UNIVERSIDAD TÉCNICA DEL NORTE

Facultad de Ingeniería en Ciencias Aplicadas

Carrera de Software

Implementación de un balanceador de carga en el Cloud FICA de la Universidad Técnica del Norte para fortalecer la alta disponibilidad del servidor de aplicaciones Wildfly

Trabajo de grado previo a la obtención del título de Ingeniero de Software presentado ante la ilustre Universidad Técnica del Norte.

Autor:

Guamán Cupacán Byron Ramiro

Director:

MSc. Xavier Mauricio Rea Peñafiel

Ibarra – Ecuador

2023

UNIVERSIDAD TÉCNICA DEL NORTE

BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	0402041503		
APELLIDOS Y NOMBRES:	GUAMÁN CUPACÁN BYRON RAMIRO		
DIRECCIÓN:	Ibarra, Imbabura.		
EMAIL:	brguamanc@utn.edu.ec		
TELÉFONO FIJO:	N/A	**TELÉFONO MÓVIL:**	0994944502

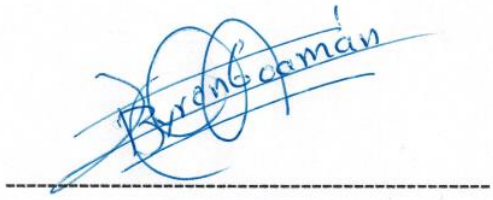
DATOS DE LA OBRA	
TÍTULO:	IMPLEMENTACIÓN DE UN BALANCEADOR DE CARGA EN EL CLOUD FICA DE LA UNIVERSIDAD TÉCNICA DEL NORTE PARA FORTALECER LA ALTA DISPONIBILIDAD DEL SERVIDOR DE APLICACIONES WILDFLY.
AUTOR(ES):	BYRON RAMIRO GUAMÁN CUPACÁN
FECHA:	04/09/2023
PROGRAMA:	PREGRADO
TÍTULO POR EL QUE OPTA:	INGENIERO DE SOFTWARE
DIRECTOR:	MSc. Xavier Mauricio Rea Peñafiel
ASESOR 1:	PhD. José Antonio Quiña Mera

2. CONSTANCIAS

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

Ibarra, a los 04 días del mes de septiembre de 2023

EL AUTOR:



ESTUDIANTE

Byron Ramiro Guamán Cupacán
C.I: 0402041503

CERTIFICACIÓN DIRECTOR

Ibarra 04 de septiembre del 2023

CERTIFICACIÓN DIRECTOR DEL TRABAJO DE TITULACIÓN

Por medio del presente yo MSc. Rea Mauricio, certifico que el Sr. Byron Ramiro Guamán Cupacán portador de la cédula de ciudadanía número 0402041503, ha trabajado en el desarrollo del proyecto de grado: **"Implementación de un balanceador de carga en el Cloud FICA de la Universidad Técnica del Norte para fortalecer la alta disponibilidad del servidor de aplicaciones Wildfly",** previo a la obtención del Título de Ingeniero en Software realizado con interés profesional y responsabilidad que certifico con honor de verdad.

Es todo en cuanto puedo certificar a la verdad

Atentamente

Firmado electrónicamente por:
XAVIER MAURICIO REA PEÑAFIEL

_____

MSc. Xavier Rea Mauricio Peñafiel

DIRECTOR DE TRABAJO DE GRADO

DEDICATORIA

De manera especial, dedico el presente trabajo de grado a mis padres, Martha y Ramiro. Sinceramente me encuentro agradecido con ustedes por cada sacrificio que han realizado durante todo este proceso educacional. Agradezco a mi madre por cuidarme, entregarme su gran cariño y brindarme su bendición a pesar de la distancia. De igual forma, quiero reconocer a mi padre por su constante esfuerzo y dedicación en su trabajo, asegurando que nunca nos faltara nada. Es por estas razones y muchas más, en este texto deseo manifestar que este logro también les pertenece, y que me siento orgulloso de compartir esta importante meta con dos personas tan valiosas y fundamentales en mi vida como lo son ustedes.

También le dedico este trabajo a mi hermana Mayra por apoyarme y ser un gran ejemplo en mi vida desde siempre, y por demostrarme y enseñarme que con esfuerzo y humildad todo es posible. Reconozco que gracias a ti, he aprendido valiosas lecciones que me han ayudado a crecer como persona, y estoy muy agradecido por contar con una hermana como tú.

También le quiero dedicar a este esfuerzo a mi compañera de vida Paola, por tomar mi mano y apoyarme en cada paso de este largo camino. Gracias por brindarme un segundo hogar cuando más lo necesite, por ser mi fuente de fortaleza e inspiración, y por enseñarme a creer en mis capacidades y motivarme a creer que los sueños se pueden hacer realidad si luchas por ellos.

Byron Ramiro Guamán Cupacán

AGRADECIMIENTO

Agradezco a la Universidad Técnica del Norte por abrirme sus puertas y formarme como un profesional de calidad, a mi tutor MSc. Mauricio Rea por compartir su conocimiento y experiencia en el desarrollo de este trabajo, como también por su confianza en mi persona y sus palabras de motivación que me ayudaron a seguir adelante incluso en los momentos más desafiantes. De igual forma agradezco a cada uno de los docentes de la Carrera de Ingeniería en Software por su paciencia y dedicación dentro del aula, su compromiso con la enseñanza y su pasión por transmitir conocimiento que nos ha inspirado a crecer y aprender cada día.

Agradezco a mi familia por todo el apoyo brindado durante esta etapa, por brindarme un hogar y un plato de comida a donde quiera que vaya, por bríndame un consejo y una mano amiga con quien contar. Cada uno de ustedes aportó de manera significativa para que pudiera lograr esta meta, y me siento feliz de contar con una familia unida para todas las situaciones que se puedan presentar.

Agradezco a mis amigos por todas las aventuras que hemos compartido dentro y fuera del aula, por los almuerzos y tardes de risa juntos, por sus mensajes de ánimo y por ser una gran compañía. Sinceramente espero seguir contando con su amistad y también quiero desearles que nunca se rindan, ya que de igual forma quiero que cumplan cada una de sus metas y sueños, y estar ahí para celebrar juntos.

Y por último y no menos importante, quiero extender un agradecimiento a las personas que ya no se encuentran conmigo, ya que en vida dejaron una huella importante en mí y siempre las llevaré en mi corazón.

Byron Ramiro Guamán Cupacán

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE CUADROS

RESUMEN

El presente trabajo tiene como objetivo implementar un balanceador de carga en el Cloud FICA de la Universidad Técnica del Norte para fortalecer la alta disponibilidad del servidor de aplicaciones Wildfly.

Inicialmente, el documento presenta la introducción donde se detallan los antecedentes, el planteamiento del problema, los objetivos, el alcance, la metodología a seguir y la justificación del proyecto.

En el capítulo uno se realizó la base teórica del proyecto que está compuesta por conceptos fundamentales acerca de la alta disponibilidad en servidores de aplicaciones y balanceadores de carga, como también una revisión a la norma ISO/IEC 25010 y la descripción de diferentes trabajos relacionados.

En el capítulo dos se realizó el desarrollo de la implementación, donde se detalló los materiales y métodos utilizados, el proceso de la implementación del balanceador de carga y el proceso de desarrollo y validación de la guía de alta disponibilidad para Wildfly.

En el capítulo tres se detallan las pruebas y la interpretación los resultados obtenidos, como también la validación de estos mediante la aplicación de las métricas de la subcaracterísticas de la disponibilidad de la norma ISO/IEC 25010.

Finalmente, se describen las conclusiones, recomendaciones, bibliografía y los diferentes apéndices del proyecto.

ABSTRACT

The objective of this work is to implement a load balancer in the FICA Cloud of the Universidad Técnica del Norte to strengthen the high availability of the Wildfly application server.

Initially, the document presents the introduction where the background, the problem statement, the objectives, the scope, the methodology to be followed and the justification of the project are detailed.

In chapter one, the theoretical basis of the project was developed, which is composed of fundamental concepts about high availability in application servers and load balancers, as well as a review of the ISO/IEC 25010 standard and the description of different related works.

In chapter two, the development of the implementation was carried out, detailing the materials and methods used, the process of implementing the load balancer and the development and validation process of the high availability guide for Wildfly.

Chapter three details the tests and the interpretation of the results obtained, as well as the validation of these through the application of the metrics of the availability sub-characteristics of the ISO/IEC 25010 standard.

Finally, the conclusions, recommendations, bibliography and the different appendices of the project are described.

INTRODUCCIÓN

Antecedentes

En los últimos años según Kone (2021), se hace énfasis en las tecnologías de alta disponibilidad debido a que los usuarios esperan que los sistemas funcionen continuamente, lo cual implica que estén disponibles 24/7, o también que estos presenten un tiempo de inactividad mínima. La pérdida o interrupción de un servicio es inaceptable y podría generar significativas pérdidas económicas, además de generar inconformidad por parte de los usuarios, razón por la cual ciertas organizaciones desarrollan e implementan sofisticados sistemas informáticos y de comunicación donde la alta disponibilidad es esencial (Sinisterra et al., 2012).

Para una organización, contar con una configuración que garantice un alto nivel de disponibilidad de sus servicios es una ventaja para sus usuarios, y ante la presencia de un fallo o un alto nivel de solicitudes, la alta disponibilidad garantizará que el sistema tenga una gran capacidad tolerancia a fallos y de recuperabilidad, que implica operar mientras se presenten incidentes y de restablecer el estado del sistema en un corto periodo de tiempo con el objetivo continuar con el flujo normal de trabajo.

Planteamiento del Problema

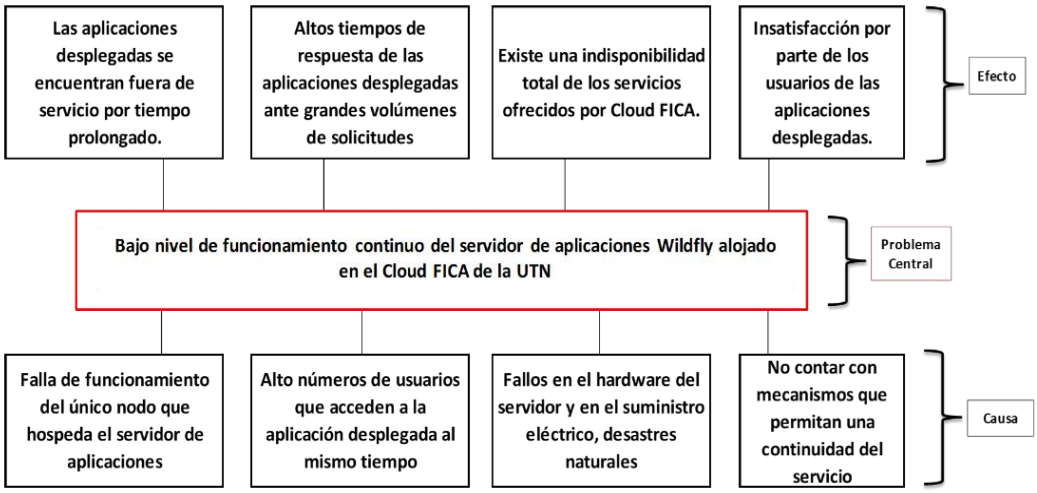
Actualmente la Facultad de Ingeniería y Ciencias Aplicadas (FICA) de la Universidad Técnica del Norte (UTN) cuenta con un entorno virtualizado denominado Cloud FICA, el cual está gestionado mediante el software Proxmox VE, y es utilizado para establecer varias máquinas virtuales con el fin de brindar diferentes servicios (Vizcaíno, 2021). Entre los servicios en funcionamiento se encuentra el servidor de aplicaciones Wildfly para el despliegue de aplicaciones Java Enterprise, como es el caso del Sistema Integrado de Actividad Docente (SIAD), el cual tiene como objetivo optimizar el seguimiento de las funciones de la docencia (Morillo, 2021).

En un proceso de observación se ha demostrado que el servidor de aplicaciones Wildfly desplegado en el Cloud FICA no cuenta con una configuración propuesta que garantice una alta disponibilidad del servicio, en consecuencia, se evidencia inconsistencias en la

funcionalidad y pérdidas de disponibilidad del servidor de aplicaciones por tiempos prolongados, como se evidencia en la Figura 1, por lo cual, se plantea implementar un balanceador de carga, el cual es una técnica de alta disponibilidad que permite al usuario tener respuestas oportunas de la aplicación, incluso en grandes presencias de volúmenes de solicitudes (Wildfly, 2018).

Figura 1

Diagrama del problema



Objetivos

Objetivo General

Implementar un balanceador de carga en el Cloud FICA de la Universidad Técnica del Norte para fortalecer la alta disponibilidad del servidor de aplicaciones Wildfly.

Objetivos Específicos

- Crear una base teórica en función a la alta disponibilidad y balanceo de carga para un servidor de aplicaciones.
- Desplegar el balanceador de carga en el Cloud FICA de la Universidad Técnica del Norte para el servidor de aplicaciones Wildfly.
- Desarrollar una guía de implementación de alta disponibilidad del servidor de aplicaciones Wildfly.
- Verificar el funcionamiento de la técnica implementada mediante pruebas de rendimiento a una aplicación desplegada, y validar los resultados obtenidos en base de la subcaracterística de disponibilidad del estándar ISO/IEC 25010.

Alcance

Este proyecto tiene como finalidad implementar un balanceador de carga para el servidor de aplicaciones Wildfly alojado en el Cloud FICA de la Universidad Técnica del Norte. Esta técnica de alta disponibilidad se implementará mediante una instalación de Wildfly y el subsistema Modcluster. La tecnología de Modcluster, utiliza la conexión de cada nodo para transmitir los factores de equilibrio de carga al lado del servidor y los eventos del ciclo de vida a httpd mediante un conjunto de métodos HTTP, denominados como Mod-Cluster Management Protocol (Modcluster, 2023). En esta técnica, es necesario la creación de un grupo de servidores Wildfly, donde se establecerá los diferentes nodos denominados como máster y esclavo, que en conjunto con el balanceador de carga, permitirá una distribución de peticiones y mantener una disponibilidad del servicio en caso de que alguna instancia falle.

Para el desarrollo de implementación se estableció tres arquitecturas con requisitos, y escenarios diferentes. En la primera arquitectura de la implementación se definirá un entorno virtualizado local mediante el software KVM, el cual es una solución de virtualización completa para Linux (KVM, 2016) y permitirá establecer las máquinas virtuales necesarias para desplegar la técnica propuesta. Posteriormente se realizará la configuración de las instancias del servidor de aplicaciones Wildfly dentro del grupo se servidores para su integración con el balanceador de carga, y de esta manera, comprobar su correcto funcionamiento mediante una aplicación de demostración, y así verificar que se realiza la distribución de cargas entre los diferentes nodos, como también verificar que se mantiene la disponibilidad del sistema simulando la indisponibilidad de cada una de las diferentes instancias del servidor de aplicaciones.

En la segunda arquitectura se accederá al Cloud FICA, el mismo que está gestionado mediante el software Proxmox VE, y así establecer los nodos necesarios para replicar la implementación de alta disponibilidad entre el balanceador de carga y los servidores de aplicaciones Wildfly, donde sus configuraciones y pruebas serán tomadas de la arquitectura inicial, mencionada anteriormente. Adicionalmente, existirá una tercera arquitectura donde se comprobará el funcionamiento continuo del servidor de aplicaciones Wildfly mediante la

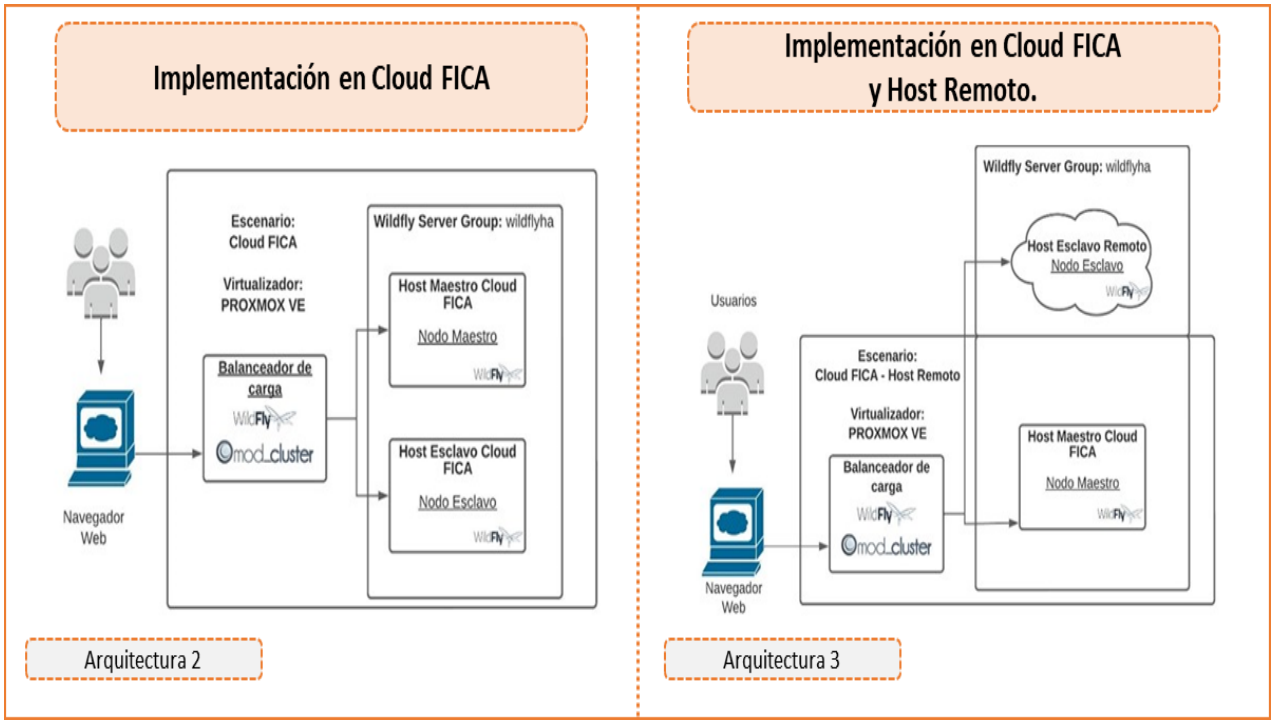
creación y configuración de un nodo secundario en una máquina virtual alojada en un host remoto, completando así la implementación, tal como se muestra en la arquitectura 3 de Figura 2.

En todas las arquitecturas descritas anteriormente, el balanceador de carga y los servidores de aplicaciones Wildfly se desplegarán sobre el sistema operativo CentOS Stream 8, el cual es una distribución de Linux con características de ser una plataforma consistente y manejable de código abierto (Centos, 2023). Conjuntamente, todas las configuraciones y pasos de instalación de la arquitectura final se documentarán en una “Guía de implementación de alta disponibilidad del servidor de aplicaciones Wildfly”.

Para la realización de pruebas se utilizará el software Apache JMeter, el cual es un software de código abierto y servirá para cargar el comportamiento funcional y medir el rendimiento de aplicaciones web, entre otras funciones de prueba (Apache Software Foundation, 2022). En la sección de pruebas de rendimiento se tomará como referencia pruebas de carga y estrés, para la posterior validación de los resultados bajo las métricas de la subcaracterística de disponibilidad de la norma ISO/IEC 25010.

Figura 2

Arquitecturas del proyecto



Metodología

Para el cumplimiento del primer objetivo del presente proyecto, se realizará una búsqueda y análisis de información mediante fuentes bibliográficas, publicaciones científicas y guías oficiales de instalación, en las que se detalle una conceptualización a la alta disponibilidad y al balanceo de carga, en relación con los servidores de aplicaciones.

Para el segundo objetivo se llevará la implementación del balanceador de carga utilizando la metodología Kanban, la cual permitirá visualizar el trabajo, limitar la acumulación de tareas, con el fin de maximizar la eficiencia (Mesh, 2020). Este desarrollo será guiado bajo el cumplimento de las tres arquitecturas propuestas, que se definirán a continuación:

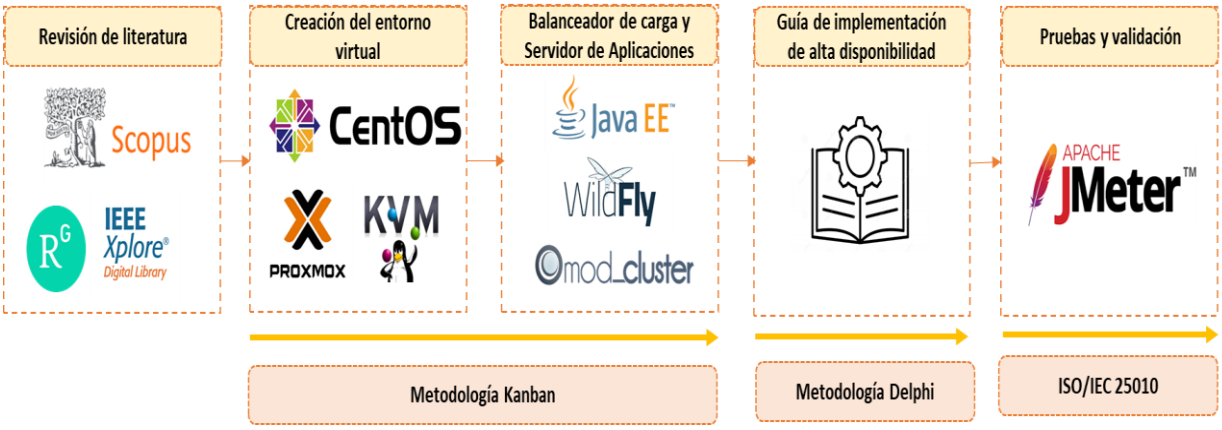
- En la primera arquitectura se creará un entorno mediante máquinas virtuales locales, para así, desplegar la técnica y comprobar su correcto funcionamiento.
- En la segunda arquitectura se accederá al Cloud FICA y se implementará la técnica propuesta mediante la creación de máquinas virtuales dentro del entorno virtualizado.
- En la tercera arquitectura se desplegará un nodo secundario en un host remoto, y así comprobar que existe una alta disponibilidad del servicio fuera del Cloud FICA.

En el tercer objetivo se realizará una guía de implementación de alta disponibilidad de Wildfly que constará como artefacto de la investigación presente, la cual contendrá los pasos, comandos y configuración necesaria, con la finalidad de que sirva como modelo para los presentes y futuros administradores del Cloud FICA, y que adicionalmente, será validada mediante la metodología Delphi, la cual se fundamenta en buscar un consenso más fiable mediante el criterio y opinión de un grupo de expertos (Cuesta, 2013).

Finalmente, en el cuarto objetivo se comprobará el correcto funcionamiento del balanceador de carga mediante pruebas de rendimiento (pruebas de carga y estrés), utilizando como referencia a la aplicación Java EE: SIAD y el software de pruebas Apache JMeter, y finalmente validar los resultados obtenidos mediante la subcaracterística de disponibilidad, correspondiente a la característica de fiabilidad contenidas en la norma ISO/IEC 25010, en donde se toma en cuenta la capacidad del sistema de estar operativo y accesible para su uso cuando se requiere (ISO25000, 2022)

Figura 2

Desarrollo del proyecto



Justificación

El presente proyecto de titulación se enfoca al Objetivo de Desarrollo Sostenible (ODS) Nro. 9: “Construir infraestructura resiliente, promover la industrialización inclusiva y sostenible y fomentar la innovación”, planteado por la ONU y UNESCO(Naciones Unidas, 2022) ,en donde se buscará implementar técnicas de alta disponibilidad que permitirán fortalecer el funcionamiento continuo de la infraestructura existente y así brindar un servicio de calidad a los usuarios de las aplicaciones desplegadas.

Dentro de este objetivo se cumplen las siguientes metas:

- 9.b. Apoyar el desarrollo de tecnologías, la investigación y la innovación nacionales en los países en desarrollo, incluso garantizando un entorno normativo propicio a la diversificación industrial y la adición de valor a los productos básicos, entre otras cosas
- 9.c. Aumentar significativamente el acceso a la tecnología de la información y las comunicaciones y esforzarse por proporcionar acceso universal y asequible a Internet en los países menos adelantados de aquí a 2020 (Naciones Unidas, 2022).

Justificación Tecnológica.

En este proyecto, los usuarios finales dependen directamente de las aplicaciones desplegadas para la realización de sus actividades académicas, razón por la cual estas herramientas tecnológicas deberán tener un funcionamiento continuo que se conseguirá mediante la implementación de una técnica de alta disponibilidad dentro del entorno

virtualizado. Igualmente, en el desarrollo de este proyecto se realizará la creación de una guía de implementación que servirá para poder replicar la configuración de alta disponibilidad en otros entornos virtualizados y ser una orientación para futuros proyectos tecnológicos.

Justificación Académica.

El proyecto buscará resolver problemas de insatisfacción por parte de los docentes y estudiantes los cuales son usuarios de las aplicaciones Java EE alojadas en servidor de aplicaciones del Cloud FICA, y así, brindar un entorno donde se pueda desplegar un sistema completamente funcional y que esté disponible las 24 horas del día.

CAPÍTULO 1

Marco Teórico

1.1 Servidor de Aplicaciones

1.1.1 Definición de Servidor de Aplicaciones

Un servidor de aplicaciones es un programa de servidor en redes distribuidas que tienen el objetivo de proporcionar un entorno que permita realizar las configuraciones necesarias para la ejecución de una aplicación (IBM, 2021ª). El servidor de aplicaciones en conjunto con el servidor web ofrecen una respuesta dinámica y personalizada para cada solicitud del cliente.

También se lo puede definir como un contenedor de aplicaciones que cuenta con una configuración centralizada para un despliegue más eficiente, y por lo general, la administración del servidor de aplicaciones se lo dejará a un grupo especializado que estará a cargo de desplegar la infraestructura necesaria para su funcionamiento y que permitirá una disminución en el tiempo de desarrollo de aplicaciones (Parra, 2016).

Según Saau & Germán (2013), los servidores de aplicaciones gestionan la lógica del negocio y el acceso de datos a las aplicaciones, son un intermediario para la seguridad y el mantenimiento, y además presentan funciones multiproceso y multihilo para atender distintas peticiones, como también enviar solicitudes entre un conjunto de máquinas conectadas entre sí, denominado clúster, con el objetivo de tener funciones de balancero de carga y disponibilidad.

Para (Sánchez, 2011), los servidores de aplicaciones, que además de atender peticiones http, permiten entender instrucciones de lenguajes avanzados y traducirlos, así como también acceder a recursos de otros servidores, donde el proceso empieza cuando el usuario generalmente desde su navegador requiere el servicio y el servidor de aplicaciones atiende la petición e interpreta el código de la aplicación con el fin de traducirla y mostrar al usuario de forma entendible en su máquina.

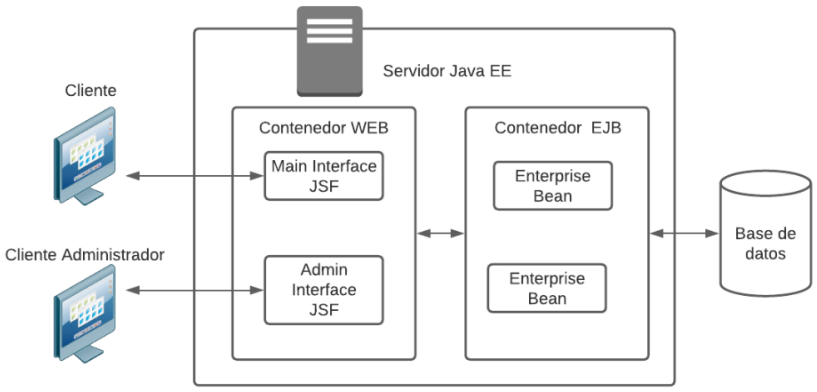
1.1.2 Servidores de Aplicaciones Java Enterprise

La tecnología Java Enterprise Edition (Java EE), permiten el desarrollo de aplicaciones a nivel empresarial que pueden ser desplegadas en una serie de servidores que cumplan con el estándar requerido para el mismo. Estos servidores de aplicaciones Java EE tienen la ventaja de ser multiplataforma por ser basados en Java, permitiendo que puedan utilizar las aplicaciones en todas las arquitecturas de Windows y Linux sin necesidad de cambios significativos en su configuración, como también tener la capacidad necesaria para ser una plataforma completa, estable, segura y rápida (Vera, 2018).

El modelo de un servidor Java EE, como se muestra en la Figura 3, se define principalmente por un contender web que permite la ejecución de los Servlets, el cual es un programa Java que se ejecuta en el servidor y recibe una consulta por parte del cliente, realiza las operaciones y las devuelve, y así generar páginas web de forma dinámica. También cuenta con los contenedores Enterprice Java Beans (EJB), lo cuales proporcionan un modelo de componentes estándar del lado del servidor, y entre ellos pueden ser: un objeto remoto (EJB Session), un objeto que garantice el mapping objeto/relacional (EJB Entity), y además los EJB que permiten realizar operaciones asíncronas (EJB Message).

Figura 3

Modelo de servidor Java EE



Existe una diversidad de servidores de aplicaciones Java EE, como se muestra en la Figura 4, y según Zamora (2015), define que entre los servidores de aplicaciones Java EE más importantes en el mercado son:

- Comerciales: Oracle WebLogic, IBM WebSphere, Red Hat JBoss EAP.

- Open Source: Oracle GlassFish, WildFly.

Figura 4

Servidores de aplicaciones Java EE

Nota: Adaptado de *Servidores web y PaaS* [Fotografía], por Luis Zamora, 2015, Experto Java (http://expertojava.ua.es/experto/restringido/2015-16/paas/paas01.html).

1.1.3. Servidor de Aplicaciones Wildfly

Wildfly en su versión 26.1.3, es un servidor de aplicaciones de código abierto patrocinado por Red Hat ™, el cual es un proyecto preliminar a JBoss Enterprise Application Plataform (JBoss EAP). Según su portal oficial Wildfly (2023), presenta sus principales características:

- Potente: Define a la configuración de Widlfy como centralizada simple y centrada en el usuario, ya que está organizado por subsistemas para una mayor comprensión, y las diferentes formas de administración se presentan de manera unificada, como una consola de administración basada en Web, una CLI, una API Java nativa, una API REST basada en HTTP/JSON y una puerta de enlace JMX.
- Modular: Utiliza JBoss Modules para proporcionar un verdadero aislamiento de la aplicación, que consiste en ocultar las clases de la implementación del servidor de aplicaciones y enlazando únicamente con los JAR que la aplicación necesita, consiguiendo una mayor optimización en la carga de clases.

- Ligero: Adopta un enfoque agresivo para la gestión de memoria, donde se minimiza la asignación de pila utilizando metadatos indexados generalmente en la caché, como también una consola de administración 100% independiente que se inicia instantáneamente y no requiere memoria del servidor, logrando que Wildfly funcione con la configuración JVM estándar y en dispositivos pequeños, y dejando más espacio para los datos de la aplicación y tener mayor escalabilidad.
- Basado en estándares**:** Implementa lo último en estándares Java empresariales, mejorando la productividad de los desarrolladores y generando marcos de trabajo que eliminan la repetición de tareas y la reducción la carga técnica.

1.2 Alta Disponibilidad

1.2.1 Definición de Disponibilidad

Primeramente, se define la característica de disponibilidad, como la capacidad de un servicio o de un sistema a ser accesible y utilizable por los usuarios cuando estos los requiera, y que la información pueda ser recuperada siempre que se la necesite, tomando en cuenta que se deberá evitar su pérdida o su bloqueo (Santos, 2014).

La disponibilidad en conjunto con la confidencialidad y la integridad, son un conjunto de características que definirán a un sistema como seguro o fiable, y que dependiendo del sistema que se trabaje se dará la prioridad a una característica en específico. Dentro aplicaciones que se consideran críticas, o aplicaciones en donde la interrupción de sus servicios se considere inaceptables por el flujo de trabajo que conllevan, se hace referencia a un término denominado alta disponibilidad.

1.2.2 Definición de Alta disponibilidad

La alta disponibilidad se define como un conjunto de técnicas que tienen como objetivo tener una funcionalidad de un servicio de forma continua e ininterrumpida, es decir, que se asegurará que el servicio funcione 24/7, y en el caso de la existencia de fallos, estos sean solventados de una forma eficiente y transparente al usuario.

Para Kone (2021), la alta disponibilidad es el conjunto de recursos compartidos en todo el sistema que cooperan para garantizar los servicios esenciales, y que combina software

con hardware con el objetivo de minimizar el tiempo de inactividad y de restaurar los servicios en segundos cuando un componente falla.

El concepto de alta disponibilidad en una aplicación se define según Gómez & León (2010), como el inicio automático de un servicio en otra máquina de un conjunto de máquinas, en caso de la existencia de fallos en el hardware o de la aplicación, y que posteriormente el servicio será nuevamente migrado a la máquina original.

1.2.3 Beneficios de la Alta Disponibilidad

La alta disponibilidad permitirá un flujo de trabajo continuo, ya que va a reducir al mínimo el riesgo de interrupción del servicio, pérdida de datos transaccionales, tener datos incompletos o errores de procesamiento de mansajes (Kone, 2021).

Según IBM (2021b), hace referencia que la satisfacción del usuario se puede ver afectada negativamente por tiempos de inactividad frecuentes o de larga duración, y mediante la implementación de soluciones de alta disponibilidad, los servicios de hardware y software pueden restaurarse cuando se produce un error, a menudo en menos de un minuto, conllevando a un flujo de trabajo continuo, donde los usuarios no van a notar que se ha producido algún problema. De igual manera, considera que la implementación de la alta disponibilidad proporciona a la organización contar con un tiempo de espera mínimo y realizar ahorros potenciales en situaciones donde el costo del tiempo de inactividad es alto.

En el estudio realizado por Perafan et al., (2018), demostró que la implementación de soluciones de alta disponibilidad brinda protección ante posibles vulnerabilidades en la corrupción de datos, así como una mayor escalabilidad que permite la adición de otro nodo al clúster que permite dependiendo de la robustez que se necesite, y finamente en la fase de pruebas se evidenció la ausencia de fenómenos de congestión del servidor frecuentes.

Sinisterra et al., (2012), consideran que la alta disponibilidad es fundamental dentro de la configuración de aplicaciones web de mucho tráfico, ya que ante cualquier fallo dejaría al servidor completamente inutilizable, y el uso de nodos de respaldo permitiría reparar el servicio mientras los usuarios navegan y utilizan las funcionalidades de la aplicación con normalidad.

Dentro de un servidor de aplicaciones, como es el caso de Wildfly, los servicios de alta disponibilidad permitirá garantizar la continuidad funcional de las aplicaciones Java EE implementadas, donde se va a cubrir problemas fundamentales como la pérdida de disponibilidad de la aplicación cuando del nodo que la aplicación falla, así como la pérdida de disponibilidad de la aplicación cuando existen retrasos en el tiempo de respuesta cuando el servidor tiene una gran cantidad de volúmenes de solicitudes. (Wildfly, 2018).

1.2.4 Indisponibilidad de un Servicio

La indisponibilidad se define como la consecuencia de una falla de un sistema, y se presenta como la pérdida del acceso al servicio solicitado. Estas fallas producen tiempos de inactividad, que se presentan como planificadas o no planificadas, y se detallan a continuación:

El tiempo de inactividad planificado, por lo general, es el resultado de un mantenimiento al sistema, o de algún evento anteriormente planeado, como por ejemplo la implementación de un parche al software del sistema o de cambios en la configuración que requieran el reinicio del sistema.

El tiempo de inactividad no planificado es un gran problema que afecta a la disponibilidad de un servicio, y se presentan por alguna incidencia como fallos de potencia, fallos en los componentes de hardware, una caída por recalentamiento, una ruptura lógica o física en las conexiones de red, rupturas de seguridad catastróficas o fallos en el sistema operativo y aplicaciones(Gómez & León, 2010). Otros factores, como tener un alto número de usuarios a una aplicación, puede conllevar al colapso de la capacidad de un servidor de aplicaciones, conllevando a un tiempo de respuesta muy alto o una indisponibilidad total del servicio, produciendo que se tenga un tiempo de inactividad no planificado.

Un objetivo de la alta disponibilidad es ofrecer un 99.999% de tiempo de actividad del sistema, lo cual indicaría un tiempo de inactividad de 25 segundos al mes, y de 50 milisegundos al día, como se muestra en la Tabla 1. Este nivel de disponibilidad está destinado para los sectores más importantes como la salud, gobierno y servicios financieros que necesitan de este nivel de disponibilidad por razones de cumplimiento normativo o de

competencia (Red Hat, 2022b). Por lo general, las empresas optan por ofrecer un nivel del servicio que va desde el 99% al 99,9% de nivel de servicio, como es el caso de Google Cloud, que ofrece el 99.9% de disponibilidad para la interconexión dedicada, y que consideran un nivel adecuado para las aplicaciones no fundamentales que puedan tolerar cierto nivel de inactividad (Google Cloud, 2022), como también Azure de Microsoft detalla en su Acuerdo de Nivel de Servicio (SLA) un tiempo de actividad y conectividad de un 99.95% para las aplicaciones web de App Service , y un 99.99% para su servicio de SQL Database (Microsoft, 2022).

Tabla 1

Niveles de disponibilidad de un servicio

Nivel de disponibilidad	**Tiempo de inactividad por mes.**	**Tiempo de inactividad por día.**
99%	7 horas 18 minutos	11 minutos 36 segundos
99.9%	43 minutos 50 segundos	1 minuto 27 segundos
99.99%	4 minutos 24 segundos	8 segundos 49 milisegundos
99.999%	25 segundos	50 milisegundos

1.3 Balanceo de carga

1.3.1 Definición de Balanceo de Carga

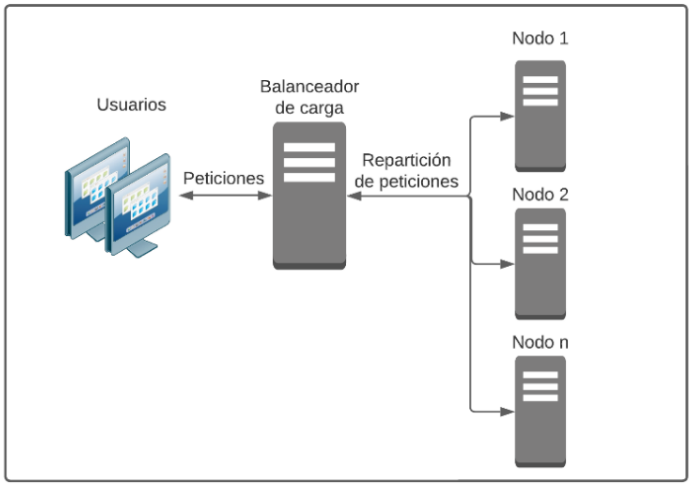
Según Guzmán et al., (2021), el balanceo de carga es un servicio que provee alta disponibilidad por medio de detección automática de fallas de servidores y la repartición del tráfico de clientes a través de servidores restantes, mientras se provee al usuario un servicio continuo. Además, se tiene la característica de ser un servicio virtual de total transparencia para los usuarios finales, y que proporcionará tiempos de respuestas más rápidos.

Una arquitectura que cuente con un servicio de balanceo de carga está compuesta por varios nodos, tal como se muestra en la Figura 5, que actúan como la cara visible o front-end de este, y que se encargaran de repartir las peticiones de servicio entrantes en los diversos

nodos existentes (Gutiérrez & Benítez, 2013). Esta disposición tiene ventanas significativas como una mayor escalabilidad, ya que se podrá ir sumando nodos al sistema y obtener una mayor capacidad de respuesta frente a peticiones, así como ofrecer una mayor tolerancia a fallas ante la caída de uno de los nodos ya el servicio seguirá funcionando en los nodos que queden disponibles.

Figura 5

Arquitectura básica de un sistema con balanceador de carga



Para Niño (2020), el funcionamiento de un balanceador de carga se centra en equipos que reciben peticiones, recogen información en tiempo real de la capacidad operativa actual de los equipos, y posteriormente se enrutan dichas peticiones de forma individual al servidor que se encuentre en una mejor condición para prestar el servicio adecuado.

Entre las características más principales de un balanceador de carga presentadas por Gómez y León (2010), se tiene evitar la saturación de una máquina, donde se va a procurar que los distintos procesos que accedan a las máquinas afecten al funcionamiento normal del aplicativo, así como también gestionar los recursos computacionales de manera inteligente, ya que se va a optimizar todos los recursos disponibles dando como resultado un acceso más rápido y estable a los aplicativos.

1.3.2 Beneficios de Implementar un Balanceador de Carga

Entre los principales beneficios de un balanceador de carga presentados por SIMÉXICO (2016), se tiene:

- Escalabilidad: El balanceador de carga distribuye las peticiones entre varios servidores, haciendo la capacidad global del proceso y servicio crezca a comparación de trabajar con un solo servidor.
- Disponibilidad: Este servicio monitoriza el estado de los servidores y aplicaciones, de forma que, si existen fallos en uno de estos, se dejará de enviar peticiones.
- Mantenimiento: En el caso ser necesario apagar una máquina para actualizarla o repararla, el resto del conjunto seguirá dando el servicio, para posteriormente añadirla al grupo cuando se encuentre operativa.
- Seguridad: Se considera como una primera línea de defensa, rechazando diferentes tipos de ataques.
- Calidad de servicio: La implementación de un balanceador de carga genera un mejor tiempo de respuesta, y de la capacidad de ofrecer servicios en función del tipo de usuario.

1.3.3 Tipo de Balanceadores de Carga

Un balanceador de carga se puede presentar tanto como un dispositivo físico, el cual se trata de una instancia virtualizada que se ejecuta en hardware especializado, así como un producto de software diseñados para mejorar efectivamente el rendimiento y la seguridad de las aplicaciones web, independientemente dónde estén alojadas (Bastidas, 2021). Cada una de las soluciones presentan diferentes funcionalidades importantes, entre las que podemos encontrar:

- Balanceo de carga por hardware: Son servidores dedicados únicamente para realizar la tarea de balanceo, excluyéndolos de otros servicios o propósitos. Esta característica les permite tener una gran potencia y estabilidad dentro de la infraestructura, Sin embargo, esta solución tiene ciertas desventajas como aumentar el precio de montaje de los data centers de las organizaciones, exigir un mantenimiento muy alto, así como también funcionar como una caja cerradura, lo cual conlleva al desarrollador no poder manipular el código fuente ni la estructura (Pazmiño, 2014).

- Balanceo de carga por software: Son servidores configurados para realizar la tarea de balanceo, para lo cual inicialmente el servidor debe tener un sistema operativo y una aplicación dedicada para el balanceo. Generalmente se denomina balanceadores de carga en capa 4 del modelo OSI, la cual hace referencia a la capa que se encarga de administrar la transferencia de datos, así como garantizar que los datos recibidos sean idénticos a los transmitidos (Oracle, 2010). Entre sus mayores ventajas es reducir costes económicos a comparación de las soluciones basadas en hardware.

1.3.4. Algoritmos de Balanceo de Carga

Dentro de las soluciones de balanceo de carga por software, se distribuye las peticiones a los diferentes servidores en base de un algoritmo, y según la investigación de Hurtado (2011), tenemos los siguientes:

- Round Robin (RR): Se considera como una técnica simple que asegura el envío de cada solicitud de cliente a un servidor diferente de forma rotativa, como por ejemplo, si se tiene tres servidores: A, B y C, se enviará las solicitudes primero al servidor A, luego al B, luego al C, y así sucesivamente. Este algoritmo tiene la desventaja de no tener en cuenta la carga existente en el servidor, lo cual conlleva a que un servidor reciba muchas solicitudes y se sobrecargue.
- Weighted Round Robin (WRR): Este algoritmo funciona de una manera similar a Round Robin, pero asigna más peticiones a los nodos con un mayor prioridad o también denominado peso, provocando durante un período de tiempo que los nodos recibirán un número de solicitudes en proporción a su peso. Por ejemplo, si se tienen 3 servidores: A, B y C, y sus prioridades son 4, 3 y 2, las conexiones se efectuarían de la siguiente forma: AABACABBC, donde A hace 4 conexiones, B hace 3 conexiones y C hace 2 conexiones y todo de una forma secuencial.
- Least-Connection (LC): Utiliza el método de menos conexiones, donde el equilibrador de carga selecciona el servicio utilizando el valor (N), que corresponde al número de transacciones activas, distribuyendo las peticiones a los servidores que menos cargas activas tengan

- Weighted Least-Connection (WLC): Este algoritmo efectúa las conexiones por prioridad a un porcentaje, donde se le define un número para cada servidor, el cual servirá para realizar operaciones para asignar cuantas conexiones van a poder soportar cada equipo hasta pasar al siguiente en la lista de servidores.
- Locality Based Least-Connection (LBLC): Este algoritmo se queda con la dirección de destino, y si el servidor con el que se está efectuando operaciones se encuentra demasiado cargado, lo destina a otro servidor con menos carga.
- Locality Based Least-Connection with Replication (LBLCR): Recopila la dirección destino, y comprueba que servidor está menos cargado, y en el caso de que todos los servidores están cargados, se espera hasta poder introducir la solicitud a algún servidor que deje alguna conexión libre y esté menos cargado.
- Destination-Hashing (DH): Este algoritmo asigna a la IP de destino un servidor específico haciendo un hash de la conexión.
- Source-Hashing (SH): Este algoritmo asigna a la IP de origen de un servidor específico haciendo un hash de la conexión.

1.3.5 Balanceador de Carga Modcluster

Según su documentación oficial (Modcluster, 2023), lo define como un equilibrador de carga inteligente, y que al igual que otros balanceadores de carga como mod_jk y mod_proxy, Modcluster utiliza un canal de comunicación para el reenvío de solicitudes httpd a un conjunto de nodos del servidor de aplicaciones. Este balanceador aprovecha una conexión adicional entre los nodos del servidor de aplicaciones y httpd, que es utilizada para transmitir factores de equilibrio de carga y eventos del ciclo de vida del lado del servidor a través de un conjunto personalizado de métodos HTTP, denominados como Mod-Cluster Management Protocol (MCMP). Este balanceador cuenta con una serie de ventajas sobre otros balanceadores de carga basados en httpd, como:

- Configuración dinámica de trabajadores httpd: La mayor parte de la configuración del proxy reside en los servidores de aplicaciones, ya que este transmite eventos del ciclo

de vida (p. ej., inicio/apagado del servidor) a los proxies, lo que les permite autoconfigurarse de manera efectiva.

- Cálculo del factor de equilibrio de carga del lado del servidor: Utiliza factores de equilibrio de carga calculados y proporcionados por los servidores de aplicaciones en lugar de calcularlos en el proxy, lo cual permite obtener un conjunto de métricas de carga más sólido y preciso que el que está disponible en el proxy.
- Control detallado del ciclo de vida de la aplicación web: En Modcluster, cada servidor reenvía un evento del ciclo de vida de la aplicación web (por ejemplo, implementar/retirar la implementación) al proxy, informándole que inicie/detenga las solicitudes de enrutamiento enviadas a ese servidor.
- AJP es opcional: A diferencia de otros módulos como mod_jk y mod_proxy, Modcluster no requiere AJP, el cual es un método para un servidor web para comunicarse con un servidor de aplicaciones asociadas. Por lo tanto, las conexiones httpd a los nodos del servidor de aplicaciones pueden usar HTTP, HTTPS o AJP.

Modcluster trabaja en la capa 7 en el modelo OSI, lo cual significa que funciona a nivel de aplicación, y se compone de los servicios y aplicaciones de comunicación estándar.

El funcionamiento de Modcluster se basa en inspeccionar la petición HTTP, lo cual permite analizar la petición y sus encabezados, lo cual se usa como estrategia para el balanceo de carga, y con esto se puede realizar el balanceo en base a la cadena de la consulta, en cookies o en algún encabezado que se escoge (Pazmiño & Hernán, 2014).

Este balanceador de carga tiene las siguientes opciones de integración con el servidor de aplicaciones Wildfly:

- Utilizando el servidor HTTP Apache, el cual es un servidor web gratuito y de código abierto destinado para plataformas UNIX mantenido y desarrollado por Apache Software Foundation, y que cuenta con la característica de ser altamente personalizable por su estructura basada en módulo, lo cual permite a los administradores del servidor activar o desactivar funcionalidades adicionales (Bustos, 2023). Dentro del este servidor se deberá integrar los módulos proporcionados por su

repositorio oficial y realizar la configuración necesaria para su funcionamiento, y que servirá como front-end para la recepción y redireccionamiento de peticiones al lado del trabajador o back-end mediante la integración de los contenedores Modcluster integrados en el servidor de aplicaciones Wildfly.

- Utilizando una instalación de Wildfly, que desde su versión 9 integra el subsistema Modcluster, y permite configurar una instancia del servidor de aplicaciones como un front-end de Modcluster para sus aplicaciones de back-end, eliminando la necesidad de usar un servidor web nativo como Apache como equilibrador de carga para un grupo de servidores WildFly (Marchioni, 2021).

1.4 ISO/IEC 25010

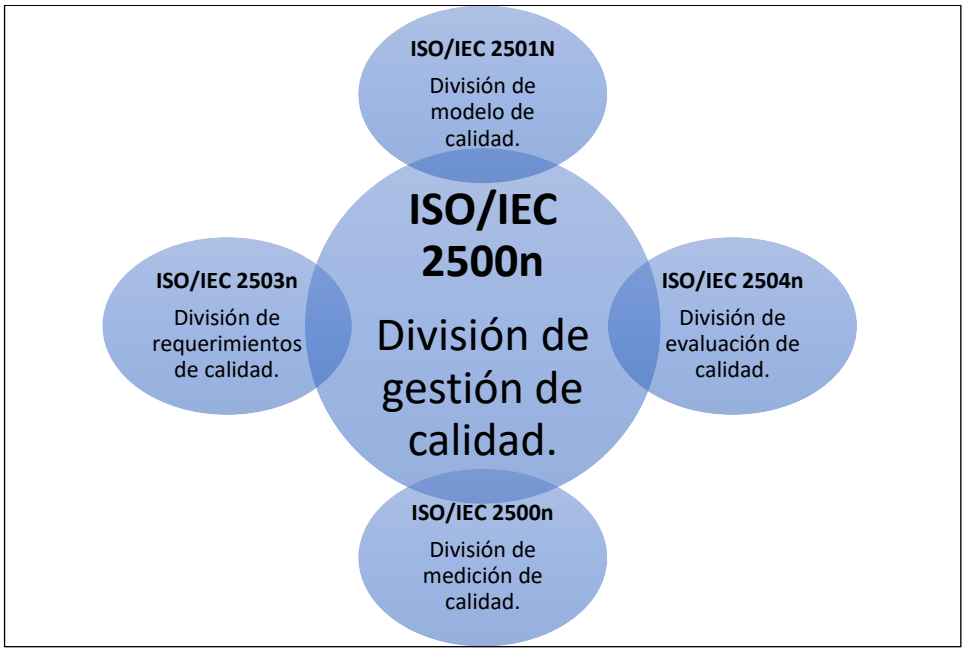
1.4.1 Familia ISO/IEC 25000

La familia ISO/IEC 25000, o también denominada SQuaRE (Requisitos y Evaluación de Calidad de Productos de Software), hace referencia a un conjunto de normas que facilitan la elaboración de un marco común encargado de medir la calidad del producto software(Reina & Patiño, 2019). Estas normas surgen para crear modelos, métricas, procesos y herramientas de evaluación de calidad del software como producto, como también, permiten asegurar la calidad del producto entregado, optimizando el tiempo de entrega, los recursos utilizados y el costo del personal (Roa et al., 2015).

La familia ISO/IEC 25000 está compuesta por varias divisiones tal como se muestran en la Figura 6.

Figura 6

Divisiones de la norma ISO/IEC 25000

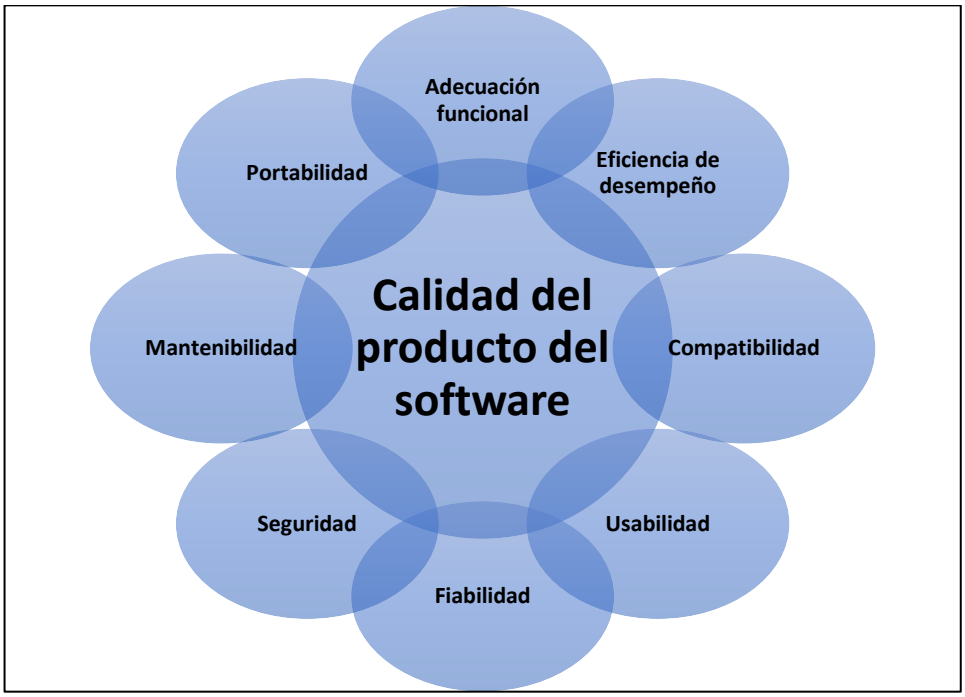


1.4.2 ISO/IEC 25010

La norma ISO/IEC 25010 sirve como un punto de referencia para establecer un sistema para la evaluación de la calidad del producto, y que cuenta con ocho características específicas para evaluar la calidad del software, presentadas en la Figura 7.

Figura 7

Características de calidad ISO/IEC 25010

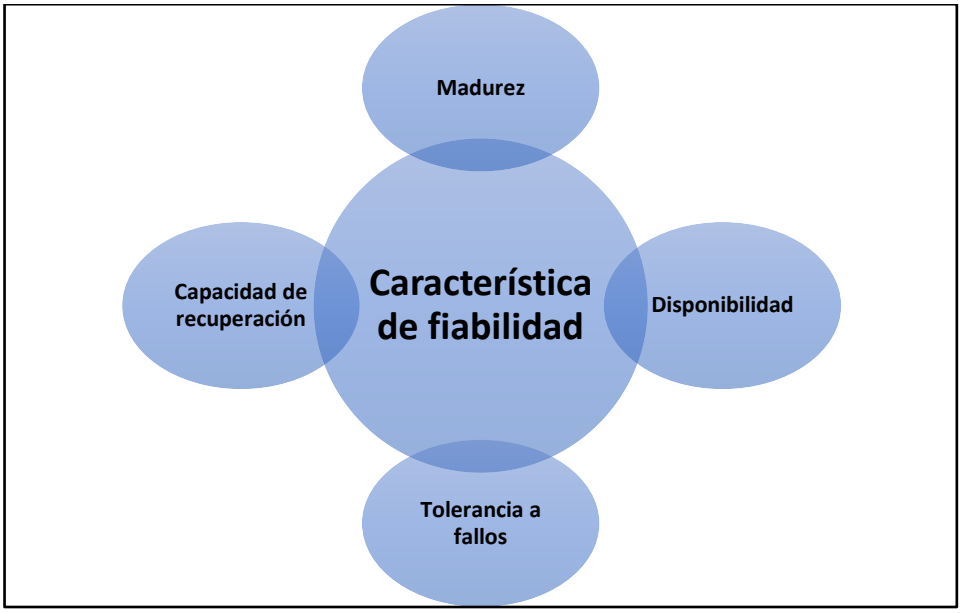


1.4.3 Subcaracterística de Disponibilidad

En el presente trabajo se abordará y se evaluará la subcaracterística de disponibilidad, la cual está contenida en la característica de fiabilidad del modelo del producto de del software, como se muestra en la Figura 8. Según ISO25000 (2022), define a esta subcaracterística como la capacidad del sistema o componente de estar operativo y accesible para su uso cuando se requiere.

Figura 8

Subcaracterísticas correspondientes a fiabilidad.



1.4.4 Métricas de la Subcaracterística de Disponibilidad

Tomando en cuenta las características y subcaracterísticas presentadas en la norma ISO/IEC 25010, el estándar ISO/IEC 25023 presenta un conjunto de métricas necesarias para evaluar cuantitativamente la calidad de un producto. La mayoría de las métricas se basan sus mediciones en proporciones determinadas por un total y un obtenido, donde el resultado de aplicar la métrica corresponde al porcentaje cumplido para la misma (Calabrese, 2018).

En la Tabla 2 (Balseca, 2014), se muestran las métricas proporcionadas por estándar ISO/IEC 25023 para la subcaracterística de disponibilidad, como también el método y valores para la evaluación de estas, y que son utilizadas para la validación de los resultados del presente trabajo

Tabla 2

Cálculo de métricas de disponibilidad

Subcaracterís tica	Métrica	Propósito de la métrica de calidad	Método de aplicación	Fórmula	Valor deseado	Tipo de medida
Disponibilidad	Tiempo de servicio	¿Cuál es el tiempo de servicio del sistema que proporciona realmente?	Tomar el tiempo de servicio del sistema que proporciona actualmente y tomar el tiempo de servicio regulado en el cronograma operacional	X=A/B A= Tiempo de servicio que se proporciona actualmente. B=Tiempo de servicio del sistema regulado en el cronograma operacional. Donde B>0	0<=x>=1 Cuanto más se acerque a 1 es lo mejor	X= Tiempo/ Tiempo A=Tiempo B=Tiempo
	Tiempo medio de inactividad	¿Cuál es el tiempo promedio que el sistema está inactivo después de que ocurre un fallo?	Tomar el tiempo total de inactividad y contar el número de fallos observados.	X=A/T A= Número de fallos observados T=Tiempo total de inactividad	X=A/T El más cercano a 0/t es el mejor	X= Contable/ Tiempo A=Tiempo T=Contable

Nota: Tomado y adaptado de Balseca (2014).

1.5 Trabajos Relacionados

En el estudio de Parra (2016), se implementó en el Ministerio de Educación del Ecuador una arquitectura que permita obtener una alta disponibilidad en sus servidores de aplicaciones. Este proyecto se realizó mediante la configuración de un conjunto de servidores de aplicaciones JBoss EAP 6.4 en modo Domain y la instalación y configuración de un servidor Web Apache 2.4 con los módulos de balanceo de Modcluster en su versión 1.2.11, que fueron levantados sobre una infraestructura de servidores que utilizaban el sistema operativo Red Hat Enterprise Linux v7. Como resultados se obtuvo centralizar la administración de aplicaciones, así como una mayor escalabilidad en el número de servidores de aplicaciones de alta transaccionalidad asegurando la disponibilidad de las aplicaciones.

En el trabajo de Pazmiño (2014), se desarrolló una guía metodológica para la implementación de un clúster de alta disponibilidad para servidores de aplicaciones para la empresa Soporte Libre Cia. Ltda, ubicada en la ciudad de Quito, Ecuador. Para el desarrollo se utilizó la plataforma de virtualización Kernel-based Virtual Machine (K.V.M), donde se realizó la instalación y configuración de 3 máquinas virtuales con el sistema operativo Red Hat Enterprise Linux v6.6, de las cuales dos fueron destinados para la instalación de los servidores de aplicaciones JBoss EAP 6.3 utilizando la configuración Standalone, y la restante para la configuración del servidor Web Apache y los módulos de mod_custer en su versión 1.2.1. Como resultado, se menciona que la guía metodológica desarrollada servirá como una herramienta que asegurará una correcta y rápida implementación de alta disponibilidad en sus servidores, facilitando el trabajo de los técnicos de la empresa Soporte Libre Cia. Ltda.

En el estudio de Vera (2018), se realizó la implementación de un clúster de alta disponibilidad para aumentar la disponibilidad de los servidores de aplicaciones en la institución de Autoridad Nacional del Servicio Civil del Estado Peruano. El desarrollo de este proyecto se realizó utilizando el sistema operativo Red Hat Enterprise Linux 6.4 para las máquinas virtuales, en las cuales se realizó la implementación de los diferentes nodos para los servidores de aplicaciones JBoss EAP 6.2 utilizando la configuración Domain, así como también la implementación del nodo de servidor Web Apache y configuración del módulo

mod_jk utilizado para el funcionamiento del balanceo de carga. Con esta implementación se logró un aumento en la disponibilidad de las aplicaciones desplegadas, asegurando su accesibilidad y operatividad en todo momento.

En el proyecto de Cali (2011), se implementó un clúster de nivel de aplicación que proporciona una alta disponibilidad en la empresa Casa Moeller Martínez C.A. El proyecto se realizó utilizando una arquitectura de 3 máquinas virtuales que operan con el sistema operativo Red Hat Enterprise Linux v5.6. En el equipo dedicado al balanceo de carga se configuró el servidor Web Apache juntamente con el módulo mod_jk, así como en los equipos dedicados para los servidores de aplicaciones se configuró la plataforma JBoss AS 5.1. La solución propuesta mediante el balanceo de carga permitió tener redundancia sin necesidad de migraciones de datos o interrupciones de servicios, con la característica de ser transparente para el usuario final, y que la cantidad de servidores que realizan el balanceo de carga es escalable horizontalmente

En el trabajo de Perafan et al., (2018), se diseñó un clúster de alta disponibilidad para el Ambiente del Aula Virtual de la Facultad de Ingeniería de la Universidad de Carabobo. Para el desarrollo de este proyecto se utilizó el virtualizador VirtualBox, y el sistema operativo CentOS 7 para los diferentes nodos del clúster. Posteriormente se configuró los servidores Web Apache con una topología de trabajo activo/pasivo, integrando también al servidor de base de datos, para dar una disponibilidad continua al entorno de prueba diseñado sobre la plataforma de aprendizaje Moodle. Para la evaluación del desempeño del clúster se utilizó términos de fiabilidad y accesibilidad, y que proporcionaron resultados favorables, en donde se observó que se proporciona una alta disponibilidad para dos servidores de prueba utilizados mediante aplicaciones de uso libre, como también definir a esta arquitectura como una configuración segura, robusta y altamente flexible.

CAPÍTULO 2

Desarrollo

2.1 Materiales y métodos

2.1.1 Herramientas de trabajo

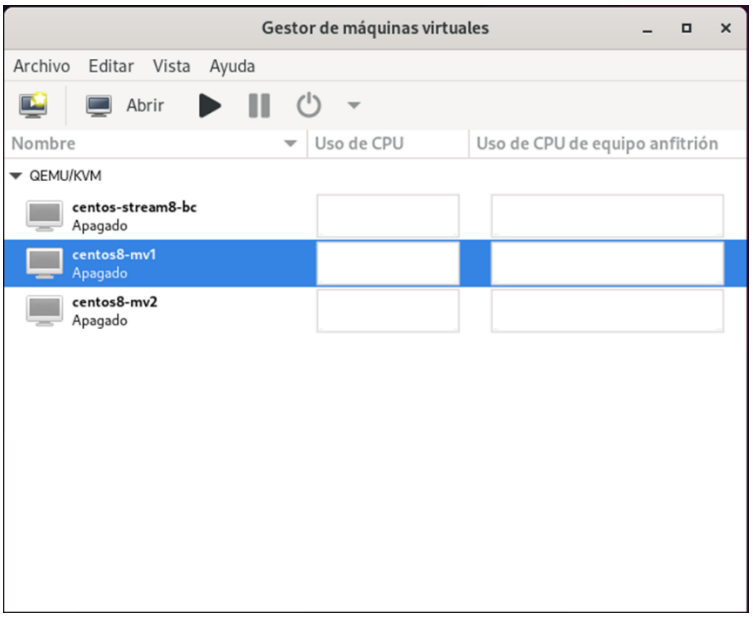
KVM

KVM (for Kernel-based Virtual Machine), es una solución de código abierto que permite una virtualización completa para Linux, y permite la ejecución de varias máquinas virtuales con imágenes Linux o Windows, donde cada instancia cuenta con su propio hardware virtualizado exclusivo, como adaptador gráfico, tarjeta de red, las CPU, memoria y discos duros (KVM, 2016). Una característica importante es que a partir del 2007 se incorporó esta tecnología al kernel de Linux, permitiendo recibir mejoras, correcciones y nuevas funciones en su sistema de formas inmediata (Red Hat, 2022ª).

Para la administración de las instancias virtuales se utilizó el software Virtual Machine Manager, y en la Figura 9 se puede observar la interfaz gráfica donde se presenta funcionalidades de creación y gestión de las diferentes máquinas virtuales, así como también la visualización de un monitor de recursos donde se presenta el uso de CPU, tanto interno como del equipo del anfitrión.

Figura 9

Interfaz gráfica del gestor de máquinas virtuales.



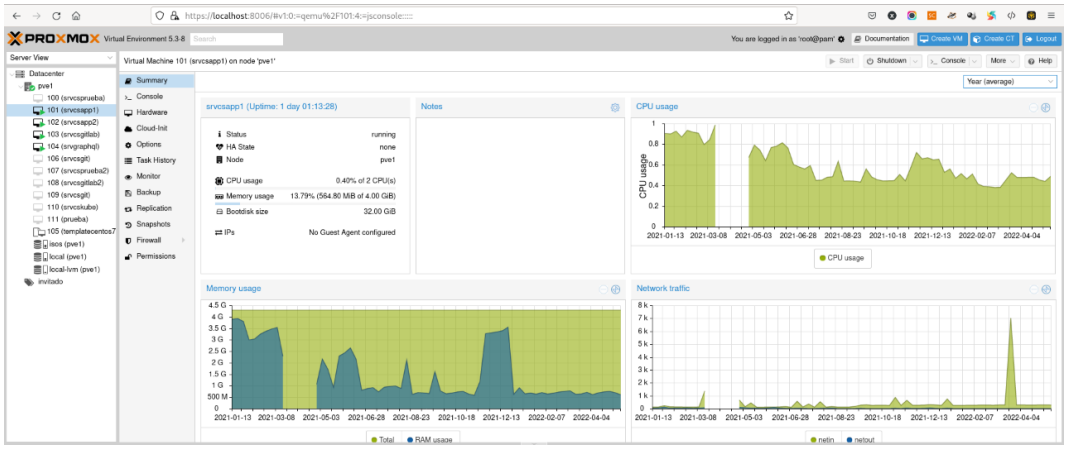
PROXMOX VE

Es una plataforma de código abierto basado en Debian GNU/Linux, que integra las tecnologías de virtualización QEMU/KVM para máquinas virtuales y LXC para los contenedores de Linux. Este software está destinado para una virtualización empresarial, denominándola una tecnología perfecta para virtualizar una infraestructura de TI donde se tenga una carga de trabajo exigente y se necesite aumentar la eficiencia con un gasto mínimo (PROXMOX, 2023).

Esta plataforma incorpora una interfaz web que permite administrar fácilmente las máquinas virtuales, contenedores, almacenamiento, gestión de nodos y características de alta disponibilidad, entre otras herramientas, tal como se muestra en la Figura 10.

Figura 10

Interfaz web de la plataforma PROXMOX VE



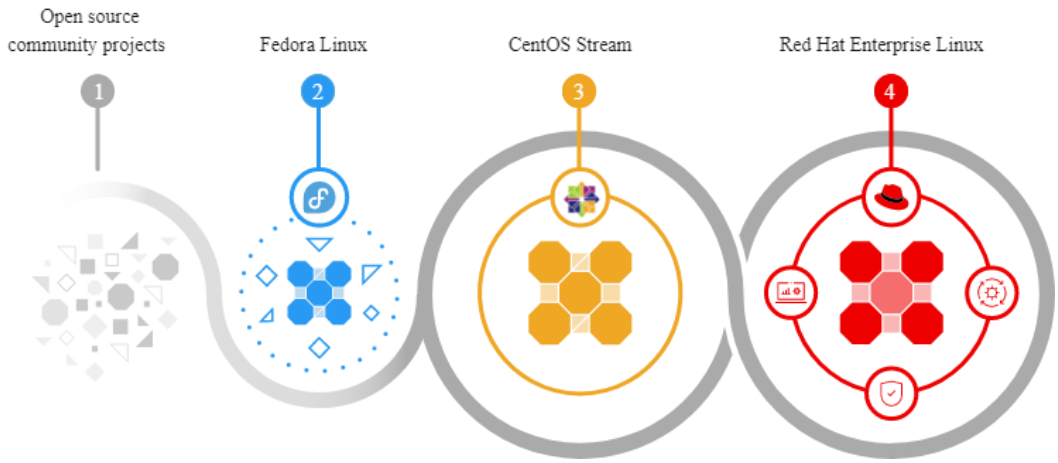
CentOS Stream 8

Es una distribución de entrega continua de código abierto de Linux desarrollado por Red Hat, que se posiciona como un intermediario entre el desarrollo de Fedora y Red Hat Enterprise Linux (RHEL), tal como se muestra en la Figura 11, donde Fedora reúne las mejores ideas de proyectos de la comunidad de código abierto disponible, mientras que Centos permite realizar aportes importantes a la próxima versión de RHEL, siendo este último una entrega de producción con una base más segura, compatible y flexible para cargas de trabajo .

Esta plataforma ofrece un sistema de grado empresarial con un flujo constante de contenido y actualizaciones, y considerada también como una distribución estable de alto rendimiento y disponibilidad, ya que el software debe cumplir los mismos estándares de la distribución comercial RHEL, convirtiéndose en una gran alternativa de sistema operativo de código abierto.

Figura 11

Ecosistema de software proporcionado por Red Hat



Nota: Adaptado de *Ecosistema de Red Hat* [Fotografía], por Red Hat, 2019, ¿Qué es CentOS Linux? (https://www.redhat.com/es/topics/linux/what-is-centos-stream).

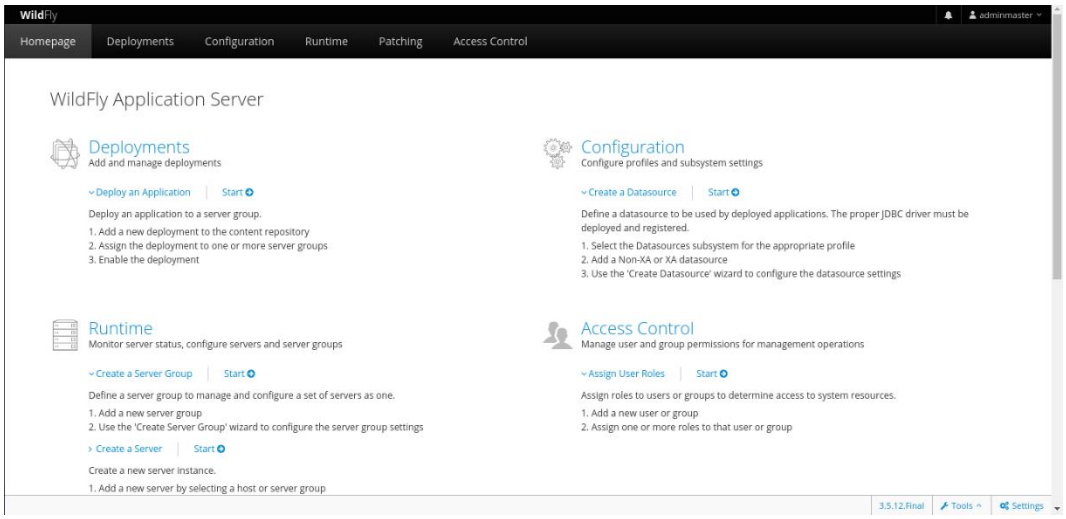
Servidor de aplicaciones Wildfly

Es un servidor de aplicaciones de código abierto para el despliegue que contiene características de ser potente, modular, ligero y basado en estándares, y que es utilizado dentro del Cloud FICA como software para el despliegue de aplicaciones con tecnología Java Enterprise.

Para el presente proyecto se utilizará varias instancias de Wildfly para los diferentes componentes de la arquitectura propuesta. Una instancia de Wildfly se configurará como balanceador de carga mediante el subsistema Modcluster, y la siguientes se establecerán como un grupo de servidores que estarán configuradas en modo dominio, lo cual permitirá administrar un conjunto de instancias de Widlfy y compartir configuraciones entre ellas, así como también tener una única consola de gestión, como se presenta en la Figura 12.

Figura 12

Consola de administración Web de Wildfly.

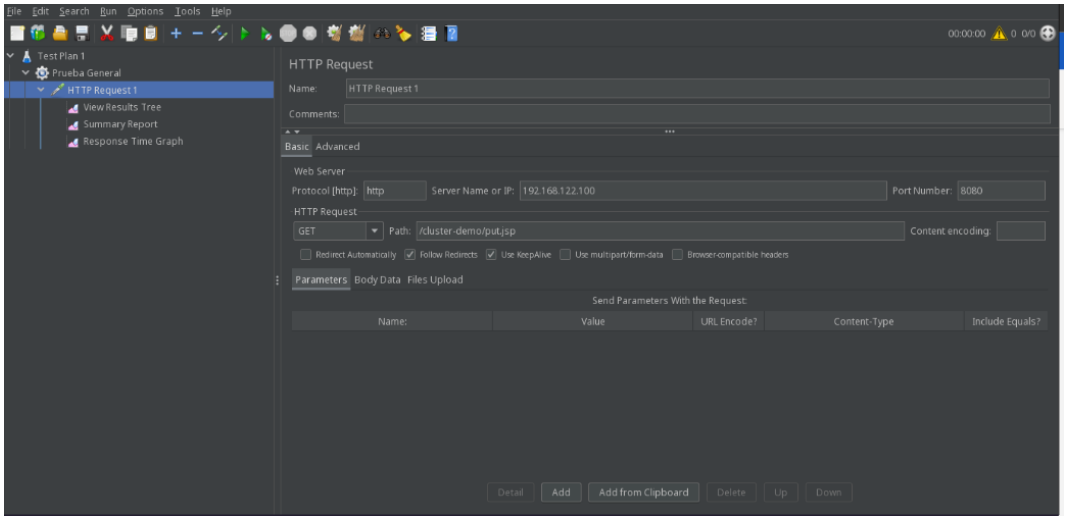


Apache JMeter

Es un software de código abierto desarrollada en el lenguaje Java, diseñado para probar el comportamiento funcional y medir el rendimiento estático como dinámicos de aplicaciones web (Apache Software Foundation, 2022). Entre las funcionalidades de este software se incluye la capacidad de cargar y probar el rendimiento de aplicaciones, servidores y diferentes tipos de protocolos, lo cual es indicado para la sección de pruebas de la presente investigación.

Figura 13

Interfaz gráfica del software JMeter



2.1.2 Metodología Kanban

Según (Castellano, 2019), esta metodología busca conseguir un flujo de trabajo productivo, organizado y eficiente, donde se requiere una comunicación en tiempo real sobre la capacidad y transparencia del trabajo total. Esta metodología se basa en una serie de principios que muestran a continuación:

- Visualización: permite una visualización total del desarrollo del proyecto y sus diferentes tareas que la componen, facilitando la organización y realización de modificaciones si fuera necesario.
- Calidad: se verifica que se realizase correctamente cada tarea desde el inicio.
- Disminución de desperdicios: se verifica que se realice lo justo y necesario.
- Priorización y flexibilidad: permite tener una gestión correcta del tiempo con un orden determinado para cada tarea.
- En proceso: permite la modificación continua de las actividades que se realiza.
- Mejora continua: permite tener una mejora indefinida, permitiendo mejorar continuamente los procesos en función de los objetivos planteados.

Para la implementación de esta metodología es necesario la utilización de señales visuales que mantienen activo el proceso de desarrollo del proyecto, y que pueden ser desde tarjetas, tableros, señales visuales o electrónicas. Por lo general, estos tableros se componen de secciones o columnas donde se van a clasificar cada tarjeta de actividades por su estado, que puede ser: pendiente, en proceso o finalizado.

Microsoft Planner

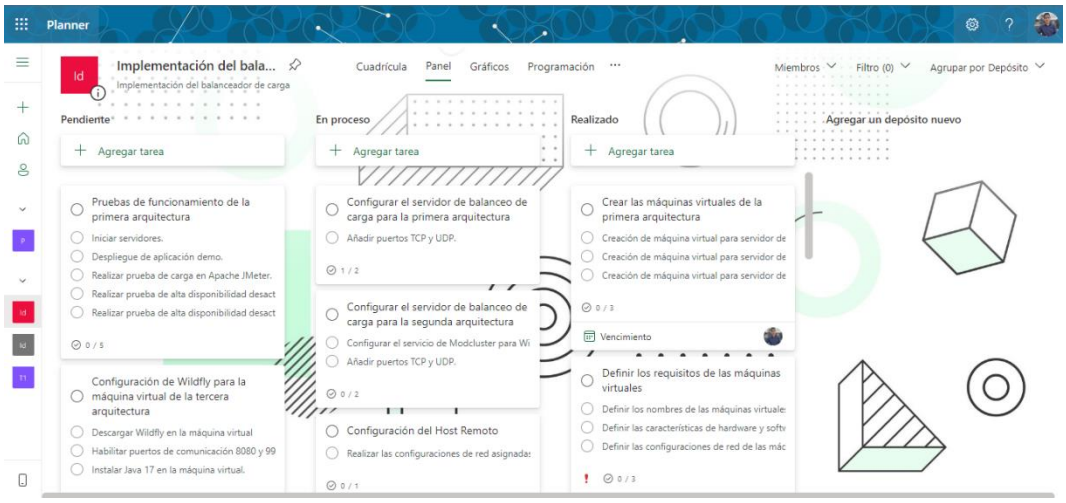
Microsoft Planner es una solución completa para la gestión de tareas y trabajos mediante la creación de un tablero Kanban visualmente atractivo, como se muestra en la Figura 15, donde se incluyen características de añadir archivos, listas de comprobación y etiquetas codificadas por color (Microsoft, 2023).

Estos tableros o paneles Kanban permiten organizar las tareas de forma coherente y estructurada, reduciendo las interrupciones que suelen surgir al momento de realizar proyectos. El objetivo final de estos paneles Kanban es impulsar la eficiencia, ya que garantiza

que los miembros del equipo no inviertan tiempo en tareas innecesarias, centrándose en tareas específicas a un ritmo constante (Microsoft 365 Team, 2021).

Figura 14

Tablero Kanban realizado con Microsoft Planner



2.1.3 Metodología Delphi

Este método tiene como objetivo final conseguir un consenso fiable de la herramienta o instrumento desarrollado, donde se establecerá un conjunto de preguntas que son presentadas a un grupo de expertos sobre tema de estudio específico, y que debido a su experiencia profesional cuentan con la capacidad de emitir juicios de valor que permitirán rediseñar el instrumento hasta que los expertos lo consideren como válido y que cumpla con los criterios de la investigación.

Las características básicas que debe tener el método Delphi según Martínez et al. (2019) son:

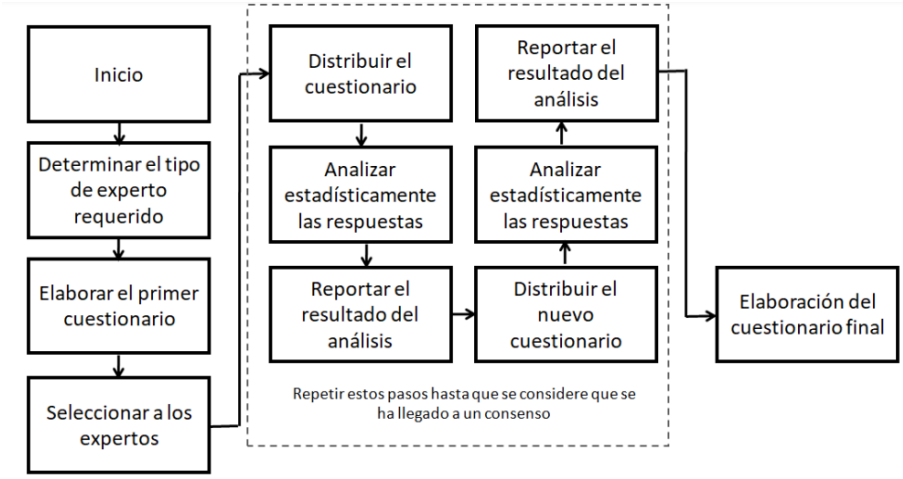
- Anonimato: permite disminuir la influencia que puede tener las opiniones del experto por parte de otros miembros de la comunidad.
- Interacción y retroalimentación controlada: se recibe directamente los comentarios y sugerencias por parte del personal que conoce del tema, permitiendo obtener una mayor confianza de las propuestas y correcciones recibidas.

- Respuesta del grupo en forma estadística: al completar los resultados permite cambiar los datos cualitativos a cuantitativos para proponer el siguiente análisis en caso de ser necesario

En la investigación de Martínez et al. (2019), presenta que esta metodología se compone de una serie de pasos estructurados y flexibles que permiten adecuarse a las características del problema planteado, así como también menciona que la validación de instrumentos es importante para garantizar la seriedad de la información presentada en una investigación, y asegurar la confiabilidad de la herramienta planteada. En la Figura 15 se visualiza el proceso de validación mediante el método Delphi propuesto por George & Liñan (2018).

Figura 15

Proceso general del método Delphi.



Nota: Adaptado de “Aplicación del Método Delphi Modificado para la Validación de un Cuestionario de Incorporación de las TIC en la Práctica Docente” (p. 119), por C. George & L. Liñán, 2018, *Revista Iberoamericana De Evaluación Educativa, 11*(1).

2.2 Implementación del balanceador de carga

2.2.1 Metodología Kanban para el desarrollo de proyecto.

La implementación de del balanceador de carga en sus diferentes arquitecturas se desarrolló bajo la guía de la metodología Kanban. Se estableció el tablero de trabajo: “Implementación del balanceador de carga”, el cual está definido por 18 tarjetas que detallan

las diferentes tareas y subtareas del proyecto, incluyendo su respectiva fecha de inicio y de vencimiento, prioridad y progreso. Las diferentes tareas que se definió para el desarrollo de la implementación se detallan en el Apéndice A.

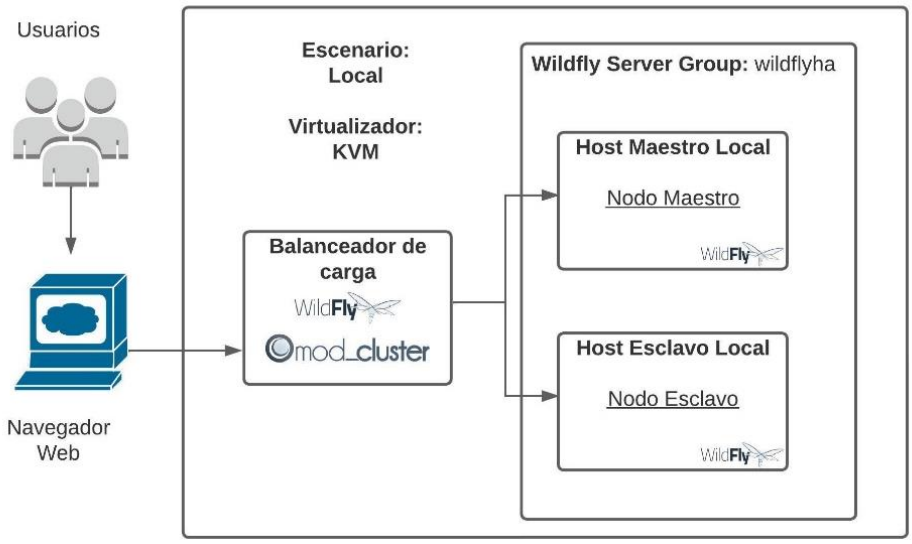
2.2.2 Diseño de arquitectura

Para el proyecto se definió 3 arquitecturas donde se contó con requisitos y escenarios diferentes, por lo cual se requirió un diseño específico para cada etapa y que son presentadas a continuación:

- En la Figura 16 se presenta la primera arquitectura del proyecto. Se definió al software KVM como plataforma de virtualización y corresponde a una implementación local de la solución de alta disponibilidad.

Figura 16

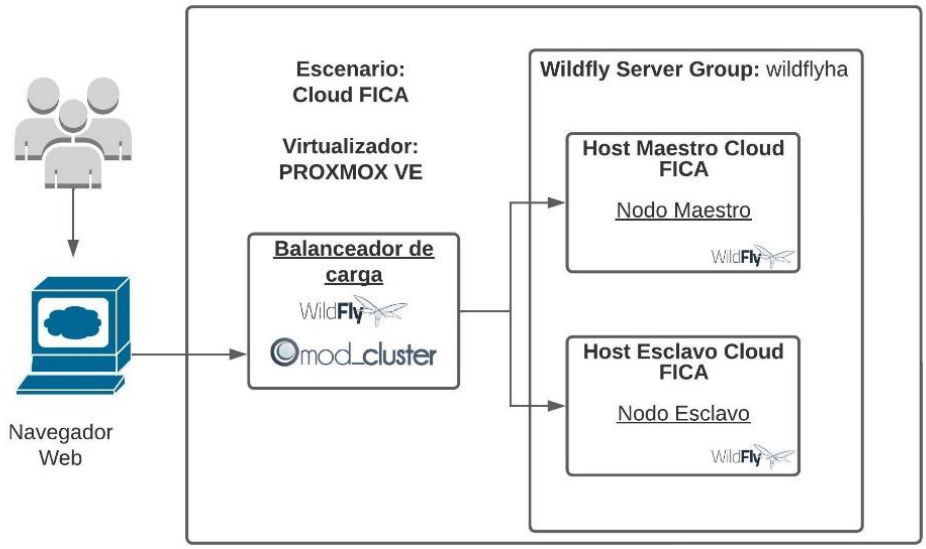
Primera arquitectura del proyecto.



- En la Figura 17 se presenta la segunda arquitectura del proyecto. Se definió al Cloud Fica de la Universidad Técnica del Norte como escenario de la implementación y al software PROXMOX VE como plataforma de virtualización. Esta arquitectura corresponde a una implementación a nivel de servidor de la solución de alta disponibilidad.

Figura 17

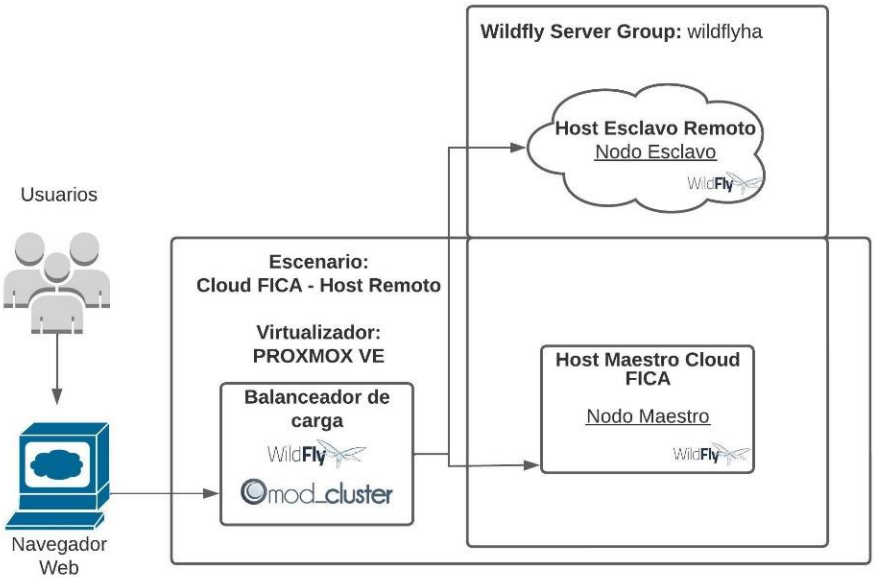
Segunda arquitectura del proyecto.



- En la Figura 18 se presenta la tercera arquitectura del proyecto, en la que se utilizó parte de la arquitectura anterior, desplazando al Nodo Wildfly Esclavo a un host remoto. Esta arquitectura corresponde al diseño final del proyecto, el cual presenta la solución de alta disponibilidad con una integración entre el Cloud Fica y un host remoto.

Figura 18

Tercera arquitectura del proyecto.



2.2.3 Definición de requisitos de máquinas virtuales

Se estableció una serie de requisitos para cada máquina virtual que conforman las diferentes arquitecturas, las cuales se detallan a continuación:

- En la Tabla 3 se definió los aspectos generales de las máquinas virtuales, como nombre de la máquina virtual, descripción y nombres de usuarios.

Tabla 3

Definición general de las máquinas virtuales

Arquitectura	Máquina virtual	Descripción	Nombre usuario
Primera	mv-srvbc	Nodo Balanceador de Carga Local	mvsrvbc
	mv-srvapp1	Nodo Wildfly Maestro Local	mvsrvapp1
	mv-srvapp2	Nodo Wildfly Esclavo Local	mvsrvapp2
Segunda	mv-cloud-srvbc	Nodo Balanceador de Carga en Cloud Fica	mvcloudsrvbc
	mv-cloud-srvapp1	Nodo Wildfly Maestro en Cloud Fica	mvcloudsrvapp1
	mv-cloud-srvapp2	Nodo Wildfly Esclavo en Cloud Fica	mvcloudsrvapp2
Tercera	mv-remoto-srvapp2	Nodo Wildfly Esclavo en Host Remoto	mvremotosrvapp2

- En la Tabla 4 se definió las especificaciones de hardware y software de cada máquina virtual.

Tabla 4

Definición de especificaciones de hardware y software de las máquinas virtuales

Máquina virtual	Hardware	Software
mv-srvbc	CPU: 2 RAM:4 GB DISCO: 20 GB	Wildfly 26.1.3
mv-srvapp1	CPU: 2 RAM:4 GB DISCO: 20 GB	Wildfly 26.1.3
mv-srvapp2	CPU: 2 RAM:4 GB DISCO: 20 GB	Wildfly 26.1.3
mv-cloud-srvbc	CPU: 2 RAM: 4GB DISCO: 20GB	Wildfly 26.1.3
mv-cloud-srvapp1	CPU: 2 RAM: 4GB DISCO: 30GB	Wildfly 26.1.3
mv-cloud-srvapp2	CPU: 2 RAM: 4GB DISCO: 30GB	Wildfly 26.1.3
mv-remoto-srvapp2	CPU: 2 RAM: 4GB DISCO: 30GB	Wildfly 26.1.3

- En la Tabla 5 se definió las configuraciones de red de cada máquina virtual.

Tabla 5

Configuraciones de red de las máquinas virtuales

Máquina virtual	Nombre Host	Dirección IP
mv-srvbc	srvbc.balanceador.local	192.168.122.100/24
mv-srvapp1	srvapp1.maestro.local	192.168.122.101/24
mv-srvapp2	srvapp2.esclavo.local	192.168.122.102/24

mv-cloud-srvbc	srvbc.balanceador.cloud	10.24.8.87/24
mv-cloud-srvapp1	srvapp1.maestro.cloud	10.24.8.88/24
mv-cloud-srvapp2	srvapp2.esclavo.cloud	10.24.8.89/24
mv-remoto-srvapp2	srvapp2.esclavo.remoto	10.24.8.90/24

2.2.4 Creación de máquinas virtuales

La creación de las máquinas virtuales fue el punto inicial para la estructuración de las arquitecturas propuestas. A continuación, se detalla el proceso de creación en las diferentes tecnologías de virtualización requeridos para cada arquitectura.

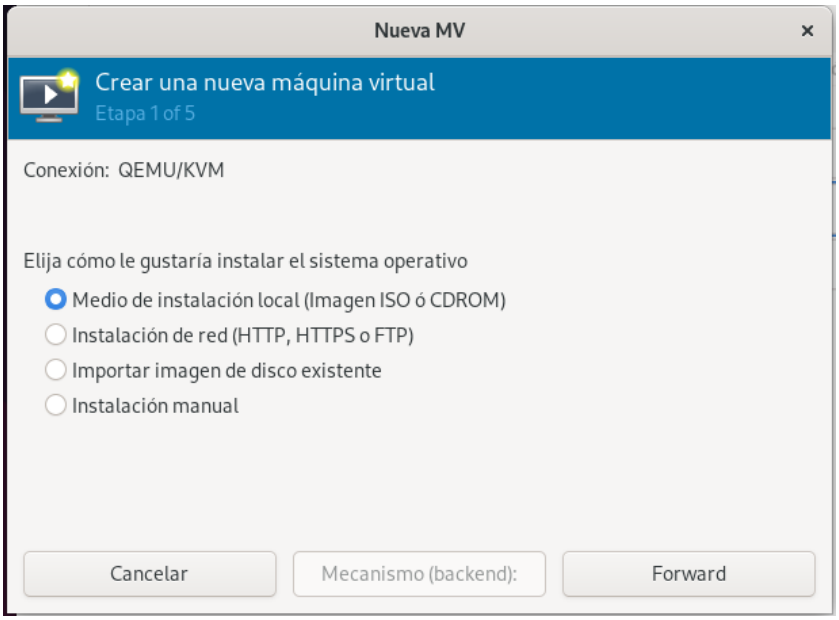
Creación de máquinas virtuales en virtualizador KVM

Para el cumplimiento de la primera arquitectura se creó las diferentes máquinas virtuales en el equipo anfitrión mediante el software de virtualización KVM. Los pasos necesarios para la creación de las máquinas virtuales se detallan a continuación.

Como se presenta en la Figura 20, se accedió al software Virtual Machine Manager y se seleccionó la opción: “Crear una nueva máquina virtual” y se seleccionó la opción: “Medio de instalación local (Imagen ISO o CDROM)”

Figura 19

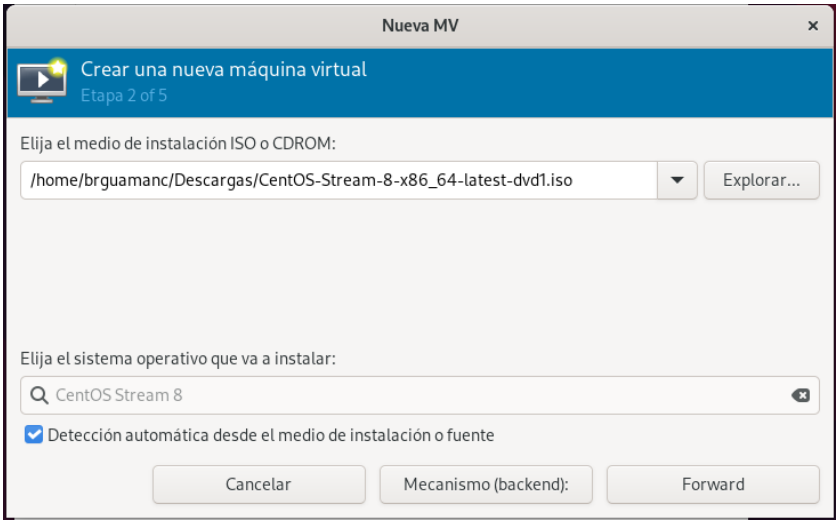
Ventana de creación de nueva máquina virtual de KVM



Como se presenta en la Figura 20, se seleccionó la imagen del sistema operativo CentOS Stream 8, la cual fue descargada anteriormente desde su repositorio oficial (http://isoredirect.centos.org/centos/8-stream/isos/x86_64/) y fue guardada en el almacenamiento del equipo anfitrión. Adicionalmente, se seleccionó la opción de: “Detección automática desde el medio de instalación o fuente”.

Figura 20

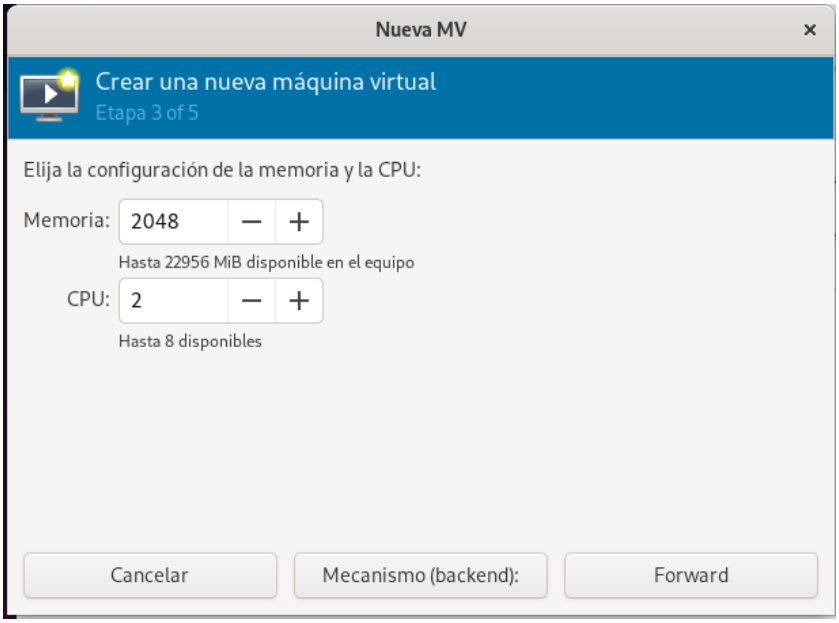
Ventana de selección de medio de instalación de KVM



Como se presenta en la Figura 21, se asignó la cantidad de memoria RAM y el número de CPU según los requisitos establecidos en la Tabla 5 para cada máquina virtual.

Figura 21

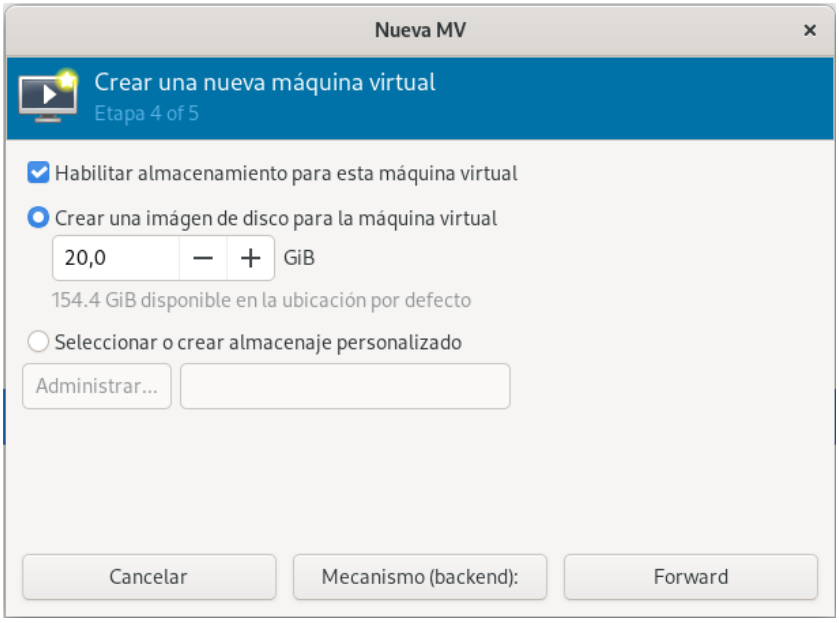
Ventana de asignación de memoria RAM y CPU de KVM.



Como se presenta en la Figura 22, se seleccionó la opción de: “Habilitar almacenamiento para esta máquina virtual” y se definió la cantidad de almacenamiento según los requisitos establecidos en la Tabla 5.

Figura 22

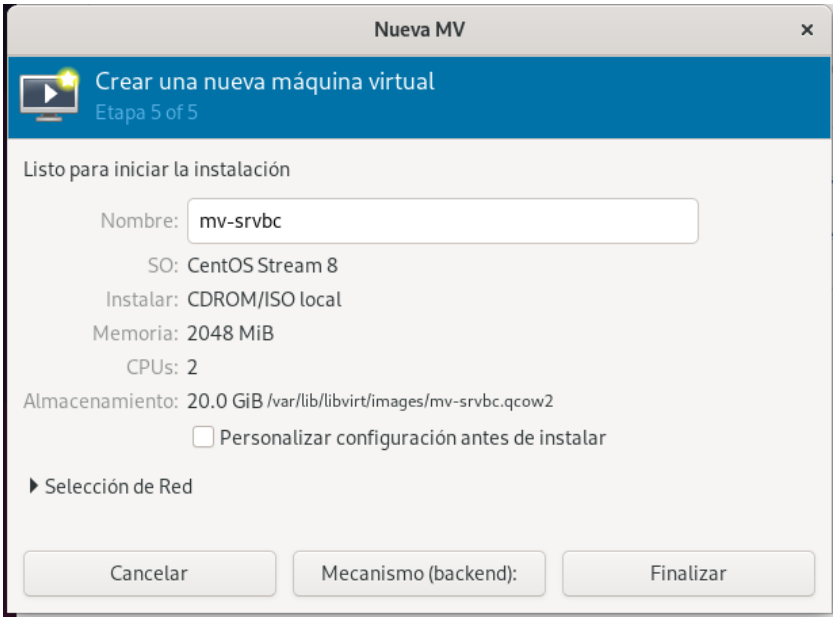
Ventana de asignación de almacenamiento de KVM



Como se presenta en la Figura 23, se asignó el nombre de la instancia, tomando en cuenta los requisitos definidos en la Tabla 3 para cada máquina virtual de la primera arquitectura.

Figura 23

Ventana de asignación nombre a la instancia de KVM



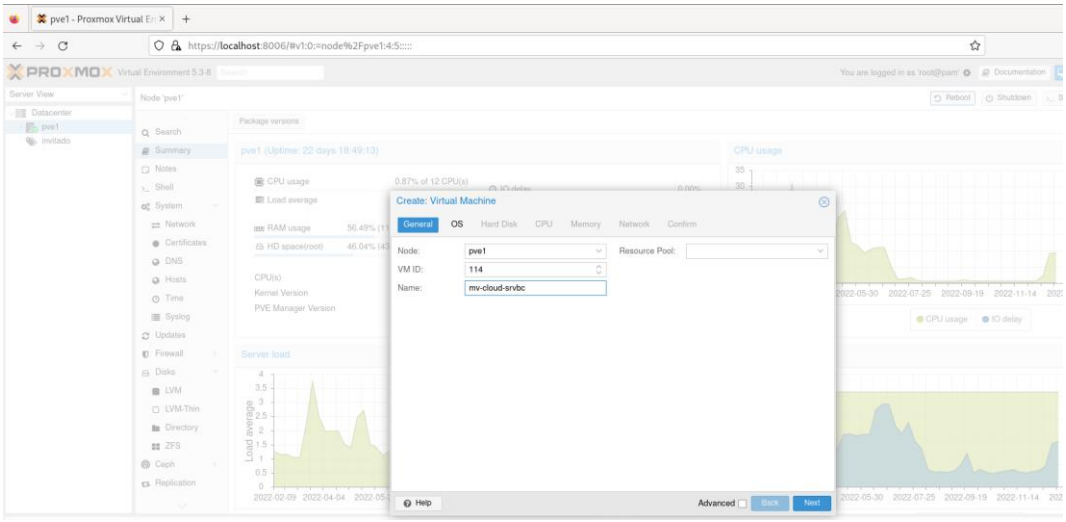
Creación de máquinas virtuales en virtualizador PROXMOX VE

Para el cumplimiento de la segunda arquitectura se accedió al Cloud FICA de la Universidad Técnica del Norte y creó las diferentes máquinas virtuales en el servidor mediante el software de virtualización empresarial PROXMOX VE. Los pasos necesarios para la creación de las máquinas virtuales se detallan a continuación.

Como se presenta en la Figura 24, se accedió a la consola de administración web de Proxmox Ve y se seleccionó la opción de: “Create Virtual Machine” y posteriormente se ingresará el nombre de la máquina virtual, definido en la Tabla 3 para cada máquina virtual de la segunda arquitectura.

Figura 24

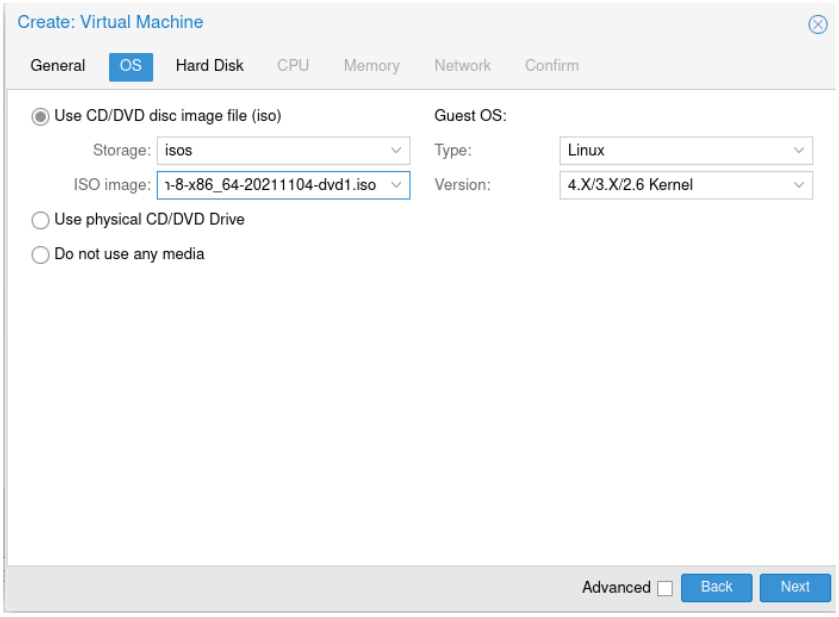
Ventana de creación nueva máquina virtual de PROXMOX VE



Como se presenta en la Figura 25, se seleccionó la imagen del sistema operativo CentOS Stream 8, la cual fue descargada anteriormente desde su repositorio oficial (http://isoredirect.centos.org/centos/8-stream/isos/x86_64/) y fue guardada en el almacenamiento del servidor. El software de virtualización detectará de forma automática el tipo de imagen ingresada al igual que su versión.

Figura 25

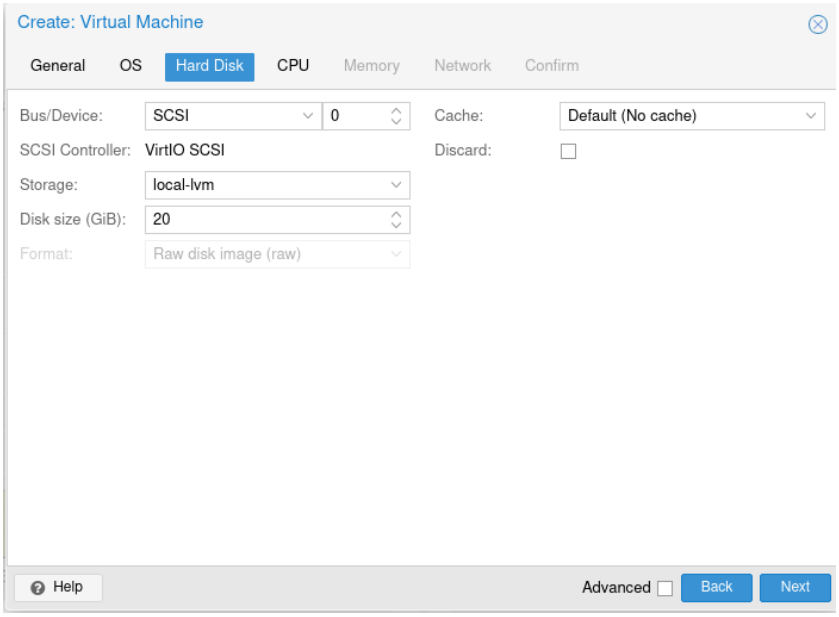
Ventana de selección de medio de instalación de PROXMOX VE



En la Figura 26 se presenta la ventana de configuración del disco duro de la máquina virtual, donde se asignó la cantidad de memoria según los requisitos presentados en la Tabla 4. El dispositivo se dejó por predeterminado en: “SCSI” y como almacenamiento: ”local-lvm”.

Figura 26

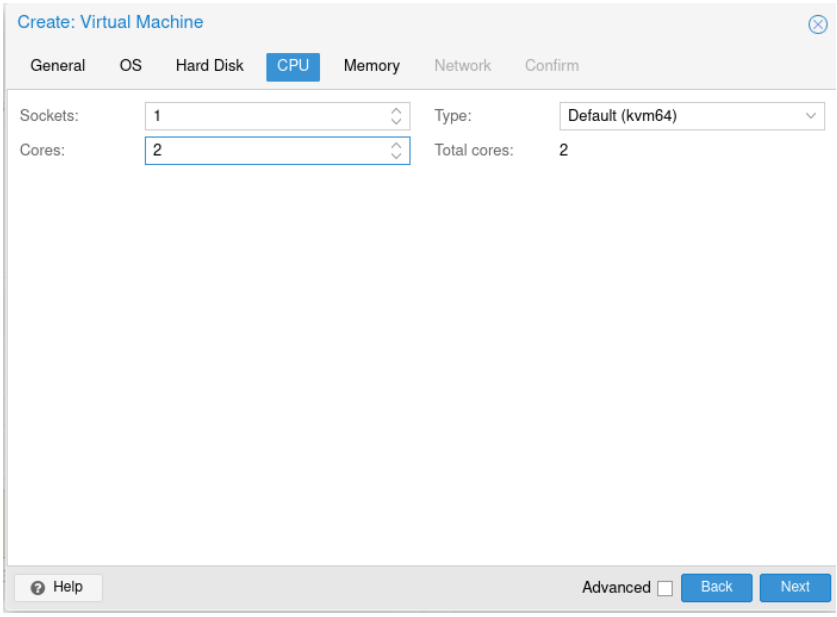
Asignación de disco duro de PROXMOX VE



Como se presenta en la Figura 27, se asignó la cantidad de cores virtuales, definidos bajó el parámetro CPU de la Tabla 4. El número de sockets se dejó por defecto en uno y el tipo en: “kvm64”.

Figura 27

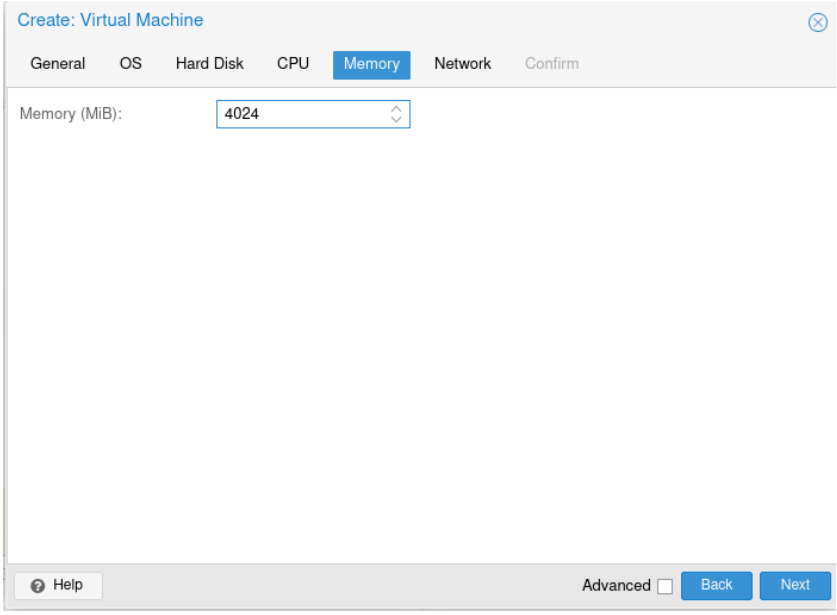
Ventana de asignación de CPU de PROXMOX VE



Como se presenta en la Figura 28, se asignó la cantidad de memoria RAM definido para cada máquina virtual según la Tabla 4.

Figura 28

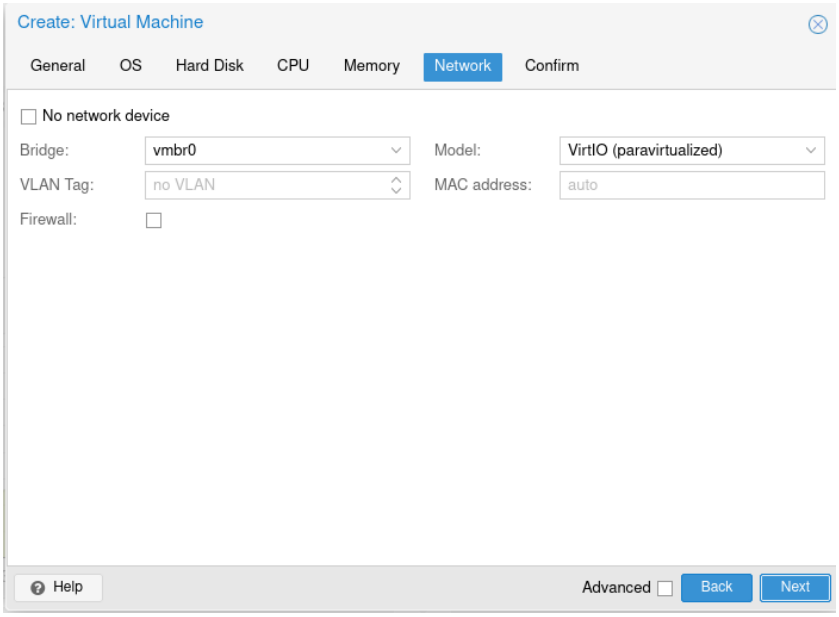
Ventana de asignación de memoria de PROXMOX VE



Como se observa en la Figura 29, se configuró los parámetros de red, que para este proyecto se dejó los parámetros establecidos por defecto.

Figura 29

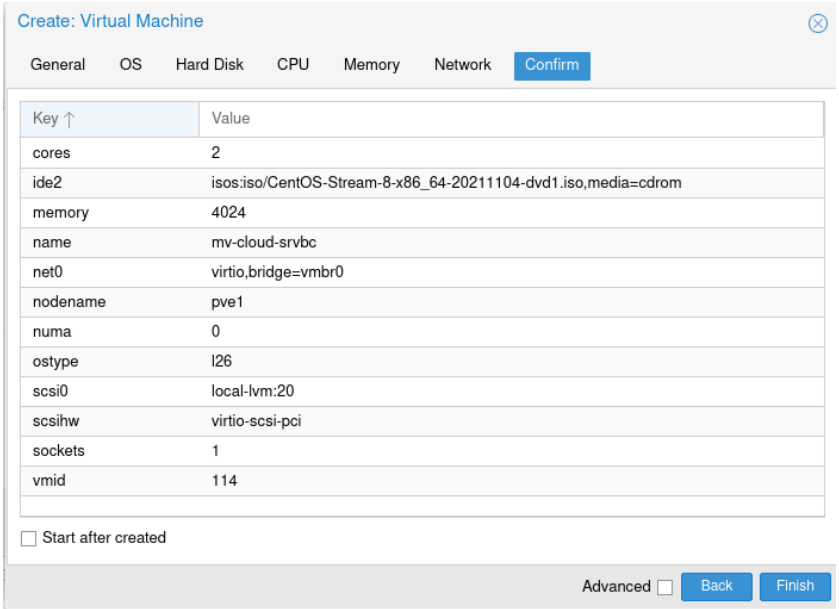
Ventana de configuración de red de PROXMOX VE.



Finalmente se presenta la ventana de confirmación de creación de la máquina virtual, donde se detallan los diferentes parámetros ingresados en los pasos anteriores, tal como se observa en la Figura 30. Este proceso se realizará para las diferentes máquinas virtuales que conforman la segunda arquitectura del proyecto.

Figura 30

Ventana de confirmación de creación de PROXMOX VE



2.2.5 Instalación y configuración del sistema operativo

Una vez culminado con la creación de las máquinas virtuales, según se establece en la sección 2.2.4, se continuó con la instalación y configuración del sistema operativo CentOS Stream 8 como se detalla a continuación.

Como se observa en la Figura 31, se escogió la oción de: “Install CentOS Stream 8-stream.” del menú de arranque inicial para la instalación del sistema operativo.

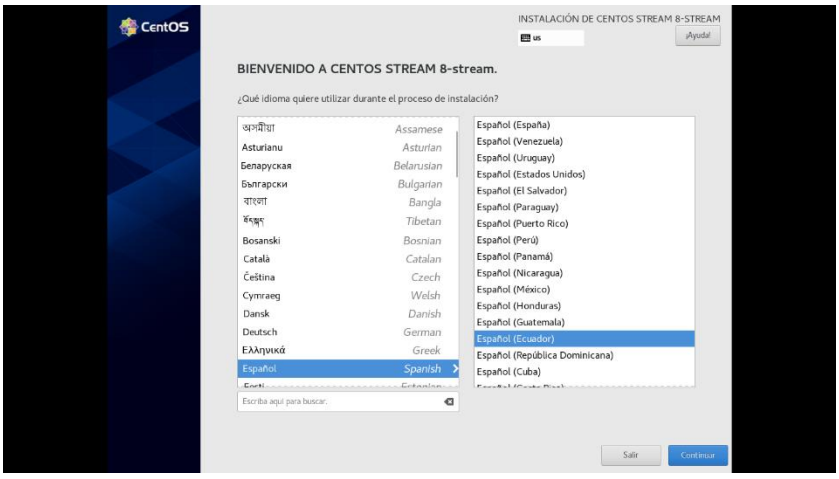
Figura 31

Menú de arranque inicial del sistema operativo.

Como se observa en la Figura 32, se seleccionó el idioma de Español (Ecuador) como predefinido para la instalación.

Figura 32

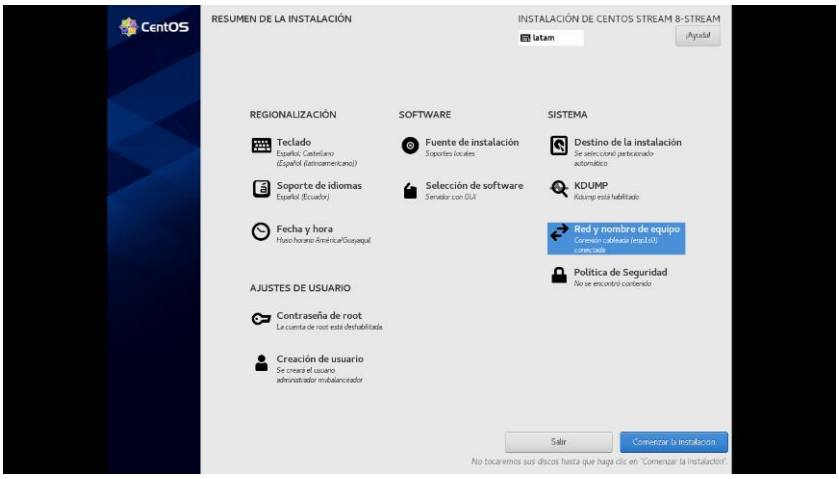
Ventana de selección de idioma de instalación.



Luego, se presenta la ventana denominada: “Resumen de la instalación”, presentada en la Figura 33, donde se definirá las configuraciones necesarias para cada apartado de la instalación del sistema operativo en las máquinas virtuales.

Figura 33

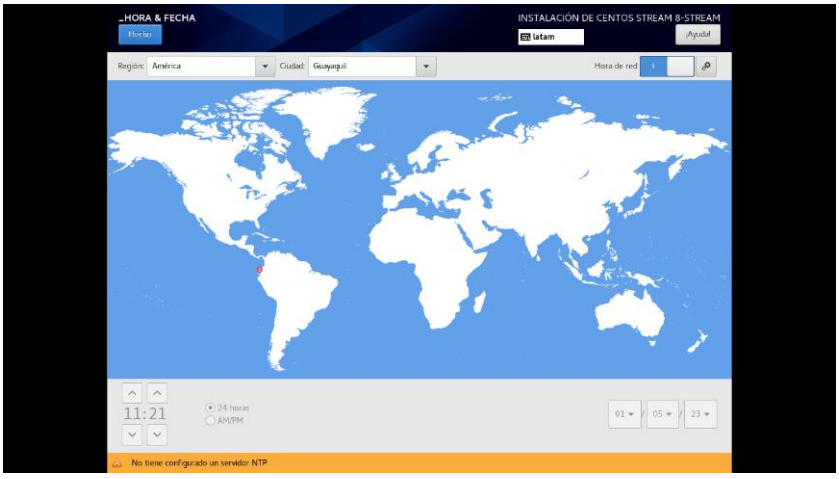
Ventana de resumen de la instalación del sistema operativo



Como se muestra en la Figura 34, se seleccionó la Región América-Guayaquil en el apartado de hora y fecha.

Figura 34

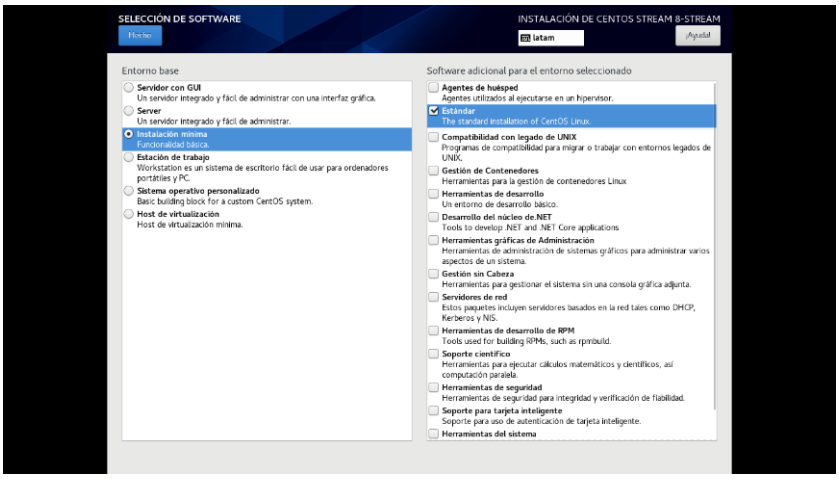
Apartado de selección de Fecha y Hora del sistema del sistema operativo



Como se muestra en la Figura 35, en el apartado de selección de software, se seleccionó la opción de: “Instalación mínima” para el entorno base, así como la opción de: “Estándar” para el software adicional del entorno seleccionado,

Figura 35

Ventana de selección de entorno base y software adicional del sistema operativo



Como se observa en la Figura 36, en el apartado de destino de instalación, se seleccionó el disco duro virtual creado, como también se seleccionó la opción: “Automática”, para la configuración del almacenamiento y asignación de particiones del disco duro.

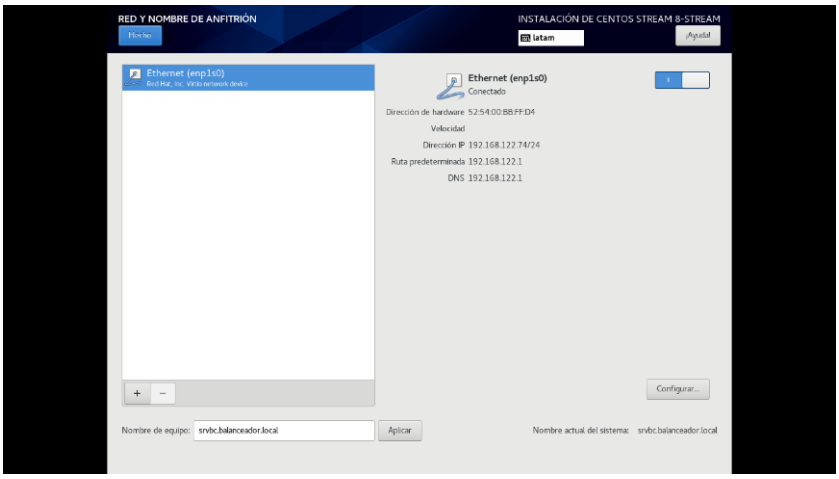
Figura 36

Ventana de selección de dispositivo del sistema operativo

Como se observa en la Figura 37, en el apartado de red y nombre del anfitrión, se activó la conexión predefinida de la máquina virtual, como también se definió el nombre de host, el cual fue tomado de la Tabla 5, para cada máquina virtual.

Figura 37

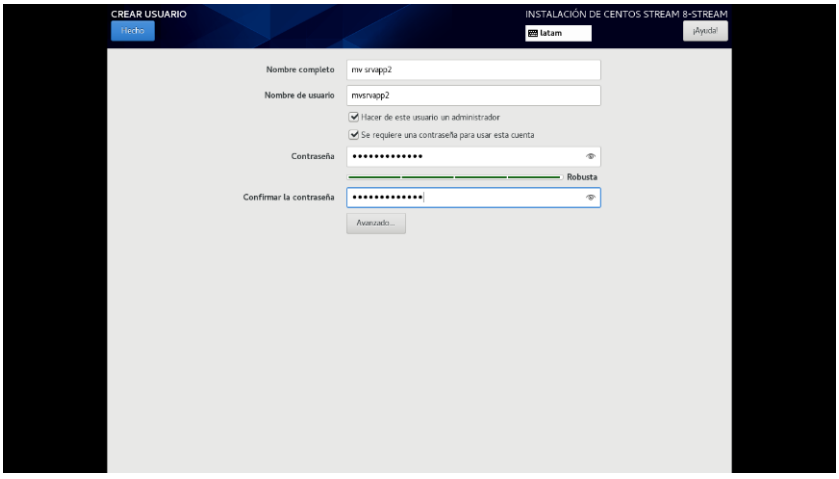
Ventana de red y nombre del anfitrión del sistema operativo



Como se presenta en la Figura 38, en el apartado de crear usuario, se definió el nombre de usuario para cada máquina virtual, el cual se establecerá a partir de los requisitos definidos en la Tabla 5, como también se definió contraseñas que califiquen como “Robusta” para cada instancia. Se selección la opción de: “Hacer de este usuario un administrados”, como también la opción de: “Se requiere una contraseña para usar esta cuenta”.

Figura 38

Ventana de crear usuario del sistema operativo



Una vez completados los parámetros anteriores, se procedió a la instalación, lo cual conllevó varios minutos. Luego, se reinició la máquina virtual, presentado la interfaz de consola del sistema operativo, como se muestra en la Figura 39, concluyendo con una instancia lista para su uso.

Figura 39

Interfaz de consola del sistema operativo CentOS Stream 8.

Para completar el proceso de creación de máquinas virtuales, se realizó la actualización de paquetes del sistema operativo CentOS Stream 8. La actualización se la realizó mediante la ejecución del siguiente comando desde el terminal del sistema operativo:

- sudo dnf update

Configuración de red de las máquinas virtuales

Posteriormente se realizó la configuración de red cada máquina virtual, para lo cual se accedió el archivo: “ifcfg-enp0s3”, y se accedió mediante el siguiente comando:

- sudo nano /etc/sysconfig/network-scripts/ifcfg-enp0s3

Se modificó el parámetro “BOOTPROTO” en el archivo de configuración, con el objetivo de asignar una dirección estática a las máquinas virtuales, tal como se muestra a continuación:

- BOOTPROTO=none

Se añadió los siguientes parámetros en el archivo de configuración, tomando en cuenta las configuraciones de red presentadas en la Tabla 5 para cada máquina virtual:

- DNS1=192.168.XXX.XXX
- IPADDR=192.168.XXX.XXX
- PREFIX=24
- GATEWAY=192.168.XXX.XXX

Completada las configuraciones presentadas anteriormente se guardó los cambios del archivo, y se reinició las máquinas virtuales con el objetivo de reflejar los cambios realizados. El reinicio de las máquinas virtuales se realizó mediante el comando:

- shutdown -r

2.2.6 Configuración de servidor remoto

Para el cumplimiento de la tercera arquitectura del proyecto, se simuló la configuración de un servidor remoto dentro de la red local que se encuentra el Cloud FICA. Mediante el software de virtualización KVM se creó una máquina virtual que cuenta con un sistema operativo CentOS Stream 8 y que utilizó la configuración de red definida en la Tabla 5.

Para la creación de la máquina virtual se siguió los pasos definidos en la sección 2.2.4, y para la instalación del sistema operativo los pasos definidos en la sección 2.2.5. La configuración de este equipo permitió la simulación de tener varios servidores físicos dentro de la institución, y así verificar el funcionamiento de la técnica de alta disponibilidad.

2.2.7 Descarga y configuración de Wildfly

Completado la configuración y puesto en funcionamiento de cada máquina virtual de las diferentes arquitecturas, se realizó la descarga y configuración del servidor de aplicaciones Wildfly en cada instancia como se describe a continuación.

Como requisito para el funcionamiento del servidor de aplicaciones Wildfly, se realizó la instalación de Java en su versión 17 mediante la ejecución del siguiente comando en la terminal:

- sudo dnf install -y java-17-openjdk-devel

A continuación, se realizó la descarga de los ficheros correspondientes al servidor de aplicaciones Wildfly en su versión 26.1.3 desde su repositorio oficial mediante el siguiente comando:

- wget https://github.com/wildfly/wildfly/releases/download/26.1.3.Final/wildfly.26.1.3.Final.tar.gz -P /tmp

Los ficheros comprimidos se descargaron en la ruta: “/tmp”, y posteriormente se realizó la descomprensión en el directorio: “/opt”, mediante el siguiente comando:

- sudo tar xf /tmp/wildfly-26.1.3.Final.tar.gz -C /opt/

Los ficheros descomprimidos se ubicaron en la ruta “/opt”, y posteriormente se creó un enlace simbólico al directorio de instalación de Wildfly:

- sudo ln -s /opt/wildfly-26.1.3.Final opt/wildfly

Se creó un nuevo usuario y un grupo de sistema denominado:” wildfly”. Además, se definió el directorio de inicio de sesión del nuevo usuario.

- sudo groupadd -r wildfly
- sudo useradd -r -g wildfly -d /opt/wildfly

Se cambió la propiedad del directorio de la instalación para la ejecución bajo el usuario y grupo creado:

- sudo chown -RH wildfly: /opt/wildfly

Con la finalidad de acceder a la consola de administración web de Wildfly y a la herramienta de línea de comandos (CLI) se creó los diferentes usuarios de Wildfly presentados en la Tabla 6.

Tabla 6

Definición de usuarios de Wildfly.

Máquina virtual	Usuario	Descripción
mv-srvbc	adminBalanceador	Usuario encargado para la administración del subsistema de Modcluster.
mv-srvapp1	adminMaestro	Usuario encargado para la administración de los servidores de aplicaciones.
mv-srvapp1	nodoEsclavo	Usuario utilizado para realizar la configuración y conexión con la máquina maestro.
mv-cloud-srvbc	adminBalanceador	Usuario encargado para la administración del subsistema de Modcluster.

mv-cloud-srvapp1	adminMaestro	Usuario encargado para la administración de los servidores de aplicaciones.
mv-cloud-srvapp1	nodoEsclavo	Usuario utilizado para realizar la configuración y conexión con la máquina maestro.

Para la creación de los usuarios se accedió al directorio que contiene el archivo que ejecuta el script para la creación del usuario. Posteriormente se ejecutó el script de creación mediante los siguientes comandos:

- cd /opt/wildfly/bin
- sudo ./add-user.sh

Como se observa en la Figura 40, se escogió la opción: “a) Management User (mgmt-users.properties)” y se ingresó los parámetros de nombre de usuario y contraseña. Adicionalmente se confirmó que el nuevo usuario se añada al “ManageRealm” y se confirmó que el usuario se habilite para la conexión con otras instancias.

Figura 40

Salida de pantalla del script de creación de usuarios de Wildfly

```
[mvsrvapp1@srvapp1 bin]$ sudo /opt/wildfly/bin/add-user.sh
[sudo] password for mvsrvapp1:

What type of user do you wish to add?
 a) Management User (mgmt-users.properties)
 b) Application User (application-users.properties)
(a): a

Enter the details of the new user to add.
Using realm 'ManagementRealm' as discovered from the existing property files.
Username : adminsrvapp1
Password recommendations are listed below. To modify these restrictions edit the add-user.properties configuration file.
 - The password should be different from the username
 - The password should not be one of the following restricted values {root, admin, administrator}
 - The password should contain at least 8 characters, 1 alphabetic character(s), 1 digit(s), 1 non-alphanumeric symbol(s)
Password :
Re-enter Password :
What groups do you want this user to belong to? (Please enter a comma separated list, or leave blank for none)[  ]:
About to add user 'adminsrvapp1' for realm 'ManagementRealm'
Is this correct yes/no? yes
Added user 'adminsrvapp1' to file '/opt/wildfly-26.1.3.Final/standalone/configuration/mgmt-users.properties'
Added user 'adminsrvapp1' to file '/opt/wildfly-26.1.3.Final/domain/configuration/mgmt-users.properties'
Added user 'adminsrvapp1' with groups  to file '/opt/wildfly-26.1.3.Final/standalone/configuration/mgmt-groups.properties'
Added user 'adminsrvapp1' with groups  to file '/opt/wildfly-26.1.3.Final/domain/configuration/mgmt-groups.properties'
Is this new user going to be used for one AS process to connect to another AS process?
e.g. for a slave host controller connecting to the master or for a Remoting connection for server to server Jakarta Enterprise Beans calls.
yes/no? yes
To represent the user add the following to the server-identities definition <secret value="QWRtaW5fMTIz" />
[mvsrvapp1@srvapp1 bin]$
```

2.2.8 Configuración de Modo Dominio

Configuración de Host Wildfly Maestro

Esta configuración se la realizó en las máquinas virtuales denominadas como “Maestras” establecidas en las diferentes arquitecturas del proyecto, para lo cual primero se

accedió al directorio que contiene el archivo de configuración de dominio para la máquina maestra denominado “host-master.xml” mediante los siguientes comandos:

- cd /opt/wildfly/domain/configuration
- sudo nano host-master.xml

En este archivo se añadió en la configuración las interfaces, tal como se muestra en la Figura 41, denominadas “private” e “unsecured” con el objetivo de establecer un punto de conexión entre la máquina maestra y la máquina esclava.

Figura 41

Interfaces del archivo de configuración

```
<interfaces>
    <interface name="management">
        <inet-address value="${jboss.bind.address.management:192.168.122.101}"/>
    </interface>
    <interface name="public">
        <inet-address value="${jboss.bind.address:192.168.122.101}"/>
    </interface>
    <interface name="private">
        <inet-address value="192.168.122.101"/>
    </interface>

    <interface name="unsecured">
   <inet-address value="192.168.122.101" />
</interface>
 </interfaces>
```

Configuración de Host Wildfly Esclavo

Esta configuración se la realizó en las máquinas virtuales denominadas como “Esclavas” establecidas en las diferentes arquitecturas del proyecto. El proceso de configuración se detalla a continuación.

Se accedió al directorio que contiene el archivo de configuración de dominio para la máquina maestra denominado “host-slave.xml” mediante los siguientes comandos:

- cd /opt/wildfly/domain/configuration
- sudo nano host-slave.xml

En el archivo “host-slave.xml” abierto anteriormente se procedió a realizar las siguientes configuraciones:

- Es primeras líneas de este archivo se añadió el parámetro “name=slave”, tal como se muestra en la Figura 42.

Figura 42

Definición de parámetro “name” en archivo de configuración

```
<?xml version="1.0" ?>

<host xmlns="urn:jboss:domain:19.0" name="slave">
```

- Se añadió las interfaces denominadas “private” y “unsecured”, como se muestra en la Figura 43.

Figura 43

Interfaces del archivo de configuración

```
<interfaces>
    <interface name="management">
        <inet-address value="${jboss.bind.address.management:192.168.122.102}"/>
    </interface>
    <interface name="public">
        <inet-address value="${jboss.bind.address:192.168.122.102}"/>
    </interface>
    <interface name="private">
        <inet-address value="192.168.122.102"/>
    </interface>
    <interface name="unsecured">
   <inet-address value="192.168.122.102" />
</interface>
</interfaces>
<jvms>
```

- Como se muestra en la Figura 44, se añadió el método de autenticación para la conexión con la máquina maestra, en el que se incluyó el nombre de usuario y contraseña de Wildfly definida para la conexión con la máquina maestra, definido en la Tabla 6.

Figura 44

Método de autenticación con la máquina maestra.

```
<subsystem xmlns="urn:jboss:domain:core-management:1.0"/>
<subsystem xmlns="urn:wildfly:elytron:15.1" final-providers="combined-providers" disallowed-providers="OracleUcrypto">
    <authentication-client>
        <authentication-configuration name="hostAuthConfig" authentication-name="nodoesclavo" realm="ManagementRealm" sasl-mechanism-selector="DIGEST-MD5">
            <credential-reference clear-text="adminesclavo"/>
        </authentication-configuration>
        <authentication-context name="hcAuthContext">
            <match-rule authentication-configuration="hostAuthConfig"/>
        </authentication-context>
    </authentication-client>
```

- Se añadió los parámetros del controlador de dominio, indicando la dirección IP de la máquina maestra, el nombre de usuario, el contexto de autenticación, el puerto y el protocolo.

Figura 45

Configuración del controlador de dominio

```
<domain-controller>
     <remote protocol="http-remoting" host="192.168.122.101" port="9990"
             username="nodoesclavo" authentication-context="hcAuthContext"/>
</domain-controller>
```

Ejecución de servidores de aplicaciones

Como paso anterior a la ejecución de los servidores de aplicaciones, se realizó la configuración de los puertos del firewall TCP y UDP necesarios para el funcionamiento y comunicación entre los nodos del grupo de servidores. Los puertos que se configuró son los siguientes:

- Puerto TCP 8009: Puerto utilizado por el protocolo Apache JServ (AJP) para el balanceo de carga y comunicación de los nodos de un clúster HTTP.
- Puerto TCP 9990: Puerto de la consola de administración web.
- Puerto UDP 45688: Puerto utilizado para el descubrimiento de nodos en clústers HA mediante UDP.

La configuración de los diferentes puertos se lo realizó mediante el siguiente comando:

- sudo firewall-cmd --zone=public --add-port= *nro. puerto / protocolo* –permanent
- sudo firewall-cmd --reload

Completado la configuración del firewall, se procede a la ejecución de los servidores. Para todas las instalaciones de Wildfly, los scripts de ejecución se encuentran en el directorio “/bin”, y se accedió mediante el comando:

- cd /opt/wildfly/bin

Para la ejecución de los servidores denominados “Maestros” se lo realizó mediante la ejecución del siguiente script:

- sudo ./domain.sh --host-config=host-master.xml -b *dirección-ip-serv-maestro* -bmanagement *dirección-ip-serv-maestro* -Djgroups.bind_addr=*dirección-ip-serv-maestro*

Para la ejecución de los servidores denominados “Esclavos” se lo realizó mediante la ejecución del siguiente script ubicado en el directorio “/opt/wildfly/bin”:

- sudo ./domain.sh --host-config=host-slave.xml -b *dirección-ip-serv-esclavo* -Djboss.domain.master.address= *dirección-ip-serv-maestro* -Djgroups.bind_addr= *dirección-ip-serv-esclavo*

Los parámetros de ejecución para los diferentes servidores de aplicaciones se detallan a continuación:

- -b: Este parámetro indica la dirección de red que utilizará el servidor de aplicaciones.
- -bmanagement: Este parámetro indica la dirección de red que utilizará el servidor de aplicaciones para el acceso a sus opciones de administración.
- -Djboss.domain.master.address: Este parámetro indica la dirección de red del controlador de host maestro.
- -Djgroups.bind_addr: Este parámetro indica la dirección de red que se utilizará para la configuración de la interfaz de JGroups, y es la encargada de proporcionar un soporte de comunicación grupal para los servicios de alta disponibilidad entre los diferentes nodos.

Configuración de grupo de servidores.

Para la creación y configuración del grupo de servidores de Wildfly se accedió a la consola de administración web de la máquina maestra, mediante el siguiente enlace en el navegador:

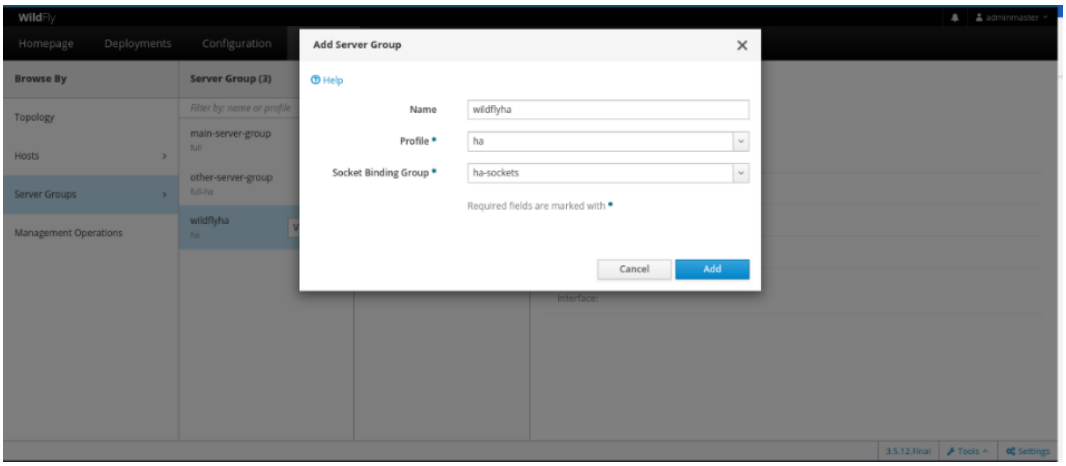
- http://*dirección-ip-serv-maestro*:9990/console/index.html

Esta configuración se realizó únicamente en los servidores denominados Maestros, y que se detalla a continuación.

Una vez ingresado a la consola de administración web se dirigió a la sección denominada “Runtime” , y en la opción de “Add Server Group” se creó el grupo de servidores denominado “wildflyha”, asignándole un perfil de alta “ha” y un grupo de enlace de socket “ha-sockets”, como se muestra en la Figura 46.

Figura 46

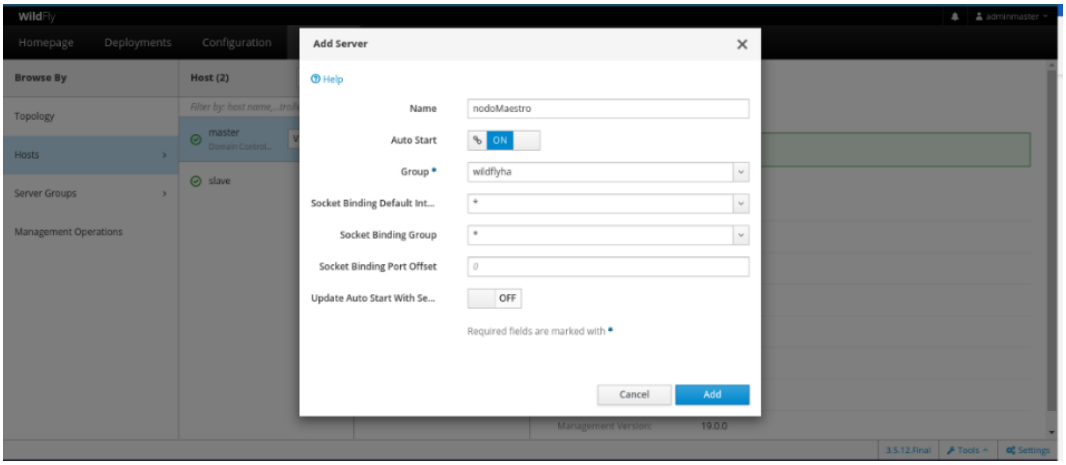
Creación de grupo de servidores



Como se muestra en la Figura 47, se añadió el servidor definido como “nodoMaestro” para el host principal o también denominado “master”. A este servidor se le asignó el grupo de servidores “wildflyha” creado anteriormente.

Figura 47

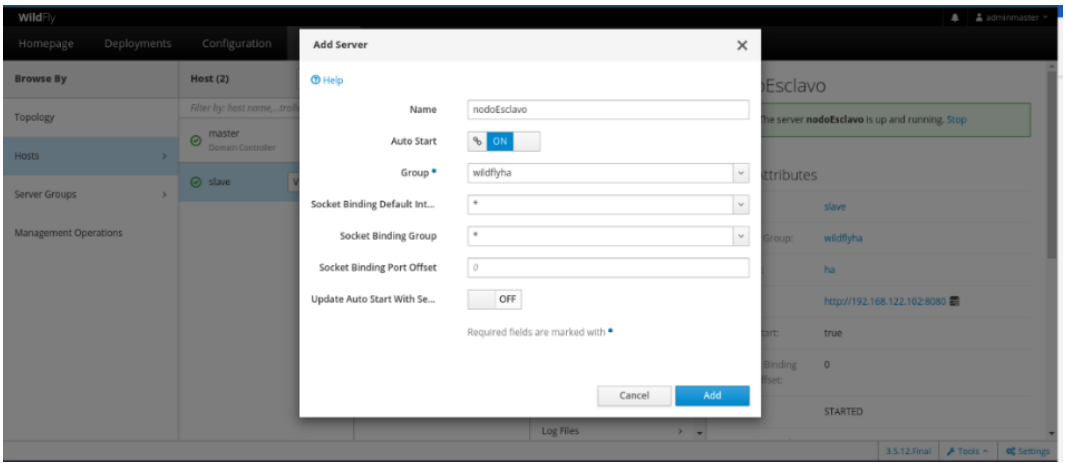
Creación de nodoMaestro



Como se muestra en la Figura 48, se añadió el servidor definido como “nodoEsclavo” para el host secundario o también denominado “slave”. A este servidor se le asignó el grupo de servidores “wildflyha” creado anteriormente.

Figura 48

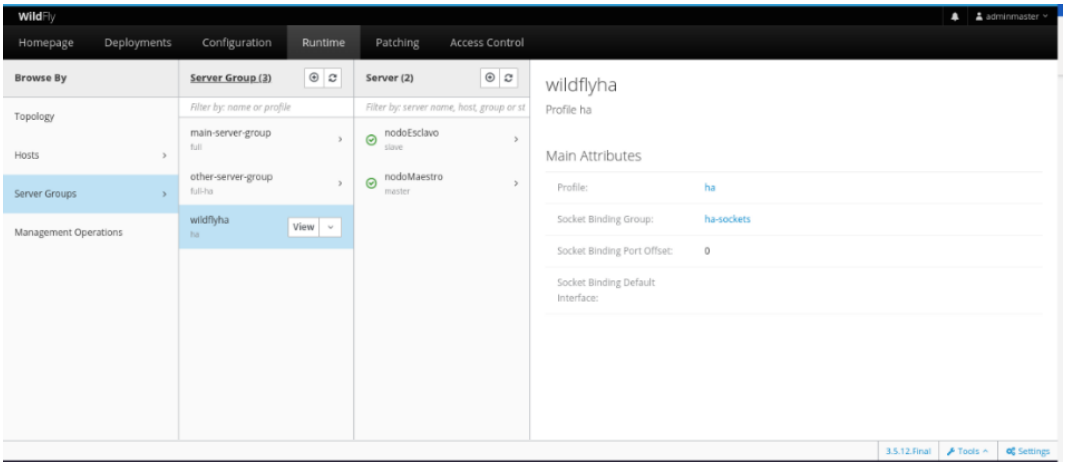
Creación de nodoEsclavo



En la Figura 49 se presenta el grupo de servidores con los nodos creados anteriormente.

Figura 49

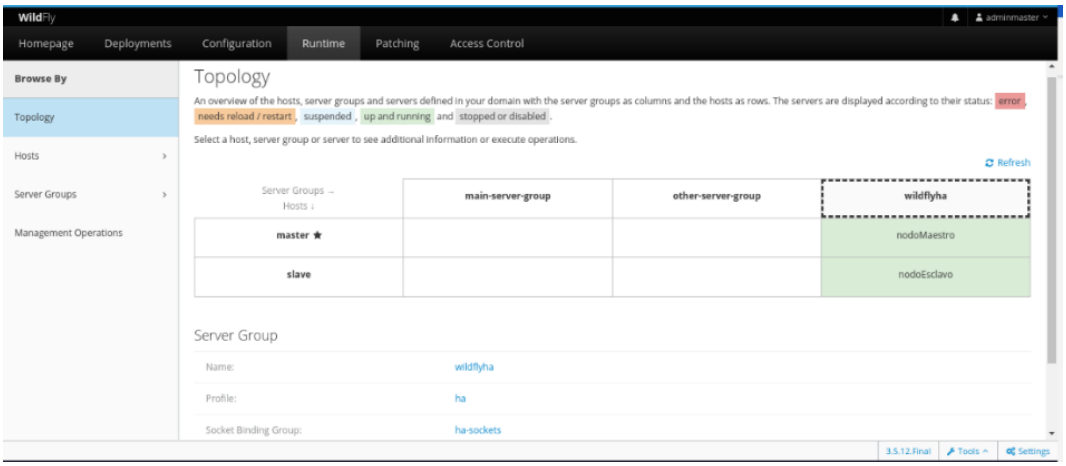
Grupo de servidores con nodos correspondientes



En la Figura 50 se presenta de forma visual la topología final configurada, indicando adicionalmente que están funcionales y listo para el despliegue de aplicaciones.

Figura 50

Topología de grupo de servidores



2.2.9 Configuración del balanceador de carga

El subsistema Modcluster permite utilizar una instancia de Wildfly como balanceador de carga y que adicionalmente tenga funciones de conmutación de sesiones por error de un nodo dentro de un grupo de servidores. Este servicio se ejecutó en las máquinas virtuales denominadas “Nodo balanceador de carga” de las diferentes arquitecturas, y su configuración se detalla a continuación.

Como punto inicial, se realizó la configuración de los puertos del firewall TCP y UDP necesarios para la comunicación con las consolas de administración y comunicación de las aplicaciones web hacia los usuarios. Los puertos que se configuró son los siguientes:

- Puerto TCP 9990: Puerto de la consola de administración web.
- Puerto TCP 8080: Puerto por defecto para la comunicación de las aplicaciones web implementadas.

La configuración de los diferentes puertos se lo realizó mediante el siguiente comando:

- sudo firewall-cmd --zone=public --add-port= *nro. puerto* / *protocolo* –permanent
- sudo firewall-cmd –reload

Posteriormente, se puso en funcionamiento al servidor en su configuración “standalone” con el archivo de configuración “standalone-ha.xml”. Para esta terea se accedió al directorio “/opt/wildfly/bin” y se ejecutó el servidor mediante los siguientes comandos:

- cd /opt/wildfly/bin

- sudo ./standalone.sh -c standalone-ha.xml -b *dirección-ip-balanceador* -bmanagement *dirección-ip-balanceador*

  Los parámetros del comando anterior se detallan a continuación:

- -c: Este parámetro especifica el archivo de configuración que ejecutará el servidor de aplicaciones.
- -b: Este parámetro indica la dirección de red que utilizará el servidor de aplicaciones.
- -bmanagement: Este parámetro indica la dirección de red que utilizará el servidor de aplicaciones para el acceso a sus opciones de administración

Una vez inicializado el servicio, se configuró el subsistema Modcluster mediante la herramienta de línea de comandos (CLI) de Wildfly , la cual se accedió mediante la ejecución de los siguientes comandos en la terminal.

- ./jboss-cli.sh—connect—controller=*dirección-ip-balanceador*:9990

El subsistema Undertow tiene la capacidad de actuar como un proxy inverso de alto rendimiento compatible con Modcluster. Mediane ejecución de los siguientes comandos la herramienta de línea de comandos (CLI) de Wildfly se realizó la configuración del filtro y el grupo de multidifusión necesario para el servicio de Modcluster:

- /subsystem=undertow/configuration=filter/mod-cluster=modcluster:add(advertise-socket-binding=modcluster,management-socket-binding=http)
- /subsystem=undertow/server=default-server/host=default-host/filter-ref= modcluster:add
- :reload-servers

Finalmente se realizó la configuración de los servidores de aplicaciones maestros para su integración con el balanceador de carga. Se configuró mediante la mediante la herramienta de línea de comandos (CLI) de Wildfly, ejecutando los siguientes comandos en la consola.

- ./jboss-cli.sh --connect --controller=*dirección-ip-serv-maestro*:9990

Para el grupo de sockets de alta disponibilidad “ha-sockets”, se definió una lista estática de proxies mediante la creación de un socket remoto que contendrá el host del servidor que contiene el servicio de Modcluster juntamente con el puerto de comunicación.

- /socket-binding-group=ha-sockets/remote-destination-outbound-socket-binding=proxy1:add(host=*dirección-ip-serv-maestro*, port=8080)

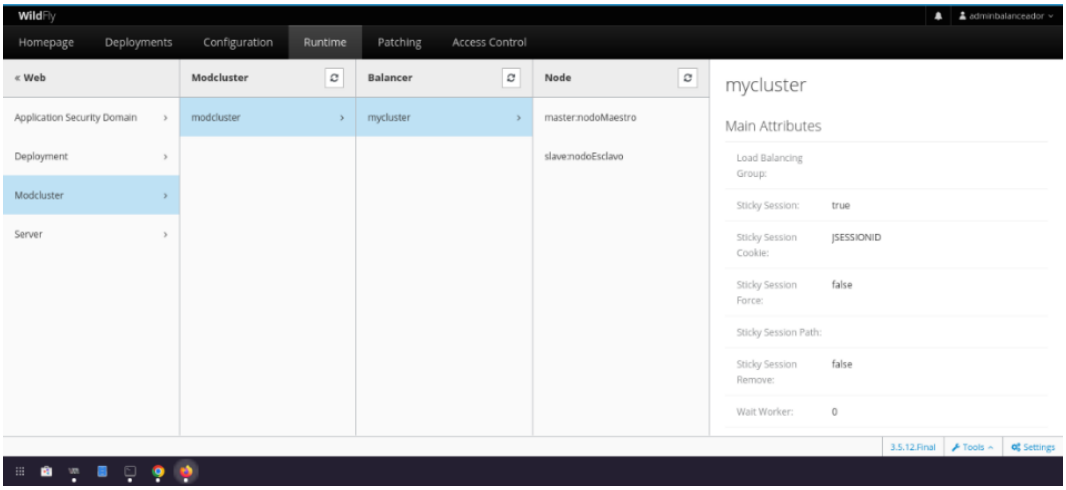
Para el perfil de alta disponibilidad “ha” se hizo referencia al socket remoto añadiendo el atributo “proxies” y el valor establecido en el punto anterior.

- /profile=ha/subsystem=modcluster/proxy=default:write-attribute (name=proxies,value=[proxy1])

Finalmente, para comprobar la conexión del subsistema con los nodos del servidor de aplicaciones, se accedió a la interfaz web de administración de Wildfly mediante el enlace dirección “http://*dirección-ip-balanceador*:9990/console/index.html” en el navegador. En la Figura 51 se presenta la ventana correspondiente a la información general del subsistema Modcluster, donde se presenta la información del balanceador y de los nodos configurados.

Figura 51

Interfaz web de administración de Wildfly para Modcluster



2.2.10 Demostración de balanceo de carga

Despliegue de aplicación demo.

Para la verificación de las funcionalidades del balanceador de carga y de alta disponibilidad del servidor de aplicaciones, se realizó el despliegue de la aplicación JavaEE

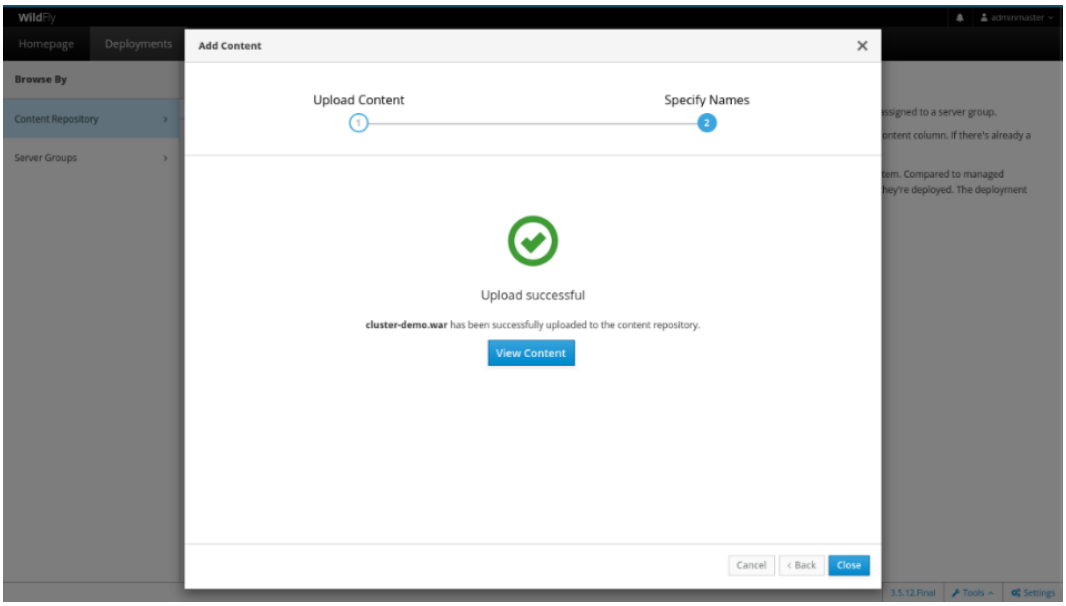
denominada “cluster-demo.war” mediante la consola web de administración. Esta aplicación proporcionada por Wildfly se puede acceder mediante el repositorio:

- https://github.com/liweinan/cluster-demo

En la Figura 52 se presenta la subida del archivo “cluster-demo.war” al repositorio de contenido del servidor de aplicaciones Wildfly.

Figura 52

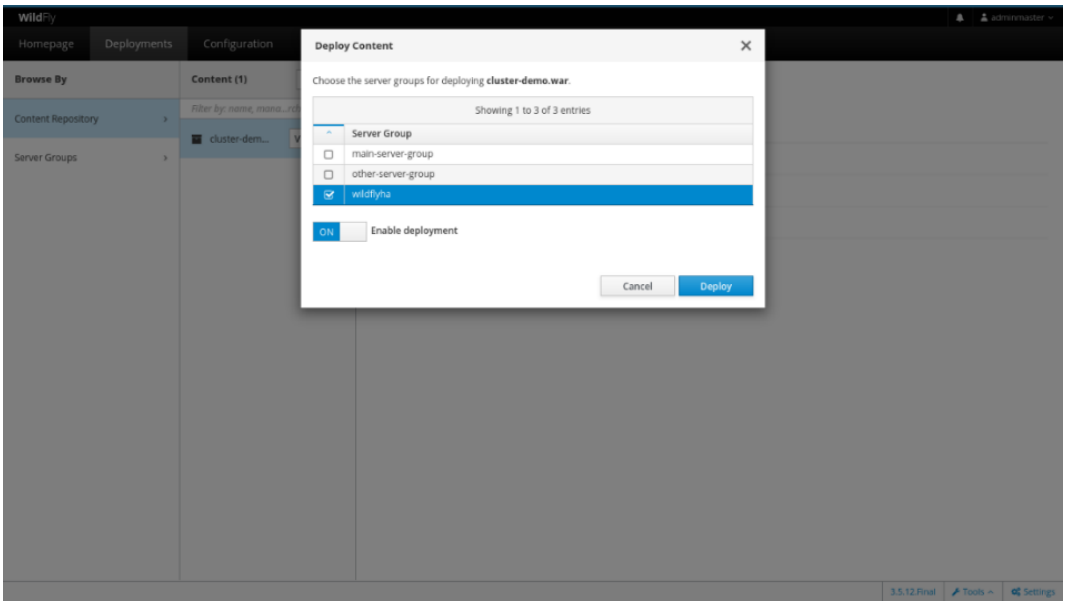
Subida del archivo “cluster-demo.war” al repositorio de contenido



En la Figura 53 se presenta el despliegue de la aplicación al grupo de servidores de alta disponibilidad “wildflyha”, y que gracias a su configuración de dominio se ejecutará en los diferentes nodos del grupo de aplicaciones.

Figura 53

Despliegue de aplicación en grupo de servidores "wildflyha"



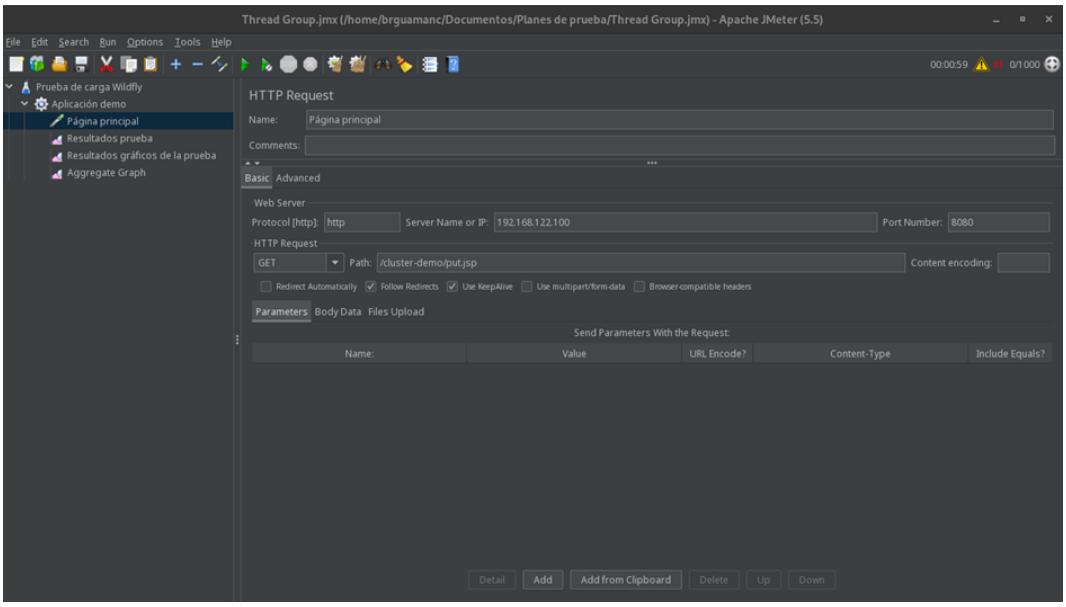
Pruebas de carga.

Se estableció un plan de pruebas mediante el software Apache JMeter para comprobar el funcionamiento del balanceo de carga entre nodos del grupo de servidores de aplicaciones. En la Figura 54 se observa los diferentes parámetros que fueron establecidos para la realización de la prueba de carga y que se detallarán a continuación.

- Protocol (http): htttp
- Server Name or IP: En este parámetro se define la dirección de enlace del balanceador de carga para la redirección de peticiones.
- Port Number: 8080
- HTTP Request: Get
- Path: /cluster-demo/put.jsp

Figura 54

Plan de pruebas



Se estableció diferentes números usuarios (50,100,200,500,1000) que accederán al servidor de balanceado de carga en un intervalo de 60 segundos, tal como se muestra en la Figura 55, con el objetivo de observar el número de cargas que se ha redirigido a cada nodo del grupo de servidores de aplicaciones, verificado mediante la consola de administración web de Wildfly mediante el atributo “Elected”, que representa el número de peticiones que fueron atendidas por cada nodo, tal como se muestra en la Figura 56.

Figura 55

Número de usuarios y periodo de tiempo de la prueba.

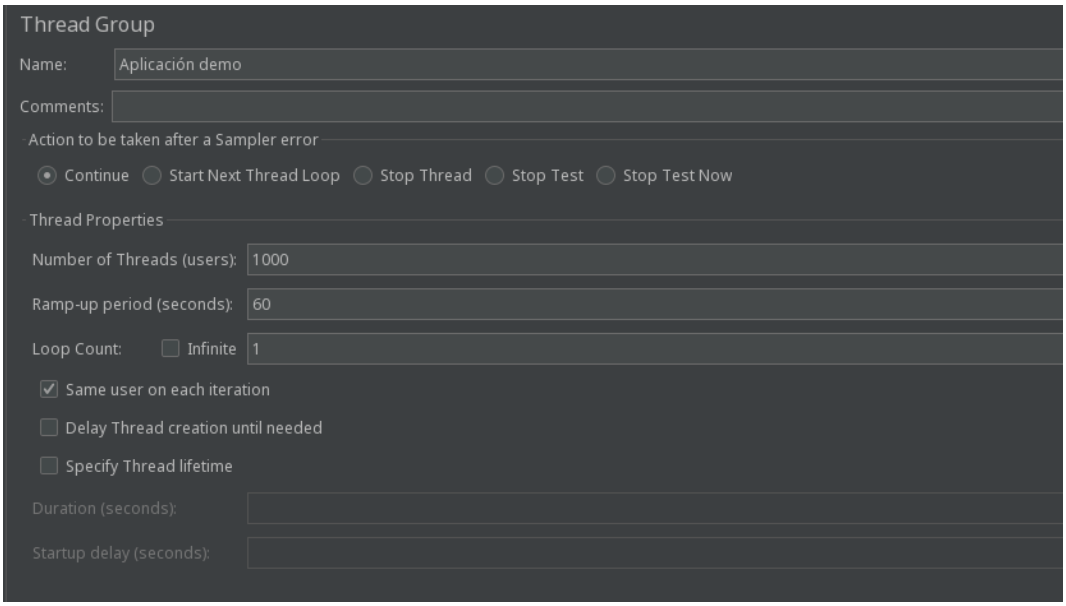


Figura 56

Ejemplo de visualización de número de peticiones recibidas en el nodo Esclavo

A continuación, se presenta los diferentes resultados de la prueba de distribución de carga presentados en cada arquitectura de proyecto, tomando en cuenta los parámetros generales definidos anteriormente.

En la Tabla 7 se presentan los resultados obtenidos en la primera arquitectura posterior a la prueba de carga.

Tabla 7

Resultados de prueba de balanceo de carga de la primera arquitectura

Número de usuarios	**Periodo de tiempo de la prueba (seg)**	**Nodo del grupo de servidores de Wildfly**	**Número de peticiones recibidas por nodo**
50	60	Nodo Maestro	23
		Nodo Esclavo	27
100	60	Nodo Maestro	52
		Nodo Esclavo	48
200	60	Nodo Maestro	112
		Nodo Esclavo	88
500	60	Nodo Maestro	244
		Nodo Esclavo	256
1000	60	Nodo Maestro	489
		Nodo Esclavo	511

En la Tabla 8 se presentan los resultados obtenidos en la segunda arquitectura posterior a la prueba de carga.

Tabla 8

Resultados de prueba de balanceo de carga de la segunda arquitectura

Número de usuarios	Periodo de tiempo de la prueba (seg)	Nodo del grupo de servidores de Wildfly	Número de peticiones recibidas por nodo
50	60	Nodo Maestro	24
		Nodo Esclavo	26
100	60	Nodo Maestro	55
		Nodo Esclavo	45
200	60	Nodo Maestro	107
		Nodo Esclavo	93
500	60	Nodo Maestro	248
		Nodo Esclavo	252
1000	60	Nodo Maestro	496
		Nodo Esclavo	504

En la Tabla 9 se presentan los resultados obtenidos en la tercera arquitectura posterior a la prueba de carga.

Tabla 9

Resultados de prueba de balanceo de carga de la tercera arquitectura

Número de usuarios	Periodo de tiempo de la prueba (seg)	Nodo del grupo de servidores de Wildfly	Número de peticiones recibidas por nodo
50	60	Nodo Maestro	28
		Nodo Esclavo	22
100	60	Nodo Maestro	47
		Nodo Esclavo	53
200	60	Nodo Maestro	106
		Nodo Esclavo	94
500	60	Nodo Maestro	255
		Nodo Esclavo	245
1000	60	Nodo Maestro	502
		Nodo Esclavo	498

Pruebas de disponibilidad

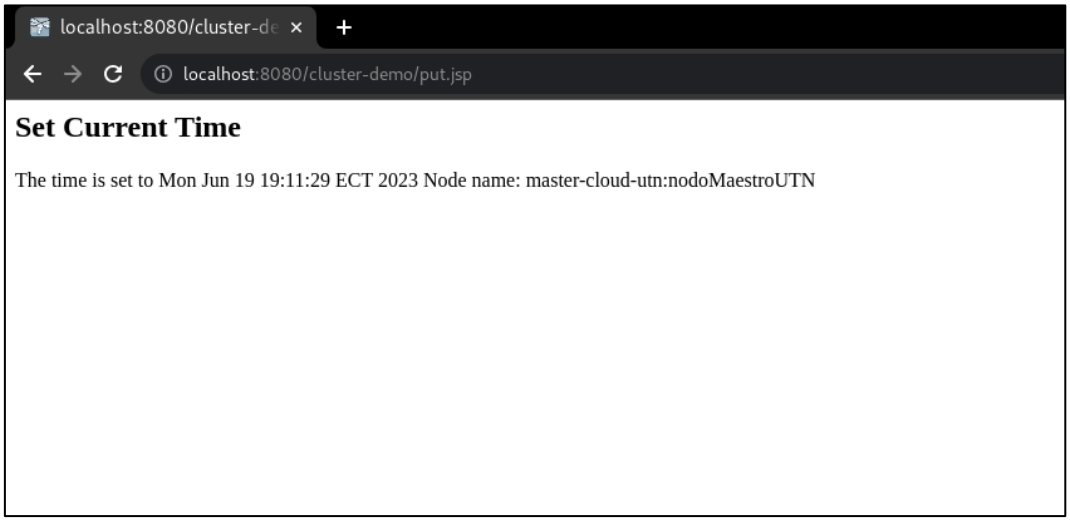
Para la realización de las pruebas de disponibilidad se accedió a la de aplicación demo desplegada, la cual consta de dos páginas: la primera denominada “put.jsp” está encargada de capturar en sesión la fecha y hora en la que se accedió y de mostrar el nodo que atendió la petición, y la página denominada “get.jsp” que está encargada de mostrar la hora almacenada en sesión. Mediante la simulación de indisponibilidad del nodo que atendió la petición se pude comprobar que los datos de sesión del usuario se han replicado en el siguiente nodo y la navegación continua normalmente, confirmando así que se configuró correctamente la solución de alta disponibilidad.

El proceso de la prueba de disponibilidad se replicó en cada arquitectura del proyecto y se detalla a continuación:

- Primero, se accedió a la página “put.jsp” en el navegador, donde se mostrará y se almacenará en sesión la fecha y hora actual, juntamente con el nombre del nodo que atendió la aplicación. En la Figura 57 se observa que en la prueba realizada la petición fue dirigida al nodo maestro. La petición puede ser atendida inicialmente por el nodo maestro o por el nodo esclavo, y no se considera como un problema para el desarrollo de la prueba.

Figura 57

Visualización de fecha, hora y nombre del nodo que atendió la petición



- A continuación, se accedió a la página “get.jsp” en el navegador, donde se mostrará los datos de sesión del usuario que fueron definidos en el punto anterior, tal como se presenta en la Figura 58.

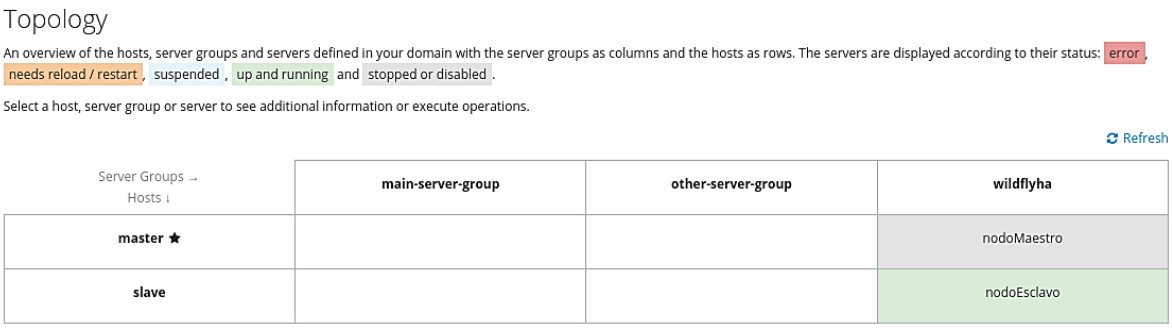
Figura 58

Visualización de los datos generados por la aplicación demo

- Se realizó la prueba de indisponibilidad el nodo que recibió la petición mediante la consola de administración web de Wildfly. En esta demostración se suspendió el nodo maestro, tal como se muestra en la Figura 59.

Figura 59

Visualización gráfica del estado de suspensión del nodo Maestro



- Se recargó la página y se verificó que el navegador no presente mensajes de error o de problemas de conexión. Adicionalmente se verificó que los datos de sesión del usuario se visualizan con normalidad, tal como se presenta en la Figura 60. De esta manera se comprueba que se trasladó los datos de sesión del usuario al siguiente nodo.

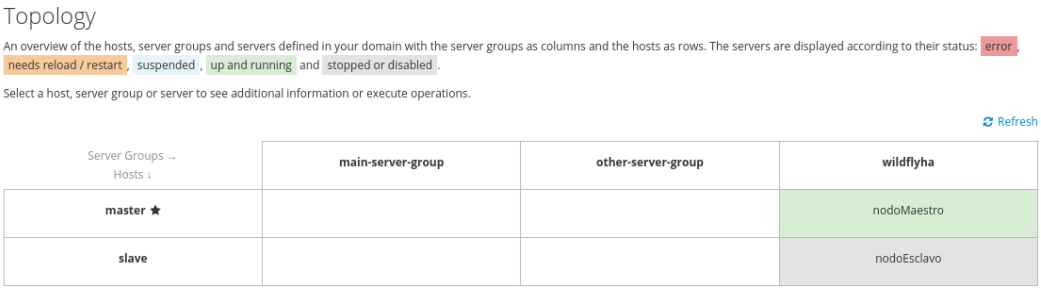
Figura 60

Visualización de los datos guardados en sesión del usuario

- Para completar con la prueba, se replicó la prueba simulando la indisponibilidad del nodo que recibió inicialmente la petición, tal como se muestra en la Figura 61.

Figura 61

Visualización gráfica del estado de suspensión del nodo Esclavo



A continuación, se presenta los diferentes resultados de la prueba de disponibilidad realizados en cada arquitectura de proyecto:

- En la Tabla 10 se presentan los resultados de las pruebas de disponibilidad realizadas en la primera arquitectura.

Tabla 10

Resultados de pruebas de disponibilidad de la primera arquitectura

Descripción de la prueba	Resultado esperado	Cumple (Si/No)
Simular la indisponibilidad del nodo maestro.	Al recargar la página en el navegador no se presentan mensajes en pantalla de error de conexión o de interrupción.	Si
	Al recargar la página en el navegador no se pierde la comunicación a la aplicación desplegada.	Si
	Al recargar la página del navegador los datos de sesión se mantienen y el usuario puede continuar navegando con normalidad.	Si
Simular la indisponibilidad del nodo esclavo.	Al recargar la página en el navegador no se presentan mensajes en pantalla de error de conexión o de interrupción.	Si
	Al recargar la página en el navegador no se pierde la comunicación a la aplicación desplegada.	Si
	Al recargar la página del navegador los datos de sesión se mantienen y el usuario puede continuar navegando con normalidad.	Si

- En la Tabla 11 se presentan los resultados de las pruebas de disponibilidad realizadas en la segunda arquitectura.

Tabla 11

Resultados de pruebas de disponibilidad de la segunda arquitectura

Descripción de la prueba	Resultado esperado	Cumple (Si/No)
Simular la indisponibilidad del nodo maestro.	Al recargar la página en el navegador no se presentan mensajes en pantalla de error de conexión o de interrupción.	Si
	Al recargar la página en el navegador no se pierde la comunicación a la aplicación desplegada.	Si
	Al recargar la página del navegador los datos de sesión se mantienen y el usuario puede continuar navegando con normalidad.	Si
Simular la indisponibilidad del nodo esclavo.	Al recargar la página en el navegador no se presentan mensajes en pantalla de error de conexión o de interrupción.	Si
	Al recargar la página en el navegador no se pierde la comunicación a la aplicación desplegada.	Si
	Al recargar la página del navegador los datos de sesión se mantienen y el usuario puede continuar navegando con normalidad.	Si

- En la Tabla 12 se presentan los resultados de las pruebas de disponibilidad realizadas en la tercera arquitectura.

Tabla 12

Resultados de pruebas de disponibilidad de la tercera arquitectura

Descripción de la prueba	Resultado esperado	Cumple (Si/No)
Simular la indisponibilidad del nodo maestro.	Al recargar la página en el navegador no se presentan mensajes en pantalla de error de conexión o de interrupción.	Si
	Al recargar la página en el navegador no se pierde la comunicación a la aplicación desplegada.	Si
	Al recargar la página del navegador los datos de sesión se mantienen y el usuario puede continuar navegando con normalidad.	Si
Simular la indisponibilidad del nodo esclavo.	Al recargar la página en el navegador no se presentan mensajes en pantalla de error de conexión o de interrupción.	Si
	Al recargar la página en el navegador no se pierde la comunicación a la aplicación desplegada.	Si
	Al recargar la página del navegador los datos de sesión se mantienen y el usuario puede continuar navegando con normalidad.	Si

2.3 Guía de implementación de alta disponibilidad del servidor de aplicaciones Wildfly

2.3.1 Propósito de la guía

Esta guía tiene la finalidad de ser un modelo de referencia para la implementación de una tecnología que permita obtener una alta disponibilidad dentro del Cloud FICA de la Universidad Técnica del Norte para servidores de aplicaciones Java Enterprise, específicamente Wildfly. Además, la guía servirá como base para futuros proyectos de investigación de la institución, en las cuales se requiera del despliegue de una aplicación Java Enterprise en una arquitectura de alta disponibilidad, o ampliar el conocimiento acerca de tecnologías de alta disponibilidad.

2.3.3 Desarrollo de la guía

Para el cumplimiento del desarrollo de la guía se definió las siguientes secciones en la que está estructurada la guía y se detallan a continuación:

- Portada: En esta sección se definió los datos informativos de la institución, título de la guía, código de la guía y versión, nombre de encargado de la elaboración de la guía, la ubicación y año de desarrollo de la guía.
- Introducción: En esta sección se definió de forma general los antecedentes del proyecto, problemática y la solución desarrollada, como también una síntesis del funcionamiento de la técnica implementada.
- Objetivos: en esta sección se definió el objetivo general y los objetivos específicos a desarrollar en el proyecto.
- Desarrollo: En esta sección se describió la arquitectura del proyecto y el proceso de la implementación realizado, el cual se tomó del contenido presentado en la sección 2.2 del presente trabajo de grado y se adecuó de forma específica para el Cloud FICA de la Universidad Técnica del Norte. Adicionalmente, para la verificación y constancia del funcionamiento se describió las pruebas de carga y de disponibilidad realizadas a la aplicación demo desplegada.

- Conclusiones: En esta sección se definió las conclusiones presentadas luego de realizar la implementación de la técnica de alta disponibilidad.
- Observaciones y recomendaciones: En esa sección se definió las observaciones y recomendaciones que se podrían aplicar en futuros proyectos de investigación acerca de arquitecturas de alta disponibilidad para servidores de aplicaciones.

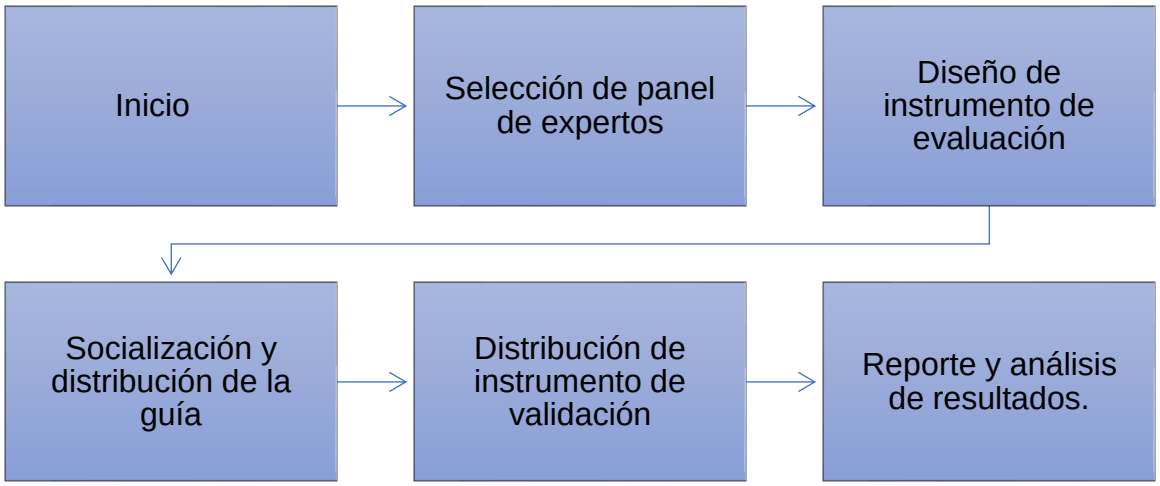
El resultado final del desarrollo de la guía se encuentra el Apéndice B del presente trabajo de grado.

2.3.4 Validación de la guía

Tomando como base el proceso de validación mediante Delphi de George & Liñan (2018) definido en la sección de materiales y métodos del presente trabajo, se definió el proceso presentado en la Figura 62 para la validación de la guía desarrollada en el presente trabajo de grado.

Figura 62

Metodología final para validación de guía



Como punto inicial, se definió el panel de expertos que realizaron el proceso de validación y que fueron tomados en cuenta en base de los siguientes aspectos:

- Ser profesionales en el área de tecnología o informática.
- Tener conocimiento en desarrollo de proyectos tecnológicos.
- Tener conocimiento en levantamiento de arquitecturas basadas en software libre.

El perfil profesional de los expertos que se seleccionó para realizar el proceso de validación se presenta a continuación:

Experto nro. 1: Ing. Henry Patricio Farinango Endara, tiene una formación académica de Ingeniero en Electrónica y Redes de Comunicación y actualmente desempeña las funciones de Técnico Docente en la Universidad Técnica del Norte en la Carrera de Ingeniería en Telecomunicaciones

Experto nro. 2: Ing. Santiago Javier Meneses Narváez, tiene una formación académica como Ingeniero en Electrónica y Redes de Comunicación y que actualmente desempeña las funciones de especialista en Infraestructura y Redes computacionales en la empresa Insoluciones.

Experto nro. 3: Ing. Pablo Andrés Landeta, tiene una formación académica como Ingeniero en Sistemas Computacionales y actualmente desempeña las funciones de docente en la Universidad Técnica del Norte en la Carrera de Ingeniería en Tecnologías de la Información, como también de Especialista en Infraestructura y uso de servicios AWS para la UTN.

Seleccionados los expertos, se procedió a la creación del instrumento que permitirá la evaluación de la guía base de los siguientes parámetros: Redacción y contenido, Claridad, y Replicabilidad. Cada parámetro constó de una serie de ítems a modo de pregunta con el fin de tener una validación objetiva, donde cada ítem será calificado en una escala de Likert de 1 a 4 donde 1 corresponde a “No cumple el criterio”, 2 corresponde a “Cumple parcialmente el criterio”, 3 corresponde a “Cumple aceptablemente el criterio” y 4 a “Cumple satisfactoriamente el criterio”. Para la presentación y evaluación de los diferentes parámetros al grupo de expertos se estructuró un formulario digital mediante la herramienta Microsoft Forms, y que es presentado en el Apéndice 3.

Completado la creación del instrumento de evaluación, se procedió a la socialización de la guía a cada uno de los expertos seleccionados con el objetivo que obtengan un contexto acerca del trabajo desarrollado. Posteriormente se distribuyó el cuestionario y se sociabilizó

el método de evaluación de cada uno de los parámetros, donde cada experto tuvo un tiempo respectivo tanto para la revisión de la guía como para la realización de la evaluación de esta.

Al completar el tiempo de revisión y evaluación por cada uno de los expertos, se obtuvo los diferentes resultados presentados en la Tabla 13, donde se presenta un resumen de las diferentes calificaciones obtenidas con su media obtenida por cada ítem.

Tabla 13

Resultados de proceso de validación.

Pará-metro	Nro. ítem	Enunciado ítem	Calf. Experto 1	Calf. Experto 2	Calf. Experto 3	Calf. media
Redacción y contenido	1	¿La guía contiene un lenguaje adecuado y comprensible para el público objetivo?	4	4	4	**4**
	2	¿La guía utiliza correctamente las normas gramaticales y ortográficas?	4	4	4	**4**
	3	¿La gráficas y tablas presentadas en la guía están adecuadamente etiquetadas y tituladas para facilitar su comprensión?	4	4	4	**4**
	4	¿La calidad visual de las figuras presentadas en la guía es adecuada para su interpretación?	3	3	3	**3**
Claridad	5	¿La guía es clara y fácil de entender en relación con su temática?	3	2	4	**3**
	6	¿La guía contiene una estructura organizada y coherente que facilite la comprensión y	4	3	4	**3.66**

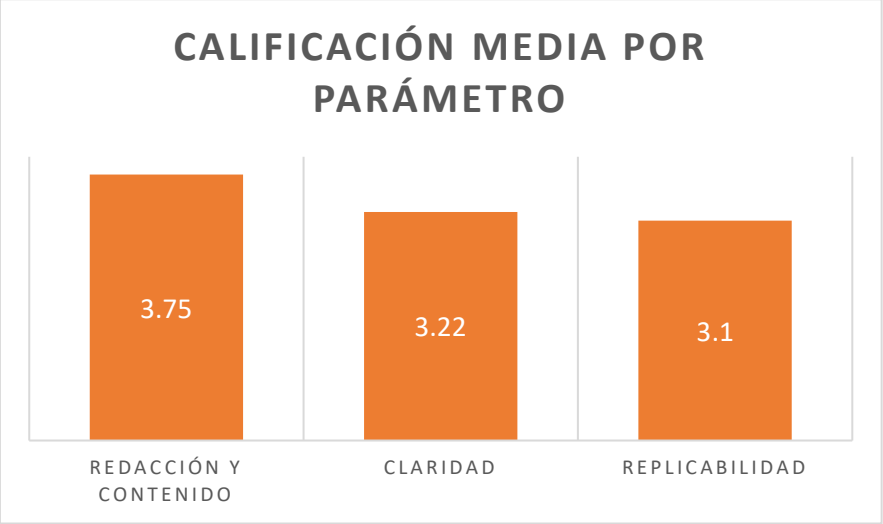
		seguimiento del contenido?				
	7	¿Se proporciona suficiente detalle y claridad en los pasos y configuraciones documentados para conseguir los objetivos planteados?	3	3	3	**3**
Replicabilidad	8	¿Se proporciona información detallada y suficiente para que otros investigadores puedan replicar la implementación de forma precisa?	3	3	3	**3**
	9	¿Las tecnologías utilizadas en la guía son confiables y adecuadas para replicar la implementación en otros entornos?	3	4	4	**3.66**
	10	¿La guía ofrece recomendaciones o sugerencias relacionadas con el tema desarrollado?	2	3	3	**2.66**

Se analizó la calificación media de cada ítem, donde se obtuvo una calificación de 3 puntos en la mayoría de los ítems presentados, lo cual representa como una calificación positiva. Sin embargo, se tomó en cuenta el ítem nro.10 ya que obtuvo una calificación menor de 2.66 y se lo seleccionó como un punto para mejorar, para lo cual se añadió más recomendaciones en relación con el tema desarrollado en la versión final de la guía.

Posteriormente, se realizó una calificación media por cada parámetro, tal como se muestra en la Figura 63, obtenido obteniendo así: una calificación promedio de 3.75/4 para el parámetro de Redacción y contenido, una calificación promedio de 3.22/4 para el parámetro de Claridad, y finalmente una calificación de 3.10/4 para el parámetro de Replicabilidad.

Figura 63

Calificación obtenida por cada parámetro de la guía.



Para valorar cada parámetro como satisfactorio se definió que este obtenga una calificación promedio mayor al 75% de la nota máxima, lo que en la escala de 4 puntos correspondería a que cada parámetro obtenga una calificación promedio mayor a 3 puntos.

Tomando en cuenta los criterios presentados anteriormente y presentando las correcciones del ítem individual con calificación menor, se concluye como satisfactorio con una calificación media mayor a 3 puntos para el parámetro de Redacción y contenido, Claridad y Replicabilidad, completando así el proceso de validación mediante Delphi para la guía de implementación.

CAPÍTULO 3

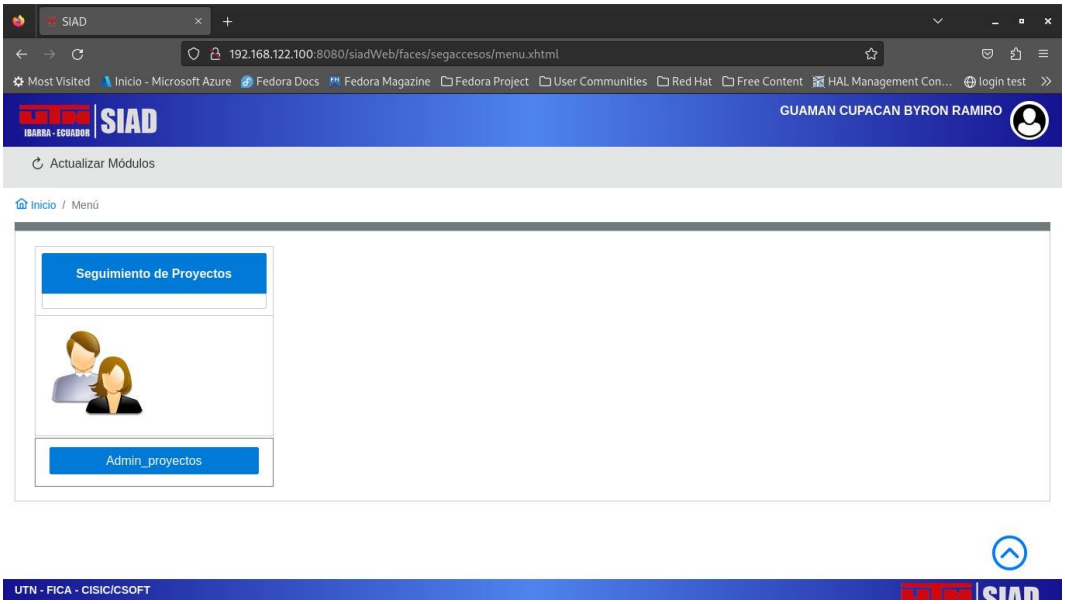
Resultados

3.1 Pruebas

Se realizó el despliegue y configuración de la aplicación Java Enterprise: Sistema Integrado de Actividad Docente (SIAD) con el objetivo de medir el rendimiento de la arquitectura desarrollada mediante una aplicación funcional. Se tomó en cuenta la segunda arquitectura desarrollada en el presente trabajo, compuesta por un balanceador de carga y un grupo de servidores conformado por dos nodos desplegados en el Cloud FICA de la Universidad Técnica del Norte.

Figura 64

Aplicación Java EE SIAD implementada



Se realizó de las respectivas pruebas de carga y estrés mediante el software Apache JMeter a la página de Login de la aplicación SIAD, donde se estableció diferentes números de hilos (usuarios) para así observar las diferentes métricas que son calculadas y presentadas por la aplicación. Los valores que se utilizará para el análisis son los siguientes:

- Tiempo promedio de respuesta: corresponde a la media aritmética que es calculada entre la suma de todos los tiempos de respuesta divididos para la cantidad de peticiones realizadas, donde su valor es representado en milisegundos (ms)

- Porcentaje de error: corresponde al porcentaje de solicitudes que presentaron error o también considerado como el porcentaje de pruebas fallidas.
- Rendimiento: corresponde a la carga del servidor y se calcula como el número de solicitudes sobre una unidad de tiempo, teniendo en cuenta que el tiempo se obtienen desde que inicia la prueba hasta que termina de procesar la última petición. Su valor se expresa en solicitudes/segundo y expresa la cantidad de peticiones totales que fueron procesadas en el periodo de 1 segundo.

Las pruebas se inicialmente se realizaron en la arquitectura que actualmente está en funcionamiento en el Cloud FICA, que se compone de un solo servidor de aplicaciones en su configuración standalone. Los resultados obtenidos de esta primera fase de pruebas se presentan a continuación en la Tabla 14, donde se obtuvo un tiempo promedio por petición de 4 a 5428 ms, un rendimiento de 100.8/seg a 444.4/seg y un porcentaje de error del 0%, indicando así que se han procesado correctamente todas las peticiones.

Tabla 14

Resultados de pruebas de rendimiento a arquitectura Standalone

Arquitectura	Número de hilos (usuarios)	Tiempo promedio de respuesta (ms)	Rendimiento	% de error
Wildfly Standalone (1 nodo)	100	4	100.8/seg	0.00%
	400	5	333.1/seg	0.00%
	800	363	597/seg	0.00%
	1000	1070	444.4/seg	0.00%
	2000	4674	200.1/seg	0.00%
	4000	5428	345.1/seg	0.00%

Posteriormente, se realizó las pruebas a la arquitectura de alta disponibilidad desarrollada. Los resultados obtenidos se presentan a continuación en la Tabla 15 donde se obtuvo un tiempo promedio por petición de 14 a 1844 ms, un rendimiento de 98.5/seg a 415.17/seg y un porcentaje de error del 0%, indicando así que se han procesado correctamente todas las peticiones para esta arquitectura.

Tabla 15

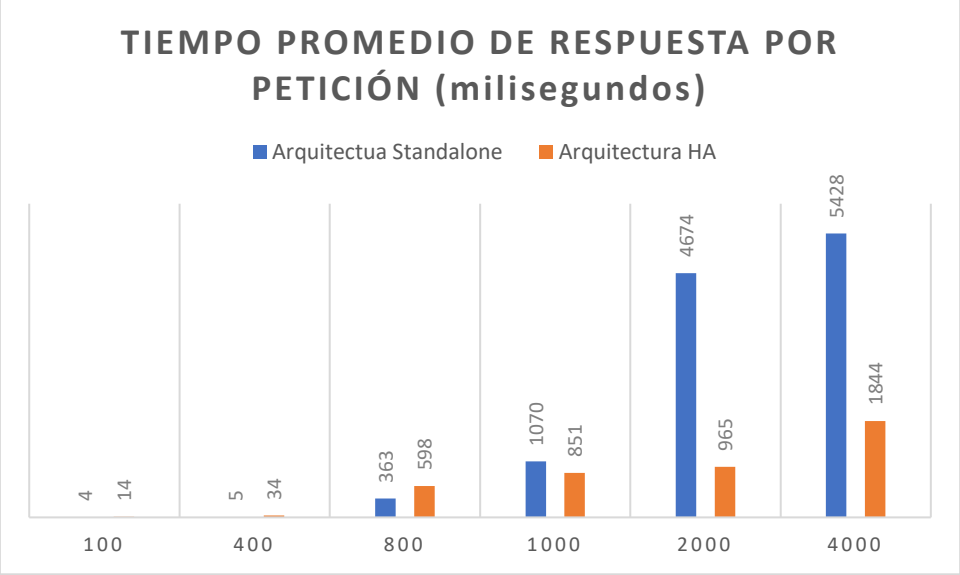
Resultados de pruebas de rendimiento a arquitectura HA

Arquitectura	Número de hilos (usuarios)	Tiempo promedio de respuesta (ms)	Rendimiento	% de error
Balanceador de carga + Wildfly Domain (2 nodos)	100	14	98.5/seg	0.00%
	400	34	195.1/seg	0.00%
	800	598	323.4/seg	0.00%
	1000	851	292/seg	0.00%
	2000	965	389.3/seg	0.00%
	4000	1844	415.7/seg	0.00%

3.2 Análisis e interpretación de resultados

Figura 65

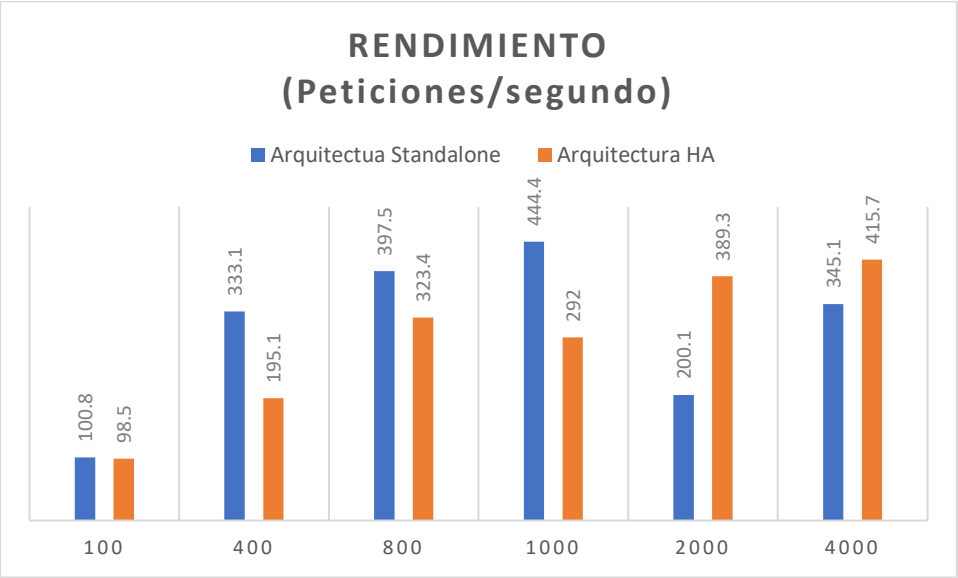
Tiempo promedio de respuesta por petición



En la Figura 65 se observa que la arquitectura de alta disponibilidad presenta un tiempo promedio de respuesta por petición significativamente menor, ya que esta arquitectura está compuesta por dos nodos a diferencia de la arquitectura standalone, que permite la distribución de las cargas de trabajo entre los diferentes servidores de aplicaciones y así obtener un procesamiento más efectivo, evitando problemas de saturación ante un gran volumen de peticiones.

Figura 66

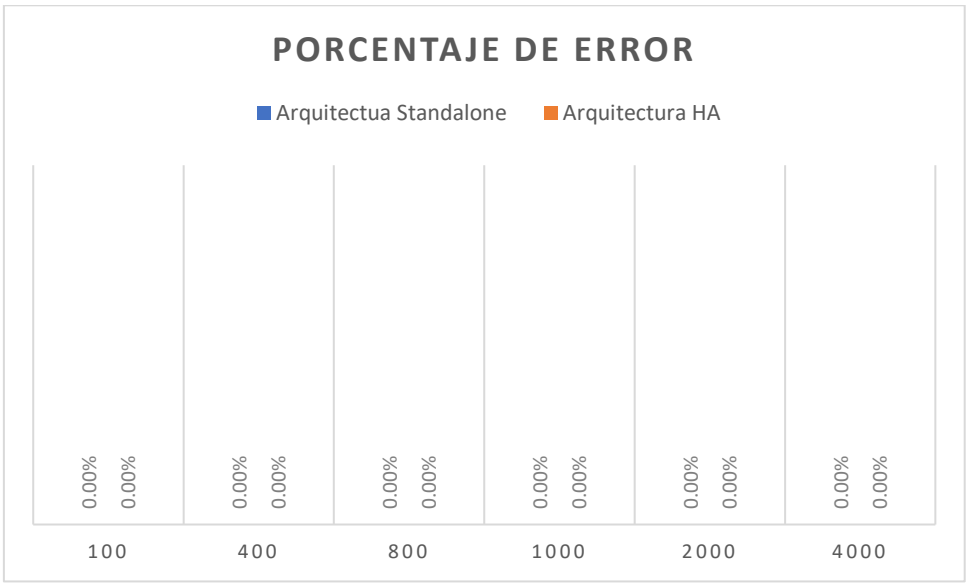
Comparación de rendimiento entre arquitecturas



En la Figura 66 se observa que la arquitectura standalone presenta un nivel de rendimiento mayor cuando se trabaja hasta un número de 1000 usuarios simultáneos, sin embargo, ante una mayor cantidad de hilos (2000 y 4000), la arquitectura de alta disponibilidad ofrece un nivel de rendimiento mayor, lo cual indica un comportamiento ideal al tratarse de una arquitectura que permite tiene como objetivo proporcionar un funcionamiento continuo y eficiente ante un gran número de solicitudes.

Figura 67

Comparación del porcentaje de error entre arquitecturas.

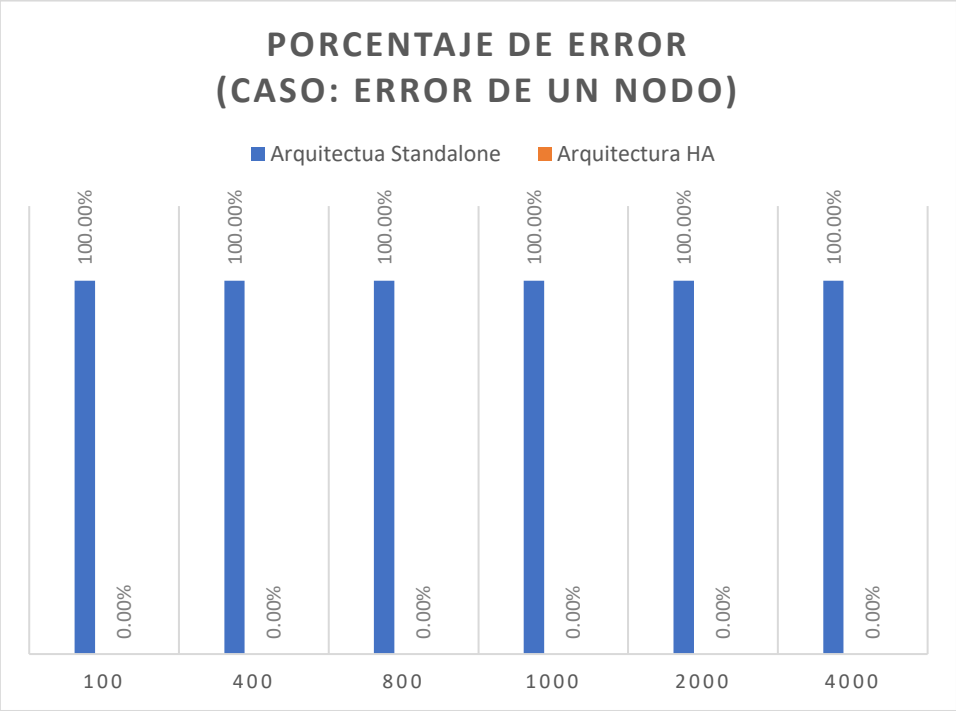


En la Figura 67 se presenta que las dos arquitecturas presentan un nulo porcentaje de error durante la ejecución de las diferentes pruebas, indicando que se han procesado correctamente todas las peticiones en los diferentes escenarios de pruebas. La arquitectura Standalone y la arquitectura de alta disponibilidad se muestran estables incluso ante una gran cantidad de usuarios simultáneos.

Se hace énfasis que en la arquitectura de alta disponibilidad tiene la ventaja de ofrecer un funcionamiento continuo y un porcentaje de error del 0% incluso ante la presencia de la presencia de un error en un nodo que conforma el grupo de servidores, tal como se muestra en la Figura 68, donde se ha desactivado el nodo Maestro y se ha replicado las pruebas, mientras en la arquitectura standalone no es posible replicar el fallo de un nodo ya que en esta configuración no se cuenta con un nodo de respaldo que permita el funcionamiento continuo del servidor de aplicaciones.

Figura 68

Comparación del porcentaje de error entre arquitectura ante la presencia de un fallo.



3.4 Validación de resultados

Para la validación de resultados obtenidos de las pruebas para la arquitectura de alta disponibilidad, se tomó como referencia las métricas de: Tiempo de servicio y Tiempo medio de inactividad, definidas en la norma ISO/IEC 25023.

Para la obtención del Tiempo de servicio, se estableció como referencia a una semana de ejecución continua para el tiempo de servicio regulado en el cronograma operacional, lo cual corresponde a un tiempo total de 10080 minutos, y así poner contrastar con el tiempo de servicio que el sistema proporciona actualmente donde se toma como referencia el tiempo real que el servidor de aplicaciones brindó durante una semana de funcionamiento continuo.

Para la obtención del Tiempo medio de inactividad, se tomó en cuenta el valor correspondiente al porcentaje de error obtenidos en las pruebas de rendimiento para un número de 4000 hilos de la Figura 68, donde adicionalmente se simula el caso de fallo de un nodo del servidor de aplicaciones para cada una de las arquitecturas a evaluar.

Para la evaluación cuantitativa de la subcaracterística de disponibilidad se estableció una calificación sobre 10 puntos, la cual será calculada mediante una matriz que está estructurada por: la fórmula de la métrica a calificar, el valor deseado, el valor obtenido, la ponderación, la valoración parcial por métrica y el valor final de la subcaracterística.

Para el cálculo de la métrica de Tiempo de servicio se tiene la fórmula: X=A/B, donde “A” corresponde al Tiempo de servicio que se proporciona actualmente y “B” al Tiempo de servicio del sistema regulado en el cronograma operacional , y para el cálculo de la métrica de Tiempo medio de inactividad se tiene la fórmula: X=A/T, donde “A” corresponde al Número de fallos observados y “B” al Tiempo total de inactividad.

En la Tabla 16 se presenta los resultados de la evaluación para la arquitectura standalone , y en la Tabla 17 los resultados obtenidos de la evaluación para la arquitectura de alta disponibilidad desarrollada en el presente trabajo.

Tabla 16

Validación de resultados de arquitectura standalone.

Sub-característica	Métrica	Fórmula	Valor deseado	Valor obtenido	Ponderación (/10)	Valor final
Disponibilidad	Tiempo de servicio	X=A/B	Deseado: 1	A=10080/min B=10080/min **X=1**	**10**	**5.5**
	Tiempo medio de inactividad	X=A/T.	Deseado: 0/min Peor caso: >=10/min	A=4000 T=4.5 min **X=888.8/min**	**1**	

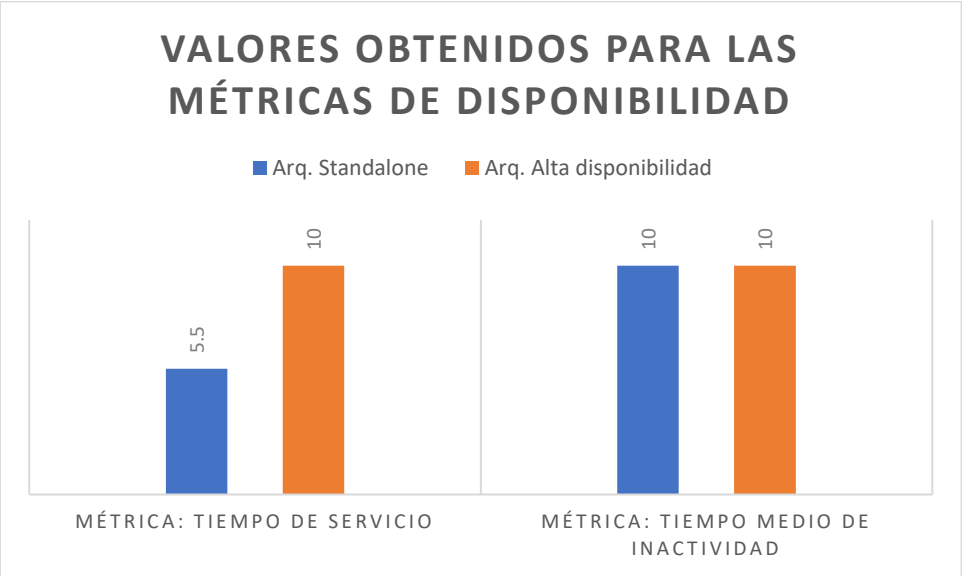
Tabla 17

Validación de resultados de arquitectura de alta disponibilidad

Sub-característica	Métrica	Fórmula	Valor deseado	Valor obtenido	Ponderación (/10)	Valor final
Disponibilidad	Tiempo de servicio	X=A/B	Deseado: 1	A=10080/min B=10080/min **X=1**	**10**	**10**
	Tiempo medio de inactividad	X=A/T	Deseado: 0/min Peor caso: >=10/min	A=0 T=0 min **X=0/min**	**10**	

Figura 69

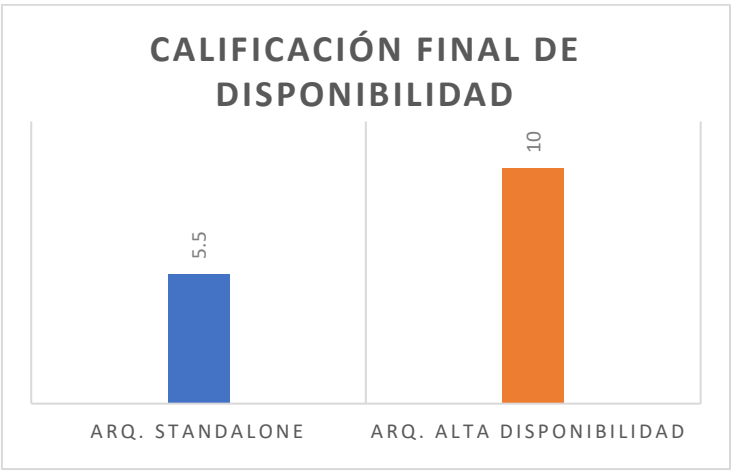
Valores obtenidos para las métricas de disponibilidad



Como se observa en la Figura 69, se observa un resumen de los diferentes valores obtenidos luego de su evaluación, donde se obtuvo para la arquitectura standalone un valor parcial de 10/10 para Tiempo de servicio y un valor de parcial de 1/10 para el Tiempo medio de inactividad, y para la arquitectura de alta disponibilidad un valor parcial de 10/10 para Tiempo de servicio y un valor de parcial de 10/10 para el Tiempo medio de inactividad.

Figura 70

Calificación de cada arquitectura para subcaracterística de disponibilidad



Como se observa en la Figura 70, se obtuvo una calificación final para la subcaracterística de disponibilidad de 5.5/10 puntos para la arquitectura standalone y de 10/10 puntos para la arquitectura de alta disponibilidad.

Finalmente, para definir el grado de satisfacción obtenido para la subcaracterística de disponibilidad se utilizó los niveles de puntuación y de grados de satisfacción definidas en la norma ISO/IEC 25040, presentada en la Tabla 18 .

Tabla 18

Escala de medición de satisfacción de la norma ISO/IEC 25040.

Escala de medición	Niveles de puntuación	Grado de satisfacción
8.00 – 10.00	Cumple los requisitos	Muy satisfactorio
5.00 – 7.95	Aceptable	Satisfactorio
1.00 – 4.95	Inaceptable	Insatisfactorio

En relación del valor total de la subcaracterística de disponibilidad presentada en la Figura 70 para cada arquitectura y considerando la escala de medición presentada en la Tabla 18, se considera como un nivel de puntuación: “Aceptable” y un grado de satisfacción: “Satisfactorio” para la arquitectura standalone , y un nivel de puntuación: “Cumple los requisitos” y un grado de satisfacción: “Muy satisfactorio” para la arquitectura de alta disponibilidad.

Observando los valores presentados en la Figura 70 y los grados de satisfacción definidos anteriormente para cada arquitectura, se concluye que la arquitectura desarrollada en el presente trabajo cumple satisfactoriamente con los requisitos de disponibilidad propuestos, además de confirmar que se ha fortalecido la alta disponibilidad con respecto a la arquitectura standalone que estaba funcionando actualmente.

CONCLUSIONES

La revisión de la literatura permitió tener una conceptualización de la alta disponibilidad para el servidor de aplicaciones Wildfly, conocer acerca de la subcaracterística de disponibilidad de la norma ISO/IEC 25010 y sus métricas, y se recopiló de información de trabajos relacionados que facilitó la creación de una nueva arquitectura basada en componentes de software libre.

La implementación de la técnica de alta disponibilidad permitió que el Cloud FICA obtenga las características de balanceo de carga y de conmutación por error para el servidor de aplicaciones Wildfly, fortaleciendo así la continuidad de servicio de las aplicaciones Java Enterprise desplegadas.

La creación de la Guía de Implementación de Alta Disponibilidad de Wildfly permitió construir una documentación con pasos y configuraciones claras y replicables, conformando una base importante para futuros proyectos académicos y de investigación acerca arquitecturas de alta disponibilidad, adicionalmente se validó mediante la metodología Delphi, obteniendo una calificación satisfactoria por parte del grupo de expertos bajo los parámetros de Redacción y contenido, Claridad y de Replicabilidad.

Se desplegó la aplicación Java Enterprise: Sistema Integrado de Actividad Docente en la arquitectura diseñada, y mediante las pruebas de carga y estrés se obtuvo resultados favorables a diferencia de la arquitectura standalone que se contaba anteriormente, obteniendo valores que demuestran que la arquitectura de alta disponibilidad tiene un menor tiempo promedio de respuesta, un mayor porcentaje de rendimiento y un nulo porcentaje de error, incluso ante grandes volúmenes de peticiones y ante la presencia de error de un nodo.

Se realizó la validación de resultados en base de la norma ISO/IEC 25010 para la arquitectura de alta disponibilidad y se obtuvo una calificación de 10/10 para la métrica de Tiempo de servicio y de 10/10 para Tiempo medio de inactividad, que en conjunto representa una puntación de 10/10 para la subcaracterística de disponibilidad, calificándola finalmente como: “Muy satisfactoria” en la escala de medición de satisfacción de la norma ISO/IEC 25040.

RECOMENDACIONES

Se recomienda usar las mismas versiones de Wildfly para la configuración del balanceador de carga y el grupo de servidor de aplicaciones, como también la actualización a sus últimas versiones para así tener un mayor nivel de compatibilidad y funcionalidad de la técnica de alta disponibilidad implementada.

Se recomienda seguir configuraciones que estén recomendadas dentro de la sección de documentación oficial de Wildfly de acuerdo a la versión que se esté utilizando, con la finalidad de evitar configuraciones obsoletas e incompatibles.

Se recomienda la utilización de la tecnología Modcluster mediante la configuración de Wildfly como balanceador de carga, ya que en este caso se cuenta con la consola de administración web que permite una mejor visualización de cada nodo que compone la arquitectura y detalles de la carga que recibe cada nodo del grupo de servidores de aplicaciones.

Para trabajos futuros se recomienda la integración de otras técnicas de alta disponibilidad dentro de la infraestructura (alta disponibilidad de bases de datos, redundancia de hardware, entre otros), para así garantizar tener todo un entorno de alta disponibilidad que permita ofrecer un funcionamiento continuo ante la presencia de errores en otras secciones de la infraestructura.

BIBLIOGRAFÍA

Apache Software Foundation. (2022). *Apache JMeter - Apache JMeter™*. https://jmeter.apache.org/

Balseca, E. (2014). *Evaluación de calidad de productos de software en empresas de desarrollo de software aplicando la norma ISO/IEC 25000*. http://bibdigital.epn.edu.ec/handle/15000/9113

Bastidas, J. (2021). *Análisis, diseño e implementación de balanceador de carga segmentando la red del sistema de balanzas de pequeñas y medianas empresas del sector industrial ciudad de Guayaquil.* http://repositorio.ug.edu.ec/handle/redug/52246

Bustos, G. (2023). *¿Qué es Apache? Una guía detallada*. https://www.hostinger.es/tutoriales/que-es-apache/

Calabrese, J. (2018). *ASISTENTE PARA LA EVALUACIÓN DE CALIDAD DE PRODUCTO DE SOFTWARE SEGÚN LA FAMILIA DE NORMAS ISO/IEC 25000 UTILIZANDO EL ENFOQUE GQM*. https://core.ac.uk/download/pdf/301082286.pdf

Cali, F. (2011). *Implementación de un cluster a nivel de aplicación en la empresa Casa Moeller Martínez C.A. utilizando una plataforma y herramientas Open Source* [Escuela Politécnica Nacional]. https://bibdigital.epn.edu.ec/handle/15000/4086

Castellano, L. (2019). *Kanban. Metodología para aumentar la eficiencia de los procesos. 3C Tecnología. 8*(1), 30–41. https://doi.org/10.17993/3ctecno/2019

Centos. (2023). *The CentOS Project*. https://www.centos.org/

Cuesta, J. (2013). *Aplicación de la técnica Delphi en el proceso de validación de un instrumento para la evaluación de la calidad de vida en centros para personas con Trastornos del Espectro del Autismo.* https://riull.ull.es/xmlui/handle/915/4443

George, C., & Liñan, L. (2018). Aplicación del Método Delphi Modificado para la Validación de un Cuestionario de Incorporación de las TIC en la Práctica Docente. *Revista Iberoamericana de Evaluación Educativa*, *11*(1), 113–135. https://doi.org/10.15366/RIEE2018.11.1.007

Gómez, A., & León, E. (2010). *Diseño de alta disponibilidad en servidores Solaris 10 para aplicación en el Centro de Cómputo de la U.C.S.G.* http://repositorio.ucsg.edu.ec/handle/3317/1222

Google Cloud. (2022). *Establece el 99.9% de disponibilidad para la interconexión dedicada | Cloud Interconnect | Google Cloud*. https://cloud.google.com/network-connectivity/docs/interconnect/tutorials/dedicated-creating-999-availability?hl=es-419

Gutiérrez, A., & Benítez, Y. (2013). *Clúster de alta disponibilidad y balanceo de carga sobre un servidor web*. https://ddd.uab.cat/record/114971

Guzmán, J., Sánchez, J., & Gómez, J. (2021). ALGORITMOS DE CALIDAD DE SERVICIO PARA BALANCEAR CARGAS EN REDES DEFINIDAS POR SOFTWARE. *REVISTA COLOMBIANA DE TECNOLOGIAS DE AVANZADA (RCTA)*, *1*(37). https://ojs.unipamplona.edu.co/ojsviceinves/index.php/rcta/article/view/1123

Hurtado, C. (2011). *Propuesta de un servidor Web Apache 2 sobre un Cluster en Linux*. http://dspace.uclv.edu.cu:8089/handle/123456789/4676

IBM. (2021a). *Servidores de aplicaciones - Documentación de IBM*. https://www.ibm.com/docs/es/i/7.2?topic=serving-application-servers

IBM. (2021b). *Ventajas de sistemas de alta disponibilidad - Documentación de IBM*. https://www.ibm.com/docs/es/mfci/7.6.2?topic=systems-advantages-highly-available

ISO25000. (2022). *ISO/IEC 25010*. https://iso25000.com/index.php/normas-iso-25000/iso-25010

Kone, D. (2021). *High Availability Systems*. https://helda.helsinki.fi/handle/10138/337743

KVM. (2016). *Kernel Virtual Machine*. https://www.linux-kvm.org/index.php?title=Main_Page&oldid=173792

Marchioni, F. (2021). *Configuración de WildFly como equilibrador de carga para un clúster - Mastertheboss*. https://www.mastertheboss.com/jbossas/jboss-cluster/configuring-wildfly-as-load-balancer-for-a-cluster/

Martínez, M., Sánchez, B., & Camacho, A. (2019). Método Delphi: validar un instrumento para la medición de características de un libro de texto de probabilidad y estadística. *Revista de Educación Técnica*, *3*, 8–18. https://doi.org/10.35429/JOTE.2019.7.3.8.18

Mesh, J. (2020). *Metodología Kanban: revoluciona tu manera de trabajar más ágil*. https://blog.trello.com/es/metodologia-kanban

Microsoft. (2022). *Métricas empresariales - Microsoft Azure Well-Architected Framework*. https://learn.microsoft.com/es-es/azure/architecture/framework/resiliency/business-metrics

Microsoft. (2023). *Microsoft Planner: Qué es y los más recientes updates | Microsoft Latinx*. https://blogs.microsoft.com/latinx/2018/03/16/microsoft-planner-que-es-y-los-mas-recientes-updates/

Microsoft 365 Team. (2021). *Kanban Solution | Microsoft 365*. https://www.microsoft.com/es-ww/microsoft-365/business-insights-ideas/resources/how-to-use-a-kanban-solution-to-manage-your-team-tasks

Modcluster. (2023). *Modcluster Documentation*. https://docs.modcluster.io/

Morillo, P. (2021). *Desarrollo del módulo de plantillas para el sistema integrado de actividad docente (SIAD) de la carrera de software de la Universidad Técnica del Norte, aplicando el estándar ISO/IEC 25010 para mantenibilidad.* http://repositorio.utn.edu.ec/handle/123456789/11745

Naciones Unidas. (2022). *Portada - Desarrollo Sostenible*. https://www.un.org/sustainabledevelopment/es/

Niño, D. (2020). *DISEÑO DE UN MODELO DE VIRTUALIZACIÓN PARA LA IMPLEMENTACIÓN DE UN SISTEMA DE SERVIDORES EN ALTA DISPONIBILIDAD* . http://hdl.handle.net/20.500.12494/17050

Oracle. (2010). *Modelo de referencia OSI*. https://docs.oracle.com/cd/E19957-01/820-2981/ipov-8/index.html

Parra, V. (2016). *IMPLEMENTACIÓN DE SERVIDOR DE APLICACIONES JBOSS MODO DOMAIN PARA EL MINISTERIO DE EDUCACIÓN DEL ECUADOR .* http://www.dspace.uce.edu.ec/handle/25000/6869/statistics

Pazmiño, A. (2014). *Guía metodológica para la implementación de un clúster de alta disponibilidad para servidores de aplicaciones, utilizando Open Source. Caso de estudio: empresa soporte libre.* http://repositorio.puce.edu.ec/handle/22000/9583

Perafan, H., Nazareth, G., Rey, D., & Duarte, D. (2018). Design of a high availability cluster for a university virtual educational environment. *Revista INGENIERÍA UC*, *25*(1). https://www.researchgate.net/publication/326235419_Design_of_a_high_availability_cluster_for_a_university_virtual_educational_environment

PROXMOX. (2023). *Proxmox: potentes soluciones de servidor de código abierto*. https://www.proxmox.com/en/

Red Hat. (2022a). *¿Qué es KVM?* https://www.redhat.com/es/topics/virtualization/what-is-KVM

Red Hat. (2022b). *¿Qué es la alta disponibilidad?* https://www.redhat.com/es/topics/linux/what-is-high-availability

Reina, E., & Patiño, S. (2019). Evaluación de la calidad en uso de un sistema web/ móvil de control de asistencia a clases de docentes y estudiantes aplicando la norma ISO/IEC 25000 SQuaRe. *RISTI - Revista Iberica de Sistemas e Tecnologias de Informacao*, *E19*, 108. https://www.researchgate.net/publication/335754151_Evaluacion_de_la_calidad_en_uso_de_un_sistema_web_movil_de_control_de_asistencia_a_clases_de_docentes_y_estudiantes_aplicando_la_norma_ISOIEC_25000_SQuaRe/citations

Roa, P. A., Morales, C., & Gutiérrez, P. (2015). Norma ISO/IEC 25000. *Tecnología Investigación y Academia*, *3*(2), 27–33. https://geox.udistrital.edu.co/index.php/tia/article/view/8373

Saau, J., & Germán, R. (2013). *Herramienta para la administración de clúster en servidores de aplicaciones JBoss.* https://repositorio.uci.cu/jspui/handle/ident/8516

Sánchez, J. (2011). *Implantación de Aplicaciones Web*. https://jorgesanchez.net/

Santos, J. (2014). *Seguridad y Alta Disponibilidad (Grado Superior)* (RA-MA, Ed.).

Sinisterra, M. M., Henao, T. M. D., & López, E. G. R. (2012). Clúster de balanceo de carga y alta disponibilidad para servicios web y mail. *Informador Técnico*, *76*, 93–93. https://doi.org/10.23850/22565035.34

SITMÉXICO. (2016). *BALANCEADORES DE CARGA*. https://sitmexico.com/balanceadores-de-carga.php

Vera, R. (2018). *Diseño e implementación de un clúster usando JBoss EAP para aumentar la disponibilidad de los servidores de aplicaciones en una entidad del Estado* . https://hdl.handle.net/20.500.12672/10074

Vizcaíno, Y. (2021). *Optimización de aplicaciones Java Enterprise monolíticas mediante el uso de contenedores Docker.* http://repositorio.utn.edu.ec/handle/123456789/10981

Wildfly. (2018). *High Availability Guide*. http://docs.wildfly.org/12/High_Availability_Guide.html

Wildfly. (2023). *WildFly - Homepage*. https://www.wildfly.org/

Zamora, J. (2015). *Servidores web y PaaS*. http://expertojava.ua.es/experto/restringido/2015-16/paas/paas01.html

Apéndice A. Tareas y subtareas de Tablero Kanban

Definición de tareas y subtareas contenidas en el tablero Kanban del proyecto

Tarea	Subtarea	Fecha de inicio	Fecha de vencimiento	Prioridad
Definir las arquitecturas del proyecto	Diseñar la segunda arquitectura del proyecto. Diseñar la primera arquitectura del proyecto. Diseñar la tercera arquitectura del proyecto.	10/04/2023	12/04/2023	Importante
Definir los requisitos de las máquinas virtuales	Definir los nombres de las máquinas virtuales. Definir las características de hardware y software de las máquinas virtuales. Definir las configuraciones de red de las máquinas virtuales.	13/04/2023	14/04/2023	Importante
Crear las máquinas virtuales de la primera arquitectura	Creación de máquina virtual para servidor de balanceo de carga. Creación de máquina virtual para servidor de aplicaciones maestro. Creación de máquina virtual para servidor de aplicaciones esclavo.	17/04/2023	18/04/2023	Importante
Instalar el sistema operativo en las máquinas virtuales de la primera arquitectura	Completar instalación del sistema operativo con los parámetros definidos para cada máquina virtual. Realizar las configuraciones de red asignadas para cada máquina virtual.	18/04/2023	18/04/2023	Importante
Configuración de Wildfly para	Descargar Wildfly en las máquinas virtuales.	19/04/2023	21/04/2023	Importante

las máquinas virtuales de la primera arquitectura	Crear usuarios de Wildfly en las máquinas virtuales. Habilitar puertos de comunicación TCP 8080 y 9990. Instalar Java 17 en las máquinas virtuales.			
Configurar el grupo de servidores de aplicaciones para la primera arquitectura	Configurar Host Maestro. Configurar Host Esclavo. Ejecución de servidores. Crear grupo de servidores. Agregar servidor: Nodo Maestro. Agregar servidor: Nodo Esclavo.	24/04/2023	25/04/2023	Importante
Configurar el servidor de balanceo de carga para la primera arquitectura	Configurar el servicio de Modcluster para Wildfly. Añadir puertos TCP y UDP.	26/04/2023	26/04/2023	Importante
Pruebas de funcionamiento de la primera arquitectura	Despliegue de aplicación demo. Realizar prueba de carga en Apache JMeter. Realizar prueba de alta disponibilidad desactivando nodo maestro. Realizar prueba de alta disponibilidad desactivando nodo esclavo.	27/04/2023	28/04/2023	Importante
Crear las máquinas virtuales de la segunda arquitectura	Creación de máquina virtual para servidor de balanceo de carga.	01/05/2023	12/05/2023	Importante

	Creación de máquina virtual para servidor de aplicaciones maestro. Creación de máquina virtual para servidor de aplicaciones esclavo.			
Instalar el sistema operativo en las máquinas virtuales de la segunda arquitectura	Completar instalación del sistema operativo con los parámetros definidos para cada máquina virtual. Realizar las configuraciones de red asignadas para cada máquina virtual.	12/05/2023	12/05/2023	Importante
Configuración de Wildfly para las máquinas virtuales de la segunda arquitectura	Descargar Wildfly en las máquinas virtuales. Crear usuarios de Wildfly en las máquinas virtuales. Habilitar puertos de comunicación TCP 8080 y 9990. Instalar Java 17 en las máquinas virtuales.	15/05/2023	17/05/2023	Importante
Configurar el grupo de servidores de aplicaciones para la segunda arquitectura	Configurar Host Maestro. Configurar Host Esclavo. Crear grupo de servidores. Agregar servidor: Nodo Maestro. Agregar servidor: Nodo Esclavo.	18/05/2023	19/05/2023	Importante
Configurar el servidor de balanceo de carga para la segunda arquitectura	Configurar el servicio de Modcluster para Wildfly. Añadir puertos TCP y UDP.	22/05/2023	22/05/2023	Importante

Pruebas de funcionamiento de la segunda arquitectura	Iniciar servidores. Despliegue de aplicación demo. Realizar prueba de carga en Apache JMeter. Realizar prueba de alta disponibilidad desactivando nodo maestro. Realizar prueba de alta disponibilidad desactivando nodo esclavo.	23/05/2023	26/05/2023	Importante
Configuración del Host Remoto	Realizar las configuraciones de red asignadas para el Host Remoto	29/05/2023	29/05/2023	Importante
Configuración de Wildfly para la máquina virtual de la tercera arquitectura	Descargar Wildfly en la máquina virtual. Habilitar puertos de comunicación TCP 8080 y 9990. Instalar Java 17 en la máquina virtual.	30/05/2023	30/05/2023	Importante
Configurar el grupo de servidores de aplicaciones para la tercera arquitectura	Configurar Host Esclavo Agregar servidor: Nodo Esclavo	31/05/2023	31/05/2023	Importante
Pruebas de funcionamiento de la tercera arquitectura	Iniciar servidores. Despliegue de aplicación demo. Realizar prueba de carga en Apache JMeter. Realizar prueba de alta disponibilidad desactivando nodo maestro.	31/05/2023	02/06/2023	Importante

	Realizar prueba de alta disponibilidad desactivando nodo esclavo.

Apéndice B. Guía de Implementación de Alta Disponibilidad de Wildfly.

UNIVERSIDAD TÉCNICA DEL NORTE

Facultad de Ingeniería en Ciencias Aplicadas

Guía de Implementación de Alta Disponibilidad de Wildfly para el Cloud FICA de la Universidad Técnica del Norte.

INF-IMPL-CLOUDUTN-2023

Versión: 1.0

Elaborado por:

Byron Guamán

Ibarra – Ecuador

2023

Elaborado por: Byron Guamán Cargo: Tesista Institución: Universidad Técnica del Norte	Firma: 
Revisada por: Mauricio Rea P. Cargo: DOCENTE FICA Institución: Universidad Técnica del Norte	Firma:  Firmado electrónicamente por: XAVIER MAURICIO REA PEÑAFIEL

Tabla de contenidos

Índice de figuras

Índice de tablas

1. Introducción

Actualmente la Facultad de Ingeniería y Ciencias Aplicadas (FICA) de la Universidad Técnica del Norte (UTN) tiene en funcionamiento una plataforma de virtualización Proxmox y está destinada a la creación de máquinas virtuales para el levantamiento de diferentes servicios. En una instancia virtual se tiene en funcionamiento el servidor de aplicaciones Wildfly para el despliegue de aplicaciones Java Enterprise desarrolladas por los estudiantes de la institución con fines académicos. En un proceso de observación se detectó que el servidor de aplicaciones no cuenta con una configuración que garantice la alta disponibilidad de las aplicaciones desplegadas, así como también se evidenció problemas de funcionalidad y pérdidas de disponibilidad por tiempos prolongados.

Como solución a la problemática presentada, se realizó la implementación de una solución de la alta disponibilidad para el servidor de aplicaciones, permitiendo obtener una funcionalidad continua e ininterrumpida, asegurando que el servicio funcione 24/7.

Wildfly permitió la configuración de una instancia del servidor de aplicaciones para que actúe como balanceador de carga y así integrar una arquitectura que contenga características de alta disponibilidad, como la distribución de peticiones entre los diferentes nodos que conformen el grupo de servidores para evitar problemas de saturación ante presencia de grandes volúmenes de solicitudes, y el traslado de la sesión de un nodo a otro en caso de que presentarse algún error.

El balanceador de carga trabaja mediante el subsistema Modcluster que utiliza la conexión de cada nodo para la transmisión de factores de equilibrio de carga al lado del servidor mediante un conjunto de métodos HTTP denominados como Mod-Cluster Management Protocol. Adicionalmente, Wildfly cuenta con una configuración denominada: “Dominio”, que permite obtener una gestión centralizada de cada nodo que conforma el grupo de servidores, y que en conjunto con el balanceador de carga permitió conformar una arquitectura de alta disponibilidad.

2. Objetivos

2.1 Objetivo General

- Implementar una solución de alta disponibilidad para el servidor de aplicaciones Wildfly en la plataforma virtualizada de la FICA de la Universidad Técnica del Norte.

2.3 Objetivos Específicos

- Definir la arquitectura de alta disponibilidad
- Configurar el grupo de servidores de aplicaciones
- Configurar e integrar el balanceador de carga
- Realizar pruebas de carga y disponibilidad

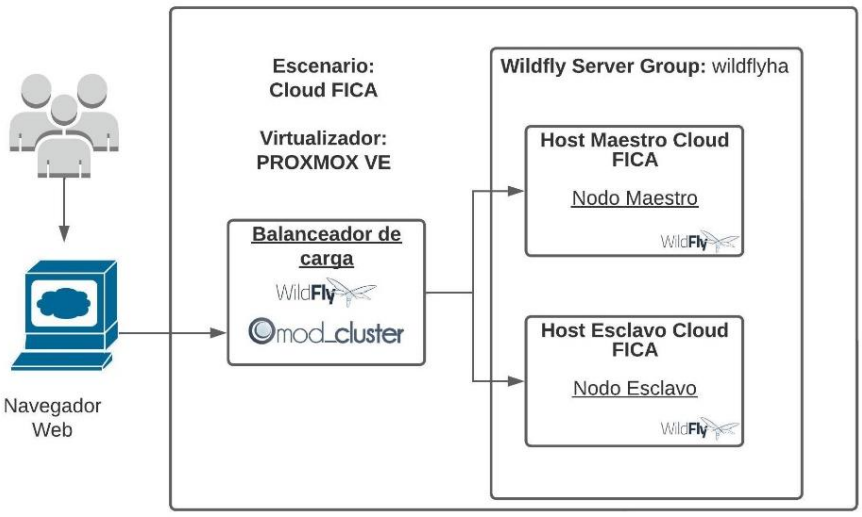
3. Desarrollo

3.1 Definición de arquitectura

Como se presenta en la Figura 1, la arquitectura de alta disponibilidad implementada está compuesta por un balanceador de carga que actuará como front-end para la recepción de las peticiones de los usuarios, las cuales serán redireccionadas a los diferentes nodos del grupo de servidores. El grupo de servidores está configurado bajo el modo de trabajo: “Dominio” y está compuesto por un host “Maestro” y “Esclavo”, en los cuales se configuró los diferentes nodos que están encargados de actuar como back-end para realizar el despliegue y presentación de las aplicaciones Java Enterprise al usuario.

Figura 1

Arquitectura presentada para la guía



3.2 Definición de máquinas virtuales

Para el cumplimiento de la implementación de la solución de alta disponibilidad, se creó mediante el software de virtualización Proxmox VE tres máquinas virtuales con el sistema operativo CentOS Stream 8. En la Tabla 1 se detalla las características de hardware y la descripción de cada máquina virtual utilizada.

Tabla 1

Características generales de máquinas virtuales

Máquina virtual	Hardware	Descripción
mv-cloud-srvbc	CPU: 2 RAM: 4GB DISCO: 20GB	Nodo Balanceador de Carga en Cloud Fica
mv-cloud-srvapp1	CPU: 2 RAM: 4GB DISCO: 30GB	Nodo Wildfly Maestro en Cloud Fica
mv-cloud-srvapp2	CPU: 2 RAM: 4GB DISCO: 30GB	Nodo Wildfly Esclavo en Cloud Fica

En la Tabla 2 se presenta las diferentes configuraciones de red utilizadas para cada máquina virtual.

Tabla 2

Configuraciones de red de las máquinas virtuales

Máquina virtual	Hostname	Dirección IP
mv-cloud-srvbc	srvbc.balanceador.cloud	10.24.8.87/24
mv-cloud-srvapp1	srvapp1.maestro.cloud	10.24.8.88/24
mv-cloud-srvapp2	srvapp2.esclavo.cloud	10.24.8.89/24

3.3 Instalación de Wildfly.

En las tres máquinas virtuales se realizó la descarga de Wildfly para su posterior configuración dependiendo de su función asignada. La instalación base de Wildfly se detalla a continuación.

Como requisito para el funcionamiento del servidor de aplicaciones Wildfly, se realizó la instalación de Java en su versión 17 mediante la ejecución del siguiente comando en la terminal:

- sudo dnf install -y java-17-openjdk-devel

A continuación, se realizó la descarga de los ficheros correspondientes al servidor de aplicaciones Wildfly en su versión 26.1.3 desde su repositorio oficial mediante el siguiente comando:

- wget https://github.com/wildfly/wildfly/releases/download/26.1.3.Final/wildfly.26.1.3.Final.tar.gz -P /tmp

Los ficheros comprimidos se descargaron en la ruta: “/tmp”, y posteriormente se realizó la descomprensión en el directorio: “/opt”, mediante el siguiente comando:

- sudo tar xf /tmp/wildfly-26.1.3.Final.tar.gz -C /opt/

Los ficheros descomprimidos se ubicaron en la ruta “/opt”, y posteriormente se creó un enlace simbólico al directorio de instalación de Wildfly:

- sudo ln -s /opt/wildfly-26.1.3.Final opt/wildfly

Se creó un nuevo usuario y un grupo de sistema denominado:” wildfly”. Además, se definió el directorio de inicio de sesión del nuevo usuario.

- sudo groupadd -r wildfly
- sudo useradd -r -g wildfly -d /opt/wildfly

Se cambió la propiedad del directorio de la instalación para la ejecución bajo el usuario y grupo creado:

- sudo chown -RH wildfly: /opt/wildfly

Con la finalidad de acceder a la consola de administración web de Wildfly y a la herramienta de línea de comandos (CLI) se creó los diferentes usuarios de Wildfly presentados a continuación:

- adminBalanceador: Usuario de Wildfly encargado de la administración del balanceador de carga.
- adminMaestro: Usuario de Wildfly encargado de la administración del grupo de servidores.
- nodoEsclavo: Usuario de Wildfly utilizado para realizar la conexión de los nodos esclavos.

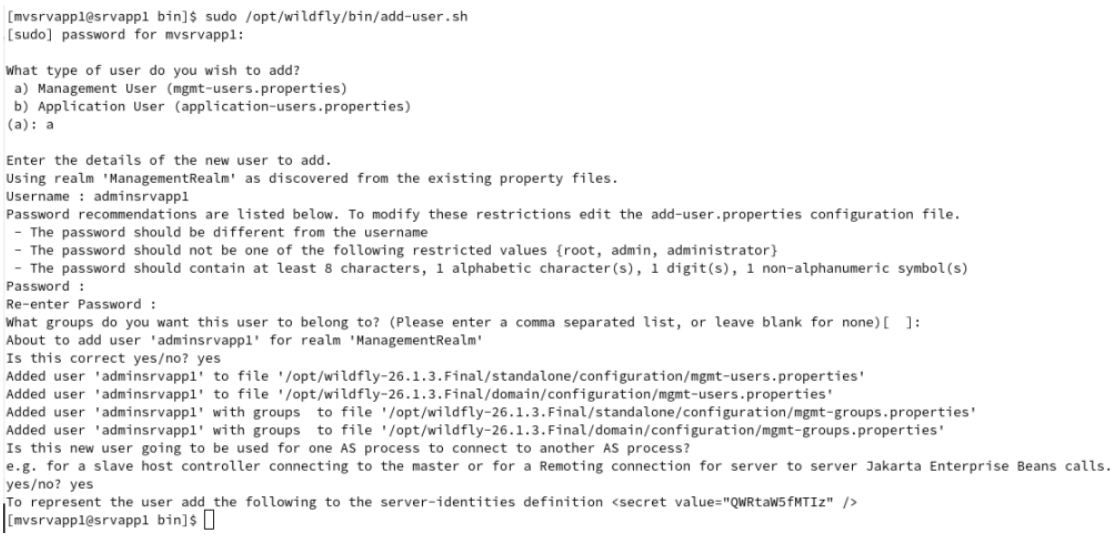
Para la creación de los usuarios se accedió al directorio que contiene el archivo que ejecuta el script para la creación del usuario. Posteriormente se ejecutó el script de creación mediante los siguientes comandos:

- cd /opt/wildfly/bin
- sudo ./add-user.sh

Como se observa en la Figura 2, se escogió la opción: “a) Management User(mgmt-users.properties)” y se ingresó los parámetros de nombre de usuario y contraseña. Adicionalmente se confirmó que el nuevo usuario se añada al “ManageRealm” y se confirmó que el usuario se habilite para la conexión con otras instancias de Wildfly.

Figura 2

Ejecución de script de creación de usuarios de Wildfly.



3.4 Configuración de Modo Dominio para Wildfly

El modo Dominio de Wildfly permite tener un punto único para la administración y despliegue de las aplicaciones Java EE, como también tener una gestión eficiente de cada host y nodo que conforma el grupo de aplicaciones. A continuación, se presentan los diferentes pasos que se realizó para la configuración del grupo de servidores de Wildfly.

3.4.1 Configuración del Host Wildfly Maestro

Esta configuración se realizó en la máquina virtual: “mv-cloud-srvapp1”, y se detalla a continuación.

Inicialmente se accedió al directorio que contiene el archivo de configuración de dominio: “host-master.xml” para la máquina maestra mediante los siguientes comandos en consola:

- cd /opt/wildfly/domain/configuration
- sudo nano host-master.xml

En las primeras líneas del archivo se configuró el nombre del host: “master-cloud-utn”, correspondiente al host maestro del grupo de servidores, tal como se muestra en la Figura 3.

Figura 3

Definición de nombre de host maestro

```
<?xml version="1.0" ?>

<host xmlns="urn:jboss:domain:19.0" name="master-cloud-utn">
```

Se añadió en el archivo la configuración de las interfaces, tal como se muestra en la Figura 4, denominadas “private” e “unsecured” con el objetivo de establecer un canal de conexión entre la máquina maestra y la máquina esclava.

Figura 4

Interfaces del archivo de configuración

```
<interfaces>
    <interface name="management">
        <inet-address value="${jboss.bind.address.management:127.0.0.1}"/>
    </interface>
    <interface name="public">
        <inet-address value="${jboss.bind.address:127.0.0.1}"/>
    </interface>
    <interface name="private">
        <inet-address value="127.0.0.1"/>
    </interface>
    <interface name="unsecure">
        <inet-address value="127.0.0.1"/>
    </interface>
</interfaces>
```

3.4.2 Configuración del Host Wildfly Esclavo

Esta configuración se realizó en la máquina virtual: “mv-cloud-srvapp2”. El proceso de configuración se detalla a continuación.

Se accedió al directorio que contiene el archivo de configuración de dominio para la máquina esclava denominado “host-slave.xml” mediante los siguientes comandos:

- cd /opt/wildfly/domain/configuration/
- sudo nano host-slave.xml

En el archivo “host-slave.xml” abierto anteriormente se procedió a realizar las siguientes configuraciones:

En las primeras líneas del archivo se configuró el nombre del host: “slave-cloud-utn”, correspondiente al host esclavo del grupo de servidores, tal como se muestra en la Figura 5

Figura 5

Definición de nombre de host esclavo

```
<?xml version="1.0" ?>

<host xmlns="urn:jboss:domain:19.0" name="slave-cloud-utn">
```

Se añadió las interfaces denominadas “private” y “unsecured”, como se muestra en la Figura 6.

Figura 6

Interfaces del archivo de configuración

```
<interfaces>
    <interface name="management">
        <inet-address value="${jboss.bind.address.management:127.0.0.1}"/>
    </interface>
    <interface name="public">
        <inet-address value="${jboss.bind.address:127.0.0.1}"/>
    </interface>
    <interface name="private">
        <inet-address value="127.0.0.1"/>
    </interface>
    <interface name="unsecure">
        <inet-address value="127.0.0.1"/>
    </interface>
</interfaces>
```

Como se muestra en la Figura 7, se añade el método de autenticación para la conexión con la máquina maestra, en el que se incluye el nombre de usuario y contraseña de Wildfly definida para la conexión con la máquina maestra.

Figura 7

Método de autenticación con la máquina maestra.

```
<authentication-client>
    <authentication-configuration name="hostAuthConfig" authentication-name="nodoEsclavo" realm="ManagementRealm"
        <credential-reference clear-text="[redacted]"/>
    </authentication-configuration>
    <authentication-context name="hcAuthContext">
        <match-rule authentication-configuration="hostAuthConfig"/>
    </authentication-context>
</authentication-client>
```

Se añadió los parámetros del controlador de dominio, indicando la dirección IP e la máquina maestra, el nombre de usuario, el contexto de autenticación, el puerto y el protocolo.

Figura 8

Configuración del controlador de dominio

```
<domain-controller>
    <remote protocol="http-remoting" host="10.24.8.88" port="9990" authentication-context="hcAuthContext" username="nodoEsclavo"/>
</domain-controller>
```

3.4.3 Ejecución de los servidores de aplicaciones

Como paso anterior a la ejecución de los servidores de aplicaciones, se realizó la configuración de los puertos del firewall TCP y UDP necesarios para el funcionamiento y comunicación entre los nodos del grupo de servidores. Los puertos que se configuró son los siguientes:

- Puerto TCP 8009: Puerto utilizado por el protocolo Apache JServ (AJP) para el balanceo de carga y comunicación de los nodos de un clúster HTTP.

- Puerto TCP 9990: Puerto de la consola de administración web.
- Puerto UDP 45688: Puerto utilizado para el descubrimiento de nodos en clústers HA mediante UDP.

La configuración de los diferentes puertos se lo realizó mediante el siguiente comando:

- sudo firewall-cmd --zone=public --add-port= *nro. puerto* / *protocolo* –permanent
- sudo firewall-cmd --reload

Completado la configuración del firewall, se procede a la ejecución de los servidores. Para todas las instalaciones de Wildfly, los scripts de ejecución se encuentran en el directorio "/bin", y se accedió mediante el comando:

- cd /opt/wildfly/bin

Para la ejecución del servidor "Maestro" se lo realizó mediante la ejecución del siguiente script:

- sudo ./domain.sh --host-config=host-master.xml -b 10.24.8.88 -bmanagement 10.24.8.88 -Djgroups.bind_addr=10.24.8.88

Para la ejecución del servidor denominado "Esclavo" se lo realizó mediante la ejecución del siguiente script:

- sudo ./domain.sh --host-config=host-slave.xml -b 10.24.8.89 -Djboss.domain.master.address=10.24.8.88 -Djgroups.bind_addr=10.24.8.89

Los parámetros de ejecución para los diferentes servidores de aplicaciones se detallan a continuación:

- -b: Este parámetro indica la dirección de red que utilizará el servidor de aplicaciones.
- -bmanagement: Este parámetro indica la dirección de red que utilizará el servidor de aplicaciones para el acceso a sus opciones de administración.
- -Djboss.domain.master.address: Este parámetro indica la dirección de red del controlador de host maestro.
- -Djgroups.bind_addr: Este parámetro indica la dirección de red que se utilizará para la configuración de la interfaz de JGroups, y es la encargada de proporcionar un soporte

de comunicación grupal para los servicios de alta disponibilidad entre los diferentes nodos.

3.4.4 Configuración de grupo de servidores.

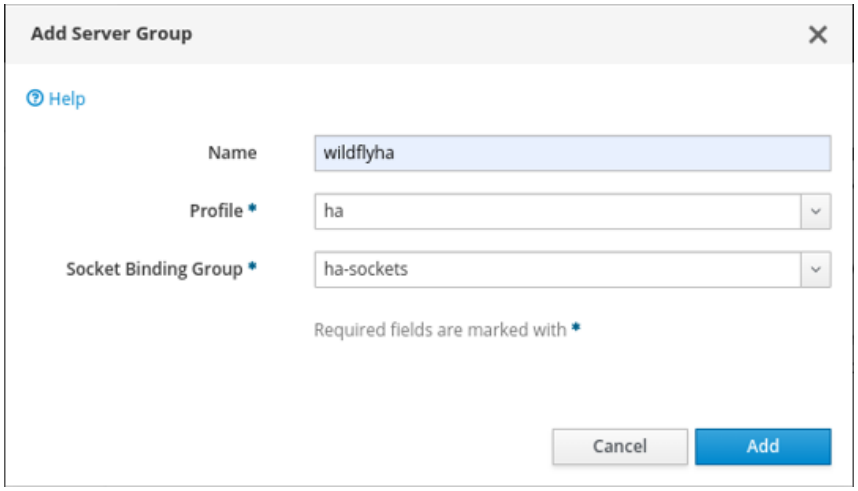
Para la creación y configuración del grupo de servidores de Wildfly se accedió a la consola de administración web de la máquina maestra, mediante el siguiente enlace en el navegador:

- http://10.24.8.88:9990/console/index.html

Una vez ingresado a la consola de administración web se dirigió a la sección denominada “Runtime” , y en la opción de “Add Server Group” se creó el grupo de servidores denominado “wildflyha”, asignándole un perfil de alta “ha” y un grupo de enlace de socket “ha-sockets”, como se muestra en la Figura 9.

Figura 9

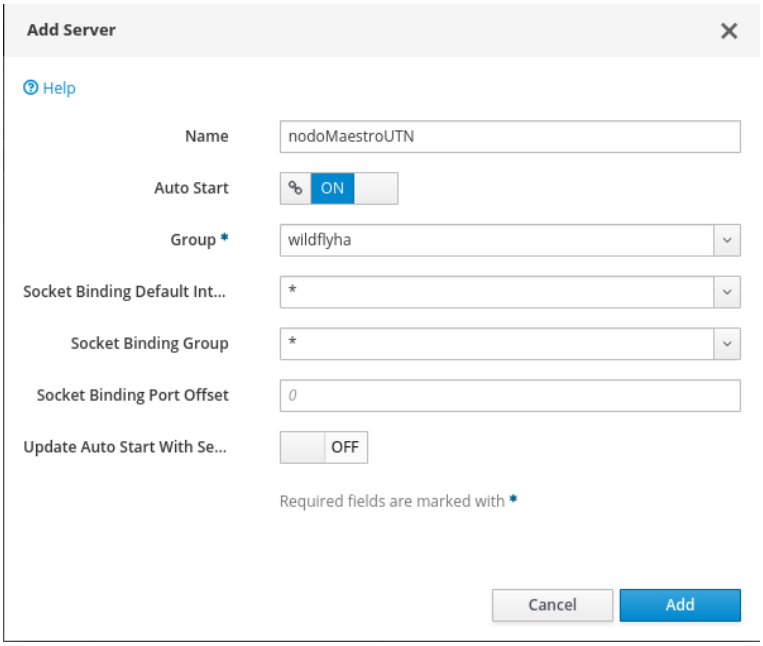
Creación de grupo de servidores



Como se muestra en la Figura 10, se añadió el servidor definido como: “nodoMaestroUTN” para el host principal o también denominado “master-cloud-utn”. A este servidor se le asignó el grupo de servidores: “wildflyha” creado anteriormente.

Figura 10

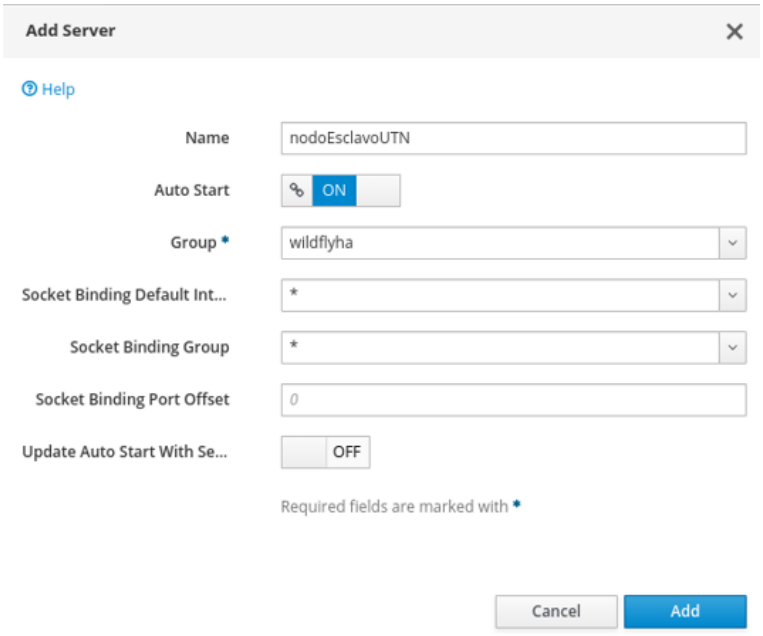
Creación de nodoMaestroUTN



Como se muestra en la Figura 11, se añadió el servidor definido como “nodoEsclavoUTN” para el host secundario o también denominado “slave-cloud-utn”. A este servidor se le asignó el grupo de servidores “wildflyha” creado anteriormente.

Figura 11

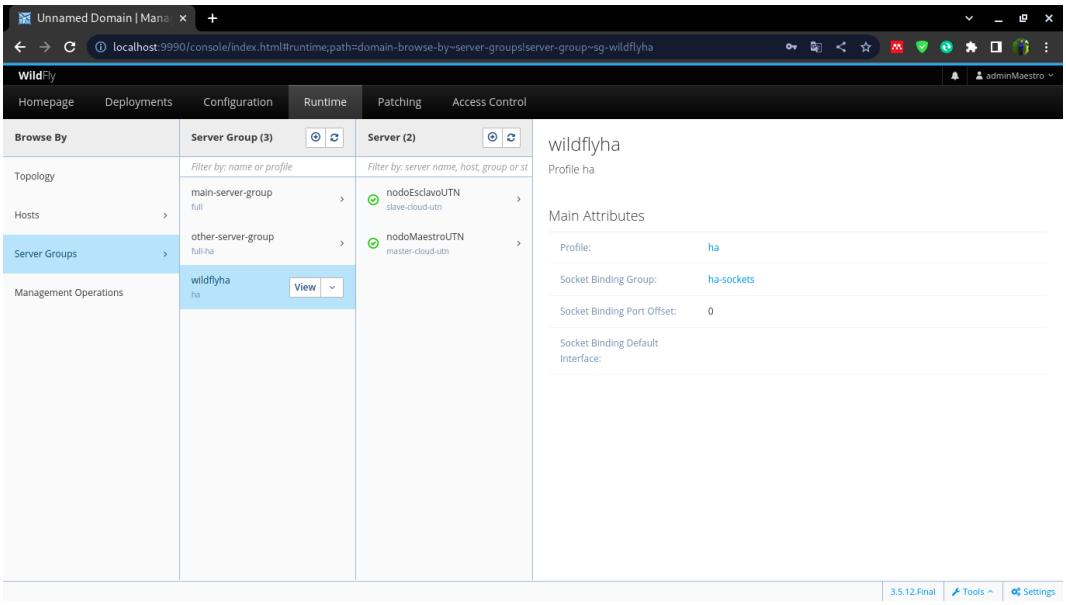
Creación de nodoEsclavoUTN



En la Figura 12 se observa la consola de administración web de Wildfly con el grupo de servidores y los diferentes nodos creados.

Figura 12

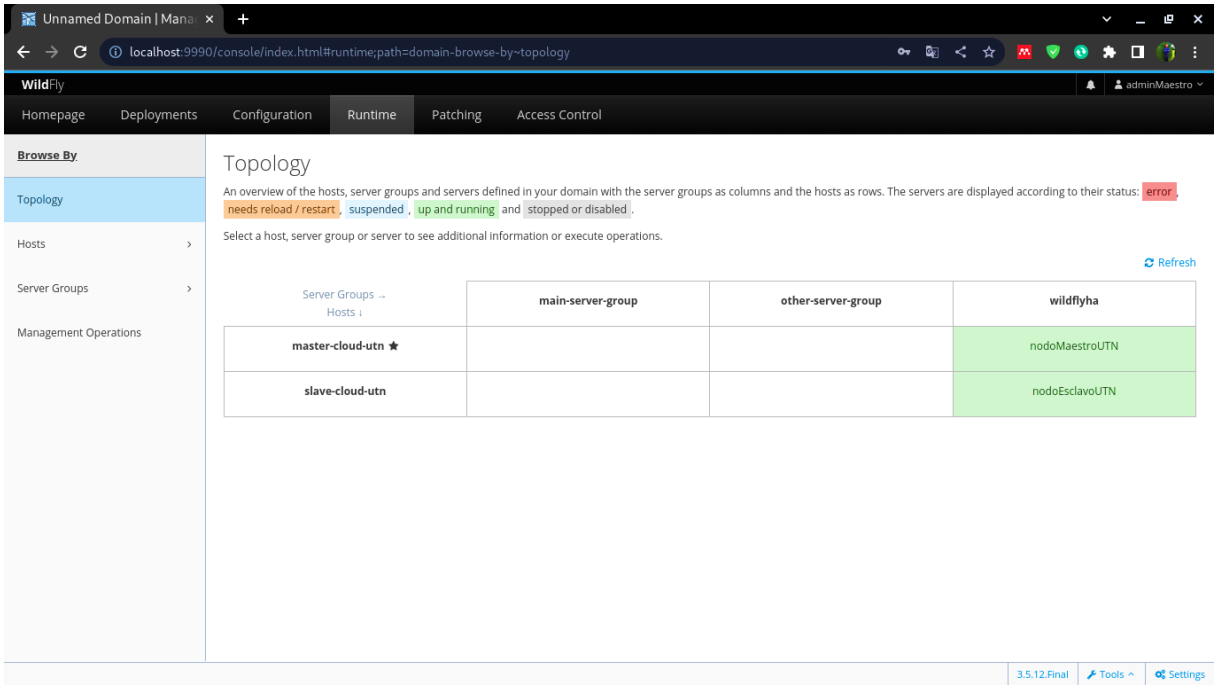
Grupo de servidores con nodos correspondientes



En la Figura 13 se presenta de forma visual la topología final configurada, indicando adicionalmente que están funcionales y listo para el despliegue de aplicaciones.

Figura 13

Topología de grupo de servidores



3.5 Configuración del balanceador de carga

El subsistema Modcluster permite utilizar una instancia de Wildfly como balanceador de carga y que adicionalmente tenga funciones de conmutación de sesiones por error de un

nodo dentro de un grupo de servidores. Este servicio se ejecutó en la máquina virtual: “mv-cloud-srvbc” y su configuración se detalla a continuación.

Como punto inicial, se realizó la configuración de los puertos del firewall TCP y UDP necesarios para la comunicación con las consolas de administración y comunicación de las aplicaciones web hacia los usuarios. Los puertos que se configuró son los siguientes:

- Puerto TCP 9990: Puerto de la consola de administración web.
- Puerto TCP 8080: Puerto por defecto para la comunicación de las aplicaciones web implementadas.

La configuración de los diferentes puertos se lo realizó mediante el siguiente comando:

- sudo firewall-cmd --zone=public --add-port= *nro. puerto* / *protocolo* –permanent
- sudo firewall-cmd –reload

Se puso en funcionamiento al servidor en su configuración: “standalone” con el archivo de configuración: “standalone-ha.xml”. Para esta tarea se accedió al directorio “/opt/wildfly/bin” y se ejecutó el servidor mediante los siguientes comandos:

- cd /opt/wildfly/bin
- sudo ./standalone.sh -c standalone-ha.xml -b 10.24.8.87 -bmanagement 10.24.8.87

Los parámetros del comando anterior se detallan a continuación:

- -c: Este parámetro especifica el archivo de configuración que ejecutará el servidor de aplicaciones.
- -b: Este parámetro indica la dirección de red que utilizará el servidor de aplicaciones.
- -bmanagement: Este parámetro indica la dirección de red que utilizará el servidor de aplicaciones para el acceso a sus opciones de administración

Una vez inicializado el servicio, se configuró el subsistema Modcluster mediante la herramienta de línea de comandos (CLI) de Wildfly, la cual se accedió mediante la ejecución de los siguientes comandos en la terminal.

- cd /opt/wildfly/bin
- ./jboss-cli.sh –connect --controller=10.24.8.87:9990

El subsistema Undertow tiene la capacidad de actuar como un proxy inverso de alto rendimiento compatible con Modcluster. Mediane ejecución de los siguientes comandos la herramienta de línea de comandos (CLI) de Wildfly se realizó la configuración del filtro y el grupo de multidifusión necesario para el servicio de Modcluster:

- /subsystem=undertow/configuration=filter/mod-cluster=modcluster:add(advertise-socket-binding=modcluster,management-socket-binding=http)
- /subsystem=undertow/server=default-server/host=default-host/filter-ref= modcluster:add
- :reload-servers

Finalmente se realizó la configuración del servidor de aplicaciones maestro para su integración con el balanceador de carga. Se configuró mediante la mediante la herramienta de línea de comandos (CLI) de Wildfly en la máquina virtual: “mv-cloud-srvapp1” ejecutando los siguientes comandos:

- cd /opt/wildfly/bin
- ./jboss-cli.sh --connect --controller=10.24.8.88:9990

Para el grupo de sockets de alta disponibilidad “ha-sockets”, se definió una lista estática de proxies mediante la creación de un socket remoto que contendrá el host del servidor que contiene el servicio de Modcluster con el puerto de comunicación.

- /socket-binding-group=ha-sockets/remote-destination-outbound-socket-binding=proxy1:add(host=10.24.8.87, port=8080)

Para el perfil de alta disponibilidad “ha” se hizo referencia al socket remoto añadiendo el atributo “proxies” y el valor establecido en el punto anterior.

- /profile=ha/subsystem=modcluster/proxy=default:write-attribute(name=proxies, value=[proxy1])

Finalmente, tal como se observa en la Figura 14, se registró que en la salida de consola del servidor de balanceado de carga cada uno de los nodos configurados en el grupo de

servidores aplicaciones, confirmando que se ha completado correctamente de creación de la arquitectura de alta disponibilidad.

Figura 14

Salida de consola de servidor de balanceo de carga

```
18:47:18,561 INFO  [org.jboss.as] (Controller Boot Thread) WFLYSRV0051: Admin console listening on http://10.24.8.87:9990
18:47:42,167 INFO  [io.undertow] (default task-1) UT005053: Registering node master-cloud-utn:nodoMaestro, connection: ajp://10.24.8.88
:8009/?#
18:47:42,184 INFO  [io.undertow] (default task-1) UT005045: Registering context /, for node master-cloud-utn:nodoMaestro
18:47:42,192 INFO  [io.undertow] (default task-1) UT005045: Registering context /wildfly-services, for node master-cloud-utn:nodoMaestr
o
18:47:50,459 INFO  [io.undertow] (default task-1) UT005053: Registering node slave-cloud-utn:nodoEsclavo, connection: ajp://10.24.8.89:
8009/?#
18:47:50,463 INFO  [io.undertow] (default task-1) UT005045: Registering context /, for node slave-cloud-utn:nodoEsclavo
18:47:50,466 INFO  [io.undertow] (default task-1) UT005045: Registering context /wildfly-services, for node slave-cloud-utn:nodoEsclavo
18:47:52,246 INFO  [io.undertow] (default task-1) UT005045: Registering context /cluster-demo, for node master-cloud-utn:nodoMaestro
18:48:00,497 INFO  [io.undertow] (default task-1) UT005045: Registering context /cluster-demo, for node slave-cloud-utn:nodoEsclavo
```

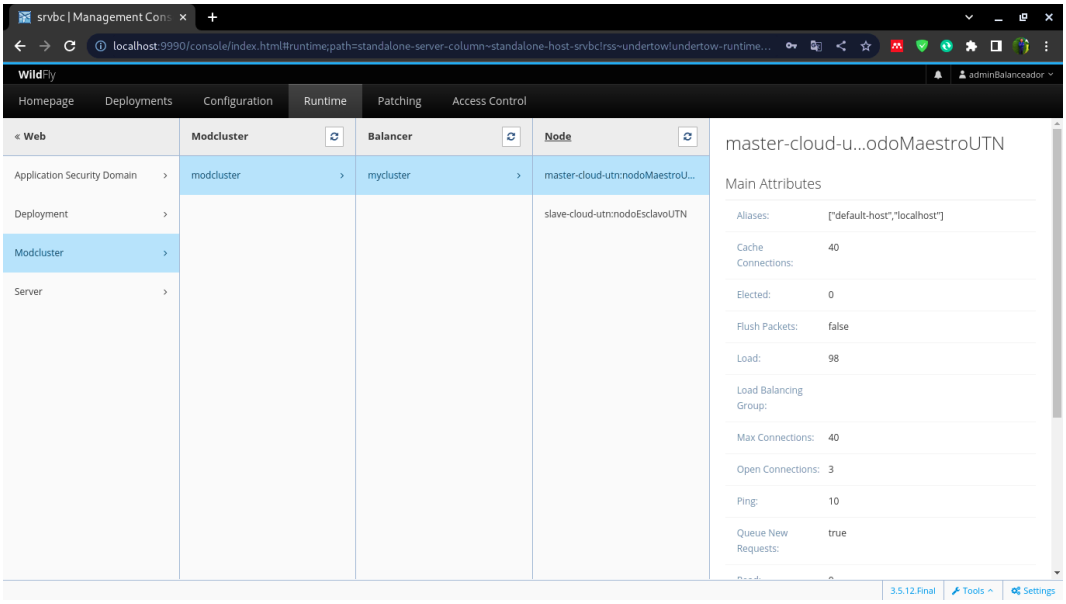
De igual forma se puede obtener información detallada de cada nodo de una forma visual mediante la consola de administración de Wildfly, la cual se accedió mediante el enlace en el navegador:

- http://10.24.8.87:9990/console/index.html

En la Figura 15 se observa la información general del subsistema Modcluster, donde se presenta los atributos de cada nodo, como su nombre, número de peticiones redirigidas, el nivel de carga, ping, número de conexiones abiertas, entre otros.

Figura 15

Interfaz web de administración de Wildfly para Modcluster



3.6 Pruebas

3.6.1 Despliegue de aplicación demo.

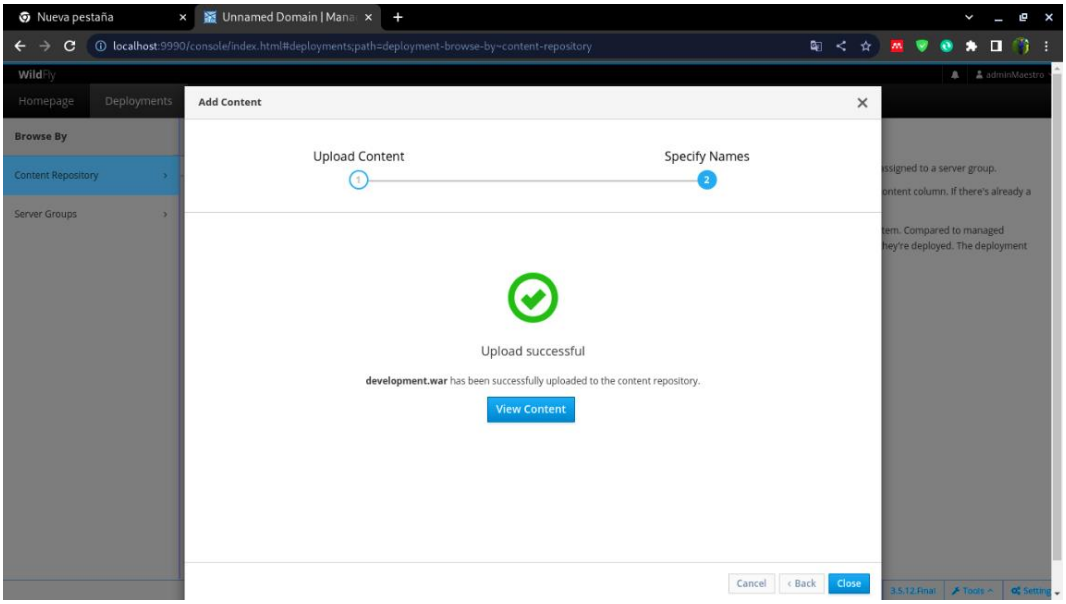
Para la verificación de las funcionalidades del balanceador de carga y de alta disponibilidad del servidor de aplicaciones, se realizó el despliegue de la aplicación JavaEE denominada “cluster-demo.war” mediante la consola web de administración. Esta aplicación proporcionada por Wildfly se puede acceder mediante el repositorio:

- https://github.com/liweinan/cluster-demo

En la Figura 16 se presenta la subida del archivo “cluster-demo.war” al repositorio de contenido del servidor de aplicaciones Wildfly.

Figura 16

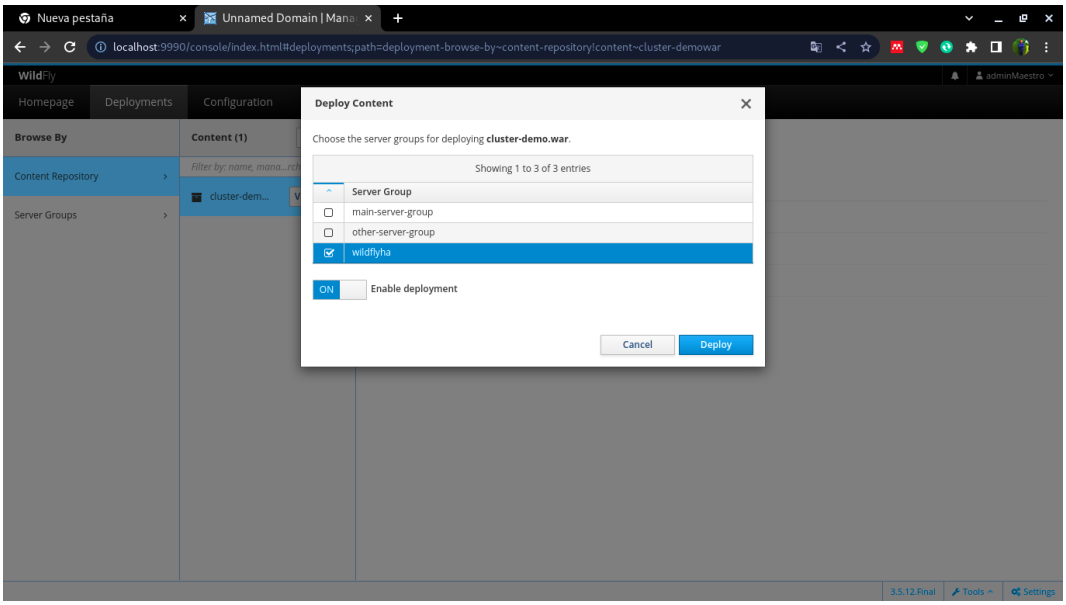
Subida del archivo “cluster-demo.war” al repositorio de contenido



En la Figura 17 se presenta el despliegue de la aplicación al grupo de servidores de alta disponibilidad “wildflyha”, y que gracias a su configuración de dominio se ejecutará en los diferentes nodos del grupo de aplicaciones.

Figura 17

Despliegue de aplicación en grupo de servidores "wildflyha"



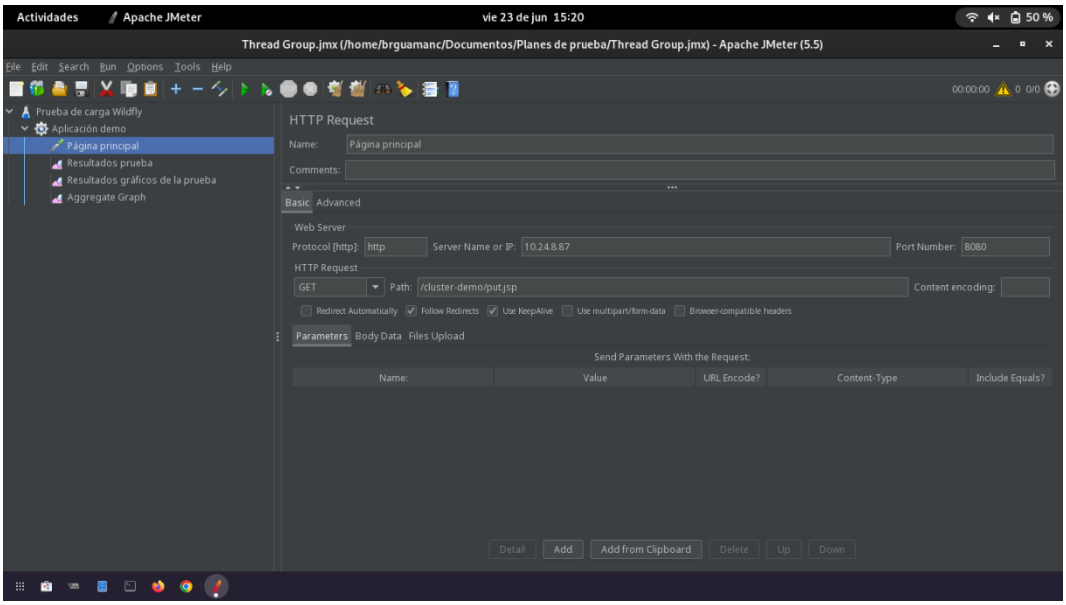
3.6.2 Prueba de carga

Se estableció un plan de pruebas mediante el software Apache JMeter en su versión 5.5 para comprobar el funcionamiento del balanceo de carga entre nodos del grupo de servidores de aplicaciones. En la Figura 18 se observa los diferentes parámetros que fueron establecidos para la realización de la prueba de carga y que se detallarán a continuación.

- Protocol (http): htttp
- Server Name or IP: 10.24.8.87
- Port Number: 8080
- HTTP Request: Get
- Path: /cluster-demo/put.jsp

Figura 18

Ventana principal del plan de pruebas



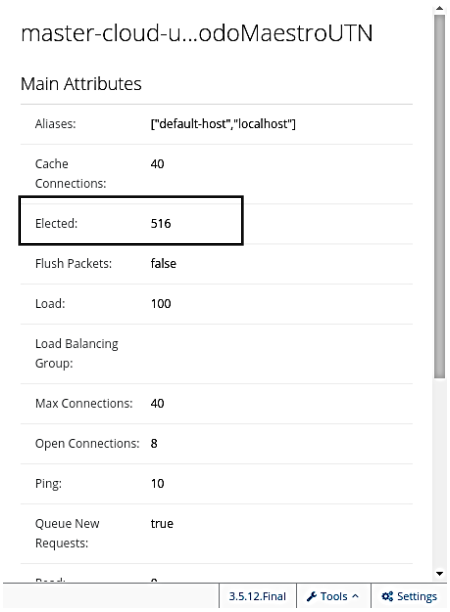
Se estableció diferentes números usuarios (50,100,200,500,1000) que accederán al servidor de balanceado de carga en un intervalo de 60 segundos, tal como se muestra en la Figura 19, con el objetivo de observar el número de cargas que se ha redirigido a cada nodo del grupo de servidores de aplicaciones, verificado mediante la consola de administración web de Wildfly mediante el atributo “Elected”, que representa el número de peticiones que fueron atendidas por cada nodo, tal como se muestra en la Figura 20.

Figura 19

Definición del número de usuarios y periodo de tiempo de la prueba

Figura 20

Ejemplo de verificación de número de peticiones recibidas en el nodo Maestro



3.6.3 Resultados de pruebas de carga.

En la Tabla 3, se presenta los diferentes resultados de las pruebas de balanceo de carga con los diferentes números de usuarios y tomando en cuenta los parámetros generales definidos en la sección anterior.

Tabla 3

Resultados de prueba de balanceo de carga.

Número de usuarios	Periodo de tiempo de la prueba (seg)	Nodo del grupo de servidores de Wildfly	Número de peticiones recibidas por nodo
50	60	Nodo Maestro	23
		Nodo Esclavo	27
100	60	Nodo Maestro	52
		Nodo Esclavo	48
200	60	Nodo Maestro	112
		Nodo Esclavo	88
500	60	Nodo Maestro	244
		Nodo Esclavo	256
1000	60	Nodo Maestro	489
		Nodo Esclavo	511

3.6.4 Pruebas de disponibilidad

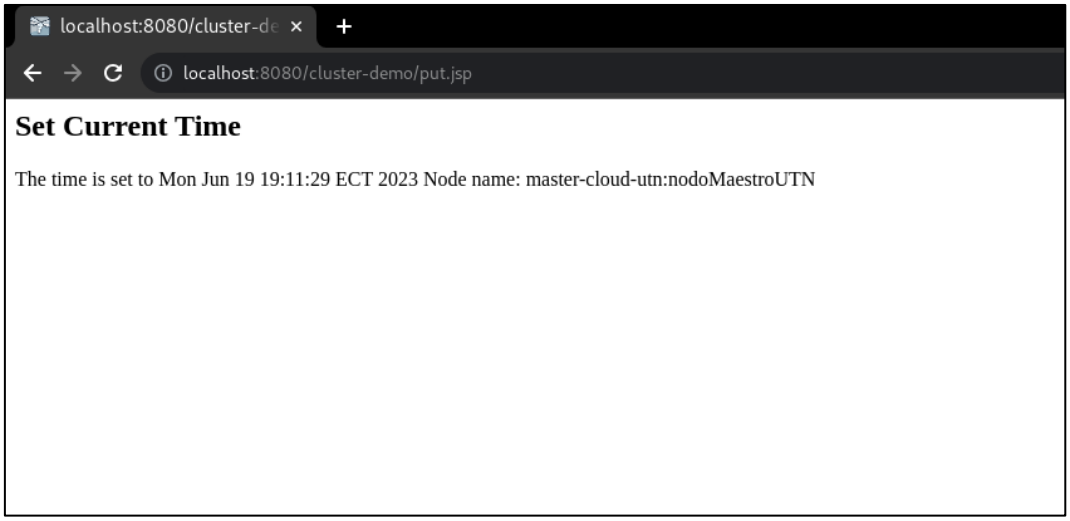
Para la realización de las pruebas de disponibilidad se accedió a la de aplicación demo desplegada, la cual consta de dos páginas: la primera denominada “put.jsp” está encargada de capturar en sesión la fecha y hora en la que se accedió y de mostrar el nodo que atendió la petición, y la página denominada “get.jsp” que está encargada de mostrar la hora almacenada en sesión. Mediante la simulación de indisponibilidad del nodo que atendió la petición se pude comprobar que los datos de sesión del usuario se han replicado en el siguiente nodo y la navegación continua normalmente, confirmando así que se configuró correctamente la solución de alta disponibilidad.

El proceso de la prueba de disponibilidad se detalla a continuación:

- Primero, se accedió a la página “put.jsp” en el navegador, donde se mostrará y se almacenará en sesión la fecha y hora actual, juntamente con el nombre del nodo que atendió la aplicación. En la Figura 21 se observa que en la prueba realizada la petición fue dirigida al nodo maestro. La petición puede ser atendida inicialmente por el nodo maestro o por el nodo esclavo, y no se considera como un problema para el desarrollo de la prueba.

Figura 21

Visualización de fecha, hora y nombre del nodo que atendió la petición



- A continuación, se accedió a la página “get.jsp” en el navegador, donde se mostrará los datos de sesión del usuario que fueron guardados en el punto anterior, tal como se presenta en la Figura 22.

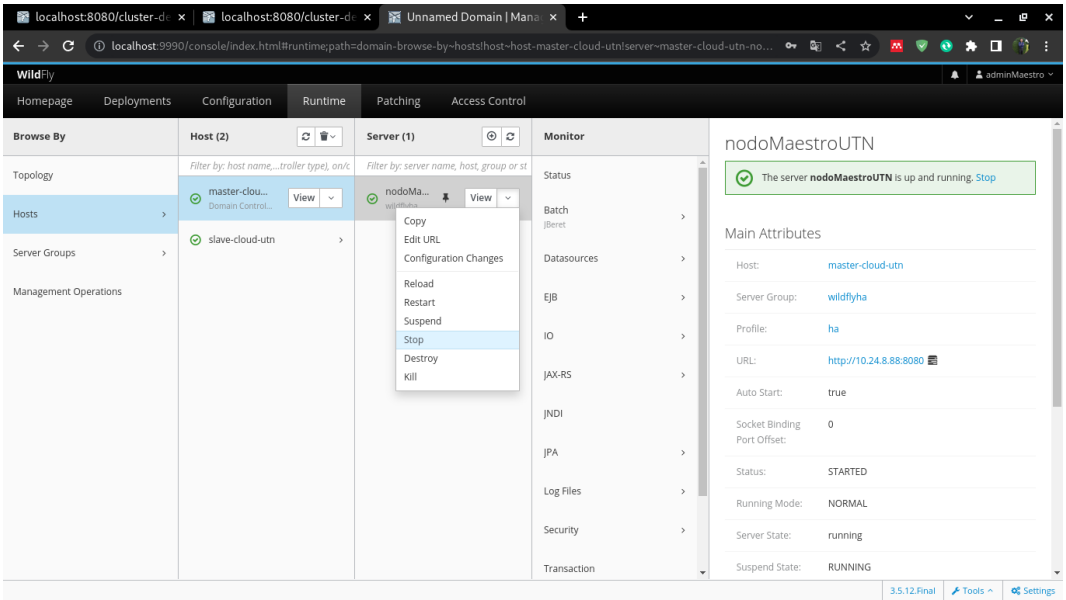
Figura 22

Visualización de los datos almacenados en sesión

- Se realizó la prueba de indisponibilidad el nodo que recibió la petición mediante la consola de administración web de Wildfly. En esta demostración se suspendió el nodo maestro, tal como se muestra en la Figura 23.

Figura 23

Visualización gráfica del estado de suspensión del nodo Maestro



- Se recargó la página y se verificó que el navegador no presente mensajes de error o de problemas de conexión. Adicionalmente se verificó que los datos de sesión del usuario se visualizan con normalidad, tal como se presenta en la Figura 24. De esta manera se comprueba que se trasladó los datos de sesión del usuario al siguiente nodo.

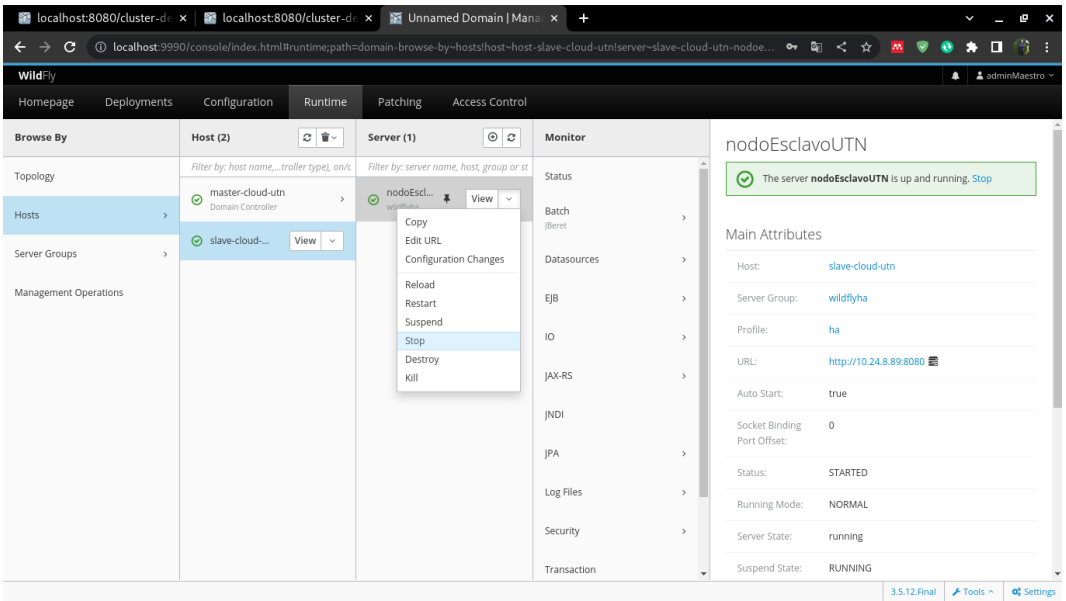
Figura 24

Visualización de los datos guardados en sesión del usuario

- Para completar con la prueba, se replicó la prueba simulando la indisponibilidad del nodo que recibió inicialmente la petición, tal como se muestra en la Figura 25.

Figura 25

Visualización gráfica del estado de suspensión del nodo Esclavo



3.6.5 Resultados de prueba de disponibilidad

A continuación, en la Tabla 4, se presenta los diferentes resultados de la prueba de disponibilidad realizada.

Tabla 4

Resultados de pruebas de disponibilidad

Descripción de la prueba	Resultado esperado	Cumple (Si/No)
Simular la indisponibilidad del nodo maestro.	Al recargar la página en el navegador no se presentan mensajes en pantalla de error de conexión o de interrupción.	Si
	Al recargar la página en el navegador no se pierde la comunicación a la aplicación desplegada.	Si
	Al recargar la página del navegador los datos de sesión se mantienen y el usuario puede continuar navegando con normalidad.	Si
Simular la indisponibilidad del nodo esclavo.	Al recargar la página en el navegador no se presentan mensajes en pantalla de error de conexión o de interrupción.	Si
	Al recargar la página en el navegador no se pierde la comunicación a la aplicación desplegada.	Si
	Al recargar la página del navegador los datos de sesión se mantienen y el usuario puede continuar navegando con normalidad.	Si

4. Conclusiones

Se implementó satisfactoriamente la arquitectura de alta disponibilidad en el Cloud FICA de la Universidad Técnica del Norte para el servidor de aplicaciones Wildfly.

La integración entre el balanceador de carga y el grupo de servidores de aplicaciones de Wildfly se realizó correctamente.

Se comprobó un correcto funcionamiento de la técnica de alta disponibilidad para el despliegue de aplicaciones Java EE mediante el cumplimiento exitoso de las diferentes pruebas de funcionamiento y disponibilidad.

5. Observaciones y Recomendaciones

La arquitectura implementada está compuesta por componentes de software libre y que junto al contenido presentado a la guía puede ser replicable para futuros proyectos e investigaciones acerca de la alta disponibilidad en servidores de aplicaciones.

Se recomienda utilizar las mismas versiones de Wildfly para el balanceador y el grupo de servidores de aplicaciones, con el fin de evitar incompatibilidades al momento de la implementación.

Se recomienda seguir configuraciones que estén recomendadas dentro de la sección de documentación oficial de Wildfly de acuerdo a la versión que se esté utilizando, con la finalidad de evitar configuraciones obsoletas e incompatibles.

Se recomienda la utilización de la tecnología Modcluster mediante la configuración de Wildfly como balanceador de carga, ya que en este caso se cuenta con la consola de administración web que permite una mejor visualización de cada nodo que compone la arquitectura y detalles de la carga que recibe cada nodo del grupo de servidores de aplicaciones.

Para trabajos futuros se recomienda la integración de otras técnicas de alta disponibilidad dentro de la infraestructura (alta disponibilidad de bases de datos, redundancia de hardware, entre otros), para así garantizar tener todo un entorno de alta disponibilidad que permita ofrecer un funcionamiento continuo ante la presencia de errores en otras secciones de la infraestructura.

Apéndice C. Encuesta para validación de Guía mediante Delphi

Validación de guía de implementación mediante metodología Delphi.

Facultad de Ingeniería y Ciencias Aplicadas - Carrera de Software

Tema: Validación de guía de implementación de alta disponibilidad.

Objetivo: El siguiente cuestionario tiene como finalidad la validación de una guía de implementación mediante la metodología Delphi, la cual corresponde a un entregable del trabajo de grado: "Implementación de un balanceador de carga en el Cloud FICA de la Universidad Técnica del Norte".

Instrucciones.
El cuestionario consta de 10 ítems divididos en 3 categorías (Redacción y contenido, Claridad, Replicabilidad), los cuales van a ser calificadas según su percepción como experto en una escala de Likert de 4 puntos, en donde se tiene en cuenta que :

Escala	*Indicador*
1	No cumple con el criterio Cumple
2	parcialmente el criterio Cumple
3	aceptablemente el criterio Cumple
4	completamente el criterio

Tiempo estimado: 10 minutos

Agradecemos su colaboración.

* Obligatoria

* Este formulario registrará su nombre, escriba su nombre.

1. **Categoría: Redacción y contenido** *

	1	2	3	4
¿La guía contiene un lenguaje adecuado y comprensible para el público objetivo?	◯	◯	◯	◯
¿La guía utiliza correctamente las normas gramaticales y ortográficas?	◯	◯	◯	◯
¿Las gráficas y tablas presentadas en la guía están adecuadamente etiquetadas y tituladas para facilitar su comprensión ?	◯	◯	◯	◯
¿La calidad visual de las figuras presentadas en la guía es adecuada para su interpretación?	◯	◯	◯	◯

2. **Categoría: Claridad** *

	1	2	3	4
¿La guía es clara y fácil de entender en relación con su temática?	◯	◯	◯	◯
¿La guía contiene una estructura organizada y coherente que facilite la comprensión y seguimiento del contenido?	◯	◯	◯	◯
¿La descripción de cada paso y configuración en la guía proporciona suficiente detalle y claridad para llevar a cabo las acciones necesarias?	◯	◯	◯	◯

3. **Categoría: Replicabilidad** *

	1	2	3	4
¿La información y procedimientos presentados en la guía proporciona información detallada y suficiente para que otros investigadore s puedan replicar la implementación de forma precisa?	◯	◯	◯	◯
¿Las tecnologías utilizadas en la guía son confiables y adecuadas para replicar la implementación en otros entornos?	◯	◯	◯	◯
¿La guía ofrece recomendaciones o sugerencias relacionadas con el tema desarrollado?	◯	◯	◯	◯