Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

COMPARACIÓN DE LOS SERVIDORES DE APLICACIONES ORACLE ORIENTADO A SU CONFIGURACIÓN Y DESPLIEGUE DE APLICACIONES

Sindy Sabrina Rodas López
Asesorado por el Ing. Iván Nicolás García Solís

Guatemala, noviembre de 2014

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

COMPARACIÓN DE LOS SERVIDORES DE APLICACIONES ORACLE ORIENTADO A SU CONFIGURACIÓN Y DESPLIEGUE DE APLICACIONES

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

SINDY SABRINA RODAS LÓPEZ
ASESORADO POR EL ING. IVÁN NICOLÁS GARCÍA SOLÍS

AL CONFERÍRSELE EL TÍTULO DE

INGENIERA EN CIENCIAS Y SISTEMAS

GUATEMALA, NOVIEMBRE DE 2014

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Murphy Olympo Paiz Recinos
VOCAL I	Ing. Alfredo Enrique Beber Aceituno
VOCAL II	Ing. Pedro Antonio Aguilar Polanco
VOCAL III	Inga. Elvira Miriam Ruballos Samayoa
VOCAL IV	Br. Narda Lucía Pacay Barrientos
VOCAL V	Br. Walter Rafael Véliz Muñoz
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Murphy Olympo Paiz Recinos
EXAMINADOR	Ing. Oscar Alejandro Paz Campos
EXAMINADOR	Ing. Pedro Pablo Hernández Ramírez
EXAMINADOR	Ing. Jose Ricardo Morales Prado
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

COMPARACIÓN DE LOS SERVIDORES DE APLICACIONES ORACLE ORIENTADO A SU CONFIGURACIÓN Y DESPLIEGUE DE APLICACIONES

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, el 15 de enero de 2014.

Sindy Sabrina Rodas López

Guatemala, 20 de agosto de 2014

Ingeniero:
Carlos Azurdia
Coordinador de Proyectos de Graduación
Escuela de Ciencias y Sistemas
Facultad de Ingeniería

Respetable ingeniero

La presente es para hacer de su conocimiento que he revisado el trabajo de graduación que lleva como título **"COMPARACIÓN DE LOS SERVIDORES DE APLICACIONES ORACLE ORIENTADO A SU CONFIGURACIÓN Y DESPLIEGUE DE APLICACIONES"**, redactado y desarrollado por la estudiante Sindy Sabrina Rodas López quien se identifica con carné universitario **2007-15087** de la carrera de **Ingeniería en Ciencias y Sistemas**.

Con la revisión y corrección del presente trabajo de graduación hago constar que ha alcanzado los objetivos propuestos, por lo tanto el autor de este trabajo y mi persona, como asesor, nos hacemos responsables de contenido del mismo.

Sin otro particular, me suscribo a usted.

Atentamente,

Iván Nicolás García Solís
Ingeniero en Ciencias y Sistemas
Colegiado No. 8870

Ing. Iván García
Colegiado 8870
Asesor

Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 10 de Septiembre de 2014

Ingeniero
Marlon Antonio Pérez Turk
Director de la Escuela de Ingeniería
En Ciencias y Sistemas

Respetable Ingeniero Pérez:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación de la estudiante **SINDY SABRINA RODAS LÓPEZ** con carné **2007-15087,** titulado: **"COMPARACIÓN DE LOS SERVIDORES DE APLICACIONES ORACLE ORIENTADO A SU CONFIGURACIÓN Y DESPLIEGUE DE APLICACIONES",** y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,

Ing. Carlos Alfredo Azurdia
Coordinador de Privados
y Revisión de Trabajos de Graduación

ESCUELA DE CIENCIAS Y SISTEMAS

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERÌA
ESCUELA DE CIENCIAS Y SISTEMAS
TEL: 24767644

El Director de la Escuela de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del asesor con el visto bueno del revisor y del Licenciado en Letras, del trabajo de graduación ***"COMPARACIÓN DE LOS SERVIDORES DE APLICACIONES ORACLE ORIENTADO A SU CONFIGURACIÓN Y DESPLIEGUE DE APLICACIONES"****, realizado por la estudiante SINDY SABRINA RODAS LÓPEZ, aprueba el presente trabajo y solicita la autorización del mismo.*

"ID Y ENSEÑAD A TODOS"

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION DE INGENIERIA EN CIENCIAS Y SISTEMAS

Ing. Marlon Antonio Pérez Türk
Director, Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 06 de noviembre 2014

Universidad de San Carlos
de Guatemala

Facultad de Ingeniería
Decanato

Ref.DTG.D.617-2014

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería en Ciencias y Sistemas, al trabajo de graduación titulado: **COMPARACIÓN DE LOS SERVIDORES DE APLICACIONES ORACLE ORIENTADO A SU CONFIGURACIÓN Y DESPLIEGUE DE APLICACIONES,** presentado por la estudiante universitaria: **Sindy Sabrina Rodas López,** después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, se autoriza la impresión del mismo.

IMPRÍMASE.

Ing. Alfredo Enrique Beber Aceituno
Decano a.i.

Guatemala, noviembre de 2014

/cc

ACTO QUE DEDICO A:

Dios	Por permitirme llegar a este punto de mi carrera y acompañarme en todo momento.
Mis padres	Julio Rodas y Arely López, por su apoyo y amor incondicional.
Mis hermanos	Astrid y Maximiliano Rodas, por animarme y apoyarme a siempre seguir adelante.
Mi sobrino	Santiago Pereira Rodas, por haber traído tanta alegría en esta etapa de mi vida.
Mi familia	Mis abuelos, tíos y primos por creer en mí y en lo que soy capaz de hacer.

AGRADECIMIENTOS A:

La Universidad de San Carlos de Guatemala	Por permitirme formar parte de tan prestigiosa institución de la que estoy orgullosa de ser estudiante.
Facultad de Ingeniería	Por el conocimiento aprendido y las experiencias vividas durante todos estos años.
Mis amigos de la Facultad	Definitivamente la universidad no hubiera sido lo mismo sin ustedes. Los llevo en mi corazón siempre.
Mis amigos de la vida	Por su apoyo, comprensión y ánimos brindados durante mis estudios. Forman parte importante de mi vida y de este logro.
Ing. Iván García	Por ser más que asesor, un amigo, por guiarme y compartir sus conocimientos conmigo durante la realización de este trabajo de graduación.

Jorge Mario Arriaza	Mejor amigo y compañero de vida. Por ser mí apoyo y ayudarme a lograr lo que he logrado hasta hoy.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES V
LISTA DE SÍMBOLOS VII
GLOSARIO IX
RESUMEN XIII
OBJETIVOS XV
INTRODUCCIÓN XVII

1. CONCEPTOS BÁSICOS DE SERVIDORES DE APLICACIONES 1
   1.1. Definición de servidor de aplicaciones 1
   1.2. Arquitectura de funcionamiento de un servidor de aplicaciones 1
   1.3. Características comunes 3
   1.4. Ventajas 4
   1.5. Desventajas 5

2. CARACTERÍSTICAS DE LOS SERVIDORES DE APLICACIONES ORACLE 7
   2.1. Conceptos introductorios WebLogic 7
   2.2. Conceptos introductorios GlassFish Server 8
   2.3. Alta disponibilidad 10
      2.3.1. WebLogic Server cluster 10
      2.3.2. GlassFish Server cluster 12
   2.4. Seguridad 13
      2.4.1. WebLogic Security Service 14
      2.4.2. GlassFish Server System Security 23

2.5. Herramientas de administración 27
2.5.1. Herramientas de WebLogic 27
2.5.2. Herramientas de GlassFish 30
2.6. Estándares soportados 32

3. ADMINISTRACIÓN DE SERVIDORES DE APLICACIONES ORACLE 35
3.1. Tareas comunes de administración de WebLogic Server 35
3.1.1. Instalación 35
3.1.2. Inicio y detención de servidores 39
3.1.3. Configurar las conexiones a servidores web. 42
3.1.4. Configurar las conexiones a bases de datos 44
3.1.5. Configurar servicios de mensajería 45
3.1.6. Monitorear servicios y recursos 48
3.1.7. Optimizar el rendimiento de una aplicación 51
3.1.8. Configurar registro en bitácoras 52
3.2. Tareas comunes de administración de GlassFish 54
3.2.1. Instalación 54
3.2.2. Inicio y detención de servidores 58
3.2.3. Configurar las conexiones a la base de datos 59
3.2.4. Configurar los servicios de mensajería 60
3.2.5. Monitorear servicios y recursos 62
3.2.6. Optimizar el rendimiento de una aplicación 63
3.2.7. Configurar el registro en bitácoras 64

4. CONFIGURACIÓN PARA EL DESPLIEGUE DE APLICACIONES 67
4.1. Proceso de despliegue de aplicaciones para un WebLogic Server 67
4.1.1. Utilizar archivos empaquetados 67

4.1.2. Nombres de despliegue predeterminados 69
4.1.3. Requerimientos de nombres de aplicaciones 69
4.1.4. Directorio de instalación de la aplicación 70
4.1.5. Ciclo de vida de la configuración 72
4.1.6. Descriptor de despliegue 74
4.1.7. Plan de despliegue 75
4.2. Despliegue de aplicaciones para un GlassFish Server 75
4.2.1. Funcionalidad general 76
4.2.2. Descriptores y anotaciones..... 77
4.2.3. Despliegue basado en módulos..... 78
4.2.4. Despliegue basado en aplicaciones 80
4.2.5. Eventos de ensamblaje y despliegue 80

5. PRÁCTICA 83
5.1. Ambiente de pruebas WebLogic Server 83
5.1.1. Instalación de software 83
5.1.2. Creación y configuración de instancia 84
5.1.3. Despliegue de aplicaciones 85
5.1.4. Conexión a base de datos (JDBC) 86
5.2. Ambiente de pruebas GlassFish Server 86
5.2.1. Instalación de software 87
5.2.2. Creación y configuración de instancia 87
5.2.3. Despliegue de aplicaciones 88
5.2.4. Conexión a base de datos (JDBC) 88
5.3. Resumen comparativo 89
5.3.1. Aspecto gráfico 90

CONCLUSIONES.....95
RECOMENDACIONES 97
BIBLIOGRAFÍA.....99

ÍNDICE DE ILUSTRACIONES

FIGURAS

1. Arquitectura de funcionamiento de un servidor de aplicaciones 2
2. Relación entre usuarios, grupos, principales y sujetos 16
3. Descripción del reino de seguridad de WebLogic Server..... 19
4. Proveedores de seguridad de WebLogic Server..... 22
5. Mapeo de roles 25
6. Funcionamiento del Conector de WebLogic para servidores web. 44
7. Directorio de instalación de aplicaciones 71
8. Asistente de instalación WebLogic Server 91
9. Asistente de instalación GlassFish Server 91
10. Inicio de la consola de administración de WebLogic Server 92
11. Inicio de la consola de administración de GlassFish Server 92
12. Consola de administración de WebLogic Server..... 93
13. Consola de administración de GlassFish Server..... 93

TABLAS

I. Comparación de alta disponibilidad 13
II. Comparación de seguridad 26
III. Comparación de herramientas de administración 32
IV. Estándares soportados por WebLogic y GlassFish Server 33
V. Comparativo instalación de software 89
VI. Comparativo configuración de instancia 89
VII. Comparativo despliegue de aplicaciones..... 90

VIII. Comparativo conexión a base de datos (JDBC)..... 90

LISTA DE SÍMBOLOS

Símbolo	Significado
Gb	Gigabytes
Ghz	Gigahertz

GLOSARIO

Api — Conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro programa como una capa de abstracción.

C++ — Es un lenguaje de programación diseñado para extender el lenguaje de programación C con mecanismos que permitan la manipulación de objetos.

Cluster — Aplica a los conjuntos de computadoras construidos mediante la utilización de componentes comunes que se comportan como si fuesen una única computadora.

Framework — Es un esquema para el desarrollo y/o la implementación de una aplicación.

JAR — Es un tipo de archivo que permite ejecutar aplicaciones escritas en lenguaje Java.

Java EE — Es una plataforma de programación para desarrollar y ejecutar programas de aplicaciones en el lenguaje de programación Java. Permite utilizar arquitectura de capas y componentes modulares ejecutándose sobre un servidor de aplicaciones.

Jython	Es un lenguaje de programación de alto nivel, dinámico y orientado a objetos basado en Python e implementado en Java.
Linux	Término utilizado para referirse a la combinación del núcleo libre similar a Unix.
Middleware	Es un software que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones, redes, hardware y/o sistemas operativos.
Multiusuario	Es la característica de un programa o sistema operativo que permite proveer servicio y procesamiento a múltiples usuarios simultáneamente.
Pool	Es un conjunto de recursos inicializados que se mantienen listos para su uso, en lugar de ser asignados y destruidos bajo demanda.
RAM	Es donde la computadora guarda los datos que está utilizando en el presente. Es considerado temporal porque los datos y programas permanecen en ella mientras la computadora está encendida o no sea reiniciada.
RDBMS	Tipo de sistema de gestión de bases de datos que soporta tablas relacionales.

Sandbox	Es un mecanismo para ejecutar programas con seguridad y de manera separada.
Script	Es un programa usualmente simple, que por lo regular se almacena en un archivo de texto plano.
Single sign-on	Es un procedimiento de autenticación que habilita al usuario para acceder a varios sistemas con una sola instancia de identificación.
Sistema legacy	Es un sistema antiguo que funciona en una organización y no pueden ser remplazados por que tiene un alto costo o porque funciona eficaz y eficientemente y no es necesario cambiarlo.
URL	Uniform Resource Locator
WAR	Web Application Archive
XML	Extensible Markup Language
ZIP	Formato de compresión sin perdida.

RESUMEN

En el mundo de los servidores de aplicaciones, Oracle provee dos alternativas para utilizar con varios tipos de aplicaciones. Estas alternativas son WebLogic Server y GlassFish Server. Los servidores de aplicaciones son los encargados de ejecutar las aplicaciones previamente desarrolladas, por lo que deben tener un ambiente configurado especialmente para esta tarea y para que se ejecute de la mejor manera.

Entre los beneficios que se pueden obtener al utilizar estos servidores de aplicaciones, está la posibilidad de configurar alta disponibilidad para asegurarnos que las aplicaciones estén disponibles en todo momento y que en caso de alguna falla, se pueda tener un sitio de contingencia que sea capaz de seguir proporcionando el servicio. Además de esto, se obtiene la capacidad de proveer seguridad por medio de una arquitectura en capas, medidas de seguridad como auditoría, autenticación y la posibilidad de incluir proveedores externos de seguridad si así lo desea el usuario.

También cada servidor cuenta con sus propias herramientas de administración que facilitarán las tareas necesarias para asegurar el funcionamiento adecuado del servidor y las tareas que se deben realizar para un correcto despliegue de aplicaciones. Desplegar una aplicación quiere decir colocarla en el servidor y que esté disponible para ser utilizada. Los dos servidores cuentan con características y herramientas que buscan básicamente el mismo objetivo, la diferencia radica en que tan amigable al usuario son estas herramientas y que tan fácil es aprender a utilizarlas para lograr tener el servidor funcionando y las aplicaciones accesibles a todos los usuarios.

OBJETIVOS

General

Determinar que servidor de aplicaciones Oracle es el más recomendado basándose en su facilidad de uso y la utilidad que representa al usuario según las capacidades que brinda.

Específicos

1. Especificar capacidades de los servidores de aplicaciones Oracle.

2. Determinar las tareas de administración más comunes en los servidores de aplicaciones Oracle.

3. Definir conceptos y procedimientos básicos para el despliegue de aplicaciones.

4. Comprobar la facilidad de uso mediante la configuración de ambientes de prueba con ambos servidores.

INTRODUCCIÓN

Los productos Oracle en su mayoría son utilizados por grandes empresas que son las que pueden costear este tipo de tecnologías. En cuanto a los servidores de aplicaciones, Oracle proporciona dos opciones que en teoría ofrecen la misma funcionalidad con la diferencia que una necesita licenciamiento como es WebLogic Server y una que es gratuita como lo es GlassFish Server.

El tema a tratar con estos servidores es encontrar razones, aparte del tema económico, de por qué se debe elegir uno sobre otro basándose en la facilidad de uso y la utilidad que le representa al usuario. Para poder determinar estos factores, se desarrollan cinco capítulos con la información relevante que ayuda a este objetivo.

En el primer capítulo, se define qué es un servidor de aplicaciones, cuáles son sus características comunes para entender qué capacidades se obtienen al utilizarlos, sus ventajas, desventajas y la arquitectura sobre la que funcionan, de esta forma se introduce al lector en el tema de los servidores de aplicaciones.

En el segundo capítulo se determinan las características que proporcionan los servidores de aplicaciones Oracle, se compara cómo maneja cada uno características como la alta disponibilidad, seguridad y las herramientas de administración.

En el tercer capítulo se listan algunas de las tareas de administración más comunes de ambos servidores, tratando así de determinar la complejidad de

cada una de estas tareas con respecto a administrar un WebLogic Server o un GlassFish Server. Complejidad que influye en el factor facilidad de uso que se busca comprobar con este trabajo.

En el cuarto capítulo se abarcan los temas relacionados con el despliegue de aplicaciones, explicando de qué trata este proceso y qué tareas están involucradas para cada uno de los servidores.

Por último, en el capítulo cinco se realiza una pequeña práctica donde se configuran los servidores y se despliega una aplicación de prueba para verificar el funcionamiento de los mismos y determinar también de una forma práctica la facilidad de uso de sus herramientas.

1. CONCEPTOS BÁSICOS DE SERVIDORES DE APLICACIONES

1.1. Definición de servidor de aplicaciones

Un servidor de aplicaciones es el encargado de ejecutar la lógica de negocio sobre la que se construyen las aplicaciones y gestionar el acceso a los datos de las mismas. Básicamente los servidores de aplicaciones se encargan de los requerimientos no funcionales para que los desarrolladores puedan enfocarse en los requerimientos funcionales. Es un ejemplo del modelo cliente-servidor donde el cliente ejecuta peticiones de requerimientos y el servidor se encarga de procesar y responder estas peticiones. Entre los principales beneficios de la utilización de servidores de aplicaciones está la centralización y la disminución de la complejidad en el desarrollo de aplicaciones.

1.2. Arquitectura de funcionamiento de un servidor de aplicaciones

Partiendo del modelo cliente-servidor, los servidores de aplicaciones implementan una arquitectura de tres capas donde se divide la funcionalidad para optimizar el uso de recursos, como se muestra en la figura 1.

Figura 1. **Arquitectura de funcionamiento de un servidor de aplicaciones**

Fuente: Qué es un servidor de aplicaciones. http://es.over-blog.com/Que_es_un_servidor_de_aplicaciones-1228321779-art127891.html. Consultado 5 de octubre de 2013.

- Cliente: es el que contiene los componentes de usuario, la lógica de la aplicación específica del usuario y la interfaz gráfica.

- Lógica del negocio: constituye un ambiente multiusuario y mantiene los componentes compartidos de la aplicación. En este punto se lleva a cabo la coordinación de transacciones de múltiples usuarios con la capa de almacenamiento y viceversa. Las operaciones con un uso masivo de datos deben ejecutarse en este nivel.

- Almacenamiento: en este nivel se encuentra la base de datos. Se especializa en dar un servicio de persistencia a los datos de la aplicación y permite manejar grandes volúmenes de ellos. Un aspecto a tomar en cuenta es que la rapidez en la comunicación entre capas depende si están alojadas en la misma máquina o hay un tiempo de transmisión en red que debe ser considerado.

Al tener la mayor parte del código en el nivel de la aplicación, esta se aísla de la interfaz de usuario y de la base de datos por lo que se pueden hacer cambios sin que esto afecte a los otros niveles. Asimismo, los datos están centralizados y fácilmente accesibles sin necesidad de moverlos todos hasta el cliente.

1.3. Características comunes

Los servidores deben tener ciertas características o brindar ciertos servicios para que estos sean considerados servidores de aplicaciones. Entre éstas características mencionamos las siguientes:

- Alto rendimiento: el servidor está preparado para recibir una gran demanda de procesamiento de información cuyas operaciones involucran uso de procesadores, memoria, disco y otros recursos de hardware, por lo que la comunicación entre ellos debe ser rápida y de igual forma será la entrega de datos hacia las aplicaciones en el cliente.

- Alta disponibilidad: se refiere a que el servidor de aplicaciones debe estar funcionando las 24 horas del día los 365 días del año, o en su defecto, reduciendo el tiempo que los servidores se encuentran fuera de alcance al mínimo posible. Para poder garantizar la continuidad en el servicio se deben utilizar técnicas de balanceo de carga y recuperación ante fallos.

- Escalabilidad: es la capacidad del servidor de aplicaciones de crecer cuando se incrementa la carga de trabajo, por ejemplo cuando aumenta considerablemente el número de peticiones de clientes. Cada servidor tiene una capacidad finita de recursos y sólo puede servir un número limitado de peticiones, por lo que si la demanda incrementa se debe ser

capaz de incorporar nuevas máquinas para atender de forma aceptable la demanda.

- Independencia del cliente: esta característica garantiza que se pueda acceder a aplicaciones alojadas en el servidor de aplicaciones desde cualquier cliente. Entre los clientes más comunes pueden ser: un navegador web, una aplicación java, C++ y otros.

- Integración con otros sistemas: el servidor de aplicaciones debe tener la posibilidad de conectarse con cualquiera de las bases de datos más utilizadas, aplicaciones o sistemas *legacy* previamente existentes, dando así la posibilidad de reutilización y de fácil integración.

1.4. Ventajas

Existen variedad de ventajas sobre la utilización de los servidores de aplicaciones Oracle, para los objetivos a cubrir de este documento y el alcance del mismo, se puede mencionar las siguientes, aunque no se limitan a estas:

- Al centralizar la lógica de negocio en uno o varios servidores se puede garantizar las actualizaciones de las aplicaciones para todos los usuarios, por lo tanto no hay riesgo de que versiones antiguas accedan y manipulen la información en una manera desactualizada o incompatible.

- Configuración centralizada, los cambios a la configuración de la aplicación tal como cambiar el servidor de base de datos o los ajustes del sistema, se pueden realizar en un solo lugar.

- Seguridad, un punto central por medio del cual acceder a los datos y gestionar partes de la misma aplicación provee seguridad, manteniendo las responsabilidades de autenticación lejos de clientes potencialmente inseguros sin exponer la capa de la base de datos.

- Rendimiento, al limitar el tráfico de la red a el tráfico entre capas, se percibe que el modelo multicapa mejora el rendimiento de aplicaciones grandes que son utilizadas masivamente.

- Costo total de propiedad, TCO por sus siglas en inglés, en combinación las ventajas antes mencionadas representan ahorro en costos en las compañías cuando desarrollan aplicaciones empresariales.

1.5. Desventajas

Así como existen ventajas en el uso de servidores de aplicaciones Oracle, también existen principalmente tres desventajas que se detallan a continuación, para efectos de este documento y su alcance, pero que no se limitan a estas:

- Si los servidores se caen, los clientes pierden el acceso a las aplicaciones.

- La dependencia a la conexión a una red, incluyendo internet, expone la vulnerabilidad que cualquier falla en la red termine todas las conexiones a la aplicación.

- Finalmente la seguridad, ya que los datos que pueden ser delicados pueden ser transferidos a través de redes públicas.

Cabe mencionar que los primeros dos problemas se pueden atacar utilizando redundancia de recursos, por ejemplo servidores, conexiones de red e internet de contingencia, y el problema de seguridad se puede mitigar utilizando métodos de cifrado y conexiones seguras aplicadas en lugares donde exista interacción con información delicada. Aunque no hay una forma de evitar completamente estos inconvenientes si se puede reducir considerablemente el riesgo de que sucedan.

2. CARACTERÍSTICAS DE LOS SERVIDORES DE APLICACIONES ORACLE

2.1. Conceptos introductorios WebLogic

El funcionamiento de WebLogic se basa en dominios y cómo están configurados. Un dominio es una unidad básica de administración de un WebLogic Server, este consiste en una o más instancias de WebLogic Server con sus recursos asociados, que se manejan con un solo Administration Server. Una instancia siempre forma parte de un dominio y cada instancia puede ejecutarse en diferentes máquinas físicas. El Administration Server es el punto central desde el cual se configuran y manejan todos los recursos en un dominio. Usualmente se configura un dominio para incluir instancias adicionales de WebLogic Server llamadas Managed Servers. El despliegue de aplicaciones, servicios *web*, y otros, se hace en los Managed Servers, y el Administration Server se utiliza para configuración y administración únicamente.

El Administration Server opera como la entidad de control central para la configuración del dominio en su totalidad. Mantiene los documentos de configuración del dominio y distribuye los cambios en los documentos de configuración de los Managed Servers. También se puede utilizar como un lugar central para monitorear todos los recursos en un domino.

Los Managed Servers albergan aplicaciones de negocio, componentes de aplicaciones, servicios *web,* y otros, y los recursos asociados. Para optimizar el rendimiento estos mantienen una copia de solo lectura del documento de configuración del dominio. Cuando se inicia el Managed Server, se conecta con

el Administration Server correspondiente al dominio para sincronizar su copia del documento de configuración con la que mantiene el Administration Server. Para ambientes de producción, que requieren un alto rendimiento y disponibilidad se puede configurar para que los Managed Servers operen como *clusters*.

Un *cluster* es un grupo de múltiples instancias de WebLogic Server ejecutándose simultáneamente y trabajando juntas para proveer escalabilidad y confiabilidad. Un dominio puede tener múltiples *clusters* de WebLogic Server así como múltipes Managed Servers que no estén configurados como *clusters.* El Group Management Service (GSM) es un componente de infraestructura que está habilitado para las instancias en un *cluster y* permite que las instancias participen en un *cluster* detectando cambios en los miembros del *cluster* y notificando a todas las instancias de estos cambios*.*

Para facilitar el manejo de las instancias de un WebLogic Server distribuidas en varias máquinas existe el Node Manager, herramienta que permite iniciar, detener y reiniciar el Administration Server y los Managed Servers de forma remota. El proceso de Node Manager no está asociado directamente a un dominio pero puede administrar todo los dominios presentes en la máquina donde esté configurado. El Node Manager debe estar instalado en todas las máquinas en donde se encuentren las instancias que se desea controlar.

2.2. Conceptos introductorios GlassFish Server

Existen varios componentes de GlassFish que se organizan de cierta manera para proveer un ambiente completo de servidor de aplicaciones. Para iniciar se tiene una instancia que se le llama GlassFish Server ejecutándose en

una sola JVM. Esta proporciona las capacidades necesarias para habilitar el acceso a los clientes y el manejo de recursos. Un dominio es un grupo de uno o más instancias que se administran juntas. Una instancia siempre pertenece a un solo dominio y cada instancia dentro del dominio puede ejecutarse en diferentes máquinas físicas.

Cada dominio tiene un servidor de administración y uno o más instancias. El dominio mantiene su propia configuración, archivos de bitácoras y las áreas para el despliegue de aplicaciones. Desde la perspectiva de administración y configuración, un dominio representa un ambiente de ejecución completo de GlassFish Server que es responsable de albergar y administrar aplicaciones y recursos.

El Domain Administration Server, DAS de un dominio, es una instancia especial que se dedica a albergar aplicaciones administrativas. Este es responsable de la autenticación del administrador, de manejar las peticiones de administración, y comunicarse con las demás instancias en el dominio para realizar tareas administrativas. Este mantiene también un repositorio de la configuración y las aplicaciones desplegadas en el dominio. Cada instancia mantiene también una copia en su repositorio para que cuando llegue a fallar el DAS no se vea ningún impacto en el rendimiento o disponibilidad.

Un *cluster* es un grupo de instancias que comparten las mismas aplicaciones, recursos e información de configuración. Típicamente, los miembros que forman parte de un *cluster* se crean en diferentes lugares físicos para mejorar el rendimiento y la disponibilidad. Los *clusters* simplifican la administración de los servidores ya que solo se debe mantener una configuración y el despliegue de las aplicaciones no tiene que hacerse en cada instancia.

Dentro del dominio, también se encuentran los procesos llamados nodos agentes. Estos procesos se ejecutan en todas las máquinas físicas donde se ejecuten instancias de GlassFish Server incluyendo la máquina que alberga al DAS. Los nodo agente es el encargado de la creación e inicialización de instancias al recuperar la información del repositorio del DAS, iniciar, detener y reiniciar instancias, de sincronizar el repositorio de configuración local con el repositorio central del DAS entre otros.

2.3. Alta disponibilidad

Se dice que un sistema posee alta disponibilidad cuando su diseño e implementación asegura la posibilidad de tener una continuación operacional. Es decir, aún cuando llegara a ocurrir un incidente o falla no se llega a la interrupción del servicio para el usuario final.

2.3.1. WebLogic Server cluster

Para poder alcanzar los objetivos de un sistema que cuenta con alta disponibilidad, al utilizar WebLogic Server cluster se obtienen las capacidades mencionadas a continuación:

- *Application failover:* es cuando el componente de una aplicación que esté realizando algún trabajo en particular deja de estar disponible por cualquier razón, una copia del componente termina de ejecutar el trabajo pendiente. Para que el nuevo componente tome el lugar del componente fallido debe cumplirse lo siguiente:

    - Debe existir una copia del componente disponible.

- La información debe estar disponible para otros objetos y para el programa que maneja el *failover*, donde se define la localización y el estatus operacional de todos los objetos, para que pueda determinarse que el primer objeto falló antes de terminar su trabajo.

- También debe haber información disponible acerca del progreso de los trabajos que se encuentran en proceso de ejecución, para que el componente que asume el cargo de un trabajo interrumpido sepa la cantidad de trabajo que fue completado antes que el primer objeto fallara.

- WebLogic Server da soporte para migraciones manuales y automáticas de una instancia en *cluster* desde una máquina hacia otra. Un servidor administrado que pueda ser migrado se le llama servidor migrable. Esta característica está diseñada para ambientes con requerimientos de alta disponibilidad. Esta capacidad es útil para:

  - Asegurar disponibilidad ininterrumpida de servicios que solo pueden correr en una instancia a la vez en un momento determinado. Un servidor administrado configurado será migrado automáticamente a otra máquina en caso de fallas.

  - Facilita proceso de relocalización de un servidor administrado y todos los servicios que este alberga.

  El proceso de migración de un servidor traslada un servidor administrado en su totalidad, incluyendo dirección IP y las aplicaciones que se

encuentran en él, hacia un grupo de máquinas disponibles definidas anteriormente.

- Balanceo de carga: esto es la distribución nivelada de trabajos y comunicaciones a través de los recursos de redes y computación configurados. Para que se pueda dar el balanceo de carga se debe cumplir lo siguiente:

    - Debe haber múltiples copias de un componente que pueda realizar un trabajo en particular.

    - La información acerca de la localización y estatus operacional de todos los objetos debe estar disponible.

2.3.2. GlassFish Server cluster

Siguiendo con el proposito de un sistema con alta disponibilidad, el GlassFish Server *cluster* proporciona esta capacidad a través de protección contra fallos, escalabilidad y balanceo de carga. Estas se describen a continuación:

- Protección contra fallas: si una instancia o un anfitrión en un *cluster* fallan, GlassFish Server detecta la falla y recupera el estado de la sesión del usuario. Si un balanceador de carga está configurado para el *cluster*, el balanceador de carga redirige las peticiones de la instancia fallida a otras instancias en el *cluster*. Dado que las mismas aplicaciones y recursos están en todas las instancias del *cluster,* una instancia fallida puede reemplazarse por cualquier instancia en el cluster. Para permitir que el estado de la sesión de un usuario pueda ser recuperado, cada

instancia en el *cluster* envía datos de su estado a otra instancia, mientras estos datos de estado se actualizan en las instancias, los mismos datos son replicados.

- Escalabilidad: si existe la necesidad de aumentar la capacidad del sistema se pueden agregar más instancias al *cluster* sin que las funciones del servicio se vean afectadas y sin tener que interrumpirlo. Cuando se agrega o elimina una instancia, los cambios se manejan de forma automática.

- Balanceo de carga: si una instancia en un *cluster* es distribuida a través de diferentes anfitriones, la carga de trabajo puede ser distribuida entre estos anfitriones para incrementar el rendimiento del sistema en conjunto.

Tabla I. **Comparación de alta disponibilidad**

Característica	WebLogic Server	GlassFish Server
Protección contra fallas	X	X
Balanceo de carga	X	X
Escalabilidad	X	X
Soporte para migración	X	

Fuente: elaboración propia.

2.4. Seguridad

Desplegar, gestionar y mantener la seguridad es un reto muy grande para las organizaciones que proveen servicios nuevos a clientes usando la *web*. Para servir a una red mundial de usuarios *web* una organización TI debe atacar los problemas fundamentales de mantener confidencialidad, integridad y

disponibilidad del sistema y sus datos. Retos en seguridad incluyen todos los componentes del sistema, desde la red en sí a las máquinas individuales que la conforman. Seguridad a través de la infraestructura es un emprendimiento completo y continuo que requiere vigilancia así como políticas y procedimientos de seguridad establecidos y bien comunicados.

2.4.1. WebLogic Security Service

El uso de un WebLogic Server provee al sistema de operaciones fundamentales de seguridad las cuales se describen a continuación y se detalla cómo son manejadas por el WebLogic Server.

- Auditoría: es el proceso donde se guarda la información acerca de las peticiones al servidor y las respuestas que este envía a las peticiones. En la arquitectura de seguridad de WebLogic Server un proveedor de auditoría se utiliza para estos servicios. Si está configurado, el WebLogic Security *Framework* va a llamar al proveedor de auditoría antes y después que se realicen operaciones de seguridad como autenticación o autorización, cuando se realicen cambios a la configuración del dominio, o cuando algún recurso del dominio sea invocado.

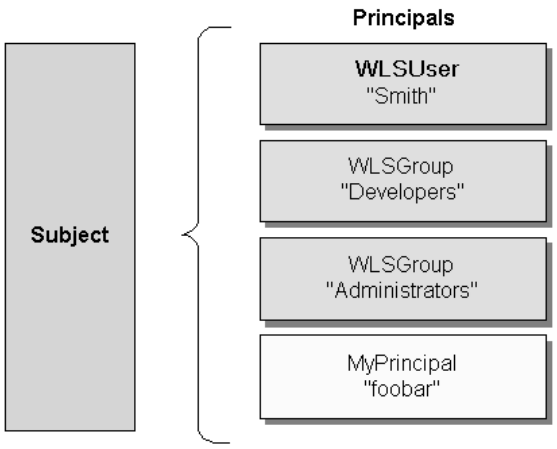
  La decisión de que eventos auditar la hace el mismo proveedor de auditoría y puede estar basado en criterios específicos y/o niveles de seguridad. Los registros que contienen la información pueden ser escritos a repositorios de salida como servidores LDAP, base de datos o un archivo simple.

- Autenticación: es el mecanismo mediante el cual los visitantes prueban que están actuando en nombre de un usuario o un sistema específico. La autenticación contesta la pregunta ¿Quién es usted? Utilizando credenciales como combinaciones contraseña/usuario. Un proveedor de autenticación es utilizado para verificar la identidad de usuarios o procesos del sistema.

  Los proveedores de autenticación también recuerdan, transportan y ponen a disposición la información de identidad a varios componentes del sistema cuando sea necesario. Durante el proceso de autenticación, el *Principal Validator* provee protección adicional para los principales (usuarios y grupos) dentro del sujeto al iniciar sesión y verificando la autenticidad de esos principales.

  Sujetos y principales, están muy cercamente relacionados. Un principal es una identidad asignada a un usuario o a un grupo como resultado de la autenticación. Ambos usuarios y grupos pueden ser utilizados como principales por los servidores de aplicaciones como WebLogic Server. En la figura 4 se puede observar la relación entre usuarios, grupos, principales y sujetos.

Figura 2. **Relación entre usuarios, grupos, principales y sujetos**



Fuente: Documentación oficial de Oracle.
docs.oracle.com/cd/E21764_01/web.1111/e13718/atn.htm.
Consulta: 5 de octubre de 2013.

- *Security Assertion Markup Language* (SAML): es un estándar que define un *framework* XML para crear, solicitar e intercambiar afirmaciones de seguridad entre entidades de software en la web. Este *framework* especifica cómo las afirmaciones y protocolos SAML se pueden utilizar para lo siguiente:

  - *Single sign-on* (SSO) entre socios del negocio.

  - El intercambio de información de identidad en servicios web de seguridad.

  WebLogic Server puede configurarse para actuar como proveedor SAML de identidad, de servicio o ambos, cuando actúa como proveedor de identidad, el proveedor de mapeo de credenciales SAML debe estar

configurado para que el proveedor de identificación pueda producir afirmaciones. Cuando actúa como proveedor de servicio debe estar configurado para consumir estas afirmaciones.

Los servicios de SAML *single sign-on* están configurados en servidores individuales, para habilitarlo en dos o más servidores en un dominio, como es el caso de *cluster*, se recomienda hacer lo siguiente:

- Crear un dominio en el cual la seguridad del almacenamiento RDBMS este configurado.

- Asegurarse que los servicios de SSO estén configurados individualmente e idénticamente en cada instancia del servicio.

- *Single Sign-On* (SSO): es la habilidad de requerirle a un usuario que inicie sesión en una aplicación únicamente una vez y obtener acceso a muchos componentes diferentes de la aplicación aun cuando estos componentes tengan sus propios esquemas de autenticación. Este le permite al usuario iniciar sesión de forma segura en todas sus aplicaciones y sitios *web* con una identidad. WebLogic Server provee SSO en los siguientes ambientes.

  - Navegadores *web* y clientes HTTP a través de SAML.

  - Clientes de escritorio

- Autorización: es el proceso por medio del cual las interacciones entre usuarios y recursos WebLogic son controlados, basados en la identidad del usuario. Este punto responde la pregunta ¿A qué puede usted

acceder? En WebLogic Server, un proveedor de autorización es utilizado para limitar las interacciones entre usuarios y recursos de WebLogic para asegurar integridad, confidencialidad y disponibilidad. Esto se realiza por medio de:

- Políticas de seguridad
- Manejadores de contexto
- Decisiones de acceso
- Adjudicaciones

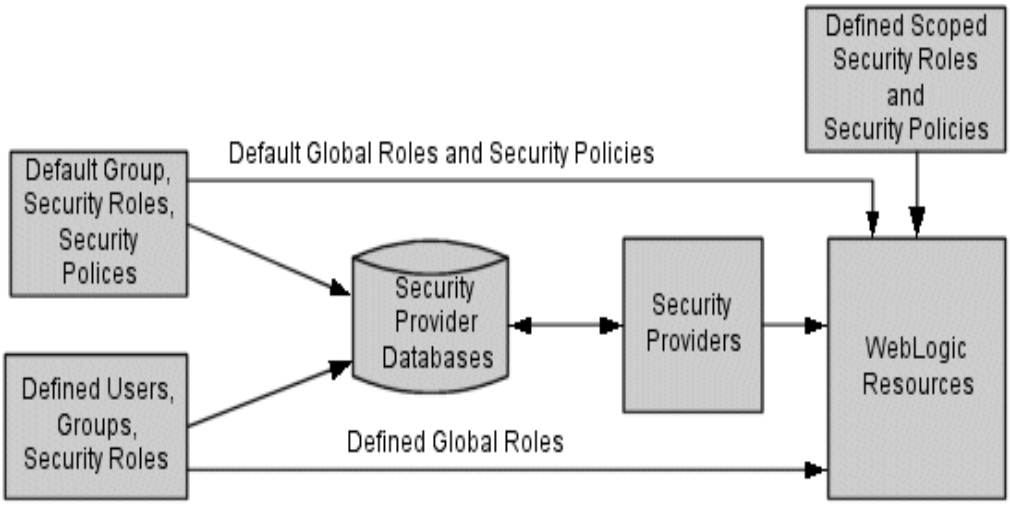
- Identidad y confianza: llaves privadas, certificados digitales y certificados de la autoridad certificadora de confianza, establecen y verifican identidad y confianza en el ambiente de WebLogic Server. La clave pública esta incrustada en un certificado digital. Una llave privada y un certificado digital proveen una identidad. Los certificados de la autoridad certificadora de confianza establecen la confiabilidad de un certificado. Cadenas de certificados y certificaciones deben ser validadas antes que una relación de confianza pueda ser establecida.

- *Secure Sockets Layer* (SSL): WebLogic Server soporta completamente comunicación a través de SSL, que permite comunicación segura entre aplicaciones conectadas a través de la *web.* WebLogic Server también incluye soporte para utilizar *Java Secure Socket Extension* (JSSE) como la pila de SSL para lo siguiente:

  - Conexiones SSL entrantes
  - Conexiones SSL salientes que usan las API's de WebLogic SSL.

Un concepto muy importante en la seguridad en WebLogic Server son los reinos de seguridad, estos abarcan mecanismos de seguridad para proteger los recursos de WebLogic. Cada reino de seguridad consiste en un grupo configurado de proveedores de seguridad, usuarios, grupos, roles y políticas de seguridad como se ve en la figura 5. Un usuario debe estar definido en un reino de seguridad para que pueda acceder a cualquier recurso de WebLogic que pertenezcan a este reino. Cuando un usuario trata de acceder a un recurso en particular, WebLogic Server trata de autenticar y autorizar al usuario, revisando el rol de seguridad asignado al mismo en el reino de seguridad y en la política de seguridad propia del recurso.

Figura 3. **Descripción del reino de seguridad de WebLogic Server**



Fuente: Documentación oficial de Oracle. docs.oracle.com/cd/E23943_01/web.1111/e13710/realm_chap.htm#i1033824. Consulta: 5 de octubre de 2013.

Los usuarios son entidades que pueden ser autenticadas en un reino de seguridad. Un usuario puede ser una persona, como una aplicación de usuario,

una entidad de software como una aplicación cliente u otra instancia de WebLogic Server. Como resultado de la autenticación, un usuario es asignado a una identidad. A cada usuario se le da una identidad única dentro del reino de seguridad. Los usuarios pueden ser colocados dentro de grupos que son asociados con roles de seguridad, o pueden ser directamente asociados con estos roles.

Los grupos son grupos de usuarios ordenados lógicamente. La mayoría de las veces los miembros de los grupos tienen algo en común. Por ejemplo, una empresa puede separar al equipo de ventas en dos grupos, agentes de ventas y gerentes de ventas. Esto se hace para que cada grupo pueda tener diferentes niveles de acceso a los recursos de WebLogic dependiendo en las funciones de su trabajo.

Un rol de seguridad es un privilegio concedido a los usuarios o grupos basados en condiciones específicas. Como los grupos, los roles de seguridad permiten la restricción de acceso a los recursos de WebLogic para varios usuarios a la vez, pero a diferencia de los grupos, los roles de seguridad:

- Son computarizados y concedidos a los usuarios o grupos de forma dinámica basados en condiciones como el nombre de usuario, la pertenencia a un grupo o la hora del día.

- Puede configurarse para que abarque recursos específicos dentro de una aplicación en un dominio de WebLogic Server (a diferencia de los grupos que siempre están configurados para todo el dominio).

Una política de seguridad es una asociación entre un recurso de WebLogic y uno o más usuarios, grupos o roles de seguridad. Las políticas de

seguridad protegen los recursos contra accesos no autorizados. Los recursos no tienen protección hasta que se cree una política de seguridad para ellos. Una condición de una política es una condición bajo la cual se creará una política de seguridad. En el editor de políticas se incluyen condiciones de fecha y hora.

Los proveedores de seguridad son módulos que proveen servicios de seguridad a las aplicaciones para proteger los recursos WebLogic. Se puede utilizar proveedores de seguridad como parte del producto de WebLogic Server, adquirir proveedores de seguridad personalizados de terceros o desarrollar su propio proveedor de seguridad.

De los proveedores de seguridad más comunes está el de base de datos, este contiene usuarios, grupos, roles de seguridad, políticas de seguridad y credenciales utilizadas por algunos tipos de proveedores de seguridad para dar estos servicios. Por ejemplo, un proveedor de autenticación requiere información acerca de usuarios y grupos, un proveedor de autorización requiere información acerca de las políticas de seguridad, otros. Estos proveedores de seguridad necesitan que está información esté disponible en una base de datos para funcionar correctamente.

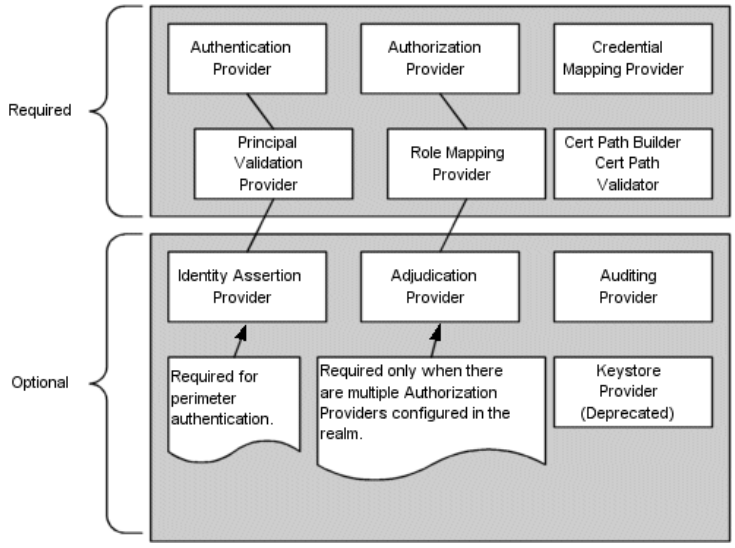
El proveedor de seguridad de base de datos puede ser el servidor incrustado de LDAP, un archivo de propiedades, o una base de datos que se esté utilizando previamente. Los proveedores de seguridad que se pueden utilizar en un reino de seguridad son los siguientes:

- Autenticación
- Afirmación de identidad

- Validaciones de principales
- Autorización
- Adjudicación
- Mapeo de roles
- Auditoría
- Mapeo de credenciales
- Búsqueda de certificados y validación
- Almacén de claves
- Adaptador de reinos de seguridad

Concretamente, estos tipos de proveedores de seguridad interactúan en WebLogic Server como se ve en la figura 4.

Figura 4. **Proveedores de seguridad de WebLogic Server**



Fuente: Documentación oficial de Oracle. docs.oracle.com/cd/E23943_01/web.1111/e13710/realm_chap.htm#i1033824. Consulta: 5 de octubre de 2013.

2.4.2. GlassFish Server System Security

Seguridad se trata de proteger información, esto es, como evitar accesos no autorizados o daños a la información mientras está almacenada o es transferida. GlassFish Server está construido bajo el modelo de seguridad de Java, que utiliza un *sandbox* donde las aplicaciones pueden ejecutarse de manera segura sin riesgos potenciales al sistema o a los usuarios. El sistema de seguridad afecta a todas las aplicaciones en el ambiente de GlassFish Server. System Security incluye las siguientes características:

- Autenticación: es la forma en que una entidad (usuario, aplicación o componente) determina que otra entidad es quien dice ser. Una entidad utiliza credenciales de seguridad para autenticarse. Las credenciales pueden ser nombre de usuario y contraseña, certificados digitales u otros. Usualmente servidores o aplicaciones requieren clientes para autenticarse ellos mismos. Adicionalmente, puede que los clientes requieran servidores para autenticarse. Cuando la autenticación es bidireccional, es llamada autenticación mutua.

  Cuando una entidad trata de acceder a un recurso protegido, GlassFish Server utiliza el mecanismo de autenticación configurado para ese recurso, para determinar si debe o no permitir el acceso. Por ejemplo, un usuario puede ingresar su nombre de usuario y contraseña en un navegador *web*, y si la aplicación verifica esas credenciales, el usuario es autenticado. El usuario es asociado con esta identidad autenticada por el resto de la sesión.

  Los tipos de autenticación que son soportados por GlassFish Server son:

- Básico, utiliza el cuadro de dialogo de inicio de sesión propio del servidor. El protocolo de comunicación es HTTP (SSL es opcional). No hay encriptación de credenciales de usuario al menos que se utilice SSL. Este tipo no es considerado seguro a menos que sea utilizado con sistemas externos de seguridad como SSL.

- *Form*, la aplicación provee de su propia página de inicio de sesión y errores personalizada. Lo mismo que el tipo anterior, no hay encriptación de credenciales a menos que se utilice SSL y el protocolo de comunicación es HTTP.

- Cert-Cliente, el servidor autentica al cliente usando un certificado de llave pública. El protocolo de comunicación es HTTPS. La encriptación de credenciales es SSL.

- *Digest,* el servidor autentica un usuario basándose en su nombre y contraseña. A diferencia de la autenticación Básica, la contraseña nunca se envía a través de la red. El uso de SSL es opcional.

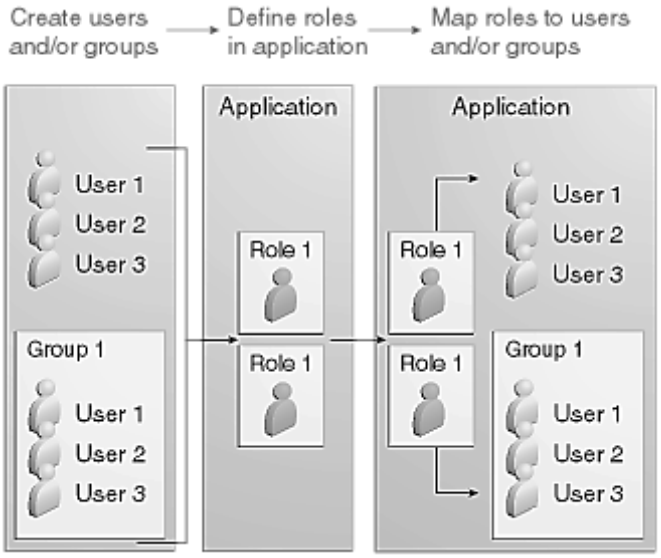
- Autorización: la autorización también es conocida como control de acceso, son los medios mediante los cuales se les da permiso a los usuarios para acceder a información o realizar operaciones. Después que un usuario es autenticado, el nivel de autorización del mismo determina que operaciones puede realizar. La autorización está basada en el rol del usuario.

  Un rol define a que aplicación y que partes de esa aplicación los usuarios pueden acceder y que cosas pueden hacer los usuarios o grupos con

esas aplicaciones. Un rol es diferente de un grupo en que un rol defina una función en una aplicación mientras que un grupo es un conjunto de usuarios que están relacionados de alguna forma. Por ejemplo, los empleados pueden pertenecer a grupos según su jornada, tiempo completo o medio tiempo. Todos los usuarios en estos grupos tienen el rol de empleados.

Los roles son definidos en el descriptor de despliegue para la aplicación. El desarrollador de la aplicación o el que la despliegue, mapea los roles a uno o más grupos en el descriptor para cada aplicación. Cuando la aplicación está siendo empaquetada y desplegada, esta especifica cómo es la relación entre usuarios, grupos y roles, como se ve en la figura 5.

Figura 5. **Mapeo de roles**



Fuente: Documentación oficial de Oracle. docs.oracle.com/cd/E26576_01/doc.312/e24940/system-security.htm#ablnx. Consulta: 5 de octubre de 2013.

- Auditoría: es el medio utilizado para capturar eventos relacionados con seguridad con el propósito de evaluar la efectividad de las medidas de seguridad. GlassFish Server utiliza módulos de auditoría para capturar los rastros de todas las decisiones de autorización y autenticación. GlassFish Server provee un módulo de auditoría predeterminado así como la capacidad de conectarse con módulos de auditoría personalizados. El alcance del módulo de auditoría es el servidor completo, esto significa que todas las aplicaciones alojadas en el servidor utilizarán el mismo módulo de auditoría.

- *Secure Sockets Layer* (SSL): es el estándar más popular para asegurar las comunicaciones y transacciones a través de internet. Las aplicaciones *web* seguras utilizan HTTPS. Este utiliza certificados para asegurar la comunicación segura y confidencial entre servidores y clientes. En una conexión SSL, ambos cliente y servidor deben cifrar información antes de enviarla y se descifra al recibirla.

Tabla II. **Comparación de seguridad**

Características	**WebLogic**	**GlassFish**
Autenticación	X	X
Auditoría	X	X
Security Assertion Markup Language (SAML)	X	
Single Sign-On (SSO)	X	
Autorización	X	X
Identidad y confianza	X	
Secure Sockets Layer (SSL)	X	X
Proveedores de Seguridad	X	

Fuente: elaboración propia.

2.5. Herramientas de administración

En este apartado se van a definir las herramientas con las que cuenta cada uno de los servidores de aplicaciones para ayudar al usuario con las tareas de administración relacionadas con cada uno.

2.5.1. Herramientas de WebLogic

Entre las herramientas de administración correspondientes a WebLogic Server están la más utilizada que es la consola de administración, la herramienta de WebLogic *scripting*, el asistente de configuración, el *Configuration Template Builder*, agentes SNMP y tareas de apache ANT, estas se detallan a continuación:

- La Consola de administración es una aplicación web ejecutada por el Servidor de Administración. Se utiliza para gestionar y monitorear un dominio activo. Un dominio de WebLogic Server está relacionado con un grupo de recursos que se gestionan como un sistema. Un dominio incluye uno o más WebLogic Servers y puede también incluir *clusters.* Una instancia de WebLogic en cada dominio es configurada como un Servidor de Administración, este servidor provee un punto central para gestionar todo el dominio y es el que alberga la consola de administración, que es una aplicación web accesible desde cualquier navegador web que tenga acceso a la red del Servidor de Administración.

  Entre las capacidades de gestión de la consola están:

  - Configurar, iniciar y detener instancias de WebLogic Server

- Configurar dominios activos
- Configurar WebLogic Server *clusters*
- Configurar servicios de WebLogic Server tal como conectividad a base de datos y mensajería
- Monitorear la salud y rendimiento de los servidores
- Monitorear el rendimiento de las aplicaciones
- Visualizar la bitácora del servidor
- Configurar parámetros de seguridad, incluyendo manejo de usuarios, grupos y roles
- Configurar y desplegar las aplicaciones
- Visualizar los descriptores de despliegue de las aplicaciones
- Editar los elementos de los descriptores de una aplicación en ejecución seleccionados

A través de la consola de administración, los administradores del sistema pueden realizar todas las tareas de administración del WebLogic Server sin tener que aprender el api JMX o su arquitectura de administración. El servidor de administración vuelve los cambios persistentes mediante atributos en el archivo config.xml para el dominio que se está manejando.

- Herramienta de Weblogic *scripting*: la WLST (WebLogic *Scripting Tool*) es una interfaz de línea de comando que se utiliza para gestionar y monitorear dominios de WebLogic Server activos o inactivos. El ambiente WLST está basado en el intérprete Java Jython, además de las funciones de *scripting,* se puede usar las características comunes de lenguajes interpretados incluyendo variables locales, variables condicionales y declaraciones de flujos de control.

- Asistente de configuración: este asistente crea la estructura de directorios apropiada para un dominio de WebLogic Server, el archivo config.xml, y los *scripts* que se pueden utilizar para iniciar servidores en el dominio. El asistente también utiliza plantillas para crear dominios y estas plantillas se pueden personalizar para duplicar los dominios. También se puede utilizar el asistente para agregar o quitar servicios de un dominio existente e inactivo.

  Se puede ejecutar el Asistente de Configuración a través de una interfaz gráfica de usuario GUI o en un ambiente de línea de comandos. Esta línea de comandos es llamada también modo consola. También se puede crear plantillas de dominios definidos por el usuario para que sean utilizadas por este asistente.

- *Configuration Template Builder:* provee la capacidad de crear fácilmente sus propias plantillas de dominio para permitir, por ejemplo, la definición y propagación de un dominio estándar a través del desarrollo de un proyecto. O para permitir la distribución de un dominio junto con una aplicación que ha sido desarrollada para ser ejecutada en ese dominio. Las plantillas que sean creadas con esta herramienta, son utilizadas como entradas para el Asistente de configuración como la base para recrear un dominio que es personalizado para un ambiente.

- Tareas Apache Ant: se pueden utilizar dos tareas Ant previstas con el WebLogic Server para ayudar a realizar tareas de configuración comunes en un ambiente de desarrollo. Ant es una herramienta basada en Java. Las tareas de configuración permiten iniciar y detener una instancia de WebLogic Server así como la creación y configuración de dominios. Cuando se combina con otras tareas, se pueden crear *scripts* de

construcción muy poderosos para demostrar o probar las aplicaciones con dominios personalizados.

- Agentes SNMP: WebLogic Server incluye la capacidad de comunicarse con sistemas de gestión utilizando *Simple Network Management Protocol* (SNMP). Los agentes SNMP de WebLogic Server le permiten integrar las gestiones de WebLogic Servers en un sistema de gestión SNMP que le da una vista simple de varios recursos de software y hardware de sistemas complejos y distribuidos.

2.5.2. Herramientas de GlassFish

Entre las herramientas de administración de GlassFish Server se encuentra la Consola de Administración, la herramienta asmadmin, las interfaces REST y la herramienta de actualización, las cuales se detallen a continuación:

- La consola de administración es una utilidad basada en un navegador que presenta una interfaz gráfica fácil de navegar que incluye una ayuda muy extensa en línea para las tareas administrativas. Para utilizar la consola de administración se debe estar ejecutando el DAS. Cada dominio tiene su propio DAS que tiene un número de puerto único. Cuando se instala GlassFish Server se elige el número de puerto para el DAS o utilizar el puerto predeterminado que es el 4848. También se especifica un nombre de usuario y una contraseña si no se acepta el predeterminado que es admin sin contraseña.

  Cuando se especifique la URL para la consola de administración, se debe colocar el puerto del dominio que será administrado. Si se está

ejecutando en la máquina donde se instaló GlassFish Server, el nombre de dominio es *localhost.* Si la consola de administración se ejecuta en una máquina diferente de la instalación se debe utilizar una conexión segura HTTPS. Para Microsoft Windows, una forma alternativa de iniciar la consola de administración de GlassFish Server es por medio del menú de inicio. Para mostrar el material de ayuda en la página de la consola administración, basta con hacer clic en el botón de ayuda.

- Asmadmin: es una herramienta de línea de comando que ejecuta sub comandos para identificar la operación o tarea que se desea realizar. Se puede ejecutar desde una ventana de comando o un script. Esta herramienta es útil para tareas automáticas y repetitivas. Para ejecutar un sub comando en la consola se debe ir al directorio /bin y escribir asmadmin seguido del sub comando.

  Se puede invocar el modo múltiple al escribir únicamente asmadmin, se siguen aceptando sub comandos hasta que se salga de este modo y se regrese a la consola estándar.

- Interfaces REST: GlassFish Server provee interfaces REST (*Representational State Transfer)* para permitir el acceso y configuración de datos para GlassFish Server, incluyendo datos que son provistos por componentes recién instalados.

- Herramienta de actualización: GlassFish Server provee de un conjunto de sistemas de empaquetado de imágenes (IPS) para actualización de software en un GlassFish Server que ya tiene aplicaciones desplegadas. Actualizaciones típicas incluyen nuevos lanzamientos de GlassFish Server y nuevos componentes o módulos del mismo. La herramienta de

actualización es una utilidad que puede ser ejecutada en la consola de administración o invocada desde la línea de comandos.

Se puede usar cualquier herramienta para agregar componentes, pero para actualizar o eliminar componentes existentes se debe usar la versión independiente. Instrucciones de cómo usar la herramienta de actualización están en la ayuda de la consola de administración o en la ayuda de la herramienta de actualización independiente. El comando pkg es la versión de consola de la herramienta de actualización.

Tabla III. **Comparación de herramientas de administración**

WebLogic Server	GlassFish Server
Consola de administración (/console)	Consola de administración
Asistente de configuración (config.sh)	Herramienta de línea de comandos (asmadmin)
Configuration Template Builder (config_builder.sh)	Herramienta de actualización (updatetool)
Tareas Apache Ant	Interfaces REST
Agentes SNMP	
Herramienta de Weblogic *scripting (wlst.sh)*	

Fuente: elaboración propia.

2.6. Estándares soportados

A continuación se lista una comparativa de los estándares que son soportados por cada uno de los servidores. Lo que indica que se pueden utilizar estos estándares en la configuración y en las aplicaciones y que estas funcionen correctamente.

Tabla IV. **Estándares soportados por WebLogic y GlassFish Server**

Estándar	WebLogic	GlassFish
Java API for XML-Based Web Services (JAX-WS)	x	x
Java Authorization Contract for Containers (JACC)	x	x
Java Platform Enterprise Edition	x	x
Java EE Application Deployment	x	x
Java EE Connector Architecture	x	
Java EE Enterprise Java Beans (EJB)	x	x
Java EE Enterprise Web Services	x	x
Java DataBase Conectivity (JDBC)	x	x
Java Message Service (JSM)	x	x
Java Naming and Directory Interface (JNDI)	x	x
Java Server Faces (JSF)	x	x
Java Server Pages (JSP)	x	x
Java EE Servlet	x	x
Java Remote Method Invocation (RMI)	x	
JavaMail	x	x
Java Architecture for XML binding (JAX-B)	x	x
Java Architecture for XML processing (JAX-P)	x	
Java API for XML Registries (JAX-R)	x	x
Java API for XML Based Remote Procedure Calls (JAX-RPC)	x	x
Java Cryptography Extension (JCE)	x	
Java Development Kit (JDK)	x	x
Java Management eXtensions (JMX)	x	x
Java Persistence API (JPA)	x	x
Java EE Management	x	x
JavaServer Pages Standard Tag Library (JSTL)	x	x
Java Transaction API (JTA)	x	x
SOAP Attachments for Java (SAAJ)	x	
Streaming API for XML (StAX)	x	
Web Services Metadata for the Java Platform	x	x
Secure Sockets Layer (SSL)	x	x
Java API for RESTful Web Service		x
Lightweight Directory Access Protocol (LDAP)	x	x
Transport Layer Security (TLS)	x	x
Hypertext Transfer Protocol (HTTP)	x	x
Simple Network Management Protocol (SNMP)	x	x
xTensible Access Control Markup Language (XACML)	x	
Context and Dependency Injection for Java (CDI)	x	x
Expression Language (EL)	x	x
Java authorization and authentication service (JAAS)	X	X
Managed Beans	x	x
Interceptors	x	x
Bean Validation	x	x

Fuente: elaboración propia.

3. ADMINISTRACIÓN DE SERVIDORES DE APLICACIONES ORACLE

3.1. Tareas comunes de administración de WebLogic Server

A continuación se muestran las tareas básicas que se deben realizar para asegurar el buen funcionamiento del WebLogic Server y de las aplicaciones que se encuentren alojadas en este.

3.1.1. Instalación

Para poder completar la instalación básica del software de WebLogic Server se deben seguir seis pasos, al terminar estos pasos se procede a configurar las características que se desea utilizar del WebLogic Server como se muestra a continuación:

Paso 1: obtener el archivo de instalación apropiado para la plataforma a utilizar. Entre los tipos de archivos que podemos escoger están los siguientes:

- Paquete de instalación específico para el sistema operativo: este tipo de instalador es una versión autónoma del programa de instalación que incluye el JDK para la plataforma seleccionada. El instalador puede ser un archivo .exe o .bin dependiendo de la plataforma seleccionada.

- Paquete de instalación genérico: este tipo de instalador es un archivo .jar. Este no incluye el JDK. Se puede utilizar este instalador para

instalar el producto en cualquier plataforma que sea soportada en la cual Java ya se encuentre instalado.

- Instalador de actualización: este permite actualizar una instalación ya existente de WebLogic Server al último parche liberado. Por ejemplo, si se tiene la versión 10.3.0 se puede actualizar a la versión 10.3.6 con este tipo de paquete. Dependiendo de la plataforma este instalador puede ser específico para un sistema operativo o puede ser genérico.

- Solo para desarrollo e instaladores suplementarios: el instalador solo para desarrollo, es un archivo .zip que se debe extraer en el directorio base de Middleware. Este contiene una instalación de WebLogic Server que incluye todos los componentes necesarios para desarrollo, este no debe utilizarse para un ambiente en producción. Un archivo .zip está disponible también para proveer de características adicionales como ejemplos.

Paso 2: completar los pre-requisitos para iniciar la instalación.

- Configuración de la plataforma: debe haber una configuración de hardware, sistema operativo, JDK y base de datos específico para el producto que se está instalando.

- Procesador y memoria RAM: de procesador se recomienda tener de mínimo 1-GHz. De memoria RAM se debe tener un mínimo de 1GB aunque el recomendado para un rendimiento ideal es 2GB.

- Disco duro: una instalación completa incluyendo SDKs requiere aproximadamente 3.9GB de espacio libre en disco duro. Esto incluye el

espacio de disco temporal que es necesario durante la instalación. Dependiendo de los componentes que se escojan para instalar y el instalador que se esté utilizando, puede necesitarse menos espacio.

- Colores de fondo: para una instalación gráfica, se requieren 8-bits de colores (256 colores), si la instalación es en modo consola o en modo silencioso, no hay un requerimiento de colores.

- JDK: el programa de instalación requiere *Java Runtime Environment* (JRE) para poder ejecutarse. Este está incluido en los programas de instalación de Windows de 32-bit, Linux x86 y algunos Unix que tienen la extensión .bin. Para otras plataformas el programa de instalación no incluye el JDK. Para ejecutar los programas de instalación con extensión .jar, se debe tener la versión apropiada de JDK instalada en el sistema y se debe incluir el directorio /bin en las variables de entorno.

- Usuario de instalación de WebLogic Server: el usuario a utilizar cuando se realice la instalación es muy importante. Cuando se haga la instalación en un sistema operativo Unix o Linux no se debe ejecutar como usuario *root.* Cuando se está haciendo una instalación de actualización, se debe ejecutar el programa con el mismo usuario que fue utilizado para la instalación inicial de WebLogic Server.

- Espacio de disco temporal: durante la instalación, el programa utiliza un directorio temporal en el que extrae los archivos necesarios para la instalación. Este directorio requiere aproximadamente 2,5 veces el espacio de disco requerido para la instalación. Para crear el directorio temporal se hace como se detalla a continuación.

En la línea de comando se debe ejecutar la opción -Djava.io.tmpdir= rutaTMPdir, en este caso el rutaTMPdir es el directorio que estamos designando como área temporal para el programa de instalación pero la carpeta no está creada.

Paso 3: determinar el modo de instalación apropiado.

- Modo gráfico: es un método interactivo basado en una GUI para la instalación del software. Puede ejecutarse en sistemas Windows o Unix.

- Modo consola: es un modo interactivo basado en texto para instalar el software desde la línea de comando también en sistemas Windows o Unix.

- Modo silencioso: este es un modo no interactivo, se puede utilizar las propiedades de un archivo XML para especificar las opciones de instalación. Se puede ejecutar el modo de instalación silencioso desde un *script* o desde la línea de comando. Este modo permite definir la configuración de instalación solo una vez y luego usar esa configuración para duplicar la instalación en varias máquinas.

Paso 4: determinar el tipo de instalación.

Cuando se realiza una instalación en modo gráfico o modo consola, el programa provee dos tipos de instalación, típica y personalizada. En una instalación típica, la mayoría de los componentes incluidos en el programa son instalados. En una instalación personalizada, se puede escoger los componentes que se quiere instalar.

Paso 5: instalar el Software.

En este paso se especificará la instalación en modo gráfico por medio de un archivo ejecutable que despliega un asistente de instalación. Para iniciar el proceso de instalación con el archivo de extensión .bin.

Paso 6: crear un dominio WebLogic.

Luego de la instalación de WebLogic Server se puede crear el dominio de varias formas:

- Utilizar el asistente de configuración para crear un dominio utilizando una o más plantillas de dominio, se puede utilizar la plantilla para principiantes wls_starter.jar que se encuentra en la carpeta MW_HOME/common/templates/domains, que contiene la configuración predeterminada y una aplicación que tiene un página de bienvenida para iniciar.
- Utilizar la herramienta de *Scripting* de WebLogic.
- Utilizar el comando *unpack* para crear el dominio desde la línea de comando utilizando una plantilla que sea compatible con la instalación actual.

3.1.2. Inicio y detención de servidores

WebLogic provee varias formas para iniciar y detener instancias de servidores, el método a elegir depende en si se prefiere utilizar la línea de comando o el modo gráfico. Se puede iniciar una instancia en cualquier computadora que tenga instalado WebLogic Server. La configuración del servidor se encuentra en el archivo de dominio config.xml que está en la

computadora que alberga el Servidor de Administración. La primera instancia en iniciar asume el rol de administrador y los demás el rol de instancia administrada.

- Iniciar el servidor de administración: para iniciar el servidor de administración hay un script pre-definido que se recomienda utilizar.

    - Iniciar una consola en la computadora que se creó el dominio.

    Cambiar el directorio al directorio en que se encuentra el dominio. De forma predefinida el directorio es *MW_HOME/user_projects/domains/ NOMBRE_DOMINIO.* Donde MW_HOME se refiere a la ruta del directorio raíz de WebLogic Server.

    - Ejecutar el siguiente script *bin/startWeblogic.sh*

    Lo que hace el script es establecer las variables de entorno invocando el script *MW_HOME /user_projects /domains /NOMBRE_DOMINIO /bin /setDomainEnv.sh.* Luego invoca el comando java weblogic.server que inicia la JVM que esta configurada para ejecutar la instancia de WebLogic Server. Cuando el servidor completa el proceso de inicio escribe el siguiente mensaje en la ventana de comandos:

    <Notice> <WebLogicServer> <BEA-000360> <Server started in RUNNING mode>

- Iniciar un servidor administrado: un servidor administrado es el que ejecuta las aplicaciones desplegadas. Este se refiere al servidor de administración para obtener toda su información de despliegue y

configuración. Una de las formas de iniciar los servidores administrados es utilizando la consola de administración, para esto debe estar configurado el Node Manager que se mencionó anteriormente y se hace de la siguiente manera:

- En el panel izquierdo de la consola, expandir el apartado *Environment* y seleccionar *Servers.*
- En la tabla de servidores hacer clic en la instancia que se desee iniciar.
- Seleccionar el apartado Control > *Start/Stop.*
- En la tabla de Status del Servidor, seleccionar el nombre del servidor servidor que se quiere iniciar y dar clic en *Start*.
- En la página del Asistente de ciclo de vida del servidor hacer clic en *si* para confirmar.

- Detener un servidor: para detener un servidor, sea este el administrador o un administrado, la manera más fácil de hacerlo es por medio de la consola de administración, teniendo en cuenta que una vez se apague el servidor administrador la consola de administración ya no estará disponible, por lo que este debe ser el último servidor en ser detenido.

  - En el panel izquierdo expandir el apartado *Environment* y seleccionar *Servers.*

- En la tabla *Servers* dar clic en el nombre del servidor que se desea detener.

- Seleccionar el apartado *Control > Start / Stop}*

- En la tabla de *Status* del servidor seleccionar el servidor que se desea detener.

- En el menú desplegable de Apagado se debe seleccionar alguna de las siguientes opciones:

  - Cuando se finalice el trabajo, este comando inicia el apagado que le da tiempo a los subsistemas de WebLogic Server para completar ciertas aplicaciones que estén actualmente en progreso.

  - Forzar el apagado ahora, este comando inicia un apagado forzoso en el cual el servidor da la instrucción a los subsistemas de abandonar inmediatamente las peticiones y apagarse.

- Hacer clic en *yes* para confirmar el apagado del servidor.

3.1.3. Configurar las conexiones a servidores web

Se puede conectar peticiones de servidores *web* a una instancia de WebLogic Server o a un *cluster* utilizando alguno de los conectores disponibles para ello. Existen conectores para los siguientes servidores *web:*

- Sun One Web Server o IPlanet
- Microsoft IIS
- Apache

Debido a que estos conectores operan en el ambiente nativo del servidor *web,* se pueden gestionar desde las herramientas de administración de ese servidor *web*. Estos conectores son módulos que se agregan en la instalación del servidor *web* para habilitar la comunicación entre WebLogic Server y aplicaciones ejecutándose en alguno de los servidores mencionados anteriormente.

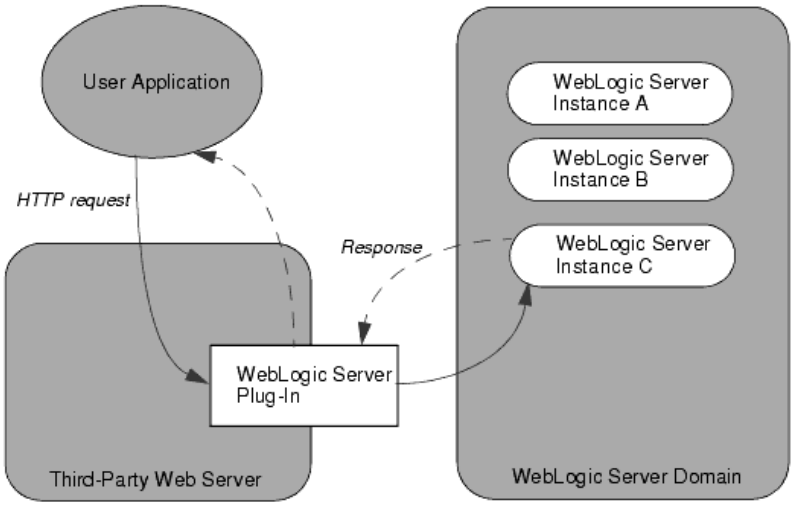
Los conectores usualmente utilizan alguna de las siguientes arquitecturas para permitir que las aplicaciones ejecutándose en otro servidor web usen las capacidades de WebLogic Server:

- Como un balanceador de carga para el servidor *web*.

- Para conectar peticiones de contenido dinámico (los servidores *web* únicamente pueden suplir contenido estático y se apoyan en WebLogic Server para el contenido dinámico).

En un escenario típico, un servidor *web* externo, recibe la petición de una aplicación de usuario y encamina la petición HTTP al conector con WebLogic Server que usa su configuración para enviar la petición a la instancia que procesa la petición y luego envía la respuesta al conector y luego al servidor *web* como se muestra en la figura 6.

Figura 6. **Funcionamiento del Conector de WebLogic para servidores web**



Fuente: Documentación oficial de Oracle. docs.oracle.com/cd/E13222_01/wls/docs81/plugins/overview.html#1035685. Consulta: 12 de octubre de 2013.

3.1.4. Configurar las conexiones a bases de datos

En WebLogic Server, se puede configurar la conectividad a bases de datos por medio de la configuración de *data sources* JDBC. Cada *data source* contiene un *pool* de conexiones a base de datos que se crean cuando la instancia del *data source* se crea. Las aplicaciones buscan al *data source* en el contexto local de la aplicación y luego solicita una conexión. Cuando termina con la conexión, la aplicación llama al método *connection.close* que devuelve la conexión al *pool* de conexiones en el *data source.*

- *Pool* de conexiones: es un grupo de conexiones JDBC hacia una base de datos que se crean cuando el *pool* de conexiones es desplegado ya sea al inicio de WebLogic Server o de manera dinámica en tiempo de ejecución. La aplicación toma prestada una conexión del *pool* y la regresa luego de cerrar la conexión.

- *Data sources*: proveen acceso al *pool* de conexiones para conectividad con la base de datos. Cada *data source* hace referencia a un *pool* de conexiones, pero se pueden definir múltiples *data source* que hagan referencia a una única conexión. WebLogic soporta dos tipos de *data source,* el *DataSource* para transacciones locales y el *TxDataSource* para transacciones distribuidas, estas últimas se recomienda utilizarlas cuando la aplicación utiliza el API de transacciones de Java (JTA), el EJB de WebLogic Server para el manejo de transacciones, y si se actualizan múltiples bases de datos en una misma transacción.

3.1.5. Configurar servicios de mensajería

WebLogic Server utiliza el servicio de mensajería de Java al que se refiere de ahora en adelante como JMS (Java Message Server*)*. JSM es un sistema de mensajería que permite a las aplicaciones comunicarse de forma asíncrona entre ellas a través del intercambio de mensajes. Un mensaje es una petición, reporte o evento que contiene información necesaria para coordinar la comunicación entre diferentes aplicaciones. Un mensaje provee un nivel de abstracción permitiendo la separación de los detalles del sistema destino y del código de la aplicación.

JSM acepta mensajes de aplicaciones productoras y los entrega a aplicaciones consumidoras. Los componentes más grandes de JSM incluyen:

- Un servidor JSM es una entidad configurada en el ambiente que actúa como contenedor de gestiones para la cola JSM y recursos de interés. Este es el principal responsable de que sus destinos mantengan la información en el almacenamiento persistente que se utilice, para que cualquier mensaje persistente que llegue a su destino, y mantener los estados de los suscriptores en esos destinos. Se puede configurar uno o más servidores JSM en un dominio y un servidor JSM puede manejar uno o más módulos JSM.

- Un módulo JSM contiene recursos de configuración, como una cola, un tema, destinos distribuidos y fábricas de conexiones, que son definidos por documentos XML, que conforman el esquema weblogic-jms.xsd.

- Aplicaciones cliente JMS que producen mensajes a los destinos o consumen mensajes de los destinos. JNDI, que provee un sistema de búsqueda y por último un almacenamiento persistente para archivar datos de los mensajes.

Para configurar los componentes de JMS se utiliza la consola de administración, siguiendo los pasos descritos a continuación:

- Si se requiere que los mensajes se archiven, se puede utilizar cualquiera de las siguientes opciones de almacenamiento:

    - Se puede utilizar un almacén basado en archivos utilizando el almacenamiento predeterminado del servidor o se puede crear uno personalizado.

- Se puede almacenar también en una base de datos accesible por JDBC.

- Configurar un servidor JMS para manejar los mensajes que lleguen a la cola y los destinos de interés en un módulo JSM.

- Configurar un módulo JSM para contener a los destinos, y otros recursos como cuotas, plantillas, llaves de destino, destinos distribuidos y fábricas de conexiones.

- Antes de crear las colas en los módulos, opcionalmente, se puede crear otros recursos JSM en el módulo que puedan ser referenciados dentro de esta cola, como plantillas, cuotas. Una vez estos recursos estén configurados, se pueden seleccionar al crear la cola u otros recursos.

- Configurar una cola o un destino de interés en el módulo JSM :

  - Configurar un destino independiente para el envío de mensajes con múltiples receptores (publicaciones o suscripciones).

  - Configurar una cola independiente para el envío de mensajes a un receptor en específico, exactamente uno (punto a punto).

- Si las fábricas de conexiones que provee WebLogic Server por defecto no son adecuadas para la aplicación, se puede crear una fábrica personalizada para permitir a los clientes crear conexiones JSM.

Weblogic JMS provee valores por defecto en algunas opciones de configuración y otros valores deben ser ingresados. Una vez esté configurado JSM, las aplicaciones pueden enviar y recibir mensajes utilizando el API JSM.

3.1.6. Monitorear servicios y recursos

Para el monitoreo se cuenta con el *Weblogic Diagnostics Framework* (WLDF) que consiste en un conjunto de componentes que trabajan juntos para reunir, archivar y acceder a información de diagnóstico acerca de la instancia de Weblogic Server y las aplicaciones que alberga. WLDF está formado por los siguientes componentes.

- Creadores de datos (bitácora y recolector)
- Recolectores de datos
- Archivadores
- Accesores
- Instrumentación
- Vigilancia y notificación
- Captura de imagen
- Tablero de monitoreo

Los creadores de datos generan información de diagnóstico que es consumida por la bitácora y el recolector. Estos componentes coordinan con el archivador la persistencia de la información y con el subsistema de vigilancia y notificación para proveer monitoreo automático. El *Accesor* interactúa con la bitácora y el recolector para exponer la información de diagnóstico actual y con el archivador para información histórica. La captura de imagen provee los medios para capturar una instantánea de diagnóstico en un estado clave del servidor.

- Creación, recolección de datos e instrumentación: información de diagnóstico es recolectada de varias fuentes, estas fuentes pueden estar clasificadas como proveedores de datos, creadores de datos que se toman en intervalos regulares de tiempo para recolectar datos actuales, o editores de datos que son creadores de datos que generan eventos de forma síncrona. Los proveedores y editores de datos se distribuyen a través de los componentes y la información generada puede recolectarse por la bitácora o el recolector.

  La instrumentación crea monitores y los inserta en puntos definidos en puntos definidos del flujo de ejecución. Estos monitores publican datos directamente al archivo.

- Archivo: un estado pasado muchas veces es crítico para el diagnóstico de fallas en el sistema. Esto requiere que el estado sea capturado y archivado para acceder a él en el futuro, creando archivos históricos. En WLDF, el archivo satisface esta necesidad con varios componentes de persistencia. Métricas de eventos y recolecciones pueden guardarse y ponerse a disposición para vistas históricas. El archivador provee interfaces de acceso para que el *Accesor* pueda exponer los datos históricos.

- Vigilancia y notificación: este sistema puede ser utilizado para crear monitores automáticos que observan estados de diagnóstico específicos y envían notificaciones basadas en reglas configuradas. Una regla de vigilancia puede monitorear datos en bitácoras, eventos del componente de instrumentación, o datos de métricas de los proveedores de datos que son recolectados.

- Accesores de datos: proveen acceso a todos los datos recolectados por el WLDF, incluyendo bitácoras, eventos y datos de métricas. El accesor interactúa con el archivador para obtener información histórica incluyendo eventos de bitácoras y métricas persistentes. Otras herramientas pueden necesitar acceso a los datos que se encuentran actualmente en un servidor inactivo, en ese caso, un accesor desconectado se puede utilizar para exportar datos a un archivo XML para accederlos después.

- Tablero de monitoreo: este tablero muestra el estado actual e histórico de las instancias de WebLogic Server y de las aplicaciones que alberga proporcionando visualizaciones de las métricas que exponen el rendimiento más crítico de estas y como cambian en el tiempo. Estados históricos son representados por datos recolectados que están persistentes en los archivos. Para ver estos datos en el tablero, se debe configurar el recolector para que capture los datos que se desean en el monitor.

- Captura de imagen de diagnóstico: esta herramienta reúne las fuentes más comunes de los estados clave de un servidor para el diagnóstico de problemas. Empaqueta este estado en un artefacto que puede ponerse a disposición de técnicos de soporte. Incluye también captura bajo demanda, que es la creación de una imagen de diagnóstico por una operación o comando emitido por la consola de administración de WebLogic Server. Notificación de imágenes que es la creación automática de imágenes de diagnóstico en respuesta a un disparador asociado a una regla. Los datos en estas imágenes pueden ser analizados para determinar posibles causas de un problema.

3.1.7. Optimizar el rendimiento de una aplicación

WebLogic Server proporciona opciones que se pueden utilizar para optimizar el rendimiento de una aplicación que se encuentre alojada en el servidor, entre las que se mencionan las siguientes:

- Deshabilitar verificaciones de página, para mejorar el rendimiento, se deben colocar los siguientes parámetros con valor de -1.

    - pageCheckSeconds
    - servlet-reload-check-secs
    - servlet Reload Check

- Utilizar etiquetas JSP personalizadas, como *cache, repeat* y *process*. Estas etiquetas se encuentran en la librería weblogic-tags.jar.

- Pre-compilar JSPs, se puede configurar WebLogic Server para precompilar las JSPs cuando una aplicación web es desplegada o cuando se arranca el servidor al colocar el parámetro jsp-descriptor como verdadero en el archivo weblogic.xml. Para evitar que se vuelvan a compilar los JSPs cada vez que un servidor se reinicia, se debe pre-compilarlas utilizando weblogic.jspc y se debe colocar en la carpeta WEB-INF/classes y archivarlos en una carpeta .war. Mantener los archivos fuente en un directorio separado de la carpeta .war elimina la posibilidad de errores causados por dependencias.

- Deshabilitar el registro de acceso, configurando el elemento *access-logging* a deshabilitado, puede mejorar el rendimiento de la aplicación

eliminando la carga que le trae al servidor la operación del registro de acceso.

- Utilizar compresión de plantillas HTML, el elemento *compress-html-template* comprime el HTML en los bloques de plantillas JSP lo que puede mejorar el rendimiento en tiempo de ejecución.

- Utilizar acuerdos de nivel de servicio para basarse en la asignación de JSPs requeridos por las aplicaciones.

3.1.8. Configurar registro en bitácoras

Los subsistemas de WebLogic Server utilizan los servicios de *Logging* para proveer información acerca de eventos como despliegue de nuevas aplicaciones o la falla de uno o más subsistemas. Una instancia de servidor los utiliza para comunicar su estado y responder a eventos específicos. Cada instancia de WebLogic Server mantiene una bitácora de servidor. Dado que cada dominio de WebLogic Server puede ejecutar concurrentemente múltiples instancias, los servicios de Logging reúnen los mensajes generados en las múltiples instancias en una bitácora de mensajes. Esta bitácora proporciona el estatus del dominio en su totalidad.

Existen dos componentes básicos en cualquier sistema de registro: un componente que produce mensajes y otro componente que los distribuye. Los subsistemas de WebLogic utilizan una característica llamada catálogo de mensajes para producir mensajes y el API de registro de Java para distribuirlos. El catálogo de mensajes provee funciones y APIs que puede utilizar la aplicación para enviar sus propios mensajes al servidor de registros.

Además del catálogo de mensajes, las aplicaciones pueden utilizar los siguientes mecanismos para envío de mensajes a la bitácora de WebLogic:

- Las APIs de weblogic.logging.NonCatalogLogger: con estas APIs, en lugar de llamar mensajes que se encuentran en el catálogo, el texto de los mensajes se coloca directamente en el código de las aplicaciones.

- Puente de servidor de registro: WebLogic Server proporciona un mecanismo por el que la aplicación de registro puede tener sus mensajes redirigidos a los servicios de *logging* sin la necesidad de hacer cambios en el código o implementar alguna propiedad de las APIs.

- Conceptos relacionados: un *Logger* registra los mensajes para un subsistema específico o aplicación. Los servicios de *logging* de WebLogic utilizan una sola instancia de java.util.logging.Logger para registro de mensajes de los catálogos, de *NonCatalogLogger* y el sistema de depuración.

  Un *Handler* es la clase que extiende java.util.logging.Handler, recibe las peticiones enviadas a un *logger.* Cada instancia del *logger* puede estar asociada con un número de *handlers* para los cuales despacha mensajes. Un *handler* se adjunta a un tipo de bitácora de mensajes específica.

El proceso del servidor *logging* funciona de la siguiente manera, los subsistemas o aplicaciones le envían peticiones al *logger*, este asigna objetos *logRecords* que son envíados al *handler* para su publicación. Ambos objetos utilizan niveles de severidad y opcionalmente filtros, para determinar si están interesados en un *logRecord* en particular. Cuando es necesario publicar un

logRecord externamente, un *handler* utiliza un formateador para localizar y darle formato a los mensajes de la bitácora antes de publicarlos.

3.2. Tareas comunes de administración de GlassFish

A continuación se muestran las tareas básicas que se deben realizar para asegurar el buen funcionamiento de GlassFish Server, así como de las aplicaciones que se encuentren alojadas en este.

3.2.1. Instalación

Para completar la instalación básica de GlassFish Server de manera exitosa, se deben seguir tres pasos entre los cuales se detalla de una manera más específica la instalación del software, como se muestra a continuación:

Paso 1. Completar los pre-requisitos necesarios para iniciar la instalación

- Espacio de disco: el tamaño de descarga depende del paquete de instalación que se elija. Si se elige un paquete ejecutable se necesita tener mínimo 62 megabytes de espacio para la versión completa y 39 megabytes para la versión web. Para un paquete comprimido (.zip), la versión completa necesita 118 megabytes y la versión web 75 megabytes.

  El espacio de instalación depende de la configuración a realizar. Pero el tamaño aproximado para la versión completa es de 250 megabytes y para la versión web 150 megabytes.

- Puertos libres requeridos: se debe tener 17 puertos disponibles para que GlassFish Server utilice. El programa de instalación automáticamente detecta los puertos que están en uso y sugiere puertos concurrentes que no tengan uso. En situaciones cuando múltiples dominios se ejecutan en una máquina, se puede dar conflicto con los puertos, para evitar esto se pueden configurar puertos específicos para el servicio de mensajería y el GMS.

- Utilizar los archivos binarios de JDK: los archivos binarios de Java y *Keytool* que se utilicen con GlassFish Server deben provenir del software de JDK, no de JRE. Para asegurarse que se cumpla este requerimiento se debe asegurar que el directorio bin del JDK se encuentre antes de JRE en las variables de entorno.

Paso 2. Escoger un modo de instalación. Entre los que se pueden escoger está paquetes comprimidos y de extracción automática. A continuación se listan las características de cada una.

- Paquete comprimido

    - Es más simple de instalar pero no provee ninguna opción de configuración durante la instalación.

    - Cualquier configuración adicional se debe realizar manualmente luego de la instalación.

    - Incluye la herramienta de actualización y el instalador de paquetes que puede ser utilizado para instalar componentes al terminar la instalación de GlassFish Server.

- Si se desea desinstalar, debe hacerse manualmente.

- Paquete de extracción automática

  - Provee un asistente de instalación basado en una interfaz de usuario con muchas opciones de configuración.

  - El asistente de instalación se puede utilizar luego de la instalación inicial para realizar tareas de configuración adicionales.

  - Puede ejecutarse en modo silencioso, que es útil para realizar instalaciones basadas en *scripts*.

  - Si es necesario desinstalar, provee un interfaz de usuario para hacerlo.

Paso 3. Este paso se compone de varias tareas que se deben realizar para completar la instalación del software únicamente, las tareas se detallan a continuación:

- Descargar el paquete de GlassFish Server que se desea instalar.
- Desde la consola, colocarse en el directorio donde se descargó el paquete.

- Iniciar el instalador con el comando chmod +x ./nombre_del_archivo y luego con el comando sh ./nombre del archivo.

- Luego de iniciar el instalador, aparece la página de Introducción y se utiliza el botón de siguiente para avanzar en el proceso de instalación.

- Se elige el tipo de instalación, típica, recomendada para desarrollo de aplicaciones, no para utilizarlo en producción. Esta opción instala el GlassFish Server y crea el DAS. Personalizada, para utilizar en un ambiente de producción.

- Especificar el directorio de instalación como directorioUsuario/glassfish3. Este directorio debe estar vacío antes de la instalación.

- Especificar si se desea instalar y habilitar la herramienta de actualización. Si se habilita, se debe especificar el servidor proxy y el puerto.

- Revisar las selecciones listas para instalar y dar clic en Instalar para proceder. Aparecerá una barra de progreso conforme se van instalando los componentes de GlassFish Server.

- Revisar las opciones de información del dominio. Para asegurar el dominio se debe colocar una contraseña para el usuario administrador.

- Revisar los resultados de la página de configuraciones, que presenta los comandos ejecutados recientemente, y dar clic en Siguiente para continuar.

- Revisar la página de resumen y dar clic en Salir para finalizar la instalación. La información de la instalación es almacenada en archivos de bitácoras que se encuentran en el directorio /tmp.

- Registrar el producto GlassFish Server, siguiendo las instrucciones que aparecen en la página de registro.

3.2.2. Inicio y detención de servidores

GlassFish Server proporciona varias formas para iniciar y detener instancias de servidores, el método a elegir depende en si se prefiere utilizar la línea de comando o el modo gráfico. El DAS se inicia y se detiene al iniciar o detener un servidor.

- Iniciar un servidor: cuando se inicia un servidor, el DAS se inicia también. Luego del arranque, el DAS se ejecuta constantemente escuchando y aceptando peticiones. Si el directorio del dominio no está especificado, el directorio predeterminado se inicia. Cada servidor debe ser iniciado de forma separada.

  Con la herramienta asadmin y el commando *start-domain* se inicia el dominio. Si hay más de un servidor debemos especificar el nombre del servidor que se desea iniciar.

- Detener un servidor: cuando se detiene un servidor, se apaga el DAS correspondiente a este. Primero deja de aceptar nuevas conexiones y espera que se completen todas las conexiones existentes. Este proceso de apagado dura unos segundos. Mientras se apaga el servidor, la consola de administración y la mayoría de subcomandos de asadmin no se pueden utilizar. Los comandos que si se pueden utilizar son útiles para detener un servidor perdido.

  El comando para detener un servidor es *stop-domain*, si existiera más de un servidor, se debe especificar el nombre del mismo.

- Reiniciar un servidor: por medio del comando *restart-domain* se puede reiniciar el DAS de un servidor en específico de forma remota. Cuando se reinicia un servidor, el DAS deja de aceptar nuevas conexiones y espera que las conexiones existentes finalicen. Este proceso de reinicio toma unos segundos, en los que no se puede utilizar la consola de administración y la mayoría de comandos de *asmadmin*.

  Este subcomando es útil particularmente par ambientes en donde la máquina del GlassFish Server está asegurada y es muy difícil acceder a ella. Con las credenciales correctas, se puede reiniciar el servidor de forma remota así como desde la misma máquina. Si este no funcionara, debe utilizarse el comando *stop-domain* seguido por un *start-domain* que funciona de la misma manera.

3.2.3. Configurar las conexiones a la base de datos

Los pasos para crear una conexión a la base de datos incluyen la creación de un *pool* de conexiones JDBC, crear un recurso JDBC para el *pool* de conexiones y por último integrar un controlador JDBC en el dominio de administración.

- *Pool* de conexiones: es un grupo de conexiones reutilizables para una base de datos en particular. Dado que el crear nuevas conexiones físicas lleva mucho tiempo, GlassFish Server mantiene un *pool* disponible para nuevas conexiones. Cuando una aplicación solicita una conexión, la obtiene de este *pool*, y cuando la aplicación cierra la conexión, esta vuelve al *pool* para ser utilizada de nuevo. Las conexiones JDBC pueden ser accesibles globalmente o pueden limitarse a una aplicación o módulo web.

Un recurso JDBC es creado al especificar con que recurso se asocia un *pool* de conexiones. Las propiedades de los *pool* de conexiones pueden variar dependiendo de la base de datos a utilizar. Algunas propiedades comunes son el nombre de la base de datos (URL), el nombre del usuario y su contraseña.

- Creación del *pool* de conexiones: para crear el *pool* de conexiones se utiliza el subcomando *create-jdbc-connection-pool* de manera remota, para registrar un nuevo *pool* de conexiones con el nombre especificado. Una conexión o un conector a un *pool* de conexiones pueden ser creados con autenticación, se puede utilizar un subcomando para especificar el nombre de usuario y la contraseña u otra información de la conexión utilizando asadmin. También se puede especificar la información de la conexión en el archivo descriptor que es un XML.

  Un *pool* de conexiones es necesario para cada base de datos, posiblemente más dependiendo de la aplicación. La creación de un *pool* de conexiones es un evento dinámico por lo que no requiere que se reinicie el servidor.

3.2.4. Configurar los servicios de mensajería

Para el servicio de mensajería, GlassFish Server utiliza *Java Message Service (*JSM). Los anfitriones JMS son los servidores de mensajes que albergan los destinos, almacenan los mensajes e interactúan con las aplicaciones para enviar y recibir mensajes a través de las conexiones. El servicio de JMS soporta los siguientes tipos de anfitriones:

- El tipo incrustado, en el que el anfitrión JSM se ejecuta en la misma JVM que la instancia de GlassFish Server, su configuración y ciclo de vida se manejan a través del servicio JMS.

- El tipo local, en el que el anfitrión JMS se ejecuta de forma separada en la misma instancia de GlassFish Server, su configuración y ciclo de vida se manejan a través del servicio JMS.

- El tipo remoto, en el que el anfitrión JMS representa una cola que es externa al servicio JMS, esta operación se maneja utilizando las herramientas administrativas de colas de mensajes.

Los recursos fábrica de conexiones, albergan la información que las aplicaciones utilizan para conectarse a un proveedor JMS. Para cada fábrica de conexiones, el servicio JMS automáticamente mantiene un conector y un *pool* de conexiones para soportar el grupo de conexiones y alguna falla que pueda ocurrir.

Los recursos destino albergan la información que las aplicaciones utilizan para especificar el destino objetivo de los mensajes que producen y la procedencia del destino de mensajes que consumen. Por cada recurso destino, el servicio JMS automáticamente mantiene un objeto administrado GlassFish. Los destinos físicos proveen los medios para crear y gestionar los destinos administrativamente, en lugar de tener que crearse dinámicamente cuando una aplicación los necesite. Mientras la creación de destinos dinámicamente es suficiente durante el desarrollo de la aplicación, administrativamente los destinos físicos se adaptan más a ambientes de producción.

Para la creación del anfitrión JMS se utiliza el comando create-jms-host que debe ir acompañado de varios atributos como: el nombre de la cola de mensajes, el puerto de la cola de mensajes, el nombre del usuario administrativo de la cola de mensajes, la contraseña correspondiente al usuario administrativo, el objeto GlassFish Server para el que se está creando este anfitrión JMS.

3.2.5. Monitorear servicios y recursos

Monitoreo, es el proceso de revisar las estadísticas de un sistema para mejorar o resolver problemas de rendimiento. El servicio de monitoreo, puede rastrear y desplegar estadísticas operacionales tales como el total de peticiones por segundo, un tiempo promedio de respuesta y el rendimiento. Al monitorear el estado de varios componentes y servicios desplegados en GlassFish Server, se puede identificar los cuellos de botella, predecir fallas y asegurarse que todo funcione como se espera. Los datos reunidos al monitoreo pueden ser útiles en la afinación del rendimiento y planificación de capacidades.

Los objetos que se pueden monitorear son componentes subcomponentes o servicios. GlassFish Server utiliza una estructura de árbol para rastear los objetos que se pueden monitorear. Dado que este árbol es dinámico, va cambiando conforme los componentes se van agregando o eliminando. En el árbol, un objeto monitoreable puede tener objetos hijos (nodos) que representan exactamente lo que puede ser monitoreado para ese objeto. Todos los objetos hijo se acceden utilizando el '.' como un carácter separador.

Las herramientas que se tienen disponibles para el monitoreo son los siguientes subcomandos de asadmin.

- *Enable-monitoring*, *disable-monitoring* o los commandos *get* y *set* son utilizados para iniciar o detener el monitoreo.

- El subcomando *monitor type* se utiliza para desplegar información básica para un objeto en particular.

- El subcomando *list --monitor* se utiliza para desplegar objetos que pueden ser monitoreados con el subcomando monitor.

- El subcomando *get* se utiliza para desplegar datos que sean entendibles, como los atributos y valores de un objeto. Este subcomando se utiliza con un parámetro comodín que muestra todos los atributos disponibles para cualquier objeto.

Por defecto, el servicio de monitoreo ya está habilitado para GlassFish Server, pero no está habilitado el monitoreo de módulos individuales. Para habilitar el nivel de monitoreo de módulos, se cambia el nivel de *low* a *high.* Se puede dejar el monitoreo en *off* para objetos que no necesitan ser monitoreados. Para un nivel de monitoreo *low* se mantienen estadísticas simples, en *high*, se mantienen estadísticas simples más estadísticas de los módulos y en *off*, no se hace monitoreo por lo que no tiene ningún impacto en el rendimiento. Para habilitar el monitoreo se utiliza el comando *enable-monitoring* y no es necesario reiniciar el servidor para que se aplique este cambio.

3.2.6. Optimizar el rendimiento de una aplicación

Para mejorar el rendimiento de una aplicación, se pueden aplicar algunas técnicas, como el uso de JSPs pre-compilados, compilar archivos JSP consume muchos recursos, por lo que pre-compilar estas páginas antes del despliegue

de las aplicaciones en el servidor va a mejorar el rendimiento de las mismas. Cuando se hace esto, solo las clases resultantes serán desplegadas.

Se puede especificar el precompilar archivos JSP cuando se despliega una aplicación a través de la consola de administración o el subcomando *deploy.* También se puede especificar el pre-compilar archivos JSP para una aplicación ya desplegada con la consola de administración, en el nodo del dominio en la sección configuración de aplicaciones.

Otra técnica es deshabilitar el volver a cargar la aplicación dinámicamente, si esta característica se encuentra habilitada, periódicamente el servidor verifica si hay cambios en las aplicaciones desplegadas y automáticamente recarga la aplicación con los cambios. Esta característica está destinada para ambientes de desarrollo por lo que debe deshabilitarse para mejorar el rendimiento. Se puede utilizar la consola de administración para deshabilitar volver a cargar la aplicación dinámicamente para una aplicación que ya está desplegada.

3.2.7. Configurar el registro en bitácoras

El registro, es el proceso mediante el cual GlassFish Server captura información acerca de eventos que ocurren, tales como errores de configuración, fallas de seguridad o funcionamiento defectuoso del servidor. Esta información es grabada en archivos de bitácoras y es usualmente la primera fuente de información cuando un problema ocurre. Verificar estas bitácoras es útil para determinar la salud del servidor.

Los registros de GlassFish server se almacenan un uno de dos tipos generales de archivos de bitácoras que son:

- Bitácoras de servidor, que almacenan la información acerca de la operación de una instancia ejecutándose en un dominio, cada instancia, miembro de un *cluster*, y cada DAS, tiene un archivo de bitácora de servidor individual.

- Bitácoras de *cluster*, que almacenan información acerca de la operación de una instancia de un *cluster*, si hay una o más configuradas en el dominio. Cada servidor que es parte de un *cluster* tiene una bitácora de *cluster,* además de una bitácora de servidor. Sin embargo, el contenido de la bitácora de *cluster* puede variar de una instancia a otra dependiendo en los factores como, de qué manera la instancia está asignada en el *cluster,* qué aplicaciones están ejecutándose en él, cómo está configurado el balanceo de carga y la respuesta a fallos.

Cada instancia de GlassFish Server tiene una bitácora dedicada y cada instancia y *cluster* tiene su propio archivo de propiedades de registro. Para configurar el registro para una instancia o un *cluster*, GlassFish Server permite elegir bitácoras específicas o archivos de registro específicos cuando se realiza lo siguiente:

- Configurar niveles de registros globales o modulares.
- Rotar bitácoras o comprimirlas en un archivo ZIP.
- Cambiar los atributos en el archivo de propiedades.
- Listar los niveles de registro o los atributos.

El DAS, así como cada configuración, instancia y *cluster*, tiene su propio conjunto de propiedades de registros que son mantenidos en archivos de configuración individuales. El archivo de propiedades de registro se llama *logging.properties* y este incluye la siguiente información:

- El nombre de la bitácora y su localización.
- Nombres y niveles de registro.
- Propiedades de rotación y formato de registros.

Para cambiar el nombre y localización de la bitácora, primero se utiliza el subcomando list-log-attributes para obtener el valor actual de estos atributos, luego se utiliza el subcomando *set-log-attributes* para especificar el nuevo nombre y localización. Estos comandos se ejecutan en el DAS por defecto, pero también se puede especificar el nombre de alguna configuración, servidor, instancia o *cluster.*

Para cambiar el nombre y la localización del archivo de propiedades, se configura la propiedad del sistema *java.util.logging.config.file*. Al configurar esta propiedad, se puede tener un solo archivo de propiedades que sea utilizado por todas las instancias ejecutándose en la misma máquina.

GlassFish server tiene la característica de rotar las bitácoras cuando llegan a un tamaño de 2 megabytes, así que también se puede modificar las propiedades de rotación. Por ejemplo se puede cambiar el tamaño en el cual el servidor debe rotar la bitácora, se puede configurar un servidor para rotar las bitácoras según un intervalo de tiempo, especificar el máximo número de bitácoras rotadas que se pueden acumular y se puede rotar las bitácoras manualmente.

4. CONFIGURACIÓN PARA EL DESPLIEGUE DE APLICACIONES

4.1. Proceso de despliegue de aplicaciones para un WebLogic Server

El término despliegue de aplicaciones se refiere al proceso de hacer que una aplicación o un módulo esté disponible para procesar peticiones de clientes en un dominio de WebLogic Server, a continuación se muestran las tareas que involucran el despliegue de aplicaciones.

4.1.1. Utilizar archivos empaquetados

Un archivo empaquetado es un solo archivo que contiene todas clases, archivos estáticos, directorios, y otros, de una aplicación o un módulo. En la mayoría de ambientes de producción, lo que recibe el administrador de aplicaciones para despliegue es almacenado como archivo empaquetado. Las unidades de despliegue que son empaquetadas utilizando la función jar tienen una extensión especifica dependiendo del tipo por ejemplo aplicaciones web es un archivo .war.

Además de un archivo empaquetado, también puede que se reciba un plan de despliegue, que es un archivo aparte que configura la aplicación para un ambiente específico.

Se puede utilizar también directorios explotados, que contienen los mismos archivos y directorios que un archivo JAR. Si se utiliza este tipo de directorios, se necesitará desempaquetar manualmente despliegues realizados

anteriormente. Sin embargo los archivos y directorios residen directamente en el sistema de archivos y no son empaquetados en un archivo con la funcionalidad jar.

Se puede elegir el despliegue de un directorio explotado bajo las siguientes circunstancias:

- Se desea realizar actualizaciones parciales en una aplicación luego del despliegue. Desplegar aplicaciones como un directorio explotado, hace que sea más fácil realizar actualizaciones de módulos individuales de la aplicación sin tener que recrear el archivo.

- Se desea desplegar una aplicación web que contiene archivos estáticos que serán actualizados periódicamente. En este caso, es más conveniente el desplegar aplicaciones como un archivo explotado porque se puede hacer la actualización y refrescar los archivos estáticos sin re crear el archivo completo.

- Se van a desplegar aplicaciones web que realizan lectura/escritura al sistema de archivos a través del contexto de la aplicación, por ejemplo, una aplicación web que trata de editar o actualizar dinámicamente partes de la misma aplicación. En este caso, los módulos que realizan las operaciones de lectura/escritura deben tener un directorio físico en el cual trabajar, no se puede obtener un archivo cuando la aplicación está desplegada como un archivo empaquetado.

Si se tiene un archivo empaquetado que se quiere desplegar como un archivo explotado, se puede utilizar la función jar para desempaquetar el archivo en un directorio. Si se está desempaquetando un archivo que contiene

otros módulos empaquetados y se quiere realizar actualizaciones parciales a esos módulos, se debe expandir los archivos incrustados también. Se debe asegurar de desempaquetar cada módulo en un subdirectorio con el mismo nombre que los archivos empaquetados.

4.1.2. Nombres de despliegue predeterminados

Cuando se realiza por primera vez una aplicación o un módulo en una o más instancias de WebLogic Server, se especifica un nombre de despliegue que describe colectivamente los archivos de despliegue, el servidor objetivo y otras opciones de configuración seleccionadas previamente. Luego se puede redesplegar o detener una unidad de despliegue en todos los servidores involucrados utilizando únicamente en nombre de despliegue.

Si no se especifica un nombre de despliegue en el momento del despliegue, la herramienta de despliegue selecciona un nombre por defecto basado en los archivos fuente de despliegue. Para archivos empaquetados, *weblogic.Deployer* utiliza el nombre de los archivos sin la extensión. Para un directorio explotado utiliza el nombre del directorio de más alto nivel que se despliegue.

4.1.3. Requerimientos de nombres de aplicaciones

Para que se pueda desplegar una aplicación de manera exitosa en un WebLogic Server, se debe tener un nombre válido de aplicación. Los requerimientos para los nombres de aplicaciones son los siguientes.

- Los nombres de las aplicaciones solo deben contener los siguientes caracteres alfanuméricos en mayúsculas o minúsculas, guión bajo, guión y punto. No se permiten caracteres adicionales.

- Los nombres que contengan el '.' deben contener al menos un carácter adicional diferente del '.'.

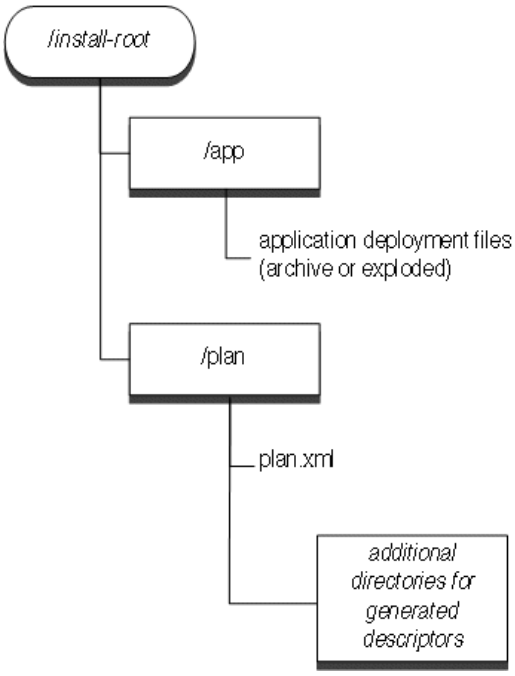
- Los nombres de las aplicaciones deben tener menos de 215 caracteres de largo.

Además del nombre de despliegue, un módulo o una aplicación puede tener también una cadena de texto de versión asociada. Esta cadena distingue el despliegue inicial de versiones re-desplegadas posteriormente. Por ejemplo cuando se actualiza una aplicación para arreglar problemas o agregar nuevas características. En sistemas de producción, es de vital importancia mantener un texto de versiones para la versión inicial y para los despliegues posteriores, al hacerlo así, permite actualizar un redesplegar una nueva versión de la aplicación sin interrumpir el servicio de los clientes existentes.

4.1.4. Directorio de instalación de la aplicación

El directorio de instalación de la aplicación separa los archivos de configuración generados de los archivos que son el núcleo de la aplicación, para que los archivos de configuración puedan ser cambiados o reemplazados fácilmente sin molestar a la aplicación en sí. La estructura del directorio también ayuda a organizar y mantener múltiples versiones de archivos de despliegue de la misma aplicación. En la figura 10 se muestra la jerarquía del directorio de instalación que almacena solo una versión de una aplicación o un módulo.

Figura 7. **Directorio de instalación de aplicaciones**



Fuente: Documentación oficial de Oracle. http: //docs.oracle.com /cd /E23943_01 /web.1111 /e13702/deployunits.htm#i1047223. Consultado: 2 de noviembre de 2013.

Es recomendado copiar todos los nuevos despliegues en un directorio de instalación antes de desplegarlos en un dominio de WebLogic Server. Realizar el despliegue desde este directorio facilita el identificar todos los archivos asociados con la unidad de despliegue.

Para crear un directorio de instalación se deben seguir los siguientes pasos:

- Seleccionar el directorio de alto nivel donde se quiere almacenar los archivos desplegados para aplicaciones y módulos en el sistema. Se

recomienda no almacenarlos en el directorio del dominio de WebLogic Server, si es posible se deben almacenar en un directorio que esté accesible para el servidor de administrador y los servidores administrados en el dominio.

- Crear un subdirectorio dedicado para la aplicación o módulo que se desea desplegar.

- Crear un subdirectorio bajo el directorio de la aplicación para designar la versión de la aplicación que se está desplegando, este directorio debe llamarse igual a la cadena de texto que designa la versión de la aplicación.

- Este subdirectorio se volverá el directorio raíz de instalación sobre el cual se desplegará la aplicación, se debe crear ahí entonces los subdirectorios \app y \plan.

- Copiar el código de despliegue de la aplicación en el subdirectorio \app, si se está desplegando la aplicación de un archivo empaquetado, simplemente copiar este archivo. Si se está desplegando de un archivo explotado, copiar todo el directorio en la carpeta \app.

- Si se tiene uno o más de un plan de despliegue para la aplicación, se debe copiar en el subdirectorio \plan, uno para cada plan que se tenga.

4.1.5. Ciclo de vida de la configuración

La configuración de despliegue para una aplicación puede ocurrir en varios puntos en el ciclo de vida de una aplicación. Cada fase de la

configuración de despliegue típicamente incluye crear y trabajar con diferentes archivos de despliegue.

- Configuración de desarrollo, durante el desarrollo, el programador crea descriptores de despliegue para una aplicación o un módulo. El programador también crea descriptores de despliegue de WebLogic Server para configurar la aplicación para despliegue en un ambiente de desarrollo de WebLogic.

- Configuración de exportación, antes de lanzar una aplicación de desarrollo, el programador o diseñador puede exportar la configuración de despliegue de una aplicación a un plan de despliegue de WebLogic Server. Exportar la configuración crea un plan de despliegue variable para todas las propiedades definidas previamente por un desarrollador en los descriptores de WebLogic Server.

  Exportar una aplicación ayuda a desarrolladores en otras áreas de la organización a desplegar aplicaciones en ambientes diferentes que los del ambiente de desarrollo del programador. El plan ideal de despliegue incluye todas las propiedades que el encargado de despliegue necesita modificar antes de desplegar la aplicación en un nuevo ambiente.

- Configuración a la hora del despliegue, un administrador o encargado de despliegue, configura la aplicación antes de desplegarla en el ambiente objetivo. Esta configuración puede utilizar el mismo plan y configuración creado durante el desarrollo, también puede utilizar versiones modificadas del plan de despliegue y la configuración, o un plan personalizado que el encargado del despliegue creo previamente para el ambiente objetivo.

- Configuración después del despliegue, luego que una aplicación ha sido desplegada a un ambiente objetivo, un administrador o encargado del despliegue puede reconfigurar la aplicación al re-desplegarla con un nuevo plan o utilizando la consola de administración para actualizar y re-desplegar un plan existente.

4.1.6. Descriptor de despliegue

La configuración básica de despliegue de una aplicación se define en un documento XML que se conoce como descriptor de despliegue, que son incluidos como parte del archivo de la aplicación que se recibe para el despliegue. Los descriptores se pueden definir en dos categorías:

- Descriptores de despliegue Java EE que definen la organización y comportamiento fundamental de aplicaciones o módulos java independientemente de donde la aplicación sea desplegada. Cada aplicación o módulo Java, necesita un descriptor específico.

- Descriptores de despliegue WebLogic Server que definen la dependencia de recursos y los parámetros de afinación que una aplicación utiliza en un ambiente específico de WebLogic Server.

Para los propósitos de un despliegue a producción, se debería tratar los dos descriptores como parte del código fuente de la aplicación. No se debe editar los descriptores de la aplicación para configurar el despliegue a un ambiente de producción, en lugar de esto se debe colocar los cambios en el plan de despliegue que se explica a continuación.

4.1.7. Plan de despliegue

Un plan de despliegue es un archivo XML opcional que se utiliza para configurar una aplicación para su despliegue en un ambiente específico de WebLogic Server. Un plan de despliegue, especifica los valores de las propiedades que normalmente estarían definidos por el descriptor de despliegue de WebLogic Server. Cuando se exporta una aplicación, el plan de despliegue usualmente actúa para anular las propiedades seleccionadas en el descriptor de despliegue WebLogic creado durante el desarrollo de la aplicación.

Típicamente, los planes de despliegue son creados por desarrolladores junto con los archivos de aplicación asociados, luego son distribuidos al administrador quien actualiza el plan para un ambiente en específico, como pruebas o producción. El plan de despliegue se almacena fuera del archivo o directorio de la aplicación.

El plan de despliegue ayuda al administrador a modificar fácilmente la configuración de una aplicación para despliegue en múltiples y diferentes ambientes de WebLogic Server sin modificar los archivos de despliegue de la aplicación. Los cambios en la configuración se aplican al agregar o modificar variables en el plan de despliegue, que puede definir la localización de las propiedades de los descriptores a modificar y el valor que se les debe asignar a esas propiedades.

4.2. Despliegue de aplicaciones para un GlassFish Server

El término despliegue de aplicaciones se refiere al proceso de hacer que una aplicación o un módulo esté disponible para procesar peticiones de

clientes, a continuación se muestran las tareas que involucran el despliegue de aplicaciones.

4.2.1. Funcionalidad general

Existe una gran variedad de módulos Java, como ejemplo se pueden mencionar: conectores, módulos web, aplicaciones y otros que se pueden desplegar de cualquiera de las formas descritas a continuación:

- Despliegue de archivos, de esta manera de despliegan las aplicaciones como un archivo empaquetado.

- Recarga dinámica, vuelve a desplegar la aplicación al crear o modificar un archivo *.reload* en el repositorio de la aplicación.

- Despliegue automático, despliega el archivo de la aplicación que es colocado en el directorio destinado para esta tarea.

- Despliegue de directorio, despliega la aplicación que se encuentra en formato directorio.

Un plan de despliegue que despliega un archivo junto con el plan que contiene los descriptores de despliegue GlassFish Server, puede aplicar a cualquiera de estas técnicas. Existen dos situaciones que requieren de protección y procesos especiales:

- Un ambiente de desarrollo provee de un conjunto de herramientas y espacios de trabajo libres para un número relativamente pequeño de desarrolladores que están creando y probando aplicaciones y módulos.

- Un ambiente de producción provee un ambiente estable y protegido, donde las aplicaciones son afinadas a su máxima eficiencia para uso comercial en lugar de desarrollo.

Algunos métodos de despliegue que son utilizados de manera exitosa en un ambiente de desarrollo, no deberían utilizarse en producción. Además cuando se realiza una re-carga, las sesiones que están en tránsito se vuelven inválidas, lo que puede no ser un problema para desarrollo, pero si puede ser muy importante en producción. El cliente debe reiniciar la sesión, lo cual es negativo para un ambiente de producción.

Para ambientes de producción, cualquier actualización debería realizarse como una actualización continua, que actualiza las aplicaciones y módulos sin interrumpir el servicio.

4.2.2. Descriptores y anotaciones

Un descriptor de despliegue es un archivo XML que describe como una aplicación Java o módulo debe ser desplegado. Cada descriptor tiene una definición de tipo de documento (DTD) o un esquema (XSD) que le corresponde que define los elementos, datos y atributos que el descriptor puede contener. El descriptor dirige a la herramienta de despliegue a desplegar un módulo o aplicación con opciones específicas y también requerimientos de configuración que se deben resolver.

Ya que la información en un descriptor es declarativa, puede ser cambiada sin requerir modificaciones al código fuente. Durante el despliegue, GlassFish Server lee la información en el descriptor y despliega la aplicación o módulo

como se le indica. Existen dos tipos de descriptores de despliegue asociados con GlassFish Server:

- Descriptor de Java EE.

- Descriptor de GlassFish Server, que proporciona opciones de configuración que son específicas para GlassFish Server. A menos que se indique lo contrario, la configuración de los descriptores de GlassFish Server anulan la configuración de los descriptores de Java.

Una anotación, también llamada *metadata*, permite un estilo de programación declarativa. Se puede especificar información dentro de la clase utilizando anotaciones. Cuando la aplicación o módulo es desplegada la información puede ser utilizada o anulada por el descriptor de despliegue.

4.2.3. Despliegue basado en módulos

Un módulo es un grupo de uno o más componentes Java que se ejecutan en el mismo tipo de contenedor, como un contenedor web por ejemplo. Este módulo utiliza anotaciones o descriptores de despliegue de ese contenedor en específico. Se puede desplegar un módulo en solitario o como parte de una aplicación. GlassFish Server soporta los siguientes tipos de módulos:

- Módulos web, que son conocidos también como aplicaciones web, es un conjunto de *servlets*, EJBs, páginas HTML, clases y otros recursos que se pueden unir y desplegar a varios servidores. Un archivo de aplicación web (WAR) es el formato estándar para ensamblar aplicaciones web. Un WAR consiste en los siguientes elementos: *servlets,* JSPs, páginas

estáticas, clases *bean,* más anotaciones o descriptores de despliegue web (web.xml y glassfish-web.xml).

- Un módulo EJB es una unidad de software desplegable que consiste en uno o más *beans* empresariales, más un descriptor de despliegue EJB. El archivo estándar para este módulo es un JAR y contiene las clases *bean* y todas las clases que utiliza, anotaciones o descriptores de despliegue (ejb-jar.xml y glassfish-ejb-jar.xml).

- Un módulo conector también se conoce como un módulo adaptador de recursos, es una unidad de software desplegable que proporciona una manera portable para que los componentes EJB se accedan a sistemas de información empresarial (EIS) externos. Un módulo conector consiste en interfaces, clases y librerías nativas para implementar el módulo, además del descriptor de despliegue. El estándar de un adaptador de recurso es un archivo RAR, cada GlassFish Server tiene sus anotaciones y descriptor de despliegue (ra.xml).

- Un módulo de aplicación cliente es una unidad de software que consiste en una o más clases y descriptores de despliegue de aplicaciones cliente (application-client.xml y glassfish-application-client.xml). El archivo que se utiliza para este módulo es un JAR.

- El módulo de ciclo de vida provee los medios de ejecución de tareas de larga o corta duración dentro del ambiente de GlassFish Server.

Se pueden desplegar módulos web, EJB y de aplicaciones cliente por separado, afuera de cualquier aplicación. Este tipo de despliegue es apropiado cuando los componentes necesitan ser accedidos por otros módulos o

aplicaciones. Este tipo de despliegue permite acceso compartido a un *bean* desde un componente web, EJB o aplicación cliente.

4.2.4. Despliegue basado en aplicaciones

Una aplicación es un conjunto lógico de uno o más módulos unidos por las anotaciones o descriptores de despliegue. Se ensamblan los componentes en archivos JAR, WAR o RAR, y luego se combinan estos archivos y opcinalmente los descriptores de despliegue en un archivo empresarial que es el que se despliega. El despliegue basado en aplicaciones es recomendado cuando todos los componentes necesitan trabajar juntos como una unidad.

4.2.5. Eventos de ensamblaje y despliegue

Las herramientas de despliegue que son proporcionadas por GlassFish Server pueden ser utilizadas por cualquier usuario autorizado como administrador para desplegar aplicaciones y módulos en cualquier ambiente GlassFish Server. Sin embargo, para un despliegue exitoso se debe realizar una planificación, solo el desarrollador sabe exactamente que es requerido por la aplicación, así que el desarrollador es responsable por el ensamblaje y despliegue inicial.

- Creación del descriptor de despliegue o anotaciones. El desarrollador crea el descriptor o sus anotaciones equivalentes utilizando las herramientas y estándares Java. Las aplicaciones de ejemplo de GlassFish Server contienen descriptores que pueden ser utilizados como plantillas.

- Ensamblaje. El desarrollador ensambla los archivos utilizando las herramientas y estándares Java, como el comando *jar.* La aplicación o módulo es empaquetado en un archivo JAR, WAR, RAR o EAR.

- Despliegue de prueba, el desarrollador realiza pruebas de despliegue del archivo.

- Entrega del archivo, el desarrollador entrega el archivo verificado al administrador para que sea desplegado en un ambiente de producción. El desarrollador incluye debe incluir instrucciones para cualquier tarea adicional que el administrador deba realizar.

- Configuración, el administrador aplica requerimientos de desplegué adicionales. Algunas veces el desarrollador indica requerimientos adicionales, como especificar la base de datos de producción, en ese caso, el administrador edita el archivo y lo vuelve a ensamblar.

- Despliegue en producción, El administrador despliega el archivo al ambiente de producción.

- Si el despliegue falla, el administrador devuelve el archivo al desarrollador, quien resuelve el problema y se lo envía de nuevo al administrador para que sea desplegado.

5. PRÁCTICA

Uno de los objetivos de este trabajo al comparar los servidores de aplicaciones Oracle es determinar cuál es el más recomendable utilizar basándose en la facilidad de uso que estos tienen. Para ayudar a determinar este factor se realizaron las configuraciones de un ambiente de pruebas de cada uno de los servidores determinando la usabilidad de los mismos por medio de los siguientes elementos:

- Si existe o no modo gráfico para realizar la tarea
- El total de pantallas navegadas
- La cantidad de clics dados
- El total de campos de información llenados

5.1. Ambiente de pruebas WebLogic Server

El ambiente de pruebas será configurado en un sistema operativo Linux, Ubuntu 12.04 y se descargó el software de la página oficial de Oracle http://www.oracle.com/technetwork/middleware/weblogic/downloads/wls-main-097127.html donde se descargó el instalador wls1036_linux32.bin.

5.1.1. Instalación de software

A continuación se realiza una descripción los aspectos y características generales que fueron recopilados durante la instalación del software para el ambiente de pruebas de WebLogic Server.

- Modo gráfico: si
- Pantallas navegadas: 8
    - Instrucciones
    - Escoger el directorio para *middleware*
    - Registrarse para actualizaciones de seguridad
    - Seleccionar el tipo de instalación
    - Seleccionar el directorio de instalación
    - Resumen de instalación
    - Progreso de la instalación
    - Mensaje de instalación completa
- Total de clics dados: 12
- Total de campos llenados: ninguno ya que se aceptaron los valores por defecto

5.1.2. Creación y configuración de instancia

A continuación se muestran los aspectos y características generales recopilados durante la creación y configuración de la instancia para el ambiente de pruebas de WebLogic Server.

- Modo gráfico: si
- Pantallas navegadas: 9
    - *Quick start*
    - Bienvenida (creación de dominio)
    - Seleccionar la fuente del dominio
    - Especificar el nombre del dominio y el lugar
    - Configurar el usuario y contraseña de administrador
    - Configurar el modo de arranque y JDK
    - Seleccionar configuración opcional

- Resumen de configuración
- Progreso de la configuración

- Total de clics dados: 18
- Total de campos llenados: 4

5.1.3. Despliegue de aplicaciones

A continuación se muestran los aspectos y características generales recopilados durante el despliegue de una aplicación en el WebLogic Server configurado anteriormente para pruebas.

- Modo gráfico: si
- Pantallas navegadas: 6
  - Pantalla principal consola de administración
  - Resumen de despliegues
  - Asistente de instalación de la aplicación: buscar despliegue que instalar
  - Asistente de instalación de la aplicación: seleccionar estilo de direccionamiento
  - Asistente de instalación de la aplicación: valores opcionales
  - Asistente de instalación de la aplicación: resumen de la aplicación a desplegar
- Total de clics dados: 8
- Total de campos llenados: 0 (se aceptaron los valores por defecto)

5.1.4. Conexión a base de datos (JDBC)

A continuación se muestran los aspectos y características generales recopilados durante la creación de una conexión a base de datos de tipo JDBC para el ambiente de pruebas de WebLogic Server.

- Modo gráfico: si
- Pantallas navegadas: 8
    - Pantalla principal consola de administración
    - Resumen de orígenes de datos JDBC
    - Propiedades de orígenes de datos
    - Selección de controlador de base de datos
    - Opciones de transacción
    - Propiedades de la conexión
    - Probar conexión a base de datos
    - Seleccionar destinos
- Total de clics dados: 22
- Total de campos llenados: 11

5.2. Ambiente de pruebas GlassFish Server

El ambiente de pruebas será configurado en un sistema operativo Linux, Ubuntu 12.04 que debe tener la versión de JDK compatible con el software de instalación, en este caso se instaló la versión de JDK 7.

5.2.1. Instalación de software

A continuación se describen algunos de los aspectos y características generales que fueron recopilados durante el proceso de instalación del software para la configuración del ambiente de pruebas de GlassFish Server.

- Modo gráfico: si
- Pantallas navegadas: 6
  - Introducción
  - Tipo de Instalación
  - Directorio de Instalación
  - Instalación de la herramienta de actualización
  - Resumen de instalación
  - Progreso de la instalación
- Total de clics dados: 7
- Total de campos llenados: 0 (se aceptaron los valores por defecto)

5.2.2. Creación y configuración de instancia

A continuación se muestran los aspectos y características generales que fueron recopilados durante la configuración y creación de una instancia de GlassFish Server en el ambiente de pruebas.

- Modo gráfico: si
- Pantallas Navegadas: 3
  - Información del dominio
  - Resultados de la configuración
  - Resumen de la configuración
- Total de clics dados: 8

- Total de campos llenados: 6

5.2.3. Despliegue de aplicaciones

A continuación se muestran los aspectos y características generales que fueron recopilados durante la realización de un despliegue de aplicación en el ambiente de pruebas de GlassFish Server.

- Modo gráfico: si
- Pantallas navegadas: 4
    - Página principal de administración
    - Aplicaciones
    - Despliegue de aplicaciones o módulos
    - Parámetros de configuración
- Total de clics dados: 8
- Total de campos llenados: 2

5.2.4. Conexión a base de datos (JDBC)

A continuación se muestran los aspectos y características generales que fueron recopilados durante la creación de la conexión a la base de datos de tipo JDBC, para el ambiente de pruebas de WebLogic Server.

- Modo gráfico: si
- Pantallas navegadas: 5
    - Página principal de administración
    - Conjunto de conexiones JDBC
    - Paso 1 de 2: preferencias generales
    - Paso 2 de 2: configuración general

- Recurso de conexiones
- Total de clics dados: 25
- Total de campos llenados: 19 (varía dependiendo de los valores asignados por defecto)

5.3. Resumen comparativo

A continuación se presentan las tablas comparativas que toman en cuenta los aspectos documentados al realizar las diferentes tareas de administración sobre los servidores.

Tabla V. **Comparativo instalación de software**

	WebLogic Server	**GlassFish Server**
Modo Gráfico	Si	Si
Total de pantallas	8	6
Total de clics	12	7
Total de campos llenados	0	0

Fuente: elaboración propia.

Tabla VI. **Comparativo configuración de instancia**

	WebLogic Server	**GlassFish Server**
Modo Gráfico	Si	Si
Total de pantallas	9	3
Total de clics	18	8
Total de campos llenados	4	6

Fuente: elaboración propia.

Tabla VII. **Comparativo despliegue de aplicaciones**

	WebLogic Server	GlassFish Server
Modo Gráfico	Si	Si
Total de pantallas	6	4
Total de clics	8	8
Total de campos llenados	0	2

Fuente: elaboración propia.

Tabla VIII. **Comparativo conexión a base de datos (JDBC)**

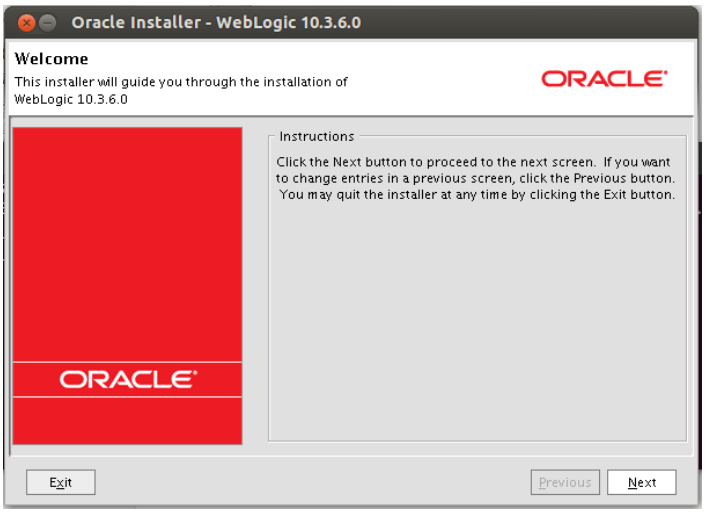
	WebLogic Server	GlassFish Server
Modo Gráfico	Si	Si
Total de pantallas	8	5
Total de clics	22	25
Total de campos llenados	11	19

Fuente: elaboración propia.

5.3.1. Aspecto gráfico

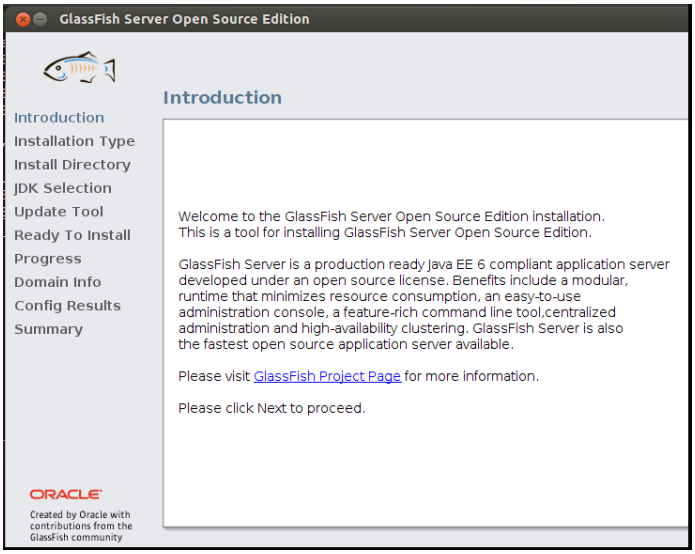
A continuación se muestran tomas de pantalla donde se aprecia la diferencia de aspecto gráfico entre WebLogic y GlassFish Sever en cuanto al asistente de configuración y la consola de administración de ambos.

Figura 8. **Asistente de instalación WebLogic Server**



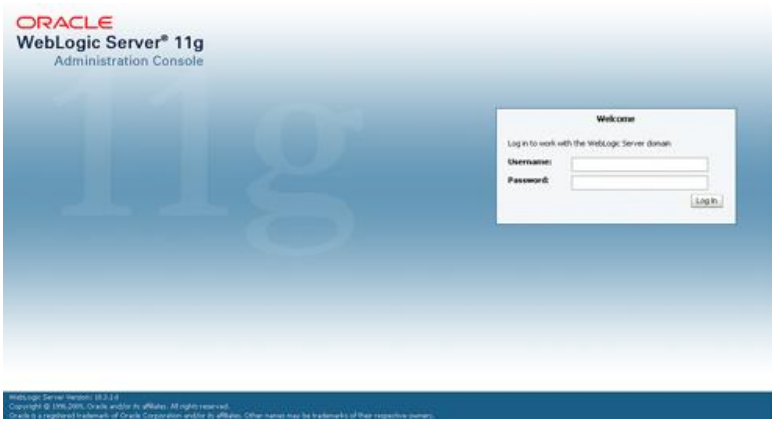
Fuente: elaboración propia.

Figura 9. **Asistente de instalación GlassFish Server**



Fuente: elaboración propia.

Figura 10. **Página de inicio de la consola de administración de WebLogic Server**

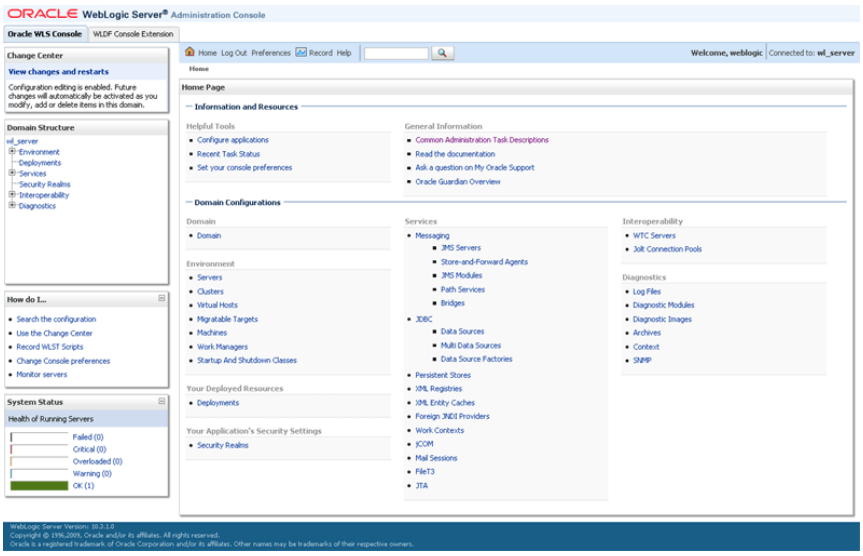


Fuente: elaboración propia.

Figura 11. **Página de inicio de la consola de administración de GlassFish Server**

Fuente: elaboración propia.

Figura 12. **Consola de administración de WebLogic Server**



Fuente: elaboración propia.

Figura 13. **Consola de administración de GlassFish Server**

Fuente: elaboración propia.

CONCLUSIONES

1. Entre las capacidades de los servidores de aplicaciones Oracle está la alta disponibilidad. Los dos servidores proveen tres características fundamentales: protección contra fallas, balanceo de carga y escalabilidad. Con estas características se garantiza la disponibilidad de las aplicaciones alojadas en los servidores y un rendimiento óptimo de las mismas. WebLogic Server a diferencia de GlassFish Server provee soporte para mantener alta disponibilidad durante la migración de un servidor.

2. Otra capacidad que ofrecen los servidores Oracle son los mecanismos de seguridad. Los que provee GlassFish Server se pueden considerar como los más básicos, sin embargo, se mantiene un nivel de seguridad adecuado al utilizarlos. Los mecanismos de WebLogic Server son mucho más completos, con más alternativas para escoger, y provee seguridad desde diferentes puntos como conexiones, inicio de sesión, permisos y otros.

3. Las tareas de administración involucran los mismos pasos para ambos servidores, entre los que se citan como la conexión a servidores web, conexión a bases de datos, monitoreo de servicios y recursos y optimización del rendimiento de las aplicaciones.

4. Para las tareas de administración y configuración, WebLogic Server proporciona más herramientas, con una interfaz de usuario más amigable y fácil de utilizar, mientras que GlassFish tiene una interfaz un

poco más simple con la diferencia que en algunos casos se deben ingresar datos en repetidas ocasiones. Asímismo, en GlassFish Server se pueden realizar las tareas de configuración en menos pantallas y con menos clics pero no de una manera muy personalizada.

5. El proceso de despliegue de aplicaciones para ambos servidores involucra opcionalmente un plan para configurar el despliegue de una aplicación en un ambiente específico, y utiliza descriptores de despliegue para definir la dependencia de recursos y parámetros que utiliza la aplicación en cada ambiente. Ambos servidores también manejan dos tipos de despliegue ya sea basado en módulos, como por ejemplo módulos web o EJB, y basado en aplicaciones que son los módulos ya unidos por los descriptores.

6. La configuración inicial de la instancia y el despliegue de las aplicaciones, son tareas más fáciles de realizar con GlassFish Server, ya que se tiene una cantidad similar de clics y campos llenados pero la cantidad de pasos (pantallas) necesarios es menor y los mismos son más simples que en WebLogic Server.

7. Para la creación de las conexiones a bases de datos, GlassFish Server tiene un asistente de configuración más corto (con menos pantallas) pero en cada pantalla se agrupa una gran cantidad de campos de información que se deben llenar, por lo que en este caso el número de pantallas no índica la facilidad de configuración de la base de datos, ya que en WebLogic se tiene distribuida la información requerida de una manera más entendible de configurar. Por lo tanto esta configuración se considera más fácil de realizar en WebLogic en lugar de GlassFish.

RECOMENDACIONES

1. Si el uso que se le va a dar al servidor de aplicaciones es personal o comercial pero no un modelo de negocio de gran nivel, se recomienda utilizar GlassFish Server, porque proporciona las mismas soluciones de una manera más simple y menos costosa.

2. Si el modelo de negocio va creciendo o se quiere integrar con un modelo ya existente se recomienda utilizar WebLogic Server, el cual es más robusto y proporciona más opciones de configuración y adaptación a otros sistemas.

3. Si la información que va a ser manejada en el servidor de aplicaciones es de carácter sensible y confidencial, se recomienda el uso de WebLogic Server ya que este permite un manejo de seguridad más extenso y con un mayor control sobre ella.

BIBLIOGRAFÍA

1. GONZÁLEZ, Roberto. *Arquitectura de aplicaciones Web* [en línea]. [Universidad de Lleida] 2010. <http://ocw.udl.cat/enginyeria-i-arquitectura/enginyeria-del-software-iii/Continguts/1%20%20 Introduccion/2-Arquitectura.pdf> [Consulta: 2 de noviembre de 2013].

2. ORACLE Corporation. *Oracle Fussion Middleware. Oracle WebLogic Server Documentation Library* [en línea]. <http:// docs.oracle.com /cd/E23943_01/wls.htm>. [Consulta: 3 de octubre de 2013].

3. ORACLE Corporation. *Oracle Fussion Middleware. Oracle GlassFish Server Documentation Library* [en línea]. <http://docs.oracle. com/cd/ E26576_01/index.htm>. [Consulta: 2 de noviembre de 2013].

4. RAMÍREZ, Edgar. *Servidores de aplicaciones* [en linea].<http:// edgaramirez.wordpress.com/2008/11/04/servidor-de-aplicaciones> [Consulta: 19 de octubre de 2013].