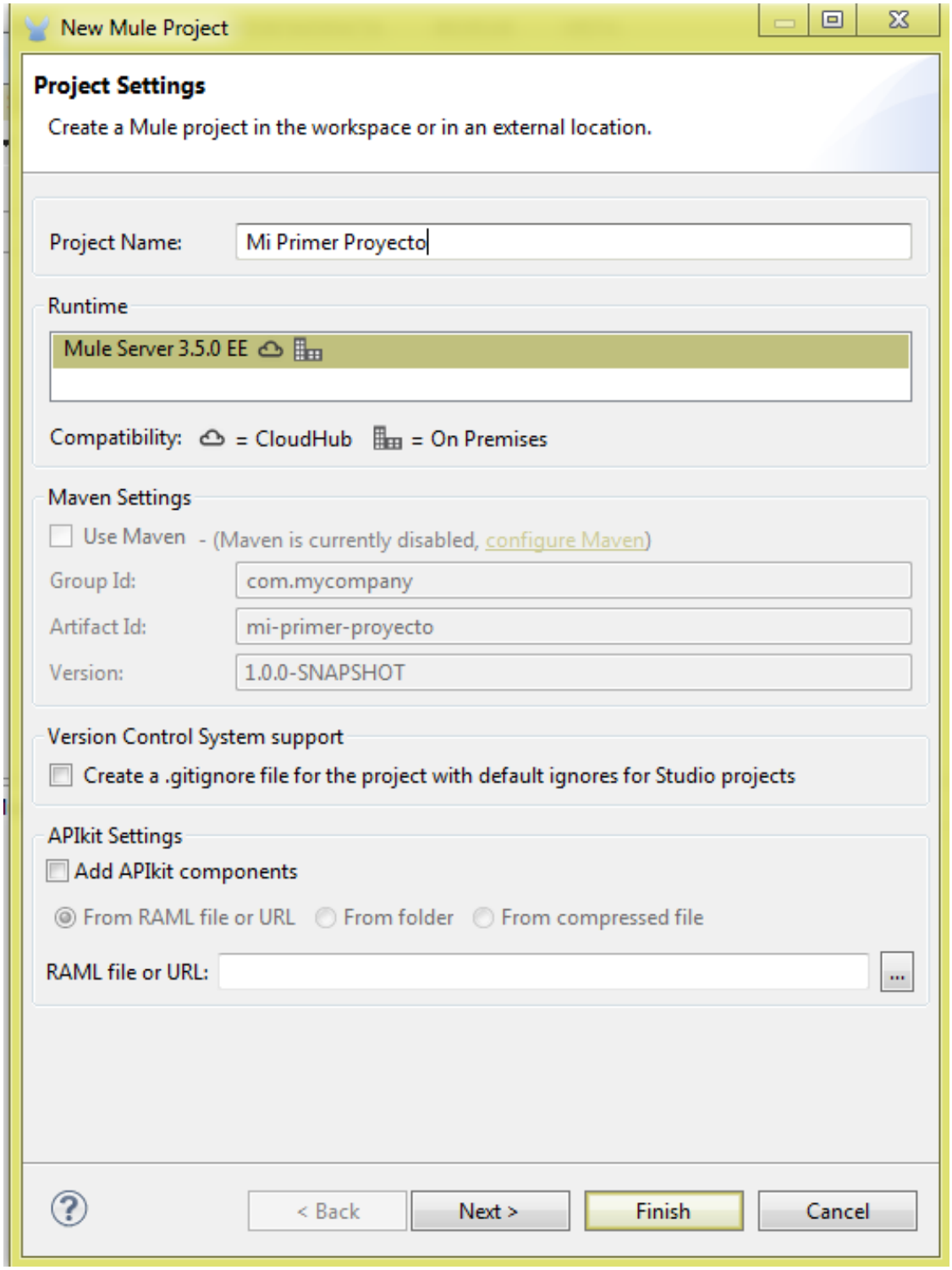


FIGURA 3.29: Nuevo proyecto de Mule ESB.

Fuente: Propia

2. AnyPoint Studio abre un lienzo en blanco. Comenzar el diseño del proyecto arrastrando y soltando un **HTTP Connector** desde la **palette** al lienzo. Se observa que AnyPoint Studio ajusta automáticamente el conector con un **flujo**, lo que le ahorra el paso de crear manualmente.

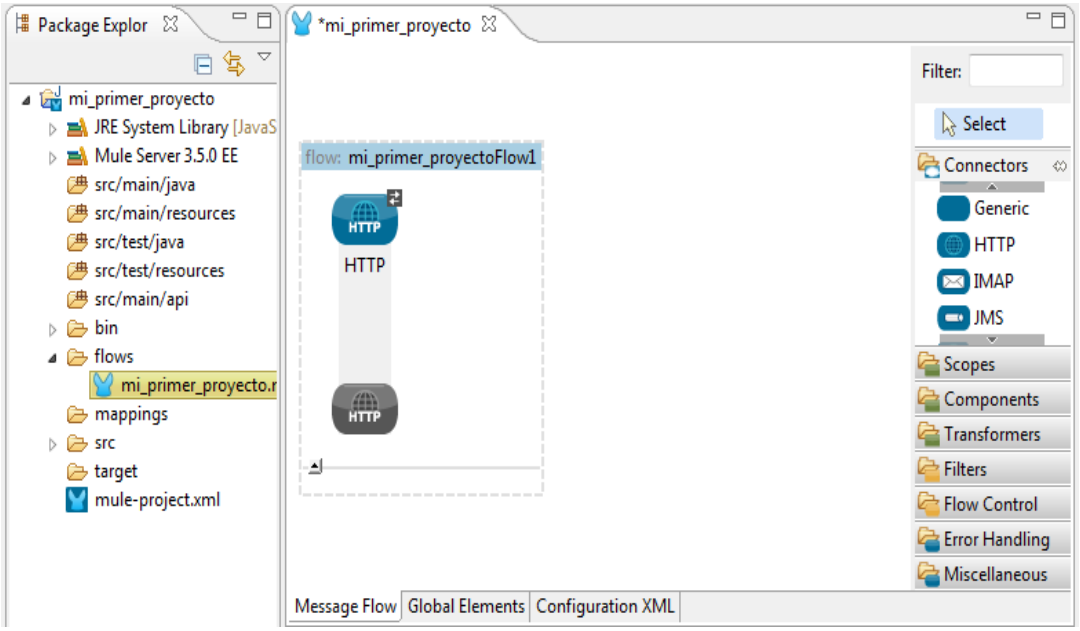


FIGURA 3.30: Lienzo del proyecto con un conector HTTP.

Fuente: Propia

3. Añadir a su flujo con un **Echo Component**, arrastrándolo cerca del conector HTTP en el lienzo.

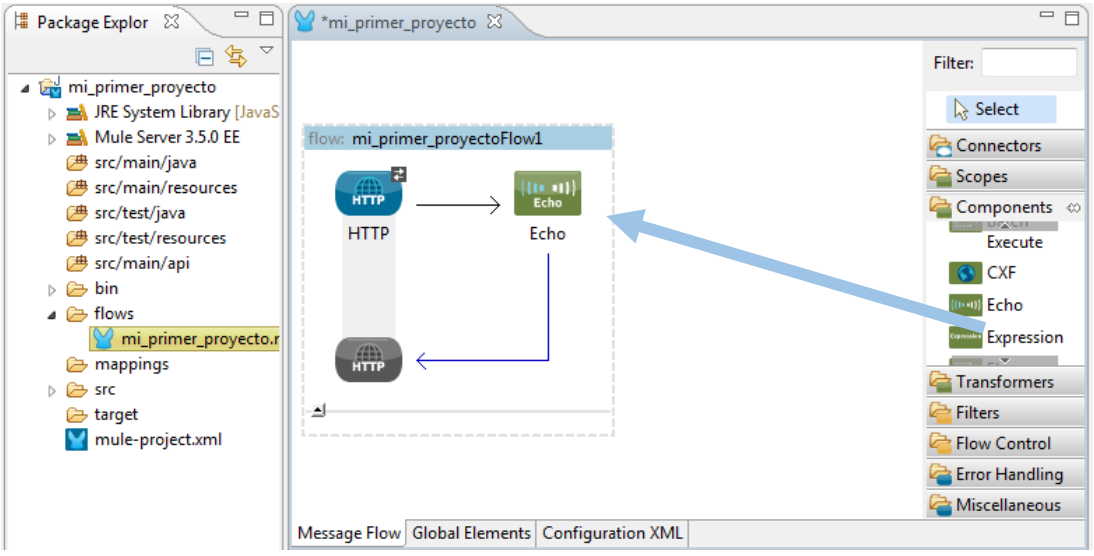


FIGURA 3.31: Agregar un componente Echo.

Fuente: Propia

Hacemos clic en la pestaña **Configuration XML,** para ver la aplicación representado en XML. Se puede ver por medio de dos vías para asegurarse que cualquier cosa que agregue o cambio en la interfaz gráfica se refleja en la configuración XML.

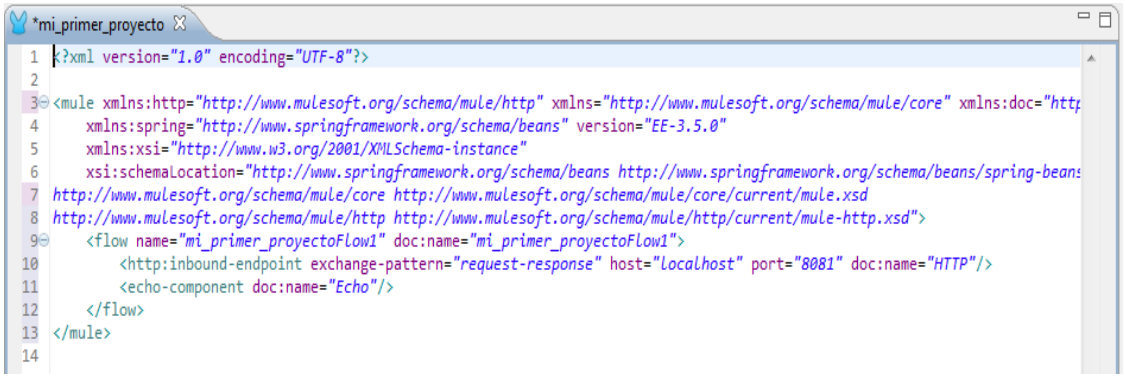


FIGURA 3.32: Configuración XML del proyecto.

Fuente: Propia

4. Hacer clic en la ficha **Message Flow** para volver al editor gráfico, haga clic en el icono **guardar** para guardar el proyecto.

Configurar el proyecto.

Ahora que se ha modelado una aplicación básica en Mule, se puede hacer los siguientes pasos para configurar cada elemento creado.

1. Haga clic en el conector HTTP en el lienzo para abrir su editor de propiedades en la consola.

2. El conector HTTP permite a la aplicación de mule poder conectarse a los recursos web a través del protocolo HTTP o HTTPS. En una aplicación más compleja, deberá utilizar este editor para configurar los valores de los atributos con el fin de definir los detalles de la conexión de la aplicación. Para este proyecto, dejaremos los valores predeterminados para todos los campos.

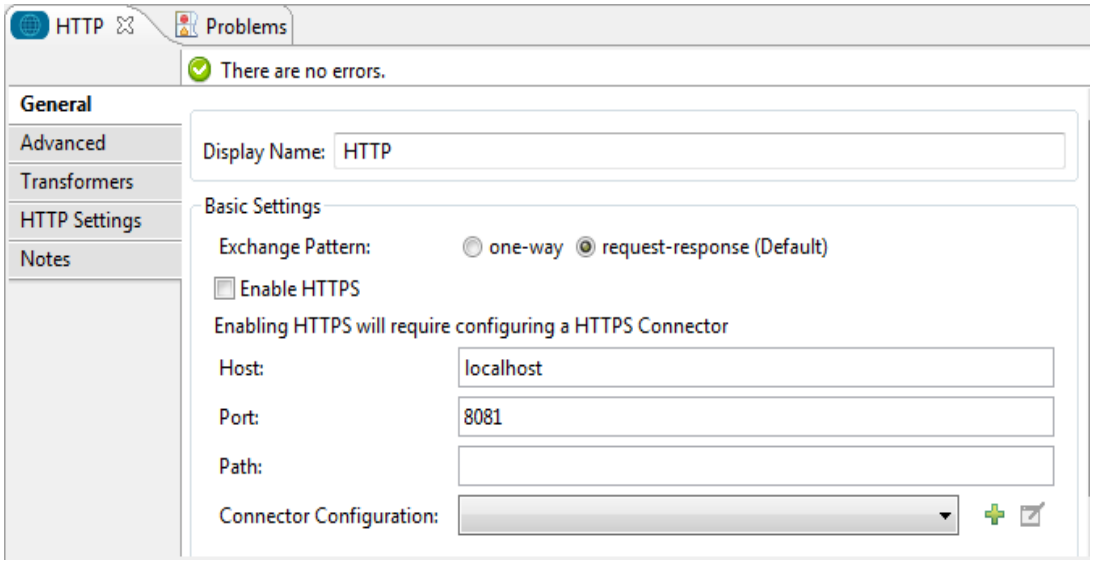


FIGURA 3.33: Propiedades HTTP.

Fuente: Propia

3. Hacer clic en el **componente Hecho** en el lienzo para abrir su **editor de propiedades** en la consola.

4. El componente de eco simplemente devuelve la carga del mensaje en respuesta a la solicitud HTTP. Para este proyecto se deja los valores predeterminados para todos los campos.

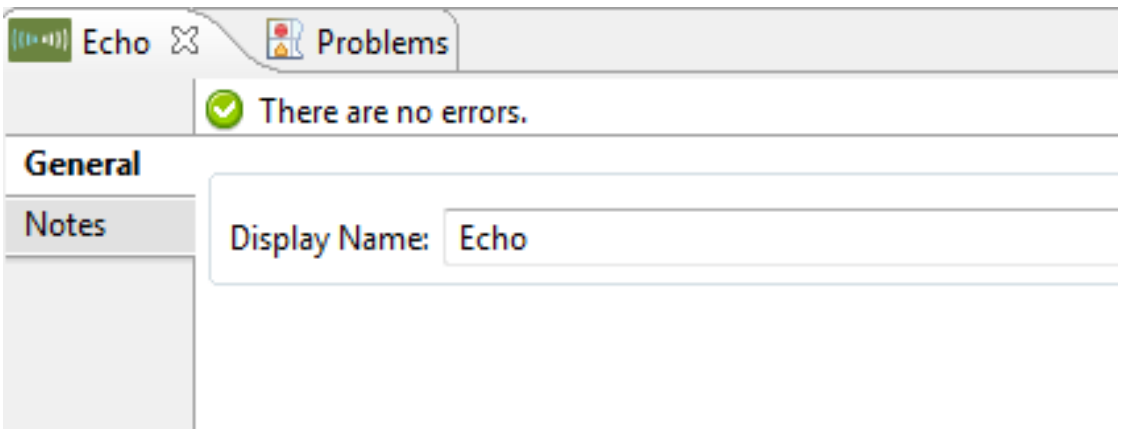


FIGURA 3.34: Propiedades Hecho.

Fuente: Propia

5. Hacer clic en el icono **save** para guardar los cambios.

Implementación del Proyecto.

Anypoint Studio viene con su propio servidor integrado, ideal para implementaciones de prueba. Se puede implementar la aplicación en el servidor incorporado para ver su funcionamiento y realizar cualquier actividad de depuración antes de implementar en un entorno de producción, como por ejemplo un servidor en el mismo recinto o CloudHub.

1. En el **Package Explorer**, hacer clic derecho en el nombre del proyecto, a continuación, seleccionar **Run As** > **Mule Application.**

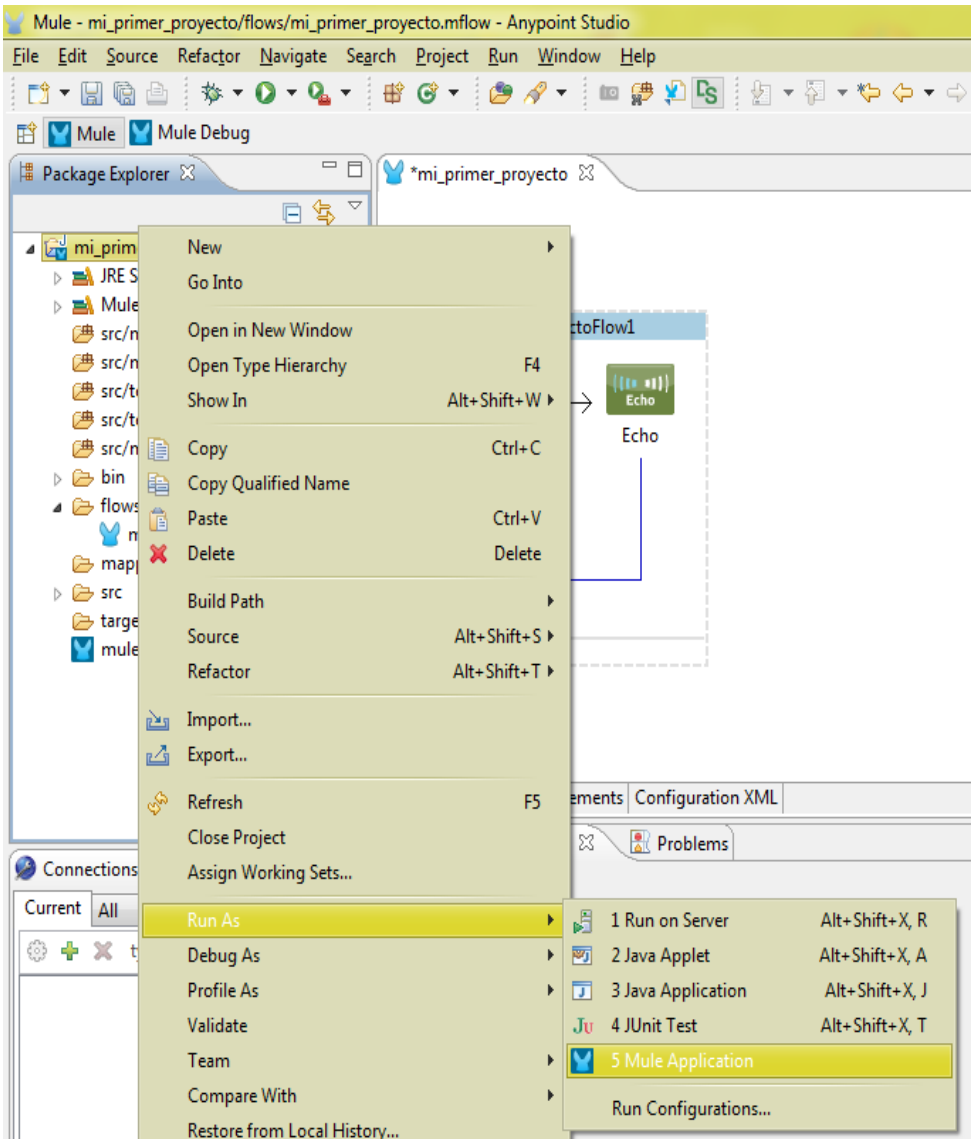


FIGURA 3.35: Ejecutar la aplicación.

Fuente: Propia

2. Studio se inicia la aplicación en el servidor incorporado, que muestra sus acciones en la consola.

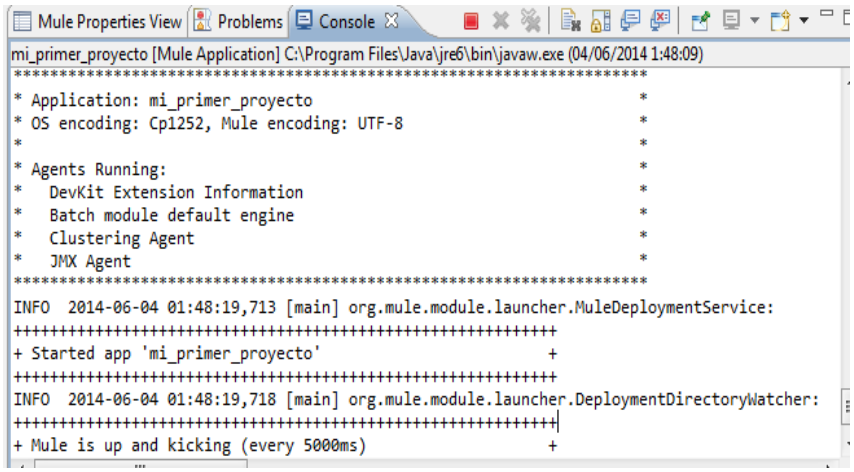


FIGURA 3.36: Consola de ejecución.

Fuente: Propia

Interactuando con el proyecto.

1. Abra un navegador Web.

2. Ir a la dirección URL de la aplicación, añadiendo una cadena que se va a convierte en la carga del mensaje. Por ejemplo:

http://localhost:8081/Hola_Mundo.

3. La aplicación acepta la solicitud a través de su extremo HTTP, entonces envía la respuesta al navegador.

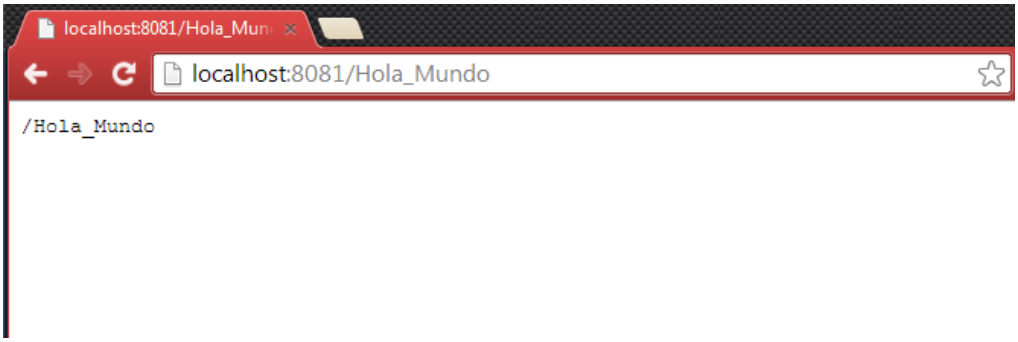


FIGURA 3.37: Mensaje mostrado en el navegador.

Fuente: Propia

3.6 BENEFICIOS MULE ESB.

En la página web oficial (MuleSoft, 2013) se presentan los siguientes beneficios de Mule ESB.

Mientras que un ESB puede parecer una opción obvia para la integración de sistemas, no todos los ESB son creados iguales. Mule ESB es un marco estable, basada en estándares que hace que la gran arquitectura de integración sea simple. A pesar de que hace todo lo que se espera de cualquier ESB en el nivel más básico como:

- La mediación.
- La orquestación.
- El enrutamiento.
- La mensajería.
- Gestión.
- Procesamiento.
- Alta disponibilidad.
- Seguridad de la empresa.

Mule ESB no es sólo un ESB, es la plataforma de integración que permite a los integradores de sistemas crear rápidamente elegantes arquitecturas orientadas a servicios, integración ligera adaptadas a las necesidades específicas de una empresa. Esta arquitectura proporciona una variedad de beneficios diferentes:

- **Small Footprint:** De tamaño de la descarga, de la memoria y uso de CPU, mule es ligero y humilde. Se desempeña bien en los servidores de las materias primas, máquinas virtuales, e incluso ordenadores portátiles para desarrolladores.

- **Fácil de aprender:** mule se integra con las herramientas de desarrollo que ya utiliza, como Eclipse, así que no hay necesidad de aprender otras tecnologías.

- **Open Source:** Construido alrededor de un núcleo y estándares de código abierto, mule ayuda a resolver necesidades inmediatas sin componentes externos que obligan a utilizar los proveedores de tecnología.

- **Nube de Integración y Conectores:** Out-of-the-box conectores en la nube proporcionan la manera más rápida y sencilla para la integración con servicios en la nube y aplicaciones. El apalancamiento mule tiene más de 100 medios de transporte y múltiples módulos para integrar diversas aplicaciones, protocolos SOAP y servicios web RESTful. Esto permite que Mule ESB se vincule con aplicaciones en la nube sin necesidad de crear código personalizado.

- **Escalable:** Mule se utiliza en aplicaciones críticas en infinidad de grandes empresas y ha escalado hasta 13k de servidores en un gran entorno distribuido.

- **Fácil de manejar:** repositorio de aplicaciones integradas de Mule ESB, el diagnóstico de rendimiento y ajuste, las alertas de SLA, y la integración con los marcos de monitoreo hacen que la gestión sea sencilla en Mule ESB.

- **Visibilidad profunda:** Desde el seguimiento de eventos de negocios para el análisis de causa raíz y el cumplimiento normativo, a la investigación de los flujos de mensajes para la depuración técnica, para obtener medidas del sistema, Mule proporciona una visibilidad profunda de las necesidades operativas de todo su negocio.

CAPITULO IV

4 INTRODUCCIÓN OPENESB.

En la página oficial de (OpenESB Community, 2013) nos dice que:

OpenESB es una herramienta de bus de servicios empresariales, que trabaja con eficiencia en la creación de aplicaciones de integración y hace que SOA sea más fácil. Inicialmente fue diseñado y desarrollado por Sun Microsystems y SeeBeyond, OpenESB se ha mejorado y mantenido por una gran comunidad que colaboran con el proyecto.

OpenESB ofrece un conjunto completo de herramientas que ayudan a diseñar, desarrollar, probar y desplegar aplicaciones de integración y aplicaciones orientadas a servicios. Basándose en JBI (Java Business Integration).

OpenESB propone un proceso de desarrollo único que promueve la migración a un servicio real de desarrollo orientado. Hoy en día, muchas empresas (Telco, Finanzas, Logística, Banco) están utilizando OpenESB para proyectos departamentales y de integración táctica.

Ellos encuentran en este bus de servicios empresariales el mejor equilibrio entre el desarrollo ergonómico, fiabilidad, potencia, escalabilidad y de código abierto. Asimismo, se encuentran en la comunidad, una pila completa de servicios de apoyo para los proyectos.

FIGURA 4.1: Logo OpenESB.

Fuente: http://www.grayareasf.com/identity.html

4.1 DEFINICIONES.

En la página web (José, Leszek, & Joaquim, 2012) define a OpenESB de la siguiente manera.

OpenESB: Permite integrar rápidamente aplicaciones empresariales y web services como las aplicaciones compuestas y débilmente acopladas. Esto permite componer y recomponer de una manera fluida y rápida las aplicaciones compuestas, con las ventajas de una Arquitectura Orientada a Servicios.

Su ejecución es el servidor de aplicaciones Glassfish Server e incluye una gran variedad de los componentes JBI, como son los de SOAP-over-HTTP-binding, y el motor de servicio WS-BPEL 2.0. Además contiene su propio motor BPEL (BPEL SE).

Aunque está basado en el SDK JBI, Open ESB amplía las capacidades de la implementación de JBI con services engines, binding components, herramientas y servicios de administración y monitorización adicionales. La integración que tiene con el entorno de desarrollo NetBeans ayuda al despliegue de aplicaciones de una manera eficiente y rápida, con una serie de facilidades como la ayuda en el desarrollo, en el control de errores y pruebas.

OpenESB está basado en estándares java. El núcleo de OpenESB se basa en las especificaciones Java JBI (Java Business Integration), donde describe cómo los componentes se van conectando al ESB. Además, OpenESB también soporta JCA (Java Connector Architecture), SOAP, WS-*, XML. (Borja, Florez, & Torres Leon, 2011)

4.2 ARQUITECTURA.

OpenESB consta de 5 partes: el framework, components, Integrated Development Environment, container y los development plugins.

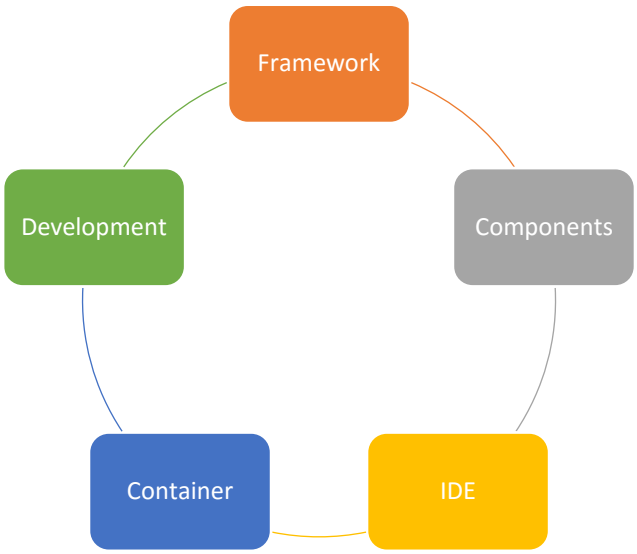


FIGURA 4.2. Principales partes de OpenESB

Fuente: Propia

4.2.1 FRAMEWORK.

El Framework consiste en una aplicación ligera JBI en Java. Esta implementación es container-agnóstica y puede funcionar en cualquier plataforma y cualquier recipiente. Aunque el desarrollo y el apoyo se centran principalmente en Glassfish V2 y V3, proyectos beta en JBoss y autónomo [máquina virtual Java | JVM] funcionan bien y están en curso (2012 Q2). Además del marco OpenESB es ligero, también es fiable y altamente escalable.

Está incrustado en una máquina virtual de Java y se comunica con otros a través de los componentes del marco de unión. Esta arquitectura combina perfectamente con las nuevas arquitecturas de la nube y permite una fácil implementación y gestión de infraestructuras muy complejas.

El Framework es completamente manejable con cualquier herramienta basada en JMX tal como JConsole o herramientas más sofisticadas, como Opsview o Nagios. El Framework implementa un bus virtual conocida como NormSalised Message Router (NMR). Este es un poderoso canal de comunicación inteligente asincrónica entre los componentes.

4.2.2 COMPONENTES.

La especificación JBI define 2 tipos de componentes: el motor de servicios (SE) y el componente de unión (BC). La SE y AC implementan la misma interfaz de contrato, sin embargo, se comportan de manera diferente:

- Componentes de unión actúan como la interfaz entre el mundo exterior y el bus, y es capaz de generar mensajes del bus tras la recepción de estímulos a partir de una fuente externa, o generar una acción / interacción externa en respuesta a un mensaje recibido desde el bus.

- Motores de servicio reciben mensajes del bus y envían mensajes al bus. SE al no tener ningún contacto directo con el mundo exterior. Necesita del bus para poder interactuar con otros componentes, si los componentes u otros motores de servicio son obligatorio.

4.2.3 ENTORNO DE DESARROLLO INTEGRADO Y PLUGINS.

OpenESB ofrece una variedad de herramientas gráficas que facilita la evolución de SOA y de integración complejas. Se proponen XLM Potente, XML Schema, WSDL, BPEL editor, mapeo de datos y redacción.

Aplicaciones editores gráficos con OpenESB. Del mismo modo, crear, implementar, ejecutar, probar y depurar las tareas son gestionadas por las herramientas gráficas. OpenESB ofrece la mejor ergonomía para los desarrollos de ESB y SOA.

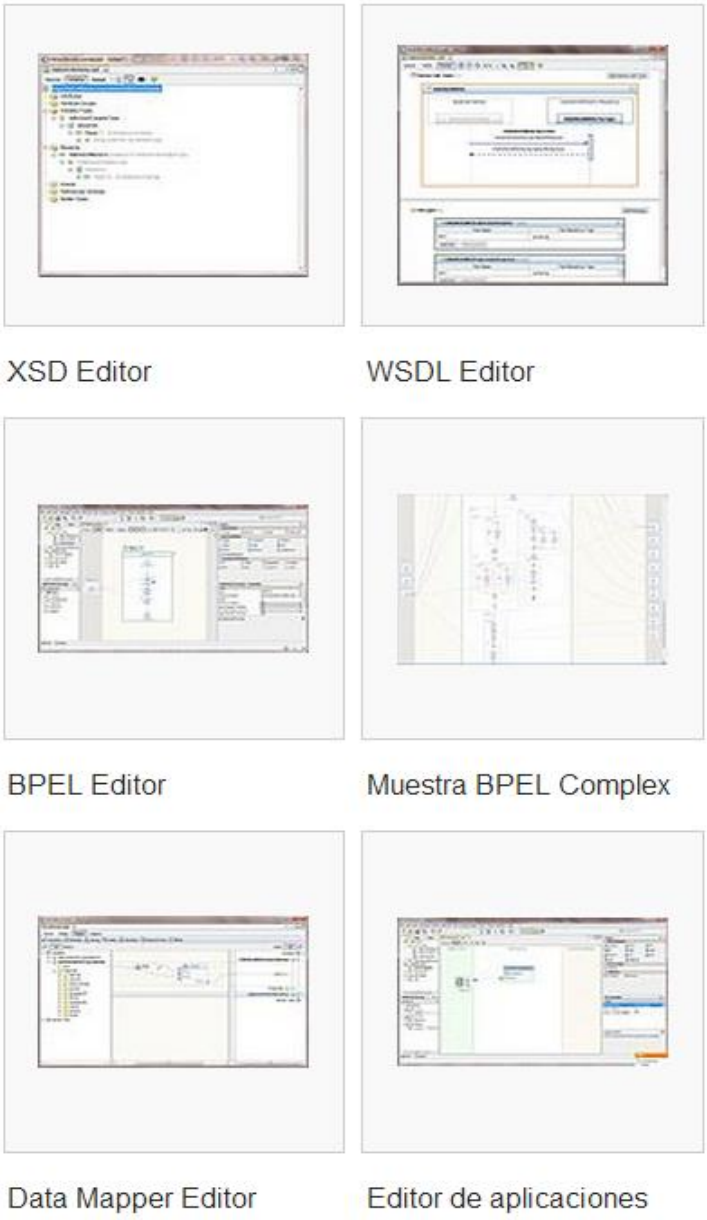


FIGURA 4.3: Herramientas gráficas de OpenESB.

Fuente: Propia

4.2.4 CONTENEDOR.

En la actualidad, OpenESB viene con Glassfish V2 como un contenedor. OpenESB en Glassfish es fiable, de fácil manejo y muy escalable con o sin Clusters. Un punto de referencia es que en la nube OpenESB mostro que procesa más de 10 millones de mensajes complejos en un elaborado proceso multicanal complejo de empresa a empresa (Archivo, FTP, SOAP, JMS).

Al mismo tiempo, genera más de 12 millones de mensajes complejos para socios externos. Informe de referencia. Con un esfuerzo extra de configuración, OpenESB ejecuta en JBoss V4 y V5 y configurado para usos productivos. Hoy en día, el desarrollo OpenESB se centra en nuevos contenedores Glassfish V3-V4, JBoss V7, OSGI.

FIGURA 4.4: Contenedores disponibles para OpenESB.

Fuente: Propia

4.3 CARACTERÍSTICAS.

En la página web Oficial (OpenESB Community, 2013) de OpenESB describe las siguientes características:

- **OpenESB es simple:** OpenESB es el ESB más fácil de instalar. 6 clics en una completa interfaz gráfica son todo lo que necesita para empezar. En menos de 5 minutos, el instalador OpenESB crea un entorno completo de desarrollo (Netbeans), un contenedor JBI (servidor de aplicaciones GlassFish) y +20 conectores y motores de servicio inmediatamente disponibles.

- **OpenESB es fácil de usar:** OpenESB ofrece un conjunto de herramientas gráficas para facilitar el desarrollo de su integración de aplicaciones. Arquitectos, desarrolladores pueden diseñar esquemas XML, WSDL, BPEL y

composición de aplicaciones con OpenESB graphical tools. Del mismo modo, los desarrolladores utilizan herramientas gráficas para crear, implementar anular la implementación, ejecutar, probar y depurar aplicaciones. OpenESB ofrece la mejor ergonomía para ESB y desarrollo SOA.

- **OpenESB es escalable y potente:** Tiene una arquitectura central, esta basa en un potente y optimizado bus. El bus está diseñado para apoyar a decenas de millones de mensajes por día en un ambiente seguro y protegido con una infraestructura de luz y un bajo presupuesto. Componentes OpenESB están optimizados para alta disponibilidad en una implementación en clúster. Por ejemplo, decenas de casos pueden ejecutar en el mismo grupo.

- **Normas y estándares:** OpenESB se basa en un estándar universal, tales como XML, XSD, WSDL y BPEL. Proporciona extenso cumplimiento de "WS-I" y la plataforma de Microsoft a través de "WSIT" (http://wsit.java.net/) el marco de servicios web desarrollado conjuntamente por Sun y Microsoft. Aproveche sus habilidades y conocimientos sobre SOA, el desarrollo y el despliegue OpenESB es compatible con los estándares y compatibles con SOA y otro proyecto de Integration. OpenESB es 100% Java. Por ejemplo, puede crear aplicaciones en múltiples plataformas de desarrollo (Windows, Mac y Linux) y se ejecutan en la plataforma de la oficina (Windows, Linux, Mac, Solaris y AIX).

- **Confiabilidad para su sistema de integración:** OpenESB propone soporte nativo para la garantía de ejecución y los procesos de consistencia (Compensación, Redelivery, redirección, proceso Atómica, la persistencia de procesos...). Como un beneficio, puede incluir la consistencia del proceso como parte de su especificación de proceso para satisfacer sus necesidades de negocio. Estas características nativas no requieren recursos o desarrollo adicionales.

- **¿Quién utiliza OpenESB?:** Cientos de proyectos se han desarrollado en muchos sectores withOpenES. Es utilizado en finanzas, banca, logística, empresas de atención médica, los ejércitos, las universidades, las

organizaciones gubernamentales y no gubernamentales eligieron OpenESB por su sistema de integración departamental o empresarial. Ellos aprecian su simplicidad, escalabilidad y potencia y TCO baratas para el desarrollo de prototipos, proyectos departamentales y estratégicos.

FIGURA 4.5: Características OpenESB.

Fuente: http://www.open-esb.net/index.php?option=com_content&view=article&id=104&Itemid=490

4.4 INSTALACIÓN Y CONFIGURACIÓN OPENESB 2.3.1.

Requisitos previos.

Los requisitos previos para la instalación de Open ESB es tener instalado y configurado el JDK.

4.4.1 INSTALACIÓN.

1. Descargar OpenESB 2.3.1.

2. Ingresamos a la página de OpenESB, en la sección Download, accedemos a la opción *OpenESB binaries.*

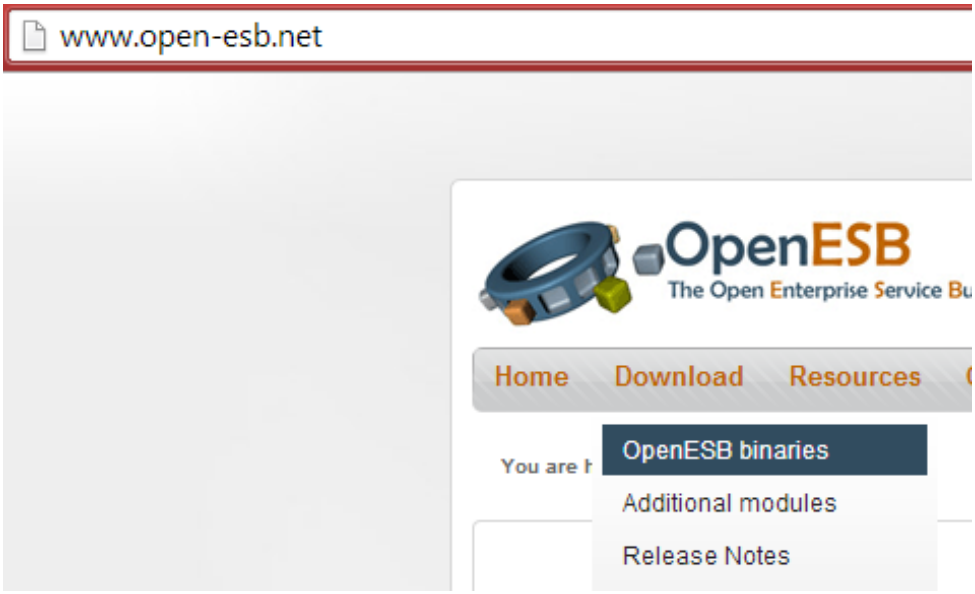


FIGURA 4.6: Página de descarga de OpenESB.

Fuente: Propia

3. Seleccionamos la versión para Windows, en la cual ya viene incluido con el bundle GlassFish y Netbeans 7.3.1, ya que los componentes ya están configurados e instalados tanto en el servidor como en el IDE para poder utilizar la metodología SOA. También ya se encuentran integrados correctamente GlassFish y NetBeans.

OpenESB version 2.3.1

OpenESB version 2.3.1 is the current version developed and packaged by the community . it is based on G

OpenESB Version 2.3.1			
Window	OpenESB V2.3.1 for Windows	MD5	OpenESB Version 2.3.1
Linux	OpenESB V2.3.1 for Linux	MD5	OpenESB Version 2.3.1
Mac OS X86	OpenESB V2.3.1 for Mac	MD5	OpenESB Version 2.3.1

FIGURA 4.7: Selección de la versión de OpenESB.

Fuente: Propia

4. Una vez que ya está descargado *openesb-v231-installer-windows.exe*, ejecutar el instalador y proceder a seguir los pasos del asistente:

FIGURA 4.8: Instalador de OpenESB.

Fuente: Propia

5. Si se hace clic en el botón Customize se observa más claramente los componentes que se van a instalar: el OpenESB Design-Time, que corresponde a NetBeans 7.3.1 con plugins SOA; y OpenESB Run-time, que es el servidor GlassFish con JBI integrado y configurado.

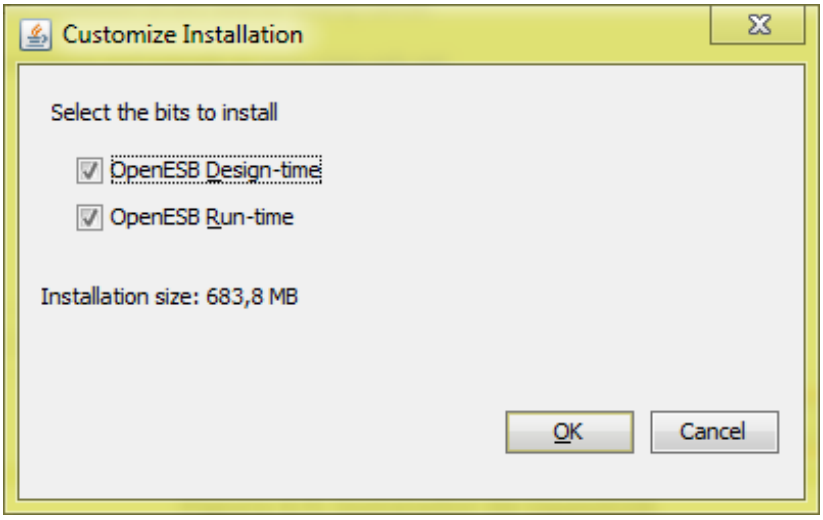


FIGURA 4.9: Componentes de instalación: IDE y plataforma de ejecución ESB.

Fuente: Propia

6. En el siguiente paso se procede aceptar el acuerdo de la licencia:

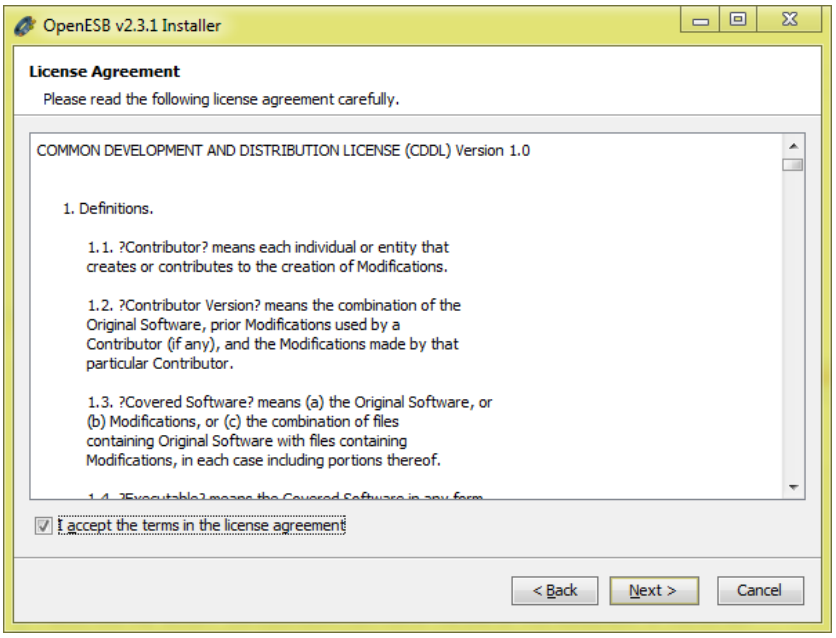


FIGURA 4.10: Acuerdo de Licencia de OpenESB v2.3.1.

Fuente: Propia

7. A continuación configuramos el entorno del IDE Netbeans para lo cual queda por defecto las opciones que presenta:

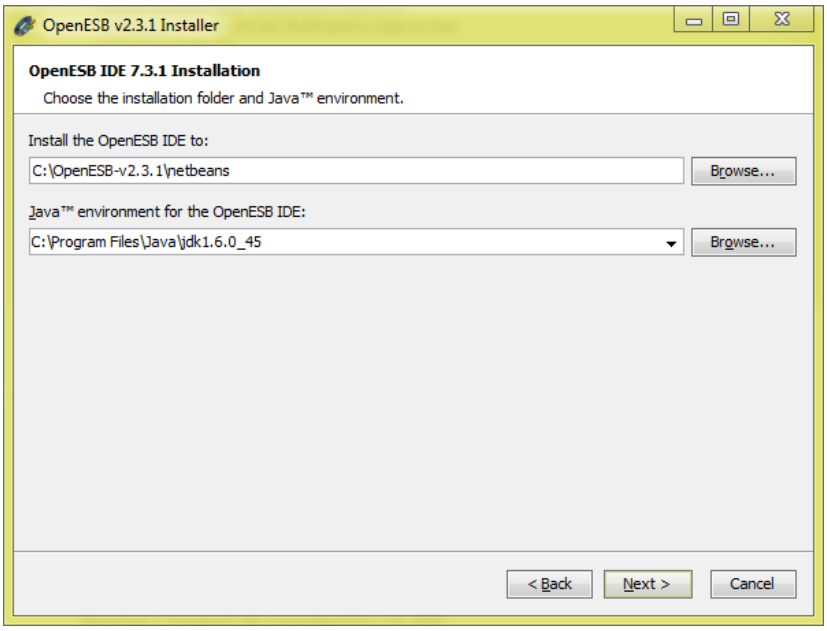


FIGURA 4.11: Carpeta de instalación para NetBeans y localización de Java.

Fuente: Propia

8. A continuación se configura OpenESB 2.3.1. Veremos detalladamente el significado de cada propiedad de la pantalla:

- El primer campo es el que permite introducir la ruta de instalación del servidor.
- En el segundo campo requiere la ruta donde está la instalación del JDK.
- Admin Username: usuario para la consola de administración de OpenESB. Dejar el valor por defecto: admin.
- Admin Password/Retype Password: contraseña para el usuario admin. Dejar el valor por defecto: adminadmin.
- A continuación se configura los puertos para las conexiones con el servidor:

- HTTP Port: 8080. Conexiones HTTP.
- HTTPS Port: 8181. Conexiones HTTP seguras.
- Admin Port: 4848. Puerto de la consola de administración de OpenESB.
- JMS Port: 7676. Puerto de implementación del API del Servicio de Mensajería.
- JMX Admin Port: 8686. Puerto para las Java Management Extensions,
- IIOP Port: 3100. Puerto para el protocolo IIOP (Internet Inter-ORB Protocol).
- IIOP SSL Port: 3820. Comunicación segura para el protocolo IIOP.
- IIOP Mutual Auth Port: 3920. Puerto para la autenticación.

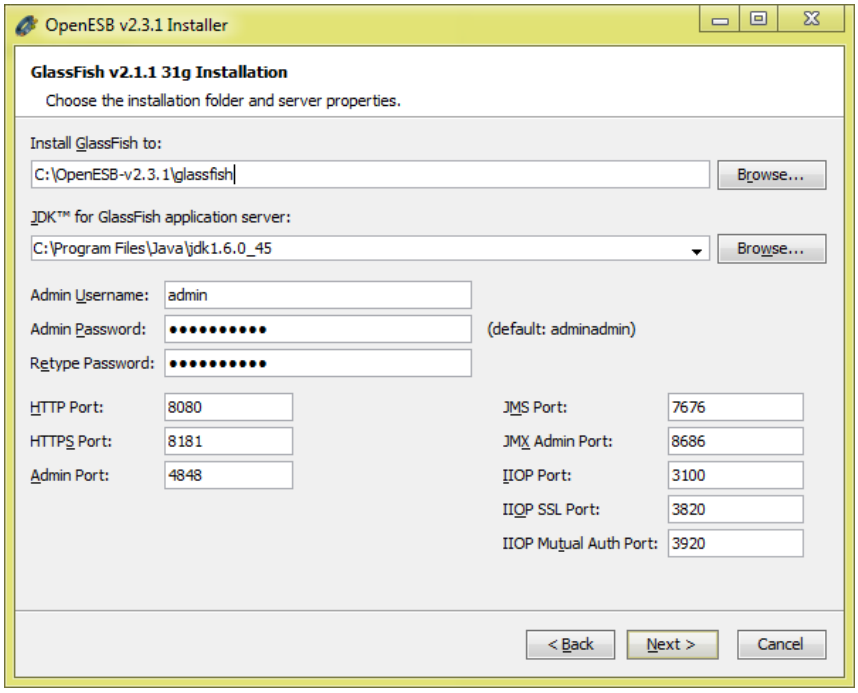


FIGURA 4.12: Rutas y puertos de la instalación de OpenESB 2.3.1.

Fuente: Propia

9. El siguiente paso nos confirma lo que se va a instalar:

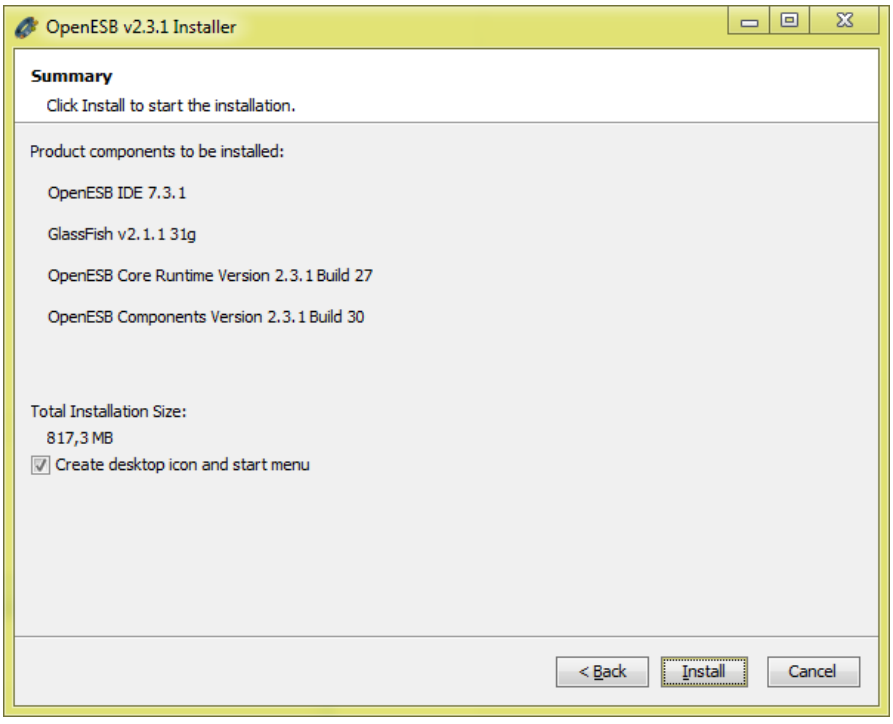


FIGURA 4.13: Resumen de la instalación del OpenESB 2.1.

Fuente: Propia

10. Iniciamos la instalación.

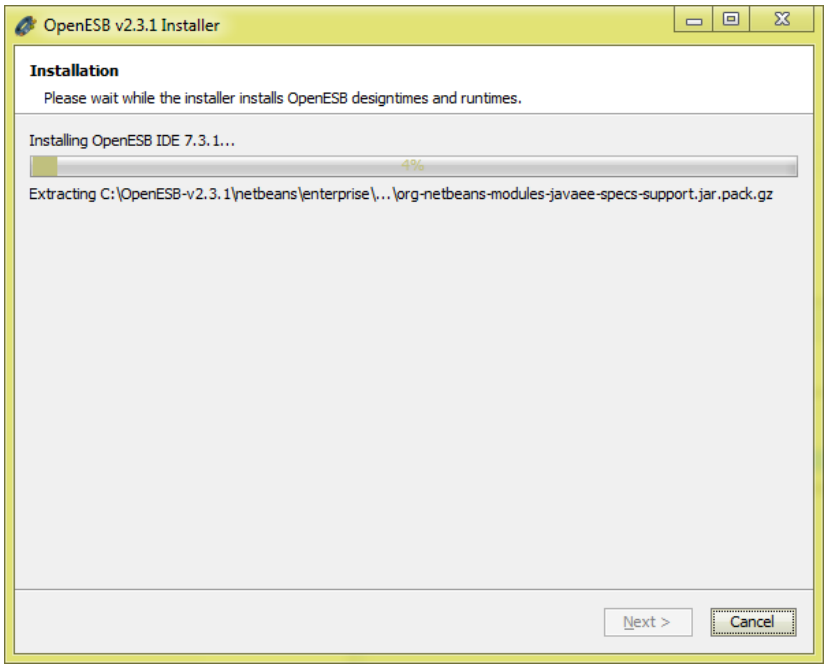


FIGURA 4.14: Instalación de OpenESB.

Fuente: Propia

11. Finalización del proceso de instalación.

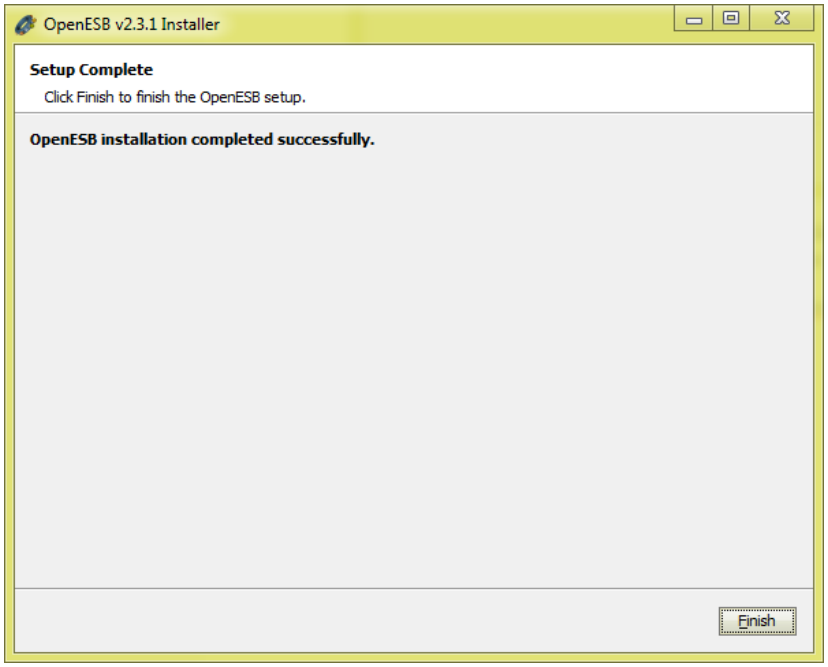


FIGURA 4.15: Finalización de la instalación de OpenESB.

Fuente: Propia

4.4.2 EJEMPLO EN OPENESB.

Introducción.

Este ejemplo muestra cómo hacer una aplicación de Hola mundo muy simple con OpenESB donde se va a ver cómo funciona el OpenESB.

4.4.2.1 CREE UN PROYECTO.

1. Una vez ya instalado procedemos a realizar un ejemplo del uso de OpenESB con Netbeans, abrimos el programa *OpenESB v2.3.*

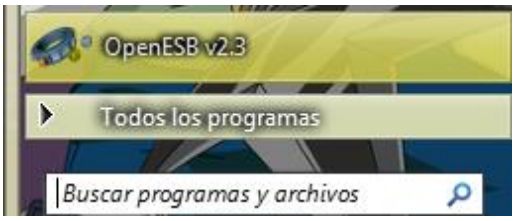


FIGURA 4.16: Menú de la opción de OpenESB.

Fuente: Propia

2. Al crear un nuevo proyecto SOA se observan que tiene las siguientes opciones:

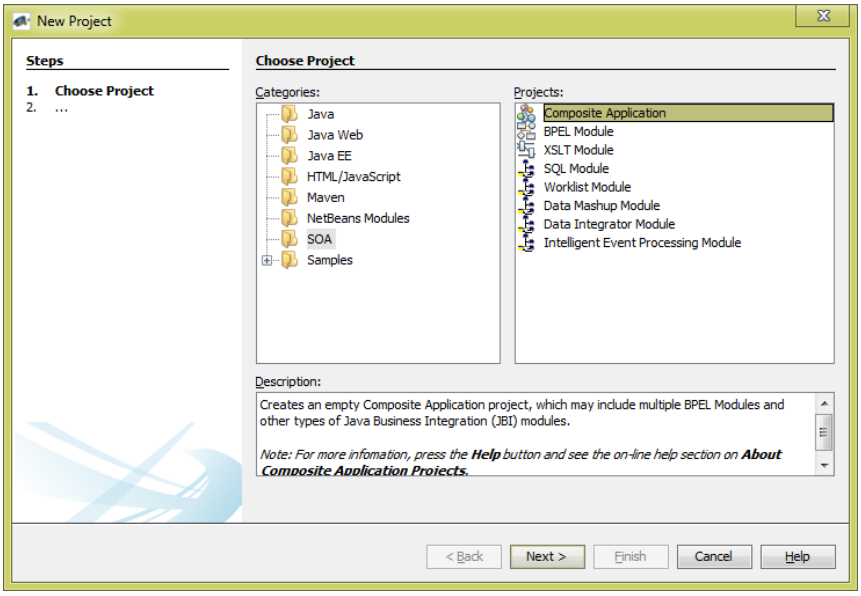


FIGURA 4.17: Opciones para crear proyectos SOA.

Fuente: Propia

3. Seleccionar en la categoría de SOA, proyecto BPEL Module.

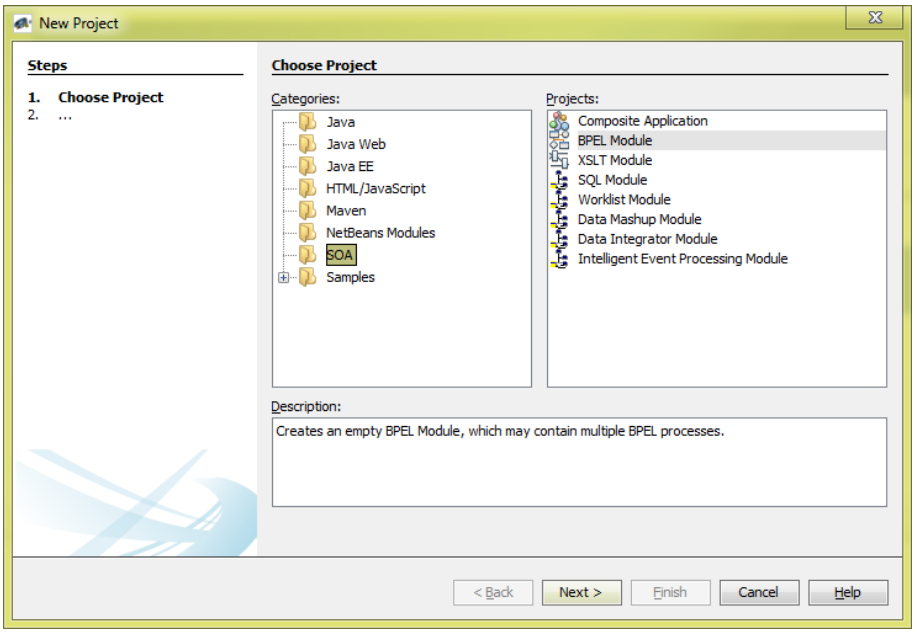


FIGURA 4.18: Crear proyecto BPEL Module SOA.

Fuente: Propia

4. Establecer el nombre de proyecto como HelloWorld.

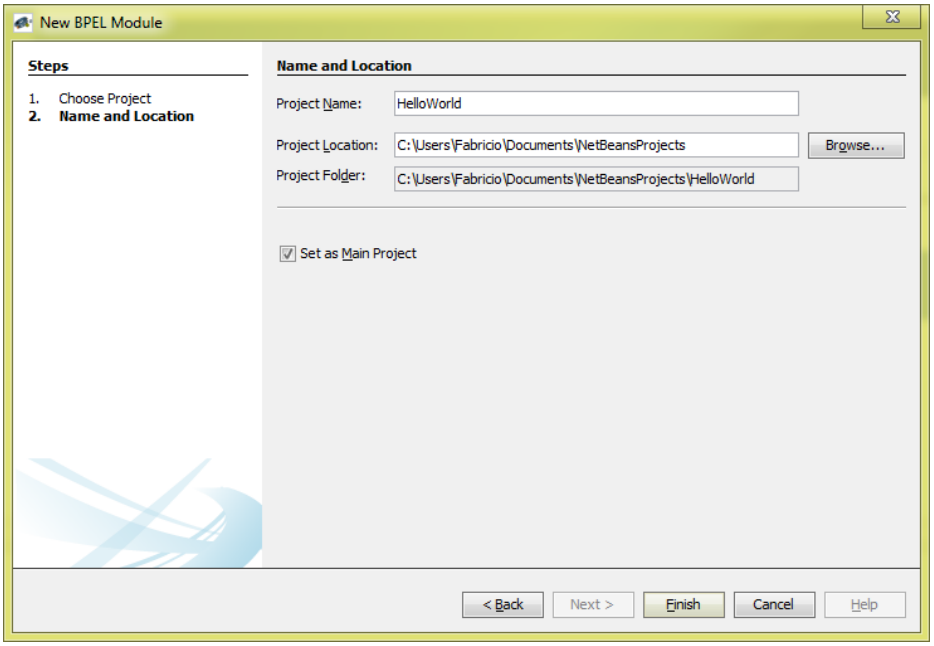


FIGURA 4.19: Poner nombre al BPEL Module SOA.

Fuente: Propia

5. Escoger un lugar conveniente para la creación del proyecto y dar clic sobre Finish al terminar se abren una ventana en Netbeans.

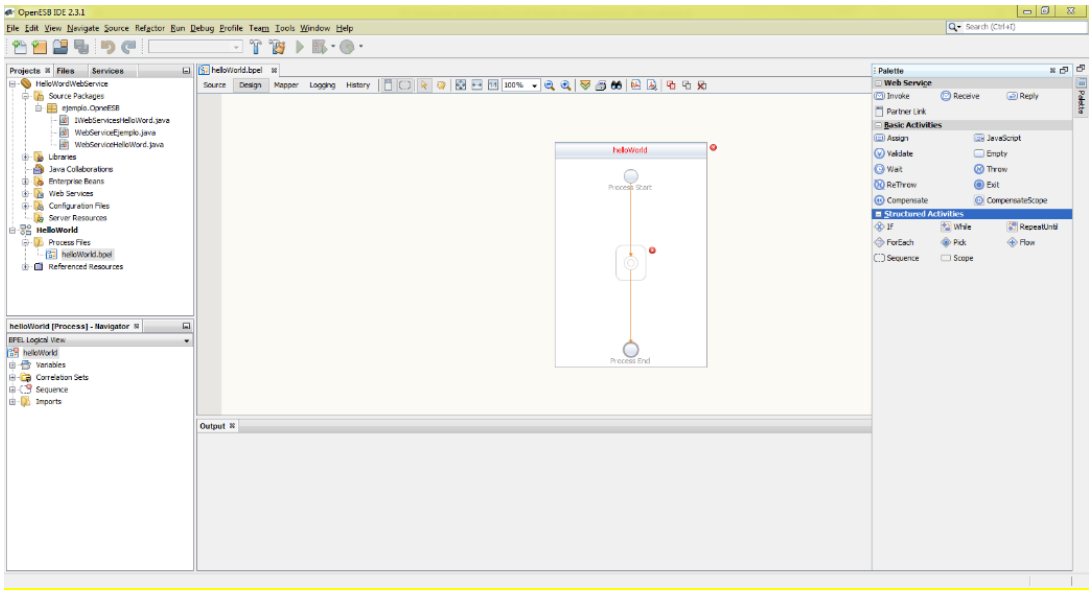


FIGURA 4.20: Proyecto de BPEL.

Fuente: Propia

4.4.2.2 CREE A UN WSDL.

1. El primer paso para desarrollar la aplicación es crear un WSDL. En la etiqueta de proyecto, seleccionar el nodo “Process Files” y clic en “WSDL Document”.

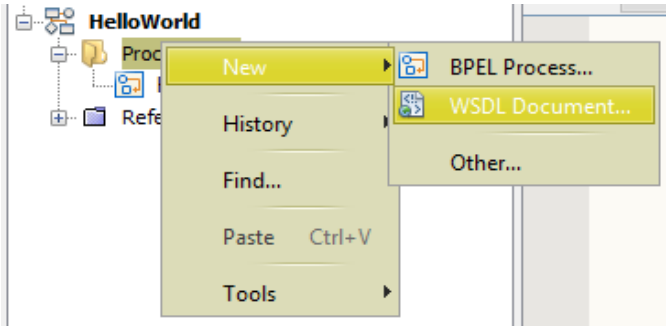


FIGURA 4.21: Creación WSDL Document.

Fuente: Propia

2. Poner de nombre “HelloWordWSDL”.

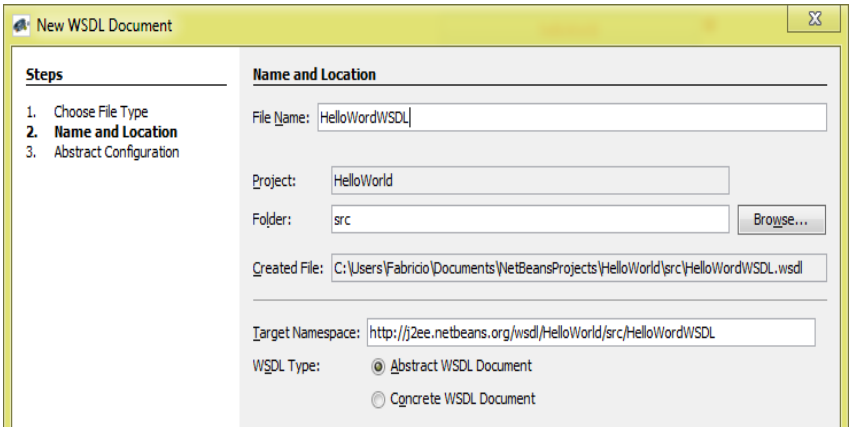


FIGURA 4.22: Poner nombre “HelloWordWSDL”.

Fuente: Propia

3. Establecer el Nombre del Archivo y mantener el WSDL Type, Conservar todos los valores por defecto entonces dar clic sobre Finish.

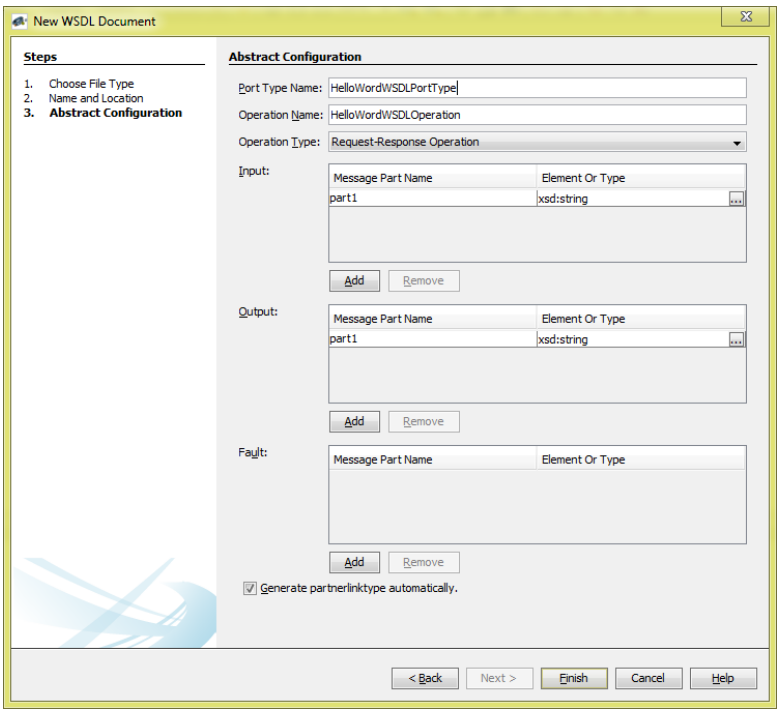


FIGURA 4.23: Opciones de la creación del WSDL.

Fuente: Propia

4.4.2.3 DISEÑAR EL BPEL.

1. Hacer doble clic en el helloWorld.bpel del nodo, observamos el editor de BPEL.

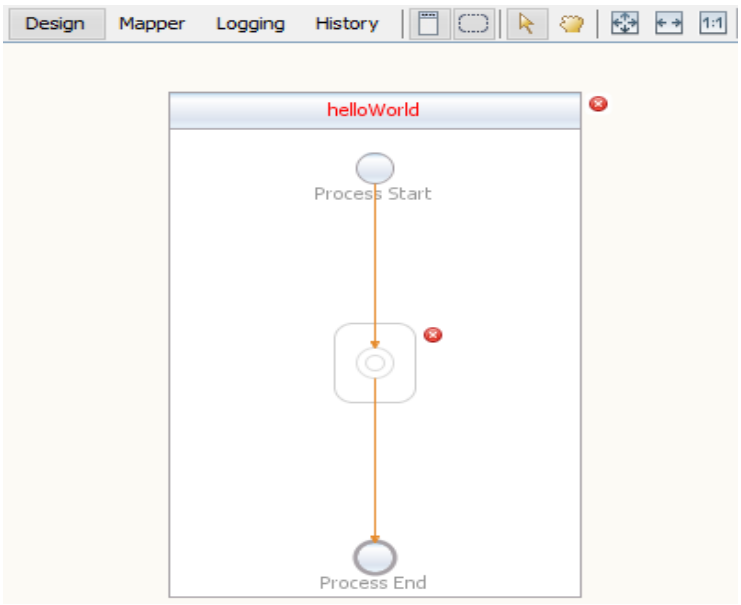


FIGURA 4.24: Editor BPEL.

Fuente: Propia

2. En el proyecto, seleccionar el HelloWorldWSDL.wsdl y arrastrarlo al lado izquierdo BPEL Editor, cuando el ratón entra en el área BPEL Editor, un círculo naranjado aparece.

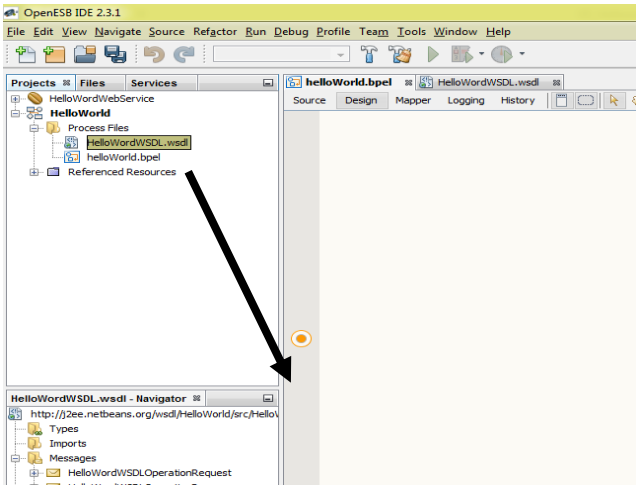


FIGURA 4.25: Poner el HelloWorldWSDL en el Editor BPEL.

Fuente: Propia

3. Dejar caer al ratón en el círculo anaranjado y el nuevo icono aparece en el Editor.

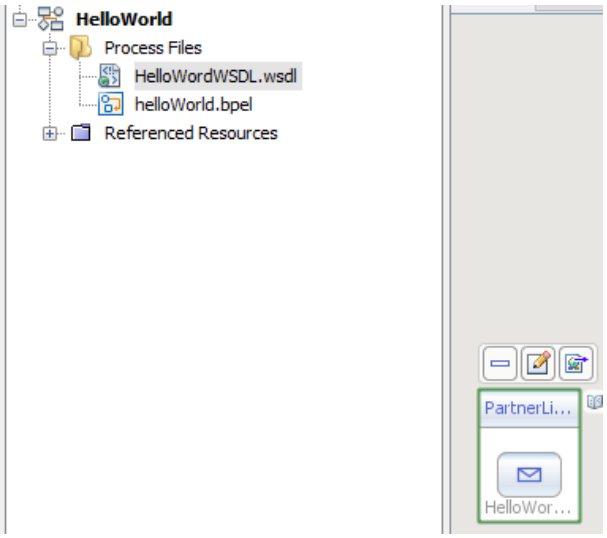


FIGURA 4.26: Icono del HelloWorldWSDL.

Fuente: Propia

4. PartnerLink será el punto de ingreso del proceso. En la ventana de la "palette" en el lado derecho de la pantalla, seleccionamos el Icono "Receive ". Arrastrarlo y ponerlo en la mitad del editor BPEL donde el círculo naranja está puesto.

FIGURA 4.27: Poner "Receive" en el Editor.

Fuente: Propia

5. Un nuevo “Receive” es exhibido en el editor BPEL. Hacer lo mismo para Assign e Reply. Al final el BPEL se parece a esto:

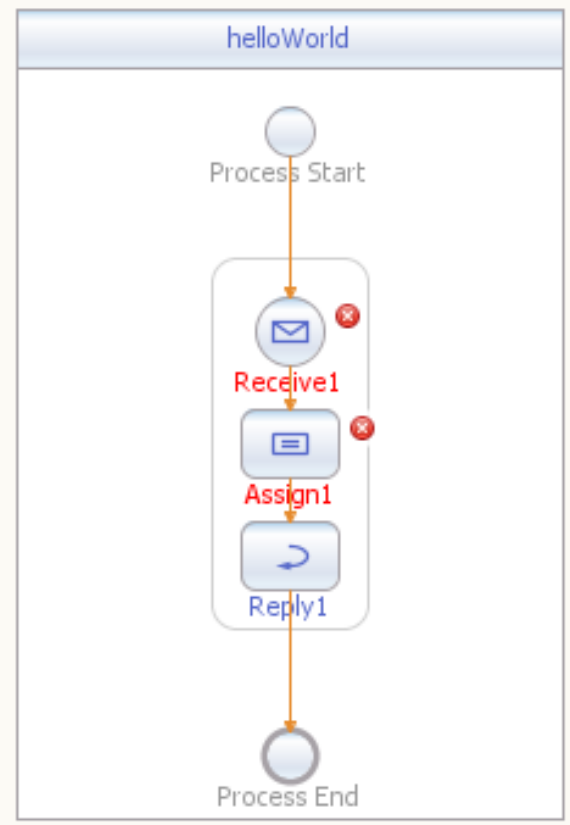


FIGURA 4.28: BPEL con Receive, Assign y Reply.

Fuente: Propia

6. Hacer clic doble en Receive1 en el editor BPEL.

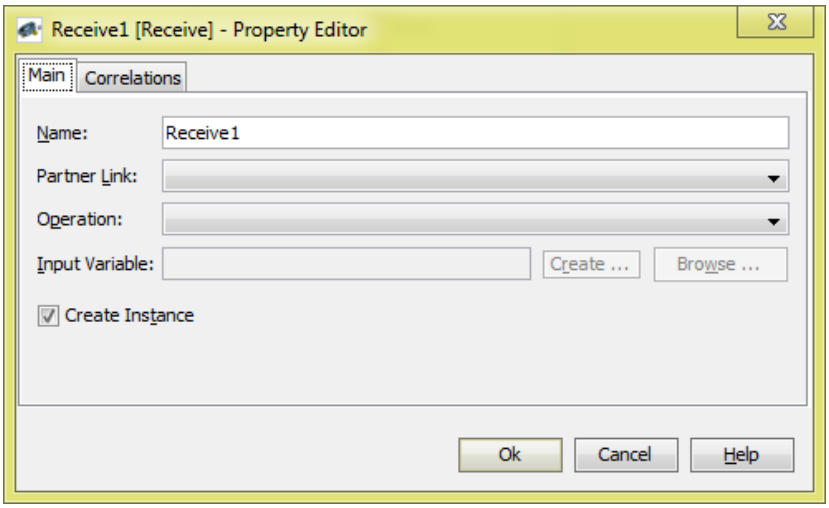


FIGURA 4.29 Propiedades de Receive1.

Fuente: Propia

7. El Editor de Receive-Property se abre. Seleccione los valores por defecto para “Partner Link” y “Operation”.

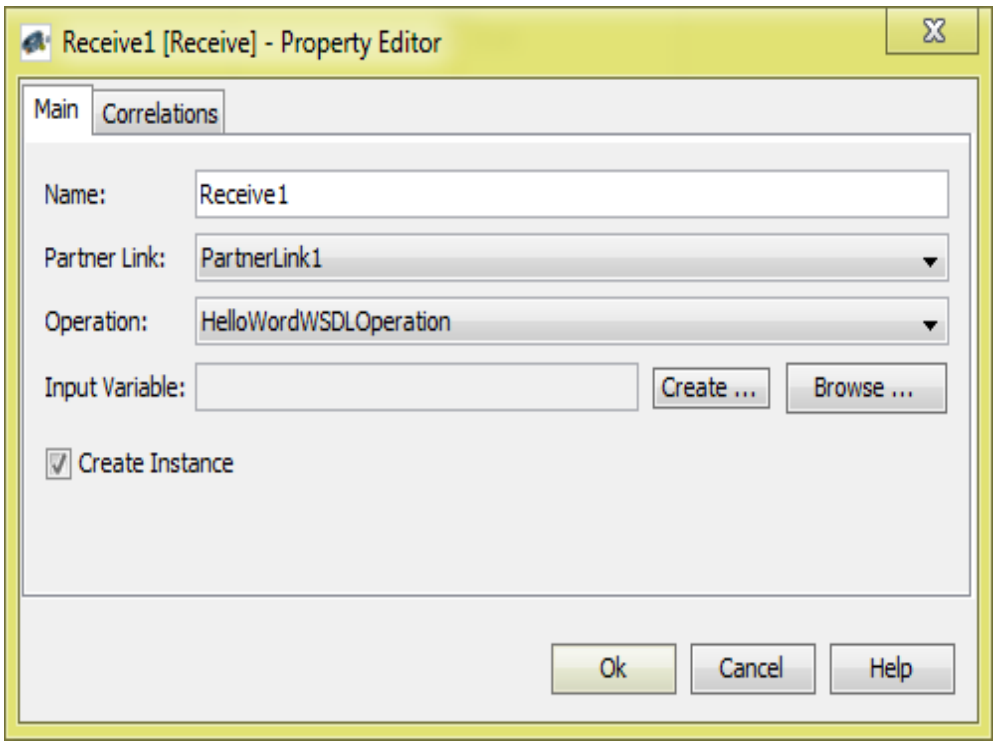


FIGURA 4.30: Seleccionar propiedades para Receive1.

Fuente: Propia

8. Hacer clic en Create, mantener los valores por defecto y hacer clic en Aceptar.

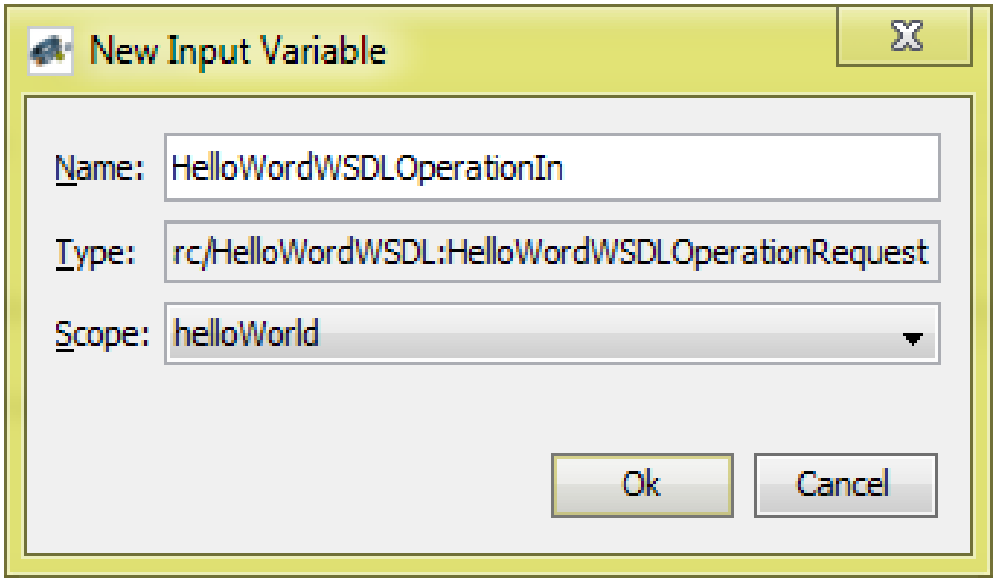


FIGURA 4.31: Creación de una nueva “Input Variable”.

Fuente: Propia

9. Se chequea si se crea la Instancia en “Recieve” y si la actividad es establecida como muestra la figura.

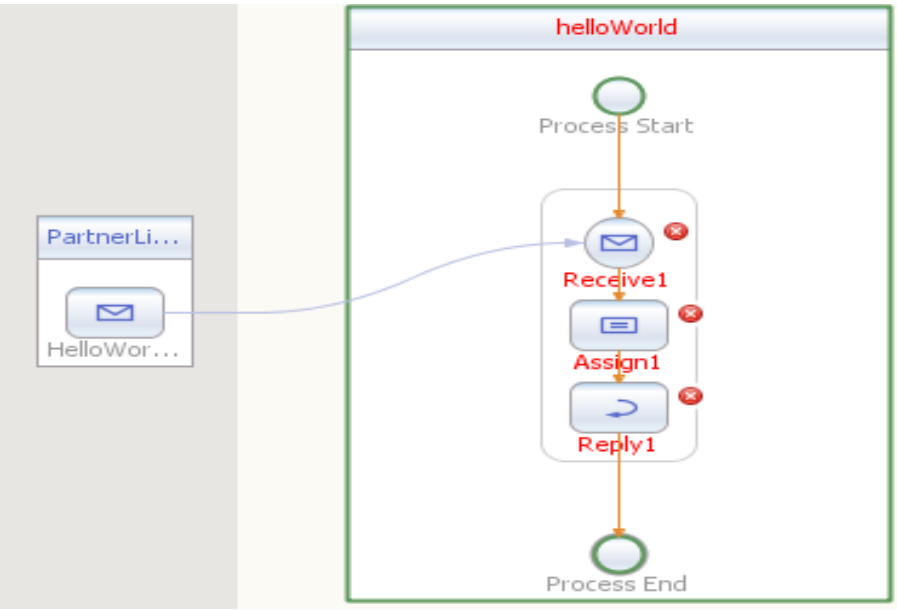


FIGURA 4.32: Instancia creada en “Receive”.

Fuente: Propia

10. Hacer doble clic en Reply1.

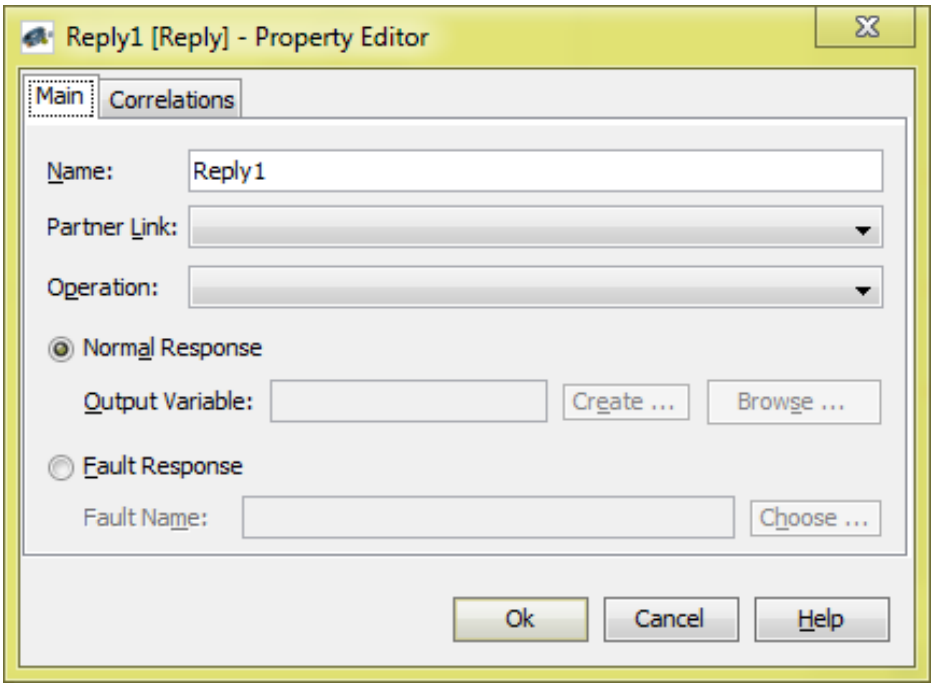


FIGURA 4.33: Propiedades de “Reply”.

Fuente: Propia

11. El Editor de Reply-Property se abre. Seleccione los valores por defecto para “Partner Link” y “Operation”.

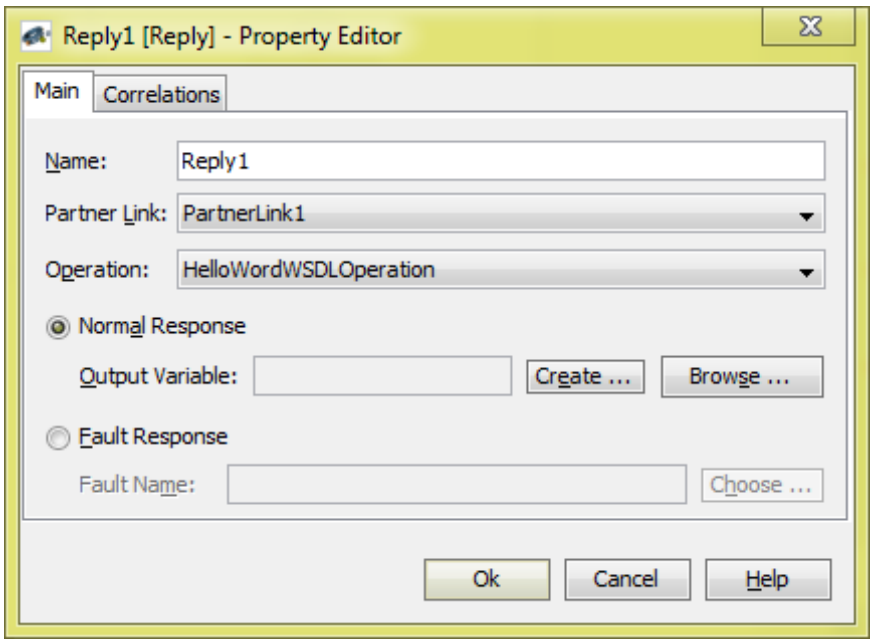


FIGURA 4.34: Seleccionar propiedades para Reply1.

Fuente: Propia

12. Hacer clic en Create, mantener los valores por defecto y hacer clic en Aceptar.

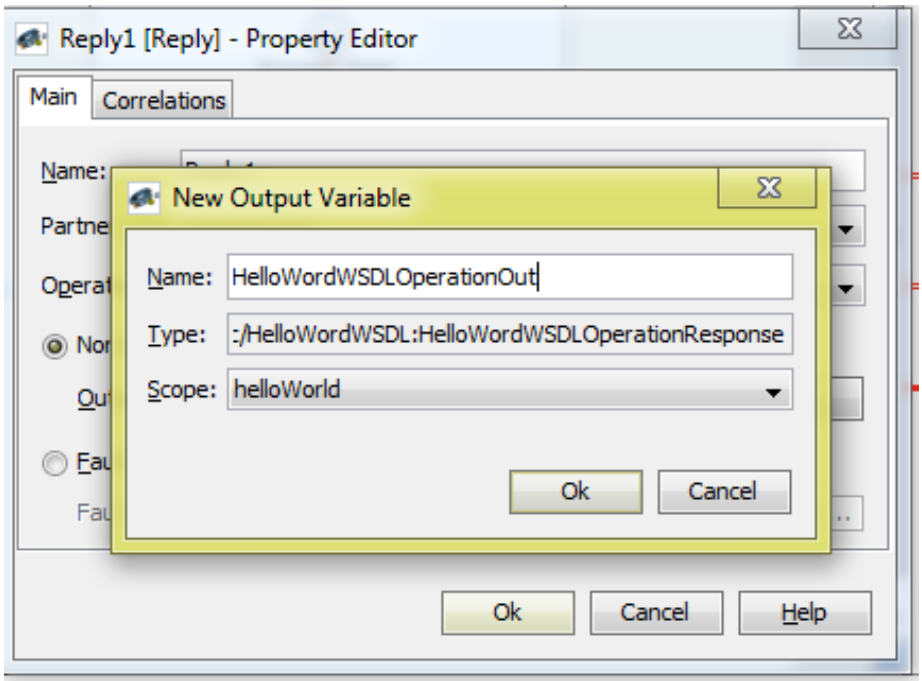


FIGURA 4.35: Creación de una nueva “Output Variable”.

Fuente: Propia

13. Se chequea si se crea la Instancia en “Reply” y si la actividad es establecida como muestra la figura.

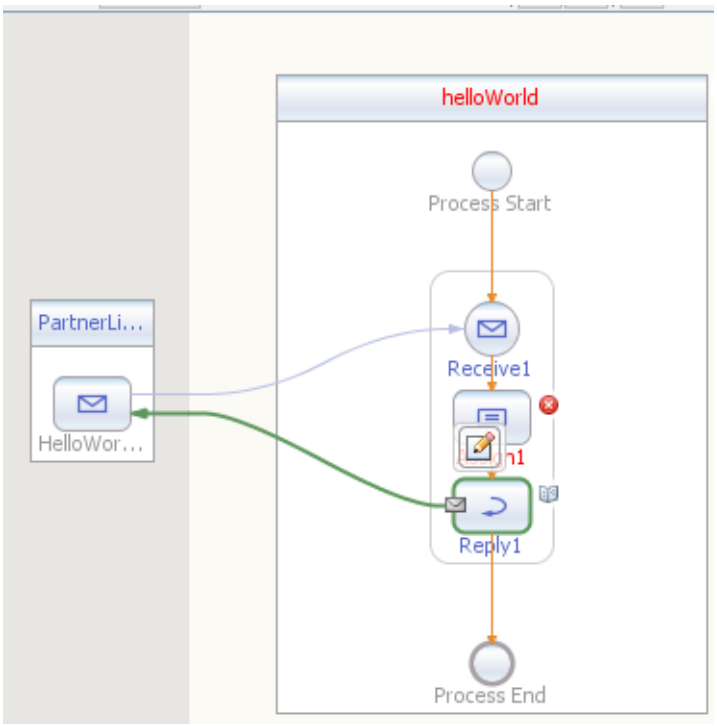


FIGURA 4.36: Instancia creada en “Receive”.

Fuente: Propia

14. Hacer clic doble en Assign1 para abrir al editor del mapper. Expandir los nodos de las variables por ambos lados.

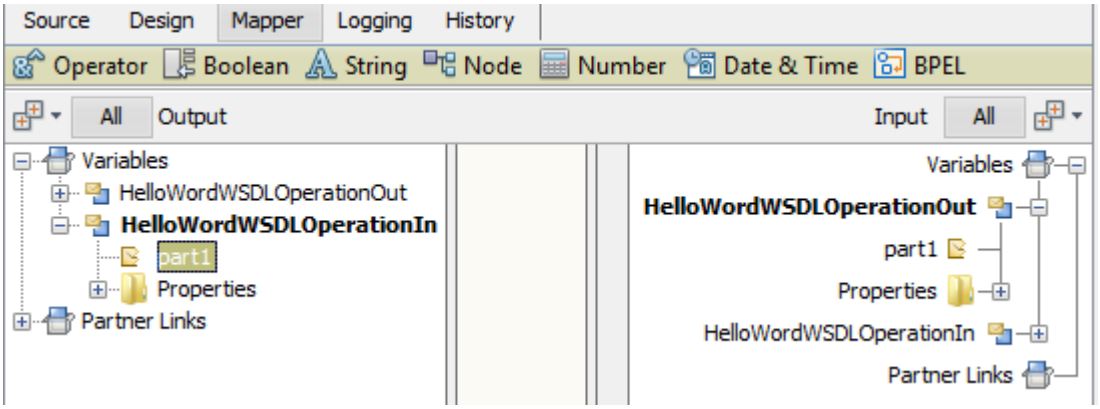


FIGURA 4.37: Editor Mapper.

Fuente: Propia

15. Hacer click en HelloWorldWSDLOperationOut Part1. Aparece una vía azul por el nivel de la part1. Selecciónele al icono “String” en el menú del mapper del editor sobresaliente y seleccione el artículo “String Literal”.

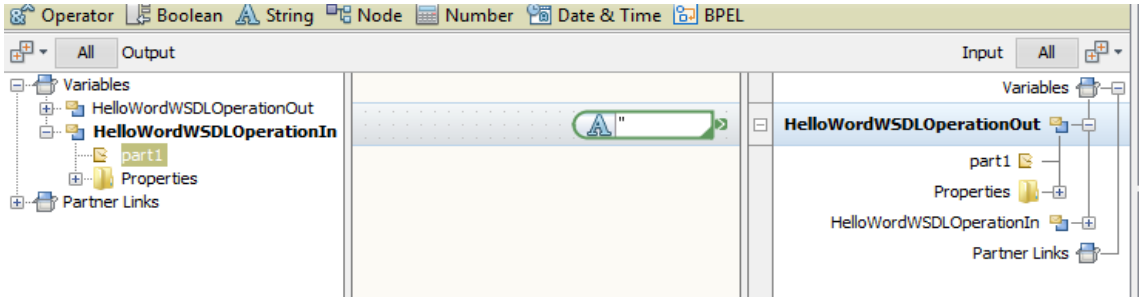


FIGURA 4.38: Selección de String Literal.

Fuente: Propia

16. Un nuevo icono aparece en la vía azul. Es el lugar para poner “hola”.

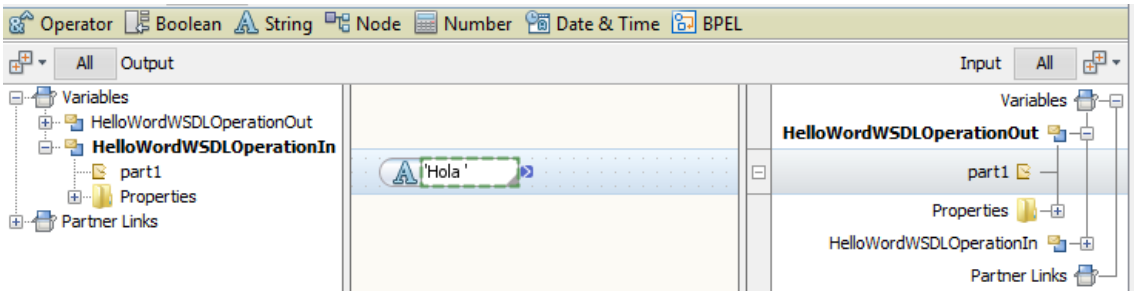


FIGURA 4.39: Escribir Hola en el String Literal.

Fuente: Propia

17. Seleccionar el icono “String” en el menú del mapper del editor y seleccione el artículo “Concat”.

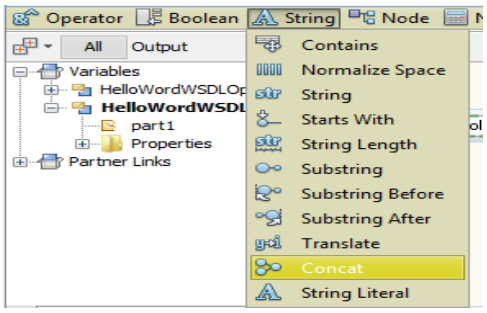


FIGURA 4.40: Insertar una variable “Concat”.

Fuente: Propia

18. Un icono Concat se agrega en la vía azul en el lado derecho de “Hola”.

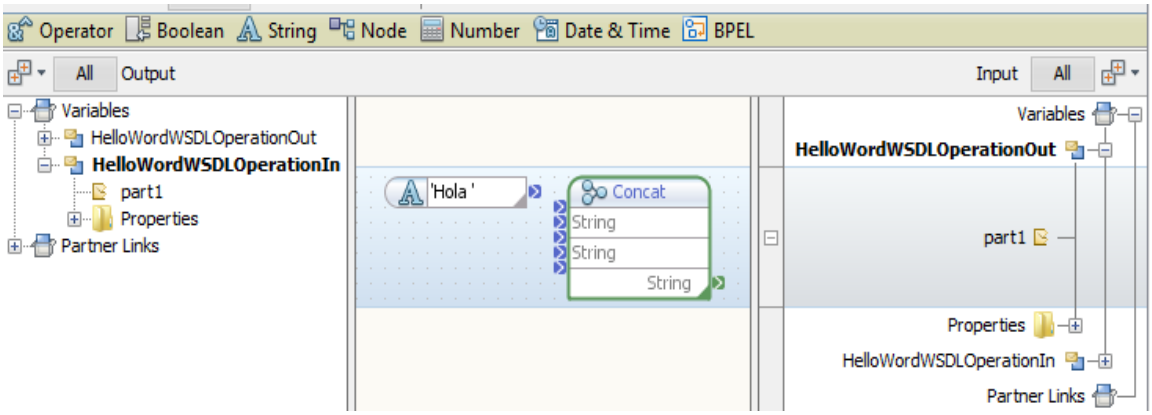


FIGURA 4.41: Concat creado.

Fuente: Propia

19. Poner los enlaces para utilizar las funcionalidades de arrastre como muestra la imagen.

- El Link 1 Hello para el String Concat.
- El Link 2 de HelloWorldWSDLOperationIn Part1 para otro Concat.
- El Concat Del Link 3 para HelloWorldWSDLOperationOut Part1.
- Concat concatena Hola + el nombre que se encuentra en el mensaje de aporte y pone el resultado en el mensaje de producción.

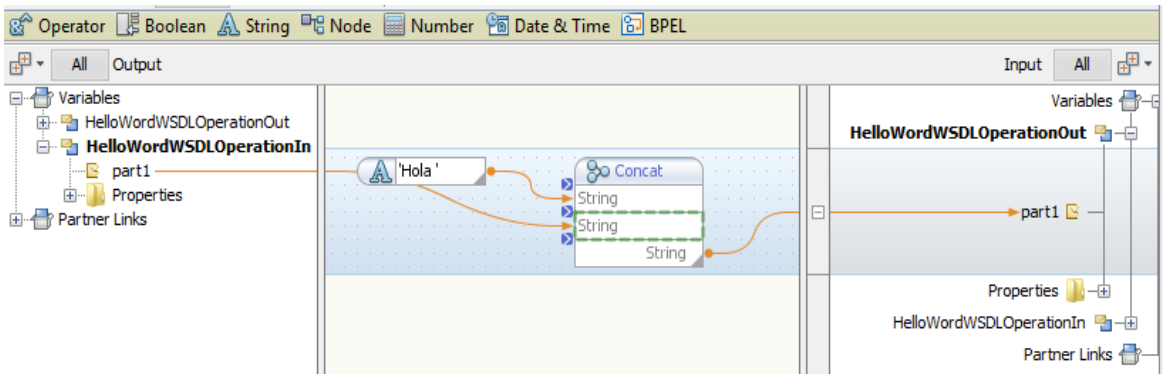


FIGURA 4.42: Concatenar las funcionalidades.

Fuente: Propia

20. Dar clic sobre “Design” para regresar al Editor BPEL, se observa que las señales rojas no están presentes, y se muestra que su BPEL es válido sin errores.

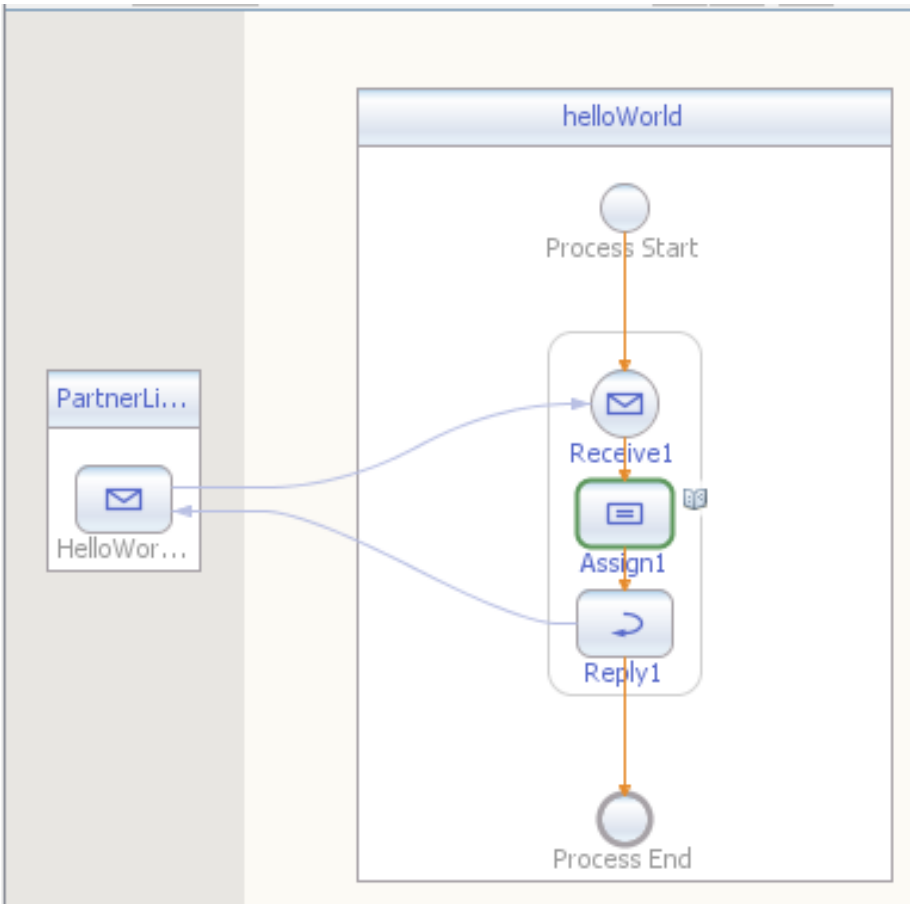


FIGURA 4.43: BPEL sin errores.

Fuente: Propia

4.4.2.4 CREE UNA COMPOSITE APPLICATION.

Para utilizar una aplicación con OpenESB usted tiene que crear una Composite Application.

1. Dar a la derecha un clic sobre la etiqueta de proyecto y hacer una selección de “New Project” seleccionar en categoría de SOA el proyecto “Composite Application”

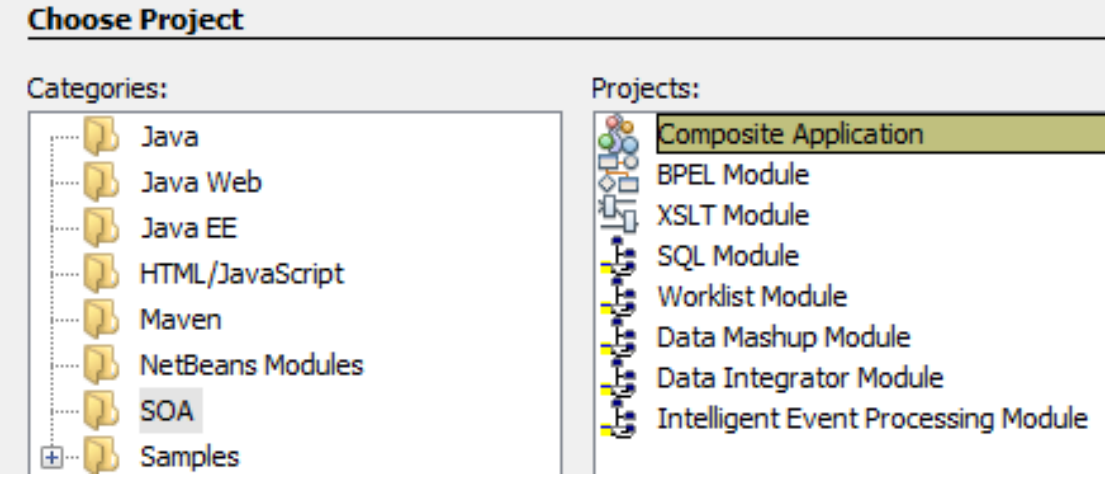


FIGURA 4.44: Creación proyecto “Composite Application”.

Fuente: Propia

2. Hacer clic en “Next” y establecer el nombre de proyecto como “HelloWorldCA”.

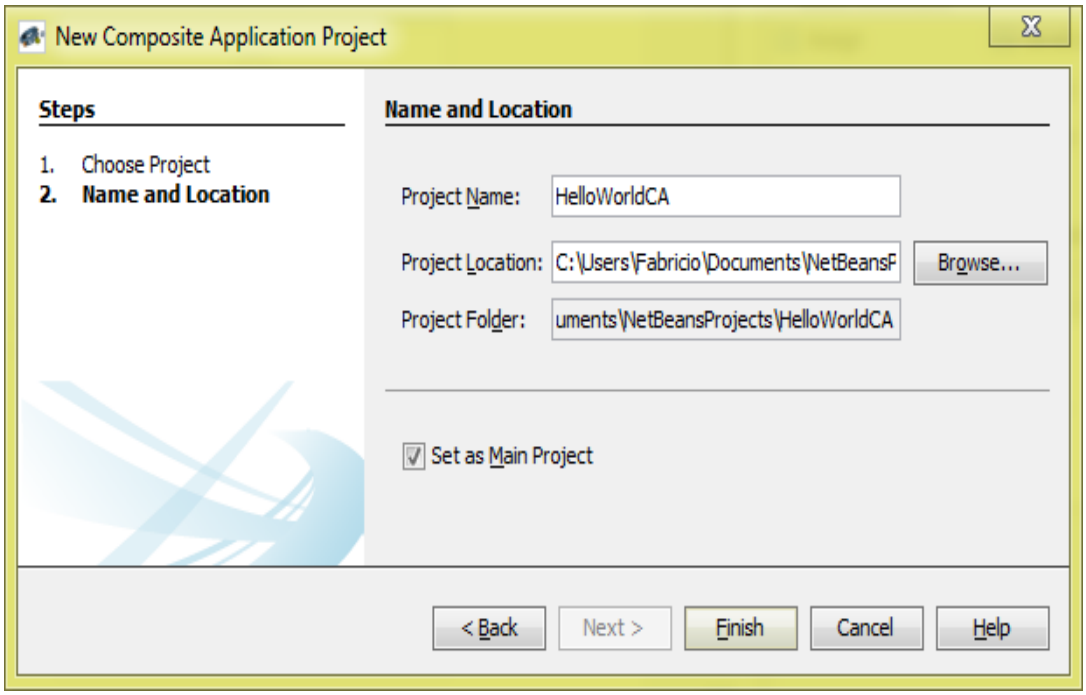


FIGURA 4.45: Poner nombre al proyecto “Composite Application”.

Fuente: Propia

3. Dar Clic Sobre “Finish” al cabo de un rato, el proyecto “HelloWorldCA” es agregado a la etiqueta de Proyecto, y la “Composite Application” abre el editor.

4.4.2.5 AGREGAR EL BPEL MODULE EN UNA COMPOSITE APPLICATION.

1. Seleccionar el proyecto “HelloWorld”, arrastrarlo y poner en el “JBI Module Lane” en el editor “Composite Application”.

FIGURA 4.46: Poner el proyecto “HelloWord” en “JBI Modules”.

Fuente: Propia

2. Dar un clic sobre el botón que es representado por un martillo  .

4.4.2.6 AGREGAR UN BINDING COMPONENT.

Nuestro proyecto está listo a ser utilizado pero sabemos precisamente cómo recibirá los mensajes el proyecto. Para enviar los mensajes al módulo BPEL, usamos protocolo SOAP. Así es que tenemos que definir un puerto SOAP donde el mensaje será enviado.

1. En la parte derecha de la pantalla, en la paleta, seleccionamos “SOAP”.

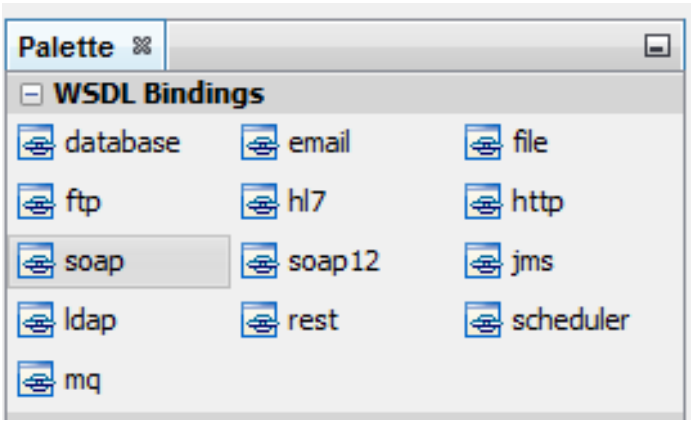


FIGURA 47: Paleta de componentes.

Fuente: Propia

2. Arrastre el icono y ponerlo en “WSDL Ports” en el editor del “Composite Application”.

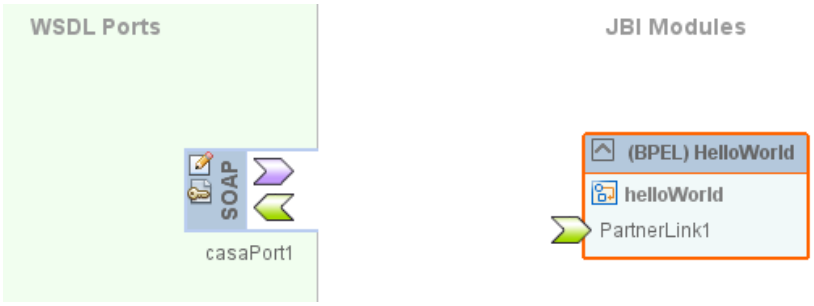


FIGURA 4.48: Agregar un SOAP.

Fuente: Propia

3. Un icono “soap” aparece en el “WSDL Ports”. Con una simple funcionalidad de arrastrar se acoplan SOAP y BPEL.

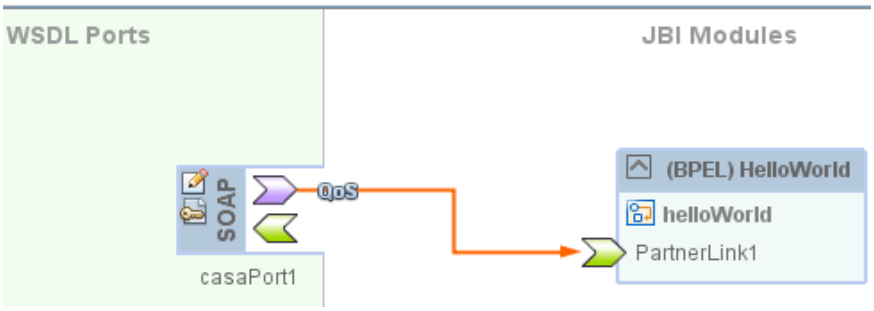


FIGURA 4.49: Enlazar el SOAP con BPEL.

Fuente: Propia

Implementación.

La aplicación está lista a ser utilizada en Glassfish.

4. Seleccionar la pestaña Services y expandir el nodo de Servers.

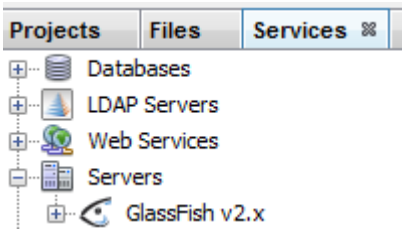


FIGURA 4.50: Propiedades Services.

Fuente: Propia

5. Haga clic en Glassfish y seleccione la opción del menú “Start”.

6. Ir de regreso a etiqueta Proyectos. Seleccionar HelloWorldCA. Dar un clic sobre “Deploy”.

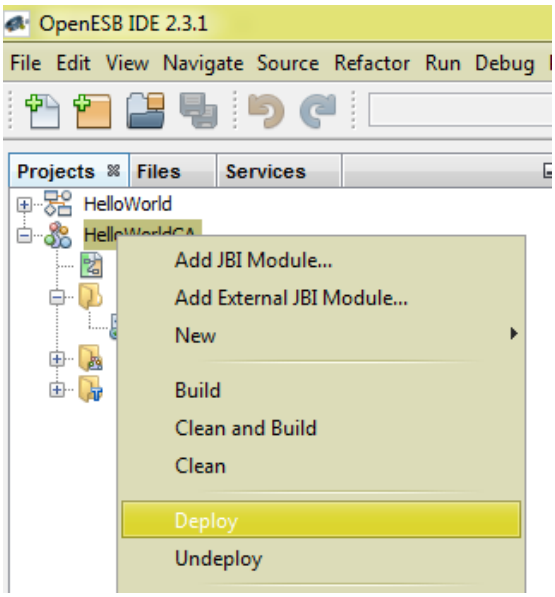


FIGURA 4.51: Menú del proyecto.

Fuente: Propia

7. Luego de pocos segundos, en el fondo de la pantalla, en el build.xml en la ventana de producción, se chequea si la ejecución es exitosa.

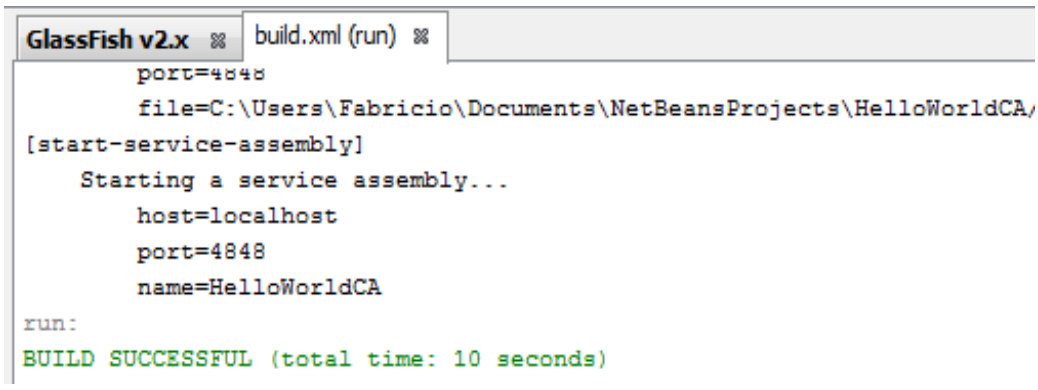


FIGURA 4.52: Ejecución correcta del proyecto

Fuente: Propia

4.4.2.7 PROBAR LA APLICACIÓN.

1. Ahora probemos la aplicación. OpenESB empotró un generador experimental simple. Le deja probar cualquier puerto SOAP en sus aplicaciones. Seleccione el nodo “Test” en el proyecto HelloWorldCA. Dé un clic derecho y seleccione “New Test Case”.

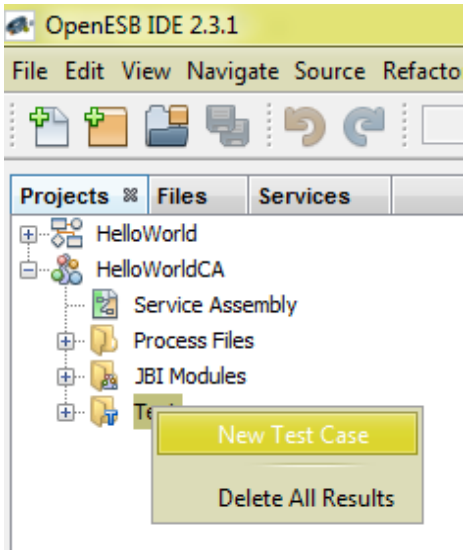


FIGURA 4.53: Creación de un Test.

Fuente: Propia

2. Poner el nombre “Prueba” y hacer clic siguiente.

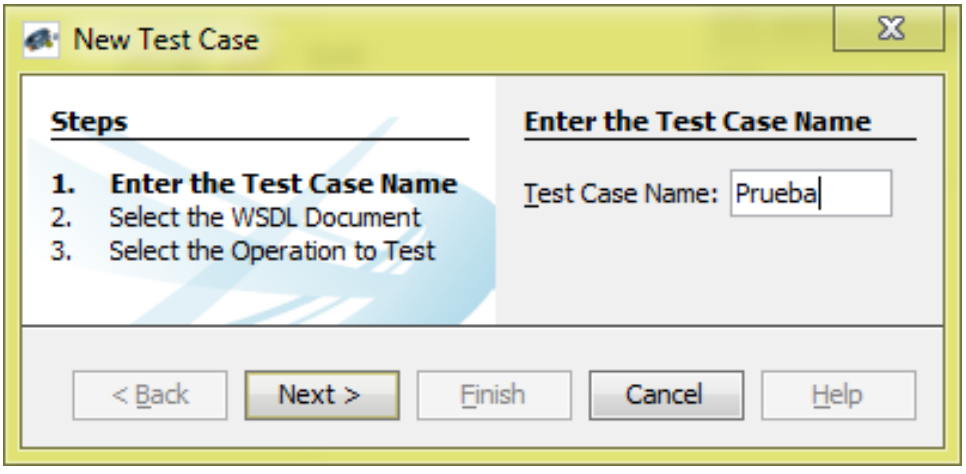


FIGURA 4.54: Poner nombre al Test.

Fuente: Propia

3. Seleccionar el HelloWorldCA y haga clic en siguiente.

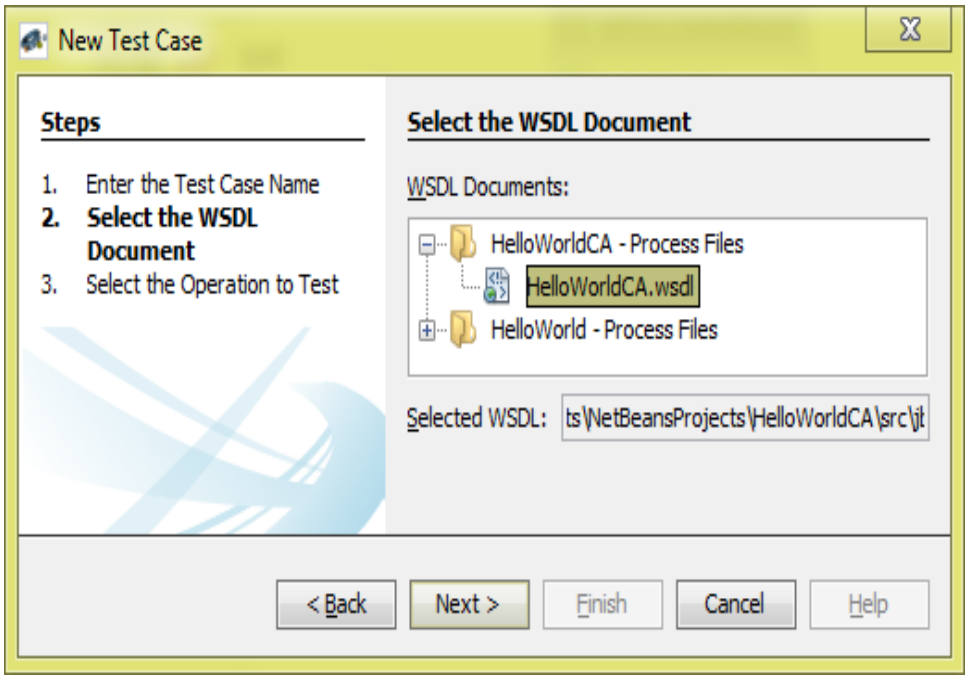


FIGURA 4.55: Seleccionar HelloWordCA.wsdl.

Fuente: Propia

4. Seleccionar HelloWorldWSDLOperation y clic en Terminan.

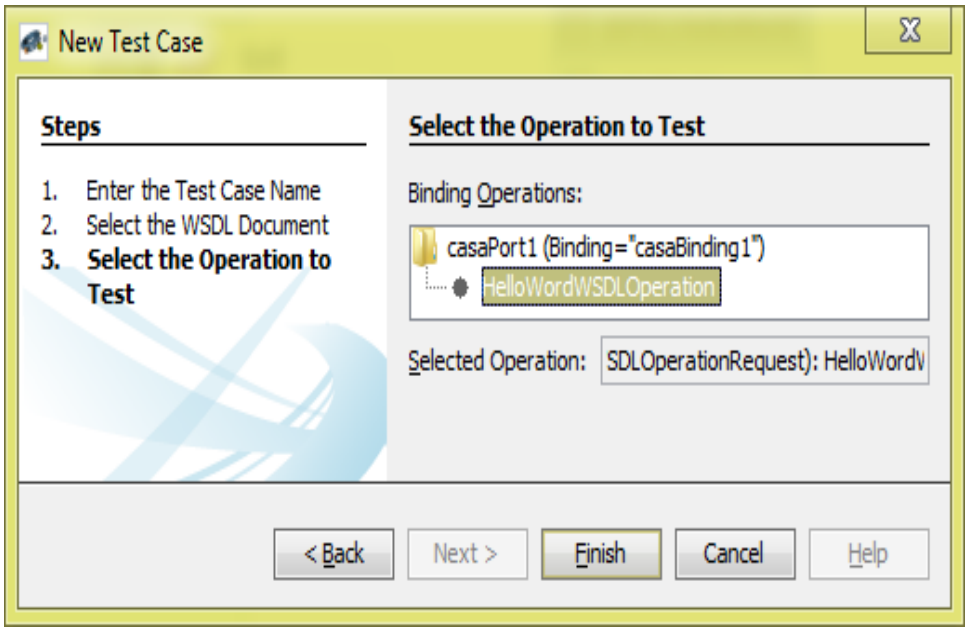


FIGURA 4.56: Seleccionar HelloWordWSDLOperation.

Fuente: Propia

5. El Editor XML se abre. Escribir un nombre en el Part1.

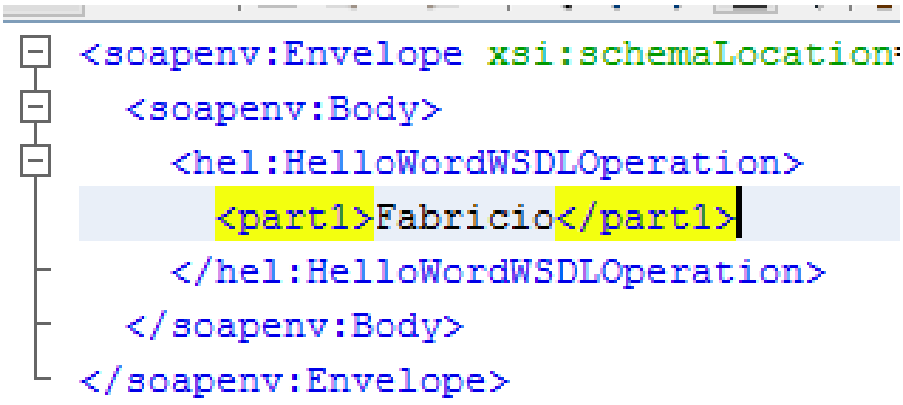


FIGURA 4.57: Escribir un nombre al proyecto.

Fuente: Propia

6. Guardar el documento XML Aparece el Test “Prueba”. Correr.

7. Prueba. Conteste “sí” ya hacer doble clic en la “Output” y no tomar en cuenta el mensaje “Failed ' asoció al resultado.

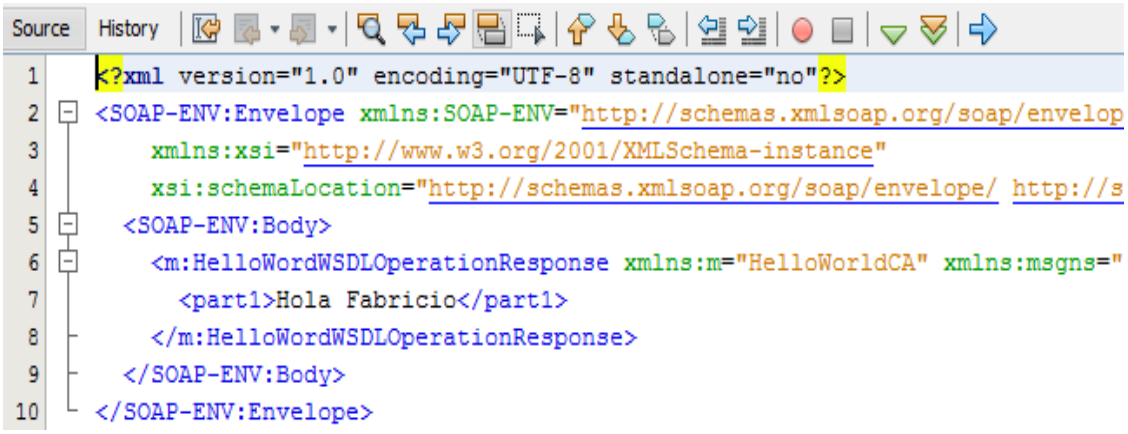


FIGURA 4.58: Resultado del Test.

Fuente: Propia

4.5 VENTAJAS Y DESVENTAJAS DE OPENESB.

4.5.1 VENTAJAS.

El libro (Exposito & Diop, 2014) define las siguientes ventajas:

- Tiene integración con el servidor de aplicaciones Glassfish de Sun.
- OpenESB emplea gran variedad de herramientas que faciliten su uso. Viene integrado con Netbeans que es un entorno de desarrollo muy confiable y que tiene una interfaz gráfica que permite la creación de las aplicaciones SOA que son fáciles de ser desplegadas en el bus de integración. Cuenta además con un editor bpel, así como editores de documentos wsdl, xsd, que facilitan el desarrollo de este tipo de aplicaciones.
- Integración con múltiples componentes. En sus versión actualizadas cuenta con cuatro Services Engines (XSLT, IEP BPEL, Java EE) y ocho componentes Binding. Además el equipo que desarrolla OpenESB se encuentra fabricando componentes adicionales.
- Presenta el motor JavaEE, que le permite la integración entre varios componentes JBI y Web Services que existen en Glassfish utilizando el motor BPEL.
- Existe una cantidad de documentación sobre su uso, con muchos ejemplos prácticos que ayudan a que la curva de aprendizaje sea mucho más rápida.

4.5.2 DESVENTAJAS.

- El gran dificultad de OpenESB es el no poder ser ejecutado como un contenedor independiente (standalone) (Exposito & Diop, 2014)
- No hay un único punto para administrar el sistema completo. Cada componente y cada servicio debe ser administrado por cada instancia JBI local. (José, Leszek, & Joaquim, 2012)

- Cada operación del sistema (desplegar, instalar, iniciar, detener) requiere un exhaustivo conocimiento de la topología del sistema. Por ejemplo, desplegar una aplicación JBI requeriría conocer la localización física en el sistema de los componentes utilizados. (José, Leszek, & Joaquim, 2012)

CAPÌTULO V

5 ESTUDIO COMPARATIVO DE LOS BUSES DE SERVICIO EMPRESARIALES.

5.1 INTRODUCCIÓN.

En este capítulo se centrará en establecer parámetros básicos de comparación que permitirán la elección de un bus de servicios empresariales adecuado que ayude la futura implementación de una aplicación con la arquitectura SOA, y que utilice la plataforma JavaEE.

Estos parámetros se van a establecer en base al estudio de los temas como son:

- ✓ Definición de la Arquitectura Orientada a Servicios y las características que son necesarias para la utilización de este paradigma en la implementación de una solución del tipo empresarial.
- ✓ Definición de la Arquitectura JavaEE y los buses de servicios empresariales existentes en el medio que implementan este tipo de arquitectura referencial posibilitando la elaboración de aplicaciones SOA.
- ✓ Definición de los lineamientos propuestos para la implementación de SOA con buses de servicios empresariales.

Se ha establecido para el desarrollo de la investigación, aunque no es la única forma de elaborar soluciones SOA.

La implementación mediante Buses de servicios empresariales, tomando en cuenta que estos estándar se encuentra debidamente soportado en múltiples plataformas, y permite el acceso a una variedad de Servicios, los cuales son soportados por computadores, también por dispositivos móviles y cualquier otra implementación que tenga acceso a Internet.

Logrando de alguna manera cumplir con otro de los principios importantes de SOA, el cual es brindar el acceso a los servicios de las aplicaciones sin importar el lenguaje y la plataforma de implementación del lado del cliente. Cabe recalcar que existen servicios que consumen información de otros, constituyéndose los segundos en clientes, que posiblemente no se encuentren implementados en las plataformas iguales o a su vez en el mismo leguaje.

La creciente cantidad de aplicaciones y sistemas heterogéneos que deben comunicarse entre ellos, no sólo de una manera interna dentro de la organización, sino también entre los diferentes sistemas de distintas organizaciones, han servido de impulso para que haya la aparición de productos de integración como son los ESB. En el campo de las fuentes abiertas es posible encontrar un amplio conjunto de buses de servicios empresariales, desarrollados por comunidades de software libre y empresas.

En los ESB se nota una mayor diferenciación entre la versión que existe en la comunidad (Community) y la versión de tipo comercial (Enterprise). En muchos casos, el producto llega inclusive a cambiar de nombre.

La página web (Cenatic, 2014) nos presenta los factores para la elección de un ESB.

Cualquier solución podría servir de ayuda para dar una respuesta a los problemas de la interoperabilidad a los que se enfrentará la Administración. Entre los factores para tener en cuenta a la hora de seleccionar una u otra opción podría situarse los siguientes:

- ✓ Los protocolos con los que el ESB puede trabajar (JMS, HTTP, etc.): los sistemas que pretende integrar, hay que determinarán las necesidades de los protocolos que debe manejar el ESB.
- ✓ El cumplimiento de los estándares: existen 2 grupos que son diferenciados claramente, los que cumplen el estándar JBI (Java Business Integration) y los que han optado por utilizar estándares propios. En principio, el estándar ayudaría para que los desarrollos que se realizan fueran portables, es decir, que se pueda utilizarlos en otros productos ESB que también cumplen las estándar.

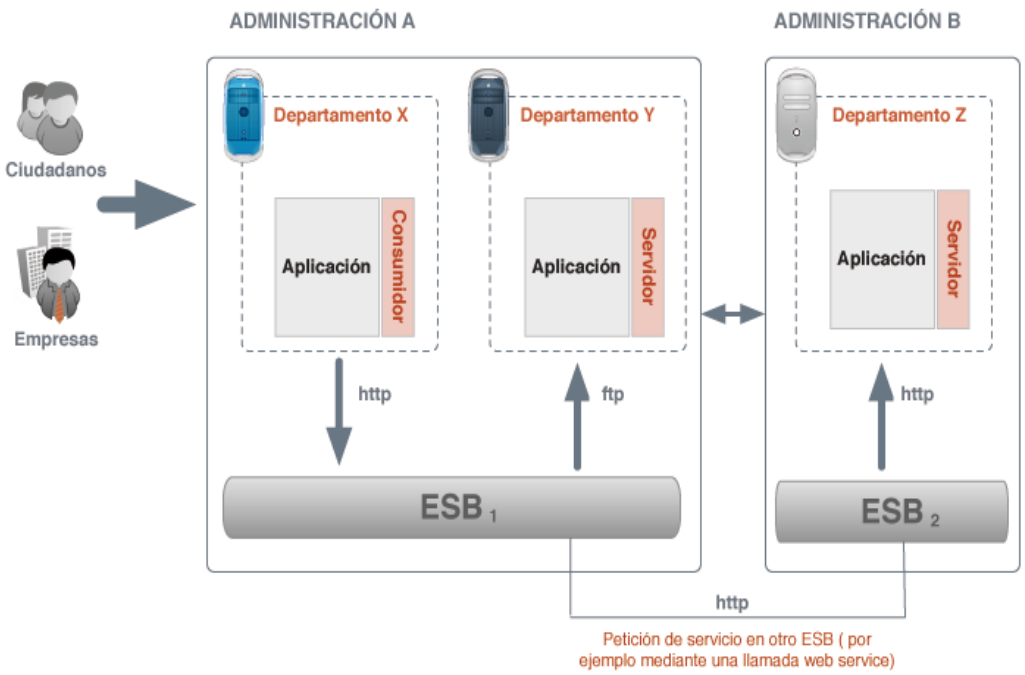


FIGURA 5.59: Diagrama funcional.

Fuente: http://www.cenatic.es/laecsp/page21/page22/page22.html

5.2 ESB PARA SOLUCIONES CON SOA.

Se necesita entonces tomar la mejor decisión sobre qué Bus de Servicios Empresarial de JavaEE se ajusta a las capacidades exigidas para poder implementar y el correcto despliegue de una aplicación, que permita el acceso a la información desde distintas plataformas, y que cada una de las implementaciones adicionales no signifique una reingeniería del proceso completo, que haga incurrir en gastos significativos a la organización o empresa.

Hay una gran variedad de Buses de Servicios Empresariales que implementan la arquitectura JavaEE, y tomando como referencia la utilización y popularidad de los ESB se puede distinguir a dos ESB que se utilizan mayoritariamente en el mundo empresarial. Los dos ESB tienen sus versiones libres que son mantenidas por Comunidades de Desarrolladores en Internet, y adicionalmente existen las versiones de paga, que se diferencian de las anteriores porque implementan mecanismos mucho más robustos en lo que tiene que ver a Seguridad.

Los ESB expuestos entonces son:

✓ Mule ESB v 3.5.

✓ OpenESB v 2.3.

Todos estos ESB se someterán al estudio previamente configurado en un equipo con el Sistema Operativo Ubuntu.

5.3 ESTABLECIMIENTO DE LOS PARÁMETROS PARA LA ELECCIÓN DE UN ESB PARA SOA.

Las características que se van a exponer aquí en este estudio son estándares de la industria y que fueron propuestos por varios autores y expertos en el desarrollo de aplicaciones SOA para la plataforma JavaEE:

Desarrollo.

- ✓ Herramientas basadas en Eclipse.
- ✓ Herramientas basadas en el IDE NetBEANS.
- ✓ Creación de Drag-and-Drop.
- ✓ Protocolos de transporte JDBC, JMS, HTTP.
- ✓ Arrastrar y soltar Servicios Web y REST.
- ✓ Experiencia.

Capacidades de integración.

- ✓ Creación y habilitación de los Servicios.
- ✓ Soporte de patrones de integración Enterprise.
- ✓ Nube de integración y oferta de plataforma (iPaaS).
- ✓ Enrutamiento de mensajes basado en contenido.

- ✓ Mensajería.
- ✓ Integración de datos.
- ✓ Transformación de datos XML del mensaje (XSLT, XQuery…).
- ✓ Orquestación de servicios.
- ✓ Seguridad.

Despliegue

- ✓ Flexibilidad y despliegue del contenedor.
- ✓ Soporte OSGI.
- ✓ SOAP (WSDL).
- ✓ WS-s*, a través del servicio de token de seguridad (STS)
- ✓ Alta Disponibilidad

Administración y Gestión.

- ✓ Monitoreo JMX.
- ✓ Supervisión de actividades de servicios.

Soporte y Documentación.

- ✓ Comunidad.
- ✓ Documentación.
- ✓ Ejemplos con código.
- ✓ Acceso al soporte técnico.
- ✓ Acceso al código fuente.

5.4 ESTUDIO COMPARATIVO.

TABLA 5.1: Parámetros para la elección de un ESB para SOA.

BUSES DE SERVICIO EMPRESARIALES.			
Parámetro de Evolución	**Mule ESB**	**OpenESB**	**Argumento**
Desarrollo			
Herramientas basadas en Eclipse	Sí	No	Mule ESB está integrado con Eclipse
Herramientas basadas en el IDE NetBEANS	No	Sí	OpenESB está integrado con NetBEANS
Creación de Drag-and-Drop	Sí	Sí	Creación rápida con los componentes que posee cada ESB
Protocolos de transporte JDBC, JMS, HTTP etc.	Parcial	Sí	OpenESB soporta: JMS, HTTP, SOAP, REST, FTP, email, sistema de ficheros, mule ESB soporta: JMS, HTTP, email, FTP
Arrastrar y soltar Servicios Web y REST	No	Sí	En OpenESB existe la posibilidad de arrastrar y soltar los Servicios Web, a diferencia de Mule ESB que no tiene esta funcionalidad.
Experiencia	Parcial	Parcial	Experiencia del desarrollador
Capacidades de integración			
Creación y habilitación de los Servicios	Parcial	Si	OpenESB proporciona una creación rápida de servicios a diferencia de Mule ESB.
Soporte de patrones de integración	Si	Si	Los patrones que soportan son: Ruteo Basado en Contenido, Ruteo Basado en Itinerario, Scatter-Gather, Messaging Gateway, Canonical Data Model, Patrón Cache.

Nube de integración y oferta de plataforma (PaaS)	Sí	Sí	Integración Oracle Cloud Services, Google App Engine, Heroku, Cloud Foundry, Microsoft Azure, Amazon AWS.
Enrutamiento de mensajes basado en contenido	Parcial	Sí	OpenESB realiza el enrutamiento basado en reglas, mule ESB no lo realiza.
Mensajería	Sí	Sí	Igual normalización de mensajes recibe y los transmite un ESB.
Integración de servicios	Sí	Sí	Igual Integración como con Facebook, Twitter, Amazon, Paypal, etc.
Transformación de datos XML del mensaje (XSLT, XQuery…)	Sí	Sí	Igual transformación de datos XML a través de XSLT y XSLTComponent, XQuery
Orquestación de servicios	Si	Sí	Igual capacidad para combinar varios servicios existentes
Seguridad	Sí	Sí	Características similares en la Seguridad al utilizar los ESB
Despliegue			
Flexibilidad y despliegue del contenedor	Sí	Parcial	Mule ESB su contenedor es más flexible y de rápido despliegue, OpenESB necesita más tiempo para poder desplegar su contenedor.
Soporte OSGI	Parcial	Parcial	No Soportan completamente el despliegue de las aplicaciones en paquetes
SOAP (WSDL)	Sí	Sí	Protocolo estándar que ambos ESB utilizan
Alta Disponibilidad	Sí	Sí	Garantizan el funcionamiento ininterrumpido del negocio

Administración y Gestión			
Monitoreo JMX	Parcial	Sí	OpenESB permite usar los servicios de monitorización/administración de aplicaciones en caliente, mule ESB no tiene esta capacidad.
Supervisión de actividades de servicios	Sí	Sí	Igual control de las actividades que realizan los servicios
Soporte y Documentación			
Comunidad	Parcial	Sí	OpenESB tiene una comunidad muy extensa y activa
Documentación	Sí	Sí	Similar documentación.
Ejemplos con código	Sí	Sí	Variedad de ejemplos para cada ESB
Acceso al soporte técnico	Sí	Sí	
Acceso al código fuente	Sí	Sí	

Fuente: Propia

Basada en el estudio de expertos en SOA, que han comparado variedad se ESB podemos decir que se ha escogido en razón a que las versiones de los buses de servicios empresariales sometidas al estudio comparativo han evolucionado sus implementaciones para estar acorde a los estándares de Referencia JavaEE. Es así que las versiones utilizadas para los dos ESB son:

- ✓ Mule ESB en la versión 3.5.
- ✓ OpenESB en la versión 2.3.

Para valorar se expone la tabla 5.2 donde muestra los parámetros de valores que se han puesto a cada parámetro de elección del ESB.

TABLA 5.2: Tabla de valoración de parámetros para la elección del ESB para SOA.

Detalle	Valor
Sí	3
Parcial	2
No	1

Fuente: Propia

5.5 PONDERACIONES

TABLA 5.3: Parámetros para la elección de un ESB para SOA (Ponderaciones).

BUSES DE SERVICIO EMPRESARIALES			
Parámetro de Evolución	**Mule ESB**	**OpenESB**	**Argumento**
Desarrollo			
Herramientas basadas en Eclipse	3	1	Mule ESB está integrado con Eclipse
Herramientas basadas en el IDE NetBEANS	1	3	OpenESB está integrado con NetBEANS
Creación de Drag-and-Drop	3	3	Creación rápida con los componentes que posee cada ESB
Protocolos de transporte JDBC, JMS, HTTP	2	3	OpenESB soporta: JMS, HTTP, SOAP, REST, FTP, email, sistema de ficheros, mule ESB soporta: JMS, HTTP, email, FTP
Arrastrar y soltar Servicios Web y REST	1	3	En OpenESB existe la posibilidad de arrastrar y soltar los Servicios Web, a diferencia de Mule ESB que no tiene esta funcionalidad.
Experiencia	2	2	Experiencia del desarrollador
Capacidades de integración			
Creación y habilitación de los Servicios	2	3	OpenESB proporciona una creación rápida de servicios a diferencia de Mule ESB.
Soporte de patrones de integración	3	3	Los patrones que soportan son: Ruteo Basado en Contenido, Ruteo Basado en Itinerario, Scatter-Gather, Messaging Gateway, Canonical Data Model, Patrón Cache.

Nube de integración y oferta de plataforma (PaaS)	3	3	Integración Oracle Cloud Services, Google App Engine, Heroku, Cloud Foundry, Microsoft Azure, Amazon AWS.
Enrutamiento de mensajes basado en contenido	2	3	OpenESB realiza el enrutamiento basado en reglas, mule ESB no lo realiza.
Mensajería	3	3	Igual normalización de mensajes recibe y los transmite un ESB.
Integración de servicios	3	3	Igual Integración como con Facebook, Twitter, Amazon, Paypal, etc.
Transformación de datos XML del mensaje (XSLT, XQuery…)	3	3	Igual transformación de datos XML a través de XSLT y XSLTComponent, XQuery
Orquestación de servicios	3	3	Igual capacidad para combinar varios servicios existentes
Seguridad	3	3	Características similares en la Seguridad al utilizar los ESB
Despliegue			
Flexibilidad y despliegue del contenedor	3	2	Mule ESB su contenedor es más flexible y de rápido despliegue, OpenESB necesita más tiempo para poder desplegar su contenedor.
Soporte OSGI	2	2	No Soportan completamente el despliegue de las aplicaciones en paquetes
SOAP (WSDL)	3	3	Protocolo estándar que ambos ESB utilizan
Alta Disponibilidad	3	3	Garantizan el funcionamiento ininterrumpido del negocio

Administración y Gestión			
Monitoreo JMX	2	3	OpenESB permite usar los servicios de monitorización/administración de aplicaciones en caliente, mule ESB no tiene esta capacidad.
Supervisión de actividades de servicios	3	3	Igual control de las actividades que realizan los servicios
Soporte y Documentación			
Comunidad	2	3	OpenESB tiene una comunidad muy extensa y activa
Documentación	3	3	Similar documentación.
Ejemplos con código	3	3	Variedad de ejemplos para cada ESB
Acceso al soporte técnico	3	3	
Acceso al código fuente	3	3	

Fuente: Propia

5.6 EXPLICACIÓN.

De la tabla anterior si todos los buses de servicios empresariales cumplieran con todos los parámetros de evaluación se lograría la suma de 78 lo que representaría el 100% de efectividad del bus de servicios;

De tal manera para la representar la valoración de cada bus de servicios empresarial se empleara una regla de tres simple, la cual es de la siguiente manera:

78 100%

67 X = (100*69)/78 = 85.89%

Obteniendo los siguientes resultados.

TABLA 5.4: Porcentajes de Análisis comparativo de ESB para SOA.

Bus de Servicios Empresarial	Mule ESB	OpenESB
Porcentaje	85.89%	93.58%

Fuente: Propia

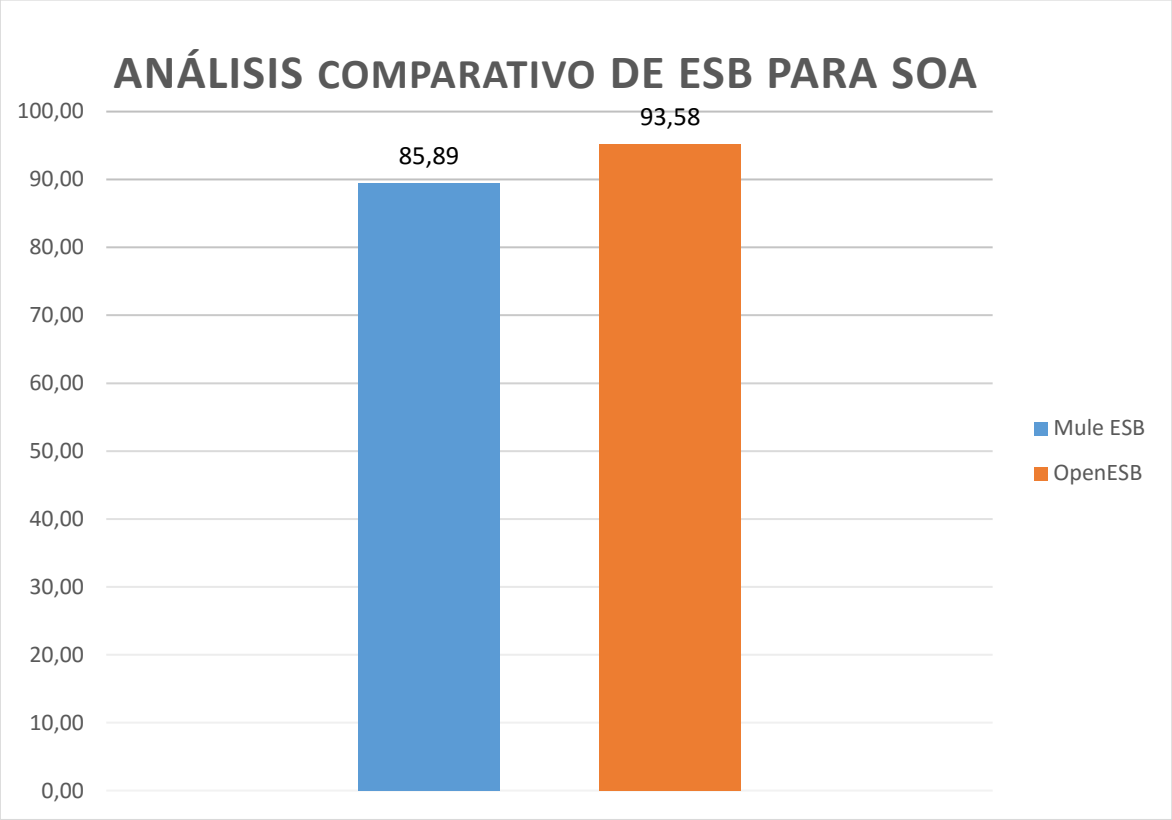


FIGURA 5.60: Análisis Estadístico de ESB para SOA.

Fuente: Propia

5.7 RESULTADOS OBTENIDOS.

OpenESB es el bus de servicios Empresariales que obtiene el mejor resultado con un 93.58%, debido a sus características, facilidad de acoplamiento, integración con herramientas, Con un 85.89% se encuentra Mule ESB y se encuentra ubicado en la segunda posición.

OpenESB es un Bus de Servicios Empresarial de código abierto, de fácil instalación, posee un soporte completo con Java EE, integración total con las herramientas de desarrollo, y es una base importante para la implementación de SOA en JAVA. El otro ESB posee deficiencias en algunas de las características mencionadas en la tabla 5.3.

5.8 DESCRIPCIÓN.

La utilización de un Bus de Servicios Empresarial adecuado, permitirá tener aspectos fundamentales que ayuden a un óptimo rendimiento del ESB en la arquitectura orientada a servicios los cuales reducirán los tiempos de respuesta de acceso a la información, los parámetros de comparación serán cuatro los cuales son:

- ✓ Servicios Proxy.
- ✓ Componente CBR (Content Based Routing).
- ✓ Transformación XSL.
- ✓ WS-Security.

5.8.1 SERVICIO PROXY.

FIGURA 5.61: Servicios Proxy.

Fuente: http://esbperformance.org/display/comparison/Framework+-+Performance+Test+Cases

Este escenario demostrará la capacidad del ESB seleccionado para actuar como capa de virtualización para los servicios web back-end de manera similar a cómo se utilizaron los servidores web Apache o equilibradores de carga de hardware frente a backends de Internet para la virtualización, la mayoría de ESB admiten estas características. Este escenario supone que el servicio sea un servicio Web SOAP, y por lo tanto espera que el ESB presente el WSDL del servicio a sus clientes.

Requisitos:

El servicio Proxy debe estar disponible a través de HTTP en una URL de la forma: http://localhost/:<puerto>/<path>/DirectProxy.

El servicio proxy debe exponer su WSDL en http a una URL de la forma: http://localhost/:<puerto>/<path>/DirectProxy?wsdl.

5.8.2 COMPONENTE CBR (CONTENT BASED ROUTING).

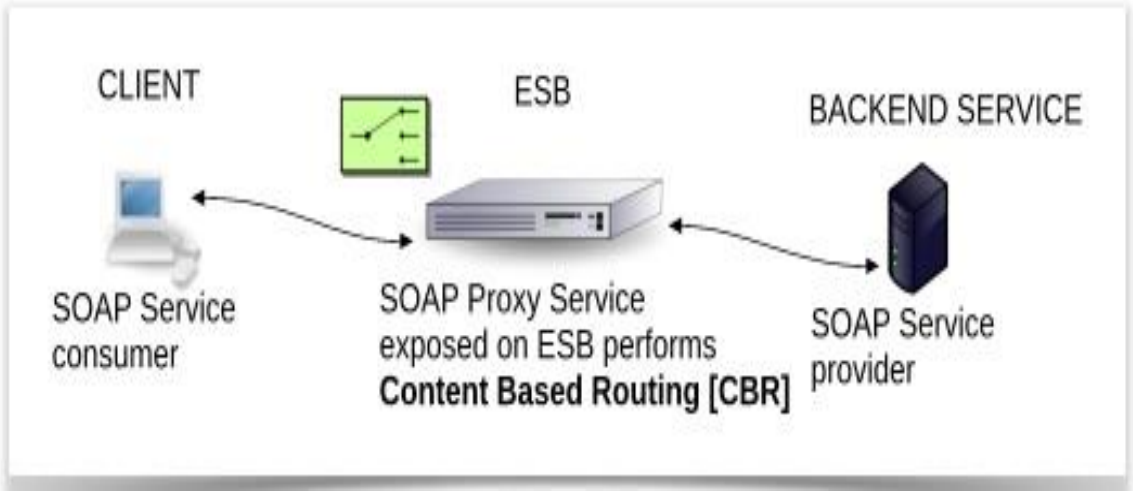


FIGURA 5.62: Componente CBR (Content Based Routing).
Fuente: http://esbperformance.org/display/comparison/Framework+-+Performance+Test+Cases

Normalmente, este patrón se utiliza en la práctica para enrutar los mensajes de solicitud a los diferentes criterios de valoración, o tratarlos de forma diferente en el ESB basado en el mensaje de atributos como la carga útil, o cabeceras de transporte, etc. Este escenario se presenta ahora en tres variantes:

- ✓ CBR en una cabecera de transporte HTTP.
- ✓ CBR en un encabezado SOAP.
- ✓ CBR en el cuerpo SOAP / Payload.
- ✓

5.8.2.1 CBR EN UNA CABECERA DE TRANSPORTES.

El primer escenario evalúa la manera más óptima para enrutar mensajes, basado en las cabeceras a nivel de transporte HTTP. Esta técnica es ampliamente utilizada en el tendido de Hesse o de otras cargas útiles que no sean SOAP, como es extremadamente eficiente. La condición realiza una sencilla comparación de String sobre un encabezado HTTP.

5.8.2.2 CBR EN UNA CABECERA SOAP.

Este escenario lleva a cabo una decisión de enrutamiento en un encabezado SOAP, ya que podría ser mejor optimizado que procesar un gran volumen de carga útil SOAP. La condición lleva a cabo una evaluación XPath sobre un encabezado SOAP. Este escenario de prueba extrae un encabezado SOAP especificado como una simple String y, a continuación, realiza una comparación con un string constante dado.

5.8.2.3 CBR EN EL CUERPO SOAP / PAYLOAD.

Este escenario demostrará la capacidad del ESB seleccionado para enrutar una carga XML basado en la evaluación de una expresión XPath.

5.8.3 TRANSFORMACIÓN XSL.

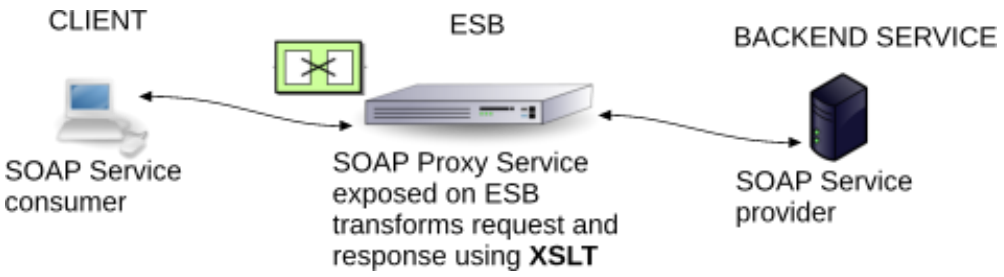


FIGURA 5.63: Transformación XSL.

Fuente: http://esbperformance.org/display/comparison/Framework+-+Performance+Test+Cases

Este escenario demostrará la capacidad del ESB seleccionado para transformar mensajes XML de carga útil (como SOAP) entre los diferentes esquemas. En la práctica las transformaciones se utilizan para convertir mensajes de XML a CSV o texto.

5.8.4 WS-SECURITY.

FIGURA 5.64: Componente CBR (Content Based Routing).

Fuente: http://esbperformance.org/display/comparison/Framework+-+Performance+Test+Cases

Este escenario demostrará la capacidad del ESB seleccionado para actuar como Gateway Security. Típicamente un ESB se utiliza para validar y verificar las credenciales de seguridad, y establecer la autenticidad de los mensajes, antes de pasarlos a través de la mediación y de los servicios internos de un entorno corporativo.

5.9 LOS ESCENARIOS DE PRUEBAS.

Los escenarios de prueba son principalmente en torno al protocolo HTTP para cargas útiles SOAP / XML que por cierto constituye una gran parte de los mensajes enviados a través de los ESB. El tamaño de los mensajes se varió de 512 bytes a 100K bytes y el número simultáneo de usuarios 20-2560 para evaluar cada ESB bajo diferentes niveles de carga.

Tamaño de los mensajes y el número de usuarios simultáneos.

La prueba de rendimiento utiliza mensajes de tamaño:

- ✓ 512 bytes.
- ✓ 1 Kbytes.
- ✓ 5 Kbytes.
- ✓ 10 Kbytes.
- ✓ 100 Kbytes.

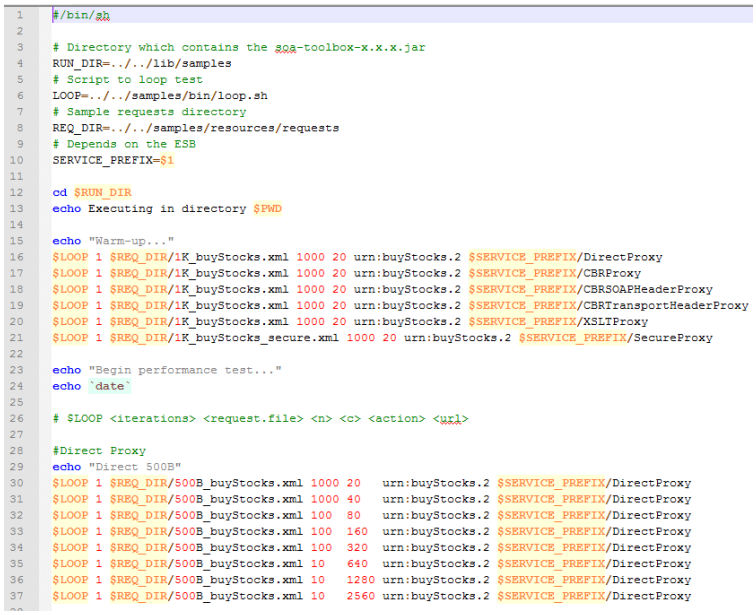
Con el número simultáneo de usuarios:

- ✓ 20 usuarios simultáneos.
- ✓ 40 usuarios simultáneos.
- ✓ 80 usuarios simultáneos.
- ✓ 160 usuarios simultáneos.
- ✓ 320 usuarios simultáneos.
- ✓ 640 usuarios simultáneos.
- ✓ 1280 usuarios concurrentes.
- ✓ 2560 usuarios concurrentes.

En base al estudio comparativo de los Buses de Servicios Empresariales propuestos y el grado de cumplimiento en los parámetros que ya se definieron, después de haber establecido estos mediante el estudio de SOA, se ha procedido a elegir el ESB adecuado. Entonces el objetivo primordial de la investigación es demostrar que el Bus de Servicios Empresarial elegido una vez realizado el estudio comparativo, asegurará un rendimiento correcto y brindará el menor tiempo de respuesta de acceso a la información de las aplicaciones cliente.

5.10 PRUEBAS DE RENDIMIENTO

En los ESB desplegará una aplicación que simulara él envió de mensajes que variara el tamaño de 512 bytes a 100Kbytes y un número simultaneo de 20 a 2560 usuarios, para simular este ambiente de trabajo utilizaremos el archivo "loadtest.sh" que incluye la lista completa de los escenarios de prueba, el archivo tiene la siguiente estructura.



```
#/bin/sh

# Directory which contains the soa-toolbox-x.x.x.jar
RUN_DIR=../../lib/samples
# Script to loop test
LOOP=../../samples/bin/loop.sh
# Sample requests directory
REQ_DIR=../../samples/resources/requests
# Depends on the ESB
SERVICE_PREFIX=$1

cd $RUN_DIR
echo Executing in directory $PWD

echo "Warm-up..."
$LOOP 1 $REQ_DIR/1K_buyStocks.xml 1000 20 urn:buyStocks.2 $SERVICE_PREFIX/DirectProxy
$LOOP 1 $REQ_DIR/1K_buyStocks.xml 1000 20 urn:buyStocks.2 $SERVICE_PREFIX/CBRProxy
$LOOP 1 $REQ_DIR/1K_buyStocks.xml 1000 20 urn:buyStocks.2 $SERVICE_PREFIX/CBRSOAPHeaderProxy
$LOOP 1 $REQ_DIR/1K_buyStocks.xml 1000 20 urn:buyStocks.2 $SERVICE_PREFIX/CBRTransportHeaderProxy
$LOOP 1 $REQ_DIR/1K_buyStocks.xml 1000 20 urn:buyStocks.2 $SERVICE_PREFIX/XSLTProxy
$LOOP 1 $REQ_DIR/1K_buyStocks_secure.xml 1000 20 urn:buyStocks.2 $SERVICE_PREFIX/SecureProxy

echo "Begin performance test..."
echo `date`

# $LOOP <iterations> <request.file> <n> <c> <action> <url>

#Direct Proxy
echo "Direct 500B"
$LOOP 1 $REQ_DIR/500B_buyStocks.xml 1000 20   urn:buyStocks.2 $SERVICE_PREFIX/DirectProxy
$LOOP 1 $REQ_DIR/500B_buyStocks.xml 1000 40   urn:buyStocks.2 $SERVICE_PREFIX/DirectProxy
$LOOP 1 $REQ_DIR/500B_buyStocks.xml 100  80   urn:buyStocks.2 $SERVICE_PREFIX/DirectProxy
$LOOP 1 $REQ_DIR/500B_buyStocks.xml 100  160  urn:buyStocks.2 $SERVICE_PREFIX/DirectProxy
$LOOP 1 $REQ_DIR/500B_buyStocks.xml 100  320  urn:buyStocks.2 $SERVICE_PREFIX/DirectProxy
$LOOP 1 $REQ_DIR/500B_buyStocks.xml 10   640  urn:buyStocks.2 $SERVICE_PREFIX/DirectProxy
$LOOP 1 $REQ_DIR/500B_buyStocks.xml 10   1280 urn:buyStocks.2 $SERVICE_PREFIX/DirectProxy
$LOOP 1 $REQ_DIR/500B_buyStocks.xml 10   2560 urn:buyStocks.2 $SERVICE_PREFIX/DirectProxy
```

FIGURA 5.65: Estructura del archivo loadtest.sh.

Fuente: http://esbperformance.org/display/comparison/Framework+-Extending+the+Test+Suite

Las aplicaciones que se ejecutan en el mundo real tienen un nivel mucho más grande, y manejan recursos de una gran escala, pero mediante este test se podrá obtener información acerca del rendimiento, escalabilidad y fiabilidad de los Buses de Servicios Empresariales.

A continuación se describen los componentes del escenario para realizar las pruebas:

- ✓ El equipo disponible para estas pruebas posee un procesador: 2 generación Intel® Core™ i5-2410M de 2.30GHz.
- ✓ Memoria: 8GB.
- ✓ El sistema operativo es Linux Ubuntu 12.04 de 32bits.
- ✓ Se instalaron los dos ESB.
- ✓ Entorno de desarrollo java JDK 1.7.

Con el entorno de para las pruebas ya completado solo resta realizar las pruebas para cada ESB ya configurado.

5.10.1 PRUEBAS DE RENDIMIENTO PARA MULE ESB.

Para realizar la prueba de rendimiento de Mule ESB se realizó los siguientes pasos:

1. Ejecutamos el bus de servicios mediante un terminal, nos dirigimos donde se encuentra instalado Mule ESB ingresamos a la carpeta “bin” y ejecutamos el archivo “mule”.

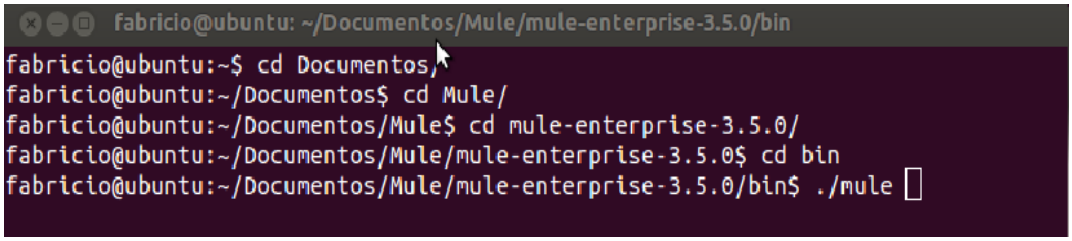


FIGURA 5.66: Iniciar Mule ESB.

Fuente: Propia

2. Ser observa como Mule ESB se ejecuta.

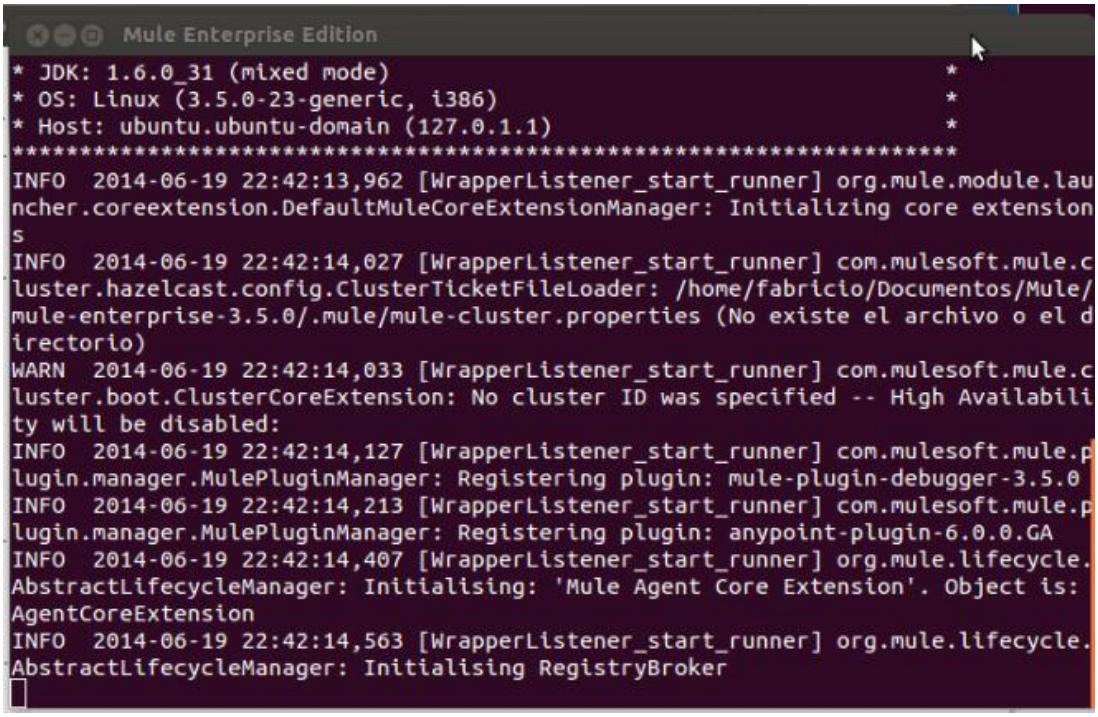


FIGURA 5.67: Ejecución de Mule ESB.

Fuente: Propia

3. Cuando ya está ejecutándose Mule ESB abrir otra terminal donde se va ejecutar el archivo que contiene las pruebas que simulara las condiciones que ya se definieron anteriormente, cuando ya tenemos abierto la terminal ingresamos a la dirección del archivo “loadtest.sh” y ejecutamos el siguiente comando “./loadtest.sh http://localhost:8585/service > mule.txt”, donde “http://localhost:8585/service” es la dirección donde se está ejecutando el ESB con los servicios que vamos a probar, el término “mule.txt” es donde se va a guardar los resultados de las pruebas.

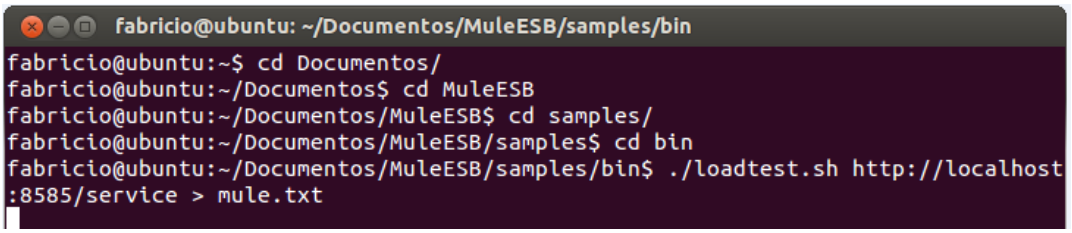


FIGURA 5.68: Ejecución del comando loadtest.sh para Mule ESB.

Fuente: Propia

4. Al ejecutar el comando nos va guardando los datos de la prueba en un archivo tipo txt como muestra la figura.

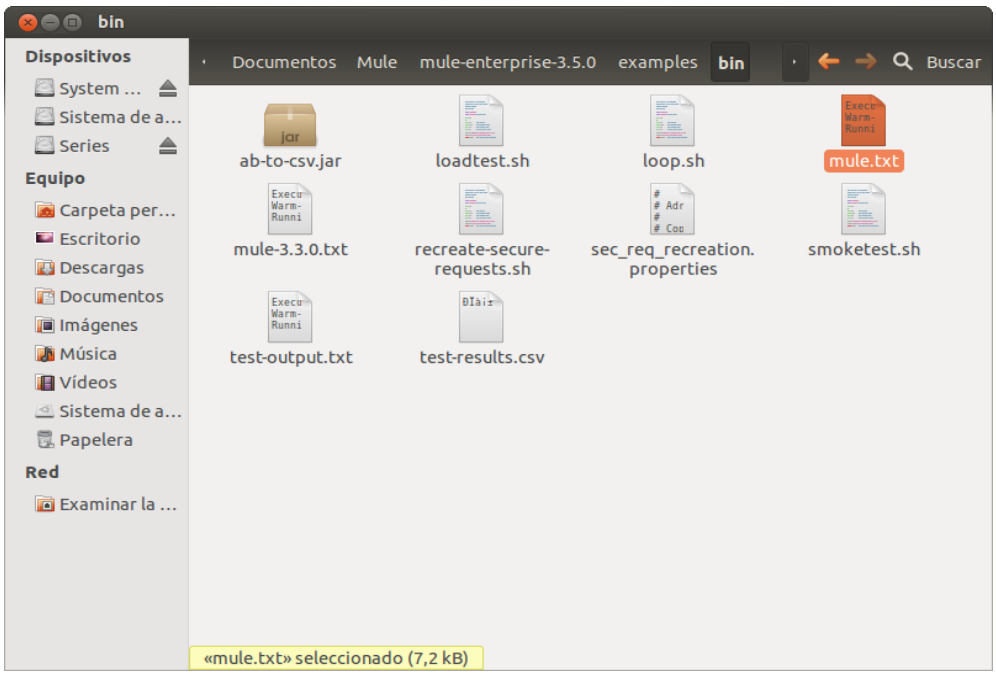


FIGURA 5.69: Archivo txt de las pruebas de Mule ESB.

Fuente: Propia

5. El resultado nos aparece como muestra la Figura 5.11 para revisar todos los resultados más detalladamente se los mostrara en tablas para una fácil interpretación.

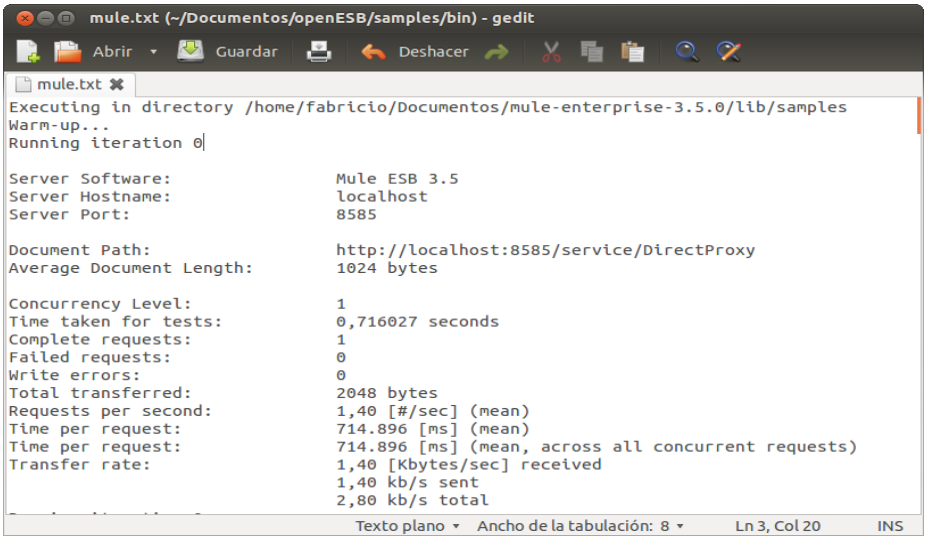


FIGURA 5.70: Resultados de Mule ESB.

Fuente: Propia

5.10.2 PRUEBAS DE RENDIMIENTO DE OPENESB.

1. Ingresamos a openESB y ejecutamos el servidor de GlassFish

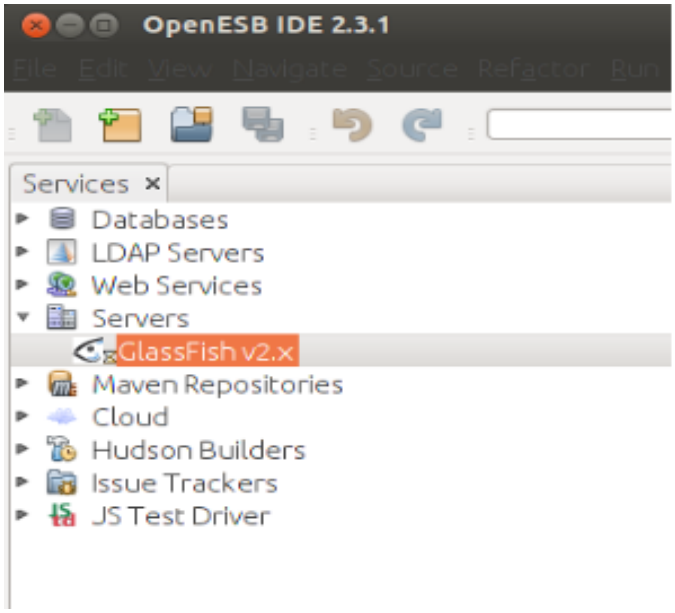


FIGURA 5.71: Ejecución GlassFish v2

Fuente: Propia

2. Cuando ya se encuentre ejecutándose el bus de servicios OpenESB ingresamos a una terminal y ejecutamos el archivo que contiene las pruebas de rendimiento con el siguiente comando.

- ./loadtest.sh http://localhost:8090/service > openESB.txt.

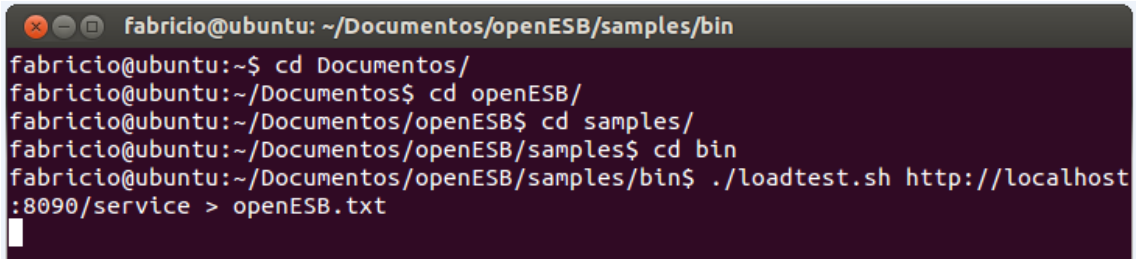


FIGURA 5.72: Ejecución del comando loadtest.sh para OpenESB.

Fuente: Propia

3. Se crea un archivo con los resultados de las pruebas realizadas al bus de servicios empresarial.

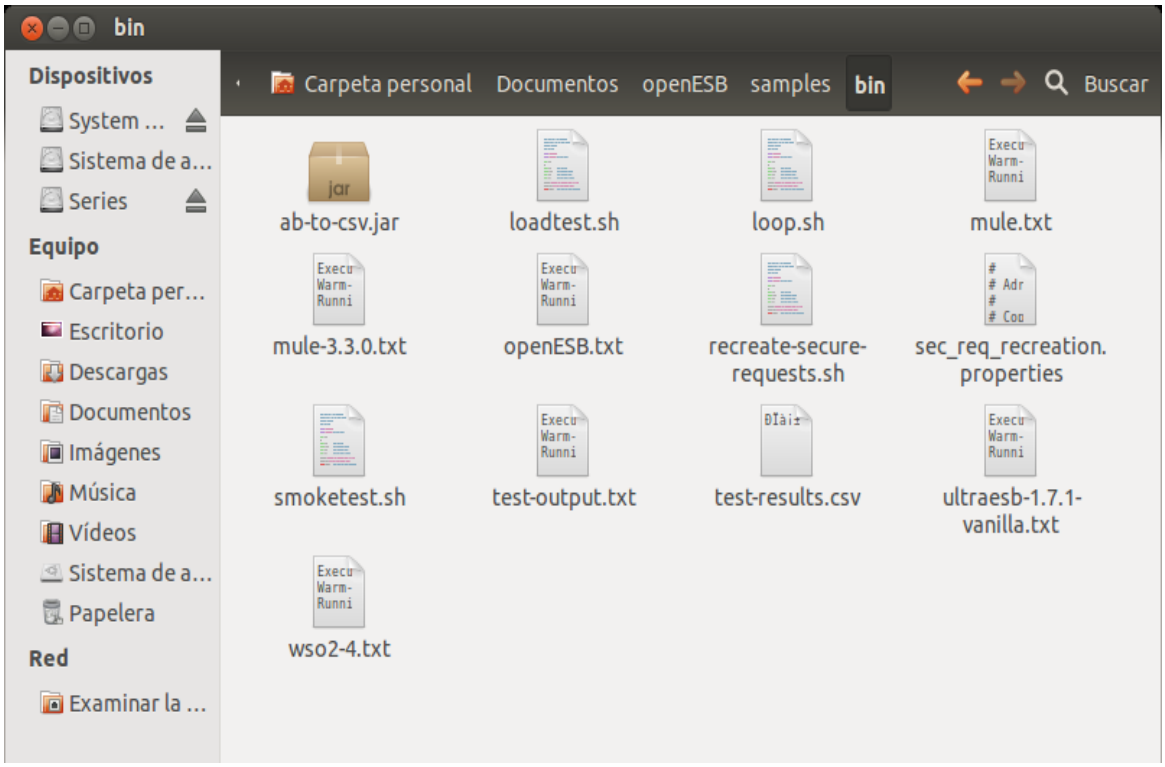


FIGURA 5.73: Archivo txt de las pruebas de OpenESB.

Fuente: Propia

4. Vemos que nos presenta los resultados de las pruebas realizadas al bus de servicios OpenESB.

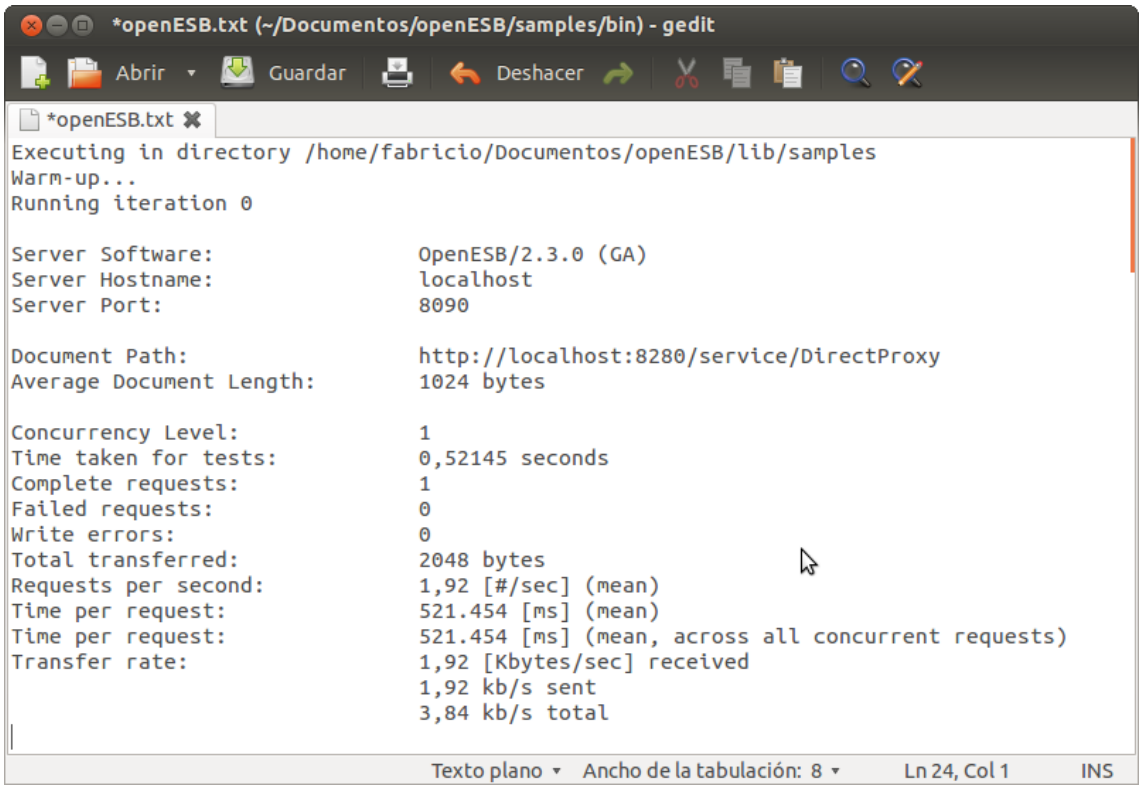


FIGURA 5.74: Resultados de OpenESB.

Fuente: Propia

5.10.3 RESULTADO DE LA PRUEBA.

Tanto Mule ESB como openESB completaron la prueba satisfactoriamente sin errores o corrupciones de mensajes, los resultados completos de la prueba se pueden encontrar en la tabla 5.5.

TPS= Transacciones por segundo; el número promedio de interacciones de la prueba que corrieron exitosamente en un intervalo de un segundo.

TABLA 5.5: Resultados de las pruebas realizadas a los ESB.

				Mule ESB						OpenESB					
Parámetros comparación				**Ronda 1**		**Ronda 2**		**Promedio**		**Ronda 1**		**Ronda2**		**Promedio**	
	Tamaño	Usuarios	Respuesta	Errores	TPS	Errores	TPS	Errores	TPS	Errores	TPS	Errores	TPS	Errores	TPS
DirectProxy	500 b	20	20000	0	2.824	0	2.794	0	2.786	0	4.207	0	4.148	0	4.171
DirectProxy	500 b	40	40000	0	4.622	0	4.454	0	4.575	0	5.033	0	5.139	0	5.082
DirectProxy	500 b	80	8000	0	4.007	0	3.854	0	3.985	0	4.843	0	5.013	0	4.940
DirectProxy	500 b	160	16000	0	4.210	0	4.191	0	4.210	0	5.433	0	5.337	0	5.472
DirectProxy	500 b	320	32000	0	4.104	0	4.044	0	4.199	0	5.721	0	6.006	0	5.817
DirectProxy	500 b	640	64000	0	4.890	0	4.655	0	4.701	0	5.948	0	6.142	0	6.066
DirectProxy	500 b	1280	256000	0	5.150	0	4.942	0	5.066	0	7.276	0	7.124	0	7.217
DirectProxy	500 b	2560	512000	0	4.988	0	4.693	0	4.847	0	6.453	0	6.328	0	6.357
DirectProxy	1K	20	20000	0	2.821	0	2.841	0	2.808	0	4.098	0	4.074	0	4.073
DirectProxy	1K	40	40000	0	4.543	0	4.365	0	4.490	0	4.980	0	4.960	0	4.963
DirectProxy	1K	80	8000	0	4.128	0	3.921	0	4.010	0	4.850	0	4.777	0	4.836
DirectProxy	1K	160	16000	0	4.085	0	3.951	0	4.047	0	5.430	0	5.444	0	5.426
DirectProxy	1K	320	32000	0	4.243	0	4.008	0	4.033	0	5.625	0	5.778	0	5.725
DirectProxy	1K	640	64000	0	4.536	0	4.555	0	4.536	0	6.281	0	5.729	0	6.026
DirectProxy	1K	1280	256000	0	4.986	0	4.887	0	4.981	0	7.165	0	7.038	0	7.064
DirectProxy	1K	2560	512000	0	4.777	0	4.751	0	4.783	0	6.723	0	6.139	0	6.577
DirectProxy	5K	20	20000	0	2.401	0	2.484	0	2.460	0	3.864	0	3.873	0	3.865
DirectProxy	5K	40	40000	0	4.295	0	4.279	0	4.277	0	4.709	0	4.664	0	4.702
DirectProxy	5K	80	8000	0	3.783	0	3.714	0	3.762	0	4.427	0	4.483		4.450

DirectProxy	5K	160	16000	0	4.020	0	3.844	0	3.922	0	4.974	0	4.753	0	4.878
DirectProxy	5K	320	32000	0	4.143	0	4.232	0	4.091	0	5.547	0	5.282	0	5.430
DirectProxy	5K	640	64000	0	4.365	0	4.285	0	4.349	0	5.375	0	5.731	0	5.634
DirectProxy	5K	1280	256000	0	4.706	0	4.672	0	4.691	0	6.414	0	6.300	0	6.351
DirectProxy	5K	2560	512000	0	4.616	0	4.508	0	4.510	0	6.548	0	5.757	0	6.038
DirectProxy	10K	20	20000	0	2.097	0	2.114	0	2.109	0	3.464	0	3.380	0	3.414
DirectProxy	10K	40	40000	0	3.910	0	3.878	0	3.884	0	4.029	0	4.030	0	4.003
DirectProxy	10K	80	8000	0	3.538	0	3.424	0	3.481	0	3.843	0	3.888	0	3.851
DirectProxy	10K	160	16000	0	3.490	0	3.595	0	3.554	0	4.089	0	4.209	0	4.136
DirectProxy	10K	320	32000	0	3.703	0	3.748	0	3.722	0	4.054	0	4.237	0	4.216
DirectProxy	10K	640	64000	0	3.890	0	3.832	0	3.869	0	4.401	0	4.301	0	4.323
DirectProxy	100K	20	20000	0	726	0	718	0	724	0	996	0	985	0	982
DirectProxy	100K	40	40000	0	1.794	0	1.815	0	1.809	0	1.113	0	1.099	0	1.098
DirectProxy	100K	80	8000	0	1.776	0	1.724	0	1.753	0	1.073	0	1.063	0	1.057
DirectProxy	100K	160	16000	0	1.811	0	1.773	0	1.789	0	1.114	0	1.108	0	1.110
DirectProxy	100K	320	32000	0	1.813	0	1.797	0	1.795	0	1.154	0	1.149	0	1.144
DirectProxy	100K	640	64000	0	1.821	0	1.840	0	1.831	0	1.165	0	1.150	0	1.152
Promedio				**0**	**3.656**	**0**	**3.588**	**0**	**3.623**	**0**	**4.512**	**0**	**4.462**	**0**	**4.490**
CBRProxy	500 b	20	20000	0	2.981	0	3.002	0	3.002	0	3.433	0	3.370	0	3.406
CBRProxy	500 b	40	40000	0	2.796	0	4.106	0	3.678	0	4.130	0	4.052	0	4.088
CBRProxy	500 b	80	8000	0	3.881	0	3.822	0	3.830	0	4.167	0	4.034	0	4.119
CBRProxy	500 b	160	16000	0	4.070	0	3.996	0	3.978	0	4.621	0	4.703	0	4.671
CBRProxy	500 b	320	32000	0	4.014	0	4.034	0	4.031	0	4.816	0	5.082	0	4.951
CBRProxy	500 b	640	64000	0	4.098	0	4.347	0	4.152	0	5.328	0	5.241	0	5.183
CBRProxy	500 b	1280	256000	0	4.594	0	4.546	0	4.552	0	5.550	0	5.630	0	5.626

CBRProxy	500 b	2560	512000	0	4.343	0	4.523	0	4.430	0	4.315	0	4.406	0	4.506
CBRProxy	1K	20	20000	0	3.021	0	2.914	0	2.964	0	3.375	0	3.343	0	3.372
CBRProxy	1K	40	40000	0	4.050	0	4.013	0	4.023	0	4.077	0	4.056	0	4.050
CBRProxy	1K	80	8000	0	3.713	0	3.768	0	3.759	0	4.098	0	4.043	0	4.064
CBRProxy	1K	160	16000	0	3.908	0	3.878	0	3.874	0	4.562	0	4.587	0	4.542
CBRProxy	1K	320	32000	0	3.876	0	3.798	0	3.938	0	4.718	0	4.869	0	4.682
CBRProxy	1K	640	64000	0	4.138	0	4.118	0	4.106	0	4.916	0	5.022	0	4.942
CBRProxy	1K	1280	256000	0	4.439	0	4.431	0	4.412	0	5.170	0	5.370	0	5.287
CBRProxy	1K	2560	512000	0	4.274	0	4.453	0	4.283	0	4.560	0	4.047	0	4.354
CBRProxy	5K	20	20000	0	2.568	0	2.518	0	2.536	0	2.737	0	2.631	0	2.586
CBRProxy	5K	40	40000	0	3.621	0	3.621	0	3.608	0	3.166	0	3.041	0	3.083
CBRProxy	5K	80	8000	0	3.358	0	3.345	0	3.348	0	3.231	0	3.099	0	3.150
CBRProxy	5K	160	16000	0	3.471	0	3.360	0	3.436	0	3.537	0	3.318	0	3.400
CBRProxy	5K	320	32000	0	3.688	0	3.629	0	3.597	0	3.694	0	3.463	0	3.511
CBRProxy	5K	640	64000	0	3.717	0	3.713	0	3.703	0	3.652	0	3.513	0	3.551
CBRProxy	5K	1280	256000	0	3.852	0	3.883	0	3.879	0	3.653	0	3.624	0	3.585
CBRProxy	5K	2560	512000	0	3.860	0	3.896	0	3.877	0	3.188	0	2.969	0	3.178
CBRProxy	10K	20	20000	0	1.769	0	1.772	0	1.788	0	1.885	0	1.891	0	1.897
CBRProxy	10K	40	40000	0	3.133	0	3.117	0	3.114	0	2.305	0	2.110	0	2.177
CBRProxy	10K	80	8000	0	2.928	0	2.892	0	2.907	0	2.342	0	2.154	0	2.221
CBRProxy	10K	160	16000	0	3.028	0	3.006	0	3.000	0	2.444	0	2.246	0	2.307
CBRProxy	10K	320	32000	0	3.065	0	3.155	0	3.096	0	2.420	0	2.248	0	2.299
CBRProxy	10K	640	64000	0	3.199	0	3.136	0	3.166	0	2.444	0	2.242	0	2.304
Promedio				**0**	**3.582**	**0**	**3.626**	**0**	**3.602**	**0**	**3.751**	**0**	**3.680**	**0**	**3.703**
CBRSOAPHeaderProxy	500 b	20	20000	0	3.011	0	2.913	0	2.999	0	3.466	0	2.412	0	3.087

CBRSOAPHeaderProxy	500 b	40	40000	0	4.184	0	4.165	0	4.159	0	4.186	0	4.112	0	4.145
CBRSOAPHeaderProxy	500 b	80	8000	0	3.854	0	3.785	0	3.798	0	4.170	0	4.321	0	4.230
CBRSOAPHeaderProxy	500 b	160	16000	0	4.068	0	3.920	0	3.974	0	4.888	0	4.664	0	4.769
CBRSOAPHeaderProxy	500 b	320	32000	0	4.146	0	3.923	0	4.011	0	5.141	0	5.372	0	5.256
CBRSOAPHeaderProxy	500 b	640	64000	0	4.205	0	4.169	0	4.228	0	5.110	0	5.570	0	5.398
CBRSOAPHeaderProxy	500 b	1280	256000	0	4.480	0	4.550	0	4.504	0	5.673	0	5.830	0	5.762
CBRSOAPHeaderProxy	500 b	2560	512000	0	4.390	0	4.312	0	4.323	0	4.895	0	5.162	0	5.098
CBRSOAPHeaderProxy	1K	20	20000	0	3.103	0	2.883	0	2.981	0	3.419	0	3.467	0	3.431
CBRSOAPHeaderProxy	1K	40	40000	0	4.039	0	4.030	0	4.031	0	4.170	0	4.143	0	4.148
CBRSOAPHeaderProxy	1K	80	8000	0	3.823	0	3.737	0	3.764	0	4.243	0	4.192	0	4.214
CBRSOAPHeaderProxy	1K	160	16000	0	3.867	0	3.936	0	3.928	0	4.872	0	4.763	0	4.783
CBRSOAPHeaderProxy	1K	320	32000	0	3.766	0	3.952	0	3.809	0	5.000	0	5.270	0	5.155
CBRSOAPHeaderProxy	1K	640	64000	0	4.272	0	4.154	0	4.183	0	5.352	0	5.222	0	5.233
CBRSOAPHeaderProxy	1K	1280	256000	0	4.450	0	4.412	0	4.425	0	5.872	0	5.575	0	5.725
CBRSOAPHeaderProxy	1K	2560	512000	0	4.518	0	4.290	0	4.383	0	4.878	0	5.125	0	5.001
CBRSOAPHeaderProxy	5K	20	20000	0	2.488	0	2.424	0	2.475	0	3.225	0	3.277	0	3.256
CBRSOAPHeaderProxy	5K	40	40000	0	3.583	0	3.617	0	3.604	0	3.938	0	3.845	0	3.908
CBRSOAPHeaderProxy	5K	80	8000	0	3.391	0	3.317	0	3.344	0	3.964	0	3.944	0	3.930
CBRSOAPHeaderProxy	5K	160	16000	0	3.401	0	3.487	0	3.459	0	4.460	0	4.367	0	4.384
CBRSOAPHeaderProxy	5K	320	32000	0	3.654	0	3.606	0	3.563	0	4.752	0	4.523	0	4.632
CBRSOAPHeaderProxy	5K	640	64000	0	3.730	0	3.707	0	3.690	0	4.799	0	4.740	0	4.789
CBRSOAPHeaderProxy	5K	1280	256000	0	3.917	0	3.900	0	3.902	0	5.290	0	5.198	0	5.214
CBRSOAPHeaderProxy	5K	2560	512000	0	3.832	0	4.142	0	3.939	0	4.589	0	4.528	0	4.606
CBRSOAPHeaderProxy	10K	20	20000	0	1.598	0	1.681	0	1.640	0	2.898	0	2.890	0	2.882
CBRSOAPHeaderProxy	10K	40	40000	0	3.136	0	3.133	0	3.136	0	3.169	0	3.411	0	3.323
CBRSOAPHeaderProxy	10K	80	8000	0	2.935	0	2.911	0	2.911	0	3.426	0	3.290	0	3.357

CBRSOAPHeaderProxy	10K	160	16000	0	3.000	0	3.018	0	3.023	0	3.842	0	3.606	0	3.680
CBRSOAPHeaderProxy	10K	320	32000	0	3.097	0	3.061	0	3.089	0	3.702	0	3.628	0	3.636
CBRSOAPHeaderProxy	10K	640	6400	0	2.592	0	2.605	0	2.612	0	2.696	0	2.838	0	2.764
Promedio				**0**	**3.618**	**0**	**3.591**	**0**	**3.596**	**0**	**4.336**	**0**	**4.310**	**0**	**4.327**
CBRTransportHeaderProxy	500 b	20	20000	0	2.436	0	2.660	0	2.512	0	4.133	0	4.087	0	4.115
CBRTransportHeaderProxy	500 b	40	40000	0	3.802	0	3.770	0	3.787	0	4.843	0	4.775	0	4.831
CBRTransportHeaderProxy	500 b	80	8000	0	3.501	0	3.568	0	3.520	0	4.750	0	4.756	0	4.741
CBRTransportHeaderProxy	500 b	160	16000	0	3.758	0	3.643	0	3.680	0	5.804	0	5.509	0	5.594
CBRTransportHeaderProxy	500 b	320	32000	0	3.731	0	3.712	0	3.720	0	6.215	0	5.640	0	5.832
CBRTransportHeaderProxy	500 b	640	64000	0	3.872	0	3.796	0	3.842	0	6.106	0	6.194	0	6.044
CBRTransportHeaderProxy	500 b	1280	256000	0	4.095	0	4.106	0	4.095	0	7.101	0	7.007	0	6.996
CBRTransportHeaderProxy	500 b	2560	512000	0	4.241	0	3.997	0	4.074	0	6.417	0	6.251	0	6.579
CBRTransportHeaderProxy	1K	20	20000	0	2.442	0	2.540	0	2.470	0	4.066	0	3.960	0	4.016
CBRTransportHeaderProxy	1K	40	40000	0	3.696	0	3.654	0	3.664	0	4.907	0	4.822	0	4.837
CBRTransportHeaderProxy	1K	80	8000	0	3.420	0	3.396	0	3.434	0	4.841	0	4.763	0	4.799
CBRTransportHeaderProxy	1K	160	15999	0	3.482	0	3.533	0	3.517	0	5.388	0	5.456	0	5.362
CBRTransportHeaderProxy	1K	320	32000	0	3.689	0	3.635	0	3.632	0	5.481	0	5.855	0	5.575
CBRTransportHeaderProxy	1K	640	64000	0	3.646	0	3.758	0	3.683	0	6.300	0	5.875	0	5.950
CBRTransportHeaderProxy	1K	1280	256000	0	3.971	0	3.935	0	3.947	0	6.999	0	6.936	0	6.918
CBRTransportHeaderProxy	1K	2560	512000	0	3.865	0	3.862	0	3.853	0	6.038	0	6.024	0	6.080
CBRTransportHeaderProxy	5K	20	20000	0	1.361	0	1.406	0	1.362	0	3.833	0	3.792	0	3.812
CBRTransportHeaderProxy	5K	40	40000	0	3.142	0	3.111	0	3.124	0	4.606	0	4.538	0	4.554
CBRTransportHeaderProxy	5K	80	8000	0	2.875	0	2.877	0	2.891	0	4.457	0	4.408	0	4.417
CBRTransportHeaderProxy	5K	160	16000	0	2.963	0	2.937	0	2.973	0	4.868	0	4.925	0	4.880
CBRTransportHeaderProxy	5K	320	32000	0	3.119	0	3.112	0	3.091	0	5.143	0	4.842	0	5.002

CBRTransportHeaderProxy	5K	640	64000	0	3.167	0	3.175	0	3.166	0	5.064	0	5.630	0	5.433
CBRTransportHeaderProxy	5K	1280	256000	0	3.299	0	3.275	0	3.284	0	6.280	0	6.214	0	6.190
CBRTransportHeaderProxy	5K	2560	512000	0	3.269	0	3.352	0	3.351	0	5.291	0	5.462	0	5.472
CBRTransportHeaderProxy	10K	20	20000	0	1.243	0	1.251	0	1.254	0	3.414	0	3.387	0	3.407
CBRTransportHeaderProxy	10K	40	40000	0	1.860	0	1.864	0	1.857	0	3.931	0	3.921	0	3.931
CBRTransportHeaderProxy	10K	80	8000	0	2.396	0	2.452	0	2.430	0	3.783	0	3.810	0	3.774
CBRTransportHeaderProxy	10K	160	16000	0	2.563	0	2.468	0	2.514	0	4.019	0	4.050	0	4.065
CBRTransportHeaderProxy	10K	320	32000	0	2.584	0	2.488	0	2.556	0	4.216	0	4.292	0	4.223
CBRTransportHeaderProxy	10K	640	6400	0	2.234	0	2.270	0	2.205	0	3.034	0	3.035	0	3.072
Promedio				**0**	**3.124**	**0**	**3.120**	**0**	**3.116**	**0**	**5.044**	**0**	**5.007**	**0**	**5.017**
XSLTProxy	500 b	20	20000	0	2.263	0	2.316	0	2.339	0	3.076	0	3.068	0	3.061
XSLTProxy	500 b	40	40000	0	3.676	0	3.547	0	3.585	0	3.662	0	3.615	0	3.631
XSLTProxy	500 b	80	8000	0	3.160	0	3.396	0	3.302	0	3.783	0	3.692	0	3.746
XSLTProxy	500 b	160	16000	0	3.537	0	3.617	0	3.561	0	4.167	0	4.144	0	4.129
XSLTProxy	500 b	320	32000	0	3.616	0	3.607	0	3.635	0	4.392	0	4.420	0	4.377
XSLTProxy	500 b	640	64000	0	3.730	0	3.712	0	3.720	0	4.476	0	4.629	0	4.563
XSLTProxy	500 b	1280	256000	0	3.928	0	3.920	0	3.923	0	4.852	0	4.369	0	4.645
XSLTProxy	500 b	2560	512000	0	3.840	0	3.809	0	3.828	0	2.906	0	3.668	0	3.503
XSLTProxy	1K	20	20000	0	1.686	0	1.958	0	1.821	0	2.873	0	2.914	0	2.913
XSLTProxy	1K	40	40000	0	3.435	0	3.390	0	3.417	0	3.497	0	3.504	0	3.506
XSLTProxy	1K	80	8000	0	3.142	0	3.210	0	3.176	0	3.640	0	3.551	0	3.560
XSLTProxy	1K	160	16000	0	3.372	0	3.375	0	3.352	0	3.982	0	4.025	0	3.977
XSLTProxy	1K	320	32000	0	3.422	0	3.300	0	3.376	0	4.218	0	4.164	0	4.159
XSLTProxy	1K	640	6400	0	2.714	0	2.713	0	2.680	0	4.188	0	4.179	0	4.212
XSLTProxy	1K	1280	256000	0	3.601	0	3.615	0	3.599	0	4.030	0	4.468	0	4.289

XSLTProxy	1K	2560	512000	0	3.685	0	3.433	0	3.593	0	2.802	0	2.986	0	2.844
XSLTProxy	5K	20	20000	0	1.249	0	1.238	0	1.236	0	2.173	0	2.183	0	2.193
XSLTProxy	5K	40	40000	0	1.785	0	1.829	0	1.803	0	2.593	0	2.639	0	2.620
XSLTProxy	5K	80	8000	0	2.143	0	2.261	0	2.215	0	2.670	0	2.678	0	2.680
XSLTProxy	5K	160	16000	0	2.204	0	2.325	0	2.276	0	2.847	0	2.930	0	2.884
XSLTProxy	5K	320	32000	0	2.238	0	2.366	0	2.305	0	2.964	0	3.010	0	2.941
XSLTProxy	5K	640	6400	0	2.025	0	1.891	0	2.036	0	2.381	0	2.441	0	2.382
XSLTProxy	5K	1280	256000	0	2.363	0	2.399	0	2.380	0	2.936	0	2.932	0	2.960
XSLTProxy	5K	2560	512000	0	2.333	0	2.378	0	2.356	0	2.350	0	2.692	0	2.337
XSLTProxy	10K	20	20000	0	1.110	0	1.155	0	1.127	0	1.634	0	1.634	0	1.641
XSLTProxy	10K	40	40000	0	1.578	0	1.593	0	1.587	0	1.907	0	1.918	0	1.913
XSLTProxy	10K	80	8000	0	1.591	0	1.618	0	1.601	0	1.955	0	1.966	0	1.960
XSLTProxy	10K	160	16000	0	1.611	0	1.641	0	1.627	0	2.071	0	2.028	0	2.047
XSLTProxy	10K	320	32000	0	1.613	0	1.615	0	1.614	0	2.038	0	2.096	0	2.077
XSLTProxy	10K	640	6400	0	1.500	0	1.517	0	1.509	0	1.721	0	1.471	0	1.651
Promedio				**0**	**2.605**	**0**	**2.625**	**0**	**2.619**	**0**	**3.093**	**0**	**3.134**	**0**	**3.113**
SecureProxy	500 b	20	20000	0	629	0	622	0	629	0	645	0	630	0	640
SecureProxy	500 b	40	40000	0	630	0	649	0	641	0	663	0	661	0	665
SecureProxy	500 b	80	8000	0	648	0	648	0	646	0	678	0	672	0	678
SecureProxy	500 b	160	16000	0	646	0	651	0	649	0	680	0	675	0	680
SecureProxy	500 b	320	32000	0	640	0	647	0	645	0	680	0	672	0	681
SecureProxy	500 b	640	6398	0	628	0	635	0	633	0	572	0	634	0	593
SecureProxy	500 b	1280	12800	0	648	0	644	0	647	0	626	0	560	0	613
SecureProxy	500 b	2560	25600	0	698	0	767	0	707	0	533	0	658	0	634
SecureProxy	1K	20	20000	0	598	0	609	0	604	0	616	0	620	0	618

SecureProxy	1K	40	40000	0	619	0	625	0	623	0	645	0	656	0	650
SecureProxy	1K	80	8000	0	619	0	627	0	624	0	651	0	654	0	651
SecureProxy	1K	160	15998	0	622	0	621	0	622	0	664	0	664	0	664
SecureProxy	1K	320	32000	0	611	0	621	0	617	0	656	0	668	0	660
SecureProxy	1K	640	6400	0	611	0	611	0	611	0	530	0	580	0	564
SecureProxy	1K	1280	12800	0	620	0	610	0	623	0	606	0	619	0	614
SecureProxy	1K	2560	25600	0	678	0	668	0	649	0	474	0	508	0	547
SecureProxy	5K	20	20000	1	531	0	535	0	534	0	384	0	391	0	385
SecureProxy	5K	40	40000	0	533	0	539	0	536	0	394	0	397	0	392
SecureProxy	5K	80	8000	0	523	0	530	0	528	0	390	0	379	0	382
SecureProxy	5K	160	16000	0	507	0	513	0	511	0	397	0	389	0	391
SecureProxy	5K	320	32000	0	510	0	516	0	513	0	371	0	357	0	364
SecureProxy	5K	640	6400	0	511	0	514	0	517	0	307	0	299	0	302
SecureProxy	10K	20	20000	0	445	0	446	0	446	0	203	0	207	0	202
SecureProxy	10K	40	39998	0	438	0	440	0	439	0	211	0	215	0	211
SecureProxy	10K	80	8000	0	435	0	436	0	437	0	203	0	219	0	211
SecureProxy	10K	160	16000	0	415	0	420	0	417	0	204	0	219	0	211
SecureProxy	10K	320	31996	1	412	0	415	0	414	0	174	0	175	0	171
SecureProxy	10K	640	6400	0	421	1	414	0	418	0	156	0	148	0	154
Promedio				**0**	**565**	**0**	**570**	**0**	**567**	**0**	**475**	**0**	**483**	**0**	**483**

Fuente: Propia

Observaciones.

En la tabla 5.5 y en la figura 5.13 se muestran los resultados resumidos de la prueba de rendimiento. En este gráfico se toma el promedio de todos los tamaños de mensajes.

TABLA 5.6: Promedios de las pruebas realizadas a los ESB.

Parámetros de comparación	Mule ESB 3.5	OpenESB 2.3
DirectProxy	3.623	4.490
CBRProxy	3.602	3.703
CBRSOAPHeaderProxy	3.596	4.327
CBRTransportHeaderProxy	3.116	5.017
XSLTProxy	2.619	3.113
SecureProxy	567	483

Fuente: Propia

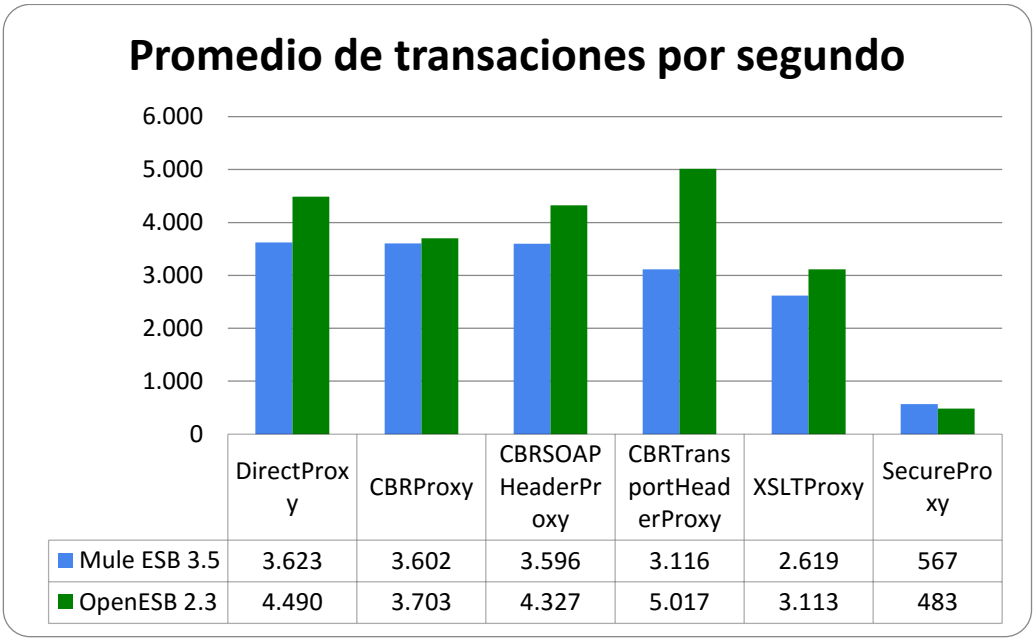


FIGURA 5.75: Análisis de los promedios de los ESB.

Fuente: Propia

Como se muestra en el figura 5.17, OpenESB con una TPS que supera a Mule ESB con la excepción del escenario de SecureProxy en el cual Mule ESB es superior que OpenESB.

5.11 ELECCIÓN DEL BUS DE SERVICIOS EMPRESARIAL ADECUADO.

En el presente trabajo se pretende demostrar por medio del estudio comparativo de los buses de servicios empresariales, en base al grado de cumplimiento de los parámetros que se establecieron anteriormente por el estándar para el uso de ESB en SOA, se ha obtenido la mejor opción.

La opción que resulto mejor fue sometida al estudio de rendimiento y ha resultado favorable con el mayor promedio de transacciones por segundo.

Para consolidar los resultados que se obtuvieron se presenta la siguiente tabla:

TABLA 5.7: Consolidación de los Valores obtenidos durante el estudio.

ESB	Estudio comparativo	Promedio Rendimiento
Mule ESB	85,89%	2,854
OpenESB	93.58%	3,522

Fuente: Propia

Se demuestra entonces que ya una vez realizado el estudio comparativo, en el cual el ESB OpenESB obtiene el mayor porcentaje de cumplimiento de todos los estándares, y posteriormente fue sometido a las pruebas de rendimiento, OpenESB obtiene el mejor rendimiento.

Entonces se llegó a comprobar de una manera favorable el problema planteado, ya que OpenESB es el ESB adecuado que asegurará rendimiento y brindará un menor tiempo de respuesta y un acceso rápido a la información.

CAPITULO VI

6 DISEÑO Y APLICACIÓN DEL PROTOTIPO UTILIZANDO METODOLOGÍA XP.

6.1 METODOLOGÍA EXTREMA XP.

El primer proyecto de la programación extrema XP se inició 06 de marzo 1996. Extreme Programming es una populares metodología de procesos ágiles. Ya que demostrado ser exitosa en una variedad de empresas de todos los tamaños e industrias de todo el mundo. Extreme Programming es exitosa porque hace hincapié en la satisfacción del cliente. En vez de poder entregar todo lo que desee en una fecha lejana en el futuro este proceso ayuda a proporcionar el software que se necesita cuando se lo necesite. Extreme Programming faculta a los desarrolladores para poder responder con mucha confianza a los requerimientos cambiantes de los clientes. Extreme Programming enfatiza en que hay que trabajar en equipo. Los gerentes, los clientes y los desarrolladores son iguales en el equipo de colaboración. Implementa un entorno que es sencillo, pero muy eficaz permitiendo a los equipos a ser muy altamente productiva. El equipo está en la capacidad de auto-organizarse en torno al problema que surge y poder resolverlo lo más eficiente posible. Mejora el proyecto de software en los cinco aspectos esenciales; la comunicación, la simplicidad, la retroalimentación, respeto y coraje. Programadores extremos están en comunicación constante con sus clientes y compañeros programadores. Ellos mantienen su diseño muy simple y limpio. Reciben la retroalimentación probando el software que desarrollan a partir del primer día. Van entregando el sistema a los clientes tan pronto como sea posible e implementan los cambios sugeridos. Cada pequeño éxito profundiza al respeto por la contribución única de cada uno de los miembros del equipo. Con esta base Extrema los programadores son capaces de responder con valentía a las necesidades cambiantes de la tecnología. El aspecto que sorprende de la programación extrema son las reglas simples.

En la página web (Ingeniría de Software U. Union Bolivariana, 2014) nos explica sobre la metodología XP.

6.1.1 ¿QUÉ ES PROGRAMACIÓN EXTREMA O XP?

- ✓ Metodología ágil de desarrollo de software.
- ✓ Se basa en diferentes ideas de cómo enfrentar ambientes cambiantes.
- ✓ Es un conjunto de prácticas y normas empleadas para el desarrollo software.
- ✓ En vez de estar planificando, analizando y diseñando para el futuro distante, hace que todo esto sea poco cada vez, a través del proceso de desarrollo.
- ✓ Originada en el proyecto C3 para Chrysler.

6.1.2 OBJETIVOS.

- ✓ Establece unas prácticas mejoradas de Ingeniería de Software en el desarrollo de proyectos.
- ✓ Garantizar que el Software desarrollando sea de muy alta calidad, haciendo que supere todas las expectativas del cliente.
- ✓ Mejorar toda la productividad de los proyectos.

6.1.3 CONTEXTO XP.

- ✓ El cliente bien definido.
- ✓ Los requisitos pueden cambiar.
- ✓ El grupo debe ser pequeño y muy integrado (máximo 10 personas).
- ✓ Equipo tiene que ser con una formación elevada y con la capacidad de aprender rápidamente.

6.1.4 CARACTERÍSTICAS XP.

- ✓ Fundamentada en los valores y prácticas.

- ✓ Metodología basada en prueba y error.
- ✓ Expresada en la forma de 12 Prácticas.
- ✓ Se soportan unas a otras.
- ✓ Conjunto completo.
- ✓ Son conocidas desde hace tiempo.

6.1.5 VALORES XP.

- ✓ La Simplicidad XP nos propone al principio hacer las cosas lo más simple que se pueda.
- ✓ Comunicación los problemas pueden cuando alguien no dijo algo importante en el momento del desarrollo, XP hace no haya falta de comunicación.
- ✓ Retroalimentación eficaz y frecuente del cliente, de todo el equipo y de los usuarios finales da una oportunidad de dirigir eficientemente el esfuerzo.
- ✓ El coraje (valor) existe en el contexto de que si función mejóralo.

El estilo XP.

- ✓ Está orientada hacia los que producen y utilizan el software.
- ✓ Combina las mejores prácticas para desarrollar software, y las lleva al extremo.
- ✓ Reduce el costo del cambio en las etapas del ciclo de vida del sistema.

6.2 ANÁLISIS Y DISEÑO DE LA APLICACIÓN DE FACTURACIÓN ELECTRÓNICA EN FUNCIÓN DE LA METODOLOGÍA XP.

Al evaluar las diferentes alternativas que existen en lenguajes de programación y/o plataformas, la aplicación se desarrolló con el lenguaje de programación Java, dado las características y la sencillez que este lenguaje provee para el trabajo y la gestión en la bases de datos, el cual es el núcleo central de la aplicación.

6.3 FASE 1: EXPLORACIÓN.

En esta fase el cliente plantea las historias de usuario las cuales son de su interés para la primera entrega del producto.

6.3.1 HISTORIAS DE USUARIO.

Las historias de usuario son las especificaciones y los requisitos del software, donde se describen brevemente las características que el sistema de facturación debe tener desde la perspectiva y necesidades del cliente.

R1.- Control y autenticación de usuarios.

TABLA 6.8: Historia de usuario: Control y autenticación de usuarios.

Historia de usuario	
Número: 1	**Usuario:** Vendedor
Nombre historia: Control y Autenticación de usuarios	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Alto
Puntos estimados: 1	**Iteración asignada: 1**
Programador responsable: Fabricio Huera	
Descripción: Al iniciar la aplicación se va a solicitar el nombre de usuario y su respectiva clave para que pueda tener acceso al sistema de facturación electrónica.	
Observaciones: En caso de no haber ningún error al momento de autenticarse la aplicación ingresara a la página de inicio, en caso de que exista un error no se ingresa y pedirá volver a ingresar.	

Fuente: Propia

R2.- Generación de factura.

TABLA 6.9: Generación de venta.

Historia de usuario	
Número: 2	**Usuario:** Vendedor
Nombre historia: Generar la Factura	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 3.5	**Iteración asignada:** 2
Programador responsable: Fabricio Huera	
Descripción: La aplicación dispondrá de la opción de generar una factura la cual nos pedirá ingresar productos que el cliente(comprador) desea adquirir	
Observaciones: En el caso de que el cliente (comprador) no desee adquirir un producto en el proceso de venta, se lo podrá eliminar	

Fuente: Propia

R3.- Generar búsqueda de productos.

TABLA 6.10: Generar Búsqueda de productos.

Historia de usuario	
Número: 3	**Usuario:** Vendedor
Nombre historia: Generar Búsqueda de Productos	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Medio
Puntos estimados: 3	**Iteración asignada:** 2
Programador responsable: Fabricio Huera	
Descripción: El vendedor seleccionará la opción del menú “Productos”, verá el listado de los artículos, tras seleccionar uno. En cada producto seleccionado se ingresará la cantidad que requiera el cliente cuyos datos serán ingresados por el vendedor.	
Observaciones: Una vez seleccionado dicho producto ya no se podrá seleccionar el mismo producto otra vez	

Fuente: Propia

R4.- Generar búsqueda del cliente.

TABLA 6.11: Generar Búsqueda del cliente.

Historia de usuario	
Número: 4	**Usuario:** Vendedor
Nombre historia: Generar Búsqueda del Cliente	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Medio
Puntos estimados: 3	**Iteración asignada:** 2
Programador responsable: Fabricio Huera	
Descripción: El vendedor seleccionará la opción del menú “Cliente”, y podrá buscar por el número de cedula o por el nombre del cliente.	
Observaciones:	

Fuente: Propia

R5.- Generación de facturas electrónicas firmadas electrónicamente.

TABLA 6.12: Generación de facturas electrónicas firmadas electrónicamente.

Historia de usuario	
Número: 5	**Usuario:** Vendedor
Nombre historia: Generación de facturas electrónicas firmadas electrónicamente	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 3	**Iteración asignada:** 3
Programador responsable: Fabricio Huera	
Descripción: Se generan las facturas electrónicas que serán firmadas electrónicamente como lo especifica el SRI.	
Observaciones: Se guardara un archivo xml con la firma electrónica	

Fuente: Propia

R6.- Generación de facturas electrónicas enviadas y autorizadas por SRI.

TABLA 6.13: Generación de facturas electrónicas enviadas y autorizadas por SRI.

Historia de usuario	
Número: 6	**Usuario:** Vendedor
Nombre historia: Generación de facturas electrónicas enviadas y autorizadas por SRI	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 3	**Iteración asignada:** 3
Programador responsable: Fabricio Huera	
Descripción: Se generan las facturas electrónicas que serán enviadas al SRI el cual las autorizar.	
Observaciones: Se generara un archivo xml donde tendrá la respuesta del SRI.	

Fuente: Propia

R7.- Envío de las facturas electrónicas por mail.

TABLA 6.14: Envío de las facturas electrónicas por mail.

Historia de usuario	
Número: 7	**Usuario:** Vendedor
Nombre historia: Envío de las facturas electrónicas por mail	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Bajo
Puntos estimados: 1	**Iteración asignada:** 3
Programador responsable: Fabricio Huera	
Descripción: Se podrá hacer llegar a los respectivos correos electrónicos de los clientes las facturas electrónicas para el ahorro de costos de distribución física de la factura.	
Observaciones: El cliente podrá observar sus facturas en su correo electrónico.	

Fuente: Propia

R8.- Emisión de las facturas en modo contingencia.

TABLA 6.15: Emisión de las facturas en contingencia

Historia de usuario	
Número: 8	**Usuario:** Vendedor
Nombre historia: Emisión de las facturas en modo contingencia	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 2	**Iteración asignada:** 3
Programador responsable: Fabricio Huera	
Descripción: Como Vendedor se quiere poder emitir las factura electrónicas en situaciones de contingencias, cuando los sistemas del SRI no estén disponibles.	
Observaciones:	

Fuente: Propia

R9.- Consulta del estado de la factura y reenvío.

TABLA 6.16: Consulta del estado de la factura y reenvío.

Historia de usuario	
Número: 9	**Usuario:** Vendedor
Nombre historia: Consulta del estado de la factura y reenvío	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable: Fabricio Huera	
Descripción: Como Vendedor se quiere poder consultar el estado de la factura, y reenviarlo para aquellas facturas que lo ameriten	
Observaciones:	

Fuente: Propia

R10.- Gestión de clientes.

TABLA 6.17: Gestión de clientes.

Historia de usuario	
Número: 10	**Usuario:** Vendedor
Nombre historia: Gestión de clientes	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Bajo
Puntos estimados: 3.5	**Iteración asignada:** 2
Programador responsable: Fabricio Huera	
Descripción: La aplicación permitirá al Vendedor ingresar, editar los datos que están relacionados con los clientes.	
Observaciones:	

Fuente: Propia

R11.- Gestión de productos.

TABLA 6.18: Gestión de productos.

Historia de usuario	
Número: 11	**Usuario:** Vendedor
Nombre historia: Gestión de Productos	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Puntos estimados: 4	**Iteración asignada:** 1
Programador responsable: Fabricio Huera	
Descripción: La aplicación va a permitir administrar los productos los cuales se podrá ingresar, editar y eliminar.	
Observaciones: Al realizar cualquiera de las opciones en la base de datos se actualizará	

Fuente: Propia

R12.- Gestión de emisor.

TABLA 6.19: Generación de facturas electrónicas.

Historia de usuario	
Número: 12	**Usuario:** Vendedor
Nombre historia: Gestión del Emisor	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Alto
Puntos estimados: 3	**Iteración asignada:** 1
Programador responsable: Fabricio Huera	
Descripción: La aplicación va a permitir administrar el emisor el cual se podrá ingresar, editar.	
Observaciones: Al realizar cualquiera de las opciones en la base de datos se actualizará	

Fuente: Propia

6.3.2 PROTOTIPOS; INTERFACES DE LA APLICACIÓN BASADO EN LO QUE EL CLIENTE DESEA.

Pantalla: Autenticación de usuarios.

FIGURA 6.76: Pantalla: Autenticación de usuarios.

Fuente: Propia

Pantalla: Generar factura y búsqueda de productos.

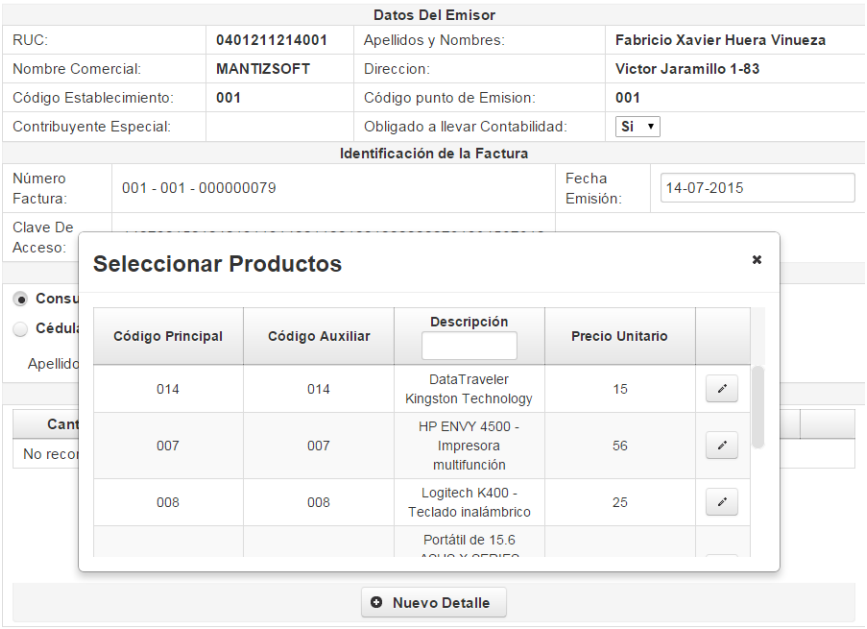


FIGURA 6.77: Pantalla: Generar factura y búsqueda de productos.

Fuente: Propia

Pantalla: Generar factura.

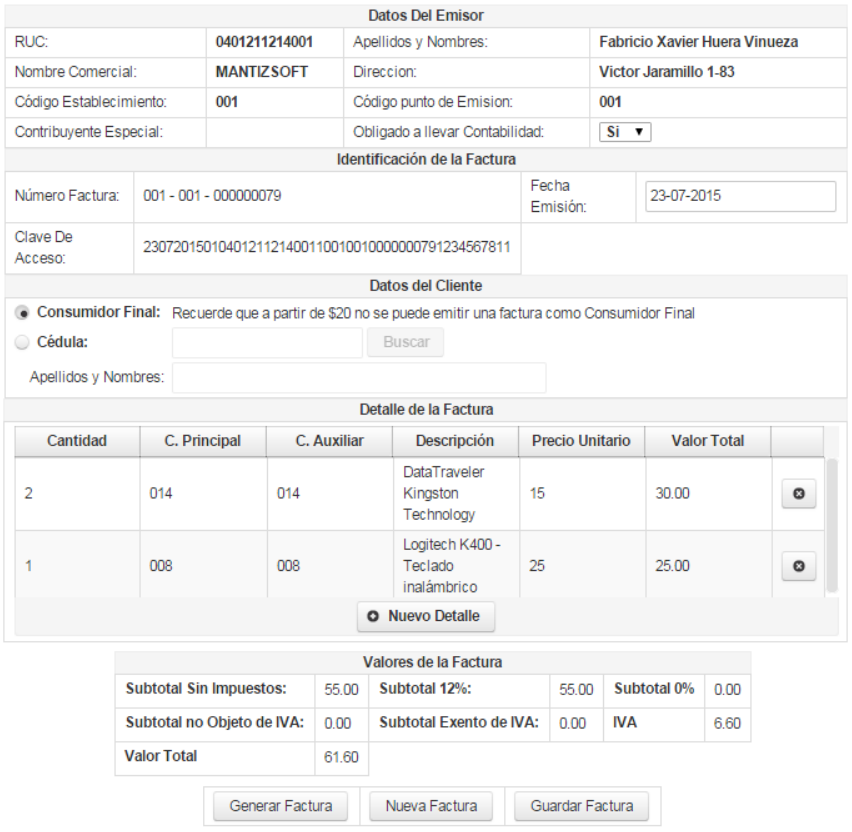


FIGURA 6.78: Pantalla: Generar factura.

Fuente: Propia

Pantalla: Gestión de productos.

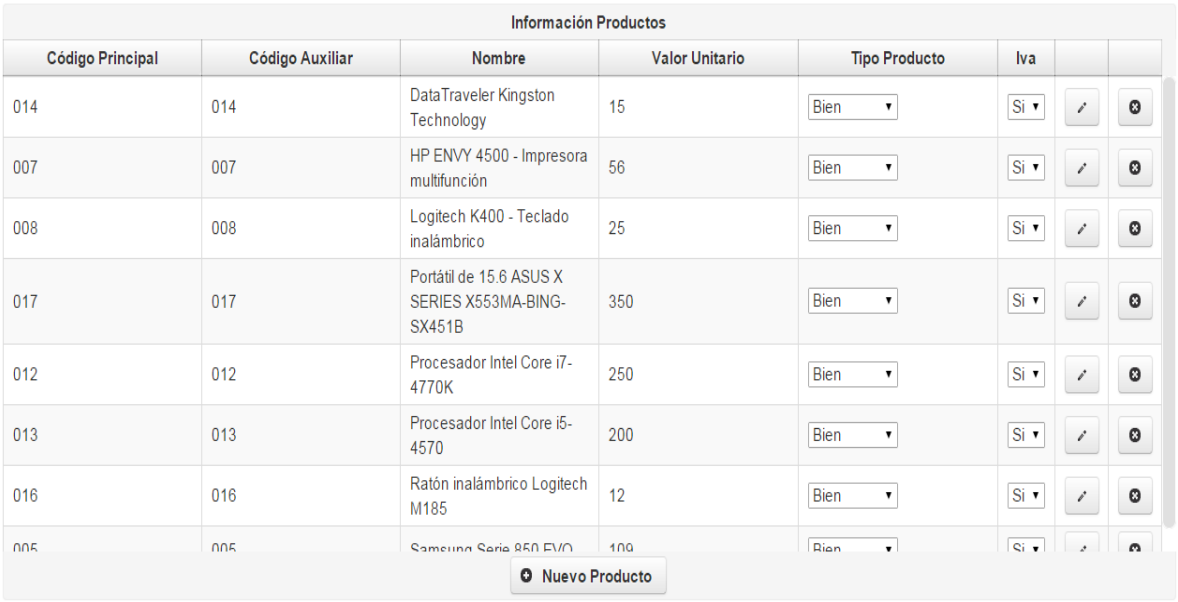


FIGURA 6.79: Pantalla: Gestión de productos.

Fuente: Propia

Pantalla: Ingreso de productos.

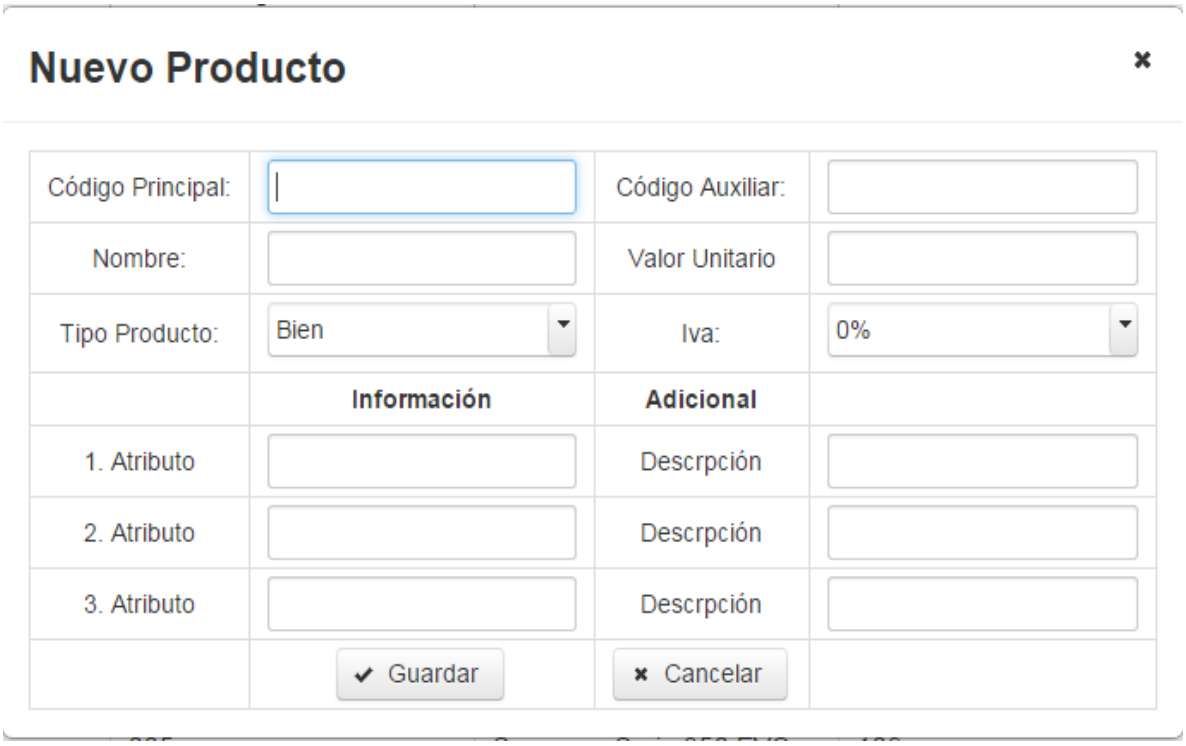


FIGURA 6.80: Pantalla: Ingreso de productos.

Fuente: Propia

Pantalla: Actualizar producto.

FIGURA 6.81: Pantalla: Actualizar producto.

Fuente: Propia

Pantalla: Gestión clientes.

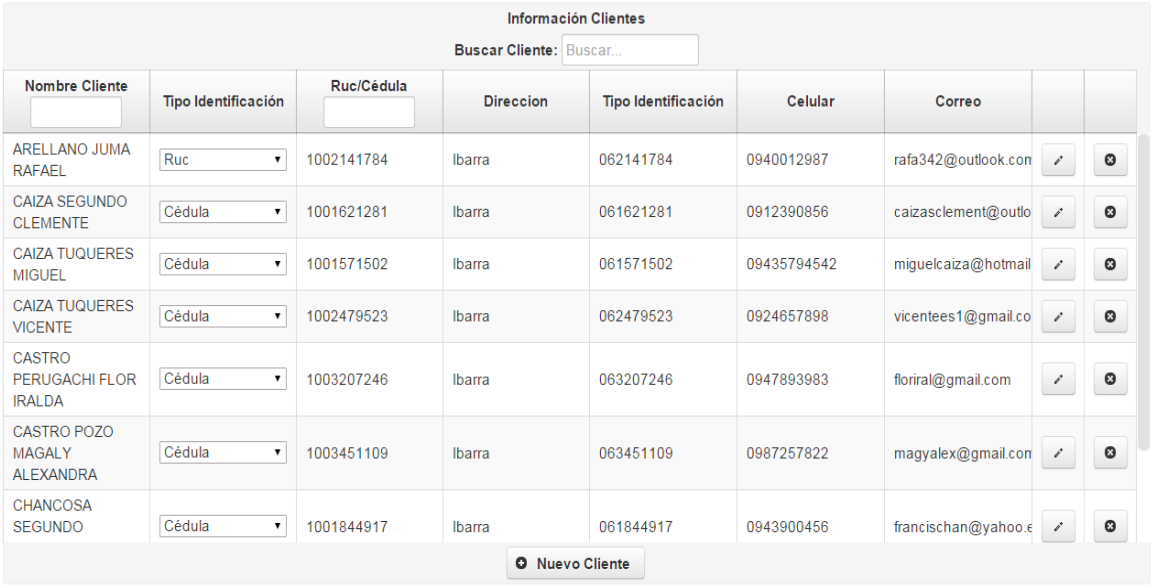


FIGURA 6.82: Pantalla: Gestión clientes.

Fuente: Propia

Pantalla: Ingreso clientes.

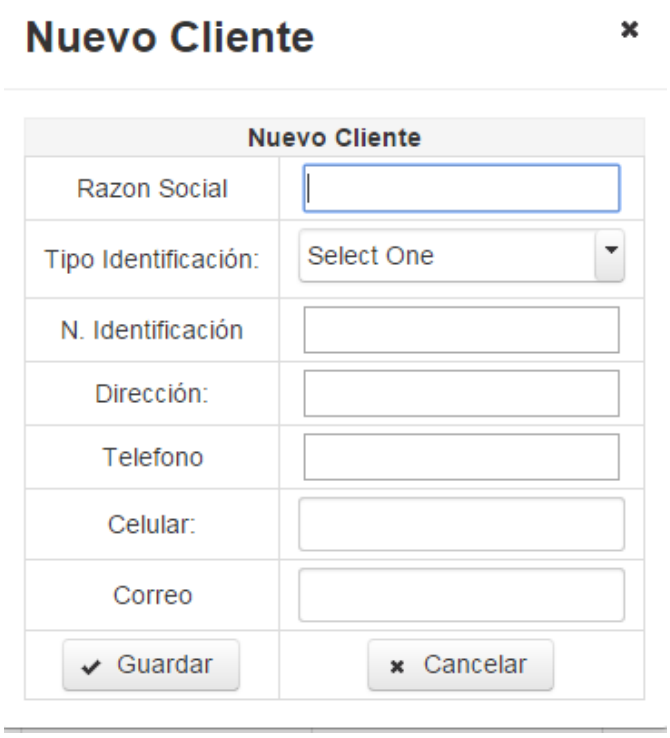


FIGURA 6.83: Pantalla: Ingreso clientes.

Fuente: Propia

Pantalla: Actualizar clientes.

FIGURA 6.84: Pantalla: Actualizar clientes.

Fuente: Propia

Pantalla: Gestión emisor.

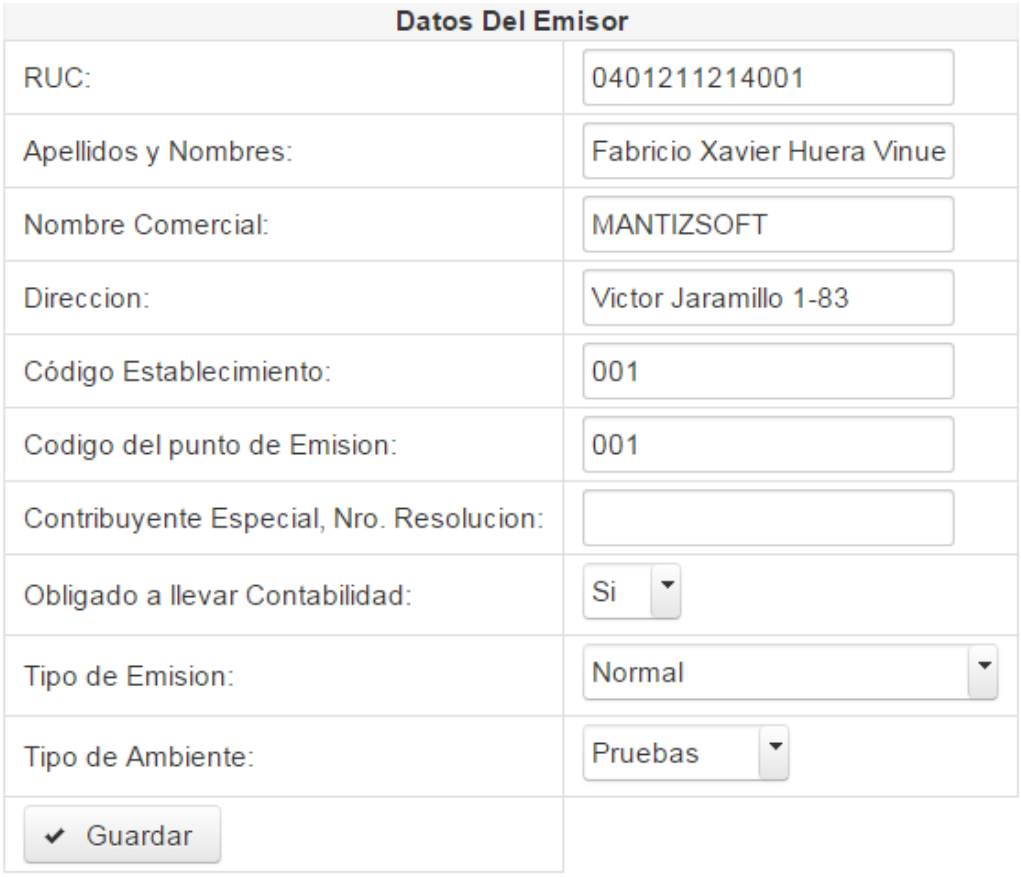


FIGURA 6.85: Pantalla: Gestión emisor.

Fuente: Propia

Pantalla: Reporte de facturas.

Reporte de Facturas

Nro. Factura	Cliente	Fecha	Clave de Acceso	Estado Factura	Total Factura
001-001-000000002	CONSUMIDOR FINAL	2015-01-30	3001201501040121121400110010010000000021234567811	TRANSMITIDO SIN RESPUESTA	10.00
001-001-000000004	CONSUMIDOR FINAL	2015-01-30	3001201501040121121400110010010000000041234567812	AUTORIZADO	10.00
001-001-000000006	Fabricio Xavier Huera Vinueza	2015-02-03	3002201501040121121400110010010000000061234567812	AUTORIZADO	13.00
001-001-000000007	Fabricio Xavier Huera Vinueza	2015-02-03	3002201501040121121400110010010000000071234567818	AUTORIZADO	10.00
001-001-000000008	Fabricio Xavier Huera Vinueza	2015-02-03	3002201501040121121400110010010000000081234567813	AUTORIZADO	62.40
001-001-000000009	Fabricio Xavier Huera Vinueza	2015-02-03	3002201501040121121400110010010000000091234567819	AUTORIZADO	15.00
001-001-000000010	Fabricio Xavier Huera Vinueza	2015-02-03	3002201501040121121400110010010000000101234567814	AUTORIZADO	50.40

F**IGURA 6.86:** Pantalla: Reporte de facturas.

Fuente: Propia

6.4 FASE 2: PLANIFICACIÓN.

Para poder elaborar la planificación, es necesario poder identificar las historias de usuarios y establecer la prioridad de cada historia, realizando y analizando una estimación del esfuerzo necesario.

6.4.1 VALORACIÓN DE HISTORIAS DE USUARIO.

Como el punto importante para la planificación de entregables, se tiene que considerar la estimación de historias de usuario, donde se especifica un tiempo de estimación para elaborar cada una de las historias con la base de 4 horas diarias.

Estimación de historias de usuario.

TABLA 6.20: Estimación tiempo en base a las historias de usuarios.

NÚMERO	HISTORIA DE USUARIO	TIEMPO ESTIMADO	
		DÍAS	HORAS
1	Control y Autenticación de usuario	4	16
2	Generar Factura	8	24
3	Generar Búsqueda de Productos	3	12
4	Generar Búsqueda del Cliente	3	12
5	Generación de facturas electrónicas firmadas electrónicamente	10	40
6	Generación de facturas electrónicas enviadas y autorizadas por SRI	10	40
7	Envío de las facturas electrónicas por mail	4	16
8	Emisión de las facturas en contingencia	10	40
9	Consulta del estado de la factura y reenvío	6	24
10	Gestión de clientes	3	12
11	Gestión de Productos	4	16
12	Gestión del Emisor	4	16

Fuente: Propia

6.4.2 PLAN DE ENTREGAS.

Para elaborar el plan de entrega del proyecto, se aplican los parámetros de la metodología que determinan un plan, mediante la estimación de fases de la metodología XP.

Plan de entrega.

TABLA 6.21: Plan de entregables.

Fases, Pasos	Hitos.	Inicio.	Fin.
Fase 1. Exploración	• Historias de Usuario	27/03/2015	19/04/2015
Fase 2. Planificación	• Plan de entrega	20/04/2015	20/05/2015
Fase 3. Plan de Iteraciones	• Plan de interacción	21/05/2015	08/08/2015
Fase 4. Producción	• Primera versión del sistema	09/08/2015	31/08/2015
Fase 5. Cierre de Proyecto	• Manual de usuario para administración del sitio • Versión final del proyecto • Entrega del proyecto	01/08/2015	05/08/2015

Fuente: Propia

6.4.3 ROLES DE USUARIO.

Se especifica lo roles que tiene cada uno de los integrantes del equipo que pertenecen a desarrollo de la aplicación, tomando en cuenta que el cliente es el más importante y desempeña un papel crucial en este equipo.

TABLA 6.22: Roles que se desempeña en el proyecto.

Roles	Encargado
Programador:	Fabricio Huera
Cliente:	Mantisoft
Tester:	Fabricio Huera
Programador/Analista:	Fabricio Huera
Diseñador de interfaz:	Fabricio Huera
Administrador de BDD:	Fabricio Huera

Fuente: Propia

6.5 FASE 3: PLAN DE INTERACCIONES.

En la fase de interacciones se presentan y se describen las historias que se llevaron a cabo en la iteración final, esta fase incluye las pruebas funcionales, la planificación del proyecto y todas las incidencias que ha tenido el proyecto. Finalmente se describe toda la evolución que se ha producido en el equipo que desarrollo el proyecto.

6.5.1 HISTORIAL DE VERSIÓN POR HISTORIA DE USUARIO.

TABLA 6.23: Cuadro Entregables: Versión por historia de usuario.

ITERACIÓN	Nº	HISTORIA DE USUARIO	PRIORIDAD ENTREGA	ACTIVIDAD	DEPENDENCIA	RIESGO	VERSION	ESTADO DE DESARROLLO
Primera	1	Control y Autenticación de usuario	1	Nueva	NA	Alto	1	Finalizado
Segunda	2	Generar de Factura	2	Nueva	1,3,4	Alto	1	Finalizado
Segunda	3	Generar Búsqueda de Productos	2	Nueva	1,11	Medio	1	Finalizado
Segunda	4	Generar Búsqueda del Cliente	2	Nueva	1,10	Medio	1	Finalizado
Tercera	5	Generación de facturas electrónicas firmadas electrónicamente	3	Nueva	1,2	Alto	1	Finalizado
Tercera	6	Generación de facturas electrónicas enviadas y autorizadas por SRI	3	Nueva	1,2,5	Alto	1	Finalizado
Tercera	7	Envío de las facturas electrónicas por mail	3	Nueva	1,2,5,6	Bajo	1	Finalizado
Tercera	8	Emisión de las facturas en contingencia	3	Nueva	1,2,5	Alto	1	Finalizado
Primera	9	Consulta del estado de la factura y reenvío	1	Nueva	1	Medio	1	Finalizado
Segunda	10	Gestión de clientes	2	Nueva	1	Bajo	1	Finalizado
Primera	11	Gestión de Productos	1	Nueva	1	Medio	1	Finalizado
Primera	12	Gestión del Emisor	1	Nueva	1	Alto	1	Finalizado

Fuente: Propia

6.5.2 SEGUIMIENTO DEL HISTORIAL DE LAS ITERACIONES.

TABLA 6.24: Cuadro Entregables: Seguimiento por iteraciones.

ITERACIÓN	Nº	HISTORIA DE USUARIO	FECHA PLANIFICACIÓN ITERACIONES		LANZAMIENTO	ESTADO DE DESARROLLO
Primera	1	Control y autenticación de usuario	09/03/2015	13/03/2015	02/06/2015	Finalizado
	9	Consulta del estado de la factura y reenvío	14/03/2015	20/03/2015	02/06/2016	Finalizado
	11	Gestión de productos	21/03/2015	25/03/2015	02/06/2017	Finalizado
	12	Gestión del emisor	26/03/2015	30/03/2015	02/06/2018	Finalizado
Segunda	2	Generar de factura	31/04/2015	08/04/2015	02/06/2019	Finalizado
	3	Generar búsqueda de productos	09/04/2015	12/04/2015	02/06/2020	Finalizado
	4	Generar búsqueda del cliente	13/04/2015	16/04/2015	02/06/2021	Finalizado
	10	Gestión de clientes	17/04/2015	20/04/2015	02/06/2022	Finalizado
Tercera	5	Generación de facturas electrónicas firmadas electrónicamente	21/04/2015	01/05/2015	02/06/2023	Finalizado
	6	Generación de facturas electrónicas enviadas y autorizadas por SRI	02/05/2015	12/05/2015	02/06/2024	Finalizado
	7	Envío de las facturas electrónicas por mail	13/05/2015	17/05/2016	02/06/2025	Finalizado
	8	Emisión de las facturas en contingencia	18/05/2015	28/05/2015	02/06/2026	Finalizado

Fuente: Propia

6.6 FASE 4: PRODUCCIÓN.

En esta fase se observa la finalización, las pruebas adicionales y revisar el rendimiento del sistema, para que este sistema pueda ser trasladado al entorno del cliente y sea implementado.

6.6.1 SEGUIMIENTO ITERACIONES.

Para el seguimiento es muy importante la comunicación entre el cliente y los desarrolladores, la finalidad es enfocarse en encontrar y establecer problemas y posibles soluciones de las tareas de desarrollo de la aplicación.

6.6.2 REPORTE POR ITERACIONES.

El objetivo principal es controlar las tareas asignadas a cada iteración, aquí se podrá visualizar todo el desarrollo del proyecto.

En Base a:

El Historia de seguimiento de las tareas activas.

TABLA 6.25: Historial Seguimiento de las tareas.

N⁰	Historia de usuario	Tarea	Estado de desarrollo	Responsable	Esfuerzo Estimado	Esfuerzo Real Invertido(días)	Esfuerzo por Realizar (días)
1	Control y autenticación de usuario	Especificación de pruebas	Completo	Fabricio Huera	0,5	1	0
		Monitoreo de herramientas XP	Completo	Fabricio Huera	0,5	4	0
		Diseño de interfaz	Completo	Fabricio Huera	0,5	3	0
		Diseño CRC	Completo	Fabricio Huera	0,5	2	0
		Diagrama de base de datos	Completo	Fabricio Huera	0,5	1	0
		Programa de interfaz	Completo	Fabricio Huera	0,5	3	0
		Pruebas de aceptación	Completo	Fabricio Huera	1	2	0
		Esfuerzo total			**4**	**16**	**0**
2	Generar factura	Especificación de pruebas	Completo	Fabricio Huera	0,5	1	0
		Monitoreo de herramientas XP	Completo	Fabricio Huera	2	5	0
		Diseño de interfaz	Completo	Fabricio Huera	1	4	0
		Diseño CRC	Completo	Fabricio Huera	0,5	1	0
		Diagrama de base de datos	Completo	Fabricio Huera	2	6	0
		Programa de interfaz	Completo	Fabricio Huera	1	4	0
		Pruebas de aceptación	Completo	Fabricio Huera	1	2	0

		Esfuerzo total			**8**	**23**	**0**
3	**Generar búsqueda de productos**	Especificación de pruebas	Completo	Fabricio Huera	0,5	1	0
		Monitoreo de herramientas XP	Completo	Fabricio Huera	0,5	2	0
		Diseño de interfaz	Completo	Fabricio Huera	0,5	1	0
		Diseño CRC	Completo	Fabricio Huera	0,5	2	0
		Diagrama de base de datos	Completo	Fabricio Huera	0,5	2	0
		Programa de interfaz	Completo	Fabricio Huera	0,5	1	0
		Pruebas de aceptación	Completo	Fabricio Huera	0,5	3	0
		Esfuerzo Total			**3**	**12**	**0**
4	**Generar Búsqueda del Cliente**	Especificación de pruebas	Completo	Fabricio Huera	0,5	1	0
		Monitoreo de herramientas XP	Completo	Fabricio Huera	0,5	2	0
		Diseño de interfaz	Completo	Fabricio Huera	0,5	1	0
		Diseño CRC	Completo	Fabricio Huera	0,5	3	0
		Diagrama de base de datos	Completo	Fabricio Huera	0,5	2	0
		Programa de interfaz	Completo	Fabricio Huera	0,5	1	0
		Pruebas de aceptación	Completo	Fabricio Huera	0,5	2	0
		Esfuerzo total			**4**	**12**	**0**
5	**Generación de facturas electrónicas firmadas electrónicamente**	Especificación de pruebas	Completo	Fabricio Huera	0,5	1	0
		Monitoreo de herramientas XP	Completo	Fabricio Huera	1	6	0
		Diseño de Interfaz	Completo	Fabricio Huera	3	8	0
		Diseño CRC	Completo	Fabricio Huera	1	2	0
		Diagrama de base de datos	Completo	Fabricio Huera	2	4	0
		Programa de interfaz	Completo	Fabricio Huera	2	3	0
		Pruebas de aceptación	Completo	Fabricio Huera	1	2	0
		Esfuerzo Total			**10**	**26**	**0**
	Generación de facturas electrónicas	Especificación de pruebas	Completo	Fabricio Huera	0,5	1	0
		Monitoreo de herramientas XP	Completo	Fabricio Huera	1	6	0
		Diseño de Interfaz	Completo	Fabricio Huera	2	8	0
		Diseño CRC	Completo	Fabricio Huera	1	2	0

6	**enviadas y autorizadas por SRI**	Diagrama de base de datos	Completo	Fabricio Huera	3	4	0
		Programa de interfaz	Completo	Fabricio Huera	2	3	0
		Pruebas de aceptación	Completo	Fabricio Huera	1	2	0
		Esfuerzo total			**10**	**26**	**0**
7	**Envío de las facturas electrónicas por mail**	Especificación de pruebas	Completo	Fabricio Huera	0,5	1	0
		Monitoreo de herramientas XP	Completo	Fabricio Huera	0,5	4	0
		Diseño de interfaz	Completo	Fabricio Huera	0,5	3	0
		Diseño CRC	Completo	Fabricio Huera	0,5	2	0
		Diagrama de base de datos	Completo	Fabricio Huera	0,5	1	0
		Programa de interfaz	Completo	Fabricio Huera	0,5	3	0
		Pruebas de aceptación	Completo	Fabricio Huera	1	2	0
		Esfuerzo total			**4**	**16**	**0**
8	**Emisión de las facturas en contingencia**	Especificación de pruebas	Completo	Fabricio Huera	0,5	1	0
		Monitoreo de herramientas XP	Completo	Fabricio Huera	1	7	0
		Diseño de interfaz	Completo	Fabricio Huera	2	5	0
		Diseño CRC	Completo	Fabricio Huera	1	3	0
		Diagrama de base de datos	Completo	Fabricio Huera	3	2	0
		Programa de interfaz	Completo	Fabricio Huera	2	6	0
		Pruebas de aceptación	Completo	Fabricio Huera	1	2	0
		Esfuerzo total			**10**	**26**	**0**
9	**Consulta del estado de la factura y reenvío**	Especificación de pruebas	Completo	Fabricio Huera	0,5	1	0
		Monitoreo de herramientas XP	Completo	Fabricio Huera	1	2	0
		Diseño de interfaz	Completo	Fabricio Huera	1	3	0
		Diseño CRC	Completo	Fabricio Huera	1	2	0
		Diagrama de base de datos	Completo	Fabricio Huera	1	1	0
		Programa de interfaz	Completo	Fabricio Huera	0,5	3	0
		Pruebas de aceptación	Completo	Fabricio Huera	1	2	0
		Esfuerzo Total			**6**	**14**	**0**
		Especificación de pruebas	Completo	Fabricio Huera	0,5	1	0

10	**Gestión de clientes**	Monitoreo de herramientas XP	Completo	Fabricio Huera	0,5	2	0
		Diseño de interfaz	Completo	Fabricio Huera	0,5	3	0
		Diseño CRC	Completo	Fabricio Huera	0,5	1	0
		Diagrama de base de datos	Completo	Fabricio Huera	0,5	1	0
		Programa de interfaz	Completo	Fabricio Huera	0,5	2	0
		Pruebas de aceptación	Completo	Fabricio Huera	0,5	2	0
		Esfuerzo Total			**3**	**12**	**0**
11	**Gestión de Productos**	Especificación de pruebas	Completo	Fabricio Huera	0,5	1	0
		Monitoreo de herramientas XP	Completo	Fabricio Huera	0,5	1	0
		Diseño de interfaz	Completo	Fabricio Huera	0,5	3	0
		Diseño CRC	Completo	Fabricio Huera	0,5	1	0
		Diagrama de base de datos	Completo	Fabricio Huera	0,5	1	0
		Programa de interfaz	Completo	Fabricio Huera	1	1	0
		Pruebas de aceptación	Completo	Fabricio Huera	0,5	2	0
		Esfuerzo total			**4**	**10**	**0**
12	**Gestión del Emisor**	Especificación de pruebas	Completo	Fabricio Huera	0,5	1	0
		Monitoreo de herramientas XP	Completo	Fabricio Huera	0,5	2	0
		Diseño de interfaz	Completo	Fabricio Huera	0,5	3	0
		Diseño CRC	Completo	Fabricio Huera	1	1	0
		Diagrama de base de datos	Completo	Fabricio Huera	0,5	1	0
		Programa de interfaz	Completo	Fabricio Huera	0,5	1	0
		Pruebas de aceptación	Completo	Fabricio Huera	0,5	1	0
		Esfuerzo total			**3**	**8**	**0**

Fuente: Propia

6.6.3 EJECUCIÓN DE ITERACIONES.

Visualiza la forma de la implementación de cada una de las historias de usuario en base a las tarjetas CRC (Responsables y Colaboración de las clases) y la especificación de los escenarios.

6.6.3.1 ESPECIFICACIÓN ESCENARIOS EN BASE A LAS HISTORIAS DE USUARIO.

Escenario N. 1: AUTENTICACIÓN DE USUARIOS.

Propósito escenario.

- Autenticar a todos los usuarios del sistema.

Tarjeta CRC: Autenticación.

TABLA 6.26: Tarjeta CRC: Autenticación

TARJETA CRC		
Número: 01	**Escenario:** Autenticación de usuarios	
Nombre CRC: Autenticación		
Responsabilidad	**Colaboradores**	**Métodos**
Autenticar usuario		LoginBean
Observaciones: Controla el ingreso de los usuario registrados en la aplicación		

Fuente: Propia

Escenario N. 2: GESTIÓN DE PRODUCTOS.

Propósito escenario.

- Administración de los productos que existen en la empresa por parte del administrador de la aplicación.
- Búsqueda por parte del vendedor los productos.

Tarjeta CRC: Productos.

TABLA 6.27: Tarjeta CRC: Producto.

TARJETA CRC		
Número: 02	**Escenario:** Gestión de productos	
Nombre CRC: Producto		
Responsabilidad	**Colaboradores**	**Métodos**
Ingreso Productos Edición Productos Buscar Productos		saveProducto saveEditProducto findProducto
Observaciones: Todos los productos serán ingresados, editados por el administrador. La Búsqueda de productos realizara el vendedor del sistema		

Fuente: Propia

Escenario N. 3: GESTIÓN DE CLIENTES.

Propósito Escenario.

- Administración de los clientes que existen en la empresa por parte del administrador de la aplicación.
- Búsqueda de clientes por parte del vendedor.

Tarjeta CRC: Clientes.

TABLA 6.28: Tarjeta CRC: Clientes.

TARJETA CRC		
Número: 03	**Escenario:** Gestión de clientes	
Nombre CRC: Clientes		
Responsabilidad	**Colaboradores**	**Métodos**
Ingreso Clientes Edición Clientes Buscar Clientes		saveCliente saveEditCliente buscarCliente
Observaciones: Todos los clientes serán ingresados, editados por el administrador. La Búsqueda de clientes realizara el vendedor del sistema		

Fuente: Propia

Escenario N. 4: GESTIÓN DEL EMISOR.

Propósito escenario.

- Administración del emisor que existen en la empresa por parte del administrador de la aplicación.

Tarjeta CRC: Emisor.

TABLA 6.29: Tarjeta CRC: Emisor.

TARJETA CRC		
Número: 04	**Escenario:** Gestión del emisor	
Nombre CRC: Emisor		
Responsabilidad	**Colaboradores**	**Métodos**
Ingreso Emisor Edición Emisor		saveEmisor saveEditEmisor
Observaciones: El emisor será ingresado, editado por el administrador		

Fuente: Propia

Escenario N. 5: REGISTRO DE FACTURAS.

Propósito escenario.

- Generar la factura.
- Validar por el SRI la factura.

Tarjeta CRC: Facturas.

TABLA 6.30: Tarjeta CRC: Facturas.

TARJETA CRC		
Número: 05	**Escenario:** Registro de facturas	
Nombre CRC: Facturas		
Responsabilidad	**Colaboradores**	**Métodos**
Nueva Factura Buscar Factura Firmar Factura Enviar Factura al SRI Enviar Factura al Cliente Guardar Factura		nuevaFactura listaFactura firmarProcesarFactura RecepcionFacturaService envioMensaje emisionNormal
Observaciones: Las facturas serán generadas, enviadas al SRI para su aprobación, enviadas al cliente y se guardaran en la BDD		

Fuente: Propia

6.6.4 DIAGRAMA DE ENTIDADES.

Se utiliza para poder visualizar las relaciones que existen entre las clases que están involucradas en la aplicación.

DIAGRAMA DE CLASES.

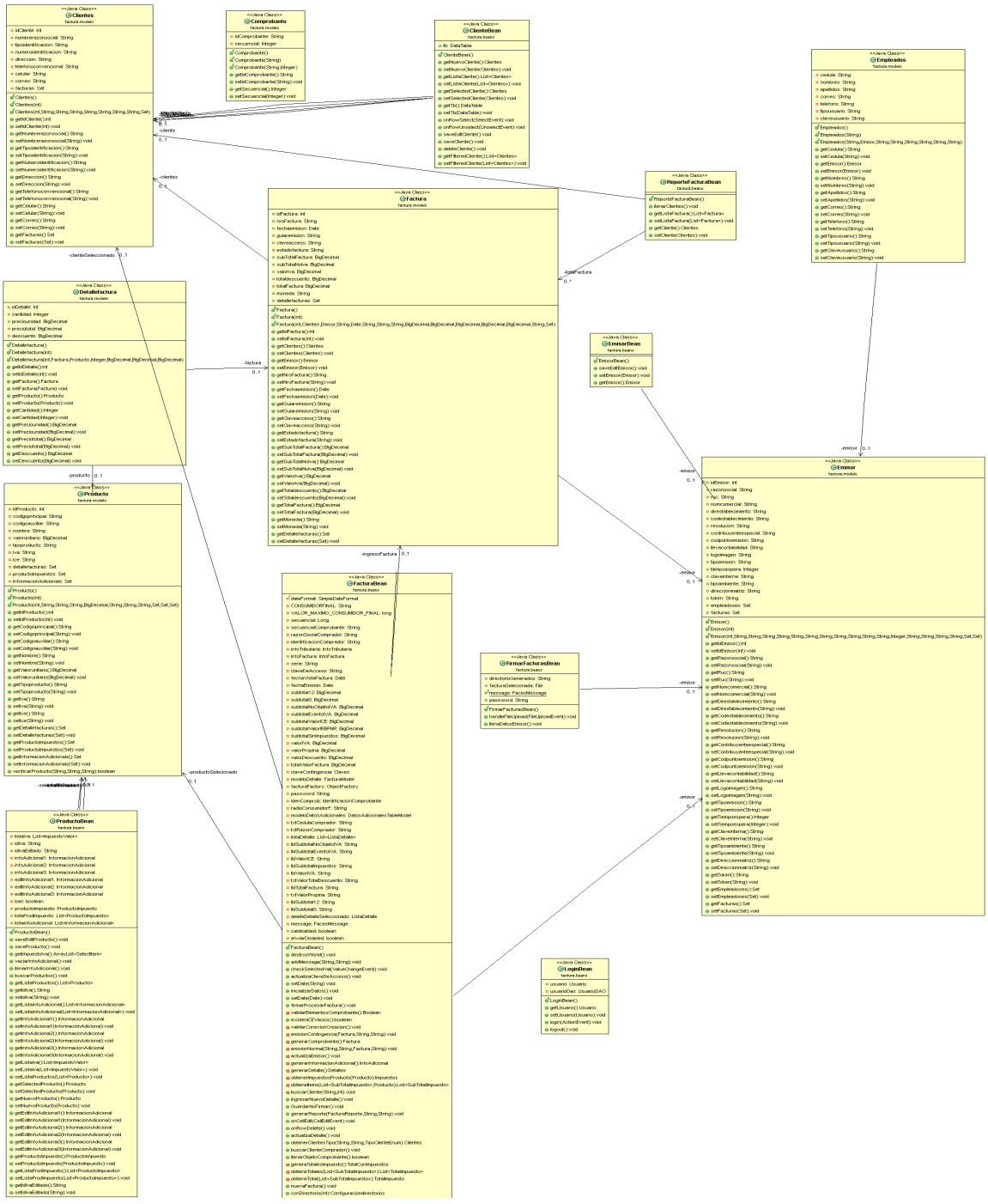


FIGURA 6.87: Diagrama de clases.

Fuente: Propia

6.6.5 ARQUITECTURA DE LA APLICACIÓN, DESCRIPCIÓN DE COMPONENTES.

Se describe toda la arquitectura de la aplicación y lo diferentes componentes.

6.6.5.1 DESCRIPCIÓN DE COMPONENTES.

TABLA 6.31: Descripción componentes de la aplicación.

Componentes	Descripción
Interfaces	**Interfaces a utilizar**
Login	Es para la verificación de usuario
Clientes	Ingresar, Modificar clientes
Emisor	Ingresar, modificar emisor
Enviar factura	Enviar facturas firmadas
Factura	Generar facturas
Firmar factura	Firmar electrónicamente las facturas
Productos	Ingresar, modificar, eliminar productos
Reporte facturas	Genera las facturas validados por SRI
Configurar WS	Configuración direcciones de web services
Clases	**Clases a utilizar**
Claves	Servirá para manipular los productos en base de datos
Clientes	Servirá para manipular los clientes en base de datos
Comprobantes	Servirá para manipular los comprobantes en base de datos
Configuración Directorios	Servirá para configurar los directorios de los comprobantes
Emisor	Generar emisor
Facturas	Generar facturas
Empleados	Verificación para loguear
DetalleFactura	Guarda el detalle de la factura
Impuestos	Para obtener los impuestos
ImpuestoValor	Sacar el valor de los impuestos
InformacionAdicional	Información adicional de los productos
Producto	Manipulación de los productos
ProductoImpuesto	Guardar los impuestos de los productos
Respuesta	Guarda las respuestas de las facturas
Usuario	Manipulación de los clientes
Base de Datos	
Claves	Contiene datos de claves
Clientes	Contiene datos de clientes
Comprobantes	Contiene datos de comprobantes
ConfiguracionDirectorios	Contiene datos de directorios

Emisor	Contiene datos del emisor
Facturas	Guarda las facturas generadas
Empleados	Contiene datos del empleado
DetalleFactura	Contiene datos de detalles de facturas
Impuestos	Contiene datos de los impuestos
ImpuestoValor	Contiene datos de los impuestos y su valor
InformacionAdicional	Contiene datos de la información de los productos
Producto	Contiene datos de los productos
ProductoImpuesto	Contiene datos de los impuestos del producto
Respuesta	Contiene datos de las respuestas de las facturas
Usuario	Contiene datos de los usuarios

Fuente: Propia

6.6.5.2 ARQUITECTURA DE LA APLICACIÓN DE FACTURACIÓN Y BASE DE DATOS.

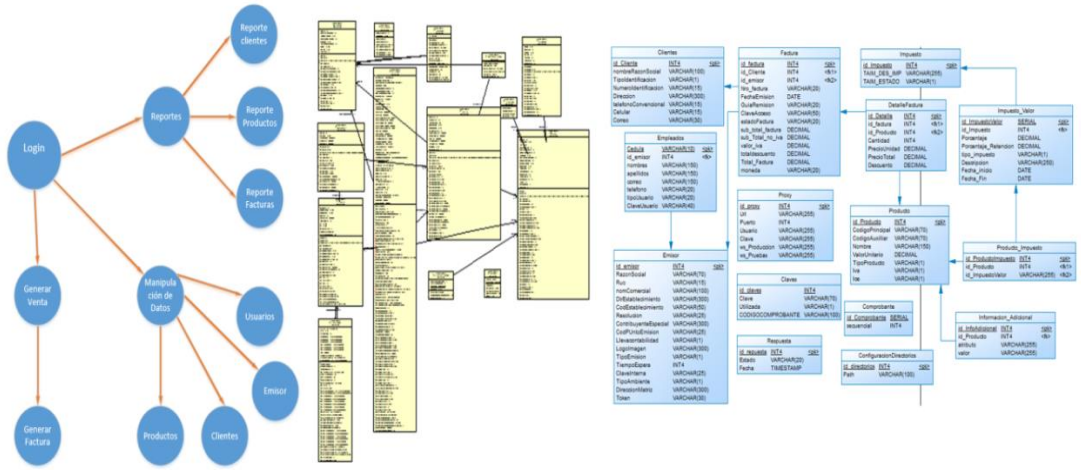


FIGURA 6.88: Arquitectura de la aplicación.

Fuente: Propia

6.7 FASE 5: CIERRE DEL PROYECTO.

Dado que el cliente ya no tiene más historias para poder ser incluidas en el sistema, es decir que se ha satisfecho las expectativas, ahora es necesario poner a prueba el rendimiento, confiabilidad y eficiencia de nuestro sistema de facturación. Para el cual se genera la documentación final que es la especificación de las pruebas del sistema.

6.7.1 ESPECIFICACIÓN DE PRUEBAS.

Las pruebas son una parte importante del sistema, debido a que las pruebas deben ser realizadas la mayor cantidad de veces lo que permitirá corregir errores y poder obtener resultados esperados.

En el sistema de Facturación electrónica las pruebas se realizaron en el escenario adecuado en el cual cumplan con todos y cada uno de los requerimientos y poder brindar toda la información de funcionalidad del sistema.

6.7.2 PRUEBAS DE ACEPTACIÓN.

Las pruebas que se realizaron fueron con la finalidad de comprobar si cada una de las historias de usuario de cada interacción correspondía y cumplían con la funcionalidad que se esperaba del sistema.

TABLA 6.32: Descripción componentes de la aplicación.

Prueba de aceptación	
Código:	**No. Historia de usuario**
Nombre de historia de usuario:	
Condiciones de ejecución:	
Entrada / Pasos de ejecución:	
Resultado esperado:	
Evaluación de la prueba:	

Fuente: Propia

- Descripción de los componentes del formato que se utilizara para las pruebas de aceptación:
- **Código.-** Número de identificación y único de la prueba de aceptación.
- **Número de historia de usuario.-** Número de historia seleccionada para la prueba de aceptación.
- **Nombre de historia de usuario.-** Nombre de historia de usuario a la cual se le realiza las pruebas.

- **Condiciones de ejecución.-** Condiciones que debe cumplir antes de la realización de la prueba de aceptación.

- **Entrada / Pasos de ejecución.-** Pasos que se siguen para poder probar la funcionalidad que tiene la historia de usuario.

- **Resultado esperado.-** Es la respuesta que se espera del sistema.

- **Evaluación de la prueba.-** Es el nivel de aceptación que el cliente da, sobre la respuesta del sistema.

  - **Aprobada:** Es cuando el sistema responde satisfactoriamente y cumple las expectativas de cliente.

  - **No aprobada:** Es cuando la respuesta que se obtiene del sistema no cumple la expectativa del cliente.

- **Observaciones.-** Información adicional que puede ser relevante en las pruebas de aceptación**.**

A continuación se describen las principales pruebas de aceptación del sistema de facturación.

La Tabla 6.26 muestra la prueba de aceptación de la historia de usuario Control y Autenticación de usuarios.

TABLA 6.33: Prueba de aceptación control y autenticación de usuarios.

	Prueba de aceptación
Código: P1	**No. Historia de usuario:** 1
Nombre historia de usuario: Control y autenticación de usuarios	
Condiciones de ejecución: Debe existir un usuario creado. Ingresar al sistema. Deben existir catálogos creados.	
Entrada / Pasos de ejecución: • Ingresar al sistema de facturación electrónica. • Ingresar usuario y clave	

Resultado esperado 1: Se ingresa con el usuario y clave al sistema donde nos mostrara las diferentes opciones que tiene el sistema.
Resultado esperado 2: Si los campos obligatorios se los deja en blanco el sistema muestra un mensaje de error “El campo es requerido”.
Evaluación de la prueba: Aprobado

Fuente: Propia

La Tabla 6.27 muestra la prueba de aceptación de la historia de usuario generar factura.

TABLA 6.34: Prueba de aceptación generar factura.

Prueba de aceptación	
Código: P2	**No. Historia de usuario:** 2
Nombre historia de usuario: Generar Factura	
Condiciones de ejecución: El usuario tiene que estar registrado en el sistema de facturación. Deben estar los catálogos creados. Debe existir el emisor. Debe estar parametrizado todo lo referente al SRI	
Entrada / Pasos de ejecución: • Ingresar al sistema de facturación electrónica. • Seleccionar la opción Generar Factura.	
Resultado esperado 1: Se abre la pantalla con el formato de la factura electrónica El emisor está ingresado automáticamente en la factura	
Resultado esperado 2: Se genera el número y la clave de acceso de la factura. Se selecciona automáticamente la opción Consumidor Final	
Evaluación de la prueba: Aprobado	

Fuente: Propia

La Tabla 6.28 muestra la prueba de aceptación de la historia de usuario generar búsqueda de productos.

TABLA 6.35: Prueba de aceptación generar búsqueda de productos.

Prueba de aceptación	
Código: P3	**No. Historia de usuario:** 3
Nombre historia de usuario: Generar búsqueda de productos	
Condiciones de ejecución: El usuario debe estar registrado en el sistema de facturación. Deben estar los catálogos creados. Debe existir el emisor. Debe estar parametrizado todo lo referente al SRI Deben estar creados y registrados los productos.	
Entrada / Pasos de ejecución: • Ingresar al sistema de facturación electrónica. • Seleccionar la opción generar factura. • Ingresar productos a la factura • Generar valores de impuestos, subtotal y el total del valor de la factura	
Resultado esperado 1: Los productos seleccionados aparecen en la factura.	
Resultado esperado 2: No permite seleccionar el mismo producto otra vez.	
Evaluación de la prueba: Aprobado	

.

Fuente: Propia

La Tabla 6.29 muestra la prueba de aceptación de la historia de usuario generar búsqueda del cliente.

TABLA 6.36: Prueba de aceptación generar búsqueda del cliente.

Prueba de aceptación	
Código: P4	**No. Historia de usuario:** 4
Nombre historia de usuario: Generar búsqueda del cliente	
Condiciones de ejecución: El usuario debe estar registrado en el sistema de facturación. Deben estar los catálogos creados.	

Debe existir el emisor. Debe estar parametrizado todo lo referente al SRI Deben estar creados y registrados los productos. Deben estar registrados los clientes.
Entrada / Pasos de ejecución: • Ingresar al sistema de facturación electrónica. • Seleccionar la opción generar factura. • Ingresar productos a la factura. • Generar valores de impuestos, subtotal y el total del valor de la factura. • Ingresar los datos del cliente.
Resultado esperado 1: Se selecciona el cliente q se buscó.
Evaluación de la prueba: Aprobado

Fuente: Propia

La Tabla 6.30 muestra la prueba de aceptación de la historia de usuario generación de facturas electrónicas firmadas.

TABLA 6.37: Prueba de aceptación generación de facturas electrónicas firmadas.

	Prueba de aceptación
Código: P5	**No. Historia de usuario:** 5
Nombre historia de usuario: Generación de facturas electrónicas firmadas	
Condiciones de ejecución: El usuario debe estar registrado en el sistema de facturación. Debe estar parametrizado todo lo referente al SRI. Debe existir la factura. Debe estar seleccionado el ambiente de pruebas del SRI.	
Entrada / Pasos de ejecución: • Ingresar al sistema de facturación electrónica. • Seleccionar la opción firmar factura. • Hacer clic en firmar • Seleccionar la factura que se va a firmar. • Esperar la respuesta del sistema	
Resultado esperado 1: Factura en estado firmado.	
Resultado esperado 2: Error en la firma de la factura.	
Evaluación de la prueba: Aprobado	

Fuente: Propia

La Tabla 6.31 muestra la prueba de aceptación de la Historia de usuario Generación de facturas electrónicas enviadas y autorizadas por SRI.

TABLA 6.38: Prueba de aceptación Generación de facturas autorizadas por SRI.

Prueba de aceptación	
Código: P6	**No. Historia de usuario:** 6
Nombre historia de usuario: Generación de facturas electrónicas enviadas y autorizadas por SRI	
Condiciones de ejecución: El usuario debe estar registrado en el sistema de facturación. Deben estar los catálogos creados. Debe existir el emisor. Debe estar parametrizado todo lo referente al SRI. Deben estar creados y registrados los productos. Deben estar registrados los clientes. Debe existir el formato de la factura. Debe estar seleccionado el ambiente de pruebas del SRI.	
Entrada / Pasos de ejecución: • Ingresar al sistema de facturación electrónica. • Seleccionar la opción Generar Factura. • Ingresar productos a la factura • Generar valores de impuestos, subtotal y el total del valor de la factura • Ingresar los datos del cliente. • Ingresar datos de venta. • Seleccionar el botón generar factura. • Esperar la respuesta del SRI.	
Resultado esperado 1: Se registra en el sistema el envío de la factura Se emite el mensaje de que la factura fue envía y aprobada por el SRI.	
Resultado esperado 2: Se registra en el sistema el envío de la factura. Se emite un mensaje de que la factura fue rechazada y el motivo del rechazo.	
Evaluación de la prueba: Aprobado	

Fuente: Propia

La Tabla 6.32 muestra la prueba de aceptación de la historia de usuario envío de las facturas electrónicas por mail.

TABLA 6.39: Prueba de aceptación Envío de las facturas electrónicas por mail.

Prueba de aceptación	
Código: P7	**No. Historia de usuario:** 7
Nombre historia de usuario: Envío de las facturas electrónicas por mail	
Condiciones de ejecución: El usuario debe estar registrado en el sistema de facturación. Deben estar los catálogos creados. Debe existir el emisor. Debe estar parametrizado todo lo referente al SRI. Deben estar creados y registrados los productos. Deben estar registrados los clientes. Debe existir el formato de la factura. Debe estar seleccionado el ambiente de pruebas del SRI.	
Entrada / Pasos de ejecución: • Ingresar al sistema de facturación electrónica. • Seleccionar la opción generar factura. • Ingresar productos a la factura • Generar valores de impuestos, subtotal y el total del valor de la factura • Ingresar los datos del cliente. • Ingresar datos de venta. • Seleccionar el botón generar factura. • Esperar la respuesta del SRI. • Enviar al cliente la factura	
Resultado esperado 1: Se registra en el sistema el envío de la factura El cliente recibe la factura electrónica en su email	
Resultado esperado 2: Se registra en el sistema el envío de la factura. No se envía la factura ya que el cliente no tiene un email	
Evaluación de la prueba: Aprobado	

Fuente: Propia

La Tabla 6.33 muestra la prueba de aceptación de la historia de usuario Emisión de las facturas en modo contingencia.

TABLA 6.40: Prueba de aceptación Emisión de las facturas en modo contingencia.

Prueba de aceptación	
Código: P8	**No. Historia de usuario:** 8
Nombre historia de usuario: Emisión de las facturas en modo contingencia	
Condiciones de ejecución: El usuario debe estar registrado en el sistema de facturación. Deben estar los catálogos creados. Debe existir el emisor. Debe estar parametrizado todo lo referente al SRI. Deben estar creados y registrados los productos. Deben estar registrados los clientes. Debe existir el formato de la factura.	
Entrada / Pasos de ejecución: • Ingresar al sistema de facturación electrónica. • Seleccionar la opción generar factura. • Ingresar productos a la factura • Generar valores de impuestos, subtotal y el total del valor de la factura • Ingresar los datos del cliente. • Ingresar datos de venta. • Seleccionar el botón generar factura	
Resultado esperado 1: Se registra en el sistema el envío de la factura No existe conexión con el SRI y la factura se genera en modo contingencia	
Evaluación de la prueba: Aprobado	

Fuente: Propia

La Tabla 6.34 muestra la prueba de aceptación de la historia de usuario Consulta del estado de la factura y reenvío**.**

TABLA 6.41: Prueba de aceptación Consulta del estado de la factura y reenvío**.**

Prueba de aceptación	
Código: P9	**No. Historia de usuario:** 9
Nombre historia de usuario: Consulta del estado de la factura y reenvío	
Condiciones de ejecución: El usuario debe estar registrado en el sistema de facturación. Deben estar los catálogos creados. Debe existir la factura.	
Entrada / Pasos de ejecución: • Ingresar al sistema de facturación electrónica. • Seleccionar la opción enviar factura. • Seleccionar el botón enviar.	
Resultado esperado 1: Se actualiza el estado de la factura en el sistema Se emite el mensaje de que la factura fue envía y aprobada por el SRI.	
Resultado esperado 2: Se actualiza el estado de la factura en el sistema Se emite un mensaje de que la factura fue rechazada y el motivo del rechazo.	
Evaluación de la prueba: Aprobado	

Fuente: Propia

La Tabla 6.35 muestra la prueba de aceptación de la Historia de usuario Gestión de clientes.

TABLA 6.42: Prueba de aceptación añadir clientes**.**

Prueba de aceptación	
Código: P10	**No. Historia de usuario:** 10
Nombre historia de usuario: Gestión de clientes	
Condiciones de ejecución: El usuario debe estar registrado en el sistema de facturación. Ingresar al sistema de facturación Deben estar los catálogos creados.	
Entrada / Pasos de ejecución: • Ingresar al sistema de facturación electrónica. • Seleccionar la opción clientes. • Seleccionar el botón añadir. • Seleccionar el tipo de identificación y el estado. • Ingresar los datos del cliente. • Seleccionar el botón guardar cliente.	
Resultado esperado 1: Se registran los datos del cliente en el sistema de facturación. La información de los clientes ingresados se muestra en la lista de clientes.	
Resultado esperado 2: Si algún campo obligatorio se lo deja vacío el sistema muestra el mensaje de error que el campo es requerido.	
Resultado esperado 3: Si los datos ingresados no cumplen con el formato de cada campo se muestra un mensaje de error de formato.	
Evaluación de la prueba: Aprobado	

Fuente: Propia

La Tabla 6.36 muestra la prueba de aceptación de la historia de usuario gestión de clientes.

TABLA 6.43: Prueba de aceptación editar clientes**.**

Prueba de aceptación	
Código: P11	**No. Historia de usuario:** 10
Nombre historia de usuario: Gestión de clientes	
Condiciones de ejecución: El usuario debe estar registrado en el sistema de facturación. Ingresar al sistema de facturación Deben estar los catálogos creados.	
Entrada / Pasos de ejecución: • Ingresar al sistema de facturación electrónica. • Seleccionar la opción clientes. • Seleccionar el cliente y hacer clic en el botón editar. • Seleccionar el tipo de identificación y el estado. • Ingresar los datos del cliente. • Seleccionar el botón editar cliente.	
Resultado esperado 1: Se editan los datos del cliente en el sistema de facturación. La información con los datos editados se muestra en la lista de clientes.	
Resultado esperado 2: Si algún campo obligatorio se lo deja vacío el sistema muestra el mensaje de error que el campo es requerido.	
Resultado esperado 3: Si los datos ingresados no cumplen con el formato de cada campo se muestra un mensaje de error de formato.	
Evaluación de la prueba: Aprobado	

Fuente: Propia

La Tabla 6.37 muestra la prueba de aceptación de la historia de usuario gestión de clientes.

TABLA 6.44: Prueba de aceptación listado de clientes**.**

Prueba de aceptación	
Código: P12	**No. Historia de usuario:** 10
Nombre historia de usuario: Gestión de clientes	
Condiciones de ejecución: El usuario debe estar registrado en el sistema de facturación. Ingresar al sistema de facturación Deben estar los catálogos creados.	
Entrada / Pasos de ejecución: • Ingresar al sistema de facturación electrónica. • Seleccionar la opción clientes.	
Resultado esperado 1: Se muestra la lista de los clientes con su información.	
Evaluación de la prueba: Aprobado	

Fuente: Propia

La Tabla 6.38 muestra la prueba de aceptación de la Historia de usuario gestión de Productos.

TABLA 6.45: Prueba de aceptación añadir productos.

Prueba de aceptación	
Código: P13	**No. Historia de usuario:** 11
Nombre historia de usuario: Gestión de productos	
Condiciones de ejecución: El usuario debe estar registrado en el sistema de facturación. Ingresar al sistema de facturación Deben estar los catálogos creados.	
Entrada / Pasos de ejecución: • Ingresar al sistema de facturación electrónica. • Seleccionar la opción productos. • Seleccionar el producto y hacer clic en el botón Ingresar producto.	

• Seleccionar el tipo de producto. • Ingresar los datos del producto. • Seleccionar el botón guardar.
Resultado esperado 1: Se guarda el producto en el sistema de facturación. La información con el producto ingresado se muestra en la lista de productos.
Resultado esperado 2: Si algún campo obligatorio se lo deja vacío el sistema muestra el mensaje de error que el campo es requerido.
Resultado esperado 3: Si los datos ingresados no cumplen con el formato de cada campo se muestra un mensaje de error de formato.
Evaluación de la prueba: Aprobado

Fuente: Propia

La Tabla 6.39 muestra la prueba de aceptación de la Historia de usuario Gestión de Productos.

TABLA 6.46: Prueba de aceptación editar producto**.**

	Prueba de aceptación
Código: P14	**No. Historia de usuario:** 11
Nombre historia de usuario: Gestión de productos	
Condiciones de ejecución: El usuario debe estar registrado en el sistema de facturación. Ingresar al sistema de facturación Deben estar los catálogos creados.	
Entrada / Pasos de ejecución: • Ingresar al sistema de facturación electrónica. • Seleccionar la opción productos. • Seleccionar el producto y hacer clic en el botón editar. • Seleccionar el tipo de producto. • Editar los datos del producto. • Seleccionar el botón editar producto.	

Resultado esperado 1: Se editan los datos del producto en el sistema de facturación. La información con los datos editados se muestra en la lista de productos.
Resultado esperado 2: Si algún campo obligatorio se lo deja vacío el sistema muestra el mensaje de error que el campo es requerido.
Resultado esperado 3: Si los datos ingresados no cumplen con el formato de cada campo se muestra un mensaje de error de formato.
Evaluación de la prueba: Aprobado

Fuente: Propia

La Tabla 6.40 muestra la prueba de aceptación de la historia de usuario gestión de clientes.

TABLA 6.47: Prueba de aceptación listado de clientes**.**

	Prueba de aceptación
Código: P15	**No. Historia de usuario:** 11
Nombre historia de usuario: Gestión de Productos	
Condiciones de ejecución: El usuario debe estar registrado en el sistema de facturación. Ingresar al sistema de facturación Deben estar los catálogos creados.	
Entrada / Pasos de ejecución: • Ingresar al sistema de facturación electrónica. • Seleccionar la opción productos.	
Resultado esperado 1: Se muestra la lista de los Productos con su información.	
Evaluación de la prueba: Aprobado	

Fuente: Propia

La Tabla 6.41 muestra la prueba de aceptación de la historia de usuario gestión de emisor.

TABLA 6.48: Prueba de aceptación guardar emisor.

	Prueba de aceptación
Código: P16	**No. Historia de usuario:** 12
Nombre historia de usuario: Gestión de emisor	
Condiciones de ejecución: El usuario debe estar registrado en el sistema de facturación. Ingresar al sistema de facturación Deben estar los catálogos creados.	
Entrada / Pasos de ejecución: • Ingresar al sistema de facturación electrónica. • Seleccionar la opción emisor. • Ingresar los datos del emisor. • Seleccionar el botón guardar.	
Resultado esperado 1: Se guarda el emisor en el sistema de facturación. La información con el emisor guardado se muestra en la página del Emisor.	
Resultado esperado 2: Si algún campo obligatorio se lo deja vacío el sistema muestra el mensaje de error que el campo es requerido.	
Resultado esperado 3: Si los datos ingresados no cumplen con el formato de cada campo se muestra un mensaje de error de formato.	
Evaluación de la prueba: Aprobado	

Fuente: Propia

La Tabla 6.42 muestra la prueba de aceptación de la historia de usuario Gestión de Emisor.

TABLA 6.49: Prueba de aceptación editar emisor**.**

Prueba de aceptación	
Código: P17	**No. Historia de usuario:** 12
Nombre historia de usuario: Gestión de emisor	
Condiciones de ejecución: El usuario debe estar registrado en el sistema de facturación. Ingresar al sistema de facturación Deben estar los catálogos creados. Debe estar creado un Emisor.	
Entrada / Pasos de ejecución: • Ingresar al sistema de facturación electrónica. • Seleccionar la opción emisor. • Ingresar o editar los datos del emisor. • Seleccionar el botón guardar.	
Resultado esperado 1: Se editan los datos del Emisor en el sistema de facturación. La información con los datos editados se muestra en la página del Emisor.	
Resultado esperado 2: Si algún campo obligatorio se lo deja vacío el sistema muestra el mensaje de error que el campo es requerido.	
Resultado esperado 3: Si los datos ingresados no cumplen con el formato de cada campo se muestra un mensaje de error de formato.	
Evaluación de la prueba: Aprobado	

Fuente: Propia

6.7.3 ANÁLISIS DE RESULTADOS.

Para realizar el análisis de los resultados se ha recolectado la información que nos ha dejado las pruebas de aceptación que se le hizo a las historias de usuario.

TABLA 6.50: Resultado de las Pruebas de aceptación.

No.	Historia de usuario	Código de prueba de aceptación	Resultado
1	Control y autenticación de usuarios	P1	Aprobado
2	Generar factura	P2	Aprobado
3	Generar búsqueda de productos	P3	Aprobado
4	Generar búsqueda del cliente	P4	Aprobado
5	Generación de facturas electrónicas firmadas	P5	Aprobado
6	Generación de facturas electrónicas enviadas y autorizadas por SRI	P6	Aprobado
7	Envío de las facturas electrónicas por mail	P7	Aprobado
8	Emisión de las facturas en modo contingencia	P8	Aprobado
9	Consulta del estado de la factura y reenvío	P9	Aprobado
10	Gestión de clientes	P10	Aprobado
11	Gestión de clientes	P11	Aprobado
12	Gestión de clientes	P12	Aprobado
13	Gestión de productos	P13	Aprobado
14	Gestión de productos	P14	Aprobado
15	Gestión de productos	P15	Aprobado
16	Gestión de emisor	P16	Aprobado
17	Gestión de emisor	P17	Aprobado

Fuente: Propia

6.7.3.1 RESULTADO DE LAS PRUEBAS DE ACEPTACIÓN.

La Tabla 6.43 muestra todos los resultados obtenidos de las pruebas de aceptación de las pruebas de usuario.

Con toda la información que se obtuvo de los resultados de las pruebas de aceptación de las pruebas de usuario, se evidencia que las pruebas de aceptación fueron aprobadas en su totalidad.

CAPÍTULO VII

7 CONCLUSIONES.

- Al comparar los ESB se observó que cada uno tiene su particularidad y distinta manera de funcionamiento, el estudio comparativo fue para demostrar cual ESB es el más apto para el sistema que se desarrolló de facturación electrónica, resultando como mejor opción OpenESB el cual se adaptó de la mejor manera al prototipo de Facturación Electrónica funcionando con un alto rendimiento y confiabilidad y es el que se adapta a las necesidades para este sistema de facturación electrónica, ya que en la simulación de este ESB observamos que proporciona simplicidad, eficiencia, durabilidad a largo plazo, ofreciendo herramientas para diseño, desarrollo, pruebas y desplegar aplicaciones.
- La utilización de Mule ESB con su plataforma de integración de peso ligero ha permitido conectar el sistema de facturación electrónica de una manera rápida y confiable que permite conectar otros sistemas y aplicaciones el cual puede manejar toda la comunicación entre los sistemas, lo que permite seguir y controlar todo lo que sucede.
- OpenESB resulto un ESB confiable, rápido y fácil al momento de integrar e implementar los procesos de negocio del sistema de facturación electrónica, OpenESB cumplió con todos los requisitos planteados en el prototipo.
- La facturación electrónica en las empresas brinda beneficios ambientales, económicos y administrativos, debido a que ya no se utiliza papel y los costos se reducen.
- La metodología XP (Programación Extrema) ayudo a que el sistema cumpla con los objetivos de funcionalidad y requerimientos que tenía el cliente, disminuyendo el tiempo y el esfuerzo en el levantamiento de los requerimientos y se enfoca en el desarrollo de todo el sistema, la metodología ayudo a que las pruebas realizadas nos sirvieran a encontrar fallas y errores del funcionamiento

del sistema; una vez que se corrigieron, el sistema de facturación cumplió con todos los requerimientos planteados.

- Los buses de servicio implementados ayudarán a cumplir con lo que esta establece en la Constitución de la República del Ecuador en el Art. 14 en el que reconoce el derecho a tener un ambiente sano y ecológicamente muy equilibrado, que nos garantice el buen vivir y que la Constitución de la República del Ecuador en el Art. 15, determina el Estado debe promover, en los sectores públicos y privados, el uso de una tecnología ambientalmente limpias y también energías alternativas que no contaminen y de muy bajo impacto. Es por eso que se ha desarrollado un prototipo de facturación electrónica que ayuda a cumplir no solo con la Constitución sino también con lo que está en el Art. 15 en el acuerdo No. 131 del ministerios del ambiente el cual establece que sea utilizado el sistema informático con cero papeles como medio idóneo para la comunicación interna entre las instituciones sujetas a dicho Acuerdo Ministerial.

7.1 RECOMENDACIONES.

- Se recomienda un estudio más exhaustivo de cada uno de los ESB para poder explotar al máximo las capacidades que tienen y no solo sea utilizado en un sistema de facturación electrónica sino en los sistemas de las empresas para un mejor funcionamiento y se recomienda para futuros trabajos de investigación los diferentes ESB que existen en la actualidad ya que nos brindan una variedad de beneficios para las empresas.

- En los escenarios fuertemente acoplados se recomienda el uso de un ESB que desacople a todas las aplicaciones, haciendo posible que el uso sea fácil y centralizado.

- Se recomienda la constante revisión de leyes y reglamentos de la emisión de comprobantes electrónicos lo que va a permitirá realizar una retroalimentación al sistema de facturación electrónica para que pueda cumplir las Normas del SRI.

- Se recomienda usar este prototipo desarrollado para ayudar, apoyar a los sistemas de facturación, estableciendo alternativas para colaborar con el medio ambiente reduciendo el uso de papel mediante la utilización de facturación electrónica desarrollada con los buses de servicio que se dio a conocer.

7.2 GLOSARIO DE TÉRMINOS.

SOA: Arquitectura Orientada a Servicios (Service Oriented Architecture).

ESB: Bus de Servicios Empresarial (Enterprise service bus)**.**

XP: Programación Extrema (eXtreme Programming).

SOAP: Protocolo Simple de Acceso a Objetos (Simple Object Access Protocol).

IT: Tecnologías de la Información (InformationTechnology).

WSDL: protocolo basado en XML (Web Services Description Language).

HTTP: Protocolo de Transferencia de Hipertexto**.**

BPEL: Lenguaje de Ejecución de Procesos de Negocio (Business Process Execution Language).

JBI: Java Business Integration.

JavaEE: Java Platform, Enterprise Edition**.**

UDDI: Universal Description, Discovery and Integration.

ERP: Sistemas de planificación de recursos empresariales (Enterprise Resource Planning)**.**

CRM: Estrategia de negocio enfocada al cliente (Customer Relationship Management)**.**

BI: Inteligencia empresarial (business intelligence).

JMS: Servicio de Mensajes Java (Java Message Service).

QoS: Calidad de Servicio (Quality of Service).

MEP: Patrón de intercambio de mensajes.

JDBC: Java Database Connectivity.

Pojo: Plain Old Java Object**.**

REST: Transferencia de Estado Representacional (Representational State Transfer)**.**

EJB: Enterprise JavaBeans**.**

API: Interfaz de programación de aplicaciones (Application Programming Interface)**.**

B2B: Business-to-business**.**

CPU: Unidad Central de Procesamiento (Central Processing Unit).

SLA: Acuerdo de nivel de servicio (Service Level Agreement).

IPaaS: Integration for the Cloud Integration platform as a service.

XQuery: Lenguaje de consulta diseñado para colecciones de datos XML

OSGi: Permita diseñar plataformas compatibles que puedan proporcionar múltiples servicios (Open Services Gateway initiative).

CSV: Documento en formato abierto sencillo (comma-separated values)**.**

7.3 BIBLIOGRAFÍA.

Borja, E., Florez, M. E., & Torres Leon, P. A. (1 de Febrero de 2011). *ESPECIALIZACIÓN EN PROCESOS PARA EL DESARROLLO DE SOFTWARE.* Obtenido de http://bibliotecadigital.usbcali.edu.co/jspui/bitstream/10819/351/1/Comparativo_%20Software_Eficacia_Borja_2011.pdf

Cenatic. (10 de Enero de 2014). *Centro Nacional de Referencia de Aplicación de las TIC*. Obtenido de Cenatic: http://www.cenatic.es/laecsp/page21/page22/page22.html

Centro de Alto Rendimiento de Accenture. (17 de 09 de 2012). *Accenture.* Obtenido de http://www.accenture.com/es-es/Pages/insight-arquitectura-orientada-servicios.aspx

Erl, T. (2012). *SOA Principles of Service Design.* Boston: Pearson Education. Inc.

Exposito, E., & Diop, C. (11 de Septiembre de 2014). *Smart SOA Platforms in Cloud Computing Architectures.* California: FOCUS Series. Obtenido de http://fergunet.blogspot.com/2007/09/open-esb-vs-servicemix.html

Gras, J. C. (20 de Marzo de 2012). *Horizontes BPM*. Obtenido de modelación, arquitectura... paradigma: http://horizontesbpm.blog.com/?p=111#_ftn2

Hurwitz, J., Bloor, R., Baroudi, C., & Kaufman , M. (2007). *Service Oriented Architecture For Dummies.* Indianápolis, Indiana: Wiley Publishing, Inc.

IDG Communications. (01 de Febrero de 2012). *Network World*. Obtenido de http://www.networkworld.es/archive/integracion-de-aplicaciones-en-esb

Ingeniría de Software U. Union Bolivariana. (12 de Enero de 2014). *Ingeniería de Software*. Obtenido de http://ingenieriadesoftware.mex.tl/52753_XP---Extreme-Programing.html

José, C., Leszek, M., & Joaquim, F. (31 de Mayo de 2012). *Expertise Solutions.* Wroclaw: Springer. Obtenido de http://www.expertise-solutions.com.pe/frameworks.html

Kendall, K., & Kendall, J. (2011). *Análisis y Diseño de Sistemas.* Naucalpan de Juárez: Pearson Educación.

López, W. N. (2009). *Mejores Prácticas SOA, Estandares y Gobernabilidad.* San Juan: Caribeean White Paper.

López-Vázquez, M. A.-P. (2012). Fundamentos de las Infraestructuras de Datos Espaciales (IDE). BibliotecaOnline SL. Obtenido de ISBN: 8493919683, 9788493919689

Marquéz Solis, S. (2007). *Web Semántica y Servicios Web Semánticos.* Universitat Oberta de Catalunya: Ingenieríe Técnica en Información De Gestión.

Microsoft Corporation. (08 de Junio de 2008). *WillyDev.* Obtenido de http://www.willydev.net/InsiteCreation/v1.0/WillyCrawler/2008.06.08.Poster.Guia%20ESB%20de%20Microsoft.pdf

Microsoft Corporation. (10 de Diciembre de 2011). *Microsoft.* Obtenido de download.microsoft.com/download/.../070717-Real_World_SOA.pdf

MuleSoft. (1 de DIciembre de 2013). *Mule Community*. Obtenido de http://www.mulesoft.org/what-mule-esb

OpenESB Community. (02 de Julio de 2013). *OpenESB The Open Enterprise Service Bus*. Obtenido de http://www.open-esb.net/index.php?option=com_content&view=featured&Itemid=496

Oposiciones TIC. (22 de Agosto de 2012). *Oposiciones TIC*. Obtenido de http://oposicionestic.blogspot.com/2012/08/arquitectura-soa-orientada-servicios.html

Ramos, J. A. (12 de Enero de 2012). *Adictos al Trabajo*. Obtenido de http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=primeros_pasos_mule#04

Rojas, D. (13 de Enero de 2009). Obtenido de IComparable: http://icomparable.blogspot.com/

Sánchez, M. S. (1 de Enero de 2009). *Sociedad de la Información.* Obtenido de http://www.sociedadelainformacion.com/

TCP. (2 de Mayo de 2014). *tcp Dirige tu negocio*. Obtenido de http://www.tcpsi.com/vermas/integracion_esb.htm

Tibco. (08 de Junio de 2008). *WillyDev.* Obtenido de Using System.Makondo;: http://www.willydev.net/InsiteCreation/v1.0/WillyCrawler/2008.06.08.Articulo.El%20papel%20de%20un%20bus%20de%20servicios%20empresariales%20(ESB)%20en%20una%20SOA.pdf