UNIVERSIDAD NACIONAL DE CHIMBORAZO

FACULTAD DE INGENIERÍA

CARRERA INGENIERÍA EN SISTEMAS Y COMPUTACIÓN

TRABAJO DE TITULACIÓN

Previo a la obtención del Título de:

INGENIERO EN SISTEMAS Y COMPUTACIÓN

TITULO DEL PROYECTO:

ANÁLISIS COMPARATIVOS DE SERVIDORES DE APLICACIONES OPEN SOURCE PARA LA PLATAFORMA JAVA EE. CASO PRÁCTICO: MÓDULO DE GESTIÓN DE JUNTAS ADMINISTRADORAS DE AGUA POTABLE Y RIEGO PARA LA DIRECCIÓN PROVINCIAL DE CHIMBORAZO DE LA SENAGUA

Autores:

Darwin Mauricio Balbuca Ramones

José Miguel Ortiz Ramírez

DIRECTOR DEL PROYECTO DE INVESTIGACIÓN:

Ing. Jorge Edwin Delgado Altamirano

RIOBAMBA – ECUADOR

2017

Los miembros del Tribunal de Graduación del proyecto de investigación de título: **"ANÁLISIS COMPARATIVOS DE SERVIDORES DE APLICACIONES OPEN SOURCE PARA LA PLATAFORMA JAVA EE. CASO PRÁCTICO: MÓDULO DE GESTIÓN DE JUNTAS ADMINISTRADORAS DE AGUA POTABLE Y RIEGO PARA LA DIRECCIÓN PROVINCIAL DE CHIMBORAZO DE LA SENAGUA"** presentado por: Darwin Mauricio Balbuca Ramones y José Miguel Ortiz Ramírez y dirigida por: Ing. Jorge Edwin Delgado Altamirano.

Una vez escuchada la defensa oral y revisado el informe final del proyecto de investigación con fines de graduación escrito en la cual se ha constatado el cumplimiento de las observaciones realizadas, remite la presente para uso y custodia en la biblioteca de la Facultad de Ingeniería de la UNACH.

Para constancia de lo expuesto firman:

Ing. Danny Patricio Velasco Silva.
Presidente del Tribunal **Firma**

Ing. Jorge Edwin Delgado Altamirano
Director del Proyecto **Firma**

Ing. Samuel Emilio Moreno Aguirre
Miembro del Tribunal **Firma**

AUTORÍA DE LA INVESTIGACIÓN

La responsabilidad del contenido de este Proyecto de Graduación, corresponde exclusivamente a: Darwin Mauricio Balbuca Ramones y José Miguel Ortiz Ramírez, autores del proyecto de investigación, al Ing. Jorge Edwin Delgado Altamirano y el patrimonio intelectual de la misma a la Universidad Nacional de Chimborazo.

Darwin Mauricio Balbuca Ramones
C. C. 140066290-2

José Miguel Ortiz Ramírez
C. C. 080207116-7

Ing. Jorge Edwin Delgado Altamirano
DIRECTOR DEL PROYECTO DE INVESTIGACIÓN

DEDICATORIA

El presente trabajo de titulación está dedicada primero a Jehová Dios, ya que gracias a él he logrado concluir con mi carrera.

A mi familia le dedico con todo mi corazón por su sacrificio y esfuerzo por creer en mi capacidad, aunque hemos pasado por momentos difíciles siempre ha estado brindándome su compresión cariño y amor. Por ser mi fuente de motivación e inspiración para poder superarme cada día más y así poder luchar para que la vida nos depare un futuro mejor. A mis amados padres y hermanos quienes con sus palabras de aliento no me dejaron decaer para que siguiera a delante y siempre sea perseverante y cumpla con mis ideales. A mis compañeros y amigos presentes y pasados quienes sin esperar nada a cambio compartieron sus conocimientos alegrías y tristezas.

A todos en general por darme el tiempo de realizarme profesionalmente

Gracias a todos

José Miguel Ortiz Ramírez

DEDICATORIA

Dedico este trabajo principalmente a Dios, por haberme dado la vida y permitirme haber llegado hasta este momento tan importante de mi formación profesional. A mi madre por ser el pilar más importante y demostrarme su cariño incondicional durante toda mi vida, ya que, sin ti no habría podido culminar esta meta tan anhelada para mí. A mi padre que a pesar de la distancia me ha brindado su apoyo incondicional, a mis hermanos y demás familiares por estar junto a mí en esta etapa y a todos en general que estuvieron apoyándome para no dejarme caer.

Darwin Mauricio Balbuca Ramones

AGRADECIMIENTO

Agradezco a Dios por haberme otorgado una familia maravillosa quienes han creído en mí siempre dándome ejemplo de superación, humildad y sacrificio enseñándome a valorar todo lo que tengo les agradezco desde lo más profundo de mi corazón por ser mi motor y fomentar en mí el deseo de superación y de triunfar en la vida lo que ha contribuido a la consecución de este logro, a mi tutor Ing. Jorge Delgado, a nuestro amigo Ing. Danny Velasco y a todas aquellas personas que durante estos cinco años estuvieron a nuestro lado apoyándome y lograron que este sueño se haga realidad. Este es un día muy especial porque culmino una exitosa y sacrificada etapa de mi vida durante la cual di lo mejor de mí y viviendo experiencias que hoy hacen de mí una persona más preparada para enfrentar la vida y servir a los demás.

Espero siempre contar con su apoyo

José Miguel Ortiz Ramírez

AGRADECIMIENTO

Agradezco Dios por haberme brindado la sabiduría a lo largo de mi carrera, por ser mi fortaleza en los momentos de debilidad y por darme una vida llena de retos y sobre todo de felicidad en mi carrera.

Le doy gracias a mi familia en especial a mis padres Víctor e Inés por el esfuerzo tan grande que pusieron para darme los estudios y poderlos culminar con gran satisfacción, a mis hermanos por apoyarme cada día.

Son muchas las personas que han formado parte de mi vida estudiantil y me tardaría una eternidad en nombrarlos a cada uno sin importar en donde estén quiero darles las gracias por formar parte de mí, por todo el apoyo que me han brindado.

Mil gracias a cada uno de ustedes.

Darwin Mauricio Balbuca Ramones

INDICE GENERAL

INDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

Hoy en día el desarrollo de Sistemas de Información conlleva el empleo de una gran cantidad de recursos, y la necesidad de actualizar estas aplicaciones tanto en diseño como en contenido conllevan al empleo de un nuevo proceso de desarrollo, en este sentido un servidor de aplicaciones constituye el recurso principal para el despliegue y acceso recurrente a sistemas de información empresariales.

Al no disponer de información necesaria sobre las principales características sobre los servidores de aplicaciones y peor aún al desconocer completamente el funcionamiento de los mismos, el desarrollador no realiza su trabajo de forma eficiente, esto provoca la insatisfacción del usuario el mismo que demanda una alta calidad.

Por otro lado la Secretaría Nacional del Agua (SENAGUA) Dirección Provincial de Chimborazo, lleva el registro de los datos correspondientes a las Juntas Administradoras de agua potable y Juntas de Riego son registrados en matrices en hojas de Excel esto dificulta el acceso y seguimiento de los datos. A esta institución le interesa disponer de un sistema web a través del cual se puede realizar el seguimiento de las juntas mencionadas anteriormente, así como también la generación de reportes sobre la situación actual de las mismas.

Con todo lo expuesto anteriormente en el presente trabajo investigativo se realiza el análisis comparativo de las funcionalidades técnicas de los servidores de aplicaciones open source para la plataforma Java EE más usados a nivel como son: GlassFish y WildFly; de éstos servidores se evaluaron los criterios de: peticiones por segundo, el tiempo medio que tarda el servidor en atender una petición, la tasa de transferencia de información y el porcentaje de pérdida de información, estos indicadores permiten determinar cuál es el servidor más óptimo para contener aplicaciones desarrolladas bajo la Plataforma Java EE.

Finalmente se explica el desarrollo e implementación del Sistema de Gestión de Juntas Administradoras de Agua Potable y Juntas de Riego, desarrollado en su totalidad con la plataforma Java en entorno web con la Arquitectura Multicapa y utiliza a GlassFish como contenedor de aplicaciones, el objetivo primordial de esta aplicación es optimizar los

procesos de seguimiento, gestión y automatización de reportes e informes estadísticos que apoyen los procesos administrativos y toma de decisiones de la Secretaria Nacional del Agua Chimborazo.

Abstract

Nowadays the development of Information Systems involves the use of a great amount of resources, and the need to update these applications in both design and content lead to the use of a new development process for this reason an application server is the main resource for the deployment and recurrent access to enterprise information systems. By not having the necessary information about the main characteristics on the application servers and worse yet to completely ignore the operation of it, the developer does not perform his work efficiently, this causes the user's dissatisfaction the same that demands a high quality. On the other hand, the National Water. Secretary department (SENAGUA) Provincial Directorate of Chimborazo, Carries the record of the data corresponding to the Managing Boards of drinking water and Irrigation Boards are registered in matrices in sheets of Excel this hinders the access and monitoring of the data. This institution is interested in having a web system through which you can follow the meetings mentioned above, as well as the generation of reports on the current situation of it. With all of the above, the comparative analysis of the technical functionalities of the open source application servers for the Java EE platform, and the most used ones are: Glass Fish and Wild Fly, from these servers the following criteria were evaluated: requests per second, the average time it takes the server to service a request, the rate of transfer of information and the percentage of loss of information. These indicators allow you to determine which server is most optimal for containing applications developed under the Java EE Platform. Finally it is explained the development and implementation of the Management System of Drinking Water Management Boards and Irrigation Boards, developed in its entirety with the Java platform in the web environment with the Multi-Layer Architecture and with the use of Glass Fish as an application container, The main objective of this application is to optimize the processes of monitoring, management and automation of reports and statistical reports that support the administrative processes and decision making of the National Secretary of Water in Chimborazo.

Reviewed by: Luis Barriga
Language Center Teacher

INTRODUCCIÓN

El desarrollo de sistemas de información está relacionado principalmente con las siguientes áreas de conocimiento como son: la ingeniería de software que sugiere herramientas, tecnologías y metodologías para dar solución a diversos tipos de problemas; los sistemas de base de datos que guían en el análisis, diseño, implementación y gestión de una base de datos; y, la ingeniería web que sugiere principios y herramientas para construcción de aplicaciones web de calidad y finalmente un servidor de aplicaciones que juegue un rol importante al momento de satisfacer las necesidades de los usuarios.

La presente investigación se compone de VI capítulos, pues así el Capítulo I inicia con un marco referencial del proyecto, seguido de los objetivos y la debida justificación. En el Capítulo II se sustenta teóricamente el presente trabajo además se realiza un breve estudio de las características que deben cumplir los servidores de aplicaciones, además se estudia de forma rápida la Plataforma Java EE y sus principales componentes. Éste capítulo servirá para fundamentar bases teóricas y científicas para determinar los principales indicadores a evaluar en el Análisis comparativo de las funcionalidades de los servidores de aplicaciones Open Source para la Plataforma Java EE.

El Capítulo III expone el Análisis comparativo de las funcionalidades técnicas de los servidores de aplicaciones GlassFish y WildFly, los mismos que en la documentación oficial de Java Oracle son los servidores más utilizados a nivel mundial. Se evaluaron los criterios Peticiones por segundo, el tiempo medio que tarda el servidor en atender una petición, la tasa de transferencia de información y el porcentaje de pérdida de información.

En el Capítulo IV se lleva a cabo el desarrollo e implementación del Sistema de Gestión de Juntas Administradoras de Agua Potable y Juntas de Riego, desarrollado en su totalidad con la plataforma Java en entorno web con el Arquitectura Multicapa y utiliza a GlassFish como contenedor de aplicaciones. Esta aplicación está destinada a optimizar los procesos de seguimiento, gestión y automatización de reportes e informes estadísticos que apoyen los procesos administrativos y toma de decisiones de la Secretaria Nacional del Agua – Chimborazo.

En el Capítulo V se definen los métodos, mecanismos, estrategias y/o procedimientos a seguirse en la investigación, se analiza los resultados del estudio comparativo y los beneficios de la misma, se discute y comprueba la hipótesis; y se finaliza la investigación en el Capítulo VI con las conclusiones y recomendaciones del proyecto de investigación.

CAPITULO I

MARCO REFERENCIAL

1.1. TÍTULO DEL PROYECTO

Análisis comparativos de Servidores de aplicaciones Open Source para la Plataforma java EE. Caso práctico: Módulo de Gestión de Juntas Administradoras de Agua Potable y Riego para la dirección provincial de Chimborazo de la SENAGUA.

1.2. PROBLEMATIZACION

1.2.1. IDENTIFICACIÓN Y DESCRIPCIÓN DEL PROBLEMA

El desarrollo de aplicaciones empresariales conlleva el empleo de una gran cantidad de recursos, y la necesidad de actualizar estas aplicaciones tanto en diseño como en contenido conllevan el empleo de un nuevo proceso de desarrollo, en este sentido la selección de la plataforma de desarrollo y el servidor de aplicaciones tienen un rol importante en el éxito o fracaso de una institución.

La crisis económica que afecta a varias países de la región latinoamericana, en la cual muchas instituciones del sector público tienen limitaciones presupuestarias para la adquisición de software especializado que sean el punto de apoyo central en los procesos de gestión dichas instituciones, el costo en licencias está relacionado directamente con el éxito o fracaso de las instituciones, el uso del software libre y aplicaciones Open Source en la mayoría de los casos resulta beneficioso. En Ecuador a través del Decreto 1014 firmado el 10 de abril de 2008 por el Presidente Rafael Correa se toma como política de Estado la adopción de Software Libre para todas sus entidades tanto en sus sistemas como también en equipamientos informáticos.

La Secretaría Nacional del Agua (SENAGUA) Dirección Provincial de Chimborazo, lleva el registro de los datos correspondientes a las Juntas Administradoras de agua potable y Juntas de Riego son registrados en matrices en hojas de Excel esto dificulta el acceso y seguimiento de los datos. A esta institución le interesa disponer de un sistema web a través del cual se puede realizar el seguimiento de las juntas mencionadas anteriormente, así como también la generación de reportes sobre la situación actual de las mismas.

La empresa Oracle Corporation propietaria de la plataforma Java muestra el listado compatible de servidores de aplicaciones para la plataforma Java EE, la cual está encabezada por WildFly y GlassFish como herramientas de software libre de código abierto en los que se puede implementar la aplicación que se va a desarrollar. Los principales beneficios de los servidores de aplicaciones son la centralización, la disminución de la complejidad en el desarrollo de aplicaciones, la escalabilidad y la integración e interoperabilidad con otras plataformas.

1.2.2. ANÁLISIS CRÍTICO

Los constantes cambios en la tecnología hacen necesario disponer de una adecuada metodología para el desarrollo de sistemas de información, si bien es cierto en la actualidad existen varios servidores de aplicaciones open source lo cual hace difícil escoger uno de éstos y aplicarlo en un proyecto de desarrollo, convirtiéndose en algo desconcertante para los desarrolladores.

La Dirección Provincial de Chimborazo de la Secretaría Nacional del Agua (SENAGUA) realiza el seguimiento de las Juntas Administradoras de Agua Potable de forma manual empleando hojas de cálculo de Excel, lo cual dificulta generar reportes e informes sobre la situación actual de las instituciones antes mencionadas. Es ahí donde es necesario implementar un sistema de entorno web que permita gestionar la información que maneja está institución.

El sitio especializado www.itbuzzpress.com[1] asegura que los servidores de aplicaciones para la plataforma Java EE están destinados a facilitar la centralización y el desarrollo de sistemas informáticos con la finalidad de que puedan crecer en el futuro, el sistema a desarrollar será la línea base y columna vertebral de futuros proyectos que puedan presentarse en la SENAGUA.

1.2.3. PROGNOSIS

Al analizar de forma comparativa los servidores de aplicaciones para la plataforma Java EE permitirán seleccionar el servidor más adecuado acorde a los requerimientos de la SENAGUA

[1] http://www.itbuzzpress.com

para implementar sistemas web en el ámbito empresarial. El servidor de aplicaciones seleccionado brindará los beneficios de centralización y disminución de la complejidad en el desarrollo de aplicaciones.

Con la implementación de un sistema web para la gestión de las Juntas administradoras de agua potable y de riego se optimizarán los procesos de seguimiento y reportes sobre la situación actual de las mismas.

1.2.4. DELIMITACIÓN

El análisis comparativo de los servidores de aplicaciones para la plataforma Java EE se limitará al estudio de los servidores de aplicaciones GlassFish y WildFly, los mismos que de acuerdo con el sitio especializado *itbuzzpress*[2], las tendencias de *google trends*[3] y varios programadores java aseguran que actualmente éstos son los servidores más usados en el desarrollo de sistemas web bajo la plataforma Java EE.

Finalizado el análisis comparativo se desarrollará un sistema informático de entorno web bajo la Plataforma Java EE para la Dirección Provincial de la Senagua[4] provincia de Chimborazo, el mismo que se lo utilizará para la automatización de los procesos de gestión y seguimiento de las Juntas Administradoras de agua potable y Juntas administradoras de riego. El sistema a desarrollar contará con los módulos de administración master, administración de juntas de agua potable, juntas de riego, reportes y estadísticas.

1.2.5. FORMULACIÓN DEL PROBLEMA

¿Cómo los servidores de aplicaciones Open Source para la Plataforma Java EE optimizan los recursos del Sistema de Gestión de las Juntas de Agua Potable y Juntas de Riego de la SENAGUA?

[2] http://www.itbuzzpress.com/various/top-5-opensource-application-servers.html
[3] https://www.google.es/trends/explore?date=today 12-m&q=wildfly,glassfish,Apache Geronimo
[4] Secretaría Nacional del Agua - http://www.agua.gob.ec/

1.3. OBJETIVOS

1.3.1. GENERAL

- Realizar el análisis comparativo de los Servidores de aplicaciones Open Source para la Plataforma Java EE enfocadas al desarrollo de sistemas de información web orientados a servicios.

1.3.2. ESPECÍFICOS

- Realizar el diagnóstico de los procesos actuales sobre la gestión de la información relacionada a las Juntas Administradoras de Agua Potable y Juntas de Riego de la SENAGUA Dirección Provincial Chimborazo
- Realizar un estudio comparativo de las funcionalidades tecnológicas de los servidores de aplicaciones Open Source para el desarrollo de sistemas de información bajo la Plataforma Java EE
- Implementar un sistema de información de entorno web para la gestión de las Juntas Administradoras de Agua Potable y Juntas de Riego en la Dirección Provincial de la Secretaría Nacional del Agua (SENAGUA).

1.4. JUSTIFICACIÓN

Actualmente en la Dirección Provincial de la SENAGUA de Chimborazo los funcionarios realizan manualmente la transcripción de los datos de las Juntas Administradoras de Agua Potable y de Riego mediante hojas de cálculo lo cual conlleva mucho esfuerzo y gran cantidad de tiempo al personal encargado de gestionar la información.

El sistema a desarrollar permitirá generar reportes sobre la situación actual de las juntas administradoras de agua potable y de riego, actividad que actualmente toma gran cantidad de tiempo al tener que buscar la información en matrices de Excel para posteriormente crear los respectivos reportes, a través del sistema se optimizará el tiempo al tener dichos reportes de forma digital e impresa.

CAPITULO II

FUNDAMENTACIÓN TEÓRICA

2.1. Java Enterprise Edition

La plataforma Java Enterprise Edition o simplemente Java EE es una plataforma de programación propiedad de Oracle Corporation[5]. Informalmente conocido como Java Empresarial, permite el desarrollo y ejecución de aplicaciones diseñadas en arquitecturas de n capas en un entorno distribuido. Las aplicaciones estarán desarrolladas esencialmente en el lenguaje de programación Java, apoyándose en componentes de software modulares que corren sobre un servidor de aplicaciones. (Pickin y otros, 2013)

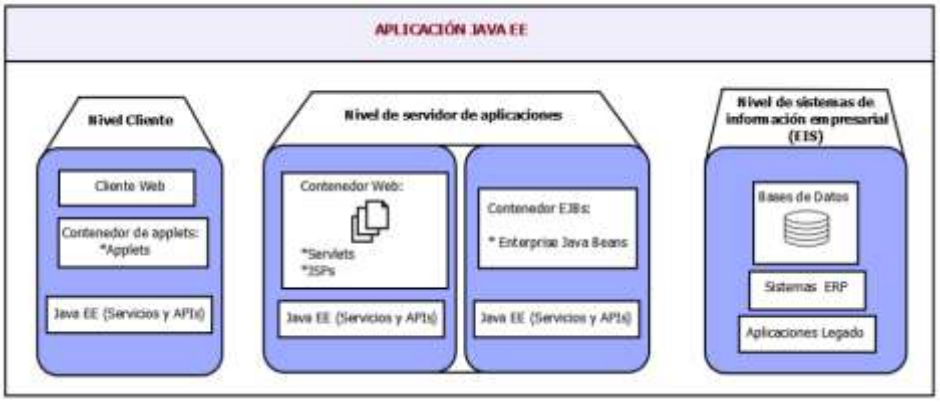


Figura 1. Arquitectura de una aplicación genérica Java EE
Fuente: (Pickin y otros, 2013)

Como se ve en la Figura anterior, los componentes se despliegan dentro de lo que se denominan *contenedores*. Los contenedores son elementos que proporcionan una interfaz entre un componente y la funcionalidad específica de más bajo nivel que soporta ese componente. Antes de que cualquier componente se pueda ejecutar, debe ser ensamblado dentro de un módulo Java EE y desplegado dentro de su contenedor. Los contenedores se encargan de tareas tales como las siguientes:

- Ofrecer un entorno de ejecución para los componentes de la aplicación.
- Gestión de los recursos y del ciclo de vida de los componentes.

[5] *Oracle Corporation*. http://www.oracle.com/index.html

- Proporcionar una vista uniforme de los servicios que requieren los componentes.
- Proporcionar herramientas de despliegue para la instalación y configuración de los componentes.

Según las especificaciones establecidas dentro de "The Java Community Process[6]", la plataforma Java EE es también considerada como un estándar, ya que se puede ver como una colección de especificaciones y directivas de programación que los productos de los proveedores deben cumplir para poder declarar que sus productos son conformes a JavaEE.

2.1.1. Historia de las Versiones

La plataforma era conocida como Java 2 Platform, Enterprise Edition or J2EE hasta la versión 1.4, a partir de la cual se denominó sencillamente Java Platform, Enterprise Edition o Java EE. A lo largo del tiempo, la plataforma fue extendiendo la funcionalidad del modelo estándar Java Platform, Standard Edition (JSE[7]) con el desarrollo de diferentes versiones:

- J2EE 1.2 (12 de Diciembre de 1999)
- J2EE 1.3 (24 de Septiembre de 2001)
- J2EE 1.4 (11 de Noviembre de 2003)
- Java EE 5 (11 de Mayo de 2006)
- Java EE 6 (10 de Diciembre de 2009)
- Java EE 7 (12 de Junio de 2013)
- Java EE 8 (22 de Septiembre de 2014, todavía en prueba la versión final se lanzará en el 2017)

Cada versión extiende la anterior con nuevos API[8] que aportan nuevos servicios y/o funcionalidades.

---

[6] The Java Community Process – JCP, http://www.jcp.org/en/home/index

[7] *Java Standard Edition (JSE)* - http://www.oracle.com/technetwork/java/javase/overview/index.html

[8] API: es un conjunto de subrutinas, funciones y que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción

2.1.2. APIS y Servicios

Java EE se construye sobre la base de Java SE, añadiendo la funcionalidad y proporcionando al desarrollador los servicios necesarios para crear una aplicación de empresa con buenas características de escalabilidad, portabilidad e integración con tecnologías anteriores. Además, el servidor de aplicaciones permite manejar transacciones, seguridad, concurrencia y gestión de los componentes empleados, y de esta manera el desarrollador puede ocuparse más en la lógica de negocio de los componentes que en tareas de mantenimiento de más bajo nivel.

Según la documentación oficial de Oracle[9] los servicios más usuales que se proporcionan y las APIs asociadas a dichos servicios son los siguientes:

- **EJBs (javax.ejb.*):** en esta API se definen las clases e interfaces necesarias para manejar los Enterprise JavaBeans (EJBs). Se especifican las interacciones entre los EJBs y sus clientes, y entre los EJBs y el contenedor. Los EJBs proporcionan servicios de comunicación, concurrencia, transacciones y control de acceso.
- **JSF (javax.faces.*):** *Java Server Faces* proporciona un entorno para simplificar el desarrollo de interfaces de usuario en aplicaciones Java EE.
- **JMS (javax.jms):** el API de *Java Messaging Service* (JMS) proporciona una manera común para que los programas Java creen, envíen, reciban y lean los mensajes de un sistema de comunicación empresarial.
- **JWS (javax.jws.*):** el *Java Web Service* (JWS) proporciona las interfaces y elementos necesarios para que las aplicaciones interactúen con Servicios Web.
- **JPA (javax.persistence.*):** este paquete proporciona las clases e interfaces que modelan la interacción entre proveedores de persistencia, clases administradas y clientes del *Java Persistence API (JPA).* El objetivo de esta API es no perder las ventajas de la orientación a objetos al interactuar con una base de datos (siguiendo el patrón de mapeo objeto-relacional).

[9] *Java EE API Specification* - http://docs.oracle.com/javaee/6/api/overview-summary.html

- **JCA (javax.resource.*):** este paquete define la *Java EE Connector Architecture* (JCA). Esta tecnología permite conectar las aplicaciones en los servidores con los sistemas de información internos de las empresas.
- **Security Service (javax.security.*):** aquí se abordan temas de seguridad como el control de acceso a recursos protegidos, autenticación y autorización, y otros.
- **Servlets y JSPs (javax.servlet.*):** esta especificación define un conjunto de APIs usadas principalmente para tratar peticiones HTTP. En el paquete se incluye también la especificación JSP para aspectos más de presentación.
- **JTA (java.transaction.*):** estos paquetes definen la *Java Transaction API* (JTA). JTA establece una serie de interfaces Java entre el manejador de transacciones y las partes involucradas en el sistema de transacciones distribuidas ([38]): el servidor de aplicaciones, el manejador de recursos y las aplicaciones transaccionales.
- **JAXP (javax.xml.*):** estos paquetes definen el API de *Java API for XML Processing* (JAXP).
- **JNDI (javax.naming.*):** definen la API de *Java Naming and Directory Interface* (JNDI). Permite a los clientes descubrir y buscar componentes y recursos a través de un nombre lógico, independientemente de la aplicación subyacente.
- **JDBC (java.sql y javax.sql):** Los paquetes java.sql y javax.sql definen el API de *Java DataBase Connectivity* (JDBC). Es un API que maneja la conectividad de los programas Java con las bases

2.1.3. Arquitectura de las aplicaciones empresariales con Java

La plataforma de JEE utiliza un modelo de aplicación distribuida multicapas, para las aplicaciones empresariales. La lógica de la aplicación, se divide en varios componentes de acuerdo a su función, y estos componentes se encuentran instalados en distintas máquinas, dependiendo de la magnitud del ambiente de multicapas, usado en la aplicación.

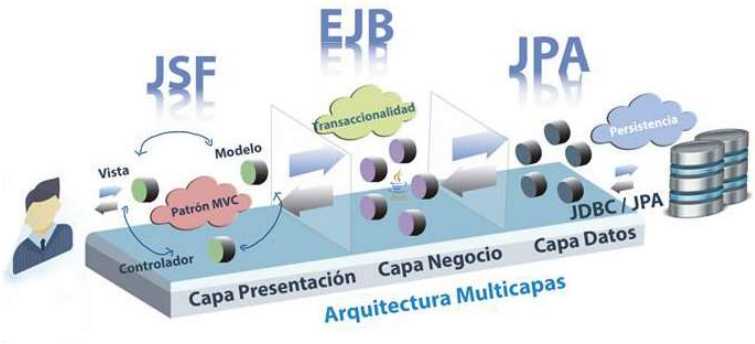


Figura 2. Arquitectura multicapa de la tecnología Java EE.

La figura anterior muestra la arquitectura multicapas JEE, dividida en una serie de partes, descritas a continuación.

2.1.3.1. Capa de Datos

Esta capa consiste en el Nivel De Datos el cual utiliza componentes de Java EE tales como JDBC y JPA, esta capa no debe de contener ninguna lógica de trabajo, y debe de abstraer el mecanismo de persistencia de la capa de servicio.

Es importante saber que el flujo de este tipo de arquitectura es de arriba hacia abajo (del nivel del cliente hacia el nivel de datos), en otras palabras el nivel del cliente puede llamar al nivel de datos pero no al revés.

2.1.3.2. Capa de Negocio

Es toda la parte lógica que resuelve o satisface las necesidades de un dominio de negocio particular, dicha lógica es manejada por Enterprise JavaBeans.

Existen tres tipos de Enterprise java beans: beans de sesión, beans de entidad y beans manejados por mensajes Maneja el software del sistema de información empresarial.

Maneja sistemas de infraestructura empresariales como planificación de recursos empresariales (ERP), procesamiento de transacciones del mainframe, sistemas de base de datos y otros sistemas de información heredados.

2.1.3.3. Capa de Presentación

Como se puede observar la arquitectura de Java EE se encuentra bien definida gracias a Oracle y a la JCP los cuales proveen componentes perfectamente

estandarizados y mediante el uso de estos, nosotros podemos construir aplicaciones empresariales exitosamente. Entonces, ¿Porque existen los frameworks para el desarrollo web en Java?

La respuesta es porque al momento de generar aplicaciones web, necesitaríamos que todos los integrantes del equipo de desarrollo conozcan en un nivel muy avanzado los componentes de la arquitectura, lo cual podría resultar muy difícil de conseguir en los mismos, y otra respuesta podría ser, que en aplicaciones muy grandes el manejo y empleo de los componentes es muy repetitivo y ocasionando redundancia dentro del sistema. Encontrado la solución a estos problemas en los diversos frameworks basado en la JVM (Struts, Spring Web MVC, JSF).

2.1.4. Java Server Faces (JSF)

En la actualidad para el desarrollo de aplicaciones de negocio se utiliza frecuentemente el patrón de diseño MVC Modelo Vista Controlador (Model View Controller) que además es sencillo de implementar en las aplicaciones web. En este patrón el modelo es modificable por las funciones de negocio. Estas funciones son solicitadas por el usuario mediante el uso de un conjunto de vistas de la aplicación que solicitan dichas funciones de negocio a través de un controlador, que es el módulo que recibe las peticiones de las vistas y las procesa.

La creación de aplicaciones basadas en el patrón MVC se ve facilitada por el uso de marcos de trabajo (frameworks). Un marco de trabajo es un conjunto de APIs y módulos normalmente acompañados de la documentación y guía de uso que definen la manera de implementar alguna de las capas de nuestra aplicación. Lo podemos ver también como la estructura o cimientos sobre los cuales se crear una aplicación.

2.1.4.1. Características principales de JSF

JSF utiliza las páginas Facelets como vista, objetos JavaBean como modelos y métodos de esos objetos como controladores. El servlet FacesServlet realiza toda la tediosa tarea de procesar las peticiones HTTP, obtener los datos de entrada, validarlos y convertirlos, colocarlos en los objetos del modelo, invocar las acciones del controlador y renderizar la respuesta utilizando el árbol de componentes.

Entrando más a fondo JSF proporciona las siguientes características destacables:

- Definición de las interfaces de usuario mediante **vistas** que agrupan **componentes** gráficos.
- Conexión de los componentes gráficos con los datos de la aplicación mediante los denominados **beans gestionados**.
- Conversión de datos y validación automática de la entrada del usuario.
- Navegación entre vistas.
- Internacionalización
- A partir de la especificación 2.0 un modelo estándar de comunicación Ajax entre la vista y el servidor.

Tal y como se detalla anteriormente, JSF se ejecuta sobre la tecnología de Servlets y no requiere ningún servicio adicional, por lo que para ejecutar aplicaciones JSF sólo se necesita un contenedor de aplicaciones (servidor de aplicaciones).

Para entender el funcionamiento de JSF es interesante compararlo con JSP, cabe recalcar que una página JSP contiene código HTML con etiquetas especiales y código Java. La página se procesa en una pasada de arriba a abajo y se convierte en un servlet, posteriormente los elementos JSP se procesan en el orden en que aparecen y se transforman en código Java que se incluye en el servlet. Una vez realizada la conversión, las peticiones de los usuarios a la página provocan la ejecución del servlet. (Mann, 2005)

En JSF el funcionamiento es distinto, una página JSF también contiene etiquetas especiales y código HTML, pero su procesamiento es mucho más complicado. La diferencia fundamental con JSP es el resultado del procesamiento interno, en el servidor, de la página cuando se realiza la petición. En JSP la página se procesa y se transforma en un servlet. En JSF, sin embargo, el resultado del procesamiento es un árbol de componentes, objetos Java instanciados el servidor, que son los que posteriormente se encargan de generar el HTML.

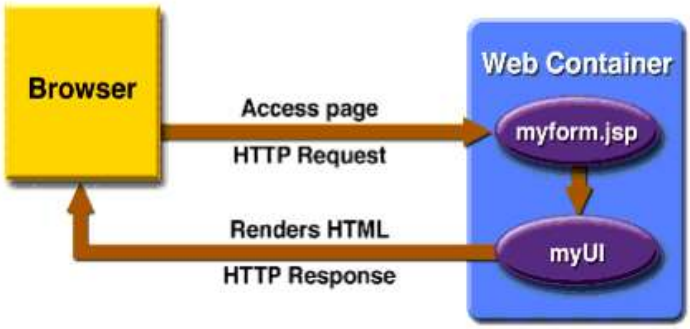


Figura 3. Funcionamiento de las páginas JSF.
Fuente: Dept. Ciencia de la Computación e IA, Universidad de Alicante

Con un poco más de detalle, cuando el usuario realiza una petición a la página JSF se realizan las siguientes acciones en orden:

[1] Se procesa la página de arriba abajo y se crea un **árbol de componentes** JSF en forma de objetos instanciados de clases del framework JSF.

[2] Se obtienen los valores introducidos por el usuario y se actualizan los beans gestionados con ellos.

[3] Se actualizan los componentes con los valores procedentes de las propiedades de los beans gestionados.

[4] Se pide a los componentes que se rendericen, generándose el código HTML que se envía de vuelta al navegador como resultado de la petición.

[5] El árbol de componentes JSF se guarda en memoria para que posteriores peticiones a la misma página JSF no tengan que crearlo, sino que utilicen el existente.

Una ventaja del enfoque de JSF es que el renderizado de la interfaz de usuario resultante es más flexible. De hecho, es posible generar con el mismo JSF distinto código para distintos dispositivos. Por ejemplo, es posible utilizar la misma aplicación JSF para servir páginas a navegadores web y dispositivos móviles.

2.1.4.2. Ciclo de Vida de Java Server Faces (JSF)

El ciclo de vida de una página JSF tiene 6 fases, como se puede ver en la siguiente figura:

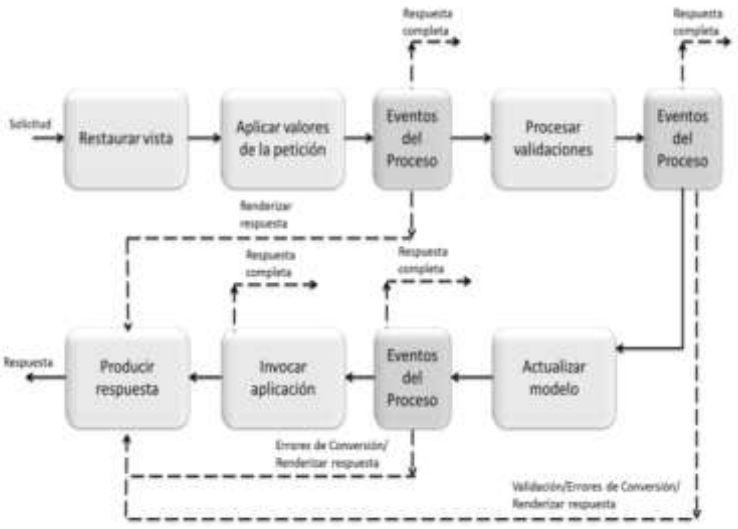


Figura 4. Ciclo de vida de JSF
Fuente: Java Server Faces Comunity (2016)

1) **Restaurar Vista:** es la primera fase, se lleva a cabo cuando se hace una petición. También se la suele conocer como Fase de Construcción del Árbol de Componentes. El objetivo fundamental de esta fase es la creación de un árbol con todos los componentes de la página.
2) **Aplicar Valores de la Petición:** cada uno de los componentes del árbol creado en la fase de restauración de la vista obtienen el valor que les corresponde de la petición realizada y lo almacenan.
3) **Procesar Validaciones:** después de almacenar los valores de cada componente, estos son validados según las reglas que se hayan declarado.
4) **Actualizar modelo:** durante esta fase los valores locales de los componentes son utilizados para actualizar los beans que están ligados a dichos componentes.
5) **Invocar aplicación** se ejecuta la acción u operación correspondiente al evento inicial que dio comienzo a todo el proceso.
6) **Producir respuesta:** la respuesta se renderiza y se regresa al cliente

2.1.4.3. Componentes de una aplicación JSF

Todas las aplicaciones JSF están formadas por un conjunto específico de archivos de configuración, los controladores beans (son las clases que actuarán como controladores de las acciones) y los archivos de contenido Web (HTML, CSS, etc).

Entre los archivos de configuración más importantes que se requieren son:

- **faces-config.xml:** el archivo de configuración maestro requerido por todas las aplicaciones JSF, contiene una referencia para todas las partes de una aplicación.
- **web.xml:** es el descriptor de despliegue J2EE Web y el archivo maestro de configuración requerida por cualquier aplicación web J2EE.

Figura 5. Principales componentes de una aplicación JSF
Fuente: JSF in action. (Mann, 2005)

2.1.4.4. Implementaciones de Java Server Faces

JSF es una especificación y, como tal, existen distintas implementaciones, Oracle siempre proporciona una implementación de referencia de las tecnologías Java, que incluye en el servidor de aplicaciones GlassFish. En el caso de JSF, la implementación de referencia es las dos implementaciones más usadas son:

- **Mojarra** es la especificación de referencia realizada por Oracle, es una de las implementaciones más populares y se incluye en distintos servidores de aplicaciones Java Enterprise, entre los que se encuentran GlassFish y JBoss.
- **MyFaces** es la implementación abierta de la Fundación Apache. Está desarrollada por la comunidad Apache y está incluida en el servidor de aplicaciones Gerónimo.

Las implementaciones de la especificación JSF proporcionan el framework y los componentes básicos (etiquetas h: y f:), sin embargo para utilizar JSF en una aplicación más avanzada se necesitan componentes más elaborados.

En el sitio web http://jsfcentral.com/ se encuentra una tabla resumen muy útil en la que se comparan las características de distintos conjuntos de componentes para JSF (más de 20). Dentro de los frameworks para JSF, se encuentran se destacan algunas alternativas como por ejemplo: Primefaces, RichFaces y ICEfaces. Cada una de estas proporciona elementos o componentes que facilitan el desarrollo de aplicaciones:

- **PrimeFaces** que es una librería de componentes visuales open source con soporte ajax y es desarrollada y mantenida por Prime Technology.
- **RichFaces** es una biblioteca de código abierto basada en Java que permite crear aplicaciones web con ajax, mantenida y desarrollada por JBoss.
- **ICEFaces** es un Framework basado en ajax que permite desarrollar Aplicaciones RIA de forma más fácil y rápida, sigue una serie de estándares que permiten trabajar en un ambiente normal de desarrollo con Java.

En el presente estudio se seleccionó a PrimeFaces como implementación de JSF para el desarrollo de aplicaciones Java EE.

2.1.4.5. PrimeFaces

Figura 6. Logotipo de PrimeFaces
Fuente: PrimeTek (2013). Primefaces.org

PrimeFaces es una Dramework de componentes visuales enriquecidos de código abierto para Java Server Faces (JSF), que facilita la creación de aplicaciones orientadas a la web. Fue desarrollado por Prime Technology compañía especializada en consultoría, JSF, JEE y Outsourcing.

PrimeFaces fue desarrollado bajo la licencia de Apache License V2 en Turquía, Tiene gran aceptación en el mercado ya que brinda componentes agradables a la vista y con un tiempo de respuesta mínimo en comparación con otros Frameworks JSF. Cuenta con un sin número de características entre las que se pueden destacar su alto soporte nativo de Ajax, adicionalmente cuenta con un kit para el desarrollo de aplicaciones móviles y es compatible con otros frameworks como RichFaces.

Las principales características de Primefaces son:

- Soporte nativo de Ajax, incluyendo Push/Comet.
- Kit para crear aplicaciones web para móviles.
- Compatibilidad con otras librerías de componentes, como JBoss RichFaces.
- Uso de javascript no intrusivo (no aparece en línea dentro de los elementos, sino dentro de un bloque <script>).
- Es un proyecto open source, activo y bastante estable entre versiones.

Las principales ventajas que ofrece PrimeFaces son:

- Los componentes que ofrece son amigables al usuario ya que cuentan con un diseño innovador mejorando la experiencia de los mismos.
- En comparación con otros frameworks JSF, PrimeFaces cuenta con más de 100 componentes, algunos de muy alta complejidad como el Dock, y otros muy sencillos como botones.

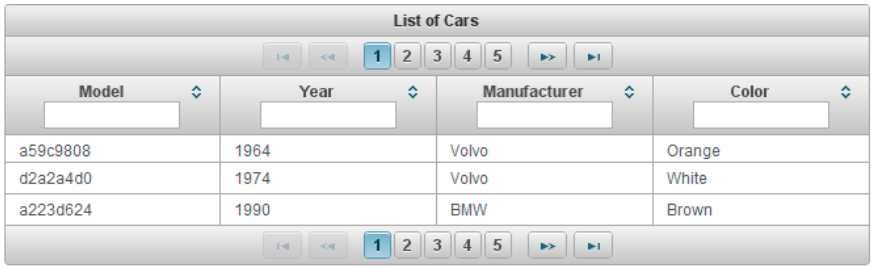


Figura 7. Ejemplo del componente <p:datatable>
Fuente: PrimeTek (2013). Primefaces.org

2.1.5. Tecnología Enterprise Java Beans (EJB)

2.1.6. Manejo de Persistencia en Java

2.2. Servidores de Aplicaciones

El concepto de servidor de aplicaciones está relacionado con el concepto de sistemas distribuidos. Un sistema distribuido, en oposición monolítica, permite mejorar tres aspectos fundamentales en una aplicación, la alta disponibilidad, la escalabilidad y el mantenimiento. En un sistema monolítico un cambio en las necesidades del sistema provoca un colapso y la adaptación a dicho cambio puede resultar catastrófica. A continuación se exponen las principales características de los servidores de aplicaciones:

- La **alta disponibilidad** hace referencia a que un sistema debe estar funcionando las 24 horas del día los 365 días al año. Para poder alcanzar esta característica es necesario el uso de técnicas de balanceo de carga y de recuperación ante fallos (*failover*).

- La **escalabilidad** es la capacidad de hacer crecer un sistema cuando se incrementa la carga de trabajo (el número de peticiones). Cada máquina tiene una capacidad finita de recursos y por lo tanto sólo puede servir un número limitado de peticiones. Si, por ejemplo, tenemos una tienda que incrementa la demanda de servicio, debemos ser capaces de incorporar nuevas máquinas para dar servicio.

- El **mantenimiento** tiene que ver con la versatilidad a la hora de actualizar, depurar fallos y mantener un sistema. La solución al mantenimiento es la construcción de la lógica de negocio en unidades reusables y modulares.

2.2.1. Servidor de aplicaciones para la plataforma Java EE

El estándar de la plataforma Java EE 7 permite el desarrollo de aplicaciones de empresas de una manera sencilla y eficiencia. Una aplicación desarrollada con la tecnología Java EE, permite ser desplegadas en cualquier servidor de aplicaciones

o servidor web que cumpla con el estándar. Un servidor de aplicaciones es una implementación de la especificación de la plataforma Java EE.

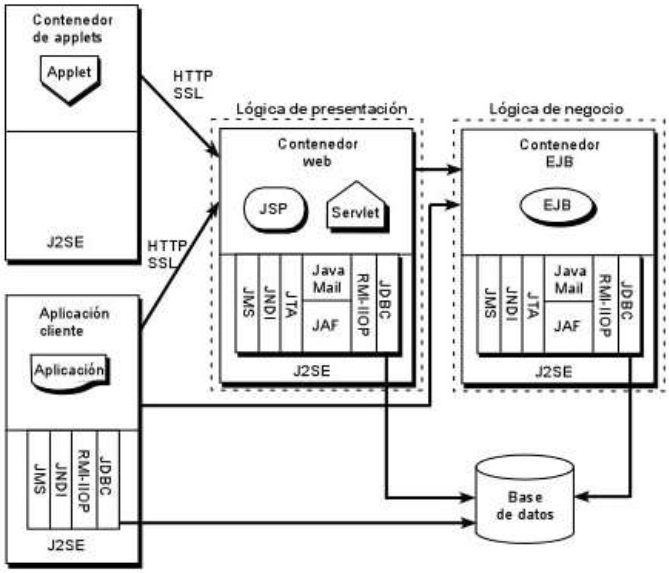


Figura 8. Arquitectura Java EE
Fuente: http://www.jtech.ua.es/j2ee/2003-2004/abierto-j2ee-2003-2004/sa/sesion1-apuntes.htm

Definimos a continuación algunos de los conceptos que aparecen en la figura anterior:

- **Cliente web (contenedor de applets)**: Es usualmente un navegador e interactúa con el contenedor web haciendo uso de HTTP. Recibe páginas HTML o XML y puede ejecutar applets y código JavaScript.
- **Aplicación cliente**: Son clientes que no se ejecutan dentro de un navegador y pueden utilizar cualquier tecnología para comunicarse con el contenedor web o directamente con la base de datos.
- **Contenedor web**: Es lo que comúnmente denominamos servidor web. Es la parte *visible* del servidor de aplicaciones. Utiliza los protocolos HTTP y SSL (seguro) para comunicarse.
- **Servidor de aplicaciones**: Proporciona servicios que soportan la ejecución y disponibilidad de las aplicaciones desplegadas. Es el corazón de un gran sistema distribuido.

Frente a la tradicional estructura en dos capas de un servidor web (Figura siguiente) un servidor de aplicaciones proporciona una estructura en tres capas que permite estructurar nuestro sistema de forma más eficiente. Un concepto que debe quedar

claro desde el principio es que no todas las aplicaciones de empresa necesitan un servidor de aplicaciones para funcionar. Una pequeña aplicación que acceda a una base de datos no muy compleja y que no sea distribuida probablemente no necesitará un servidor de aplicaciones, tan solo con un servidor web (usando servlets y jsp) sea suficiente.

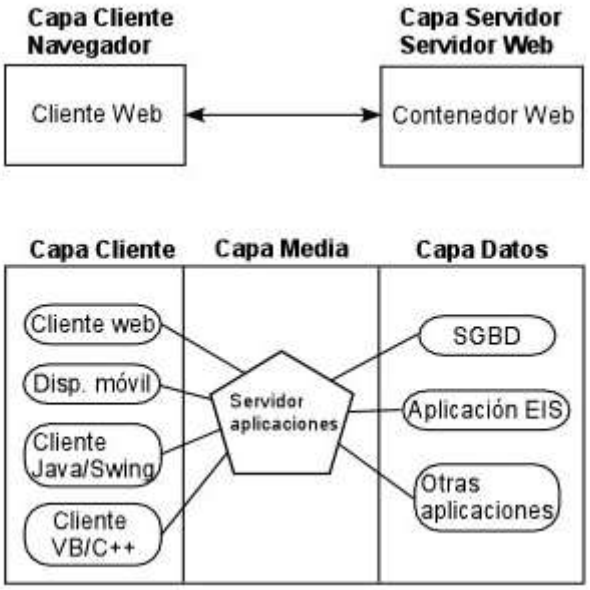


Figura 9. Arquitectura en dos capas frente a tres capas utilizando el servidor de aplicaciones.
Fuente: http://www.jtech.ua.es/

Como hemos comentado, un servidor de aplicaciones es una implementación de la especificación J2EE. Existen diversas implementaciones, cada una con sus propias características que la pueden hacer más atractiva en el desarrollo de un determinado sistema. Dentro del mundo open source se destacan WildFly, GlassFish, Apache Tomcat como los servidores más empleados en la implementación de sistemas web. En las siguientes secciones se estudia a fondo los servidores WildFly y GlassFish.

2.2.2. Servidor de Aplicaciones WildFly

Figura 10. Logo del servidor de aplicaciones WildFly
Fuente: http://wildfly.org/

WildFly, anteriormente conocido como JBoss AS, o simplemente JBoss, es un servidor de aplicaciones Java EE de código abierto implementado en Java puro, más concretamente la especificación Java EE. Al estar basado en Java, JBoss puede ser utilizado en cualquier sistema operativo para el que esté disponible la máquina virtual de Java. JBoss Inc., empresa fundada por Marc Fleury y que desarrolló inicialmente JBoss, fue adquirida por Red Hat en abril del 2006. En febrero de 2007, Marc Fleury deja Red Hat.

WildFly es software libre y de código abierto, sujeto a los requisitos de la GNU Lesser General Public License (LGPL), version 2.1. El proyecto se nutre de una red mundial de colaboradores, los ingresos de la empresa están basados en un modelo de negocio de servicios. JBoss implementa todo el paquete de servicios de J2EE.

El 20 de noviembre de 2014, JBoss Application Server se renombra WildFly. La JBoss Community y otros productos JBoss de Red Hat como JBoss Enterprise Application Platform no se renombran. Pese al cambio, JBoss sigue siendo en 2016 el término más usado para referirse al producto, tanto en términos de trabajo como en la web, debido a esto.

Desde entonces la URL http://www.jboss.org/ sirve JBossDeveloper, el portal para desarrolladores de JBoss/WildFly, pasando http://wildfly.org/ a ser la web oficial del producto.

2.2.2.1. Historia

JBoss AS es el primer servidor de aplicaciones de código abierto, preparado para la producción y certificado J2EE 1.4, disponible en el mercado, ofreciendo una plataforma de alto rendimiento para aplicaciones de e-business. Combinando una arquitectura orientada a servicios SOA, con una licencia GNU de código abierto, JBoss AS puede ser descargado, utilizado, incrustado y distribuido sin restricciones por la licencia.

Las características destacadas de JBoss incluyen:

- Producto de licencia de código abierto sin coste adicional.

- Cumple los estándares.
- Confiable a nivel de empresa
- Incrustable, orientado a arquitectura de servicios.
- Flexibilidad consistente
- Servicios del middleware para cualquier objeto de Java.
- Soporte completo para JMX.

2.2.2.2. Servicios proporcionados por WildFly

El sitio web del servidor proporciona las listas de todas las especificaciones y servicios proporcionados por el servidor, entre las cuales se destacan las siguientes:

- **JSF (2.2):** para desarrollar aplicaciones web sucesor de los JSP.
- **Servlets (3.1) y JSPs (2.3):** los servlets son la base a partir de la cual desarrollar aplicaciones web dinámicas y los JSP una forma de servlet en el que la mayor parte del código HTML, similar a PHP.
- **CDI (1.0):** proporciona inyección de dependencias de forma parecida a frameworks como Spring.
- **EJB (3.2):** beans gestionados por un contenedor administrando su ciclo de vida y proporcionales funcionalidades como persistencia y transacciones. Suelen usarse para incluir la lógica de negocio de la aplicación.
- **Bean Validation (1.1):** funcionalidad que mediante anotaciones permite indicar restricciones sobre los valores que pueden contener los beans.
- **JPA (2.1):** especificación que proporciona persistencia en una base de datos.
- **JTA (1.2):** especificación que proporciona transaccionalidad.
- **JMS (2.0):** especificación que permite a las aplicaciones comunicarse mediante mensajes de forma desacoplada.
- **JAX-RS (2.0):** especificación sobre los servicios web basados en el modelo REST sobre el protocolo HTTP.
- **JAX-WS (1.3):** especificación sobre servicios web basados en XML.
- **JavaMail (1.5):** especificación para el envío de mensajes de correo electrónico.

2.2.2.3. Ventajas de WildFly

Las ventajas de WildFly son múltiples.

- El producto está siendo constantemente actualizado y cuenta con buena documentación.
- Producto de licencia de código abierto.
- Cumple los estándares especificados en el estándar Java EE.
- Confiable a nivel de empresa.
- Orientado a arquitectura de servicios.
- Soporte completo para Java Management eXtensions.
- Ayuda profesional 24 horas.

2.2.3. Servidor de Aplicaciones GlassFish

2.2.3.1. ¿Qué es GlassFish?

El término GlassFish, traducido al español sería algo parecido como "Pez de Cristal", es el nombre de un pez que realmente existe y vive en el agua dulce; su cuerpo es transparente, por lo que sus huesos son visibles. El nombre fue elegido debido a la transparencia que los creadores querían darle al proyecto, que utiliza una licencia Open Source, concretamente la licencia Common Development and Distribution License (CDDL) v1.0 y la GNU Public License (GPL) v2.

2.2.3.2. Para que sirve GlassFish

GlassFish es un servidor de aplicaciones desarrollado por Oracle que implementa las tecnologías definidas en la plataforma Java EE y permite ejecutar aplicaciones que siguen esta especificación. La versión comercial es denominada Sun GlassFish Enterprise Server. Soporta las últimas versiones de tecnologías como: JSP, Servlets, EJBs, Java API para Servicios Web (JAX-WS), Arquitectura Java para Enlaces XML (JAXB), Metadatos de Servicios Web para la Plataforma Java 1.0, y muchas otras tecnologías.

GlassFish además de ser un servidor de aplicaciones, es una comunidad de usuarios, que descargan y utilizan libremente Glassfish, también existen partners que

contribuyen agregándole más características importantes a Glassfish. Además ingenieros y beta testers que desarrollan código y prueban las versiones liberadas para eliminar todo fallo que se encuentre, y muchos otros miembros. La comunidad fue lanzada en el año 2005 en java.net. Al igual que el pez original, la Comunidad GlassFish es transparente en cuanto a términos de entrega de código fuente, discusiones de ingenie ría, agendas, datos de descarga, etc. Tú puedes tener acceso a todo esto, además puedes formar parte de todo el proceso detrás de la comunidad GlassFish. Un ejemplo de esto es la comunidad FishCAT.

2.2.3.3. Cómo funciona un servidor de aplicaciones

Un servidor de aplicaciones proporciona generalmente gran cantidad de funcionalidades built in de forma transparente al usuario de manera que no sea necesario escribir código fuente. Estas funcionalidades son posibles ya que los componentes se ejecutan dentro del contenedor en un espacio de ejecución virtual llamado dominio de ejecución. Su función principal es la de interponerse entre las llamadas que se hacen a los métodos de los beans y las implementaciones de los mismos, de modo que entre otras cosas puede hacer las comprobaciones para verificar si el usuario que llama al método tiene los permisos adecuados, antes de llamarlo.

2.2.3.4. Modular, Integrable y Extendible

Glassfish dispone de una arquitectura Modular, se puede descargar e instalar solamente los módulos que se necesiten para las apps, con lo cual se minimiza el tiempo de inicio, consumo de memoria y espacio en disco.

Basándose en el modelo de componentes dinámico y completo para Java OSGi (Open Services Gateway Initiative), las aplicaciones y/o componentes de Glassfish pueden ser remotamente instalados, iniciados, actualizados, etc. sin necesidad de reiniciar el servidor.

Es posible ejecutar Glassfish dentro de una máquina virtual sin necesidad de disponer de instalar un servidor de aplicaciones. Es posible usar Glassfish como una librería más en la JVM, seleccionando solo lo que se necesita y probando pequeñas aplicaciones webs sin necesidad de correr todo el AppServer, teniendo en cuenta las limitaciones de no tener el AppServer instalado.

2.2.3.5. Herramientas de programación

- **AJAX.** Glassfish dispone de una tecnología y framework para Java basadas en web (Java Server Faces) llamado Woodstock, que simplifica el desarrollo de interfaces de usuario en aplicaciones J2EE en el cual se pueden incluir componentes AJAX.
- **Ruby on Rails.** Se pueden ejecutar aplicaciones basadas en Ruby Rails de dos formas diferentes. La primera es mediante jRuby que está incluido en la Java Platform y la segunda sería ejecutar Rails en un intérprete nativo de Ruby comunicándose con Gassfish mediante CGI.
- **PHP**. Puede utilizarse PHP con la implementación Quercus PHP 5 desenvolupada por Caucho en Java.

2.2.3.6. Tecnologías de Integración

- **TopLink Essentials.** Es la implementación de JPA (Java Persistence API) para la comunidad Glassfish. La API se proporciona un modelo de programación sencillo para las entidades persistencia de EJB y además incluye la herramienta para conectar diferentes proveedores de persistencia.
- **CORBA**. Glassfish incluye una implementación completa de CORBA. Esta aplicación ha ido mejorando con las diferentes versiones de Glassfish.
- **OpenMQ Messaging.** Glassfish incorpora una herramienta de mensajería que proporciona:
  - Mensajes entre los componentes del sistema
  - Distribución escalable de servidores de mensajería
  - Integración de mensajes SOAP / HTTP
  - Java y C Cliente API
- **Java Business Integration.** Glassfish incluye soporte para la API JBI. Se encarga de la integración de bus y componentes de arquitectura. La implementación incluida en Glassfish proviene del proyecto OpenESB.

2.3. Benchmarking

Después de haber considerado las características y servicios que brindan los servidores de aplicaciones GLASSFISH y JBOSS se debe tomar en cuenta que el

rendimiento es clave en la comparación de los servidores de aplicaciones. Por lo tanto la implementación de un benchmark que mida el desempeño de los servidores de aplicaciones cuando una aplicación se encuentre en tiempo de ejecución nos dará una clara visión de cuál de estos servidores de aplicaciones se adapten mejor a los requerimientos de los desarrolladores de aplicaciones.

2.3.1. BENCHMARK

Según Camp la referenciación conocida como "Benchmark", llevado al campo de los sistemas computacionales, es un proceso usado para comparar diferentes sistemas informáticos o componentes de un sistema basándose en cual tiende a ofrecer el mejor rendimiento o el acceso a más recursos como por ejemplo bases de datos o sistemas de ficheros por parte de los usuarios, eso sí bajo un mismo entorno de ejecución

Para poder garantizar los resultados obtenidos. Dicho proceso se deben efectuar siguiendo un estándar genérico considerado por Robert C. Camp que contiene las siguientes etapas:

a) **Etapa de Planeación.-** El objetivo de esta etapa es planear las investigaciones de Benchmarking respondiendo a cuestiones como son: quien, qué y cómo.
b) **Etapa de Elaboración de Benchmark.-** Ejecutada la primera etapa se lleva a cabo la realización del sistema que nos permita evaluar las herramientas que vamos a medir.
c) **Etapa de Medición de Parámetros y Generación de Resultados.-** Esta etapa requiere de los datos obtenidos de la implementación del sistema Benchmark desarrollado en la segunda etapa, los cuales son organizados según poscriterios de la planeación.
d) **Etapa de Análisis de Resultados.-** Esta etapa comprende la interpretación de los datos organizados los cuales determinan las brechas de desempeño de cada herramienta analizada.

e) **Etapa de Conclusiones y Recomendaciones.-** Una vez analizados los datos se tiene una clara idea del desempeño de las herramientas, lo cual permite recomendar una elección adecuada y fundamentada entre ellas.

CAPITULO III

ANÁLISIS COMPARATIVO DE LOS SERVIDORES DE APLICACIONES OPEN SOURCE PARA EL DESARROLLO DE APLICACIONES JAVA EE

Un servidor de aplicaciones como menciona Lindgren (2001) constituyen una herramienta basada en componentes que residen en la capa intermedia de las tres capas de la arquitectura, que permite el construir, desplegar y administrar nuevas aplicaciones a los usuarios, de una forma más ordenada debido a que contienen la lógica propia del negocio, separando la presentación de la operación, lo que hace más flexible la construcción de aplicaciones, y escalable ya que la lógica de negocio, se encuentra en un solo lugar (servidor) y es la misma para todas las presentaciones.

Los servidores de aplicación son la parte integral de arquitectura de tres capas lo que permite que estos se sincronicen con el servidor de Web para procesar la petición hecha por el cliente. La comunicación entre el servidor de aplicaciones y el dispositivo se lo puede realizar ya sea utilizando un servidor de Web que a su vez se comunica con el servidor de aplicaciones, sea este el caso el dispositivo del usuario final se apoya en base a los varios protocolos que soportan los servidores Web y navegadores Web, y en otros casos la conexión puede ser directa entre el servidor y el dispositivo dependiendo sobre cuál de las tecnologías son apoyadas por el servidor de aplicaciones, estos dispositivos podrían estar ejecutando Java Applets o aplicaciones, ActiveX, programas que se comunican usando el protocolo basado en Corba o programas que utilizan un protocolo propietario sobre TCP/IP.

GlassFish y WildFly son servidores de aplicaciones desarrollados bajo el lenguaje JAVA, y pueden correr en plataforma J2EE, ofrecen fuente abierta para los usuarios y desarrolladores, y se distribuyen de manera libre, se mantienen en la misma línea de administración de aplicaciones Web. Estos servidores proveen contenedores para los EJB, y dan la infraestructura para las extensiones de administración (JMX). A demás componentes esenciales para la mensajería en JMS, transacciones JTA/JST, persistencia CMP, seguridad JAAS, conectividad JCA. Además presentan características como modularidad, acoplablilidad. (Tomich, 2008)

WildFly (antes Jboss AS) es un servidor de aplicaciones J2EE de código abierto implementado en Java. Ofrece una plataforma de alto rendimiento para aplicaciones de e-business. Sus características principales son:

- ✓ Es distribuido bajo licencia de código abierto GPL/LGPL.
- ✓ Proporciona un nivel de confianza suficiente para ser utilizado en entornos empresariales.
- ✓ Es un servicio incrustable, por ello está orientado a la arquitectura en servicios.
- ✓ Servicio del middleware para objetos Java.
- ✓ Soporte completo para JMX (Java Management eXtensions).

GlassFish es un servidor de aplicaciones de código abierto que implementa funcionalidades de Java EE. Es gratuito y de código abierto, desarrollado por Sun Microsystems. Tiene como base al servidor Sun Java Application Server de Oracle Corporation, un derivado de Apache Tomcat.

Tanto **JBoss** como **GlassFish** son servidores de aplicación Java EE. Java EE provee estándares que permiten a un servidor de aplicaciones servir como contenedor de los componentes que conforman dichas aplicaciones. Entre los componentes podemos encontrar:

- ✓ Servlets.
- ✓ JavaServer Pages (JSP).
- ✓ JavaServer Faces (JSF).
- ✓ Enterprise JavaBeans (EJBs).
- ✓ Java API for Web Services (JAX-WS).

Estos componentes permiten implementar diferentes capas de la aplicación, como la interfaz de usuario, la lógica de negocio, la gestión de sesiones, etc.

3.1. Análisis de las Características

Una vez descritas las arquitecturas que forman la estructura de cada uno de los servidores de aplicaciones GLASSFISH y JBOSS se procederá a realizar un cuadro

comparativo en la cual se tratará de mostrar las diferentes características que cada uno de estos servidores poseen, las cuales son tomadas del sitio especializado "serverwatch" en la sección métricas para el análisis de servidores[10].

Tabla 1. Principales características de GlassFish y WildFly.

		WildFly	GlassFish	Observación
Características generales	Tipo de servidor	Servidor de Aplicaciones	Servidor de Aplicaciones	
	Versión	10.1	4.1.1	
	Fecha de lanzamiento (última versión)	19 – 08 – 2016	0ctubre – 2015	
	Patrocinador	Red Hat[11]	Oracle[12]	
	Modo de Distribución	Libre	Libre	Se distribuyen de forma gratuita en la página web oficial.
	Software Libre/Open Source	Si	Si	Contienen la fuente de software la cual es configurable (a medida).
	Licencia	LGPL 2.1	CDDL y GPL.	
Administración	Configuración GUI	Si	Si	Los dos servidores poseen GUIs estáticos.
	Administración Remota	No	Si	GLASSFISH implementa varios tipos de usuarios uno de ellos el cliente administrador.
	Protocolo SNMP configurable	No	No	El protocolo SNMP tiene funcionalidad externa respecto a los servidores de aplicaciones.
Escalabilidad	Compatible .net	No	No	Ninguno de los servidores es compatible con .Net, debido a que están basados en la arquitectura Java EE.
	Puerto 64 Bits	Si	Si	Dependiendo de la extensión de la aplicación se pueden ocupar o no los 64 bits de mapeo de la JVM.
	Soporte Clutser	Si	Si	Permiten que sea más tolerante a fallos y mantenga alta disponibilidad.
	Soporte IPv6	No	No	No está establecido ya que el protocolo básico de comunicación se mantiene en IPV4.
	Compatible Java EE	Si	Si	Basados en la arquitectura J2EE 1.4 o superior.
Seguridad	Adaptabilidad de Firewall	No	Si	Dentro de la arquitectura de GLASSFISH está incluido el manejo de firewall para el ingreso de usuarios.

[10]Web Server Benchmarks/WebServer Compare: http://www.serverwatch.com/tutorials/article.php/1363241/Web-Server-BenchmarksWebServer-Compare.htm

[11] Red Hat Inc. - https://www.redhat.com/es

[12] Oracle Corporation - https://www.oracle.com/es/index.html

	Autentificación LDAp	Si	Si	Ambos poseen métodos de autenticación para brindar seguridad en la administración utilizando el servicio de nombres propios de cada servidor.
	Software (SSL)	Si	Si	Utilizan SSL para el transporte seguro de datos a través de la encriptación de los mismos garantizando una conexión segura.
Soporte	Soporte comercial disponible	Si	Si	Ambos cuentan con los mejores servicios contratados para soporte disponible así como soporte gratuito basado en foros.
	Soporte a lista de Correo	Si	Si	De acuerdo a especificaciones de soporte solicitados por los usuarios estos son organizados mediante listas de correos que solventan problemas eventuales.
	Soporte Java Server Faces	Si	Si	Para manejar de una forma estandarizada el desarrollo de interfaces con el uso de JSP, ambos utilizan el Framework JSF que administra y ayuda a realizarlas.
	Soporte Plug-In personalizado	Si	Si	Los dos servidores utilizan la característica de plugin para brindar servicios extras de acuerdo al ambiente de prestaciones que la aplicación desarrollada brinde a los usuarios.
	Soporte Hibernate	Si	Si	Otra característica soportada para la seguridad de la lógica del negocio es implementada por estos servidores con el uso de Hibernate como alternativa
	Soporte Jboss Seam	Si	No	Por medio de este Framework ambos servidores están en la capacidad de unir varias tecnologías y estándares que brinden nuevas y distintas funcionalidades para desarrollar una aplicación Web con características que brinden en otros Frameworks como JSF y EJB.
	Soporte de conexión Eclipse IDE	Si	Si	A través de plugins los servidores se pueden conectar al IDE ECLIPSE lo que facilita su configuración por medio de una interfaz brindando a algunos desarrolladores un manejo más fácil lo que con lleva el omitir el uso de líneas de comando para ejecución de tareas.
	Más de una instancia por nodo	No	Si	WildFly permite la creación de más instancias pero únicamente una instancia por nodo, por lo que

				son únicamente instancias físicas, mientras GlassFish permite a su vez creación de más instancias lógicas dentro de un mismo nodo
Otras características	Múltiples Registros	Si	Si	Permiten registrar eventos de la aplicación.
	Servidores Virtuales	Si	Si	Ofrece la capacidad de mostrar al usuario una instancia del servidor que simula al servidor de aplicaciones.
	Capacidad JSP y Servlets	Si	Si	Ambas utilizan para el desarrollo de aplicaciones Web las últimas versiones de JSP y Servlets.
	Capacidad EJB	Si	Si	Ambos utilizan la última versión del contenedor para el manejo de lo Enterprise Java Beans.
	JAX-WS/JAX-B	Si	Si	Es la más actualizada arquitectura de manejo de código que enlaza XML con componentes estándar J2EE.

Fuente: Autores

Una vez contempladas las características técnicas entre los servidores de aplicaciones GLASSFISH y JBOSS se puede llegar a determinar de forma general que ambos servidores están diseñados para brindar al desarrollador similares servicios como los servicios de nombres, mensajería, seguridad, manejo de EJBs, etc., sin embargo la estructura de la arquitectura de cada uno de estos difiere conceptualmente en la gestión de esos servicios y de usuarios permitiendo finalmente que GLASSFISH sea un servidor de aplicaciones veloz y que JBOSS sea un servidor de aplicaciones más robusto.

Cabe recalcar que la versión considerada para el estudio de GLASSFISH maneja la tecnología J2EE y además la tecnología JEE7 que es una versión evolutiva de J2EE. La versión considerada por JBOSS para su estudio, en cambio está basada completamente en la tecnología J2EE.

3.2. Análisis comparativo mediante el uso de Benchmark

3.2.1. Determinación de Herramienta Benchmark y parámetros de evaluación

Para la evaluación del rendimiento de los servidores se realizará el proceso de benchmarking a cada uno de los servidores previamente instalados y configurados, con la finalidad de obtener una evaluación sobre cómo se comportan

bajo diferentes cargas de trabajo. El software con el cual se van a realizar las pruebas es el Apache Bench.

3.2.2. Qué es Apache Bench

Apache Bench es un benchmark[13] que funciona mediante línea de comandos de un solo subproceso para medir el rendimiento servidores web, originalmente fue diseñado para probar el rendimiento del servidor Apache, con el pasar del tiempo los desarrolladores incrementaron su funcionalidad hacia otros servidores por ejemplo: Tomcat, GlassFish, Jboss Ap, etc.

El funcionamiento de Apache Bench es relativamente sencillo, indicamos la URL a testear, el número de peticiones que queremos realizar y el número de peticiones concurrentes.

Apache Bench puede ser usado para testear el rendimiento de cualquier tipo de aplicación y/o servidor web, independientemente del framework, lenguaje o metodología que se haya empleado en el desarrollo.

3.2.2.1. Instalación de Apache Bench y GNUPlot

Apache Bench es una herramienta que se la puede usar después de instalar el paquete de Servidor Web Apache. En los sistemas GNU/Linux basados en Debian o Ubuntu la sintaxis de instalación es la siguiente:

```
sudo apt install apache2 apache2-doc apache2-utils
```

Figura 11. Línea de comandos para instalar el Servidor web Apache y sus herramientas.
Fuente: Autores

GNUPlot estará disponible luego de instalar el paquete de igual nombre.

```
sudo apt install gnuplot
```

Figura 12. Línea de comandos para instalar la herramienta GNUPlot
Fuente: Autores

[13] Benchmark:

3.2.2.2. Como usar Apache Bench

La invocación a Apache Bench se la realiza de la siguiente forma:

```
ab    -g results.csv     -n 1000      -c 20      url_aplicación
[1]         [2]             [3]          [4]            [5]
```

Figura 13. Uso de Apache Bench.
Fuente: Autores

Dónde:

[1] **ab:** se invoca a la herramienta Apache Bench.

[2] **-g results.csv:** guarda los resultados en el archivo results.csv, se los puede también guardar como .xls, txt, tsv, entre otros.

[3] **-n 100:** indica que se harán 100 peticiones

[4] **-c 10:** indica que se harán 10 peticiones concurrentes.

[5] **url_aplicación:** es la URL que vamos a testear

```
ab -g test.csv -n 100 -c 10 https://localhost:8000/
```

Figura 14. Ejemplo del uso de Apache Bench
Fuente. Autores

El resultado que obtendremos por consola será algo similar a esto:

```
Benchmarking localhost (be patient).....done

Server Software:        Apache-Coyote/1.1
Server Hostname:        localhost
Server Port:            8000
SSL/TLS Protocol:       TLSv1.2,ECDHE-RSA-AES256-SHA384,2048,256

Document Path:          /
Document Length:        4035 bytes

Concurrency Level:      10
Time taken for tests:   1.546 seconds
Complete requests:      100
Failed requests:        0
Total transferred:      429100 bytes
HTML transferred:       403500 bytes
Requests per second:    64.69 [#/sec] (mean)
Time per request:       154.594 [ms] (mean)
Time per request:       15.459 [ms] (mean, across all concurrent requests)
Transfer rate:          271.06 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:       69  131  27.3    129     210
Processing:     1   19  14.9     16      70
Waiting:        1    5   7.6      2      28
Total:         82  150  28.6    151     214

Percentage of the requests served within a certain time (ms)
  50%    151
  66%    159
  75%    171
  80%    173
  90%    187
  95%    203
  98%    214
  99%    214
 100%    214 (longest request)
```

Figura 15. Salida de la ejecución del comando de la Figura 14

Lo más relevante de estos resultados es lo siguiente:

- **Requests per second:** peticiones atendidas por segundo durante la prueba.
- **Time per request (mean):** tiempo miedo que el servidor ha tardado en atender a un grupo de peticiones concurrentes (5 o 20).
- **Time per request (mean, across all concurrent requests):** tiempo medio que el servidor ha tardado en atender una petición individual.

Con esta información pueden tener una idea de cuánto demorará el servidor en atender esa cantidad de solicitudes, pueden luego agregar un mejor sistema de caché, desactivar módulos que no usen, etc etc, volver a ejecutar la prueba y ver si el rendimiento mejoró o no.

3.2.2.3. Graficar los resultados mediante GNUPlot

Para poner en una imagen este output, es decir en un medio más agradable para los lectores se usará la herramienta GNUPlot, para ello dirigirse a la misma carpeta donde se encuentra el archivo test.csv (archivo creado anteriormente) se procederá a crear un archivo llamado plot.p, mediante el siguiente comando:

```
sudo vim plot.p
```

Figura 16. Comando para crear el archivo .plot para graficar las salidas de Apache Bench.
Fuente: Autores

El cual contendrá lo siguiente:

```
# establece la salida como imagen png, también soporta jpg, bmp, svg,
entre otros
set terminal png size 600

# guarda el archivo de salida como "results.png"
set output "results.png"

# estable el título del gráfico
set title "1000 peticiones, 20 peticiones concurrentes"

# mejora el aspecto visual—líneas entrepunteada
set size ratio 0.6

# establece el eje Y
set grid y

# agrega el label al eje x
```

```
set xlabel "peticiones"

# agrega el label al eje y
set ylabel "tiempo de respuesta (ms)"

# genera la gráfica a partir del archivo “test.csv” creado
anteriormente y agrega un título a la imagen.
plot "test.csv" using 9 smooth sbezier with lines title "std.ec"
```

Figura 17. Contenido del archivo grafico.plot
Fuente: Autores

Una vez que se ha creado el archivo plot.p y se le ha agregado lo expuesto anteriormente, ejecutar el siguiente comando:

```
gnuplot plot.p
```

Figura 18. Comando para graficar el contenido del Archivo plot.p
Fuente: Autores

El resultado dependiendo de los valores será un gráfico similar al siguiente:

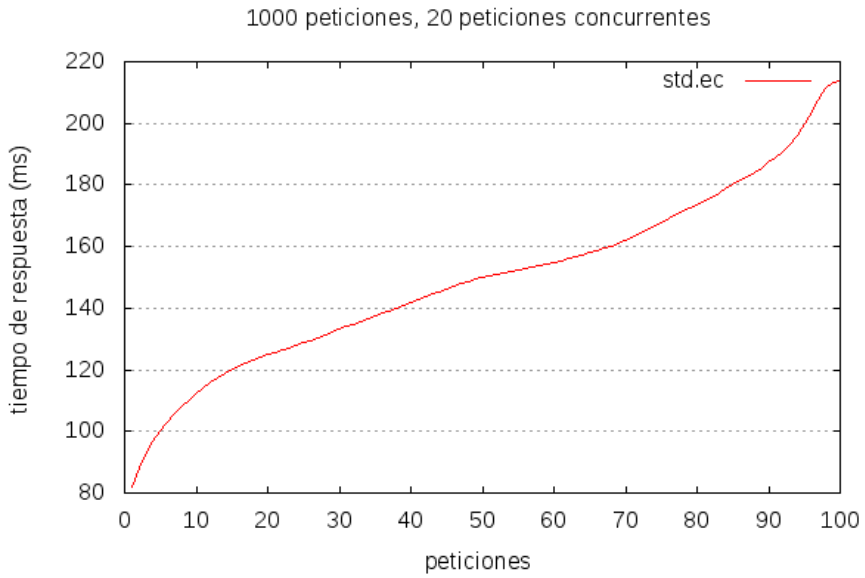


Figura 19. Gráfico que muestra el Tiempo de respuesta vs el número de peticiones realizadas en una prueba de rendimiento
Fuente: Autores

Es importante destacar que las gráficas generadas por este tipo de pruebas no se han de interpretar de manera secuencial, es decir las peticiones (eje X) no aparecen ordenadas de manera cronológica (orden en el que fueron realizadas) sino por su time (tiempo que tardaron en ser atendidas).

3.2.3. Configuración del Sistema

A la hora de comparar las prestaciones de los servidores de aplicaciones hay que tener en cuenta los siguientes criterios:

Tanto el hardware como el software sobre el que se realizará el benchmark contra los servidores de aplicaciones siempre será el mismo. El objetivo es medir las prestaciones de los tres servidores de aplicaciones bajo las mismas condiciones lo que nos aportará un análisis neutro.

Los parámetros que afectarán al rendimiento de los servidores serán los parámetros usados en la herramienta. Los valores de estos parámetros irán cambiando de igual manera para cada servidor de aplicaciones, modificándose la carga de trabajo que tienen que soportar dichos servidores.

Tanto el hardware como el software del sistema serán virtualizados, tomando en consideración los requisitos mínimos para el normal funcionamiento de cada uno de los servidores, el hardware y software a utilizar se detallan a continuación:

Hardware:

Tabla 2. Características del Hardware a virtualizar.

Hardware	Modelo
Procesador	Intel Core i5-2450M @ 2.49GHz (1 Core)
Placa base	Intel 440BX
Chipset	Intel 440BX/ZX/DX
Memoria	1 x 1024 MB DRAM
Disco	21GB VMware Virtual S
Gráficos	VMware SVGA II
Audio	Ensoniq ES1371
Red	AMD 79c970

Fuente: Autores

Software:

Tabla 3. Detalle del Software instalado en los equipos virtualizados.

Software	**Versión**
Sistema Operativo	CentOS 7
Kernel	2.6.32-279.19.1.el6.i686 (i686)

Escritorio	GNOME 3
Display Server	X Server 1.10.6
Display Driver	vmware 11.0.3
OpenGL	2.1 Mesa 7.11
Compilador	GCC 4.4.6 20120305
Sistema de archivos	ext4
Resolución de pantalla	1024x768
Sistema de capas	VMware

Fuente: Autores

3.2.4. Parámetros estadísticos a utilizar

Para realizar un conjunto de mediciones se debe conocer ciertos aspectos que son fundamentales al momento de ejecutar dicho proceso las cuales de una manera u otra incidirán al momento de dar una válida interpretación de los datos recolectados.

Después de obtener resultados lo siguiente es diseñar tablas y gráficas en las que vamos a depositar la información de las mediciones en donde se recopilará todos los datos necesarios para el análisis numérico y gráfico que nos llevará a la obtención de resultados.

Estos datos los evaluaremos tomando en cuenta conceptos de estadística que nos permitirán emitir un juicio real y concluir con acierto acerca de los objetivos planteados.

Puesto que las representaciones gráficas sin un uso de estudio de estadística no siempre consiguen ofrecer una información completa de una serie de datos, es necesario utilizar procedimientos numéricos que permitan resumir toda la información del experimento en unos números llamados parámetros estadísticos. Los parámetros que tendrán más énfasis y se pondrán en aplicación al caso de estudio se detallan a continuación.

a. **Medidas de tendencia central**: Que representan a toda la distribución. Entre las más importantes y que se presenta una breve descripción son la mediana y la moda.

- **Mediana**.- Es el valor que divide a la población o muestra en dos partes, cada una con igual número de datos, también es considerada un fractil, pues es el valor que divide la probabilidad en dos partes.
- **Moda**.- Es el valor que se presenta más veces, es decir, que tiene la frecuencia mayor, si dos o más valores les corresponden la misma frecuencia mayor, la distribución se llama bimodal o multimodal.

b. Medidas de dispersión.- Que indican si los valores están agrupados o dispersos. La más importante y que se presenta una breve descripción es la desviación estándar.

- **Desviación Estándar.-** La raíz cuadrada de la varianza se la denomina desviación estándar o típica que al restar este valor a la media (límite inferior) y sumarlo a la media (límite superior) ayuda a obtener un intervalo y permite asegurar que el 60% de los datos se encuentran dentro de él.

3.2.5. Planificación de las pruebas.

Por cada aplicación se van a realizar las siguientes pruebas:

Tabla 4. Detalle de las pruebas a realizar.

Prueba	Número de Solicitudes	Nivel de Concurrencia
P1	30000	500
P2	30000	1000
P3	30000	2000
P4	30000	2500
P5	30000	5000
P6	30000	10000
P7	30000	15000
P8	30000	20000

Fuente: Autores

Nota: El equipo físico sobre el cual se levantaron los servicios para fines de prueba es accedido a través de la siguiente dirección 190.152.181.70.

3.2.5.1. Criterios a evaluar

- **C1: Requests per second:** constituye las peticiones atendidas por segundo durante la prueba.

- **C2: Time per request (mean):** tiempo (ms) miedo que el servidor tarda en atender a un grupo de peticiones concurrentes.
- **C3: Time per request (mean, across all concurrent requests):** constituye el tiempo (ms) medio que el servidor tarda en atender una petición individual.
- **C4: Transfer rate (kb/s):** constituye la tasa de transferencia con la cual se procesan las peticiones durante la prueba.
- **C5: Total transferred (bytes):** es el total de bytes transferidos desde el servidor a los clientes.
- **C6: Failed Request:** es el total de peticiones que se pierden durante la prueba.

3.2.5.2. Ejecución de las pruebas sobre WildFly

Cada una de las pruebas de rendimiento al servidor WildFly se realizaron de forma idéntica, en cada una de las pruebas el único criterio que cambia es el número de peticiones concurrentes.

El detalle completo de las pruebas se expone en el Anexo 1, a continuación se presenta una prueba con su respectiva gráfica.

```
ab –g test1.csv –n 30000 –k –c 500 https://190.152.181.70:8443
```

Figura 20. Ejecución de Apache Bench sobre el servidor WildFly
Fuente: Autores

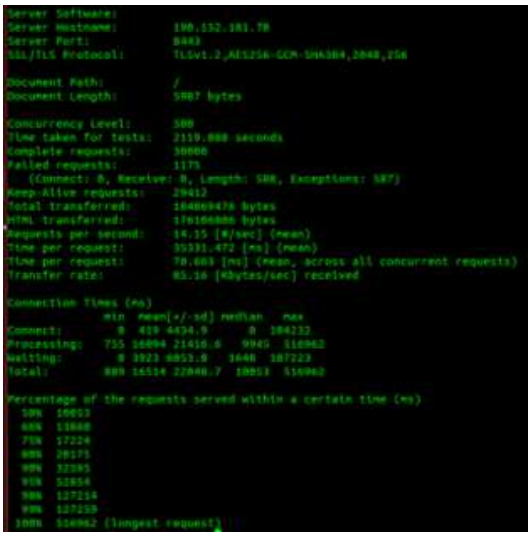


Figura 21. Salida por consola de la ejecución de Apache Bench.
Fuente: Autores

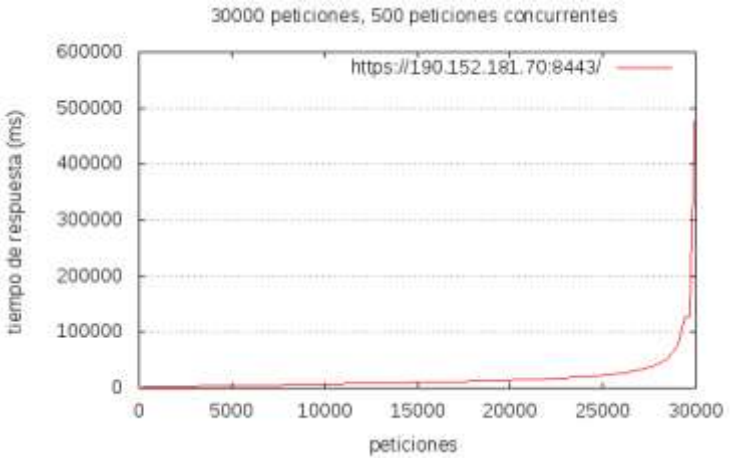


Figura 22. Gráfico de la relación Tiempo de Respuesta vs Número de Peticiones
Fuente: Autores

Una vez que se han ejecutado las pruebas al servidor mediante el uso de Apache bench, en la siguiente tabla se exponen los resultados:

Tabla 5. Resultados de las pruebas de rendimiento al Servidor WildFly

PRUEBA	C1	C2	C3	C4	C5	C6	PORCENTAJE PÉRDIDA
P1	21,72	23019,449	70,663	85,16	184869476	1175	0,0006%
P2	22,24	44970,345	67,948	85,83	179150495	3046	0,0017%
P3	24,487	81683,023	24,789	61,01	46462462	27356	0,0590%
P4	24,52	101946,449	17,954	55,25	30474214	12255	0,0400%
P5	68,54	80537,63	14,107	53,04	22987185	36899	0,16%
P6	106,78	95763,609	9,35	50,85	14606444	44365	0,30%
P7	116,08	129218,179	8,513	43,18	11290512	45022	0,40%
P8	116,51	171659,236	8,5	44,45	11606505	48841	0,42%
PROMEDIO	**62,61**	**94849,684**	**27,73**	**59,85**	**62680911,63**	**27369,88**	**0,17%**

Fuente: Autores

Nota: El detalle de la ejecución de cada una de las pruebas se lo puede observar en el Anexo 2.

3.2.5.3. Ejecución de las pruebas sobre GlassFish

Las pruebas de rendimiento sobre el servidor de Aplicaciones GlassFish se realizaron de forma similar a las pruebas realizadas al servidor WildFly, a continuación se presentan los resultados obtenidos de la evaluación.

```
ab –g test1.csv –n 30000 –k –c 500 https://190.152.181.70:8443
```

Figura 23. Evaluación al Servidor de Aplicaciones GlassFish
Fuente: Autores

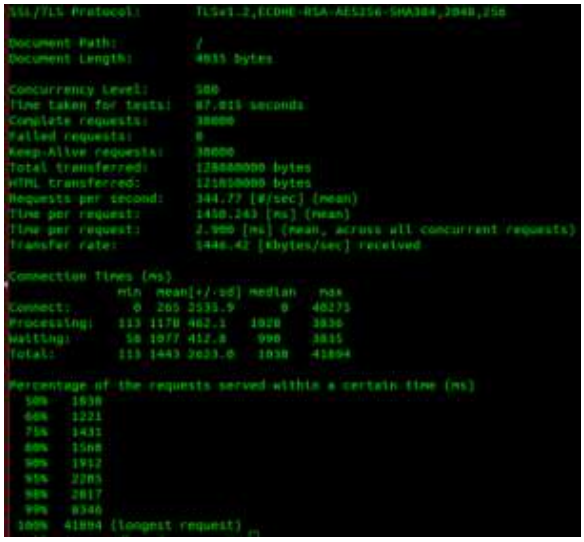


Figura 24. Salida por consola de los resultados de la evaluación.
Fuente: Autores

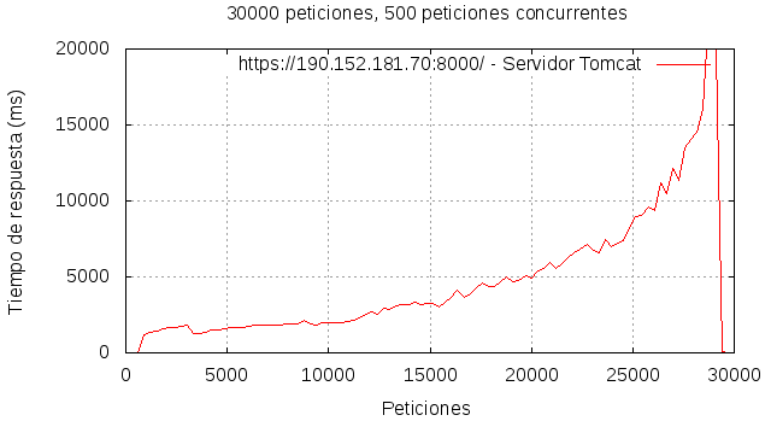


Figura 25. Gráfico de los resultados de la Evaluación
Fuente: Autores

La siguiente tabla recoge los resultados de cada una de las pruebas realizadas al servidor GlassFish.

Tabla 6. Resultados de la evaluación al servidor GlassFish

PRUEBA	C1	C2	C3	C4	C5	C6	PORCENTAJE PÉRDIDA
P1	14,15	35331,472	46,039	86,86	122852837	2174	0,0018%
P2	14,72	67948,153	44,97	84,9	117286174	4408	0,0038%
P3	40,34	49578,968	40,842	82,78	103858164	10260	0,0099%
P4	55,7	44884,406	40,779	78,77	98679627	12671	0,0130%
P5	70,88	70537,02	38,635	72,56	150315716	29997	0,020%
P6	106,95	93498,472	16,864	68,77	150239189	456850	0,30%
P7	117,47	127687,599	8,615	60	15878475	52470	0,33%
P8	117,65	170002,309	8,583	52,83	13929078	53406	0,38%
PROMEDIO	**67,23**	**82433,55**	**30,67**	**73,43375**	**96629907,5**	**77779,5**	**0,13%**

Fuente: Autores

3.3. Análisis e interpretación de resultados

En los siguientes apartados se realiza un análisis comparativo entre los servidores WildFly y GlassFish con respecto a cada uno de los criterios evaluados con Apache Bench.

3.3.1. Comparativa del Criterio "C1: Requests per second"

El primer criterio a evaluar lo constituye el número de peticiones atendidas por segundo, en la siguiente tabla se puede observar que WildFly le lleva una ligera ventaja a GlassFish, cuando atiende entre 500 y 1000 peticiones concurrentes, sin embargo a medida que se incrementan las peticiones concurrentes, esta ventaja se va perdiendo.

Tabla 7. Resultados de la evualuación al Criterio "C1: Requests per second"

Prueba	GlassFish	WildFly
P1	14,15	21,72
P2	14,72	22,24
P3	40,34	24,487
P4	55,7	24,52
P5	70,88	68,54
P6	106,95	106,78
P7	117,47	116,08
P8	117,65	116,51
Promedio	67,23	62,61

Fuente: Autores

En el siguiente gráfico se puede observar detalladamente el número de peticiones atendidas por cada servidor durante la ejecución de cada una de las pruebas de rendimiento.

Figura 26. Peticiones atendidas por segundo
Fuente: Autores

En promedio, de forma general GlassFish atiende 67,23 peticiones por segundo frente a 62,61 peticiones atendidas por segundo por parte de WildFly, observándose una diferencia de 4,62 peticiones.

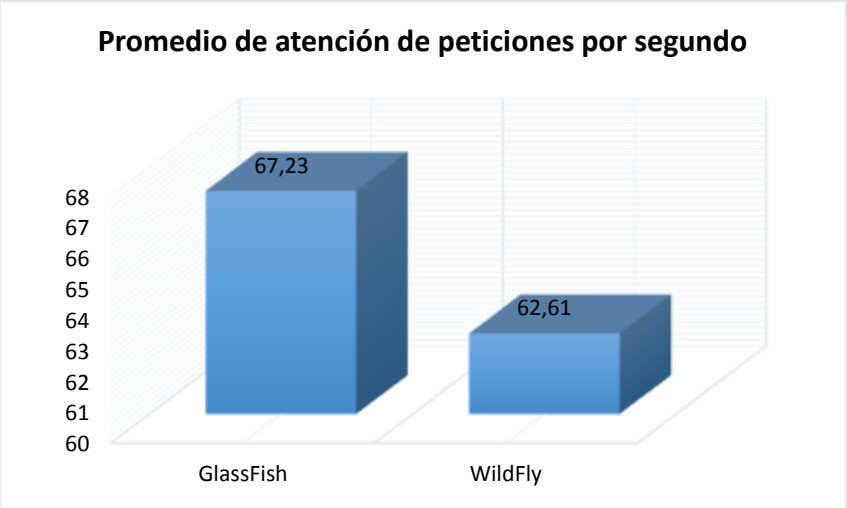


Figura 27. Promedio de peticiones atendidas por segundo.
Fuente: Autores

3.3.2. Comparativa del Criterio "C2: Time per request (mean)"

El tiempo de respuesta a una petición es la única medida que engloba el rendimiento de todo sistema de información, es considerada la medida básica universal considerada en todo tipo de pruebas de rendimiento.

Similar a la prueba anterior WildFly es más rápido que GlassFish cuando se trabajan con peticiones concurrentes comprendidas entre 500 y 1000, a medida que las peticiones concurrentes se van incrementando, GlassFish empieza a responder más rápido que WildFly.

En la siguiente tabla se observa detalladamente los resultados obtenidos en cada prueba realizada a cada uno de los servidores.

Tabla 8. Resultados obtenidos de la Evaluación del Tiempo de respuesta

Prueba	GlassFish		WildFly	
	ms	segundos	ms	segundos
P1	35331,47	35,33	23019,45	23,02
P2	67948,15	67,95	44970,35	44,97
P3	49578,97	49,58	81683,02	81,68
P4	44884,41	44,88	101946,00	101,95
P5	70537,02	70,54	80537,63	80,54

P6	93498,47	93,50	95763,609	95,76
P7	127687,60	127,69	129218,18	129,22
P8	170002,31	170,00	171659,24	171,66
Promedio	82433,55	82,43	91099,68	91,10

Fuente: Autores

El siguiente gráfico muestra de forma más didáctica los resultados mostrados en la tabla anterior.

Figura 28. Resultados obtenidos de la Evaluación del Tiempo de respuesta
Fuente: Autores

Concluidas las pruebas a cada uno de los servidores, se observa que GlassFish atiende un grupo de peticiones concurrentes 8,67s más rápido que WildFly.

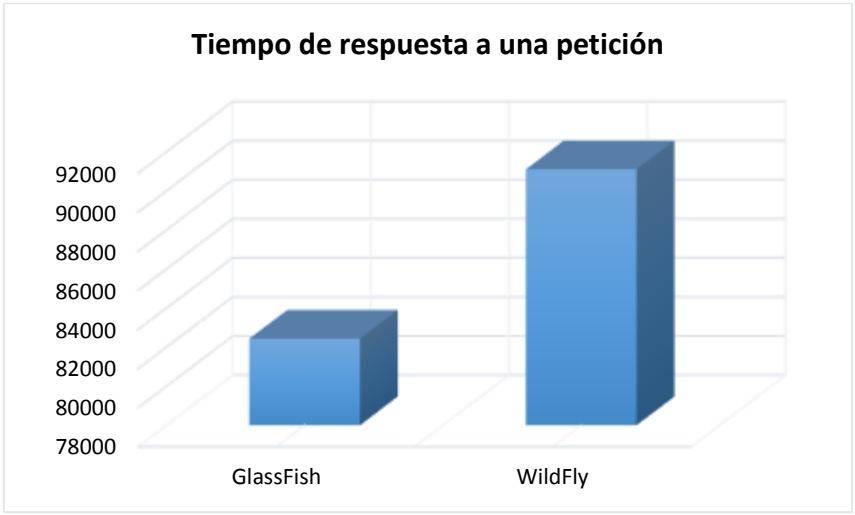


Figura 29. Resultados de la evaluación al tiempo de respuesta a una petición
Fuente: Autores

3.3.3. Comparativa del Criterio "C3: Time per request (mean, across all concurrent requests)"

En la siguiente tabla se observa que WildFly atiende las peticiones de un grupo de peticiones concurrentes (500 – 1000) más rápido que GlassFish. A medida que las peticiones concurrentes van incrementando, GlassFish atiende las peticiones más rápido que WildFly.

Tabla 9. Tiempo promedio en atender las peticiones de un grupo de peticiones concurrentes

Prueba	GlassFish	WildFly
P1	70,663	46,039
P2	67,948	44,97
P3	24,789	40,842
P4	17,954	40,779
P5	14,107	38,635
P6	9,35	16,864
P7	8,513	8,615
P8	8,5	8,583
Promedio	**27,73**	**30,67**

Fuente: Autores

A continuación se expone un gráfico con la comparación del tiempo que tarda cada servidor en atender una petición individual.

Figura 30. Tiempo que tarda cada servidor en atender una petición individual
Fuente: Autores

Como conclusión de esta comparación se observa que GlassFish es 2,94 ms más rápido que WildFly, tal como se indica en el siguiente gráfico.

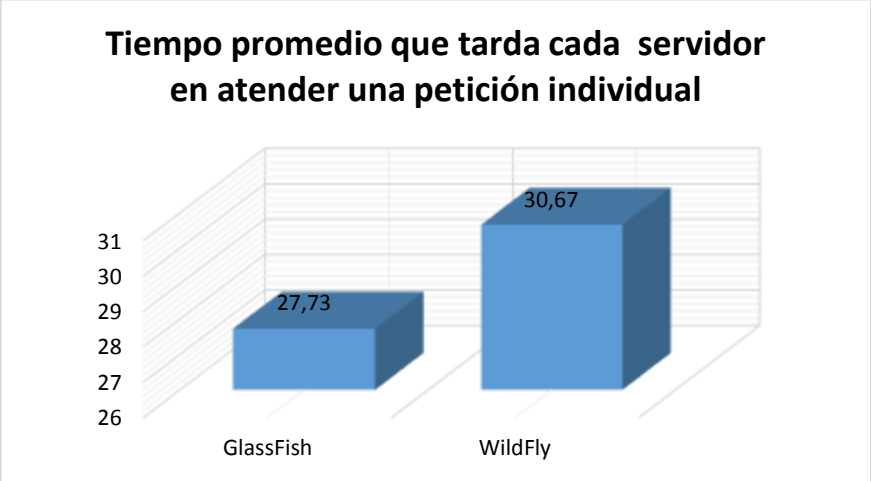


Figura 31, Tiempo promedio que tarda cada servidor en atender una petición individual
Fuente: Autores

3.3.4. Comparativa del Criterio "C4: Transfer rate (kb/s)"

Uno de los factores primordiales para que un servidor de aplicaciones sea considerado como óptimo en entornos de producción es la velocidad de transferencia.

En este apartado se puede observar que tanto a nivel general como a nivel individual GlassFish tiene una significativa ventaja frente a WildFly.

Tabla 10. Resultados de la evaluación de la tasa de transferencia.

Prueba	WildFly	GlassFish
P1	85,16	86,86
P2	85,83	84,9
P3	61,01	82,78
P4	55,25	78,77
P5	53,04	72,56
P6	50,85	68,77
P7	43,18	60
P8	44,45	52,83
Promedio	**59,85**	**73,43**

Fuente: Autores

En la siguiente figura se observa de forma gráfica lo expuesto por la tabla anterior.

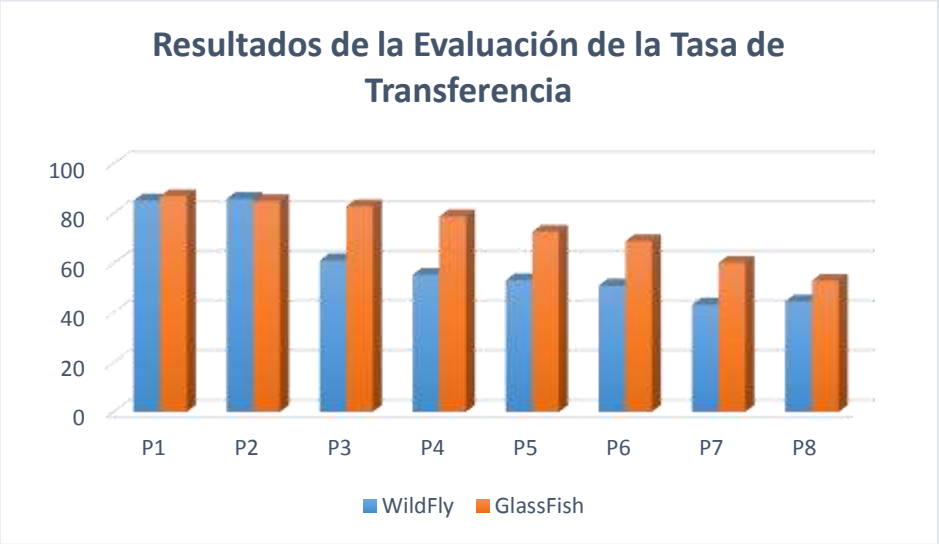


Figura 32. Resultados de la Evaluación de la Tasa de Transferencia
Fuente: Autores

Esta evaluación determina que la tasa de transferencia de GlassFish es 1,8 veces más rápida que la tasa de transferencia de WildFly. En el siguiente gráfico se puede apreciar de forma más didáctica la diferencia entre GlassFish y WildFly con respecto a la tasa de transferencia.

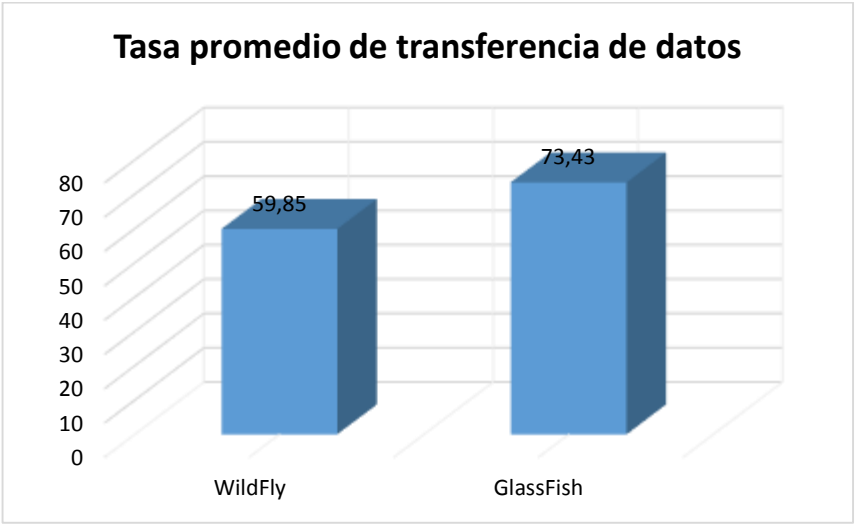


Figura 33. Tasa promedio de transferencia de datos
Fuente: Autores

3.3.5. Comparativa del Porcentaje de Pérdidas de paquetes

Esta comparativa constituye la relación entre la cantidad de bytes transmitidos y el la cantidad de bytes perdidos, en la siguiente tabla se observa que WildFly con un nivel de concurrencia que oscila entre 0 – 1000 el porcentaje de pérdidas es

relativamente bajo, por otra parte en GlassFish el porcentaje de pérdidas es relativamente bajo con respecto a la Tasa de transferencia de datos descrita en la sección anterior.

Tabla 11. Tasa de pérdida de paquetes

Prueba	WildFly	GlassFish
P1	0,00064%	0,0018%
P2	0,0017%	0,0038%
P3	0,059%	0,0099%
P4	0,040%	0,013%
P5	0,16%	0,020%
P6	0,30%	0,304%
P7	0,40%	0,33%
P8	0,42%	0,38%
Promedio	**0,17%**	**0,13%**

Fuente: Autores

La siguiente figura muestra de forma más visible lo expuesto por la tabla anterior.

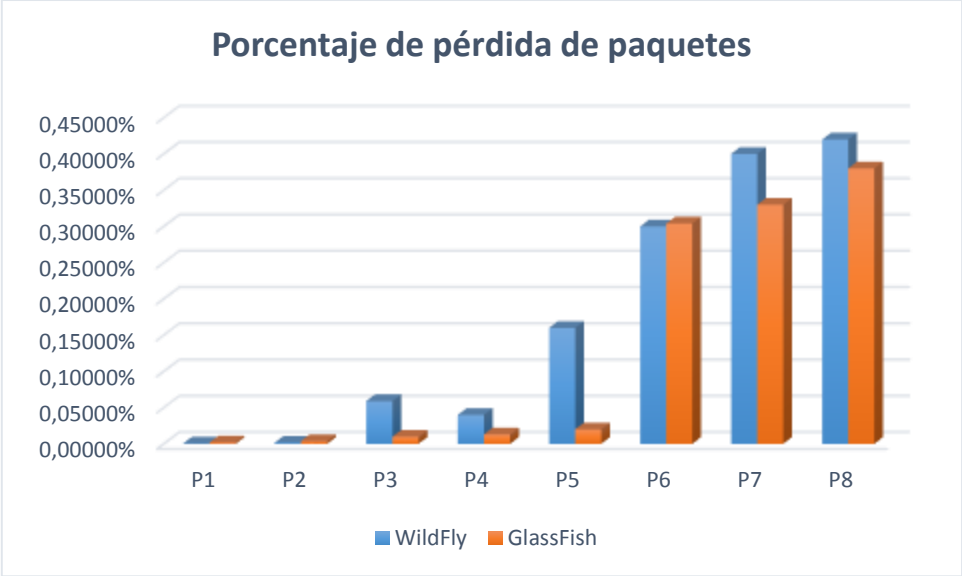


Figura 34. Porcentaje de pérdida de paquetes
Fuente: Autores

En esta comparativa GlassFish en promedio pierde menos paquetes que WildFly, sim embargo cabe recalcar que la tasa de transferencia de GlassFish es 1,8 veces más rápida que WildFly.

La siguiente figura muestra la diferencia existente entre WildFly y GlassFish.

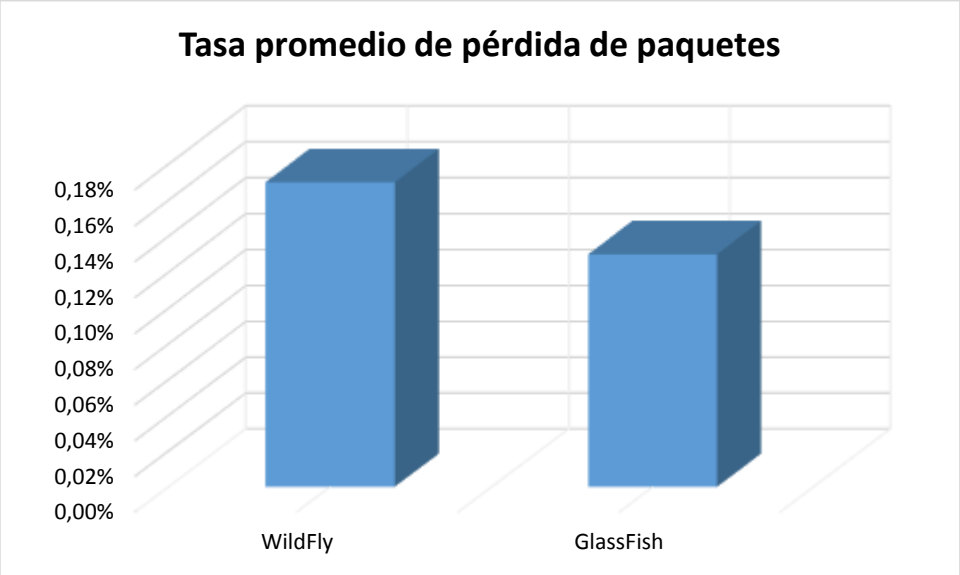


Figura 35. Tasa promedio de pérdida de paquetes
Fuente: Autores

CAPITULO IV

DESARROLLO DEL SISTEMA DE GESTIÓN DE JUNTAS DE AGUA POTABLE Y JUNTAS DE RIEGO EN LA SENAGUA CHIMBORAZO

En este capítulo se detalla el desarrollo del Sistema de Gestión de Juntas de Agua Potable y de Riego para la SENAGUA – Chimborazo, el mismo que se lo realizó sobre el servidor de aplicaciones GlassFish, servidor que en el Capítulo III resultó ser el más óptimo para contener aplicaciones Java EE.

El desarrollo de la aplicación está basado en el Modelo Iterativo – Incremental, del cual se consideran las siguientes fases:

- Planificación:
- Análisis:
- Diseño:
- Construcción:
- Pruebas
- Implementación

La SENAGUA dirección provincial Chimborazo lleva un registro de forma manual de las juntas de agua potable y de riego, de forma similar a la hora de realizar informes y reportes estos se los hace de forma impresa y en la mayoría de los casos no se los entrega a tiempo, peor aún no se cuenta con una herramienta que proporcione información específica que ayude a la toma de decisiones con respecto al seguimiento de las juntas de agua potable y riego.

Por consiguiente se ve la necesidad de automatizar el proceso de gestión de las juntas de agua potable y juntas de riego a través de la implementación de un sistema informático que permita llevar el control y seguimiento de las organizaciones antes mencionadas, generar reportes tanto digitales así como de forma impresa con la finalidad de apoyar los procesos que hasta hoy se los realiza de forma manual.

4.1. Planificación

4.1.1. Situación actual

En el Ecuador a partir del año 2008 a raíz que se puso en marcha el Decreto 1014 en el cual se establece como política pública el uso del Software Libre en los sistemas informáticos de las instituciones públicas, muchas de ellas de ellas han empezado a migrar sus sistemas hacia esta nueva plataforma sin embargo en la mayoría de casos al desconocer de esta tendencia tecnológica no se deciden a migrar sus sistemas peor aún desarrollarlos ya sea por medios propios o mediante convenios y siguen trabajando con los sistemas informáticos propietarios y tampoco han buscado la forma de automatizar sus procesos.

La Secretaría Nacional del Agua (SENAGUA) Dirección Provincial de Chimborazo, lleva el registro de los datos correspondientes a las Juntas Administradoras de agua potable y Juntas de Riego los cuales son registrados en matrices en hojas de Excel esto dificulta el acceso y seguimiento de los datos.

A esta institución le interesa disponer de un sistema web a través del cual se puede realizar el seguimiento de las juntas mencionadas anteriormente, así como también la generación de reportes sobre la situación actual de las mismas.

4.1.2. Perspectivas del producto

Se pretende desarrollar un sistema informático de entorno web, con herramientas de software libre a través del cual los funcionarios de la SENAGUA Chimborazo puedan realizar la gestión de las juntas de agua potable y juntas de riego de forma automatizada, tener a disposición en cualquier momento de una herramienta que sea capaz de generar reportes estadísticos sobre la situación actual sobre las juntas ya sean de agua potable o de riego.

El sistema desarrollar tiene como finalidad representar a la organización y sus objetivos, minimizar los problemas que pudiesen encontrarse y de esta forma apoyar sus procesos.

4.1.2.1. Características del Sistema a desarrollar

El sistema a desarrollar permitirá a la SENAGUA tomar decisiones en base a los resultados que se obtengan de las consultas y reportes sobre las JAAP[14] y JAR[15], además de llevar la información digitalmente se reduce la presencia de papeles u hojas y un respaldo histórico de la información generada.

El sistema podrá ser accesible desde cualquier parte del país por tratarse de un sistema web, siempre y cuando se tenga acceso a internet.

El sistema estará estructurado en grupos que se especifican de la siguiente manera:

- **Módulo Administración Master:** constituye el módulo central, a través del cual se realiza la configuración de las funcionalidades de los módulos que se desarrollen posteriormente.
- **Módulo Gestión de dirigentes:** este módulo permite registrar, actualizar, visualizar y/o eliminar la información sobre los dirigentes de las juntas tanto de agua potable así como también de las juntas de riego.
- **Módulo Juntas de Agua potable:** en este módulo se puede registrar, visualizar, actualizar y eliminar la información sobre las juntas de agua potable, genera reportes y estadísticas en base a la información que se va generando sobre las juntas.
- **Módulo Juntas de Riego:** en este módulo se puede registrar, visualizar, actualizar y eliminar la información sobre las juntas de riego, genera reportes y estadísticas en base a la información que se va generando sobre las juntas
- **Módulo Configuraciones:** este módulo permite agregar funcionalidades a los módulos: juntas de agua potable y juntas de riego, además permite activar o eliminar dirigentes que ya cumplieron su periodo.

4.1.2.2. Personal involucrado

Para el desarrollo del sistema, se contará con las siguientes personas:

[14] JAAP.- Acrónimo de Juntas Administradoras de Agua Potable
[15] JAR.- Acrónimo de Juntas Administradoras de Riego

Tabla 12. Personal involucrado en el desarrollo del sistema de la SENAGUA.

N°	Persona	Rol	Responsabilidades
1	Darwin Balbuca	Tesista / Analista de Sistemas	Capturar, especificar y validar requisitos, interactuando con el cliente y los usuarios mediante entrevistas. Elaborar el Modelo de Análisis y Diseño. Colaborar en la elaboración de las pruebas funcionales y el modelo de datos.
2	José Ortiz	Tesista / Analista de Sistemas	Construir prototipos. Colaborar en la elaboración de las pruebas funcionales, modelo de datos y en las validaciones con el usuario.
3	Ing. Jorge Delgado	Tutor – Director	Tutorías para el desarrollo del sistema
4	Ing. Klever Morocho	Responsable del Área de Sistemas	Facilitar la información sobre las plataformas tecnológicas utilizadas en la SENAGUA.
5	Ing. Luis Agualsaca	Coordinador Juntas de Riego	Facilitar la información sobre las juntas de riego. Revisar los entregables del proyecto.
6	Ing. Carlos Romero	Coordinador del Departamento de Recursos Hídricos	Facilitar la información concerniente a las juntas administradoras de agua potable.

Fuente: Autores

4.2. Análisis

Continuando con el desarrollo del sistema, en esta sección se lleva a cabo la especificación de requisitos y posteriormente el respectivo análisis de los mismos, al final de esta sección se tendrá una visión clara de las problemáticas que se van a solucionar mediante el desarrollo del sistema.

4.2.1. Especificación de requisitos

Por cuestiones técnicas y de desarrollo, la especificación de requisitos se la ha divididos en dos grupos: requisitos funcionales y no funcionales.

Los requisitos funcionales son los que están relacionados directamente con las funciones del sistema a desarrollar, su correcta especificación incidirá en el éxito o en el fracaso del mismo.

Los requisitos no funcionales están relacionados con la infraestructura tecnológica y con el rendimiento de la aplicación; el rendimiento de la aplicación se lo realizó en el Capítulo III en el análisis comparativo de los Servidores de Aplicaciones para la plataforma Java EE.

4.2.1.1. Requisitos funcionales

El fin que persigue en esta fase es estudiar el negocio, así como todos aquellos elementos que formaran en el desarrollo del proyecto, desde el punto de vista de los usuarios y el negocio[16] determinando de esta forma los requerimientos informáticos, operativos, técnicos, de desarrollo e implementación.

A continuación se describirá de forma que se empezaran las fases:

- Se definirá con los usuarios los requerimientos informáticos, operativos, técnicos de desarrollo e implementación. Para ello se recolectara la información necesaria sobre las necesidades de información, las condiciones en las que operan con los actuales sistemas los recursos técnicos con los que constan para desarrollar acorde el sistema actual.
- Se validaran todos los requerimientos con los usuarios, esto evitara conflictos con sus ventajas
- Elaborar el documento de especificación de requerimientos

En la siguiente tabla se detallara la fase con sus respectivos recursos, herramientas, técnicas, finalidades y lugares de aplicación:

[16] Desarrollo de sistemas de información: Una metodología basada en el modelado, Vicenc Fernadez Alarcon, Edicions UPC.

Tabla 13. Recursos requeridos para la ejecución de la fase de análisis de requisitos

TAREA	TÉCNICA HERRAMIENTA	RECURSO HUMANO	FINALIDAD	LUGAR DE APLICACIÓN
Definición de requerimientos	Observación directa.	Equipo de trabajo, usuarios	Obtener los requerimientos técnicos, operativos, informáticos de desarrollo e implementación	UNACH - SENAGUA
Validación de requerimientos	Matriz de prioridades	Equipo de trabajo, usuarios	Asegurar que os requerimientos estén acorde con las necesidades de los usuarios	UNACH - SENAGUA
Elaboración documento de especificación de requerimientos	Office 2013	Equipo de trabajo, usuarios	Crear un documento que contenga el detalle de los requerimientos	UNACH - SENAGUA

Fuente: Autores

Producto Final: al terminar la fase de determinación de los requerimientos se obtendrán un documento con las especificaciones de los requerimientos para el desarrollo del sistema de información para la SENAGUA – CHIMBORAZO.

A continuación se recogen los principales requisitos funcionales:

Tabla 14. Tabla de Requisitos funcionales

Código	Nombre	Descripción	Entradas
Req001	Gestión de Juntas administradoras de agua potable.	Permite ingresar, seleccionar, actualizar y eliminar los datos de una junta administradora de agua potable.	**Se registrará:** el nombre de la junta, el estado actual de la junta, parroquia, dirigente, dirección, correo electrónico, teléfono convencional, teléfono celular , numero de autorización, número de resolución, la fecha de registro, RUC, el caudal aforado, caudal autorizado, número de consumidores, coordenada x, coordenada y, coordenada z y observaciones.
Req002	Gestión de Juntas de riego	Permite ingresar, seleccionar, actualizar y eliminar los datos de una junta de riego.	**Se registrará:** el nombre de la junta, el estado actual de la junta, parroquia, dirigente, dirección, correo electrónico, teléfono convencional, teléfono celular , numero de autorización, número de resolución, la fecha de registro,

			RUC, el caudal aforado, caudal autorizado, número de consumidores, coordenada x, coordenada y, coordenada z y observaciones.
Req003	Gestión de dirigentes	Permite ingresar, actualizar y eliminar los datos de un dirigente.	**Se registrará:** la cédula, nombres, correo, dirección si es una persona natural o ruc, razón social y representante si es una persona jurídica.
Req004	Gestión de reportes	Permite recuperar reportes específicos sobre juntas administradoras de agua potable, juntas de riego, dirigentes, y datos que necesiten los usuarios.	**Se ingresará:** tipo de reporte y el formato de salida de los reportes.
Req005	Estadísticas	Genera gráficos estadísticos (barras y series) sobre juntas administradoras de agua potable, juntas de riego, dirigentes, y datos que necesiten los usuarios.	**Se ingresará:** el tipo de estadística que se quiere generar y el tipo de gráfico.
Req006	Inicio y cierre de sesión en el sistema	Permite ingresar al sistema mediante la autentificación y control de sesiones.	**Se ingresará:** el usuario y contraseña.
Req007	Gestión de perfiles de usuario	Permite crear, actualizar y eliminar perfiles de usuario de acuerdo a los roles que tendrán dentro del sistema.	**Se ingresará:** el tipo de rol de los usuarios.

Fuente: Autores

4.2.1.2. Requisitos no funcionales

Tabla 15. Especificación de requisitos no funcionales

CÓDIGO	REQUISITO
RNF-01	Marco Legal: el sistema informático está regulado de acuerdo a las normas y reglamentos establecidos por la Universidad Nacional de Chimborazo y las normativas de la SENAGUA.
RNF-02	Medio Ambiente del Sistema: Condiciones necesarias que necesita el sistema para poder operar de forma adecuada
RNF-03	Mecanismos de control: Políticas de seguridad, que garantice la integridad y confidencialidad de la información personal
RNF-04	Volumen de actividades: El sistema debe contar con suficiente espacio en el disco para almacenar toda la información
RNF-05	Seguridad Física del sistema

RNF-06	Seguridad Lógica del sistema
RNF-07	Características técnicas del servidor y el equipo de desarrollo
RNF-08	Lenguaje de programación, servidor web, sistema gestor de base de datos para el desarrollo
RNF-09	Software utilitario para el desarrollo
RNF-10	Recurso Humano calificado para el desarrollo del sistema
RNF-11	Características técnicas del servidor y el equipo para implementación
RNF-12	El sistema debe trabajar sobre una red de comunicaciones
RNF-13	Software utilitario para la implementación
RNF-14	Recurso Humano calificado para la implementación

Fuente: Autores

4.2.2. Análisis de requisitos

En esta fase se determinaran los elementos que intervienen en el sistema a desarrollarse su estructura, relaciones, evolución y funcionalidades; se tendrá un descripción clara de que producto se va construir, que funcionalidades aportara y que comportamiento tendrá[17]. (Cantone, 2006)

La siguiente tabla especifica las tareas y las técnicas a utilizarse para el análisis de los requisitos especificados en la tabla anterior.

Tabla 16. Técnicas y herramientas a usar en el desarrollo de la fase

Tarea	Técnicas/Herramienta	Recurso Humano	Finalidad	Lugar de Aplicación
Análisis de las necesidades del sistema	Casos de Uso Diagrama UML Diagrama de Secuencia Diagramas de Flujo	Equipo de trabajo	El análisis prepara una propuesta del sistema que sintetiza sus hallazgos y reconocimiento sobre lo que se debe hacer	UNACH - SENAGUA
Preparación del documento de análisis del sistema	Office 2013	Equipo de trabajo	Documentar los diagramas UML	UNACH – SENAGUA

[17] Implementación y Debugging: la biblia de la programación, capitulo 1: Ciclo de la vida del software, Dante Cantone, Mp Ediciones Corp. 2006.

Fuente: Autores

Este aparatado proporciona una vista coherente de los casos de uso y los actores del sistema así como los límites del sistema. Los casos de uso son ponderados y priorizados.

4.2.2.1. Actores

Actores son seres humanos con sus diferentes roles de usuario u otros sistemas que se comunican con el sistema de gestión de la SENAGUA. El tipo de actor determina su peso. Esto puede variar entre 1 y 3, donde los últimos resultan ser más complejos. Un actor humano, interactúa por medio de una (gráfica) interfaz de usuario se cuenta con un peso de 3. Una interfaz basada en un protocolo cuenta con un peso de 2 y una interfaz de programación como 1. Los pesos son usados como base para el análisis de los puntos de caso de uso.

Tabla 17. Actores del Sistema

Actor	Descripción	Peso
Administrador	Gestionará y controlará toda la información que se genere en el sistema.	1
Operador	Será la persona que se encargue de utilizar el sistema y realizar las actividades básicas.	3

Fuente: Autores

4.2.2.2. Casos de uso

Describe la interacción de un actor del sistema, esta interacción conduce a un objetivo el cual es significativo para el actor o usuario del sistema.

El peso de un caso de uso es determinado por la cantidad y complejidad de los escenarios con los cuales interactúa un caso de uso. La última columna constituye la tabla de estados de prioridad de los casos de uso, la prioridad de los casos de uso se determina a través de letras, estas letras son las consonantes en la palabra MoSCoW, que significa lo siguiente:

- **M – MUST: 'Debe tener',** este caso de uso es indispensable para el sistema al ser útil o ser válido para el caso del negocio.

- **S – SHOULD: 'Debería tener',** este caso de uso es necesario.
- **C – COULD: 'Podría tener',** este caso de uso agrega valor, pero sin este el sistema todavía no sería útil.
- **W - WON'T: 'Es deseable que tenga pero no lo tendrá esta vez',** Este caso de uso no será construido en esta iteración de desarrollo de software.

Una distribución correcta presentaría un máximo del 70% de casos con la prioridad MUST (Debe tener)

Tabla 18. Especificación de Casos de Uso

Código	Nombre del Caso de Uso	Descripción	Peso	Prioridad
CU01	Acceder al sistema	Como funcionario de la SENAGUA, deseo ingresar al sistema.	3	M
CU02	Gestionar juntas de agua potable	Como operador del Sistema deseo registrar, modificar, visualizar, buscar y eliminar la información referente a las juntas administradoras de agua potable.	3	M
CU03	Gestionar juntas de riego	Como operador del Sistema deseo registrar, modificar, visualizar, buscar y eliminar la información referente a las juntas de riego.	3	M
CU04	Gestionar Dirigentes	Como operador del Sistema deseo registrar, modificar, visualizar, buscar y eliminar la información referente a los dirigentes de las juntas de agua potable y juntas de riego.	3	M
CU05	Reportes	Como usuario del sistema necesito generar reportes generales y específicos sobre las juntas administradoras de agua potable, juntas de riego y dirigentes.	3	M
CU06	Estadísticas	Como usuario del sistema deseo generar gráficos estadísticos (series, barras y pasteles) sobre las características de las juntas de agua potable y juntas de riego.	3	M
CU07	Configuraciones	Como usuario del sistema necesito configurar los diferentes parámetros de las funciones de los módulos juntas de agua potable y juntas de riego.	1	S

CU8	Administración Global	Como usuario del sistema deseo configurar los usuarios, roles y privilegios del acceso al sistema así como también los diferentes módulos del sistema.	1	S

Fuente: Autores

4.3. Diseño

Esta fase contiene el diseño del sistema de información que cumplirá con todos los requerimientos encontrados en las fases anteriores, se definen los estándares para crear los diseños de base de datos, interfaces de usuario de entrada, salida, alertas, mensajes, errores

A continuación se detallaran las actividades a llevarse a cabo en esta fase:

- Definición de los estándares de diseño
- Diseño de la base de datos
- Diseño de las interfaces de usuario

En la siguiente tabla se detallan las tareas a desarrollarse en esta fase con sus respectivos recursos, herramientas, técnicas, finalidades y lugares de aplicación.

Tabla 19. Tareas, técnicas y herramientas a usar en la fase de Diseño.

TAREA	TÉCNICA HERRAMIENTA	RECURSO HUMANO	FINALIDAD	LUGAR DE TRABAJO
Definir los estándares de diseño	MS Office 2013	Equipo de Trabajo	Establecer una norma común que guie el diseño de los diferentes elementos del sistema	UNACH – SENAGUA
Diseñar la base de datos	Modelos de datos/	Equipo de Trabajo	Modelar la base de datos que refleje la lógica del negocio	UNACH – SENAGUA
Diseñar las interfaces de usuario	Diseño gráfico/	Equipo de Trabajo	Elaborar los bocetos de las diferentes ventanas de interacción entre los usuarios	UNACH – SENAGUA
Elaboración del documento	MS Office 2013	Equipo de Trabajo	Documentar el diseño de los componentes del sistema	UNACH – SENAGUA

Fuente: Autores

4.3.1. Arquitectura de la solución

En esta sección se detalla la arquitectura que se emplea en la aplicación para lo cual primero se indica el tipo de arquitectura elegida. Luego se presenta el diseño de la arquitectura de alto nivel que se utiliza en la solución, esto implica dividir la aplicación en componentes funcionales posicionados en capas, las cuales también son detalladas.

4.3.1.1. Representación de la arquitectura

La arquitectura a utilizar será Web. Se distinguen dos secciones, el cliente, donde se encuentra el usuario del sistema y que accederá a la aplicación por medio de un navegador (Internet Explorer o Mozilla Firefox), y la segunda sección la conforma el servidor, en donde residen los datos, las reglas y lógica de la misma.

Uno de los motivos por los que se realiza una aplicación Web es porque se sabe que este tipo de aplicaciones emplean “light clients”, que son clientes que no ejecutan demasiadas labores de procesamiento para la ejecución de la misma aplicación, lo cual es un punto esencial ya que lo que menos se desea es que en la sección cliente se realicen demasiadas tareas, solo las necesarias para que el usuario final pueda acceder a la aplicación y realizar el trabajo deseado.

El auge de las redes locales y la popularidad de Internet han posibilitado el acceso a través de computadores y otros dispositivos móviles, ha aumentado y extendido el empleo de las aplicaciones Web las cuales pueden ser utilizadas por usuarios ubicados en cualquier lugar del planeta con acceso a Internet.

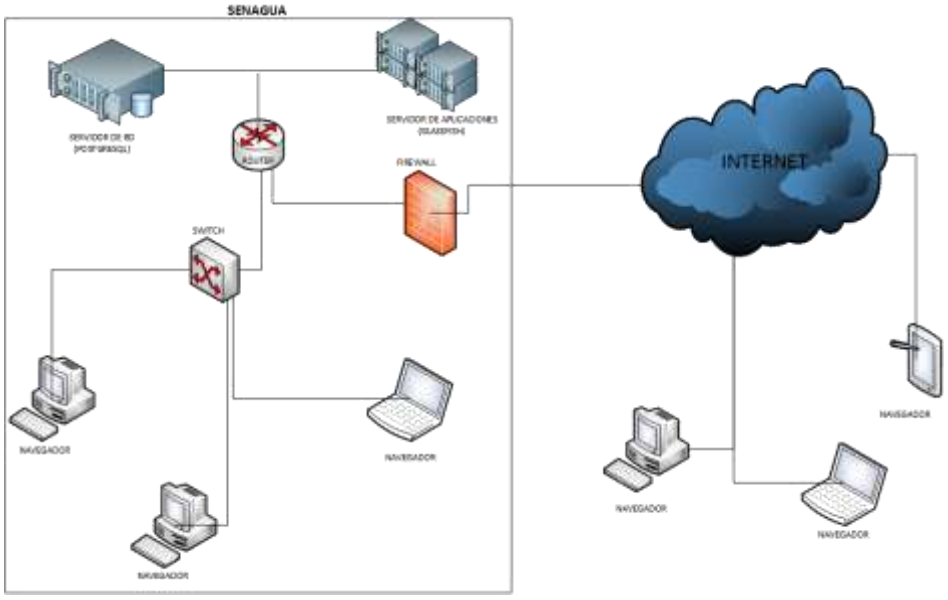


Figura 36. Arquitectura de la solución del sistema
Fuente: Autores

4.3.1.2. Diseño de la arquitectura de la solución

Se elige la arquitectura multicapas, porque es un patrón de diseño muy recomendado para aplicaciones interactivas de Java EE, separa los conceptos de diseño por lo que minimiza la duplicación de código, el control centralizado y permite que la aplicación sea más extensible. Asimismo permite enfocarse en la lógica de negocio, es decir, las funcionalidades a implementar ya que los detalles del manejo de la presentación de la aplicación son cubiertas por MVC.

Se elige JPA porque, a través de los mapeos "Object/Relational", se consigue una persistencia de datos poderosa y de alta performance. JPA soporta la mayoría de los sistemas de bases de datos SQL, lenguaje en el cual se encuentra la base de datos del sistema. Una de las principales ventajas de JPA es ofrecer facilidades para la recuperación y actualización de datos y control de transacciones.

4.3.1.3. Vista lógica

La siguiente figura muestra la vista lógica de la arquitectura, la cual detalla las capas a utilizar y los frameworks que se utilizan en cada una de ellas:

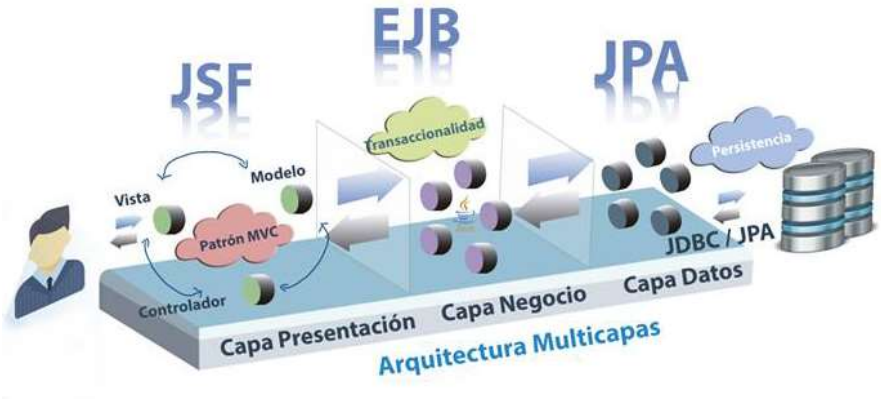


Figura 37. Arquitectura del Sistema
Fuente: Autores

Se han definido tres capas dentro de las cuales se dividirá todo el sistema:

- **Capa de Presentación (Vista):** Esta capa es la responsable de la visualización del sistema, es decir de la parte que interactuará con el usuario. Tendrá como patrón de diseño MVC, el cual provee de librerías para los controladores de interfaz y soporta distintas tecnologías para el desarrollo de las vistas, en esta capa se utiliza Java Server Faces como tecnología de desarrollo y PrimeFaces como librería de interfaz enriquecida.
- **Capa de lógica de negocios:** Esta capa presenta una "interfaz" para brindar servicios a la capa de presentación, en esta capa se incluye a los servicios que proveerán los métodos que típicamente representarán a un caso de uso, es decir, que implementarán las funcionalidades deseadas, para lo cual se emplea la tecnología EJB.

  Esta capa reduce el número de llamadas requeridas al sistema, lo cual hace más fácil su uso. En entornos remotos, esto mejora dramáticamente la performance.
- **La capa de acceso a datos:** Esta capa es una porción de código que justamente realiza el acceso a los datos. De esta manera cuando es necesario cambiar el motor de base de datos, solamente se tendrá que corregir esa capa. Mientras que la capa de datos (en el nivel de datos) es donde están los datos y se corresponde directamente con la definición de esquemas, tablas, vistas, procedimientos almacenados y todo lo que se pueda o deba

poner en un motor de base de datos. Se emplea JPA y JDBC como frameworks para de persistencia.

La realización de todos los casos de uso determinados para el sistema está comprendida por módulos y una base de datos (PostgreSQL). Cada uno de estos módulos contiene 4 tipos de clases (Web, Servicios, Modelo y las de Acceso a Base de datos).

- **Clases Web (web):** Contiene las clases controladoras es decir los ManagedBeans (View Scoped, Session Scoped, etc)
- **Clases de Servicios (service):** Las clases que se encargan del manejo de las clases del negocio. Son una especie de nexo entre la interfaz de usuario y los objetos de negocio.
- **Clases Entidades (lógica):** Las clases que representan las entidades del modelo de negocio. Contienen todos los datos del sistema. Ejemplos de estas clases son Junta, Dirigente, Parroquia, etc.
- **Clases de Acceso a Base de Datos (dao):** Las clases que proporcionan la comunicación con la Base de Datos del Sistema.

4.3.1.4. Vista de despliegue

La vista de despliegue muestra las relaciones físicas de los nodos que participan en la ejecución y de los componentes hardware y software que residen en ellos.

- **Cliente:** En este nodo se hace uso de un navegador de Internet para que los usuarios puedan acceder al sistema a través de computadoras personales.
- **Servidor de aplicaciones:** Este nodo se encarga de manejar la lógica del negocio. El equipo usuario se conecta a él para obtener los datos que requiere para completar sus procesos.
- **Servidor BD:** Este nodo contiene el servidor de base de datos del sistema.

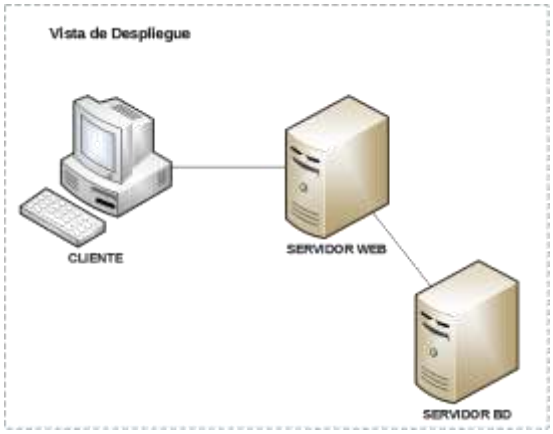


Figura 38. Vista de despliegue del sistema
Fuente: Autores

4.3.2. Diseño de bases de datos

4.3.2.1. Modelo entidad – relación

De forma general el siguiente diagrama E – R, representa a las principales entidades consideradas en el desarrollo del sistema de gestión de juntas de agua potable y riego para la SENAGUA Chimborazo, cabe recalcar que no se consideran las entidades usuario, rol y persona debido a que este tipo de entidades son usadas para la administración del sistema en general.

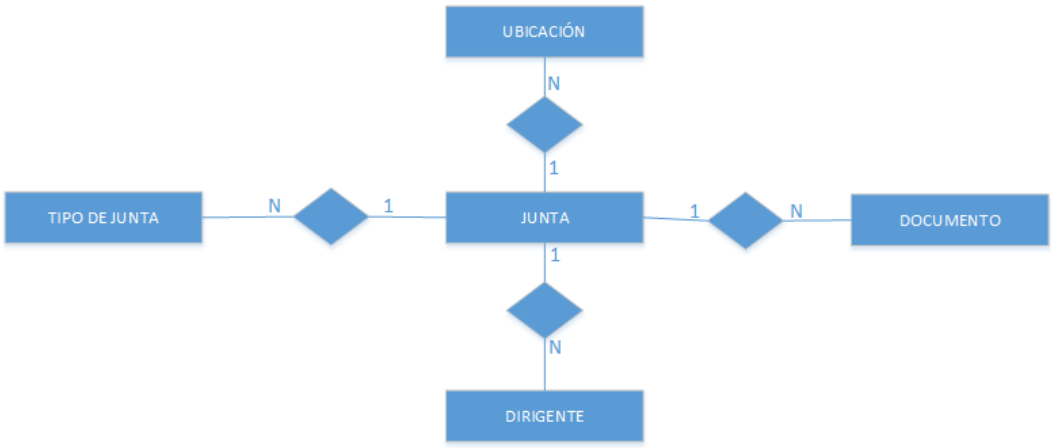


Figura 39. Diagrama E - R del sistema.
Fuente: Autores

A continuación se describen las entidades expuestas anteriormente:

- **Ubicación:** esta entidad está compuesta por las entidades Zona, provincia, cantón, parroquia y comunidad.

- **Tipo de junta:** contiene el tipo de junta que puede ser junta administradora de agua potable o junta de riego.
- **Junta:** representa a las juntas de agua potable o juntas de riego dependiendo del tipo que se quiera registrar.
- **Dirigente:** representa a la persona que se encuentra bajo la dirección de una respectiva junta.
- **Documento:** representa a los diferentes tipos de documentos que posee una junta.

4.3.2.2. Diagrama de bases de datos

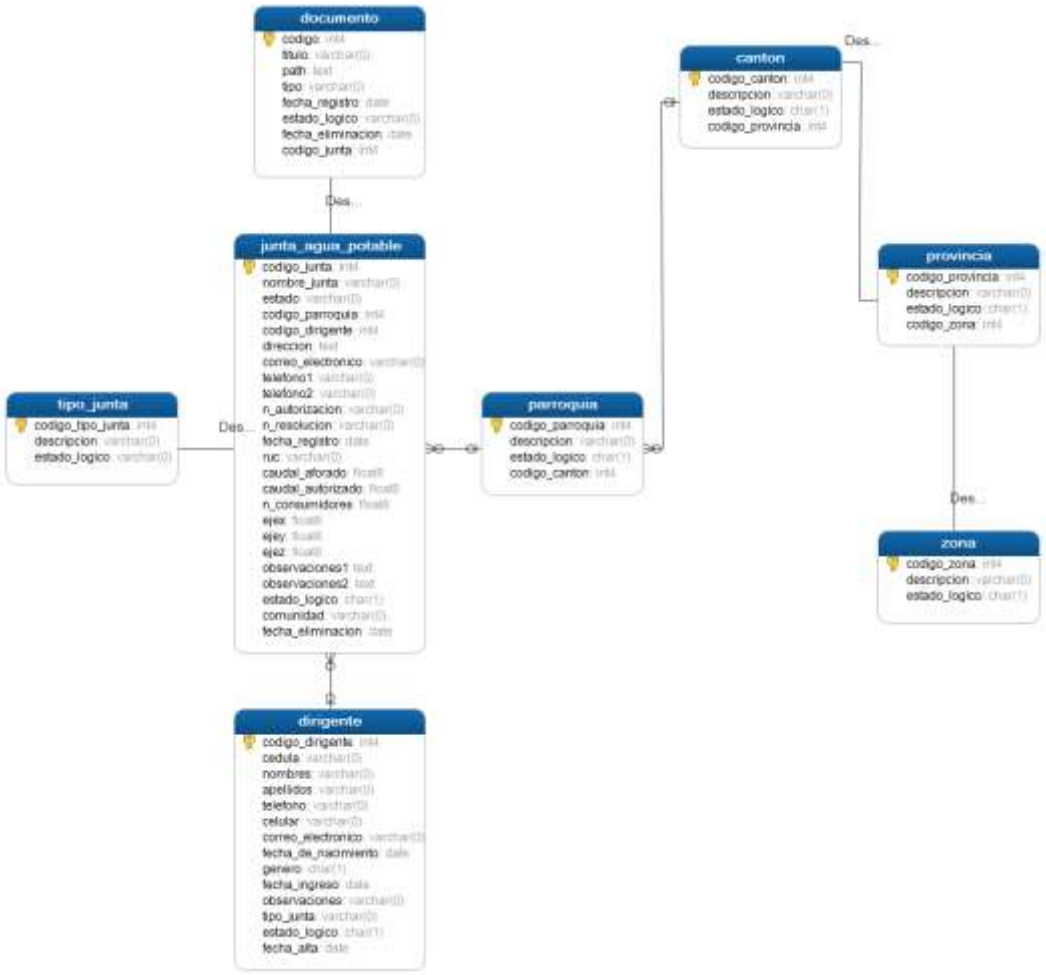


Figura 40. Diagrama de Base de datos del Sistema
Fuente: Autores

4.3.3. Diseño navegacional

En esta fase se detalla la manera de cómo se va a realizar la navegación a través del sistema.

4.3.3.1. Diagramas de navegación

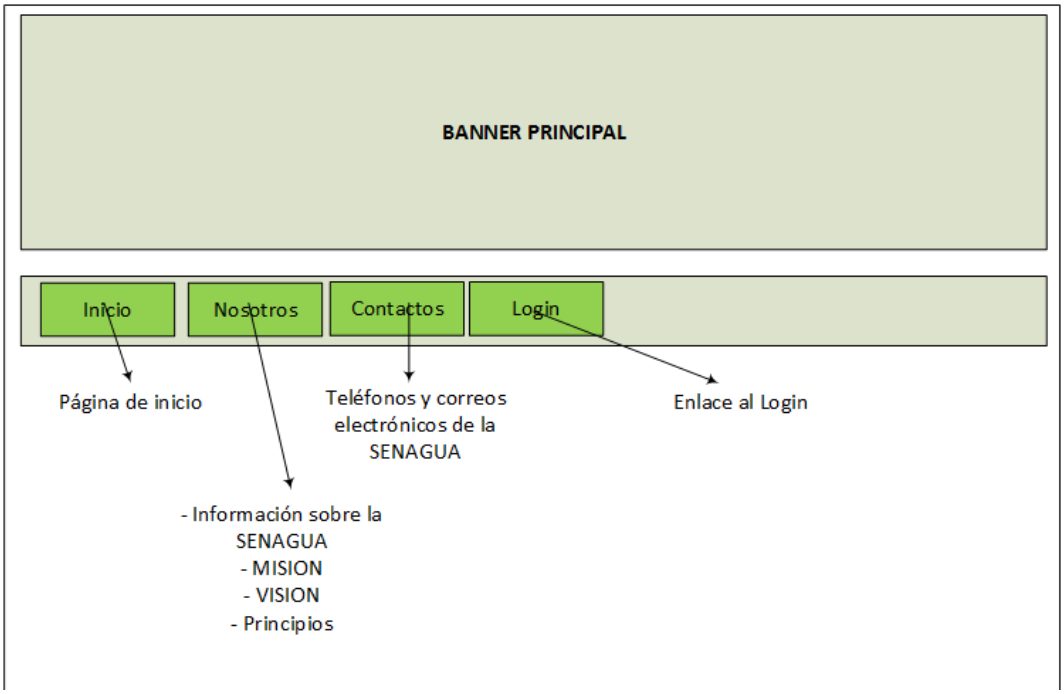


Figura 41. Diagrama de navegación del sistema.
Fuente: Autores

4.3.3.2. Mapa del sitio

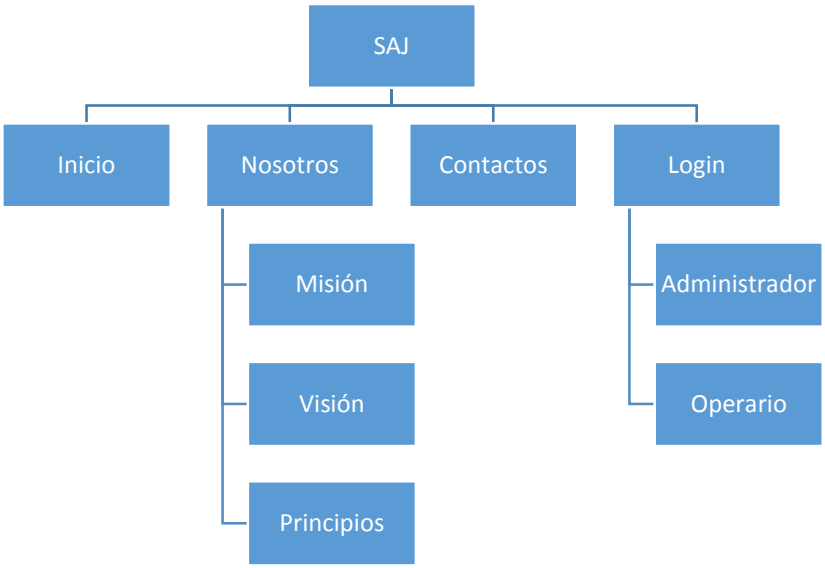


Figura 42. Mapa del Sitio
Fuente: Autores

4.4. Construcción

En esta fase se desarrollan cada uno de los módulos del sistema, empezando desde la preparación del entorno de trabajo hasta la implementación en la institución beneficiaria.

En esta fase se desarrolla el software necesario[18] llevando el diseño de la etapa anterior a un lenguaje de programación de alto nivel.

4.4.1. Preparación del entorno de trabajo

4.4.1.1. Software de Desarrollo

El desarrollo de la aplicación se lo realizará con las siguientes herramientas:

Tabla 20. Recursos software a utilizar en el desarrollo del sistema.

RECURSO DE SOFTWARE	SOFTWARE	DESCRIPCIÓN
Entorno de desarrollo integrado	Netbeans 8.02	Se utilizará para el diseñar y desarrollo de la aplicación. Entorno de formato web
Software de edición de imágenes y animaciones	Adobe Photoshop portable	Sirve para edición de imágenes para la paginas web
Sistema operativo	Ubuntu 16.04 LTS	Sistema operativo que se utiliza para las máquinas de desarrollo de la aplicación
Navegador Web	Google Chrome	Se utilizara como navegador web
Servidor de Aplicaciones	GlassFish	Servidor de Aplicaciones sobre el cual se desarrollará y posteriormente se implementará el sistema de automatización de juntas.
Gestor de bases de datos	PostgreSQL	Sistema gestor de base de datos open source mediante el cual se desarrollará la aplicación.

Fuente: Autores

[18] Kendall, K.(2005). Analisis y diseño de sistemas.Mexico: Pearson Educacion 6 Edicion.

4.4.1.2. Requerimientos de Hardware

- Requerimientos mínimos para la Instalación de GlassFish

Tabla 21. Requerimientos mínimos para instalar GlassFish

COMPONENTES	**REQUERIMIENTOS MÍNIMOS DE INSTALACIÓN PARA GLASSFISH[19]**	
	Windows	Linux
Velocidad de procesador	500MHZ	400MHZ
RAM	2GB	2GB
Disco Duro	250MB	500MB

Fuente: Autores

- Requerimientos mínimos para Instalación Java (JDK7)

Java (JDK7) es un lenguaje de programación que se utiliza en desarrollo del proyecto. En la siguiente tabla se muestran los requerimientos mínimos para su instalación

Tabla 22. Requisitos mínimos para la instalación del JDK

COMPONENTES	**REQUERIMIENTOS MÍNIMOS DE INSTALACIÓN[20]**
	Windows
Velocidad de procesador	500MHZ
RAM	1GB
Disco Duro	500MB

Fuente: Autores

- Requerimientos mínimos para Instalación PostgreSql

[19] Requerimientos de instalación de Glassfish server https://docs.oracle.com/cd/E19502-01/821-1048/abpaj/index.html

[20] Requerimientos de intalacion Java (JDK6) http://docs.oracle.com/cd/E19502-01/821-1281/abpaj/index.html

En la siguiente Tabla se muestran los requerimientos mínimos para la instalación de PostgreSql tomando en cuenta las plataformas de Windows y Linux

Tabla 23. Requerimientos mínimos para instalar PostgreSQL

COMPONENTES	REQUERIMIENTOS MÍNIMOS DE INSTALACIÓN	
	Windows	**Linux**
Velocidad de procesador	500MHZ	500MHZ
RAM	500MB	512MB
Disco Duro	250MB	250MB

Fuente: Autores

- Requerimientos mínimos para los computadores de desarrollo

Las características de hardware para las computadoras de desarrollo se han determinado examinando cada uno de los requerimientos mínimos del Software y Hardware a instalar en el equipo. Para el servidor se instala: Sistema Operativo, Java, PostgreSql, GlassFish y navegador en general.

Además se consideró aumentar un poco los requerimientos para asegurar que no haya problemas más adelante. En la siguiente Tabla se muestran los requerimientos del servidor.

Tabla 24. Requerimientos del equipo servidor.

COMPONENTES	REQUERIMIENTOS MÍNIMOS
Velocidad de procesador	1000MHZ
RAM	384MB
Disco Duro	80GB

Fuente: Autores

En el caso de las computadoras para el desarrollo se instalara: Sistema Operativo, Java, PostgreSql, GlassFish, Google Chrome y el software utilitario detallado en la siguiente tabla.

A continuación se detalla el requerimiento mínimo para las computadoras de desarrollo

Tabla 25. Requerimientos de las estaciones de trabajo.

COMPONENTES	REQUERIMIENTOS MÍNIMOS
Velocidad de procesador	Pentium 4.1.6 Ghz
RAM	1GB
Disco Duro	80GB

Fuente: Autores

4.4.2. Desarrollo de módulo y componentes

4.4.2.1. Creación de la base de datos

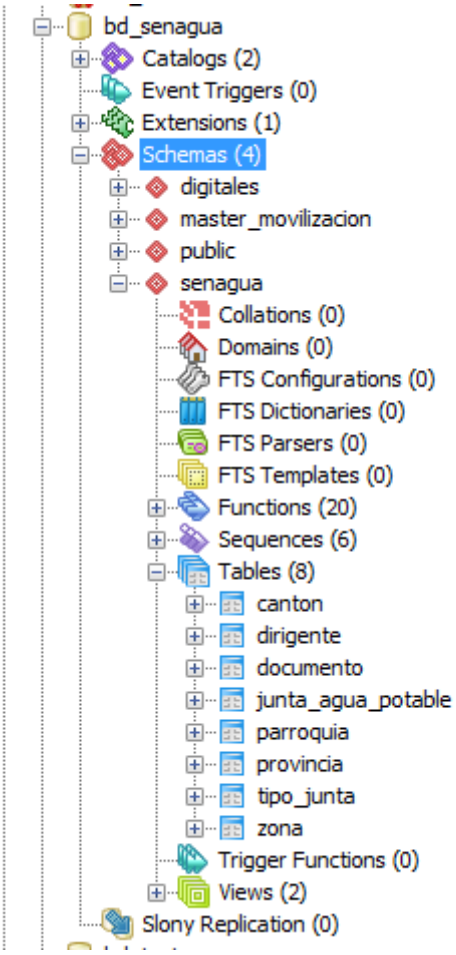


Figura 43. Base de datos del Sistema
Fuente: Autores

4.4.2.2. Desarrollo de clases

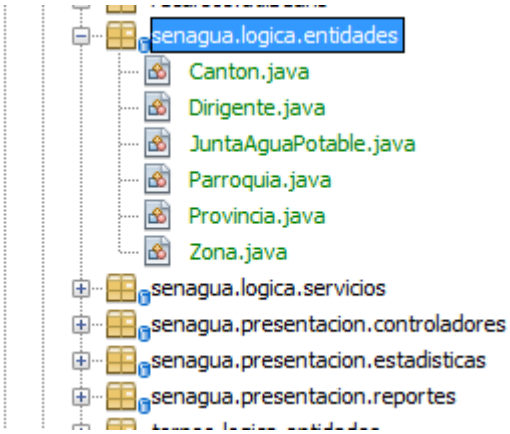


Figura 44. Entidades del sistema
Fuente: Autores

El contenido de las entidades es similar al siguiente:

```
public class JuntaAguaPotable {

    private int codigoJunta;
    private String nombreJunta;
    private String estado;
    private Parroquia parroquia;
    private Dirigente dirigente;
    private String direccion;
    private String email;
    private String telefono1;
    private String telefono2;
    private String numeroAutorizacion;
    private String numeroResolucion;
    private Date fechaRegistro;
    private String ruc;
    private Double caudalAforado;
    private Double caudalAutorizado;
    private Double consumidores;
    private Double x;
    private Double y;
    private Double z;
    private String observaciones1;
    private String observaciones2;
    private String estadoLogico;
    private String comunidad;
    private Date fechaEliminacion;

    public JuntaAguaPotable() {
    }

    public JuntaAguaPotable(int codigoJunta, String nombreJunta, String estado,
Parroquia parroquia, Dirigente dirigente, String direccion, String email, String
telefono1, String telefono2, String numeroAutorizacion, String numeroResolucion, Date
fechaRegistro, String ruc, Double caudalAforado, Double caudalAutorizado, Double
consumidores, Double x, Double y, Double z, String observaciones1, String
observaciones2, String estadoLogico, String comunidad, Date fechaEliminacion) {
        this.codigoJunta = codigoJunta;
        this.nombreJunta = nombreJunta;
```

```
        this.estado = estado;
        this.parroquia = parroquia;
        this.dirigente = dirigente;
        this.direccion = direccion;
        this.email = email;
        this.telefono1 = telefono1;
        this.telefono2 = telefono2;
        this.numeroAutorizacion = numeroAutorizacion;
        this.numeroResolucion = numeroResolucion;
        this.fechaRegistro = fechaRegistro;
        this.ruc = ruc;
        this.caudalAforado = caudalAforado;
        this.caudalAutorizado = caudalAutorizado;
        this.consumidores = consumidores;
        this.x = x;
        this.y = y;
        this.z = z;
        this.observaciones1 = observaciones1;
        this.observaciones2 = observaciones2;
        this.estadoLogico = estadoLogico;
        this.comunidad = comunidad;
        this.fechaEliminacion = fechaEliminacion;
    }

    public String getEstado() {
        return estado;
    }

    public void setEstado(String estado) {
        this.estado = estado;
    }
//Getters y setters
}
```

4.4.2.3. Desarrollo de servicios

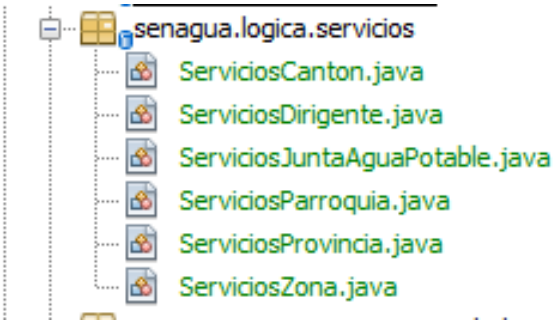


Figura 45. Capa lógica de negocio
Fuente: Autores

```
public class ServiciosJuntaAguaPotable {

    public static ArrayList<JuntaAguaPotable> llenarDatos(ConjuntoResultado
rs) throws Exception {
        ArrayList<JuntaAguaPotable> lst = new ArrayList<JuntaAguaPotable>();
        JuntaAguaPotable junta = null;
        try {
```

```
            while (rs.next()) {
                junta = new JuntaAguaPotable(
                        rs.getInt("pcodigo_junta"),
                        rs.getString("pnombre_junta"),
                        rs.getString("pestado"),

ServiciosParroquia.obtenerParroquiaDadoCodigo(rs.getInt("pcodigo_parroquia"))
,

ServiciosDirigente.obtenerDirigenteDadoCodigo(rs.getInt("pcodigo_dirigente"))
,
                        rs.getString("pdireccion"),
                        rs.getString("pcorreo_electronico"),
                        rs.getString("ptelefono1"),
                        rs.getString("ptelefono2"),
                        rs.getString("pn_autorizacion"),
                        rs.getString("pn_resolucion"),
                        rs.getDate("pfecha_registro"),
                        rs.getString("pruc"),
                        rs.getDouble("pcaudal_aforado"),
                        rs.getDouble("pcaudal_autorizado"),
                        rs.getDouble("pn_consumidores"),
                        rs.getDouble("pejex"),
                        rs.getDouble("pejey"),
                        rs.getDouble("pejex"),
                        rs.getString("pobservaciones1"),
                        rs.getString("pobservaciones2"),
                        rs.getString("pestado_logico"),
                        rs.getString("pcomunidad"),
                        rs.getDate("pfecha_eliminacion")
                );
                lst.add(junta);
            }
        } catch (Exception e) {
            lst.clear();
            throw e;
        }
        return lst;
    }

    public static ArrayList<JuntaAguaPotable> obtenerJuntas() throws
Exception {
        ArrayList<JuntaAguaPotable> lst = new ArrayList<JuntaAguaPotable>();
        try {
            String sql = "select * from senagua.f_select_juntas()";
            ConjuntoResultado rs = AccesoDatos.ejecutaQuery(sql);
            lst = llenarDatos(rs);
            rs = null;

        } catch (SQLException exConec) {
            throw new Exception(exConec.getMessage());
        }
        return lst;
    }
```

```
/*
    Más métodos de la clase
*/
}
```

4.4.2.4. Desarrollo de controladores

```
@ManagedBean
@ViewScoped
public class CtrlJuntaAguaPotable {

    private ArrayList<JuntaAguaPotable> lstJuntas;
    private ArrayList<Provincia> provincias;
    private ArrayList<Canton> cantones;
    private ArrayList<Parroquia> parroquias;
    private JuntaAguaPotable junta = new JuntaAguaPotable();
    private JuntaAguaPotable juntaSel= new JuntaAguaPotable();
    private List<Dirigente> dirigentes;
    private int codigoProvincia;
    private int codigoCanton;
    private int codigoParroquia;
    private Date fechaRegistro = new Date();
    private int codigoDirigente;

    public CtrlJuntaAguaPotable() {
        this.init();
        obtenerJuntas();
        obtenerDirigentes();
        obtenerProvincias();

    }

    private void init() {
        this.lstJuntas = new ArrayList<JuntaAguaPotable>();
        this.dirigentes = new ArrayList<Dirigente>();
        this.provincias = new ArrayList<Provincia>();
        this.cantones = new ArrayList<Canton>();
        this.parroquias = new ArrayList<Parroquia>();
    }

    public void insertarJunta(){
        try {
            junta.setFechaRegistro(Fechas.devolverFecha(fechaRegistro));

junta.setParroquia(ServiciosParroquia.obtenerParroquiaDadoCodigo(codigoParroq
uia));

junta.setDirigente(ServiciosDirigente.obtenerDirigenteDadoCodigo(codigoDirige
nte));
            if (ServiciosJuntaAguaPotable.insertarJunta(junta)) {
                this.init();

DefaultRequestContext.getCurrentInstance().execute("wdlgNuevaJunta.hide()");
```

```
                Util.addSuccessMessage("Información guardada con éxito");
                System.out.println("public void insertarEstudiante dice:
Error al guardar la información");
            } else {
                Util.addSuccessMessage("Error al guardar la información");
                System.out.println("public void insertarEstudiante dice:
Error al guardar la información");
            }
        } catch (Exception e) {
            System.out.println("insertar junta dice: "+e.getMessage());
        }
    }

    public void obtenerJuntas() {
        try {
            lstJuntas = ServiciosJuntaAguaPotable.obtenerJuntas();
        } catch (Exception e) {
            System.out.println("obtenerJuntas dice: " + e.getMessage());
        }
    }
/*
Más métodos para el controlador
*/

}
```

4.4.2.5. Desarrollo de vistas

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:p="http://primefaces.org/ui"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:f="http://java.sun.com/jsf/core">

    <h:head>
        <title>
            <ui:define name="tituloNavegador">Juntas Administradoras de Agua POtable</ui:define>
        </title>
    </h:head>
    <h:body>
        <ui:composition template="../../templates/plantillaInternaCenter.xhtml">
            <ui:define name="top">Juntas Administradoras de Agua Potable</ui:define>
            <ui:define name="contenido">
                <p:growl id="mensajes" autoUpdate="true"/>
                <h:form id="frmPrincipal">
                    <p:panel id="pnlJuntas">
                        <p:dataTable id="tblJuntas" value="#{ctrlJuntaAguaPotable.lstJuntas}"
var="junta" rowsPerPageTemplate="5,10,15,20"
                                     emptyMessage="No se han encontrado registros" paginator="true"
                                     paginatorTemplate="{FirstPageLink} {NextPageLink} {PageLinks}
{LastPageLink} {RowsPerPageDropdown}" >
                            <p:column >
                                <f:facet name="header">
                                    <p:outputLabel value="Nombre"/>
                                </f:facet>
                                <p:outputLabel value="#{junta.nombreJunta}"/>
```

```
                            </p:column>
                            <p:column >
                                <f:facet name="header">
                                    <p:outputLabel value="Cantón"/>
                                </f:facet>
                                <p:outputLabel value="#{junta.parroquia.canton.descripcion}"/>
                            </p:column>
                            <p:column >
                                <f:facet name="header">
                                    <p:outputLabel value="Parroquia"/>
                                </f:facet>
                                <p:outputLabel value="#{junta.parroquia.descripcion}"/>
                            </p:column>
                            <p:column >
                                <f:facet name="header">
                                    <p:outputLabel value="Estado"/>
                                </f:facet>
                                <p:outputLabel value="#{junta.estado}"/>
                            </p:column>
                            <p:column >
                                <f:facet name="header">
                                    <p:outputLabel value="Número Resolucion"/>
                                </f:facet>
                                <p:outputLabel value="#{junta.numeroResolucion}"/>
                            </p:column>
                            <p:column >
                                <f:facet name="header">
                                    <p:outputLabel value="Número de Autorización"/>
                                </f:facet>
                                <p:outputLabel value="#{junta.numeroAutorizacion}"/>
                            </p:column>
                            <p:column >
                                <f:facet name="header">
                                    <p:outputLabel value="Dirección"/>
                                </f:facet>
                                <p:outputLabel value="#{junta.direccion}"/>
                            </p:column>
                            <p:column >
                                <f:facet name="header">
                                    <p:outputLabel value="Dirigente"/>
                                </f:facet>
                                <p:outputLabel value="#{junta.dirigente.nombres}
#{junta.dirigente.apellidos}"/>
                            </p:column>
                            <p:column >
                                <f:facet name="header">
                                    <p:outputLabel value="Fecha de constitución"/>
                                </f:facet>
                                <p:outputLabel value="#{junta.fechaRegistro}"/>
                            </p:column>
                            <p:column exportable="false">
                                <f:facet name="header">
                                    <p:outputLabel value="Acciones"/>
                                </f:facet>
                                <p:commandButton icon="ui-icon-pencil" title="Editar"/>
                                <p:commandButton icon="ui-icon-trash" title="Eliminar"/>
                            </p:column>
                            <f:facet name="footer">
                                <p:commandButton icon="ui-icon-plus"
oncomplete="wdlgNuevaJunta.show()" title="Nueva Junta" value="Registrar"/>
                                <h:commandLink>
                                    <img src="../../resources/images/excel.png" alt=""/>
                                    <p:dataExporter type="xls" target="tblJuntas" fileName="Juntas"
/>
                                </h:commandLink>
                                <h:commandLink>
```

```
                                    <img src="../../resources/images/csv.png" alt=""/>
                                    <p:dataExporter type="csv" target="tblJuntas" fileName="Juntas"
/>
                                </h:commandLink>
                                <h:commandLink>
                                    <img src="../../resources/images/pdf.png" alt=""/>
                                    <p:dataExporter type="pdf" target="tblJuntas" fileName="Juntas"
/>
                                </h:commandLink>
                            </f:facet>
                        </p:dataTable>
                    </p:panel>
                </h:form>
            </ui:define>

            <ui:define name="dialogos">
                <p:dialog modal="true" widgetVar="dlgStatus" header="Procesando" draggable="false"
closable="false"
                          resizable="false">
                    <p:graphicImage value="/resources/images/ajaxloadingbar.gif" />
                </p:dialog>

               <!---- MÁS MÉTODOS DE LA VISTA ------>

            </ui:define>
        </ui:composition>
    </h:body>
</html>
```

Figura 46. Pantalla principal
Fuente: Autores

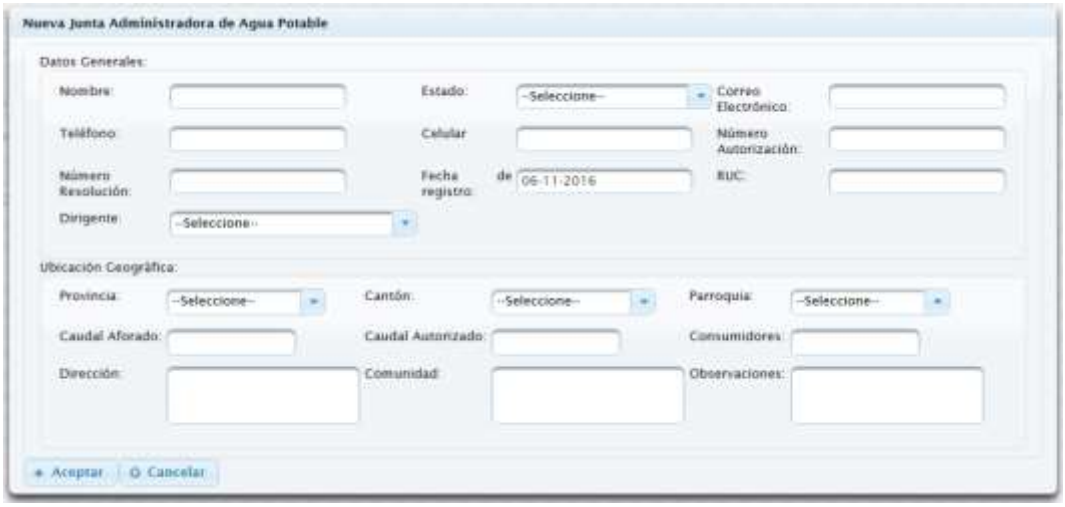
Figura 47. Vista de Login
Fuente: Autores

Figura 48. Vista principal del Módulo Gestión de Juntas de Agua Potable y Juntas de Riego
Fuente: Autores

Figura 49. Gestión de Dirigentes
Fuente: Autores

Fi**gura 50**. Formulario de registro de un nuevo dirigente
Fuente: Autores

Figura 51. Gestión de juntas Administradoras de Agua Potable y Juntas de riego
Fuente: Autores



Fi**gura 52**. Registrar una nueva junta
Fuente: Autores

Reporte disponible:			
Dirigentes	Exportar a PDF	Exportar a Excel	Exportar a Word
Juntas de Agua Potable	Exportar a PDF	Exportar a Excel	Exportar a Word

Figura 53. Formulario de reportes
Fuente: Autores

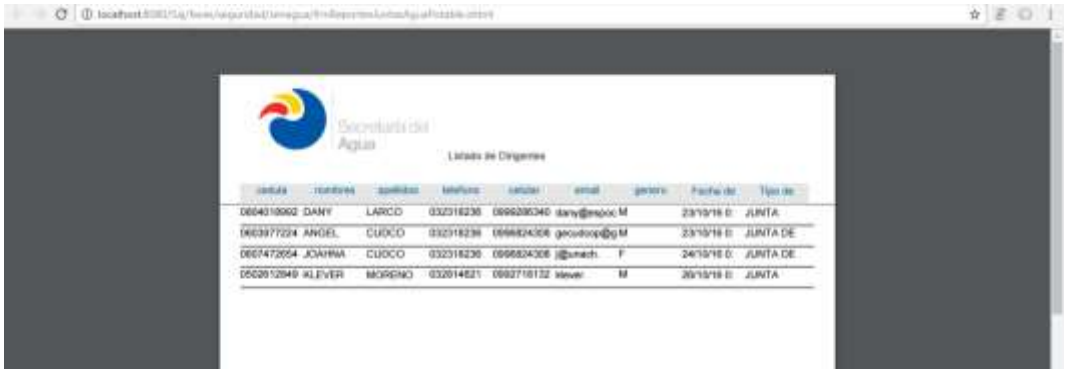


Figura 54. Vista de un reporte en la pantalla del navegador
Fuente: Autores

4.5. Implementación

Se implementó el sistema en la red local de la SENAGUA empleando un computador Dual-Core de 3.0 Ghz con 4 GB de memoria RAM bajo el sistema operativo Centos y esta accesible mediante la siguiente dirección: **192.168.10.123:8080/senagua**

CAPITULO V

METODOLOGÍA

5.1. TIPO DE ESTUDIO

Para la realización del presente trabajo se tomaron a consideración varios tipos de investigación, los mismos que se detallan a continuación:

5.1.1. Según el objeto de estudio:

- **Investigación de Campo:** debido al proceso de recolección de los requisitos de software y a la evaluación de eficiencia y satisfacción de los usuarios.

5.1.2. Según la fuente de investigación:

- **Investigación bibliográfica:** debido a los medios en los cuales está sustentada la fase teórica del presente documento, éstos medios son: libros, revistas, publicaciones, tesis, etc.

5.1.3. Según las variables:

- **Investigación Descriptiva:** debido a que mide y evalúa diversos aspectos, dimensiones o componentes del fenómeno a investigar.

5.2. MÉTODOS

- **Método Inductivo.-** Se llevará a cabo una etapa de observación y registro de los hechos. Posteriormente se realizará un análisis a fondo de los procesos y flujos de trabajo que realizan los funcionarios de la SENAGUA encargados de la gestión de las Juntas Administradoras de Agua Potable y Juntas de Riego.
- **Método Bibliográfico.-** Se determina las fuentes más importantes que proporcionen la información y documentación necesaria para entender los procesos de trabajo de la SENAGUA.

5.3. POBLACIÓN Y MUESTRA

5.3.1. Población

Las población está constituida por los funcionarios responsables de la Gestión de las Juntas Administradoras de Agua Potable y Juntas de Riego de la secretaría Nacional del Agua (SENAGUA), la misma que está compuesta por 10 personas.

5.3.2. Muestra

Al ser la población muy pequeña la muestra está constituida por el total de la población.

5.4. OPERACIONALIZACIÓN DE VARIABLES

A través de la utilización de las variables establecidas se precisan las dimensiones e indicadores que resultan relevantes para obtener el resultado esperado al momento de medir las funcionalidades del Sistema de Gestión de Juntas de Agua Potable y Juntas de Riego.

Tabla 26. Operacionalización de las Variables.

Variable	Tipo	Definición Conceptual	Dimensión	Indicadores
Los servidores de aplicaciones open source para la plataforma Java EE	Independiente	Un servidor de aplicaciones es una implementación de la especificación JEE.	WildFly GlassFish Componentes centralizados, reusables y colaborativos.	Alta disponibilidad Escalabilidad Mantenimiento
Optimización de los procesos de Gestión de Juntas Administradoras de Agua Potable y Riego de la Dirección provincial de Chimborazo de la SENAGUA	Dependiente	Capacidad de obtener resultados deseados en los recursos de información, mediante la óptima utilización de los recursos disponibles.	Acoplamiento entre módulos Independencia entre módulos Eficiencia	Usabilidad. Accesibilidad. Simplicidad. Funcionalidad. Experiencia del Usuario.

Fuente: Autores

5.5. PROCEDIMIENTOS

5.5.1. Fuentes de Información.

Entre las fuentes de información consta la Primaria y Secundaria:

a) **Primarias.-** Esta información se obtendrá basándose en la Observación y Conversación con el gerente de la empresa, trabajadores.

b) **Secundarias.-** Las fuentes secundarias se obtendrá de folletos, revistas, trípticos relativos al tema, así como del Internet.

5.5.2. Técnicas de investigación.

Las técnicas de investigación utilizadas en el presente trabajo se describen a continuación:

a) **Documental:** Permite la recopilación de información para enunciar las teorías que sustentan el estudio de los fenómenos y procesos. Incluye el uso de instrumentos definidos según la fuente documental a que hacen referencia.

b) **De Campo:** Permite la observación y el contacto directo con el objeto de estudio.

5.5.3. Instrumentos de recolección de datos.

Para la recolección de la información necesaria para la realización del presente proyecto se emplearán los siguientes instrumentos:

- ✓ **Entrevista – Encuesta:** Se realizarán encuestas y entrevistas a los técnicos de la SENAGUA para determinar si la aplicación Web cumple con las necesidades sugeridas.
- ✓ **Observación:** Se observarán los flujos de trabajo que se realizan dentro de la SENAGUA para la Gestión de Juntas Administradoras de Agua Potable y Riego.

5.6. PROCESAMIENTO Y ANÁLISIS.

5.6.1. Teoría fundamentada en datos.

La teoría fundamentada en datos es un método de investigación cualitativa que ayuda en la colecta, análisis sistemático de datos y en la generación de la teoría.

En el desarrollo de esta tesis este método se ha utilizado para precisar la colecta y el análisis general de los datos pertinentes a su ordenación en cuanto a los criterios económicos, técnicos y en cuanto al análisis de datos.

5.6.2. Análisis de tareas

En este proceso se describirá las tareas realizadas actualmente por los usuarios, sus patrones definidos de flujo de trabajo, los cuales se originan de sus esquemas mentales y las necesidades de información para realizar su trabajo. Es decir, se procura identificar "qué el usuario hace", "de qué manera lo hace", y "qué necesita para hacerlo". De esa manera, se logra el entendimiento conceptual de las tareas que deberán formar parte del sistema en desarrollo. Para la obtención de dicho entendimiento se pueden utilizar varias técnicas tales como entrevistas, observación sistemática, etc.

5.7. COMPROBACIÓN DE HIPÓTESIS

La comprobación de la hipótesis se la realiza en base a la experiencia de usuario, a través de una encuesta de satisfacción al personal que se encargará de la administración del sistema.

Para llevar a cabo el proceso de comprobación de la hipótesis a continuación se especifican la hipótesis de investigación y la hipótesis nula.

Hi: Los servidores de aplicaciones Open Source para la plataforma Java EE permitirán optimizar los procesos de Gestión de Juntas Administradoras de Agua Potable y Riego de la Dirección provincial de Chimborazo de la SENAGUA.

Ho: Los servidores de aplicaciones Open Source para la plataforma Java EE no permitirán optimizar los procesos de Gestión de Juntas Administradoras de Agua Potable y Riego de la Dirección provincial de Chimborazo de la SENAGUA.

5.7.1. Cálculos

La encuesta de satisfacción sobre el sistema se lo desarrolló en base a los indicadores de la variable dependiente, a continuación se detallan los indicadores junto con las preguntas con las que están vinculadas y las respuestas facilitadas por los funcionarios de la SENAGUA.

Tabla 27. Tabulación de las encuestas de satisfacción

Indicadores	Preguntas	F1	F2	F3	F4	F5	F6
Usabilidad	P1	3	5	4	5	4	5
	P2	4	5	4	5	4	5
Accesibilidad	P3	5	5	5	4	4	4
	P4	5	5	5	4	4	4
Simplicidad	P5	4	5	4	5	4	5
	P6	4	5	4	5	4	5
Funcionalidad	P7	3	5	5	5	4	5
	P8	4	5	5	5	4	5
Experiencia del Usuario	P9	5	5	4	4	4	5
	P10	5	5	4	4	4	5

Fuente: Autores

De acuerdo con Sommerville (2016) las pruebas serán validadas por los referentes de la institución beneficiaria del sistema, posteriormente los resultados obtenidos serán aceptados cuando posea un grado de satisfacción de al menos el 85%, para el 15% restante se definirá un plan de corrección.

Una vez que se han tabulado las encuestas de satisfacción, el siguiente paso a realizar consiste en determinar el porcentaje de aceptación de cada uno de los indicadores, en la siguiente tabla se puede observar

Tabla 28. Grado de satisfacción de los usuarios con el sistema.

Indicador	Porcentaje de Aceptación
Usabilidad	88%
Accesibilidad	90%
Simplicidad	90%
Funcionalidad	92%

Experiencia de Usuario	90%
PROMEDIO	**90%**

Fuente: Autores

5.7.2. Decisión

El grado de satisfacción de un sistema de información incide directamente en el éxito o fracaso de una institución, al tener un nivel de aceptación del 90% la hipótesis de investigación se acepta.

CAPITULO VI

CONCLUSIONES Y RECOMENDACIONES

6.1. CONCLUSIONES

- Actualmente en la SENGUA – Dirección Provincial Chimborazo la gestión de las Juntas Administradoras de Agua Potable y Juntas de Riego se realiza de forma manual donde los datos son almacenanados en hojas de cálculo, lo cual provoca dificultades al momento gestionar la información, realizar informes, reportes. A través de esta investigación se implementó un sistema informático que automatiza los procesos de gestión de las juntas de agua potable y riego así como también la generación de reportes y estadísticas.
- Los procesos y métodos de evaluación utilizados para realizar el estudio comparativo de los servidores de Aplicaciones Open Source para la plataforma Java EE permitieron determinar que el servidor más óptimo para contener el Sistema informático es el servidor GlassFish que en la estudio comparativo se observó que en promedio atiende 67,23 solicitudes por segundo, en promedio es 9 segundos más rápido que WildFly, la pérdida de paquetes es relativamente baja (0,13%) y por último la madurez de GlassFish lo hace el más usado por los desarrolladores.
- El desarrollo del Sistema informático para la automatización de los procesos de gestión de las juntas de la SENAGUA se lo realizó mediante la utilización del lenguaje de programación Java bajo entorno a través del IDE Netbeans, el servidor de base de datos PostgreSQL, el Servidor de Aplicaciones GlassFish y bajo entornos Linux, de esta manera se obtuvo una reducción en costo, libertad de uso y distribución, fomentando de esta manera la libertad de conocimiento e independencia tecnológica.

6.2. RECOMENDACIONES

- En el desarrollo de aplicaciones web se sugiere emplear frameworks que ayuden considerablemente en los procesos optimizando tiempos y líneas de código, el Servidor de Aplicaciones GlassFish optimiza el despliegue de aplicaciones hasta un 75% y la respuesta a una petición tarda aproximadamente 27,33 milisegundos.
- La utilización de las tecnologías que incorpora la plataforma EJB (Enterprise JavaBeans), JPA (Java Persistence API), CDI (Context and Dependency Injection) y JSF (Java ServerFaces) para obtener un mejor desempeño de desarrollo, manejo de estándares, limpieza de código y el fácil mantenimiento del sistema.
- Se recomienda hacer uso del patrón de diseño MVC (Model – View - Controller) proporcionado por la tecnología JSF para el desarrollo de aplicaciones web, ya que permite la reutilización de código, separando las capas de desarrollo, lo cual facilita las tareas de actualización, mantenimiento y escalabilidad.

BIBLIOGRAFÍA

Abarca C., Donoso M., Manual de Desarrollo Básico de Aplicaciones En la Plataforma J2EE en Ubuntu 7.x.

Biocca, S., et all. (2013). A Short Course on the Basic. 5a, ed., Michigan - United States. Addison-Wesley., Pag. 31-280, 371-443.

Bruce, E. (2006). Thinking in Java. 4a, ed., Massachusetts – United States., Prentice Hall., Pag. 15-91, 145-564, 725-795.

Deitel, P. (2013). Java How to program. 9a, ed., Massachusetts - United States. Prentice Hall, Pag. 71-463, 672-708, 829-930.

Fleury, M., Stark, S., Norman, R. (2005). JBoss 4.0 The Official Guide. Editorial Sams Publishing. USA.

Freeman, E. (2004). Head First Design Patterns., 1a, ed., United States., O'Really Media, Inc., Pag. 110, 281.

Gosling, J. (2005). The Javath Language Specification., 3a, ed., California – United States., Addison-Wesley., Pag. 33-73, 153-307.

Heffelfinger, D. (2007). Java EE 5 Development using GlassFish Application Server. Editorial Packt Publishing. USA.

Jendrock, E. (2013). The Java EE 6 Tutorial., 2a, ed., California - United States., Pag. 103-334, 433-687.

Lindgren, M. (2001). Application Server for E-Business. 1ra edición. Editorial Auerbach. USA.

Palacios, J. (2010). Scrum Manager: Gestión de Proyectos., 2a, ed., Creative., Pag. 44-45, 49-86.

Rumbaugh, J., Jacobson, I., Booch, G. (2000). El Proceso Unificado de Desarrollo. 1ra edición. Editorial Addison-Wesley. España.

Solfa, F. (2012). Benchmarking en el sector público., La Plata –Argentina., Pag. 9, 12-15.

Transaction Processing Performance Council (TPC); TPC Benchmark App; Versión 1.3; febrero, 2008

Sitios web

BARRY & Associates, Application Server Definition. Disponible en: http://www.servicearchitecture.com/application-servers/articles/application_server_definition.html

CSI. Consejo Superior de Administración Electrónica. (2008). Recomendaciones

Aplicables a la contratación de Equipos Informáticos. Disponible en: http://www.csi.map.es/csi/pdf/recomendaciones_contratacion.pdf

GLASSFISH WIKI. Glassfish V2 Architecture.

http://wiki.glassfish.java.net/Wiki.jsp?page=GlassFishV2Architecture.

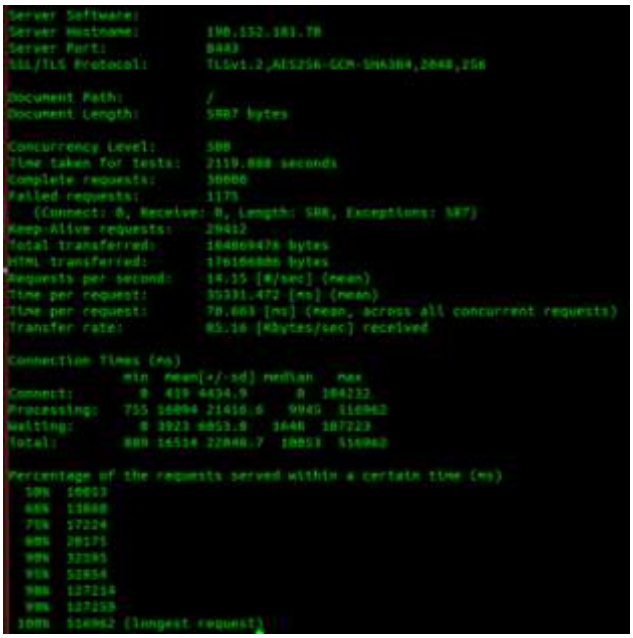
Intxauburu, M., Ochoa, C., Velasco, E., Dialnet. (2005). ¿Es el Benchmark una herramienta de aprendizaje organizacional?. Disponible en: http://dialnet.unirioja.es/servlet/fichero_articulo?codigo=2499425&orden

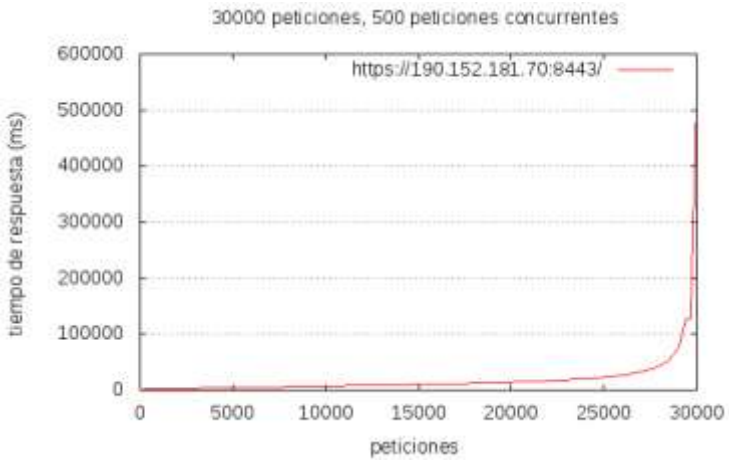
ANEXOS

Anexo 1. Ejecución de las pruebas sobre Wildfly

Nota: Para realizar las pruebas se sobrentiende que las aplicaciones han sido desplegadas previamente en el servidor.

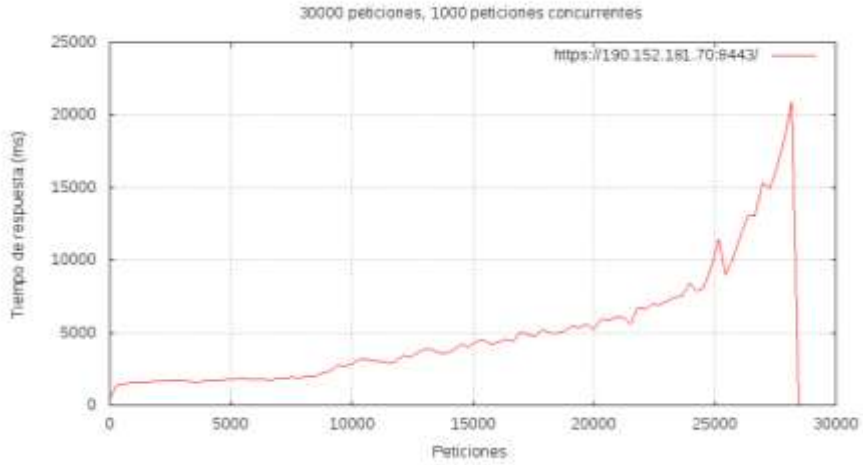
1.1) Prueba 1:



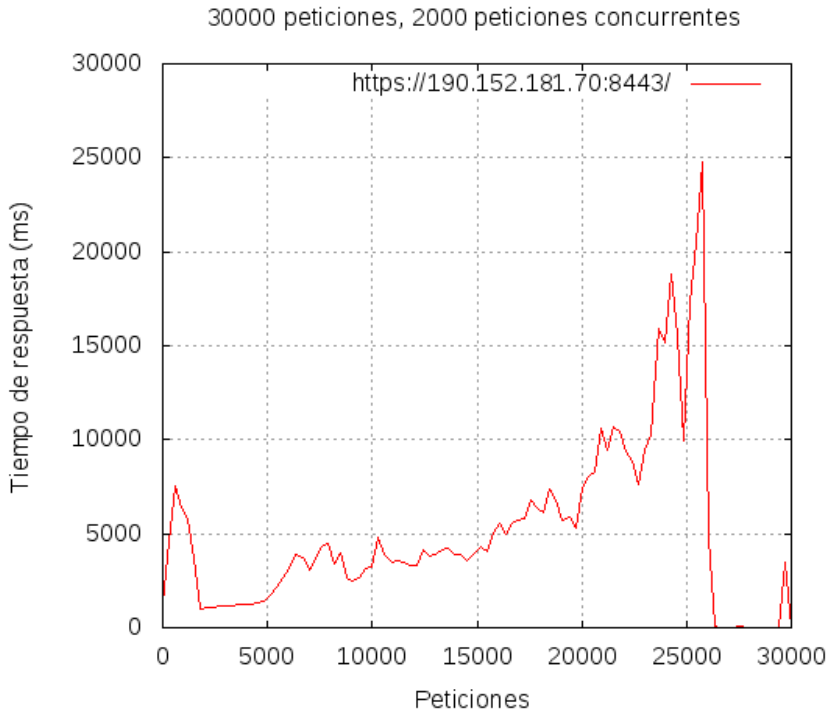


1.2) Prueba 2

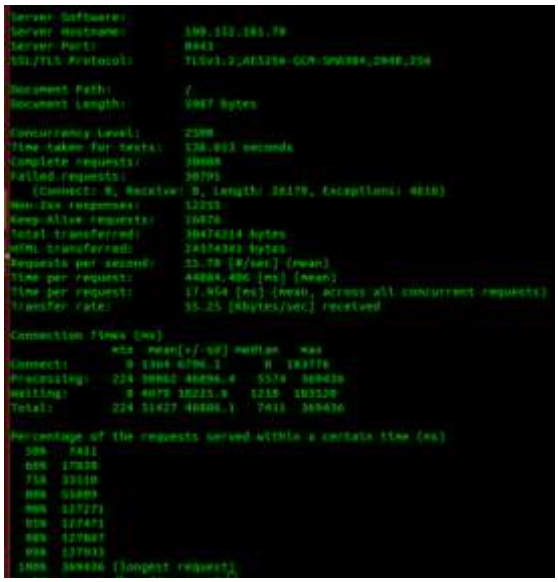
```
root@geovanny:/home/geovanny# ab -g test_2_ap1_nd.csv -n 30000 -k -c 1000 https://190.152.181.70:8443/
```

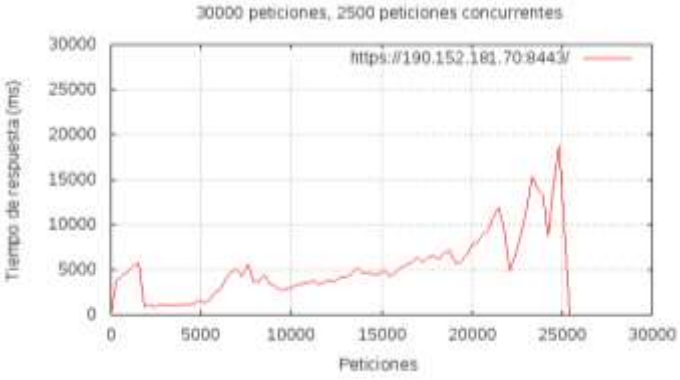


1.3) Prueba 3

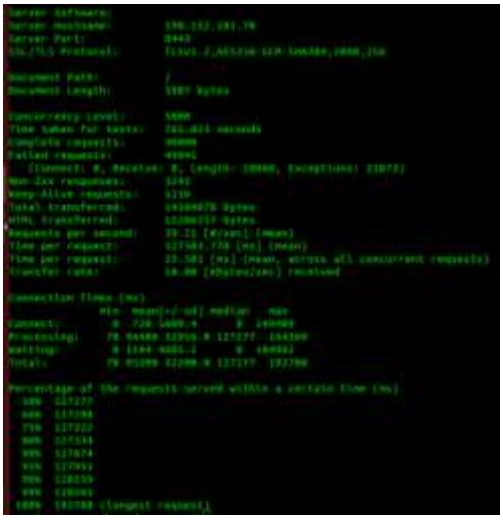


1.4) Prueba 4





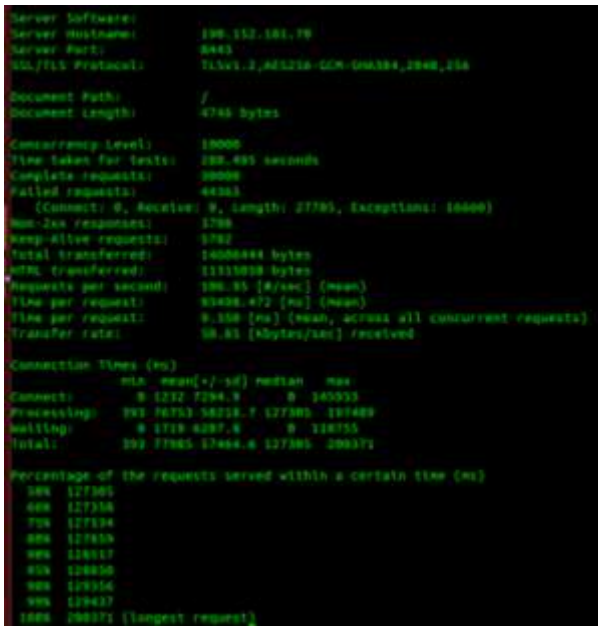
1.5) Prueba 5

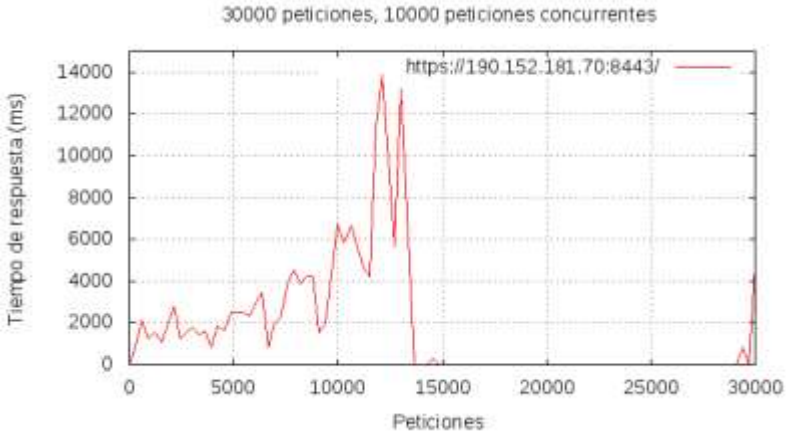



30000 peticiones, 5000 peticiones concurrentes
https://190.152.181.70:8443/
Tiempo de respuesta (ms)
Peticiones
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
0
5000
10000
15000
20000
25000
30000

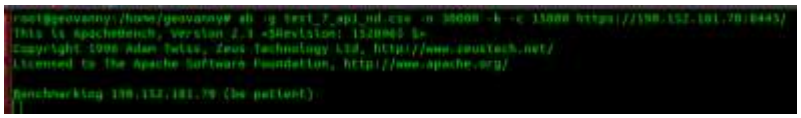
1.6) Prueba 6

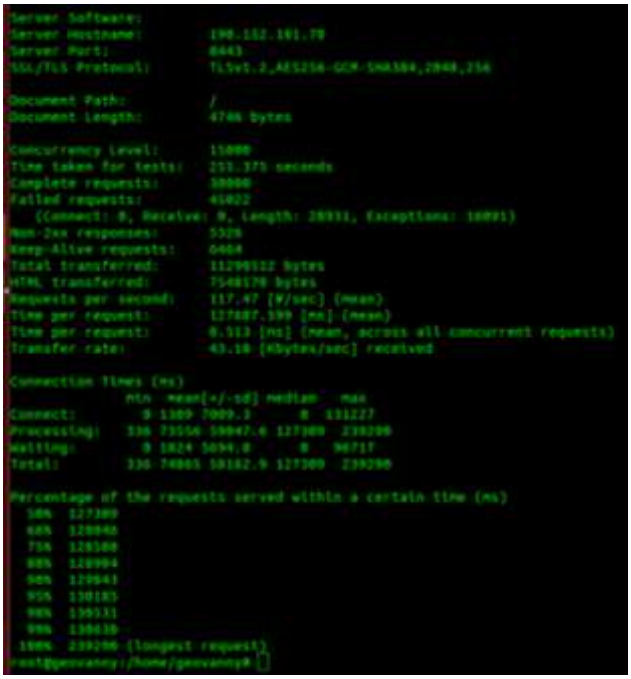

root@geovanny:/home/geovanny# ab -g test_6_ep1_nd.csv -n 30000 -k -c 10000 https://190.152.181.70:8443/
This is ApacheBench, Version 2.3 <$Revision: 1528965 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/
Benchmarking 190.152.181.70 (be patient)

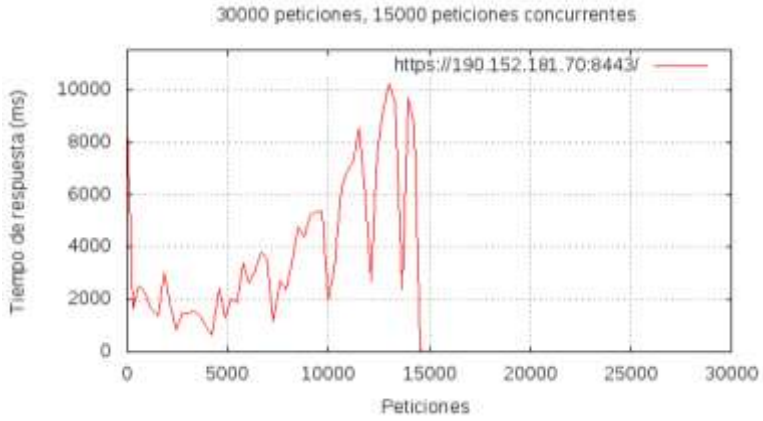

Server Software:
Server Hostname: 190.152.181.70
Server Port: 8443
SSL/TLS Protocol: TLSv1.2,AES128-GCM-SHA384,2048,256
Document Path: /
Document Length: 4746 bytes
Concurrency Level: 10000
Time taken for tests: 289.495 seconds
Complete requests: 30000
Failed requests: 44363
(Connect: 0, Receive: 0, Length: 27705, Exceptions: 16658)
Non-2xx responses: 1708
Keep-Alive requests: 1702
Total transferred: 14508444 bytes
HTML transferred: 11115038 bytes
Requests per second: 103.63 [#/sec] (mean)
Time per request: 96498.472 [ms] (mean)
Time per request: 9.650 [ms] (mean, across all concurrent requests)
Transfer rate: 50.81 [Kbytes/sec] received
Connection Times (ms)
min mean[+/-sd] median max
Connect: 0 1232 7284.9 0 145953
Processing: 193 76753 58218.7 127305 197489
Waiting: 0 1719 6287.6 0 118755
Total: 193 77985 57464.6 127305 200371
Percentage of the requests served within a certain time (ms)
50% 127305
66% 127356
75% 127534
80% 127859
90% 128517
95% 128850
98% 129356
99% 129437
100% 200371 (longest request)


30000 peticiones, 10000 peticiones concurrentes
https://190.152.181.70:8443/
Tiempo de respuesta (ms)
Peticiones
0
2000
4000
6000
8000
10000
12000
14000
0
5000
10000
15000
20000
25000
30000

1.7) Prueba 7


root@geovanny:/home/geovanny# ab -g test_7_api_nd.csv -n 30000 -k -c 15000 https://190.152.181.70:8443/
This is ApacheBench, Version 2.3 <$Revision: 1528965 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/
Benchmarking 190.152.181.70 (be patient)

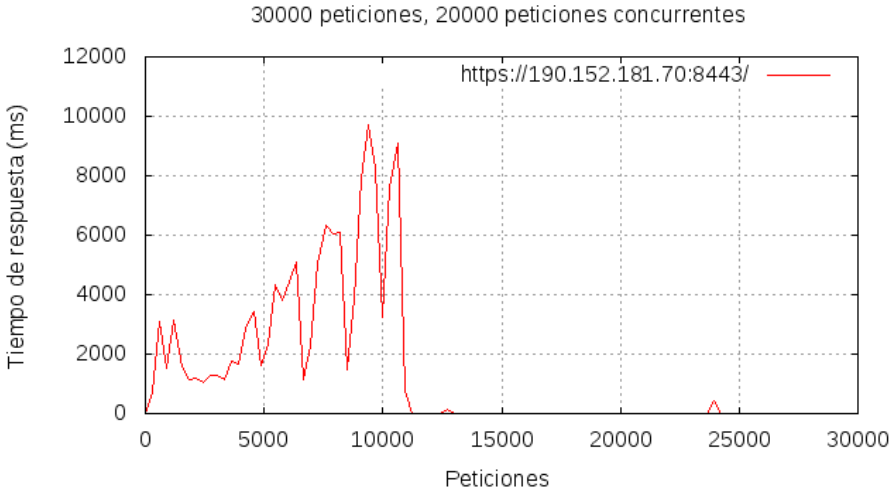

Server Software:
Server Hostname: 190.152.181.70
Server Port: 8443
SSL/TLS Protocol: TLSv1.2,AES256-GCM-SHA384,2048,256
Document Path: /
Document Length: 4746 bytes
Concurrency Level: 15000
Time taken for tests: 255.375 seconds
Complete requests: 30000
Failed requests: 45022
(Connect: 0, Receive: 0, Length: 28931, Exceptions: 16091)
Non-2xx responses: 5326
Keep-Alive requests: 6464
Total transferred: 11290512 bytes
HTML transferred: 7568170 bytes
Requests per second: 117.47 [#/sec] (mean)
Time per request: 127687.199 [ms] (mean)
Time per request: 8.513 [ms] (mean, across all concurrent requests)
Transfer rate: 43.18 [Kbytes/sec] received
Connection Times (ms)
min mean[+/-sd] median max
Connect: 0 1389 7089.3 0 131227
Processing: 336 75156 59047.6 127989 239298
Waiting: 0 1824 5094.8 0 96717
Total: 336 76865 58182.9 127989 239298
Percentage of the requests served within a certain time (ms)
50% 127989
66% 128046
75% 128108
80% 128994
90% 129041
95% 130185
98% 190511
99% 190630
100% 239298 (longest request)
root@geovanny:/home/geovanny#


30000 peticiones, 15000 peticiones concurrentes
https://190.152.181.70:8443/
Tiempo de respuesta (ms)
10000
8000
6000
4000
2000
0
0
5000
10000
15000
20000
25000
30000
Peticiones

1.8) Prueba 8

```
root@geovanny:/home/geovanny# ab -g test1_api_nd.csv -n 30000 -k -c 20000 https://190.152.181.70:8443/
```

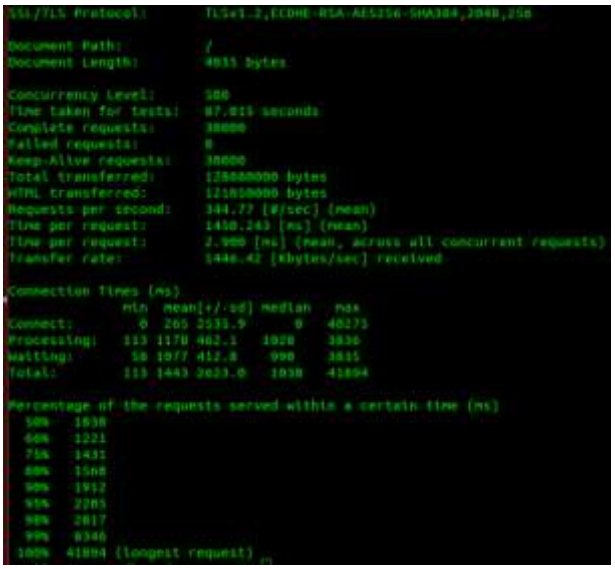
```
Server Software:
Server Hostname:        190.152.181.70
Server Port:            8443
SSL/TLS Protocol:       TLSv1.2,AES256-GCM-SHA384,2048,256

Document Path:          /
Document Length:        5087 bytes

Concurrency Level:      20000
Time taken for tests:   255.003 seconds
Complete requests:      30000
Failed requests:        40041
   (Connect: 0, Receive: 0, Length: 29911, Exceptions: 10030)
Non-2xx responses:      3540
Keep-Alive requests:    3017
Total transferred:      11606505 bytes
HTML transferred:       9437937 bytes
Requests per second:    117.65 [#/sec] (mean)
Time per request:       170002.389 [ms] (mean)
Time per request:       8.500 [ms] (mean, across all concurrent requests)
Transfer rate:          44.45 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0 1329 7489.9      0  120728
Processing:   195 80581 36212.8 120897  250432
Waiting:        0 1277 4880.2      0   95307
Total:        195 80838 35009.5 120899  250432

Percentage of the requests served within a certain time (ms)
  50%  120899
  66%  129196
  75%  129748
  80%  129971
  90%  130772
  95%  131323
  98%  131478
  99%  131588
 100%  250432 (longest request)
```

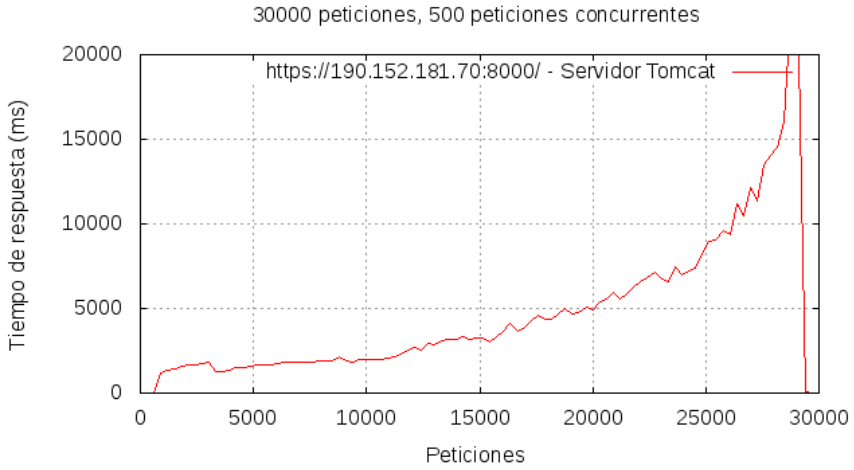


Anexo 2. Ejecución de las pruebas sobre GlassFish

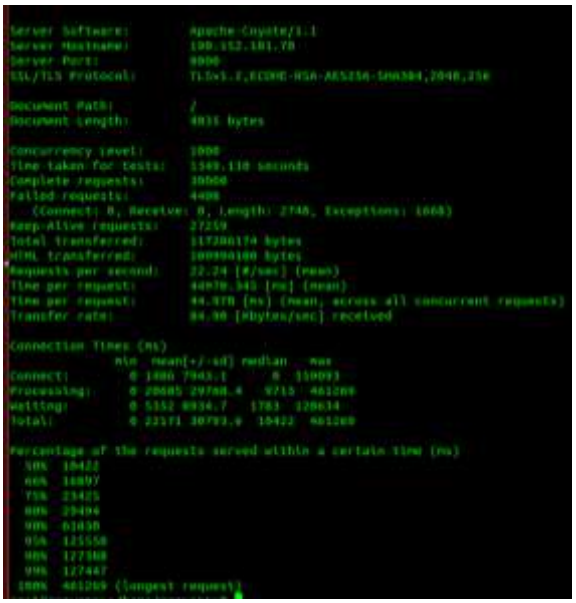
Nota: Para realizar las pruebas se sobrentiende que las aplicaciones han sido desplegadas previamente en el servidor.

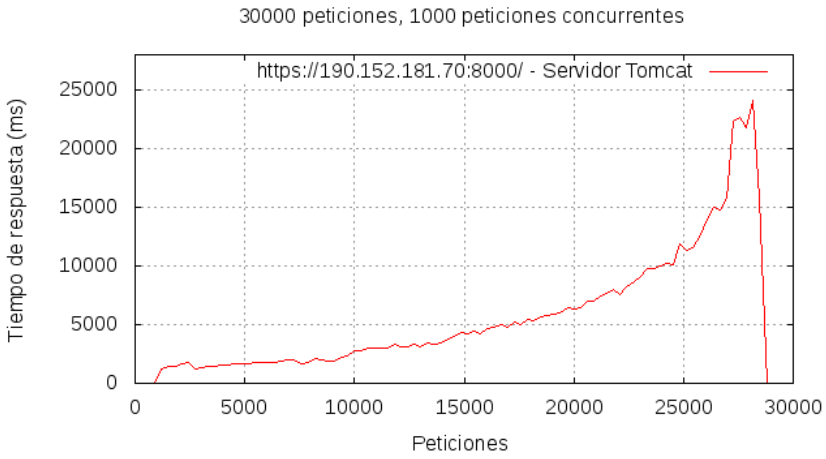
2.1) Prueba 1





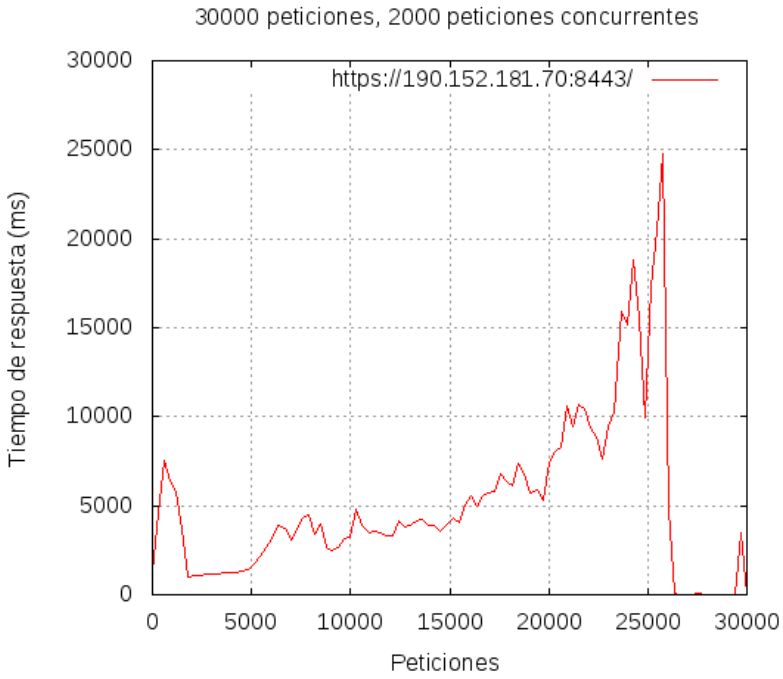
2.2) Prueba 2


Server Software: Apache-Coyote/1.1
Server Hostname: 190.152.181.70
Server Port: 8000
SSL/TLS Protocol: TLSv1.2,ECDHE-RSA-AES256-SHA384,2048,256
Document Path: /
Document Length: 4831 bytes
Concurrency Level: 1000
Time taken for tests: 1549.110 seconds
Complete requests: 30000
Failed requests: 4408
(Connect: 0, Receive: 0, Length: 2748, Exceptions: 1660)
Keep-Alive requests: 27259
Total transferred: 117300174 bytes
HTML transferred: 100990100 bytes
Requests per second: 22.24 [#/sec] (mean)
Time per request: 44970.343 [ms] (mean)
Time per request: 44.970 [ms] (mean, across all concurrent requests)
Transfer rate: 84.98 [Kbytes/sec] received
Connection Times (ms)
min mean[+/-sd] median max
Connect: 0 1886 7943.1 0 150093
Processing: 0 20685 29708.4 9715 461269
Waiting: 0 5552 8954.7 1783 128634
Total: 0 22571 30793.9 10422 461269
Percentage of the requests served within a certain time (ms)
50% 10422
66% 16897
75% 23425
80% 29494
90% 61435
95% 125558
98% 127368
99% 127447
100% 461269 (longest request)


30000 peticiones, 1000 peticiones concurrentes
https://190.152.181.70:8000/ - Servidor Tomcat
Tiempo de respuesta (ms)
25000
20000
15000
10000
5000
0
0
5000
10000
15000
20000
25000
30000
Peticiones

2.3) Prueba 3

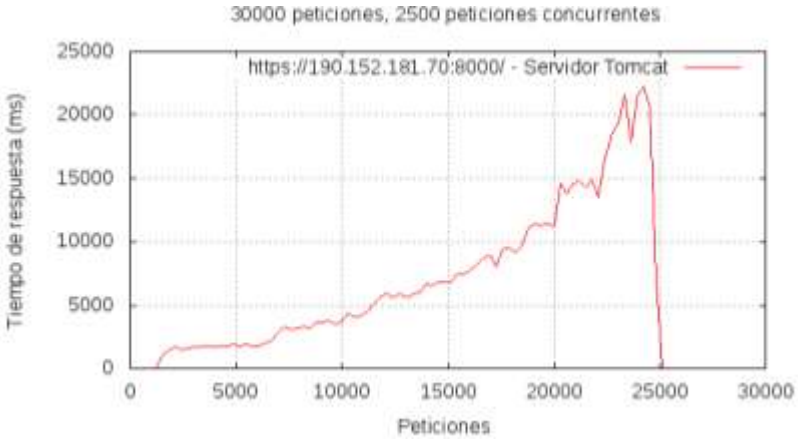

Server Software: Apache-Coyote/1.1
Server Hostname: 190.152.181.70
Server Port: 8443
SSL/TLS Protocol: TLSv1.2,ECDHE-RSA-AES256-SHA384,2048,256
Document Path: /
Document Length: 4835 bytes
Concurrency Level: 1000
Time taken for tests: 1225.245 seconds
Complete requests: 30000
Failed requests: 28200
(Connect: 0, Receive: 0, Length: 5981, Exceptions: 4359)
Keep-Alive requests: 24089
Total transferred: 163858104 bytes
HTML transferred: 97239465 bytes
Requests per second: 24.48 [#/sec] (mean)
Time per request: 81683.823 [ms] (mean)
Time per request: 40.842 [ms] (mean, across all concurrent requ
Transfer rate: 82.78 [Kbytes/sec] received
Connection Times (ms)
min mean[+/-sd] median max
Connect: 0 2611 10704.4 0 170488
Processing: 0 55075 41328.6 54198 731998
Waiting: 0 5838 18384.1 1739 185274
Total: 0 30180 44121.9 16100 731998
Percentage of the requests served within a certain time (ms)
50% 16100
66% 30618
75% 46275
80% 85318
90% 127235
95% 127319
98% 127483
99% 127558
100% 731998 (longest request)
root@geovanny:/home/geovanny#


30000 peticiones, 2000 peticiones concurrentes
https://190.152.181.70:8443/
Tiempo de respuesta (ms)
Peticiones
0
5000
10000
15000
20000
25000
30000

2.4) Prueba 4

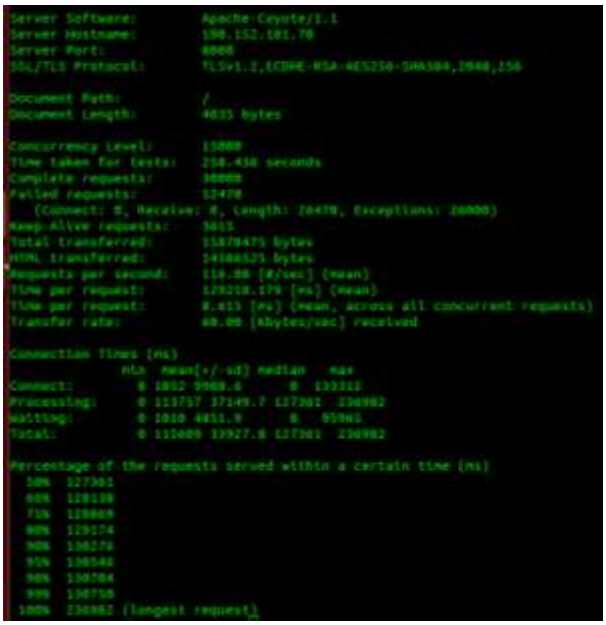
```
root@geovanny:/home/geovanny# ab -g test_4_api_java.csv -n 30000 -k -c 2500 https://190.152.181.70:8000/
This is ApacheBench, Version 2.3 <$Revision: 1528965 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 190.152.181.70 (be patient)
```

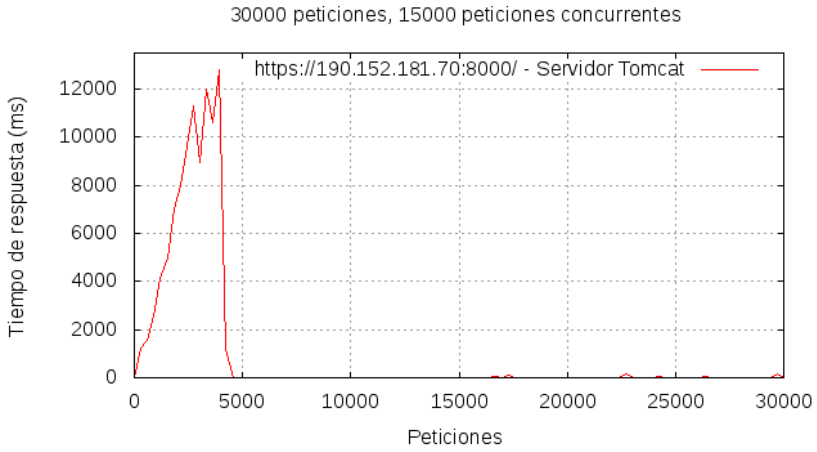
```
Server Software:        Apache-Coyote/1.1
Server Hostname:        190.152.181.70
Server Port:            8000
SSL/TLS Protocol:       TLSv1.2,ECDHE-RSA-AES256-SHA384,2048,256

Document Path:          /
Document Length:        4855 bytes

Concurrency Level:      2500
Time taken for tests:   1223.337 seconds
Complete requests:      30000
Failed requests:        12617
   (Connect: 0, Receive: 0, Length: 7128, Exceptions: 5489)
Keep-Alive requests:    22872
Total transferred:      88879627 bytes
HTML transferred:       82288528 bytes
Requests per second:    24.52 [#/sec] (mean)
Time per request:       101940.448 [ms] (mean)
Time per request:       40.778 [ms] (mean, across all concurrent requests)
Transfer rate:          70.77 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0 3132 11722.0      0  151651
Processing:     0 38888 47543.8  18956  964588
Waiting:        0 5936 18478.3   1744  385353
Total:          0 42021 47870.8  19809  964588

Percentage of the requests served within a certain time (ms)
  50%  19809
  66%  38248
  75%  64879
  80%  83954
  90%  127308
  95%  137629
  98%  137817
  99%  137877
 100%  964588 (longest request)
```



2.5) Prueba 5

2.6) Prueba 6

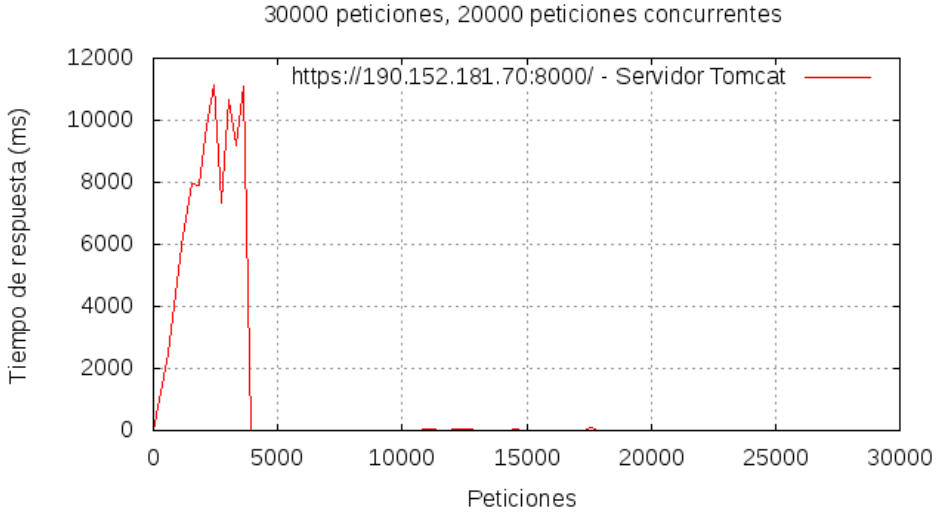




2.7) Prueba 8

```
root@geovanny:/home/geovanny# ab -g test_8_api_java.csv -n 30000 -k -c 20000 https://190.152.181.70:8000/
This is ApacheBench, Version 2.3 <$Revision: 1528965 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 190.152.181.70 (be patient)
```

```
Server Software:        Apache-Coyote/1.1
Server Hostname:        190.152.181.70
Server Port:            8000
SSL/TLS Protocol:       TLSv1.2,ECDHE-RSA-AES256-SHA384,2048,256

Document Path:          /
Document Length:        4815 bytes

Concurrency Level:      20000
Time taken for tests:   257.489 seconds
Complete requests:      30000
Failed requests:        53400
   (Connect: 0, Receive: 0, Length: 26966, Exceptions: 26434)
Keep-Alive requests:    3154
Total transferred:      13929078 bytes
HTML transferred:       12716390 bytes
Requests per second:    116.51 [#/sec] (mean)
Time per request:       171659.239 [ms] (mean)
Time per request:       8.583 [ms] (mean, across all concurrent requests)
Transfer rate:          52.83 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0 1956 10450.3      0  144488
Processing:     0 116205 35628.4 128415  179004
Waiting:        0  886 4469.0      0   94918
Total:          0 118162 31942.1 128417  224487

Percentage of the requests served within a certain time (ms)
  50%  128417
  66%  129652
  75%  130983
  80%  131139
  90%  132241
  95%  132552
  98%  132777
  99%  132953
 100%  224487 (longest request)
```



Anexo 3. Oficio de petición para la realización del módulo de gestión de juntas administradoras de riego y agua potable de la SENAGUA.

UNIVERSIDAD NACIONAL DE CHIMBORAZO
FACULTAD DE INGENIERÍA
CARRERA DE INGENIERÍA EN SISTEMAS Y COMPUTACIÓN

La Facultad de Ingeniería de la UNACH, dispone de un Sistema de Gestión certificado de acuerdo a la Norma ISO 9001 por SGS

Riobamba, 15 de septiembre del 2016

Doctor
Leonardo Velasteguí
SUBSECRETARIO DE LA DEMARCACION HIDROGRAFICA DE PASTAZA - SENAGUA
Presente.-

De mi consideración:

Reciba un saludo fraterno de quienes conformamos la Carrera de Ingeniería en Sistemas y Computación de la Universidad Nacional de Chimborazo, en relación a la reunión de trabajo realizada en el mes de Julio del 2016 con la finalidad de desarrollar el Módulo de Gestión de Juntas Administradoras de Agua Potable y Riego de la SENAGUA Zona N° 3, tengo a bien informar que mediante resolución N° 815-HCD-08-08-2016 el H. Consejo Directivo de la Facultad de Ingeniería se procede a la revisión del proyecto de investigación "ANALSIS COMPARATIVO DE SERVIDORES DE APLICACIONES OPEN SOURCE PARA LA PLATAFORMA JAVA EE CASO PRACTICO: MODULO DE GESTION DE JUNTAS ADMINISTRADORAS DE AGUA POTABLE Y RIEGO DE LA SENAGUA ZONA N° 3", a la presente fecha los estudiantes José Ortiz y Darwin Balbuca trabajaron en coordinación con el Ing. Klever Moreno para el levantamiento de información y requisitos de software, conforme el cronograma de actividades han avanzado en un 60% del proyecto. Con la finalidad de cumplir con los procedimientos académicos de la institución solicito comedidamente se digne autorizar a quién corresponda la elaboración de una carta de aceptación del proyecto de investigación antes mencionado.

Por la atención que se digne dar al presente anticipo mi agradecimiento.

Atentamente,

Ing. Jorge Delgado
DOCENTE COORDINADOR PROYECTO
ESCUELA DE INGENIERIA EN SISTEMAS
Y COMPUTACION

Campus Norte "Edison Riera R."
Avda. Antonio José de Sucre Km. 1,5 Vía a Guano
Teléfonos: (593-3)37 30 880-ext. 1414
www.unach.edu.ec
www.sistemas.unach.edu.ec
sistemas@unach.edu.ec

Anexo 4. Respuesta al oficio enviado de petición para la realización del módulo de gestión de juntas administradoras de riego y agua potable de la SENAGUA.

GOBIERNO NACIONAL DE LA REPÚBLICA DEL ECUADOR

Secretaría del Agua

Oficio Nro. SENAGUA-SDHP.18-2016-0325-O

Riobamba, 20 de septiembre de 2016

Asunto: PROYECTO DE INVESTIGACIÓN MÓDULO DE GESTIÓN DE JUNTAS ADMINISTRADORAS DE AGUA POTABLE Y RIEGO DE LA SENAGUA ZONA Nº 3

Ingeniero
Jorge Delgado
Director del Centro de Tecnologías Educativas
UNIVERSIDAD NACIONAL DE CHIMBORAZO
En su Despacho

De mi consideración:

Reciba un cordial y atento saludo, en respuesta a su oficio s/n en donde nos solicita que se indique la aceptación del proyecto de investigación **"ANALISIS COMPARADO DE SERVIDORES DE APLICACIONES OPEN SOURCE PARA LA PLATAFORMA JAVA EE CASO PRÁCTICO: MÓDULO DE GESTIÓN DE JUNTAS ADMINISTRADORAS DE AGUA POTABLE Y RIEGO DE LA SENAGUA ZONA Nº 3"** que la realizarán los estudiantes José Miguel Ortiz Ramirez y Darwin Mauricio Balbuca Ramones, debo indicar que por parte de la Demarcación Hidrográfica de Pastaza, se **AUTORIZA** la realización de dicho proyecto en coordinación con el Ing. Klever Moreno, Analista de Infraestructura DHP, para el levantamiento de información y requisitos del software.

Particular que comunico para los fines pertinentes.

Atentamente,

Documento firmado electrónicamente

Dr. Leonardo Dario Velastegui Ramos
SUBSECRETARIO DE LA DEMARCACIÓN HIDROGRÁFICA DE PASTAZA

Copia:
Señorita Ingeniera
Julia Elisa Campuzano Llaguno
Responsable de la Administración del Recurso Humano - SDH Pastaza

Señor Ingeniero
Klever Oswaldo Moreno Lema
Analista de Infraestructura en Demarcación 2 - SDH Pastaza

Anexo 5. Acta de reunión de presentación del demo del “sistema de administración de juntas de riego y drenaje y agua potable en la demarcación hidrográfica de Pastaza.

Secretaría del Agua

ACTA DE REUNIÓN PRESENTACIÓN DEL DEMO DEL "SISTEMA DE ADMINISTRACIÓN DE JUNTAS DE RIEGO Y DRENAJE Y AGUA POTABLE" EN LA DEMARCACIÓN HIDROGRÁFICA DE PASTAZA

En la ciudad de Riobamba, Provincia de Chimborazo, a los 26 días del mes de Octubre del 2016, en las oficinas de la Demarcación Hidrográfica de Pastaza, ubicadas en la calle Chile 10-51 y Darquea, se constituye por una parte el Ing. Klever Moreno Responsable de Tecnología de la Demarcación y los Señores Estudiantes de la Universidad Nacional de Chimborazo, José Miguel Ortiz Ramírez, Darwin Mauricio Balbuca Ramones y el Representante del Ing. Jorge Delgado, la revisión y observación del DEMO del "Sistema de Administración de Juntas de Riego y Drenaje Y Agua Potable", la misma que una vez revisado se realizó las observaciones correspondientes de la plataforma para la incorporación de varios ítem que requiere el sistema para un mejor control y registro de las JAAPS Y JARyD dentro de la institución.

Para lo cual se suscribe la presente Acta de presentación y para constancia y fe de lo actuado, firman las partes que intervienen en unidades.

Ing. Klever Moreno
Analista de infraestructura en demarcación 2

Ing. Carlos Romero
Director Técnico de Recursos Hídricos

Lic. Luis Agualsaca
Director de Articulación y Gestión Social

Sr. José Miguel Ortiz Ramírez
Estudiante

Sr. Darwin Mauricio Balbuca Ramones
Estudiante

Demarcación Hidrográfica de Pastaza
Telf.: + (593 3) 2960 623
www.agua.gob.ec

Anexo 6. Acta de reunión de presentación de la corrección del demo del "sistema de administración de juntas de riego y drenaje y agua potable en la demarcación hidrográfica de Pastaza.

Secretaría del Agua

ACTA DE REUNIÓN PRESENTACIÓN DEL DEMO DEL "SISTEMA DE ADMINISTRACIÓN DE JUNTAS DE RIEGO Y DRENAJE Y AGUA POTABLE" EN LA DEMARCACIÓN HIDROGRÁFICA DE PASTAZA

En la ciudad de Riobamba, Provincia de Chimborazo, a los 28 días del mes de Noviembre del 2016, en las oficinas de la Demarcación Hidrográfica de Pastaza, ubicadas en la calle Chile 10-51 y Darquea, se constituye por una parte el Ing. Klever Moreno Responsable de Tecnología de la Demarcación y los Señores Estudiantes de la Universidad Nacional de Chimborazo, José Miguel Ortiz Ramírez, Darwin Mauricio Balbuca Ramones y el Representante del Ing. Jorge Delgado, en la Presentación y revisión del "Sistema de Administración de Juntas de Riego y Drenaje Y Agua Potable", la misma que se realiza las correcciones correspondientes al Sistema antes de su entrega final.

Para lo cual se suscribe la presente Acta de presentación y para constancia y fe de lo actuado, firman las partes que intervienen en unidades.

Ing. Klever Moreno
Analista de infraestructura en demarcación 2

Ing. Edgar Asqui
Director Técnico de Agua Potable

Lic. Luis Aguaisaca
Director de Articulación y Gestión Social

Ing. Ernesto Solis
Director Técnico de Riego y Drenaje

SECRETARÍA NACIONAL DEL AGUA
TALENTO HUMANO
DEMARCACIÓN H. PASTAZA

Ing. Carlos Romero
Subsecretario de la Demarcación Hidrográfica de Pastaza Subrogante

Demarcación Hidrográfica de Pastaza
Telf.: + (593 3) 2960 623
www.agua.gob.ec

Anexo 7. Acta de entrega/recepción del sistema informático "Gestión de juntas Administradoras de Agua Potable y Riego de la SENAGUA zona 3".

ACTA DE ENTREGA / RECEPCION

El dia 18 de enero del 2017, comparecen Antropóloga Susana Pérez Subsecretaria de la Demarcación Hidrográfica de Pastaza, Ing. Kléver Moreno Analista de Infraestructura en Demarcación 2, Ing. Jorge Delgado docente de la Facultad de Ingeniería de la Universidad Nacional de Chimborazo, Sr. José Ortiz, Sr. Darwin Balbuca estudiantes de la carrera de Ingeniería en Sistemas y Computación, para la suscribir el acta de entrega y recepción definitiva del sistema informático "Gestión de Juntas Administradoras de Agua Potable y Riego de la SENAGUA Zona 3".

PRIMERA: ANTECEDENTES

En base al convenio entre la Secretaria del Agua (SENAGUA) y la Universidad Nacional de Chimborazo (UNACH) se comprometen al desarrollo del sistema informático "Gestión de Juntas Administradoras de Agua Potable y Riego de la Demarcación Hidrográfica de Pastaza", para lo cual se coordinó los requerimientos con el Ing. Kléver Moreno de la Unidad Informática de la SENAGUA.

SEGUNDA: BIENES RECIBIDOS

El sistema informático "Gestión de Juntas Administradoras de Agua Potable y Riego de la SENAGUA Zona 3", está integrado por los siguientes módulos:

- Gestión de Juntas Administradoras de Agua Potable
- Gestión de Juntas de Riego
- Gestión de dirigentes de las Juntas de Riego Repositorio de documentos de las Juntas de Agua Potable, Juntas de Riego
- Reportes y Estadísticas
- Administración máster, para gestionar las funcionalidades del sistema
- Manual de usuario

TERCERA: RECEPCION

Previa a la suscripción de la presente Acta, los representantes de la Secretaria del Agua habiendo constatado que el sistema informático "Gestión de Juntas Administradoras de Agua Potable y Riego de la Demarcación Hidrográfica de Pastaza", está instalado, y en funcionamiento, se procede a efectuar la entrega definitiva del sistema informático y suscripción de la presente Acta de Entrega / Recepción

De esta manera se da por terminada la presente acta de entrega recepción definitiva, manifestando las partes su entera conformidad en los contenidos, para lo cual firman en unidad de acto:

Antropóloga Susana Pérez
SUBSECRETARIA DE LA DEMARCACIÓN HIDROGRÁFICA DE PASTAZA

Ing. Klever Moreno
ANALISTA DE INFRAESTRUCTURA EN DEMARCACIÓN 2

Ing. Jorge Delgado
DOCENTE UNIVERSIDAD NACIONAL DE CHIMBORAZO

Sr. Darwin Balbuca
ESTUDIANTE CARRERA DE
INGENIERÍA EN SISTEMAS Y COMPUTACIÓN
UNIVERSIDAD NACIONAL DE CHIMBORAZO

Sr. José Ortiz
ESTUDIANTE CARRERA DE
INGENIERÍA EN SISTEMAS Y COMPUTACIÓN
UNIVERSIDAD NACIONAL DE CHIMBORAZO

Anexo 8. Fotografías de constancia a cerca del desarrollo y entrega del sistema.