UNIVERSIDAD NACIONAL DE CHIMBORAZO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA EN SISTEMAS Y COMPUTACIÓN

"ANÁLISIS DE LA TECNOLOGÍA ENTERPRISE JAVABEANS (EJB) PARA LA CONSTRUCCIÓN DE APLICACIONES EMPRESARIALES JEE, APLICADO AL SISTEMA DE GESTIÓN DE LABORATORIOS DE LA UNIVERSIDAD NACIONAL DE CHIMBORAZO"

TESIS DE GRADO

Previa a la obtención del título de

INGENIERO EN SISTEMAS Y COMPUTACIÓN

AUTOR (ES):

RICHAR ATILIO BUENAÑO CALDERÓN

ANGEL ESTUARDO MOYÓN AULLA

TUTOR:

Ing. Diego Palacios Mgs

RIOBAMBA-ECUADOR

2016

Los miembros del Tribunal de Graduación del proyecto de investigación de título:

"ANÁLISIS DE LA TECNOLOGÍA ENTERPRISE JAVABEANS (EJB) PARA LA CONSTRUCCIÓN DE APLICACIONES EMPRESARIALES JEE, APLICADO AL SISTEMA DE GESTIÓN DE LABORATORIOS DE LA UNIVERSIDAD NACIONAL DE CHIMBORAZO"

Presentado por:

RICHAR ATILIO BUENAÑO CALDERÓN

ANGEL ESTUARDO MOYÓN AULLA

Y dirigida por:

ING. DIEGO PALACIOS MGS

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:

Para constancia de lo expuesto firman:

Ing. Danny Velasco **Firma** ----------

Presidente del Tribunal

Ing. Diego Palacios **Firma** ----------

Miembro del Tribunal

Ing. Paúl Paguay **Firma** ----------

Miembro del Tribunal

AUTORÍA DE LA INVESTIGACIÓN

"Nosotros, Richar Atilio Buenaño Calderón y Ángel Estuardo Moyón Aulla, somos los responsables del contenido, ideas y resultados planteados en el presente proyecto de tesis, y el patrimonio intelectual del mismo pertenece a la Universidad Nacional de Chimborazo".

Richar Atilio Buenaño Calderón

Ángel Estuardo Moyón Aulla

AGRADECIMIENTO

En estos momentos de éxito quiero dar las gracias a Dios por brindarnos el don de la vida y regalarnos salud en todo este caminar, para lograr este objetivo existen muchas personas a quienes deseamos agradecer nuestros padres, amigos, profesores y en especial un profundo agradecimiento a quien nos ha dirigido en el siguiente estudio de tesis brindándonos su valioso tiempo y compartiendo con nosotros su conocimiento y de esta manera permitirnos avanzar profesionalmente en el camino de la vida.

Un reconocimiento especial a nuestro Director de Tesis, Ing. Diego Palacios Campana por su calidad humana y todo el apoyo brindado al instruirnos y guiarnos a realizar el presente trabajo investigativo. Al Ing. Paúl Paguay por su paciencia y ayuda incondicional.

DEDICATORIA

Dedicamos este estudio de tesis a nuestros padres quienes son nuestro pilar fundamental y el motor de lucha para seguir adelante, a nuestros hermanos, por brindarnos su apoyo incondicional para culminar nuestros estudios y permitirnos ser los profesionales que el día de hoy somos y a nuestros profesores, quienes compartieron sus conocimientos y experiencia en las aulas.

Richar Buenaño y Angel Moyón

INDICE GENERAL

INDICE DE TABLAS

INDICE DE ILUSTRACIONES

ÍNDICE DE ABREVIATURAS Y ACRÓNIMOS

EJB	Enterprise JavaBeans
J2EE	Java 2 Platform, Enterprise Edition
WAR	Web Application Archive
JARS	Java Archives
EAR	Enterprise Archives
API	Interfaz de programación de aplicaciones
JPA	Java Persistence API
DTIC	Dirección de Tecnologías de la Información y Comunicación
UNACH	Universidad Nacional de Chimborazo
FTP	File Transfer Protocol
GB	Gigabyte
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IIS	Internet Information Server
JAX	Java API for XML
JAXB	Java Architecture for XML Binding
JDK	Java Development Kit
JSF	Java Server Face
MVC	Model View Controller
OASIS	Advancing Open Standards for the Information Society
OASIS	Orion Academic System with Internet Services
PDF	Portable Document Format
POJO	Plain Old Java Objects
RAM	Random Access Memory

RPC	Remote Procedure Call
SAAJ	SOAP with Attachments API for Java
SOAP	Simple Object Access Protocol
SSL	Security Sockets Layer
UDDI	Universal Discovery, Description and Integration
VM	Virtual Machine
WSDL	Web Services Definition Language
WSIT	Web Service Interoperability Technologies
XHTML	eXtensible HyperText Markup Language
XML	eXtensible Markup Language
RUP	Rational Unified Process
XSD	XML Schema
GLU	Gestión de laboratorios UNACH
RUP	Proceso Rational Unificado

RESUMEN

La presente investigación tiene como objetivo un análisis comparativo de las Tecnologías de desarrollo Enterprise JavaBeans (EJB) y la Programación Tradicional para la construcción de Aplicaciones Empresariales Java Enterprise Edition (JEE), aplicado al Sistema de Gestión de Laboratorios de la Universidad Nacional de Chimborazo (UNACH), para determinar cuál de las Tecnologías de desarrollo antes mencionadas posee un mejor nivel de rendimiento.

Para su realización se creó un prototipo Gestión de Categorías para cada una de las tecnologías de desarrollo que se mencionó anteriormente, entonces se desplegaron test de carga utilizando la Herramienta JMeter para analizar y medir el desempeño del prototipo, que permita comprobar la incidencia del rendimiento de las Tecnologías de desarrollo para lo cual se utilizó estadística descriptiva y el Sistema Inteligente de Análisis Estadístico (SIAE).

De acuerdo a los resultados del test de carga y su respectivo análisis se ha llegado a obtener los siguientes resultados: La Tecnología EJB en peticiones del usuario mejoró en un 37,5%, en lo referente a carga de usuarios poseen una diferencia significativa del 0% y en el uso de hardware la tecnología EJB decrece en un 25%, comprobándose que la tecnología de desarrollo EJB incide en el rendimiento con una diferencia significativa del 8,34%. Por lo que se acepta la hipótesis; La tecnología Enterprise JavaBeans incide en el rendimiento a nivel de aplicaciones empresariales JEE frente a la tecnología de programación tradicional.

Por lo tanto se procede a desarrollar el Sistema de Gestión de Laboratorios UNACH, utilizando la metodología Proceso Racional Unificado (RUP), para el seguimiento de las iteraciones con sus respectivas historias de usuario, diagramas de clases, diagramas de casos de uso, diseños conceptuales y pruebas de aceptación en unión a la planificación, se utilizó Netbeans 8.0.2 como IDE, GlassFish Server 4.1 como servidor y PostgreSql 9.3 como motor de Base de Datos.

El desarrollo de esta aplicación es un indicador fundamental en el proceso de la Gestión de equipos y Laboratorios, permitiendo mejorar cada día la calidad Educativa en la Institución.

ABSTRACT

The objective of this research work is a comparative analysis of the EJB Enterprise JavaBeans development technologies and Traditional Programming for the construction of JEE Java Enterprise Edition Company Applications, applied to the Laboratory Management System at Universidad Nacional de Chimborazo (UNACH) in order to determine which of the development technologies mentioned has a better performance level.

For its development, a Category Management Prototype was created for every one of the development technologies mentioned before, so that a load test was opened using the JMeter Tool in order to analyze and measure the performance of the prototype, it allows checking the performance incidence of the development technologies for which descriptive statistics and the Statistical Analysis Smart System (SASS) were used.

According to the results of the load test and its corresponding analysis, it has been possible to obtain these results: The EJB technology in user requests improved 37,5%, in reference to the user load, there is an important difference of 0% and in the use of the hardware, the EJB technology decreases 25%, it was possible to check that the EJB development technology influences in its performance with an important difference of 8,34%. For this reason the hypothesis is accepted; The Enterprise JavaBeans technology influences in the performance of JEE company applications against the technology for traditional programming.

For this reason the UNACH Laboratory Management System is developed by using the Rational Unified Process (RUP) methodology for the tracking of interactions with their corresponding user's history, class diagrams, use case diagrams, conceptual designs and acceptance tests linked to planning, Netbeans 8.0.2 was used as IDE, GlassFish Server 4.1 as a server and PostgreSql 9.3 as a Database machine.

The development of this application is a fundamental indicator in the process of Equipment and Laboratory Management, since it allows improving the Education quality of the institution every day.

INTRODUCCIÓN

En la actualidad el uso de tecnologías de desarrollo para la construcción de aplicaciones empresariales JEE, se ha visto muy importante ya que estos permiten la interacción de aplicaciones web desarrollados en plataformas heterogéneas, haciendo que estas sean interoperables, escalables, fáciles de mantener e integrar con sistemas y fuentes de datos existentes. Dado estas condiciones se han desarrollado una aplicación empresarial de este tipo.

Por medio de las tecnologías de desarrollo Enterprise JavaBeans EJB se logró desarrollar una aplicación empresarial JEE, en la cual el rendimiento es superior a tecnologías de desarrollo tradicional. Dentro de este tipo de aplicaciones se puede integrar con plataformas heterogéneas.

Esta investigación pretende realizar un análisis comparativo de las tecnologías de desarrollo EJB y la Programación Tradicional, orientados al rendimiento desde el lado del cliente, para el desarrollo de aplicaciones empresariales JEE, contemplando el posterior desarrollo del Módulo de Gestión de Categorías del Sistema de Gestión de Laboratorios de la Universidad Nacional de Chimborazo (UNACH) con la tecnología de desarrollo que manifieste mejores prestaciones.

El presente proyecto de investigación está estructurado en cuatro capítulos.

En el **Capítulo I,** Trata sobre el marco referencial, en el cual se encuentra con un despliegue de manera general los antecedentes, la justificación del proyecto de tesis, los objetivos a cumplir a cabalidad y la hipótesis a demostrar con el desarrollo de la misma.

En el **Capítulo II**, se puntualizó las definiciones conceptuales de la Plataforma JEE con sus respectivas especificaciones, servidores Web como Glassfish, Apache Tomcat, JBoss y de la misma forma la tecnología Enterprise JavaBeans EJB, la misma que será enfocada en el rendimiento desde el lado del cliente.

En el **Capítulo III**, se dirige en la realización del análisis comparativo entre las tecnologías de desarrollo EJB y la Programación tradicional, enfocados en el rendimiento desde el cliente, teniendo como objetivo demostrar las fortalezas y debilidades mediante los parámetros de comparación establecidos en este capítulo.

En el **Capítulo IV**, con la tecnología de desarrollo que se seleccione en el Capítulo III, es decir, el que brinde mayores beneficios técnicos, se procederá al desarrollo del Sistema de Gestión de Laboratorios de la Universidad Nacional de Chimborazo; este capítulo se constituye por la documentación del sistema aplicado la metodología Proceso Unificado Racional (RUP) consiguiendo de esta forma relacionar lo investigativo con lo práctico.

La presente investigación finaliza emitiendo las conclusiones y recomendaciones que son resultado del trabajo realizado por parte de los investigadores, directores y colaboradores del proyecto.

El presente trabajo, servirá de soporte a la hora de elegir entre las dos tecnologías de desarrollo web mencionados anteriormente y decidir cuál es la más adecuada para el desarrollo de aplicaciones empresariales que brinden un mejor rendimiento.

CAPÍTULO I

MARCO REFERENCIAL

1.1 Antecedentes

En los últimos tiempos Java se ha convertido en un lenguaje popular para la creación de aplicaciones distribuidas. De hecho, se definió una plataforma específica para el desarrollo de aplicaciones distribuidas, JEE (Java 2 Enterprise Edition) ahora llamada JEE, que proporciona todo lo necesario para la creación de aplicaciones de gran complejidad destinadas al ámbito corporativo.

Las aplicaciones distribuidas suelen tener una estructura en la que se pueden distinguir esencialmente tres capas:

- El interface de usuario.
- La lógica de negocio.
- La capa de almacenamiento de datos.

Una de las capas más trascendentes dentro del desarrollo de aplicaciones distribuidas es la capa de lógica de negocios, que es donde estará el comportamiento o funcionalidad en sí de la aplicación; y a su vez uno de las principales componentes con el que cuenta. JEE enfocado en este ámbito, es la tecnología de Enterprise JavaBeans, que proporciona un estándar para el desarrollo de las clases que encapsulan la funcionalidad y reglas de negocio.

Los EJB son componentes del contexto de servidor que cubre la necesidad de intermediar entre la capa de interfaz de usuario y de diversos sistemas empresariales o la capas de acceso a datos.

Un EJB se ejecuta en un contenedor de EJB y ofrece al desarrollador del Bean, servicios que no necesita programar (persistencia, seguridad, transacciones, etc.)

En los últimos años la Universidad Nacional de Chimborazo en cada una de sus unidades académicas dispone de laboratorios, salas multimedia, salas de internet, talleres; para el desarrollo de actividades de apoyo académico, las unidades académicas conforme a los perfiles de las carreras proveen de las infraestructuras, recursos e insumos para su respectiva investigación, los procesos de gestión y de control de los servicios y usos de los laboratorios se los realiza en forma manual, no existe un estándar institucional, razón por la cual cada funcionario organiza la información conforme a sus necesidades y basadas en la experiencia en su puesto de trabajo. La institución no dispone de un sistema informático integrado para la gestión de servicios e información de laboratorios, motivo por la cual ha decidido desarrollar el presente sistema informático con la utilización de componentes JAVA (EJB).

1.2 Justificación

Se considera un tema de real importancia debido a que será de vital ayuda para el conocimiento de todos aquellos estudiantes y profesionales que desean incursionar en el ámbito del desarrollo utilizando componentes Enterprise JavaBeans.

La tecnología Java es una tecnología madura, extremadamente eficaz y sorprendentemente versátil, se ha convertido en un recurso escalable y posee buenas perspectivas de futuro para el desarrollo de aplicaciones.

Motivo por el cual se ha decidido desarrollar el presente tema de tesis con la utilización de componentes EJB que forma parte de la arquitectura JEE, y que proporciona al desarrollador, servicios que no necesita programar como persistencia, seguridad, control de transacciones, etc., además de permitir agrupar la lógica de negocios de una manera integrada.

Además, la Universidad Nacional de Chimborazo se encuentra en un proceso de evaluación de desempeño de carreras que deben cumplir indicadores de acreditación según el Consejo de Evaluación, Acreditación y Aseguramiento de la Calidad de la Educación Superior CEAACES, razón por la cual es necesario la implementación del Sistema de Gestión Y Control De Laboratorios en la UNACH permitirá gestionar y controlar la información de los laboratorios de una manera adecuada y dé como resultado estadísticas claras y confiables de cada uno de los mismos

1.3 Objetivos

1.3.1 Objetivo General

- Analizar la tecnología Enterprise JavaBeans (EJB) para la construcción de aplicaciones empresariales JEE, aplicado a un Sistema de Gestión y Control de Laboratorios en la Universidad Nacional de Chimborazo.

1.3.2 Objetivos Específicos

- Estudiar las características de la tecnología EJB.
- Implementar el Sistema De Gestión y Control de Laboratorios de la Universidad Nacional de Chimborazo utilizando la tecnología EJB.
- Evaluar el rendimiento del Sistema De Gestión y Control de Laboratorios de la Universidad Nacional de Chimborazo con la incorporación de componentes EJB.

1.4 Hipótesis

La utilización de la tecnología Enterprise JavaBeans (EJB) en el desarrollo de aplicaciones empresariales, incide en su rendimiento.

1.5 Delimitación

El Sistema a desarrollarse será realizado en la Universidad Nacional de Chimborazo, esta investigación será analizada, diseñada e implementada utilizando Enterprise Java Beans de acuerdo a los requerimientos específicos de los laboratorios de la Universidad Nacional de Chimborazo que necesite controlar y gestionar. La construcción del sistema de gestión y control de laboratorios realizará los siguientes procesos:

- Inventario
- Registro de uso de laboratorios
- Préstamo de equipos
- Certificados de no adeudar
- Reportes

CAPÍTULO II

FUNDAMENTACIÓN TEÓRICA

2.1. Introducción a Java

Ilustración 1: Logo de Java

Fuente: **www.java.com**

Se popularizó a partir de su primer lanzamiento que fue en 1996, con su primera versión comercial que fue la JDK 1.0. Es un Lenguaje de Programación orientado a objetos diseñado para ser multiplataforma y poder ser aplicado el mismo programa en diversos sistemas operativos. Esta característica junto con la posibilidad de utilizarlo para crear applets e insertarlos en páginas HTML, o mediante servlets y páginas jsp generar código HTML dinámico. Todo ello con la capacidad de acceder a bases de datos pagadas y libres. Java es un lenguaje muy sencillo, debido a que toda la funcionalidad se encuentra en clases que forman parte del API de java. Constantemente están saliendo al mercado de la tecnología nuevos apis, que proporcionan nuevas extensiones a las características del lenguaje JAVA. Además Java es una tecnología que se utiliza para el desarrollo de aplicaciones que convierten a la Web en un elemento más interesante y útil a la vez. Acotamos también que Java le permite jugar, cargar fotografías, chatear en línea, realizar visitas virtuales y utilizar servicios como, por ejemplo, cursos en línea, servicios bancarios en línea y mapas interactivos, que nos facilitan la vida por medio de la Internet. (Óscar Belmonte, Carlos Granell Canut, María del Carmen Erdozain Navarro, 2012)

Tipos de aplicaciones Java

- **Applet**

  Son programas java para colocar dentro de una página web los cuales pueden ser interpretados por cualquier navegador con capacidades java. Para insertar estos programas se usa una etiqueta especial (como también se insertan videos, animaciones flash u otros objetos).

  Los applets son programas independientes, pero al estar incluidos dentro de una página web las reglas de éstas le afectan. Un applet sólo puede actuar sobre el navegador. En la Actualidad mediante applets se pueden integrar en las páginas web aplicaciones multimedia avanzadas (incluso con imágenes 3D o sonido y vídeo de alta calidad). (Ed Roman, Rima Patel Sriganesh, Gerald Brose, 2007)

- **Aplicaciones de consola**

  Son programas independientes al igual que los creados con los lenguajes tradicionales.

- **Aplicaciones gráficas**

  Aquellas que utilizan las clases con capacidades gráficas (como awt (Abstract Window Toolkin - Kit de Herramientas de Ventana Abstracta) por ejemplo).

- **Servlets**

  Son aplicaciones que se ejecutan en un servidor de aplicaciones web y que como resultado de su ejecución resulta una página web. (Hernández, 2006)

- **El Kit de desarrollo Java (JDK)**

  Para desarrollar en Java hacen falta los programas que realizan el precompilado y la interpretación del código, Java Developer Kit (JDK) de Sun es el entorno más famoso y gratuito que permite la creación de los bytecodes y que incluyen herramientas con capacidad de ejecutar aplicaciones de todo tipo, que se encuentra disponible en la dirección http://java.sun.com.

Actualmente ya no se le llama así sino que se le llama SDK y en la página se referencia la plataforma en concreto. (Óscar Belmonte, Carlos Granell Canut, María del Carmen Erdozain Navarro, 2012)**.**

Versiones de Java

Como se ha mencionado anteriormente, para poder crear los bytecodes de un programa Java, hace falta el JDK de Sun. Sin embargo, Sun va renovando este kit actualizando el lenguaje. De ahí que se hable de Java 1.1, Java 1.2, etc.

En la actualidad se habla de Java 2 para indicar las mejoras en la versión. Desde la versión 1.2 del JDK, el Kit de desarrollo se llama Java 2 Developer Kit en lugar de Java Developer Kit. La última versión es la 1.4.2. Lo que ocurre con las versiones, es que para que un programa que utilice instrucciones del JDK 1.4.1, sólo funcionará si la máquina en la que se ejecuta los bytecodes dispone de un intérprete compatible con esa versión.

- **Java 1.0**

Fue la primera versión de Java y propuso el marco general en el que se desenvuelve Java, está oficialmente obsoleto, pero hay todavía muchos clientes con esta versión.

- **Java 1.1**

Mejoró la versión anterior incorporando las siguientes actualizaciones:

  - El paquete **AWT** que permite crear interfaces gráficos de usuario, GUI.
  - **JDBC** que es por ejemplo. Es soportado de forma nativa tanto por Internet Explorer como por Netscape Navigator.
  - **RMI** llamadas a métodos remotos. Se utilizan por ejemplo para llamar a métodos de objetos alojados en servidor.
  - Internacionalización para crear programas adaptables a diferentes idiomas.

- **Java 2**

Apareció en Diciembre de 1998 al aparecer el JDK 1.2. Incorpora notables mejoras como por ejemplo:

- **JFC.** Java Foundation classes. El conjunto de clases de todo para crear programas más atractivos de todo tipo. Dentro de este conjunto están:
    - El paquete Swing. Sin duda la mejora más importante, este paquete permite realizar lo mismo que AWT pero superándole ampliamente.
    - Java Media

- **Enterprise Java beans.** Para la creación de componentes para aplicaciones distribuidas del lado del servidor

- **Java Media.** Conjunto de paquetes para crear paquetes multimedia:
    - Java 2D. Paquete (parte de JFC) que permite crear gráficos de alta calidad en los programas de Java.
    - Java 3D. Paquete (parte de JFC) que permite crear gráficos tridimensionales.
    - Java Media Framework. Paquete marco para elementos multimedia
    - Java Speech. Reconocimiento de voz.
    - Java Sound. Audio de alta calidad
    - Java TV. Televisión interactiva

- **JNDI.** Java Naming and Directory Interface. Servicio general de búsqueda de recursos. Integra los servicios de búsqueda más populares (como LDAP por ejemplo).

- **Java Servlets.** Herramienta para crear aplicaciones de servidor web (y también otros tipos de aplicaciones).

- **Java Cryptography.** Algoritmos para encriptar.

- **Java Help**. Creación de sistemas de ayuda.

- **Jini.** Permite la programación de electrodomésticos.

- **Java card.** Versión de Java dirigida a pequeños dispositivos electrónicos.

- **Java 1.3 y 1.4**
  Son las últimas versiones de Java 2 que incorporan algunas mejoras. (Sánchez, 2004)

Plataformas

Actualmente hay tres ediciones de la plataforma Java 2

- **J2SE**

  Se denomina así al entorno de Sun relacionado con la creación de aplicaciones y applets en lenguaje Java. La última versión del kit de desarrollo de este entorno es el J2SE 1.4.2.

- **JEE**

  Pensada para la creación de aplicaciones Java empresariales y del lado del servidor. Su última versión es la 1.4

- **J2ME**

  Pensada para la creación de aplicaciones Java para dispositivos móviles. (Sánchez, 2004)

2.2. La plataforma Java Enterprise Edition (JEE)

Java Enterprise Edition JEE anteriormente conocida como JEE (Java Second Enterprise Edition) fue desarrollada por Sun Microsystems, iniciando con la liberación de JEE 1.3, la especificación fue desarrollada bajo el Java Community Process.

JEE 1.4 fue liberado en diciembre de 2002, posteriormente Java EE 5 fue desarrollada bajo el JSR 244, su liberación se produce el 11 de mayo de 2006; y finalmente Java EE 6 que fue desarrollada bajo el JSR 316 con su liberación el 10 de diciembre de 2009. (Yerovi, 2013)

La misión de JEE es proporcionar una plataforma independiente, portátil, multiusuario, seguro y el nivel plataforma de clase empresarial para el lado del servidor escritos en el lenguaje Java. JEE es una especificación, no un producto. JEE especifica las reglas de enfrentamiento que las personas deben estar de acuerdo en la hora de escribir software empresarial. Debido a que es una especificación JEE (destinado a atender las necesidades de muchas compa-NEI), no está intrínsecamente ligada a un proveedor; también soporta multi-plataforma desarrollo. Esto anima a los proveedores para competir, dando mejor clase de sus productos. También tiene su lado negativo, que es que las incompatibilidades entre proveedor-productos surgirá algunos problemas debido a las ambigüedades con las especificaciones, otros problemas debido a la naturaleza humana de la competencia.

JEE es una de tres plataformas Java diferentes. Cada plataforma es un súper conjunto conceptual de la próxima plataforma más pequeña. (Villacrés, 2011)

2.2.1. Arquitectura aplicaciones JEE

Conjuntamente se inician el modelo JEE y el lenguaje de programación Java y la Máquina Virtual de Java. La arquitectura JEE utiliza un modelo de aplicación de múltiples capas para aplicaciones empresariales, tales como la Capa de cliente, la capa web, la capa de negocio y la capa de datos, como visualizamos en el siguiente diagrama. (Aumaille, 2002)

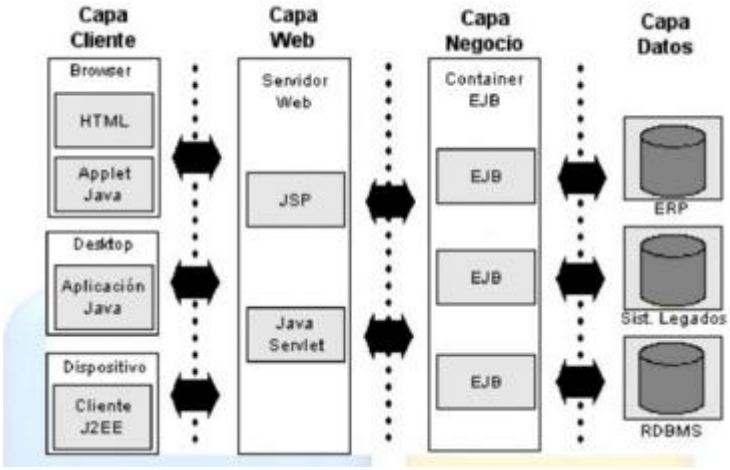


Ilustración 2: Arquitectura JEE

Fuente: Java Revolutions, Sergio Ríos

- **Capa Cliente**

  La capa de cliente está formada por la lógica de la aplicación a la que el usuario final accede directamente mediante una interfaz de usuario. La plataforma JEE tiene sustentáculo para diferentes tipos de clientes conteniendo clientes HTML, applets Java y aplicaciones Java. La lógica de la capa de cliente podría contener clientes basados en navegadores, componentes de Java que se ejecuten en un equipo de escritorio o clientes móviles de Java™ Platform, Micro Edition (plataforma J2ME™) que se ejecuten en un dispositivo móvil. (Acosta, 2013)

- **Capa Web**

  Esta capa se encuentra en el servidor web y contiene la lógica de presentación que se utiliza para generar una respuesta al cliente. Recibe los datos del usuario desde la capa cliente y basado en éstos genera una respuesta apropiada a la solicitud. En la plataforma

JEE encontraremos componentes tales como JSP'S (Java Server Pages), Servlets, JSF (Java Server Faces), clases de modelo, clases de dominio, Java Beans (Un Java Bean no es igual que un EJB (Enterprise Java Bean), son componentes totalmente diferentes), estos componentes siempre se ejecutan del lado del servidor web, como por ejemplo: en un TomCat, en un IIS, en un HTTP Server de IBM, en un Apache, etc. Cabe resaltar que tampoco en esta capa encontramos acceso directo a la capa de datos. (Acosta, 2013)

- **Capa de Negocio**

En la capa de negocio encontraremos las clases de dominio, (las cuales pueden vivir del lado de la capa web como de la capa de negocio las cuales son copias o clones para poder interactuar), por ejemplo existen clases de servicio, negocio, entidad, acceso a la base datos, clases que consumen web services, clases que invocan a otros aplicativos y esperan una respuesta o están listos para alguna función, las mismas que ejecutamos en un servidor de aplicaciones, tomando en cuenta que en una aplicación JEE existen dos servidores; tales como es el servidor de aplicaciones que procesan transacciones, ejemplos de servidores de aplicaciones: Websphere AplicationServer, Jboss, GlassFish y el servidor web que solo recibe, procesa y regresa peticiones en HTML. Esta capa se encuentra alojada en el servidor de aplicaciones y contiene el núcleo de la lógica del negocio de la aplicación. Sirve como proveedor de las interfaces necesarias para utilizar el servicio de componentes del negocio. Los componentes de negocio interactúan con la capa de datos y son implementados como componentes EJB. (Yerovi, ESTUDIO DE PATRONES DE DISEÑO EN PLATAFORMA JAVA ENTERPRISE , 2013)

- **Capa de Datos**

En la capa de datos encontramos componentes tales como servidores de bases de datos, CRM´S, Sistemas externos, ERP´S, esta capa es externa a nuestro sistema Web, siendo así la capa responsable del sistema de información de la empresa que incluye bases de datos. La única relación que se da de forma indirecta, es la que una base de datos puede ser consumida desde nuestra aplicación Web JEE como desde otra aplicación realizada en otra tecnología, es decir aquí es el punto donde las aplicaciones JEE se integran con otros sistemas no JEE o con sistemas legados. **(**Acosta**,** 2013**)**

2.2.1.1. Modelo de capas

La plataforma JEE acoge un modelo de construcción distribuido de aplicaciones multinivel para aplicaciones empresariales, donde la lógica de la aplicación se clasifica en módulos según

su función. Maneja una lógica de programación en niveles o capa, que admite encapsular y separar cada elemento de la aplicación en partes muy bien definidas para determinada función con el objetivo de simplificar el desarrollo y la comunicación entre capas, esclareciendo así las responsabilidades de cada uno de los componentes de una aplicación.
En un entorno sin niveles o capas, el desarrollo de la programación entremezclaba todas las tareas del lado del servidor como mostraremos en la siguiente imagen. (Villacrés, 2011)

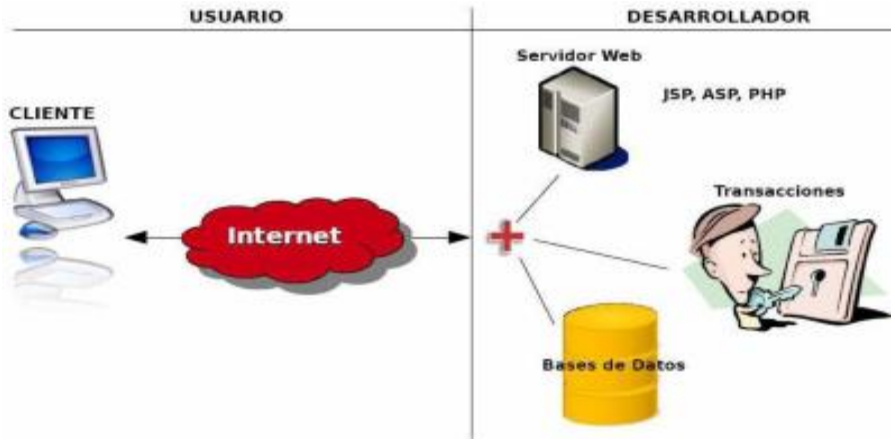


Ilustración 3: Modelo de Aplicaciones sin Capas

Fuente: Maricruz de Lourdes Acosta Yerovi

Como está en la ilustración 3. El servidor web es el delegado de realizar todos los procesos lógicos tales como crear consultas, crear formularios, ejecutar procesos, etc, un servidor web frecuentemente utilizado es Apache. En el Servidor de Bases de Datos tenemos todas las tablas, vistas y consultas que requerimos para nuestro sistema, un servidor común es PostgreSql. Las transacciones y la seguridad van juntos siempre, porque contienen la lógica de negocio entre la aplicación y el servidor de Base de Datos.

Los aplicativos JEE se basan en el tipo de aplicaciones en niveles o por capas en donde, la capa Web contiene el Servidor Web y las transacciones se realizan en la capa de Negocio con lo que las operaciones de negocio y generación de HTML dinámico que se despachan al navegador se encuentran apartadas, permitiendo un mejor control de los elementos que intervendrán en el desarrollo de aplicaciones web. **(Acosta, 2013)**

Aplicaciones Distribuidas Multicapa

La plataforma JEE™ utiliza un modelo de aplicación distribuida multi-capa. Como se manifestó anteriormente la lógica de la aplicación está dividida en componentes según sus cargos y los diferentes componentes de aplicación que componen una aplicación JEE se

instalan sobre diferentes equipos, dependiendo de la capa en el entorno JEE multicapa a la que es asignado el componente.

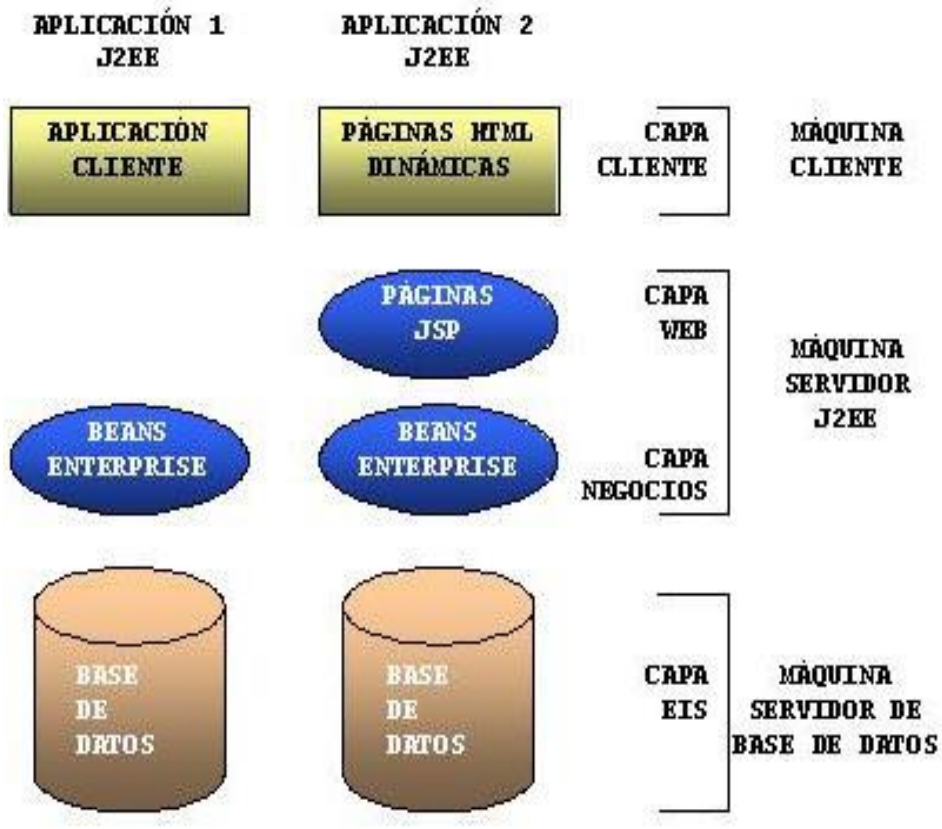


Ilustración 4: Aplicaciones Multicapa

Fuente: Maricruz de Lourdes Acosta Yerovi

La imagen cuatro nos muestra dos aplicaciones JEE multicapa divididas en las siguientes capas:

- Los elementos de la capa Cliente se ejecutan en la máquina cliente.
- Los elementos de la capa Web se ejecutan sobre el servidor JEE.
- Los componentes de la capa de Negocio se ejecutan sobre el servidor JEE.
- La capa EIS (Enterprise Information Server) se ejecuta sobre el servidor EIS.

Sin embargo un aplicativo puede estar formado de tres o cuatro capas, como se muestra en la figura cuatro. Las aplicaciones multicapa JEE normalmente se consideran de tres capas, las cuales están divididas en tres bloques diferentes:

Las máquinas clientes, la máquina servidor JEE y la máquina servidor de bases de datos. La plataforma JEE amplía el estándar de dos capas cliente-servidor, situando un servidor de aplicaciones entre la aplicación cliente y el servidor de base de datos. (Villacrés, 2011)

2.2.1.2. Componentes JEE

Las aplicaciones JEE están creadas de componentes, por lo tanto un componente JEE es una unidad de software funcional auto-contenido que se acopla en una aplicación JEE con sus clases y archivos relacionados y que interactúa con otros componentes. La especificación JEE define los siguientes componentes:

- Los aplicativos clientes y los applets son componentes clientes.
- Las tecnologías de componentes Servlet Java y JavaServer Pages (JSP) son componentes Web.
- Los componentes Enterprise JavaBeans (EJB) (beans enterprise) son componentes de la capa de negocio.

Se da a entender que los componentes JEE están programados en lenguaje Java™ y que se ejecutan de manera normal como cualquier otro lenguaje, cabe destacar que hay una gran diferencia que cuando trabajamos con la plataforma JEE, es que los componentes JEE se ensamblan en una aplicación JEE, se comprueba que están bien formados y que plasman con la especificación JEE y son desplegados en ambientes de producción donde son ejecutados y operados por el servidor JEE. (Roldán, 2015)

2.2.1.3. Contenedores JEE

Los contenedores son la interfaz entre un componente y la funcionalidad de una plataforma específica de bajo nivel que soporta ese componente. Por ejemplo, antes de ejecutar un componente Web, un bean enterprise o un componente de una aplicación cliente, debe ensamblarse dentro de una aplicación JEE y desplegarse dentro de su contenedor.

Los contenedores JEE proveen acceso a los servicios inferiores del entorno del Servidor JEE, mediante contenedores para diferentes paradigmas de componentes. Los mismos que proveen manejo de transacciones, manejo del estado, multi-threads, almacenamiento de recursos, etc., permitiendo que te puedas concentrar en resolver los problemas de negocio.

Al momento de ensamblar se da a especificar las configuraciones del servidor para cada componente de la aplicación JEE y para la propia aplicación JEE. Estas configuraciones personalizan el soporte subyacente proporcionado por el servidor JEE, que incluye servicios como JNI, seguridad, control de transacciones, etc.

Por la tanto un contenedor se consideraría como un entorno de ejecución para los componentes. **(Villacrés, 2011)**

2.2.1.4. Tipos de contenedores JEE

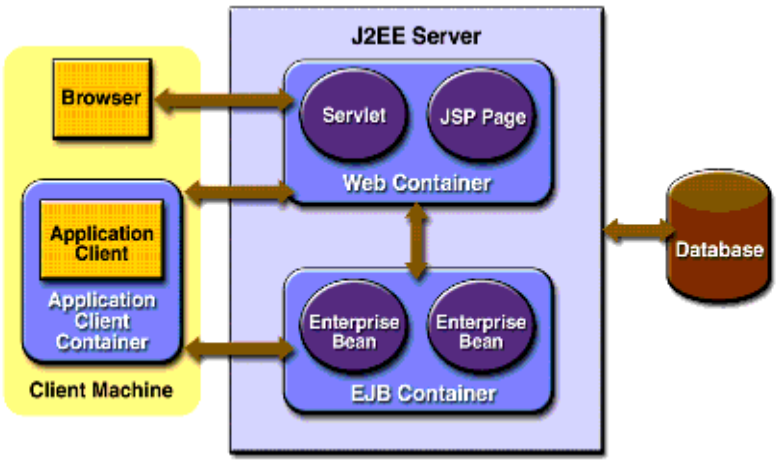


Ilustración 5: Contenedores y Servidores JEE

Fuente: www.epidataconsulting.com

En la imagen cinco, se puede observar los siguientes contenedores.

Servidor JEE: Es el ambiente en donde se ejecuta un producto JEE. Un servidor JEE proporciona contenedores EJB y Web

Contenedor Enterprise JavaBeans (EJB): Gestiona la ejecución de los beans empresariales, para las aplicaciones JEE. Los enterprise beans y el contenedor EJB corren en un servidor JEE.

Contenedor Web: Preside la ejecución de las páginas Web, servlets y algunos componentes EJB de aplicaciones JEE. Los componentes web y el contenedor web corren en el servidor JEE.

Contenedor de aplicaciones cliente: Administra la ejecución de los componentes de la aplicación cliente, esta aplicación y su contenedor se ejecutan en el cliente.

Contenedor de Applet: Gestiona la ejecución de los applets. Consiste en el navegador web y el Java Plug-in que se ejecutan juntos en el cliente. (Eric Armstrong, Jennifer Ball, Stephanie Bodoff, Debbie Carson, Ian Evans, Maydene Fisher, Dale Green, Kim Haase, Eric Jendrock., 2003)

2.2.1.5. Características y servicios de JEE

El objetivo primordial de la tecnología JEE es minimizar el desarrollo de aplicaciones empresariales al proporcionar una base común para los diferentes tipos de componentes de la plataforma JEE, por lo cual se beneficia a los desarrolladores con menos configuraciones XML, más clases simples (pojos) y empaquetado simplificado, permitiendo así mejoras de productividad con más anotaciones

La primera de las características en la plataforma JEE es los Perfiles, los mismos que son configuraciones regidas a prototipos específicos de aplicaciones. La plataforma JEE implanta un perfil web liviano encaminado a aplicaciones web de última generación, así como un perfil perfecto que contiene todas las tecnologías JEE y brinda toda la eficacia de la plataforma JEE para aplicaciones corporativas.

La segunda característica es que existen Tecnologías, entre ellas:

- Inyección de Dependencia para Java.
- API de Java para servicios web RESTFUL (JAX-RS).
- Beans gestionados.
- Contextos e Inyección de Dependencia para la plataforma Java EE, conocida como CDI.

A continuación se describen los siguientes servicios estándares que JEE específica y que se pueden emplear en cualquier servidor de aplicaciones que siga este estándar.

- **Java Message Service (JMS):** Es un servicio de mensajería Java, permite a los componentes crear, enviar, recibir y leer mensajes por medio de un un *Message-Oriented Middleware (MOM)* en una forma independiente al proveedor. Además hace viable la comunicación confiable de manera síncrona y asíncrona permitiendo la mensajería del tipo punto a punto y del tipo publicar/suscribir entre sistemas.

- **Java Transaction API (JTA):** Es la API que permite el manejo de transacciones que define las interfaces entre el administrador de transacciones y las partes involucradas en un sistema de transacciones distribuidas. Las aplicaciones logran utilizar JTA para iniciar, cerrar o abortar transacciones. Además admite al contenedor mantener una comunicación con monitores transaccionales y administradores de recursos.

- **Arquitectura de conectores JEE (JCA):** Una Interfaz de Programación de Aplicaciones de JEE que permite añadir recursos nuevos a cualquier producto JEE. La arquitectura de conector define un acuerdo entre un servidor JEE y un adaptador de recursos para permitir que se agreguen nuevos recursos.

- **Java Persistente API (JPA):** Es la API de persistencia, administra datos relacionales en aplicaciones usando la Plataforma Java en sus ediciones Standard (Java SE) y Enterprise.

- ***Java Naming Direct Interface (*JNDI*):*** Es una API estándar para el registro y acceso de servicios y objetos, que permite a los clientes descubrir y buscar, objetos y datos, a través de un nombre. Además contiene soporte para *LDAP (Lightweight Directory Access Protocol),* Naming Service, Java RMI Registry y *COS (CORBA Object Services).*

- **Java API for XML Processing *(*JAXP*):*** Se trata del soporte para el administración y tratamiento de ficheros XML.

- **JavaMail:** Es una Interfaz de Programación de Aplicaciones que admite crear aplicaciones Java para mensajería y envío de correo electrónico en forma independiente de la plataforma y del protocolo a utilizar, además facilita el envío y aceptación de correos desde código java.

- **Java Autentication and Authorization Service (JAAS):** Proporciona el servicio para la identificación de usuarios y su autorización para acceder a recursos de la aplicación JEE, además implementa una versión en Java del estándar *Plugable Authentication Module (PAM).*

- **Servicios Web (JAX-WS):** Una API de Java para la creación de servicios web, maneja anotaciones para simplificar el desarrollo y dispersión de los clientes y puntos finales de servicios web.

- **Java Database Connectivity (JDBC):** Se maneja una API estándar para ingresar a los recursos de una base de datos relacional de una forma independiente del proveedor. Se compone de dos partes, la primera parte una interfaz de proveedores para definir drivers específicos y la segunda parte una interfaz para ser utilizada por los componentes JEE.

- J**ava Interface Definition Language (JavaIDL):** Por medio de este servicio se permite a las aplicaciones actuar como clientes de servicios CORBA, solicitando objetos externos y manipulando el protocolo IIOP.

Además, suministra servicios para manejo de protocolos, tales como el HTTP/HTTPS, que es un protocolo estándar utilizado para comunicaciones seguras sobre Secure Socket Layer (SSL), IIOP/RMI, JDBC. En la figura siguiente figura se muestran las APIs de los Contenedores Web, EJB y Aplicación Cliente de JEE. (Eric Armstrong, Jennifer Ball, Stephanie Bodoff, Debbie Carson, Ian Evans, Maydene Fisher, Dale Green, Kim Haase, Eric Jendrock., 2003)

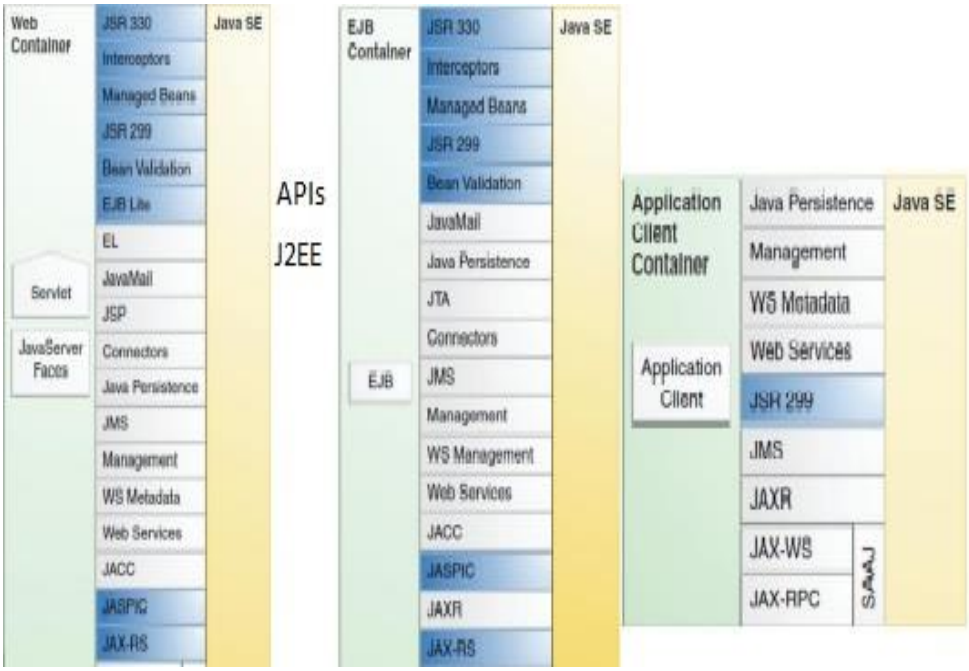


Ilustración 6: APIs de la Plataforma JEE

Fuente: Los Autores

2.2.1.6. Empaquetado de aplicaciones

Una aplicación JEE es distribuida en un fichero que se denomina Archivo Empresarial (EAR), el mismo que es un Archivo Java estándar (JAR), con una extensión ya sea .ear o .war. El uso de archivos EAR, WAR y algunos módulos hacen posible el ensamble o empaquetado de aplicaciones JEE, utilizando los mismos componentes, además que no se necesita de codificación extra, es solo un tema de empaquetado de varios nódulos JEE en un fichero EAR o WAR de JEE.

Las tareas a realizar de un desarrollador de software para entregar un archivo EAR que contiene la aplicación JEE:

1. Se ensambla los archivos JAR y WAR diseñados y desarrollados en fases anteriores en un archivo EAR de la aplicación JEE.

2. Se especifica opcionalmente y de acuerdo a nuestra necesidad el descriptor de despliegue de la aplicación JEE.
3. Se verifica que el contenido del fichero EAR esté debidamente formado y plasme con la especificación JEE.

Un fichero EAR contiene módulos JEE y descriptores de despliegue como mostraremos en la siguiente imagen. Un descriptor de despliegue es un documento XML, (.XML), aquel que describe la configuración de despliegue de una aplicación, un componente o un módulo. Los datos en el descriptor de despliegue pueden ser cambiados sin la necesidad de modificar el código fuente ya que la información en el descriptor de despliegue es declarativa. Al momento de su ejecución el servidor JEE lee el descriptor de despliegue y actúa sobre la aplicación, un componente o un módulo como pertenece. (Óscar Belmonte, Carlos Granell Canut, María del Carmen Erdozain Navarro, 2012)

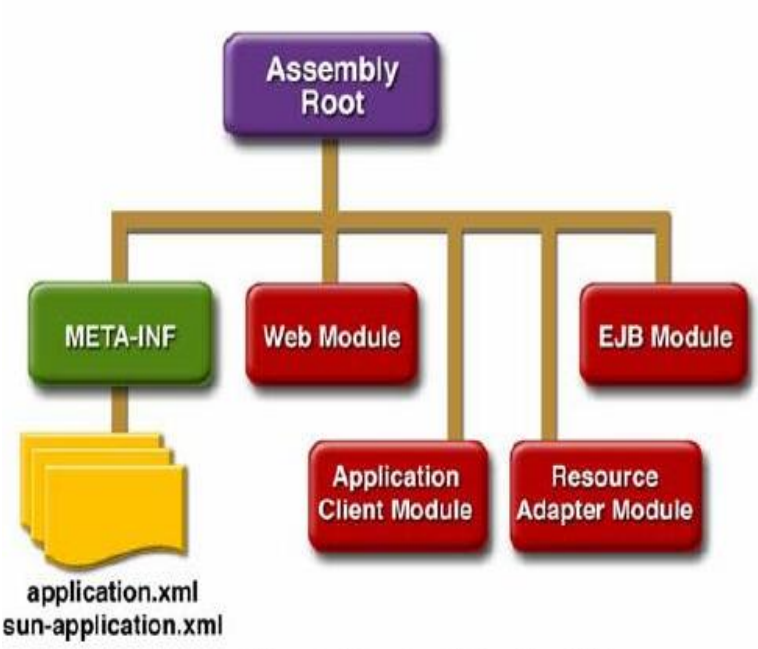


Ilustración 7: Estructura del Archivo EAR

Fuente: Los Autores

Como se dice que hay dos tipos de descriptor de despliegue:

- Un **descriptor de despliegue JEE** se a determinado por una especificación JEE y puede ser manejado para configurar el despliegue de una implementación compatible con JEE.
- Un descriptor de despliegue en tiempo de ejecución, se utiliza para configurar parámetros específicos de una implementación JEE. **(Acosta, 2013)**

2.2.2. Especificaciones JEE

En la siguiente sección se presenta un breve compendio de las especificaciones solicitadas por la plataforma JEE.

2.2.2.1. Tecnología Java Server Faces

JavaServer Faces (JSF) es una tecnología y un marco de trabajo para crear aplicaciones java JEE, basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones JEE. JSF usa JavaServer Pages (JSP) como la tecnología que permite generar las vistas, pero también se puede adaptar a otras tecnologías como XUL (Ferguson, 2007).

2.2.2.2. Tecnología Enterprise JavaBean

La tecnología Enterprise JavaBeans (EJB) permite simplificar el desarrollo de aplicaciones distribuidas, transaccionales, seguras y portátiles establecidas en la tecnología Java. La especificación EJB se presenta con un modelo de programación; es decir, convenciones o protocolos y un conjunto de clases e interfaces que crean la interfaz de programación de aplicaciones (API) EJB.

Un componente JavaBeans TM empresarial (EJB), es el cuerpo del código que contiene campos y métodos para implementar módulos de lógica de negocio. Se consigue pensar en un bean empresarial como un elemento que puede ser manipulado solo o con otros enterprise beans para ejecutar la lógica del negocio en el servidor Java JEE. En la especificación EJB 3.2, los EJBs son POJOs con una variedad de funcionalidades especiales implícitas, que se accionan en tiempo de ejecución cuando son ejecutados en un contenedor EJB.

Uno de los objetivos principales de los EJB's es suministrar a los desarrolladores un modelo que admita abstraerse de los problemas generales de las aplicaciones empresariales (concurrencia, transacciones, persistencia, seguridad) para concentrar en el desarrollo de la lógica del negocio. **(Acosta, 2013)**

A. Tipos de Enterprise Java Beans.

Se citarán los tres tipos de Enterprise JavaBeans (EJB's):

- **EJB's de entidad.**

Se encapsulan los objetos del lado del servidor que almacenan los datos. Los EJB de entidad presentan la característica fundamental de la persistencia:

- **EJB's de sesión.**

  Administran el flujo de comunicación en el servidor, sirven a los clientes como acceso a los servicios proporcionados por los otros componentes que se encuentran en el servidor representando una conversación transitoria con un cliente, cuando el cliente termina de ejecutarse, el bean de sesión y sus datos desaparecen.

- **EJB's dirigidos por mensaje.**

  Comúnmente se utiliza los mensajes JMS (Java Message Service) para combinar las características de un bean de sesión y un escucha (listener) de mensajes permitiendo a un componente de negocio recibir mensajes de forma asincrónica.

B. Ventajas de la Tecnología Enterprise JavaBeans.

- La tecnología EJB funciona sobre Java, por lo que adquiere algunas de sus ventajas como son su portabilidad, orientación a objetos, etc.
- Con la Tecnología EJB se permite ahorrar mucho trabajo en la fase de diseño (solamente hay que seguir el esquema que nos definen los EJBs, haciéndose innecesaria la aplicación de patrones) y en la fase de implementación (heredando de las clases de EJBs, especialmente si utilizamos la persistencia gestionada por contenedor CMP de EJBs).
- Se realiza automáticamente la mayoría de las consultas a la base de datos, simplificando el trabajo de diseño y programación en este ámbito.
- Con la tecnología EJB se da la facilidad de integración en aplicaciones web basadas en servlets o Java Server Pages (JSPs).
- También se da soporte para integración con servicios web, de tal manera que se pueda acceder a las funcionalidades de los EJBs desde ellos.

C. Desventajas de la Tecnología Enterprise JavaBeans.

- La desventaja que tenemos en una aplicación que ha sido realizada en función del esquema de los EJBs, es que el costo para pasar el trabajo a otro sistema que no incluya esta tecnología es muy elevado, lo cual podría ser obligatorio por motivos de mantenimiento (migración del sistema a otra tecnología) o reutilización a nivel de código en el caso de reutilizar una clase que hereda de EJB en un sistema que no los utiliza.
- La tecnología EJBs es uno de los primordiales componentes de JEE y por esta razón también depende potentemente de otras partes de JEE, tales como RMI, JNDI y JDBC. (Ed Roman, Rima Patel Sriganesh, Gerald Brose, 2007)

2.2.2.3. JavaServer Pages (JSP)

Se dice que es una tecnología que apoya a los desarrolladores de software a crear páginas web dinámicas basadas en HTML, XML, entre otros tipos de documentos. JSP es similar a PHP, pero usa el lenguaje de programación Java. Para desplegar y correr JavaServer Pages, se requiere un servidor web compatible con contenedores servlet como Apache Tomcat o Jetty.

El rendimiento de una página JSP es el mismo que tendría el servlet equivalente, ya que el código es compilado como cualquier otra clase Java. A su vez, la máquina virtual compilará dinámicamente a código de máquina las partes de la aplicación que lo requieran. Esto hace que JSP tenga un buen desempeño y sea más eficiente que otras tecnologías web que ejecutan el código de una manera puramente interpretada. (Ed Roman, Rima Patel Sriganesh, Gerald Brose, 2007)

2.2.3. Servidores de Aplicaciones Java EE

Un servidor de aplicaciones se trata de un dispositivo de software que proporciona servicios de aplicación a las computadoras cliente. Un servidor de aplicaciones gestiona las funciones de lógica de negocio y de acceso de datos a la aplicación. Los principales beneficios son la centralización y la disminución de la complejidad en el desarrollo de aplicaciones.

Algunos servidores de JEE son los siguientes:

- **JonAS**. Servidor de aplicaciones de código abierto de ObjectWeb.
- **WebLogic Application Server**. Servidor de aplicaciones desarrollado por BEA Systems posteriormente adquirida por Oracle Corporation
- **Jboss.** Desarrollado inicialmente por JBoss Inc y adquirido posteriormente por RedHat. Existe una versión de código abierto soportada por la comunidad y otra empresarial.
- **Sun Java System Application Server Platform Edition 9.0.** Servidor de aplicaciones basado en GlassFish.
- **Apache Geronimo 2.0.** Servidor de aplicaciones de Apache Software Foundation.
- **GlassFish,** Servidor de aplicaciones de código abierto de Sun. (Eric Armstrong, Jennifer Ball, Stephanie Bodoff, Debbie Carson, Ian Evans, Maydene Fisher, Dale Green, Kim Haase, Eric Jendrock., 2003)

A continuación detallaremos los más importantes.

2.2.3.1. Servidor de Aplicaciones JBoss

Servidor de aplicaciones Java extensible, reflexivo, y dinámico que se permite reconfigurar. Contiene un conjunto de componentes que implementan la especificación JEE, pero su alcance va mucho más allá de JEE. JBoss es un middleware (se puede intercambiar información entre componentes) de composición abierta, en el sentido de que los usuarios puedan ampliar los servicios de middleware de forma dinámica mediante el despliegue de nuevos componentes en un servidor que ejecuta. El modelo de componentes de middleware, es basada en el modelo Java Management extensions (JMX), y su arquitectura de meta-nivel para los Enterprise JavaBeans generalizadas. El primero requiere una nueva forma de carga de clases, que implementa JBoss. Este último contiene un potente y flexible modelo de invocación de método remoto, fundamentado en proxis dinámicos, y se basa en el uso sistemático de interceptores como artefactos programación orientada a aspectos. El servidor de aplicaciones JBoss es multiplataforma, implementa todo el paquete de servicios JEE (Matena & Hapner, 2008).

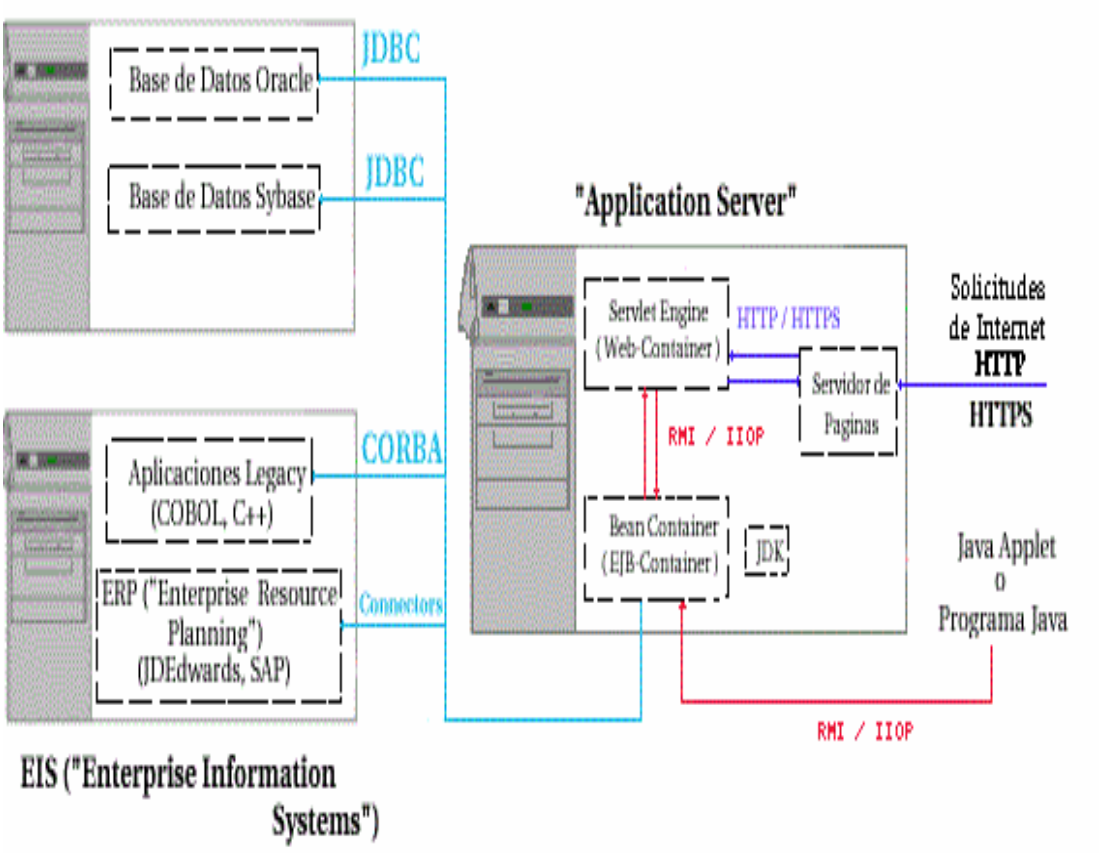


Ilustración 8: Esquema de JBOSS

Fuente: Estudio del servidor de aplicaciones Glassfish y de las aplicaciones JEE

Servicios de Jboss

- **EJB 3.0.** Implementa la especificación inicial de Enterprise JavaBeans 3.0.
- **JBoss AOP.** Está orientado a trabajar con Programación Orientada a Aspectos. Esto permitirá añadir fácilmente servicios empresariales (transacciones, seguridad, persistencia) a clases Java simples.

2.2.3.2. Apache Geronimo

Es un servidor de aplicaciones de código abierto desarrollado por la Apache Software Fundation y distribuido bajo la licencia Apache. Geronimo es actualmente compatible con la especificación Java Enterprise Edition (Java EE) 5.0. IBM ha proporcionado un apoyo considerable al proyecto a través de la comercialización, las contribuciones de código, y la financiación de varios proyectos. El núcleo de Geronimo es Java EE agnostic. Su único objetivo es la gestión de bloques de construcción de Geronimo. Se caracteriza por un diseño arquitectónico que se basa en el concepto de Inversión of Control (COI), lo que significa que el núcleo no tiene dependencia directa de cualquiera de sus componentes. El núcleo es un framework para los servicios que controla el ciclo de vida de servicio y el registro y está basado en Java EE. (Ed Roman, Rima Patel Sriganesh, Gerald Brose, 2007)

Servicios de Apache Geronimo

- **Apache Tomcat.** Servidor de HTTP y contenedor de Servlet que soporta Java Servlet 2.5 y JavaServer Pages (JSP) 2.1.
- **Jetty.** Tiene las mismas funcionalidades que Tomcat, es una alternativa.
- **Apache ActiveMQ.** Servicio de mensajería open source basado en Java Message Service (JMS) 1.1.
- **Apache OpenEJB.** Open source Enterprise JavaBeans (EJB) Container System y EJB Server que soporta Enterprise JavaBeans 3.0, incluye Container Managed Persistence 2 (CMP2) y EJB Query Language (EJBQL).

2.2.3.3. Oracle WebLogic

Es un servidor de aplicaciones JEE y también un servidor web HTTP desarrollado por BEA Systems posteriormente adquirida por Oracle Corporation. Puede ejecutarse en distintos sistemas operativos. La última versión de WebLogic forma parte de Oracle Fusion Middleware, que consiste en un conjunto de software de Oracle.

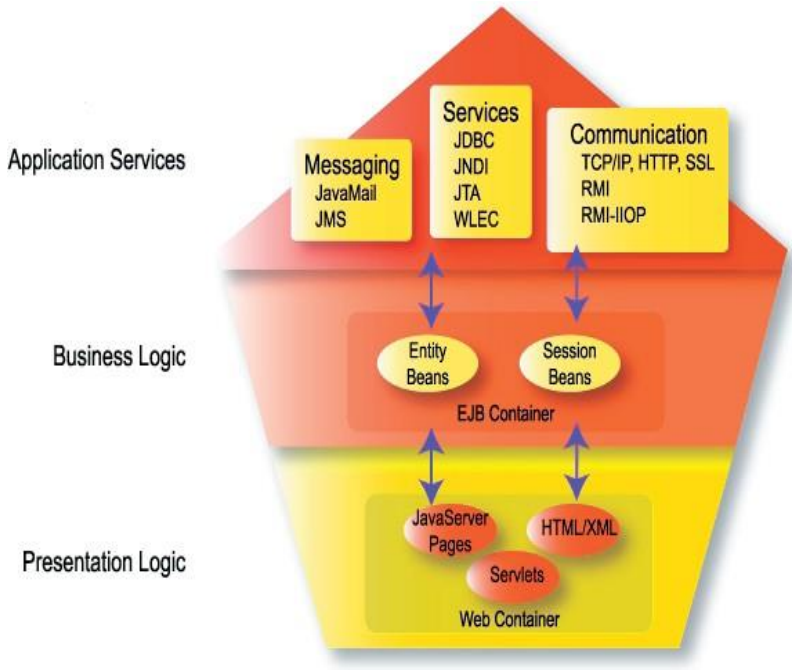


Ilustración 9: Tecnologías de WebLogic

Fuente: Estudio del servidor de aplicaciones Glassfish y de las aplicaciones JEE

2.2.3.3. Glassfish

El término Glassfish, traducido al español sería algo parecido como "Pez de Cristal. 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.

Servidor de aplicaciones desarrollado por Sun Microsystems 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.

2.3. Introducción a los EJBs

EJB es un estándar para la construcción de componentes de servidor en Java. Define un acuerdo (contrato) entre los componentes y los servidores de aplicaciones que permite a cualquier componente para ejecutarse en cualquier servidor de aplicaciones. Los componentes EJB (llamados beans de empresa) son desplegables, y pueden ser importados y se cargan en un servidor de aplicaciones, que alberga los componentes.

Las tres primeras proposiciones de EJB son los siguientes:

- Es acordada por la industria.
- La portabilidad es más fácil.
- Desarrollo rápido de aplicaciones.

(Linda DeMichiel - Sun Microsystems, Michael Keith - Oracle Corporation, 2006)

2.3.1. Beneficios de los Enterprise JavaBean

En las aplicaciones basadas en EJBs se permite concentrar más en la lógica de negocio, sin preocuparse de transacciones y connection pooling (es un modelo utilizado por las aplicaciones de software para conectarse a bases de datos utilizando un conjunto de pre - creado de objetos de conexión reutilizables) que son provistos por el contenedor.

Se dice que los EJBs son componentes lo que cuenta a favor de la reutilización de código de manera importante. Además se lleva en cuenta que existe una clara separación entre desarrollo, explotación y administración de una aplicación EJB. Otro beneficio es que el contenedor de EJBs gestiona transacciones, detalles de manejo de estado, multi-threading, connection pooling, seguridad y otros detalles de bajo nivel que el desarrollador no necesita conocer. (Ed Roman, Rima Patel Sriganesh, Gerald Brose, 2007).

2.3.2. Roles de EJB

Enterprise JavaBeans define siete roles distintos en el desarrollo de la aplicaciones y en el ciclo de vida de la implementación. Cada rol EJB puede ser realizada por un partido diferente. La arquitectura EJB específica los contratos que aseguren que el producto de cada papel EJB es compatible con el producto de los otros roles EJB. El EJB se centra en aquellos contratos que se requieren para apoyar el desarrollo y despliegue de beans de empresa ISV-escrita.

En algunos casos, una sola de las partes puede realizar varias funciones EJB. Por ejemplo, el proveedor de contenedores y el proveedor del servidor EJB pueden ser el mismo proveedor. O un solo programador puede realizar las dos funciones EJB de la empresa proveedora Bean y el ensamblador de la aplicación. Las siguientes secciones definen los siete roles de EJB.

2.3.3. Arquitectura de EJB

EJB se basan en el modelo conceptual de Java Remote Method Invocation (RMI). Por ejemplo, el acceso a objetos a distancia y el paso de parámetros para los EJB sigue la especificación de RMI. La especificación EJB no prescribe que el mecanismo de transporte tiene que ser pura RMI. El servidor Oracle8i EJB utiliza RMI sobre IIOP por su protocolo de transporte, una práctica que se está convirtiendo en común entre los fabricantes de servidores. **(Linda DeMichiel - Sun Microsystems, Michael Keith - Oracle Corporation, 2006)**

Con respecto a la arquitectura EJB, si realizamos un completo análisis de ella se pueden encontrar los siguientes componentes que la estructuran:

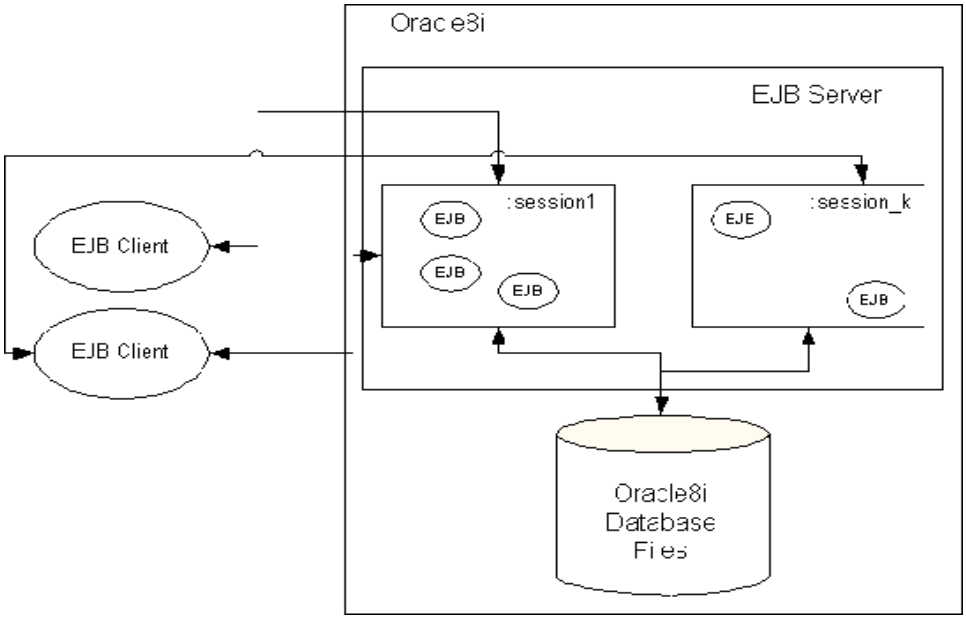


Ilustración 10: Arquitectura básica de EJB.

Fuente: https://docs.oracle.com/

2.3.5. Tipos de EJBs

EJB 1.0 y 1.1 define dos tipos diferentes de beans: beans de sesión y beans de entidad.

a) Beans de Sesión o Session Beans

Se dice que un bean de sesión representa el trabajo que se realiza para el código de cliente que está llamando. Los beans de sesión son objetos de procesos de negocio. Que implementan la

lógica de negocio, reglas de negocio y flujo de trabajo. Por ejemplo, un bean de sesión podría realizar citando precio, entrada de pedidos, la compresión de vídeo, las transacciones bancarias, operaciones de bolsa, las operaciones de base de datos, cálculos complejos, y más. Son componentes reutilizables que contienen la lógica de los procesos de negocio. (Ed Roman, Rima Patel Sriganesh, Gerald Brose, 2007)

- **Beans de Sesión con Estado *(Stateful Session Beans)***

Como hemos dicho, los beans de sesión representan los procesos de negocio. Algunos procesos de negocio se pueden realizar en una única solicitud de método, tales como el cálculo del precio de los bienes o verificar una cuenta de tarjeta de crédito. Otros procesos de negocio están más atraídos hacia fuera y pueden durar entre varias peticiones y transacciones de método. Un bean de sesión con estado es un bean que está diseñado para dar servicio a los procesos de negocio que abarcan múltiples solicitudes o transacciones de método. Para lograr esto, los beans de sesión con estado retienen estado en nombre de un cliente individual. Si se cambia el estado de una sesión de bean con estado durante una llamada a un método, ese mismo estado estará disponible para que el mismo cliente en la siguiente invocación. (Ed Roman, Rima Patel Sriganesh, Gerald Brose, 2007)

- **Beans de Sesión Sin Estado (*Stateless Session Beans*)**

Algunos procesos de negocio, naturalmente, se prestan a un único paradigma de petición. Un proceso de solicitud de negocio único es uno que no requiere de estado que se mantiene entre las distintas invocaciones de métodos. Los beans de sesión sin estado son componentes que pueden acomodar estos tipos de procesos de solicitud de negocios individuales. Son proveedores de métodos anónimos porque no son conscientes de cualquier historial del cliente. Una vez que el bean se completa esta tarea, está disponible para dar servicio a un cliente diferente y no conserva ningún conocimiento previo del cliente original.

b) Beans de Entidad o Entity Beans

Los beans de entidad no contienen lógica que los procesos de negocio modelo de datos. Los beans de sesión manejan los procesos de negocio. Los beans de sesión pueden utilizar beans de entidad para representar los datos que utilizan, similar a cómo un cajero de banco utiliza una cuenta bancaria.

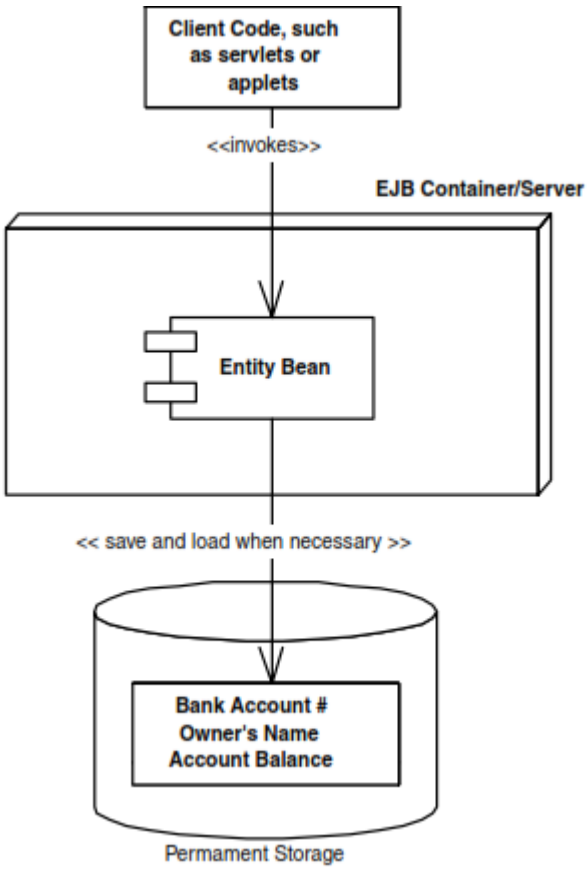


Ilustración 11: Los beans de entidad son una vista en un almacén de datos subyacente

Fuente: Mastering Enterprise JavaBeans - Third Edition.

a) Beans de Entidad de Persistencia Gestionada por Bean (*bean-Managed Persistent Entity Beans*)

Como hemos visto, los beans de entidad son componentes persistentes debido a que su estado se guarda en un almacenamiento secundario, como una base de datos relacional. Por ejemplo, mediante el uso de la tecnología de mapeo objeto-relacional, se puede tomar un objeto en memoria y mapa de ese objeto en una serie persistente de registros de bases de datos relacionales. A continuación, puede recuperar esos registros en un momento posterior para reconstruir el objeto en memoria y utilizarlo de nuevo. Otro esquema es utilizar una base de datos de objetos como su almacenamiento persistente, que almacena objetos reales en lugar de los registros relacionales. (Sánchez, 2004)

b) Beans de Entidad de Persistencia Gestionados por Contenedor (*Container-Managed Persistent Entity Beans*).

La buena noticia es que EJB permite a los desarrolladores de beans de entidad que se preocupe por la codificación de la lógica de persistencia. Un servicio de EJB 1.0 es que pueden

proporcionar contenedores, y EJB 1.1 deben proporcionar recipientes, es la persistencia automática para sus beans de entidad.(Linda DeMichiel - Sun Microsystems, Michael Keith - Oracle Corporation, 2006)

2.4. Persistencia

Una de las características claves del modelo EJB es la persistencia, que puede implantarse a nivel de bean de entidad o a nivel de contenedor. Los beans de entidad son objetos persistentes que representan los datos en un almacenamiento subyacente.

En el primer caso, el desarrollador sobrecarga los métodos de carga y almacenamiento, para guardar la información residente en los atributos del bean de entidad en una fuente de datos por ejemplo, JDBC. Este mecanismo es el más flexible (pues el desarrollador define cómo se almacenan los datos) pero liga a nivel de código la bean con el almacén de datos. Algunos proveedores están empezando a proporcionar herramientas de traducción objeto-relacional. Otra alternativa son las vistas de base de datos, sitien muchas bases de datos no permiten la actualización de múltiples tablas desde una vista. Si la persistencia la proporciona el contenedor EJB, en general el proveedor incorpora una herramienta de traducción objeto-relacional para enlazar atributos del bean con campos del almacén de datos. Los contenedores EJB pueden proporcionar la persistencia transparente de los beans de entidad de persistencia gestionados por contenedor. (Ed Roman, Rima Patel Sriganesh, Gerald Brose, 2007)

2.5. Seguridad

Aquí se presenta y explica la seguridad EJB en detalle. Vamos a empezar con una observación fundamental: en la construcción de sistemas basados en middleware empresarial, normalmente se desea integrar importantes recursos de la empresa. Debido importante también significa crítico, la seguridad puede ser uno de los aspectos más importantes de su arquitectura de aplicaciones EJB.

2.5.1. Seguridad de Aplicaciones Web

En aplicaciones JEE con una capa Web, el primer punto de interacción con el cliente es el contenedor web y los archivos JSP o servlets que alberga, como se muestra la figura 6 Los clientes envían peticiones de protocolo de transferencia de hipertexto (HTTP) y los servlets o archivos JSP llamarían casos de frijol en los contenedores EJB utilizando RMI-IIOP o SOAP.

La seguridad de las aplicaciones Web no está cubierto por las especificaciones EJB sino más bien por el servlet especificación de Java y la versión JEE especificación de la plataforma 1.4. Los conceptos de seguridad generales utilizados por JEE para ambos servlets y EJB son muy similares, pero si usted está construyendo aplicaciones web complejas, se recomienda que consulte la especificación de Java Servlet que está disponible en http://java.sun.com/products/ servlet /. (Gelernter, 2011)

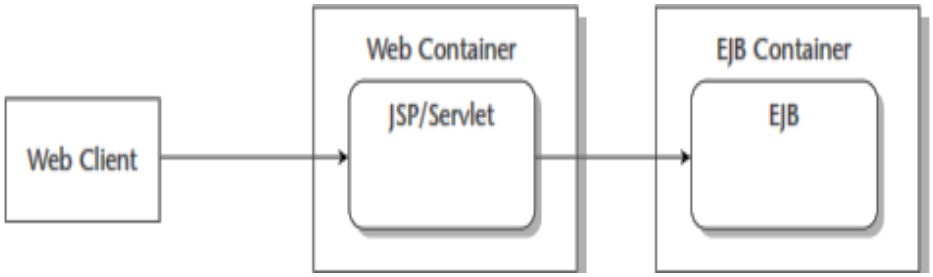


Ilustración 12: Aplicaciones Empresariales

Fuente: Mastering Enterprise JavaBeans - Third Edition

2.5.2. Seguridad EJB

Hay dos medidas de seguridad que deben pasar los clientes cuando se agrega seguridad a un sistema de EJB: autenticación y autorización. Autenticación debe ser realizado antes de que se llama a cualquier método EJB. Autorización, por el contrario, se produce al comienzo de cada llamada de método EJB. **(Ed Roman, Rima Patel Sriganesh, Gerald Brose, 2007)**

2.5.3. Autenticación en EJB

En las versiones anteriores de EJB (1.0 y 1.1), no había manera portátil para lograr la autenticación. La forma específica su código de cliente se asoció con una identidad de seguridad se deja a la discreción de su solicitud y su contenedor EJB. Esto significaba cada contenedor EJB podría manejar la autenticación diferente. La buena noticia es que desde EJB 2.0, la autenticación es ahora portátil y robusto. Puede llamar a la lógica de autenticación a través del Servicio de autenticación y autorización Java (JAAS), una API JEE separada. (Óscar Belmonte, Carlos Granell Canut, María del Carmen Erdozain Navarro, 2012)

2.5.4. Autorización de EJB

Después de que el cliente se ha autenticado, debe pasar una prueba de autorización para llamar a métodos en sus beans. El contenedor EJB hace cumplir la autorización mediante la definición de políticas de seguridad para sus beans.

2.6. Rendimiento

Las solicitudes deben ser escalables y mantener altos niveles de disponibilidad, también se espera para satisfacer las expectativas de los usuarios, problemas de rendimiento deben ser parte del proceso de diseño, la optimización del rendimiento se puede aplicar a la propia aplicación y el servidor de aplicaciones.

Al definir los requisitos de rendimiento básicamente significa que señalan sus necesidades de rendimiento de diversos puntos de vista que determinan la experiencia de usuario bajo cargas variables, el porcentaje de los recursos del sistema utilizado, la asignación de los recursos del sistema para lograr el rendimiento deseado, y así sucesivamente. Muchas veces vemos estos requisitos definidos después de que el sistema ha sido desarrollado y está a punto de desplegarse con mayor frecuencia, en la noche de las pruebas de carga. (Ed Roman, Rima Patel Sriganesh, Gerald Brose, 2007)

CAPÍTULO III

3. ANÁLISIS DE LA TECNOLOGIA EJB PARA EL DESARROLLO DE APLICACIONES EMPRESARIALES.

3.1. Introducción

En este capítulo se realiza un análisis de la tecnología Enterprise JavaBeans (EJB) frente a la Programación Tradicional, para determinar la capacidad que ofrece en cuanto a rendimiento se refiere. Se utiliza estadística descriptiva para demostrar y presentar los resultados del análisis de los datos obtenidos en la medición correspondiente. Al finalizar el análisis y evaluación de resultados, se realiza la demostración de la hipótesis previamente planteada y se selecciona la tecnología de desarrollo que brinda mejor rendimiento, para el desarrollo del Sistema de Gestión de Laboratorios UNACH.

3.2. Diseño de Investigación.

El presente trabajo se consideró una investigación no experimental transversal descriptiva porque permitió analizar la incidencia y los valores que se manifestó en cada uno de los indicadores en forma cuantitativa y cualitativa.

Se realizó la comparación de las tecnologías de programación EJB y programación tradicional, se establecen parámetros los cuales contienen indicadores y que sirvieron para la realización del presente análisis, los mismos que se han seleccionado por los autores de la tesis, en base a la información obtenida de estudios realizados en Tesis, Libros y Aplicaciones que buscan mejorar a nivel de rendimiento.

3.2.1. Definición de los Parámetros de Comparación

Peticiones del usuario.- Mediante este parámetro se pretende mostrar la eficacia, con la que la tecnología EJB y la programación tradicional de servicios web, administra procesos de rendimientos tales como: número de peticiones soportadas y tiempo de ejecución de solicitud

Carga de usuarios.- Por medio de este parámetro se pretende mostrar la eficacia en respuesta de la aplicación web, gestionando procesos de rendimiento en respuesta tales como: número de usuarios concurrentes y número de usuarios en espera.

Uso de Hardware.- Con este parámetro se pretende determinar los niveles de memoria RAM y procesador necesarios para la satisfactoria ejecución de la aplicación.

En la tabla número dos se puede observar en resumen los parámetros de comparación junto a sus respectivos indicadores y definiciones de estos, que se empleó en el presente análisis.

Parámetro	Indicador	Definición Indicador
Peticiones del usuario	Número de peticiones soportadas.	El número de peticiones HTTP que llegan al servidor Web con éxito.
	Tiempo de ejecución de solicitud.	Tiempo que transcurre desde que se envía la solicitud hasta recibir una respuesta de la tecnología de servicios web.
Carga de usuarios	Número de usuarios concurrentes	Hace referencia a la cantidad promedio de usuarios conectados simultáneamente haciendo uso del Sistema.
	Número de usuarios en espera.	Número de usuarios esperando recursos (Memoria RAM y procesador) para ser atendidos por el sistema Web.
Uso de Hardware	Memoria RAM	Cantidad de espacio en Memoria RAM necesaria para acumular un promedio de peticiones concurrentes.
	Uso de Procesador.	Porcentaje de uso del procesador por el framework de servicios web.

Tabla 1: Definición de los parámetros de comparación

Fuente: Los Autores

3.3. Tipo de Investigación.

Los tipos de investigación utilizados son investigación descriptiva y aplicada por que se basa en estudios previos y procesos aplicados en otros entornos similares al de este trabajo.
El objetivo es ver cuál de las dos tecnologías de desarrollo proporciona un mejor nivel de rendimiento a nivel de aplicaciones empresariales.

3.4. Técnicas de Investigación

En la vigente investigación se utilizó la técnica de la entrevista mediante una conversación profesional, con la que además de adquirirse información acerca de los laboratorios, tuvo importancia para iniciar con la automatización. La observación es un elemento fundamental de todo proceso investigativo, ya que permitió tomar información y se registrarla para su posterior análisis.

3.5. Metodología de investigación.

Las dos tecnologías de desarrollo web fueron evaluadas de forma comparativa mediante datos estadísticos calculados con los resultados de las pruebas realizadas a los prototipos en cada parámetro definido anteriormente. Para comparar el nivel de rendimiento de cada tecnología de desarrollo web se creará tablas de calificación y ponderación según existe una calificación y ponderación según el estudio de tesis "Análisis comparativo de metro y axis2 para el desarrollo de aplicaciones que consuman servicios web Wcf", para el rendimiento según los valores de las pruebas, con lo cual se pretende exponer la comparación entre las dos tecnologías. La tabla número 3 Niveles de Cumplimiento se utilizó para la elaboración de conclusiones de las pruebas.

La evaluación de las dos tecnologías se da mediante los parámetros descritos con anterioridad, con lo cual se obtendrán resultados cuantitativos y cualitativos que permitirán realizar una selección sustentada de una de las dos tecnologías de desarrollo web analizados. En la Tabla número 3 se ilustra los niveles de cumplimiento que fueron utilizados en el desarrollo de la presente investigación.

VALORACION	CALIFICACION	FORMA GRÁFICA	VALOR	DESCRIPCIÓN
Excelente	>75% y <=100%	★★★★	4	Todas las expectativas cumplidas
Buena	>50% y <=75%	★★★☆	3	Mayoría de las expectativas cumplidas
Regular	>25% y <=50%	★★☆☆	2	Pocas de las expectativas cumplidas
Malo	>=0% y <=25%	★☆☆☆	1	Ninguna expectativa cumplida

Tabla 2: Niveles de Cumplimiento

Fuente: Los Autores

La calificación de cada parámetro se realiza mediante la suma de los puntajes obtenidos en el análisis, haciendo uso de las siguientes formulas:

$C_{ejb}=\sum_{i=0}^{n} V_i$ (1)

$C_{pt}=\sum_{i=0}^{n} V_i$ (2)

$C_{max}=\sum_{i=0}^{n} VM_i$ (3)

De donde:

n= Número de indicadores del parámetro
$\boldsymbol{V_i}$= Valor de calificación de cada indicador
$\boldsymbol{VM_i}$= Valor máximo de calificación de cada indicador
$\boldsymbol{C_{ejb}}$= Calificación para la tecnología de desarrollo web EJB en el parámetro.
$\boldsymbol{C_{pt}}$= Calificación para la tecnología de programación tradicional en el parámetro.
$\boldsymbol{C_{max}}$=Calificación máxima sobre el que se mide el parámetro.

Las formulas 4 y 5 sirven para calcular los porcentajes de cada uno de los parámetros de comparación en cada tecnología de desarrollo web.

$\boldsymbol{P_{ejb}}=\left(\frac{C_{ejb}}{C_{max}}\right)*100\%$ (4)

$\boldsymbol{P_{pt}}=\left(\frac{C_{pt}}{C_{max}}\right)*100\%$ (5)

De donde:

$\boldsymbol{P_{ejb}}$: Porcentaje de la tecnología de desarrollo web EJB.
$\boldsymbol{P_{pt}}$=Porcentaje de la tecnología de desarrollo web tradicional.

3.6. Instrumentos de Medición

Los instrumentos de medición permitió medir los indicadores de los parámetros de comparación planteados anteriormente, sometiendo a prueba a los prototipos de cada tecnología de desarrollo. Estos instrumentos se han seleccionado en base a los parámetros de comparación, análisis de los autores de la presente investigación. A continuación enunciaremos la selección de herramientas y el por qué elegimos una de ellas.

3.6.1. Herramientas para pruebas de Rendimiento

Las herramientas, en este caso son útiles, por ejemplo, en las pruebas que necesitan tener concurrencia de usuarios al servidor a probar. Algunas de ellas son:

JMeter: Es utilizada para realizar pruebas de rendimiento, de stress, de carga y de volumen, sobre recursos estáticos o dinámicos (archivos, Servlets, scripts Perl, Objetos Java, BB.DD., Servidores de FTP, etc.). Puede ser utilizado para simular una sobrecarga en un servidor, una red o un objeto, para poner a prueba su resistencia o para analizar el rendimiento global para diferentes tipos de carga. Puede usarse para hacer un análisis gráfico de rendimiento o para probar su servidor /script / comportamiento del objeto con sobrecargas concurrentes.

OpenSTA: Es un conjunto de herramientas que tiene la capacidad de realizar secuencias de comandos HTTP y HTTPS para pruebas de sobrecarga, para medir el rendimiento de aplicaciones en plataformas Win32. Permite captarlas peticiones del usuario generadas en un navegador Web, luego guardarlas, y poder editar para su posterior uso.
WEbLoad: Permite realizar pruebas de rendimiento, a través de un entorno gráfico en el cual se pueden desarrollar, grabar y editar script de pruebas.

3.6.2. Comparativa de las herramientas de medición

En la tabla número 4 se realiza un análisis comparativo de las Herramientas de Medición, para efectuar las pruebas de los prototipos de la Tecnología EJB y la Programación Tradicional. Las herramientas que se analiza en la siguiente tabla son: OpenSTA, Apache JMeter y WebLoad.

Criterio de Evaluación	**Descripción**	**OpenSTA**	**Apache JMeter**	**WebLoad**
Protocolos	Los protocolos que pueden ser capturados, manipulados y simulados por la aplicación	HTTP 1.0/ 1.1/ HTTPS(SSL), SOAP/XML	HTTP,FTP,SOAP/XML, RPC,JDBC	HTTP/S, WAP, AJAX,ActiveX, Java, web services,.
Playback functions	Ejecución de los scripts y facilidades de debug los scripts	Vista extendida del log donde se ven los colores de los parámetros y	Herramienta GUI. ENTONCES HAY MUCHOS Listeners, los cuales son usados para	Vista extendida del log. El cual muestra los pedidos y los datos de respuesta

		los mensajes del servidor	capturar la grabación y replicar los mensajes	
Parametrización	Cambio dinámico de valores para variables pasadas desde el cliente al servidor durante el POST para asegurar una simulación más real del comportamiento	Gran cantidad de facilidades para data entry, incluyendo interfaces Wizard para generar automáticamente datos de prueba.	La Parametrización se puede hacer por interfaz en el control 'Users Parameters'.	Gran cantidad de facilidades para data entry, incluyendo interfaces Wizard para generar automáticamente datos de prueba
Simulación de la velocidad de conexión de los usuarios	Habilidad de emular las diferentes velocidades de la red que pueden ser utilizadas por los usuarios	No lo permite	No lo permite. Pero puede programarse en Java	No lo permite en la versión Open Source.
Reportes y análisis	Facilidades para examinar e investigar los resultados de las pruebas incluyendo contadores y recursos monitoreados.	Gráficos simples y suficientes como para analizar los resultados de carga y uso de recursos. Los datos pueden ser exportados a Excel.	Puede crear gráficos pero no reportes.	WebLOAD console muestra reportes online de las sesiones que están corriendo. El usuario puede crear sus propias vistas de las estadísticas que estos reportes muestran, a elegir entre gráficos o textuales.
Extensibilidad	La habilidad de incrementar la funcionalidad de la herramienta.	Pueden escribirse módulos en SCL. Son Open Source.	Funciones Beanshell/JAVA pueden ser definidas y usadas	Permite agregar objetos Java ActiveX o COM en los test Scripts. El Framework de

			como plug-in.	WebLoad es flexible.

Tabla 3: Características de las herramientas de medición

Fuente: José Pablo Sarco, Testing de Performance

De acuerdo al análisis comparativo JMeter destaca por sus fortalezas a nivel de versatilidad, estabilidad y por ser de uso gratuito.

Java Visual VM.- Es una herramienta incluida en el JDK y permite visualizar el comportamiento de las aplicaciones en relación al consumo y uso de la memoria RAM y procesador respectivamente. Con esta se pretende medir los recursos de hardware necesarios para cada prototipo en compatibilidad a la Herramienta JMeter.

3.6.3. Herramienta SIAE

Para la tabulación de información y la comprobación de la hipótesis, se utilizó la herramienta Sistema Inteligente de Análisis Estadístico (SIAE), permitiendo visualizar los resultados estadísticos para su análisis e interpretación correspondiente utilizando estadística descriptiva, con el cual se analiza, describe y representa los resultados recolectados en el proceso de medición de los parámetros para la búsqueda posterior de las conclusiones.

3.7. Escenario de Pruebas

La Universidad Nacional de Chimborazo en cada una de sus unidades académicas dispone de laboratorios, salas multimedia, salas de internet, talleres; para el desarrollo de actividades de apoyo académico, las unidades académicas conforme a los perfiles de las carreras proveen de las infraestructuras, recursos e insumos para su respectiva investigación, los procesos de gestión y control de servicios y usos de los laboratorios se los realiza en forma manual, no existe un estándar institucional, razón por la cual cada funcionario organiza la información conforme a sus necesidades y basadas en la experiencia en su puesto de trabajo. En la cual se aplicó un análisis de los EJB aplicado al desarrollo de un Sistema de Gestión Y Control De Laboratorios en la Universidad Nacional de Chimborazo, con el cual se optimizará la gestión de los Laboratorios.

Se formalizó un único escenario de prueba que contó con equipo de hardware y software, y servirá de ambiente para ejecutar los prototipos de cada tecnología de desarrollo web, a fin de obtener resultados que permitan en lo posterior evaluar a los mismos.

3.7.1. Equipo utilizado

El escenario de prueba está conformada por un servidor virtual, en donde se han ejecutado todas las pruebas. La máquina es una computadora portátil que tiene las especificaciones de hardware detalladas en la Tabla 5.

Característica	**Maquina Servidor**
Procesador	Intel(R) Core(TM) i5-3210M CPU 2.50GHz
Memoria RAM	4,00GB
Disco Duro	500GB

Tabla 4: Características del Servidor

Fuente: Los autores

3.7.2. Software utilizado

En las especificaciones y componentes de software que han sido parte del escenario y se ha utilizado para la realización de las pruebas se detallan en la Tabla siguiente.

Característica	**Descripción**
Sistema Operativo	Centos 6.5
IDE	Netbeans 8.0.2
Servidor Web	GlassFish Server Open Source Edition 4.0
JDK	jdk1.8.0_91
Pruebas de Carga	JMeter 3.0
Administración de recursos	Java Visual VM 1.3.8

Tabla 5: Especificaciones de software utilizado

Fuente: Los autores

3.8. Construcción de los prototipos

El propósito de los prototipos es proveer una fuente de información oportuna para con esta realizar una posterior comparación y verificación de las tecnologías con las que se han desarrollado los mismos.

Para la demostración de la hipótesis de este estudio se han realizado dos pequeñas aplicaciones web que tienen la funcionalidad de mostrar el módulo de gestión de categorías el primero utilizando la programación con la tecnología Enterprise JavaBeans (EJB), y la segunda utilizando la programación tradicional, quienes estarán con las siguientes funcionalidades.

- **Inserción, Eliminación, Selección y Edición.**

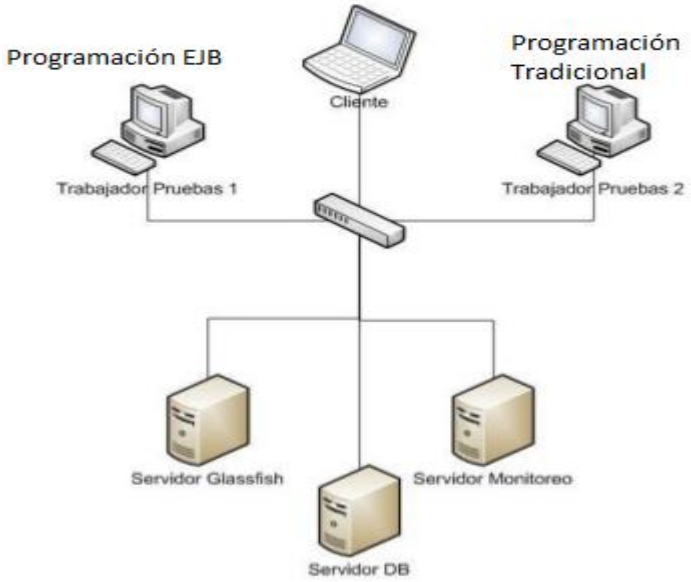


Ilustración 13: Escenario de prueba

Fuente: Los Autores

3.8.1. Prototipo Enterprise Java Beans (EJB)

La arquitectura habitual de cuatro capas consta de una capa de presentación, que muestra la interfaz gráfica de usuario, la capa de negocio contiene el flujo y la lógica de proceso en la cual se realiza el análisis, la capa de persistencia es la abstracción de la base de datos en objetos y la capa de base de datos suele ser un sistema gestor de base de datos relacional que se conecta por medio de un pool de conexiones. Los EJB se encuentran en la capa de lógica de negocio y la capa de persistencia para los beans de entidad.

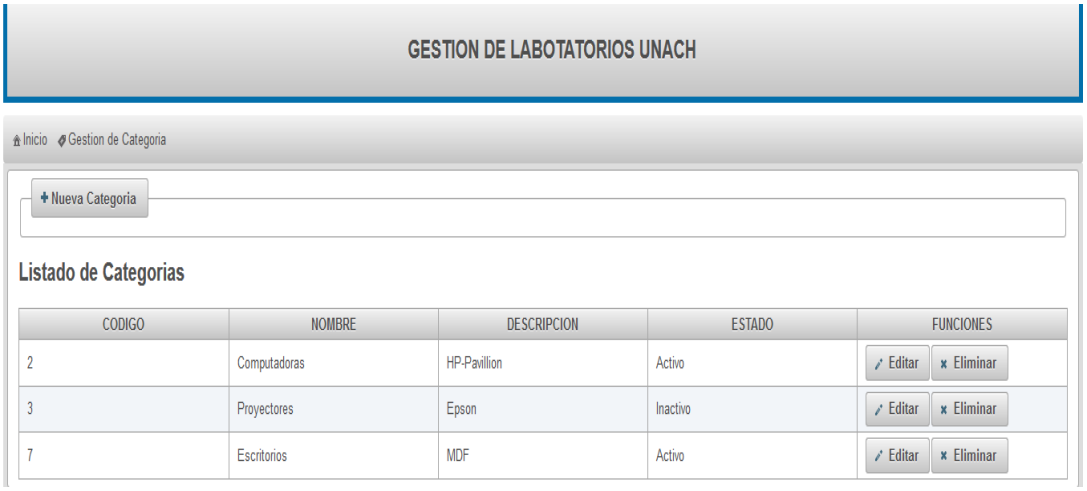


Ilustración 14: Prototipo Tecnología de programación EJB

Fuente: Los autores

3.8.2. Prototipo Programación Tradicional

La programación estructurada define componentes para la representación de información por medio de estructuras simples y en muchos casos objetos, por otro lado para la interacción del usuario la lógica de negocio se centra en la capa de modelo; por lo tanto las transacciones se lo realizan por medio de acceso a datos.

Se aplicó para la capa de presentación Java Server Faces JSF. El prototipo fue desplegado en el servidor web GlassFish Server 4.1 Open Source Edition, por medio de acceso a datos se consiguió la interoperabilidad del prototipo Gestión de Categorías.

Ilustración 15: Prototipo Tecnología de programación tradicional

Fuente: Los autores

Una vez definidas el hardware, software y prototipos se deja preparado el escenario para realizar las pruebas que proveerán los datos necesarios para la comparación de la programación EJB y la tradicional.

En la Ilustración 17, se muestra el modelo relacional del prototipo a comparar, realizado en el SGBD Postgres, con el modelo relacional. La investigación tendrá un alcance con la Gestión de Categorías a nivel de equipos.

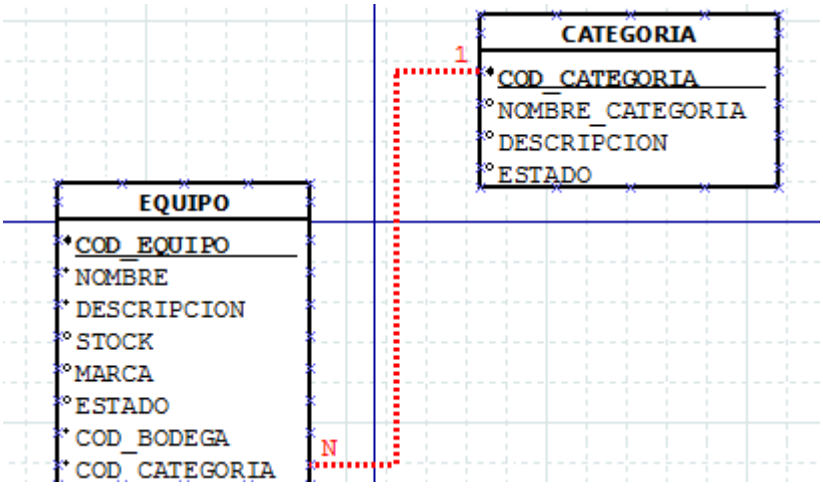


Ilustración 16: Diagrama relacional de la base de datos del prototipo

Fuente: Los Autores

3.9. Desarrollo de las pruebas con los parámetros de comparación

Se inició la realización de las pruebas en cada uno de los parámetros e indicadores de comparación establecidos para las tecnologías de servicios web, en las que se detallan el proceso de obtención de resultados. Los mismos que fueron evaluados, calificados, graficados y se realizará su respectiva interpretación.

Para los parámetros peticiones del usuario, carga de usuarios y uso de hardware, se empleó dos indicadores adicionales tales como la frecuencia y el número de usuarios concurrentes por prueba.

La frecuencia es el número de ejecuciones realizadas sobre las pruebas, que para el presente trabajo de investigación posee un valor de 10, en base a los 10 semestres que dispone cada carrera de la Facultad de Ingeniería.

El número de usuarios concurrentes, hace referencia a la cantidad total de alumnos que se encuentran realizando sus estudios en la Facultad de Ingeniería de la Universidad Nacional de Chimborazo.

El total es de: 2173 alumnos.

3.9.1. Peticiones del usuario

Por medio de este parámetro se pretende mostrar la eficacia de la tecnología de desarrollo web administra procesos de rendimientos tales como: número de peticiones soportadas y tiempo de ejecución de solicitud.

Indicador	Descripción
I1. Número de peticiones soportadas.	Número de peticiones soportadas por la Tecnología de servicios Web.
I2. Tiempo de ejecución de solicitud.	Tiempo que transcurre desde que se envía la petición hasta recibir una respuesta por parte de la tecnología de servicios web.

Tabla 6: Peticiones del Usuario

Fuente: Los autores

A partir de las aplicaciones prototipo descritas anteriormente se ejecutaron un grupo de pruebas de carga de manera automatizada con la herramienta JMeter, dando de esta manera una mayor certificación a los resultados obtenidos.

Para la obtención de los resultados se realizó el siguiente procedimiento:

1. Ejecución de la herramienta JMeter, con el objetivo de realizar la prueba de carga.

2. Creación de un plan de Pruebas para cada uno de los prototipos

3. Adición de un grupo de hilos, para simular el número de peticiones http realizadas al prototipo desplegado en Glassfish.

4. Adicionar en el grupo de hilos un mostrador y realizar la configuración de la petición HTTP.

5. Agregar dentro del mismo grupo de hilos un receptor de tipo Resumen y Agregado.

6. Ejecutar la prueba.

7. Observar resultados en los receptores anteriormente añadidos.

8. Ir a 6.

I1. Número de peticiones soportadas.

Este indicador hace referencia a la cantidad de peticiones que atiende la tecnología de desarrollo web, en un período determinado de tiempo, que para este proyecto la unidad será en segundos. Inmediatamente de ejecutar las pruebas se obtiene resultados individuales con los cuales se calcula la media o promedio, los errores relativos y porcentuales, desviación estándar e intervalos de confianza, con la finalidad de establecer datos estadísticos confiables que servirán a su vez en la evaluación del indicador. Las pruebas completas ejecutadas se pueden apreciar en la parte de anexos y sus dos primeros resultados, más el promedio en la Tabla 8.

Prototipo	**Prueba**	x_0	**(x)**	$\lvert x_0\text{-}x\rvert$	$\frac{(\lvert x_0 - x\rvert)}{x_0}$	$\bar{x}$	$(\bar{x}\text{-}x)$	$(\bar{x}\text{-}x)^2$
Tecnología EJB	1	68	68	0,0	0,0000	70,9	2,9	8,41
	2	72,8	73	0,8	0,0110		-2,1	4,41
Programación Tradicional	1	47,8	48	0,8	0,0167	52,95	4,95	24,50
	2	52,4	52	0,4	0,0076		0,95	0,9025

Tabla 7: Resultados de las pruebas para el número de peticiones soportadas

Fuente: Los Autores

Cálculo de errores para resultados de las pruebas en los prototipos

$$\varepsilon_a = \frac{(\sum_{i=1}^{n}(|x_0 - x|))}{n}$$

$$\varepsilon_r = \frac{\left(\sum_{i=1}^{n} \frac{(|x_0 - x|)}{x_0}\right)}{n}$$

$$\varepsilon_p = \frac{\left(\sum_{i=1}^{n} \frac{(|x_0 - x|)}{x_0}\right)}{n} \times 100\%$$

De donde:

$\boldsymbol{\varepsilon_a}$**:** Es el error absoluto de las pruebas.

$\boldsymbol{\varepsilon_r}$**:** Es el error relativo de las pruebas

$\boldsymbol{\varepsilon_p}$**:** Es el porcentaje de error relativo de las pruebas

$\boldsymbol{(\sum_{i=1}^{n}(|x_0\text{-}x|))}$**:** Es la sumatoria de la diferencia absoluta entre el valor verdadero y el valor tomado.

x_0: Es el resultado inicial obtenido de la prueba.
x: Es el resultado obtenido de la prueba.
n: Es el número de pruebas realizadas.

Para la prueba con la Tecnología EJB el error absoluto, relativo y porcentual con respecto a los resultados son:

$\varepsilon_{a(ejb)}=\frac{5,5}{10}$= 0,5 peticiones/segundo

$\varepsilon_{r(ejb)}=\frac{0,0705}{10}$=0,0071 peticiones/segundo

$\varepsilon_{r(ejb)}=\frac{0,0705}{10}*100\%$=0,71%

Para la prueba con la Programación Tradicional el error absoluto, relativo y relativo porcentual con respecto a los resultados son:

$\varepsilon_{a(pt)}=\frac{3,5}{10}$= 0,35 peticiones/segundo

$\varepsilon_{r(pt)}=\frac{0,0661}{10}$=0,0066 peticiones/segundo

$\varepsilon_{r(pt)}=\frac{0,0661}{10}*100\%$=0,66%

Cálculo de la desviación estándar:

$$S = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$

De donde:

$\boldsymbol{S}$: Desviación estándar

$\bar{X}$: Es el valor promedio de las prueba.

n: Es el número de pruebas realizadas.

El valor de la desviación estándar para la programación con La Tecnología EJB de servicios web Metro es:

$S_{(ejb)} = \frac{406,1}{9} = 45,1$

El valor de la desviación estándar para la programación tradicional:

$S_{(pt)} = \frac{1680,03}{9} = 186,67$

Cálculo del intervalo de confianza:

$$IC = \bar{x} \pm Z\left(\frac{S}{\sqrt{n-1}}\right)$$

De donde:

IC: Intervalos de confianza.

$\bar{X}$: Media de los valores tomados

$\mathcal{Z}$: Es el nivel de confianza elegido, determinado por el valor de α. Para una confianza del 95% (α= 0,05), este valor es de 1,96 según la tabla de valores $\mathcal{Z}$ más usados.
$\boldsymbol{S}$: Desviación estándar
$\boldsymbol{n}$: Es el número de pruebas realizadas.

Los intervalos de confianza para la programación con La Tecnología EJB son:

$IC_{(ejb)}$ = 70,9 ±1.96($\frac{45,1}{\sqrt{9}}$)

$IC_{(ejb)}$ = 70,9 ± 29,48

Los intervalos de confianza para la programación tradicional:

$IC_{(pt)}$ = 52,95 ±1.96($\frac{186,67}{\sqrt{9}}$)

$IC_{(pt)}$ = 52,95 ± 121,96

De esta forma el resumen de los resultados en número de peticiones soportadas, sobre segundo de las pruebas realizadas a los prototipos se aprecia en la Tabla 9.

Tecnología	**Frecuencia**	**Promedio**	**Error Relativo**	**Porcentaje Error**	$S_{\bar{X}}$	IC
Tecnología EJB	10	70,9	0,0071	0,71	45,12	[100,38; 41,42]
Programación Tradicional	10	52,95	0,0066	0,66	186,67	[174,91;-69,01]

Tabla 8: Resumen de los resultados de la medición del número de peticiones soportadas.

Fuente: Los Autores

Interpretación de Resultados

La tecnología EJB presenta un mayor número de peticiones soportadas por segundo al responder a 70,9 peticiones por segundo en comparación con la programación tradicional que responde a 52,95 peticiones por segundo, indicando una relación del 74,68%.

Análisis de diferencia significativa.

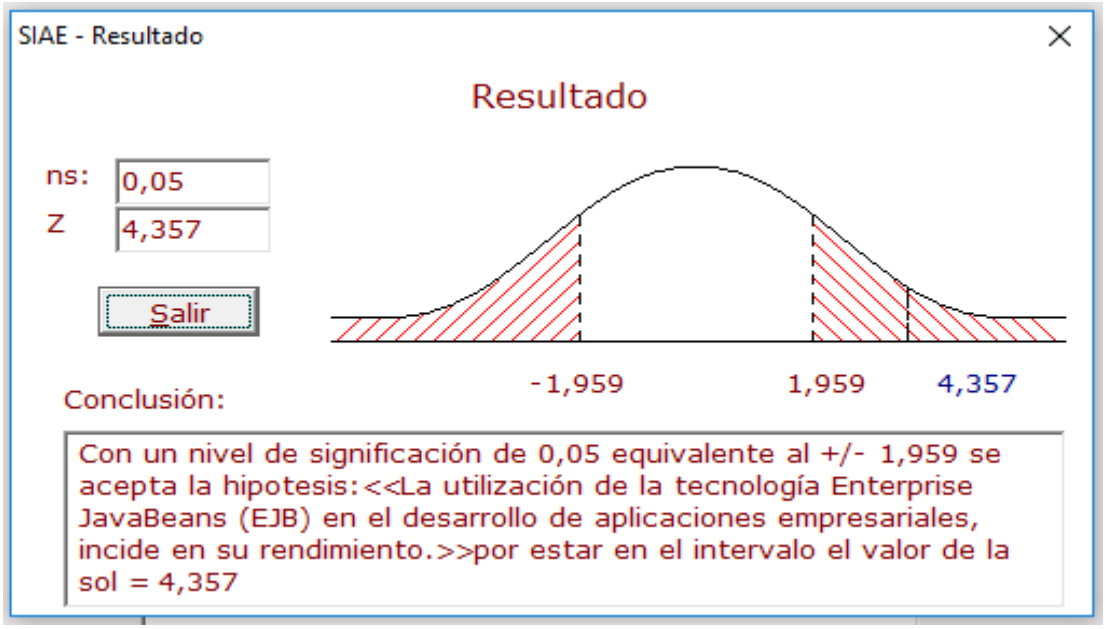


Ilustración 17: Indicador número de peticiones soportadas

Fuente: Los Autores

Según el análisis de la Herramienta SIAE 2.0, H1: La utilización de la tecnología Enterprise JavaBeans (EJB) en el desarrollo de aplicaciones empresariales, incide en el rendimiento **porque** se encuentra en el intervalo de la solución = 4,357. Por cuanto le asignamos una ponderación distinta entre las dos tecnologías analizadas.

Calificación

Rango de valores	Valoración Cuantitativa	Valoración Cualitativa	Forma Gráfica
67 a 75	4	Excelente	★★★★
58 a 66	3	Bueno	★★★☆
49 a 57	2	Regular	★★☆☆
Menor a 49	1	Malo	★☆☆☆

Tabla 9: Calificación del indicador número de peticiones soportadas

Fuente: Los Autores

La tecnología EJB soporta a 70,9 peticiones en un segundo, con lo cual y en base a la tabla de calificación para el presente indicador, se le asigna una calificación **Excelente**. Su equivalencia grafica será 4 estrellas.
La programación tradicional JSF atiende 52,95 peticiones en un segundo, se le asigna al indicador una calificación con valor **Regular**. Su equivalencia grafica será 2 estrellas.

I2. Tiempo de ejecución de solicitud.

Este indicador mide el tiempo que pasa desde que se envía la solicitud hasta recibir una respuesta. Dicho tiempo tendrá como unidad de medida el segundo.

Al ejecutar las pruebas se obtiene resultados individuales, con los cuales se calcula la media o promedio, los errores absolutos, relativos y porcentuales, desviación estándar e intervalos de confianza, con la finalidad de establecer datos estadísticos confiables que servirán a su vez en la evaluación del indicador. Las pruebas completas ejecutadas se pueden apreciar en la parte de anexos y sus dos primeros resultados, más el promedio en la Tabla 11.

Prototipo	Prueba	x_0	(x)	$\lvert x_0$-x$\rvert$	$\frac{(\lvert x_0 - x\rvert)}{x_0}$	$\bar{x}$	($\bar{x}$-x)	($\bar{x}$-x)²
Tecnología EJB	1	10,77	11	0,8	0,0713	**11,1**	0,10	0,01
	2	7,80	8	0,8	0,1024		3,1	9,61
Programación Tradicional	1	29,18	29	0,2	0,0062	**30,15**	1,15	1,32
	2	24,17	24	0,2	0,0070		6,15	37,82

Tabla 10: Resultados de las pruebas para el tiempo de ejecución de solicitud.

Fuente: Los Autores

Cálculo de errores para resultados de las pruebas en los prototipos

$$\varepsilon_a = \frac{(\sum_{i=1}^{n}(|x_0 - x|))}{n}$$

$$\varepsilon_r = \frac{\left(\sum_{i=1}^{n} \frac{(|x_0 - x|)}{x_0}\right)}{n}$$

$$\varepsilon_p = \frac{\left(\sum_{i=1}^{n} \frac{(|x_0 - x|)}{x_0}\right)}{n} \times 100\%$$

Para la prueba con la Tecnología EJB el error absoluto, relativo y porcentual con respecto a los resultados son:

$\varepsilon_{a(ejb)} = \frac{7,1}{10} = 0,71$ segundos

$\mathcal{E}_{r(ejb)}=\frac{0,6402}{10}$=0,064 segundos

$\mathcal{E}_{r(ejb)}=\frac{0,6402}{10}$*100%=6,40%

Para la prueba con la Programación Tradicional el error absoluto, relativo y relativo porcentual con respecto a los resultados son:

$\mathcal{E}_{a(pt)}=\frac{3,5}{10}$=0,35 segundos

$\mathcal{E}_{r(pt)}=\frac{0,1176}{10}$=0,0118 segundos

$\mathcal{E}_{r(pt)}=\frac{0,1176}{10}$*100%=1,176 %

Cálculo de la desviación estándar:

$$S = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$

El valor de la desviación estándar para la programación con La Tecnología EJB es:

$S_{(ejb)}=\frac{7,56}{9}$=7,84

El valor de la desviación estándar para la programación tradicional:

$S_{(pt)}=\frac{6380,9}{9}$=708,99

Cálculo del intervalo de confianza:

$$IC = \bar{x} \pm Z\left(\frac{S}{\sqrt{n-1}}\right)$$

Los intervalos de confianza para la programación con La Tecnología EJB son:

$IC_{(ejb)}$ =11,1 ±1.96($\frac{7,84}{\sqrt{9}}$)

$IC_{(ejb)}$ =11,1 ± 5,12

Los intervalos de confianza para la programación tradicional (Java Servlets):

$IC_{(pt)}$ = 30,15 ±1.96($\frac{708,99}{\sqrt{9}}$)

$IC_{(pt)}$ = 30,15 ± 463,21

De esta forma el resumen de los resultados en tiempo de ejecución de solicitud, en un segundo de las pruebas realizadas a los prototipos se aprecia en la Tabla 11.

Tecnología	**Frecuencia**	**Promedio**	**Error Relativo**	**Porcentaje Error**	$S_{\bar{X}}$	IC
Tecnología EJB	10	11,11	0,0640	6,40	7,84	[16,23;5,99]
Programación Tradicional	10	30,15	0,0118	1,18	78,99	[493,36;-433,05]

Tabla 11: Resumen de los resultados de la medición del tiempo de ejecución de solicitud.

Fuente: Los Autores

Interpretación de Resultados

La tecnología EJB presenta un menor tiempo de ejecución de solicitud, al responder 2173 peticiones en 11,11 segundos en comparación a la tecnología de programación tradicional que responde en un tiempo de 30,15 segundos, indicado una relación de 36,85%.

Debido a que el indicador es un tiempo, los rangos de valores son inversos, ya que a menor tiempo de ejecución de solicitud mayor rendimiento.

Análisis de diferencia significativa.

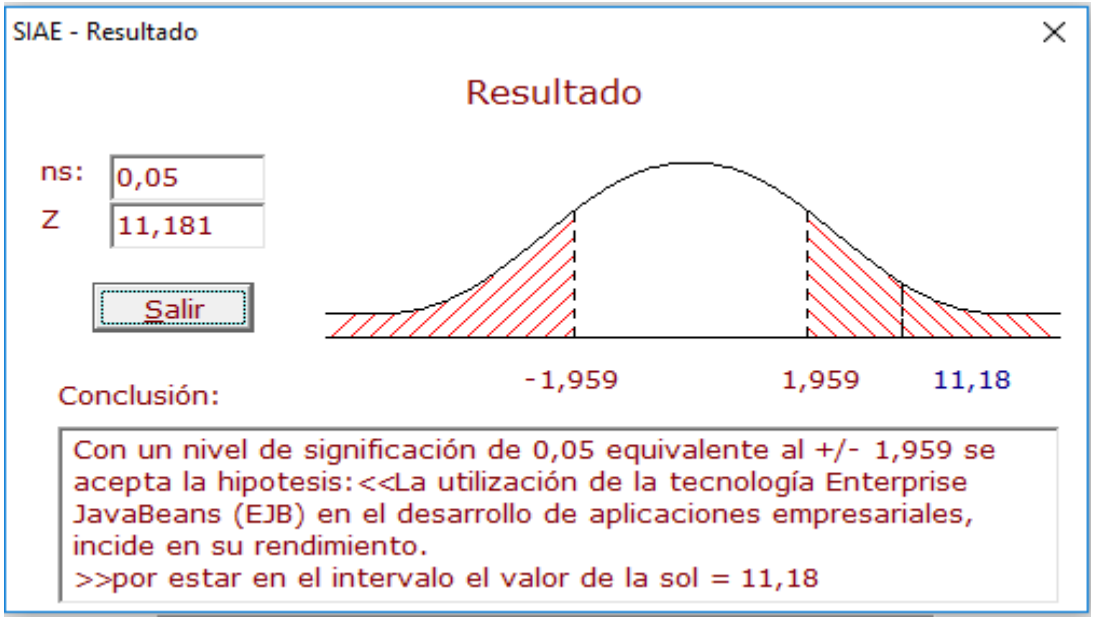


Ilustración 18: Resultado del tiempo de ejecución de solicitud

Fuente: Los Autores

Según el análisis de la Herramienta SIAE 2.0, H1: La utilización de la tecnología Enterprise JavaBeans (EJB) en el desarrollo de aplicaciones empresariales, incide en el rendimiento **porque** se encuentra en el intervalo de la solución = 11,18. Por cuanto le asignamos una ponderación distinta entre las dos tecnologías analizadas.

Calificación

Rango de valores	Valoración Cuantitativa	Valoración Cualitativa	Forma Gráfica
Menor a 10	4	Excelente	★★★★
10 a 20	3	Bueno	★★★☆
21 a 31	2	Regular	★★☆☆
Mayor a 31	1	Malo	★☆☆☆

Tabla 12: Calificación del indicador tiempo de ejecución de solicitud

Fuente: Los Autores

La tecnología de desarrollo EJB posee un tiempo de respuesta promedio de 11,1 segundos, con lo que en base a la tabla de calificación para el presente indicador se le asigna una calificación de Bueno. Su equivalencia grafica será de 3 estrellas.

La tecnología de programación tradicional al tener un tiempo de respuesta promedio de 30,15 segundos, se le asigna al indicador una calificación de regular. Su equivalencia gráfica será 2 estrellas.

3.9.1.1. Evaluación de resultados

Para evaluar el parámetro peticiones del usuario se utiliza las calificaciones obtenidas por los indicadores en las pruebas realizadas y de las fórmulas (1, 2 y 3) previamente definas en el método de evaluación.

Los resultados que se obtenga a partir del cálculo se presentarán en forma numérica y gráfica. Para después realizar una interpretación de los mismos. La calificación máxima del parámetro de comparación se establece mediante la suma de los valores máximos de calificación de cada indicador.

Es así que:

$$C_{max}=\sum_{i=0}^{n} Vi=4+4=8$$

La calificación numérica para la tecnología de programación EJB en el parámetro peticiones del usuario, se calcula mediante la sumatoria de las calificaciones de los indicadores obtenidas en las pruebas. De esta forma:

$$C_{(ejb)}=\sum_{i=0}^{n} Vi=4+3=7$$

La calificación numérica para la tecnología de programación tradicional en el parámetro peticiones del usuario, se calcula mediante la sumatoria de las calificaciones de los indicadores obtenidas en las pruebas. De esta forma:

$$C_{(pt)}=\sum_{i=0}^{n} Vi=2+2=4$$

El nivel de cumplimiento de la tecnología EJB expresado en porcentajes es igual a la división de la calificación obtenida por este entre la calificación máxima del parámetro y todo esto multiplicado por el cien por ciento.

$$P_{(ejb)}=\left(\frac{C_{(ejb)}}{C_{max}}\right)X100\% = \frac{7}{8}x100\%=87.5\%$$

De igual forma el nivel de cumplimiento de la programación tradicional expresado en porcentajes es igual a la división de la calificación obtenida por este entre la calificación máxima del parámetro y todo esto multiplicado por el cien por ciento.

$$P_{(pt)}=\left(\frac{C_{(pt)}}{C_{max}}\right)X100\% = \frac{4}{8}x100\%=50\%$$

Los resultados del cumplimiento de las tecnologías de desarrollo web se expresan en la figura 19.

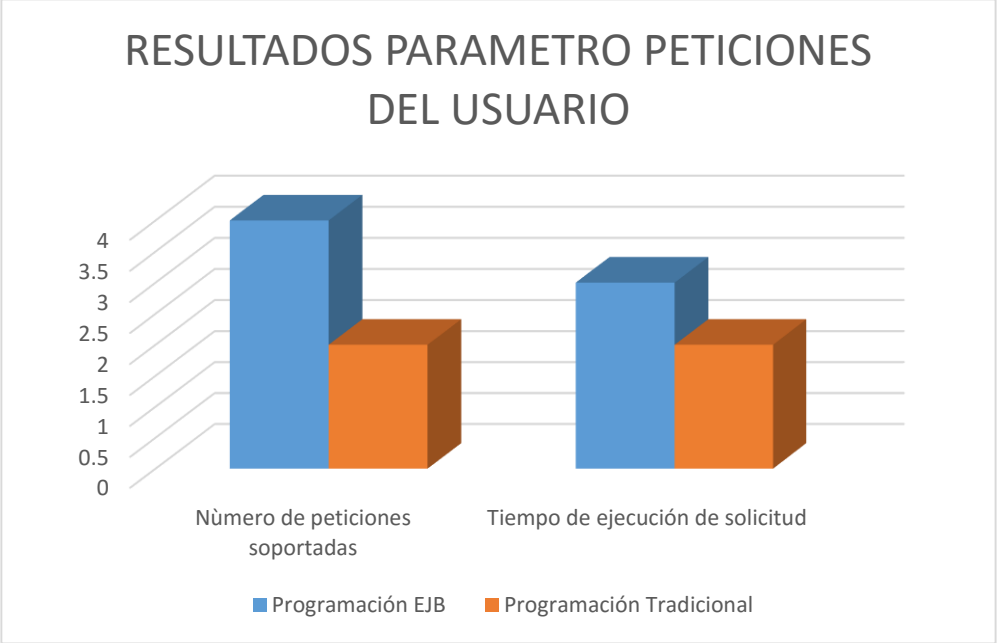


Ilustración 19: Resultados Parámetro Peticiones del Usuario

Fuente: Los Autores

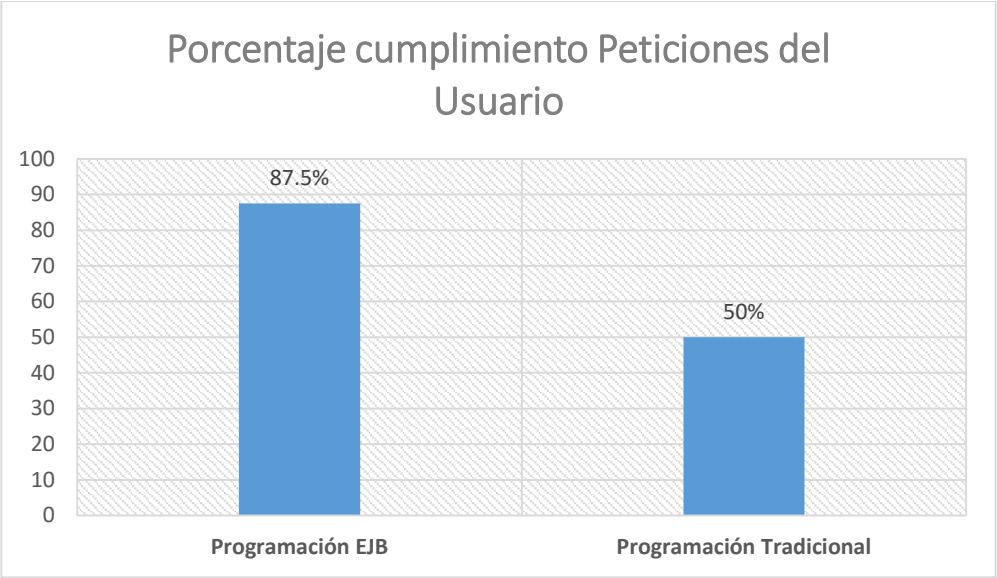


Ilustración 20: Porcentaje cumplimiento Peticiones del Usuario

Fuente: Los Autores

3.9.1.2. Interpretación de resultados

De acuerdo a los niveles de cumplimiento que permitió establecer el equivalente cualitativo de acuerdo a porcentajes, se valoró los resultados obtenidos por cada tecnología de programación. Los resultados obtenidos en el parámetro Peticiones del Usuario revelan que la tecnología de programación EJB posee un nivel de cumplimiento del 87,5% de los indicadores establecidos, lo que equivale a **Excelente**. Mientras que la tecnología de Programación Tradicional cumple con el 50% de los indicadores establecidos, lo que equivale a **Regular**.

3.9.2. Carga de usuario

Por medio de este parámetro se pretendió mostrar la eficacia en respuesta de la aplicación web, gestionando procesos de rendimiento tales como: número de usuarios concurrentes y número de usuarios en espera.

Indicador	**Descripción**
I1) Número de usuarios concurrentes	Número de usuarios logueados, utilizando la aplicación
I2) Número de usuarios en espera	Número de usuarios esperando recursos para ser atendidos.

Tabla 13: Indicadores del parámetro Carga de Usuario

Fuente: Los Autores

A partir de las aplicaciones prototipo descritas anteriormente se ejecutaron un grupo de pruebas de carga de manera automatizada con la herramienta JMeter, dando de esta manera una mayor certificación a los resultados obtenidos.

Para la obtención de los resultados se realizó el siguiente procedimiento:

1. Ejecución de la herramienta JMeter, con el objetivo de realizar la prueba de carga.

2. Creación de un plan de Pruebas para cada uno de los prototipos

3. Adición de un grupo de hilos, para simular el número de peticiones http realizadas al prototipo desplegado en GlassFish.

4. Adicionar en el grupo de hilos un mostrador y realizar la configuración de la petición HTTP.

5. Agregar dentro del mismo grupo de hilos un receptor de tipo Resumen y Agregado.

6. Ejecutar la prueba.

7. Observar resultados en los receptores anteriormente añadidos.

8. Ir a 6.

I1. Número de usuarios concurrentes

Cuando hablamos de usuarios concurrentes, queremos decir cuántos usuarios están logueados en la aplicación durante un instante dado. Es decir, puede que la aplicación tenga N usuarios registrados, pero es muy poco probable que todos operen sobre la aplicación al mismo tiempo.

Al elaborar las pruebas se obtuvo resultados individuales, con los cuales se calculó la media o promedio, los errores absolutos, relativos y porcentuales, desviación estándar e intervalos de confianza, con la finalidad de establecer datos estadísticos confiables que servirán a su vez en la evaluación del indicador. Las pruebas completas ejecutadas se pueden apreciar en la parte de anexos y sus dos primeros resultados, más el promedio en la Tabla 14.

Prototipo	**Prueba**	x_0	**(x)**	$\|x_0$-x$\|$	$\frac{(\|x_0 - x}{x_0}$	$\bar{x}$	$(\bar{x}$-x)	$(\bar{x}$-x)²
Tecnología EJB	1	1284	1284	0,0	0,0000	1323	39,0	1519,44
	2	689,93	689,9	0,9	0,0013		633,1	400790,3
Programación Tradicional	1	476,1	476	0,1	0,0002	806,53	330,53	109250,1
	2	837,04	837	0,0	0,0000		-30,47	928,4209

Tabla 14: Resultados de las pruebas para el número de usuarios concurrentes.

Fuente: Los Autores

Cálculo de errores para resultados de las pruebas en los prototipos

$$\varepsilon_a = \frac{(\sum_{i=1}^{n}(|x_0 - x|))}{n}$$

$$\varepsilon_r = \frac{\left(\sum_{i=1}^{n} \frac{(|x_0 - x|)}{x_0}\right)}{n}$$

$$\varepsilon_p = \frac{\left(\sum_{i=1}^{n} \frac{(|x_0 - x|)}{x_0}\right)}{n} \times 100\%$$

Para la prueba con la Tecnología EJB el error absoluto, relativo y porcentual con respecto a los resultados son:

$\mathcal{E}_{a(ejb)}=\frac{5,1}{10}$= 0,51 usuarios/segundo

$\mathcal{E}_{r(ejb)}=\frac{0,0039}{10}$= 0,0004 usuarios/segundo

$\mathcal{E}_{r(ejb)}=\frac{0,0039}{10}$=0,039%

Para la prueba con la Programación Tradicional el error absoluto, relativo y relativo porcentual con respecto a los resultados son:

$\mathcal{E}_{a(pt)}=\frac{3,3}{10}$=0,33 usuarios/segundo

$\mathcal{E}_{r(pt)}=\frac{0,0041}{10}$= 0,0004 usuarios/segundo

$\mathcal{E}_{r(pt)}=\frac{0,0041}{10}$= 0,041%

Cálculo de la desviación estándar:

$$S = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$

El valor de la desviación estándar para la programación con La Tecnología EJB de servicios web Metro es:

$S_{(ejb)}=\frac{995564,84}{9}$=110618,32

El valor de la desviación estándar para la programación tradicional:

$S_{(pt)}=\frac{1010106,51}{9}$=112234,06

Cálculo del intervalo de confianza:

$$IC = \bar{x} \pm Z\left(\frac{S}{\sqrt{n-1}}\right)$$

Los intervalos de confianza para la programación con La Tecnología EJB son:

$$IC_{(ejb)} = 1323 \pm 1.96(\frac{110618,32}{\sqrt{9}})$$

$$IC_{(ejb)} = 1323 \pm 72270,63$$

Los intervalos de confianza para la programación tradicional

$$IC_{(pt)} = 806,53 \pm 1.96(\frac{112234,06}{\sqrt{9}})$$

$$IC_{(pt)} = 806,53 \pm 73326,25$$

De esta forma el resumen de los resultados en número usuarios concurrentes sobre segundo de las pruebas realizadas a los prototipos se aprecia en la Tabla 15.

Tecnología	**Frecuencia**	**Promedio**	**Error Relativo**	**Porcentaje Error**	$S_{\bar{X}}$	IC
Tecnología EJB	10	1323	0,0004	0,0385	110618,32	(73593,64; -70947,62)
Programación Tradicional	10	806,53	0,0004	0,0408	112234,06	(74132,78; -72519,72)

Tabla 15: Resumen de los resultados de la medición del número de usuarios concurrentes.

Fuente: Los Autores

Interpretación de Resultados

La tecnología EJB presenta un mayor número de usuarios concurrentes al responder 1323 usuarios por segundo, en comparación a la programación tradicional web que responde 806,53 usuarios concurrentes por segundo, indicado una relación de 60,96%. Debido a que el indicador es un número de usuarios, los rangos de valores no son inversos, ya que a mayor número de usuarios mayor rendimiento.

Análisis de Diferencia Significativa

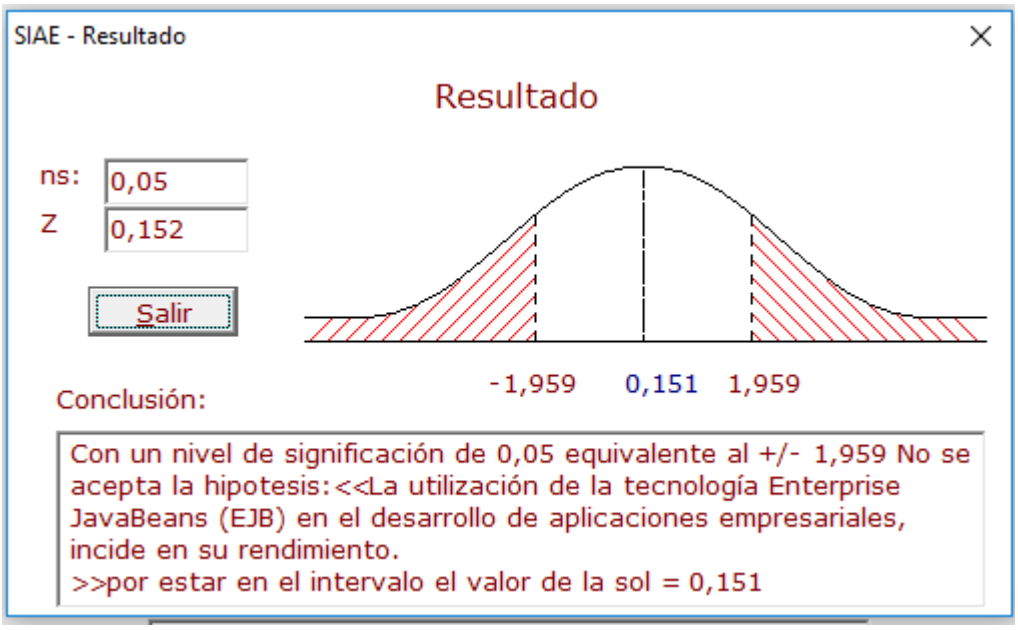


Ilustración 21: Indicador número de usuarios concurrentes

Fuente: Los Autores

Con un nivel de significación de 0,05 equivalente al +/- 1,959 no se acepta la hipótesis H1. En la cual la utilización de la tecnología Enterprise JavaBeans (EJB) en el desarrollo de aplicaciones empresariales, no incide en su rendimiento **porque** sus valores medidos según la herramienta SIAE 2.0 no son significativos ya que se encuentra en el intervalo el valor de la solución en **0,151**, por cuanto le ubicamos con la misma ponderación.

Calificación

Rango de valores	**Valoración Cuantitativa**	**Valoración Cualitativa**	**Forma Gráfica**
1000 a 1400	4	Excelente	★★★★
600 a 999	3	Bueno	★★★☆
200 a 599	2	Regular	★★☆☆
Menor a 200	1	Malo	★☆☆☆

Tabla 16: Calificación del indicador número de usuarios concurrentes

Fuente: Los Autores

La tecnología EJB soporta 1323 usuarios por segundo, se le asigna al indicador una calificación con valor **Bueno.** Su equivalencia grafica será 4 estrellas. La programación tradicional Web soporta 806,53 usuarios por un segundo, se le asigna al indicador una calificación con valor **Bueno**. Su equivalencia grafica será 3 estrellas.

I2. Número de usuarios en espera.

Cuando hablamos de usuarios en espera, queremos decir cuántos usuarios están o no logueados esperando recursos un instante dado. Al elaborar las pruebas se obtiene resultados individuales, con los cuales se calcula la media o promedio, los errores absolutos, relativos y porcentuales, desviación estándar e intervalos de confianza, con la finalidad de establecer datos estadísticos confiables que servirán a su vez en la evaluación del indicador. Las pruebas completas ejecutadas se pueden apreciar en la parte de anexos y sus dos primeros resultados, más el promedio en la Tabla 17.

Prototipo	Prueba	x_0	(x)	$\lvert x_0$-x$\rvert$	$\frac{(\lvert x_0 - x\rvert)}{x_0}$	$\bar{x}$	($\bar{x}$-x)	($\bar{x}$-x)²
Tecnología EJB	1	888,97	889	1,0	0,0011	850	-39,0	1519,44
	2	1483,1	1483	0,1	0,0000		-633,0	400689
Programación Tradicional	1	1696,90	1697	0,9	0,0005	1366,51	-330,49	109223,6
	2	1336	1336	1,0	0,0007		30,51	930,8601

Tabla 17: Resultados de las pruebas para el número de usuarios en espera.

Fuente: Los Autores

Cálculo de errores para resultados de las pruebas en los prototipos

$$\varepsilon_a = \frac{(\sum_{i=1}^{n}(|x_0 - x|))}{n}$$

$$\varepsilon_r = \frac{\left(\sum_{i=1}^{n} \frac{(|x_0 - x|)}{x_0}\right)}{n}$$

$$\varepsilon_p = \frac{\left(\sum_{i=1}^{n} \frac{(|x_0 - x|)}{x_0}\right)}{n} \times 100\%$$

Para la prueba con la Tecnología EJB el error absoluto, relativo y porcentual con respecto a los resultados son:

$\mathcal{E}_{a(ejb)}=\frac{4,9}{10}$=0,49 usuarios/segundo

$\mathcal{E}_{r(ejb)}=\frac{0,0058}{10}$=0,0006 usuarios/segundo

$\mathcal{E}_{r(ejb)}=\frac{0,0058}{10}$=0,058%

Para la prueba con la Programación Tradicional el error absoluto, relativo y relativo porcentual con respecto a los resultados son:

$\mathcal{E}_{a(pt)}=\frac{7,1}{10}$=0,71 usuarios/segundo

$\mathcal{E}_{r(pt)}=\frac{0,0052}{10}$= 0,0005 usuarios/segundo

$\mathcal{E}_{r(pt)}=\frac{0,0052}{10}$=0,052 %

Cálculo de la desviación estándar:

$$S = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$

El valor de la desviación estándar para la programación con La Tecnología EJB de servicios web Metro es:

$$S_{(ejb)} = \frac{1023703,031}{9} = 113744,78$$

El valor de la desviación estándar para la programación tradicional:

$$S_{(pt)} = \frac{1010106,50}{9} = 112234,06$$

Cálculo del intervalo de confianza:

$$IC = \bar{x} \pm Z\left(\frac{S}{\sqrt{n-1}}\right)$$

Los intervalos de confianza para la programación con La Tecnología EJB son:

$$IC_{(ejb)} = 850 \pm 1.96\left(\frac{}{\sqrt{9}}\right)$$

$$IC_{(ejb)} = 850 \pm$$

Los intervalos de confianza para la programación tradicional.

$$IC_{(pt)} = 1366,51 \pm 1.96\left(\frac{74313,26}{\sqrt{9}}\right)$$

$$IC_{(pt)} = 1366,51 \pm 73326,2497$$

De esta forma el resumen de los resultados en número de usuarios en espera, sobre segundo de las pruebas realizadas a los prototipos se aprecia en la Tabla 19.

Tecnología	**Frecuencia**	**Promedio**	**Error Relativo**	**Porcentaje Error**	$S_{\bar{X}}$	IC
Tecnología EJB	10	850	0,0006	0,0576	113744,78	[75163,25;-73463,27]

Programación Tradicional	10	1366,5	0,0005	0,0520	112234,06	[74692,76;-71959,74]

Tabla 18: Resumen de los resultados de la medición del número de usuarios en espera.

Fuente: Los Autores

Interpretación de Resultados

La tecnología EJB presenta un menor número de usuarios en espera al mantener 850 usuarios en espera, en comparación a la programación tradicional de servicios web que mantiene 1366,5 usuarios en espera, indicado una relación de 62,20%.

Debido a que el indicador es un número, los rangos de valores no son inversos, ya que a menor número mayor rendimiento.

Análisis de diferencia significativa.

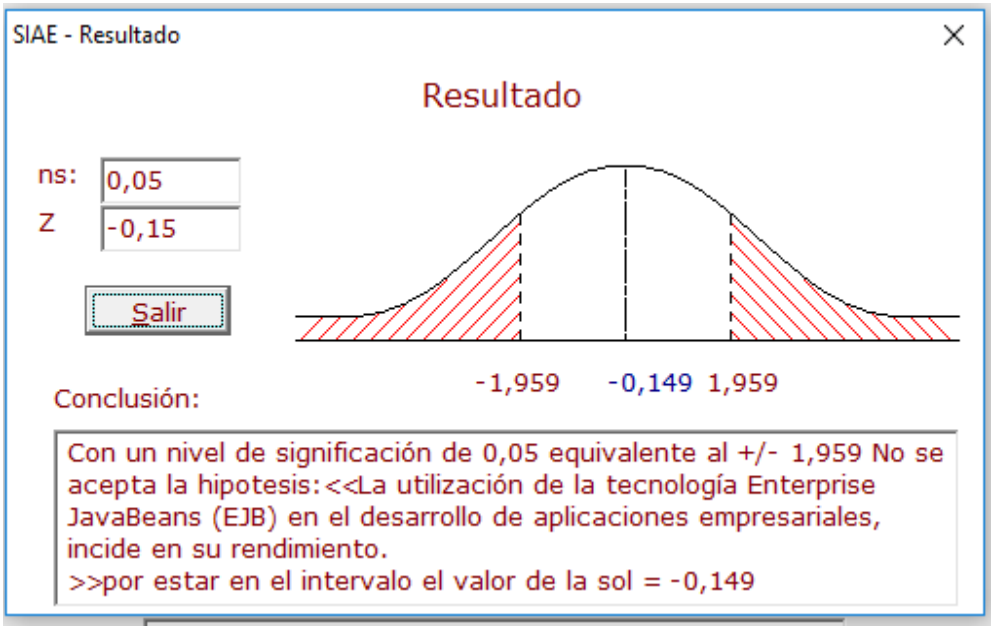


Ilustración 22: Indicador número de usuarios en espera

Fuente: Los Autores

Con un nivel de significación de 0,05 equivalente al +/- 1,959 no se acepta la hipótesis H1. En la cual la utilización de La Tecnología Enterprise JavaBeans (EJB) en el desarrollo de aplicaciones empresariales, no incide en su rendimiento **porque** sus valores medidos según la herramienta SIAE 2.0 no son significativos ya que se encuentra en el intervalo el valor de la solución en **-0,149**, por cuanto le ubicamos con la misma ponderación.

Calificación.

Rango de valores	**Valoración Cuantitativa**	**Valoración Cualitativa**	**Forma Gráfica**
<=200 a 599	4	Excelente	
600 a 999	3	Bueno	
1000 a 1400	2	Regular	
Mayor a 1400	1	Malo	

Tabla 19: Calificación del indicador de número de usuarios en espera.

Fuente: Los Autores

La tecnología EJB mantiene 850 usuarios en espera por segundo, se le asigna al indicador una calificación con valor **Bueno**. Su equivalencia grafica será 3 estrellas. La programación tradicional web mantiene 1366,5 usuarios en espera en un segundo, se le asigna al indicador una calificación con valor **Bueno**. Su equivalencia grafica será 3 estrellas.

3.9.2.1. Evaluación de resultados

Para evaluar el parámetro peticiones del usuario se utiliza las calificaciones obtenidas por los indicadores en las pruebas realizadas y de las fórmulas (1, 2 y 3) previamente definas en el método de evaluación.

Los resultados que se obtenga a partir del cálculo se presentarán en forma numérica y gráfica. Para después realizar una interpretación de los mismos. La calificación máxima del parámetro de comparación se establece mediante la suma de los valores máximos de calificación de cada indicador.

Es así que:

$$C_{max}=\sum_{i=0}^{n} Vi=4+4=8$$

La calificación numérica para la tecnología de programación EJB en el parámetro carga de usuarios, se calcula mediante la sumatoria de las calificaciones de los indicadores obtenidas en las pruebas. De esta forma:

$$C_{(ejb)}=\sum_{i=0}^{n} Vi=3+3=6$$

La calificación numérica para la tecnología de programación tradicional en el parámetro carga de usuarios, se calcula mediante la sumatoria de las calificaciones de los indicadores obtenidas en las pruebas. De esta forma:

$$C_{(pt)}=\sum_{i=0}^{n} Vi=3+3=6$$

El nivel de cumplimiento de la tecnología EJB expresado en porcentajes es igual a la división de la calificación obtenida por este entre la calificación máxima del parámetro y todo esto multiplicado por el cien por ciento.

$$P_{(ejb)}=\left(\frac{C_{(ejb)}}{C_{max}}\right) X100\% = \frac{6}{8} x100\%=75\%$$

De igual forma el nivel de cumplimiento de la programación tradicional expresado en porcentajes es igual a la división de la calificación obtenida por este entre la calificación máxima del parámetro y todo esto multiplicado por el cien por ciento.

$$P_{(pt)}=\left(\frac{C_{(pt)}}{C_{max}}\right) X100\% = \frac{6}{8} x100\%=75\%$$

Los resultados del cumplimiento de las tecnologías de desarrollo web se expresan en la figura siguiente.

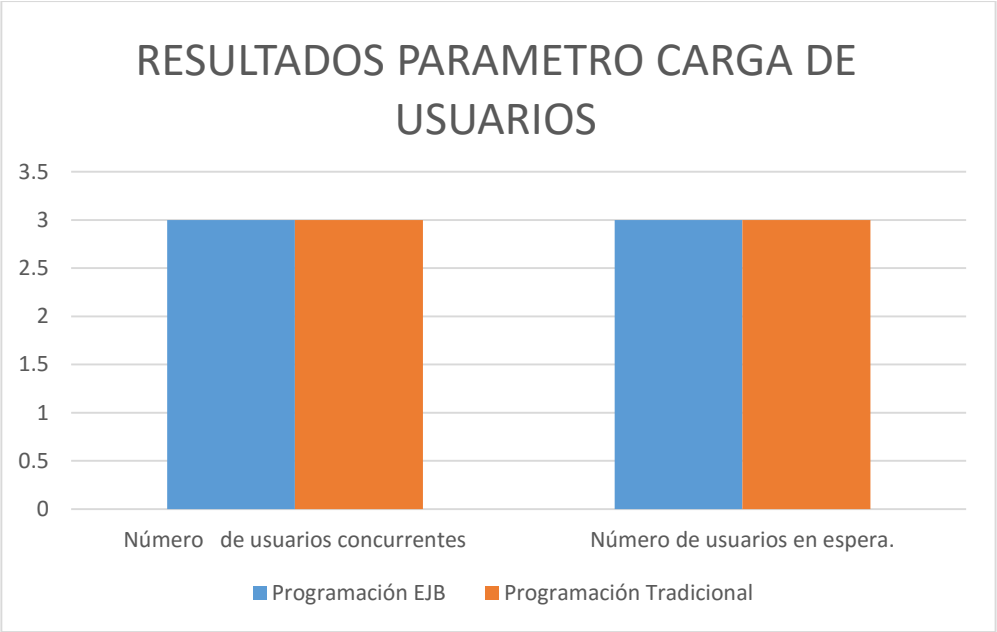


Ilustración 23: Resultados Parámetro Carga de Usuarios

Fuente: Los Autores

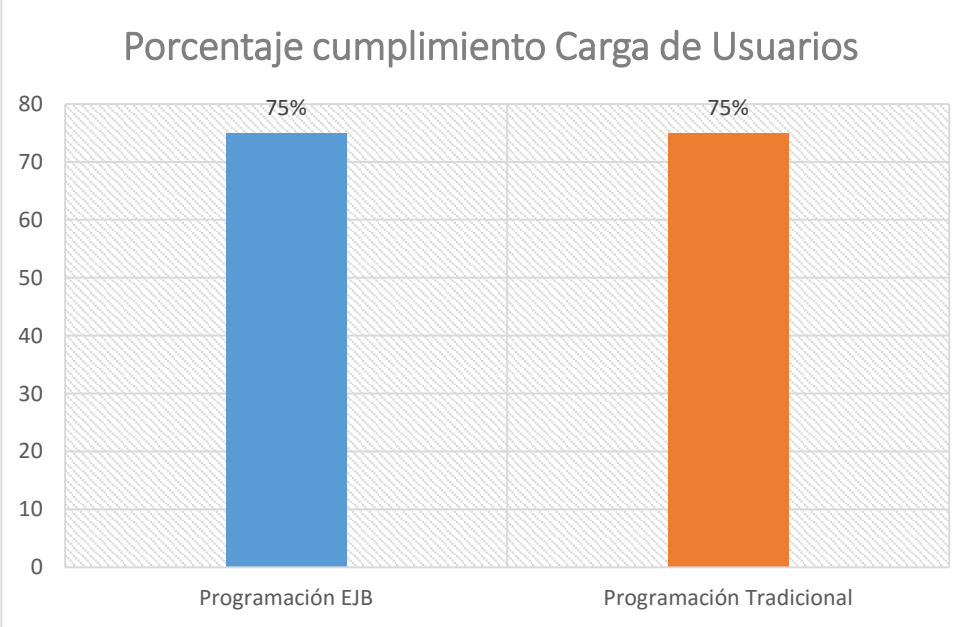


Ilustración 24: Porcentaje cumplimiento Carga de Usuarios

Fuente: Los Autores

3.9.2.2. Interpretación de resultados

De acuerdo a los niveles de cumplimiento que permite establecer el equivalente cualitativo de acuerdo a porcentajes, se valorará los resultados obtenidos por cada tecnología de programación web.

Los resultados obtenidos en el parámetro Carga de Usuarios revelan que la tecnología de programación EJB posee un nivel de cumplimiento del 75% de los indicadores establecidos, lo que equivale a B**uena**. De igual la tecnología de Programación Tradicional cumple con el 75% de los indicadores establecidos, lo que equivale a **Buena**.

3.9.3. Uso de hardware

Con este parámetro se pretende determinar la gestión de valores y códigos que la aplicación requiere almacenar y ejecutar dichos procesos a nivel de memoria RAM y procesador necesarios para la satisfactoria ejecución.

Indicador	**Descripción**
Memoria RAM	Cantidad de espacio en Memoria RAM necesaria para almacenar un promedio de peticiones concurrentes.
Uso del Procesador	Porcentaje de uso del procesador por la tecnología de desarrollo web.

Tabla 20: Descripción de los indicadores Uso de Hardware.

Fuente: Autores

A partir de las aplicaciones prototipo descritas anteriormente se ejecutaron 10 pruebas de carga de manera automatizada con la herramienta JMeter, captando los resultados mediante la herramienta Java Visual VM.

Para la obtención de los resultados se realizó el siguiente procedimiento:

1. Ejecución de la herramienta JMeter, con el objetivo de realizar la prueba de carga.
2. Creación de un plan de Pruebas para cada uno de los prototipos
3. Adición de un grupo de hilos, para simular el número de peticiones http realizadas al prototipo desplegado en Glassfish.
4. Adicionar al grupo de hilos un mostrador y realizar la configuración de la petición HTTP.
5. Ejecutar la prueba.
6. Observar resultados en la herramienta de Java visual VM
7. Ir a 6.

I1. Memoria RAM

Este indicador se hace referencia a la cantidad de espacio en Memoria RAM necesaria para recolectar un promedio de peticiones concurrentes realizadas a la aplicación web. Inmediatamente de ejecutar las pruebas se obtiene resultados individuales con los cuales se calcula la media o promedio, los errores absolutos, relativos y porcentuales, desviación

estándar e intervalos de confianza, con la finalidad de establecer datos estadísticos confiables que servirán a su vez en la evaluación del indicador. Las pruebas ejecutadas sobre el presente indicador se pude apreciar en la parte de anexos y sus dos primeros resultados en la Tabla 21.

Prototipo	Prueba	x_0	(x)	$\|x_0$-x$\|$	$\frac{(\|x_0 - x\|)}{x_0}$	$\bar{x}$	($\bar{x}$-x)	($\bar{x}$-x)²
Tecnología EJB	1	246,25	246	0,3	0,0010	**201,1**	-44,9	2016,01
	2	115,24	115	0,2	0,0021		86,1	7413,21
Programación Tradicional	1	167,63	168	0,6	0,0038	**182,38**	14,38	206,78
	2	171,34	171	0,3	0,0020		11,38	129,50

Tabla 21: Resultados de las pruebas para el uso de memoria RAM

Fuente: Los Autores

Cálculo de errores para resultados de las pruebas en los prototipos

$$\varepsilon_a = \frac{(\sum_{i=1}^{n}(|x_0 - x|))}{n}$$

$$\varepsilon_r = \frac{\left(\sum_{i=1}^{n} \frac{(|x_0 - x|)}{x_0}\right)}{n}$$

$$\varepsilon_p = \frac{\left(\sum_{i=1}^{n} \frac{(|x_0 - x|)}{x_0}\right)}{n} \times 100\%$$

Para las pruebas con la Programación EJB el error absoluto, relativo y relativo porcentual con respecto a los resultados son:

$\mathcal{E}_{a(ejb)} = \frac{4,7}{10} = 0,47$ MB

$\mathcal{E}_{r(ejb)} = \frac{0,0233}{10} = 0,0023$ MB

$\mathcal{E}_{r(ejb)} = \frac{0,0233}{10} = 0,233$ %

Para la prueba con la Programación Tradicional el error absoluto, relativo y relativo porcentual con respecto a los resultados son:

$\mathcal{E}_{a(pt)}=\frac{4,8}{10}$=0,48 MB

$\mathcal{E}_{r(pt)}=\frac{0,0266}{10}$=0,0027 MB

$\mathcal{E}_{r(pt)}=\frac{0,0266}{10}$=0,266 %

Cálculo de la desviación estándar:

$$S = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$

El valor de la desviación estándar para la programación con La Tecnología EJB de servicios web Metro es:

$S_{(ejb)}=\frac{35422,1}{9}$=3935,79

El valor de la desviación estándar para la programación tradicional:

$S_{(pt)}=\frac{5702,64}{9}$=633,63

Cálculo del intervalo de confianza:

$$IC = \bar{x} \pm Z\left(\frac{S}{\sqrt{n-1}}\right)$$

Los intervalos de confianza para la programación con La Tecnología EJB son:

$IC_{(ejb)}$ = 201,1 ±1.96($\frac{2187,4}{\sqrt{9}}$)

$IC_{(ejb)}$ = 201,1 ± 2571,38

Los intervalos de confianza para la programación tradicional (Java Servlets):

$IC_{(pt)}$ = 182,38±1.96($\frac{2648,7}{\sqrt{9}}$)

$IC_{(pt)}$ = 182,38 ± 413,9697126

De esta forma el resumen de los resultados en Megabytes de las pruebas realizadas a los prototipos se aprecian en la Tabla 22.

Tecnología	**Frecuencia**	**Promedio**	**Error Relativo**	**Porcentaje Error**	$S_{\bar{X}}$	IC
Tecnología EJB	10	201,1	0,0023	0,2329	935,79	[2772,45 ;-2370,31]
Programación Tradicional	10	182,38	0,0027	0,2656	633,63	[596,35 ; -231,59]

Tabla 22: Resumen de los resultados de la medición del uso de memoria RAM.

Fuente: Los Autores

Interpretación de Resultados

La tecnología EJB presenta una mayor cantidad de uso de memoria RAM al ocupar 201,1 MB en comparación a la tecnología de programación tradicional que ocupa 182,38 MB. Estableciendo una relación del 90,71%.

Debido a que el indicador es espacio en memoria, los rangos de valores son inversos, ya que el objetivo es reducir requerimientos de hardware.

Análisis de Diferencia Significativa

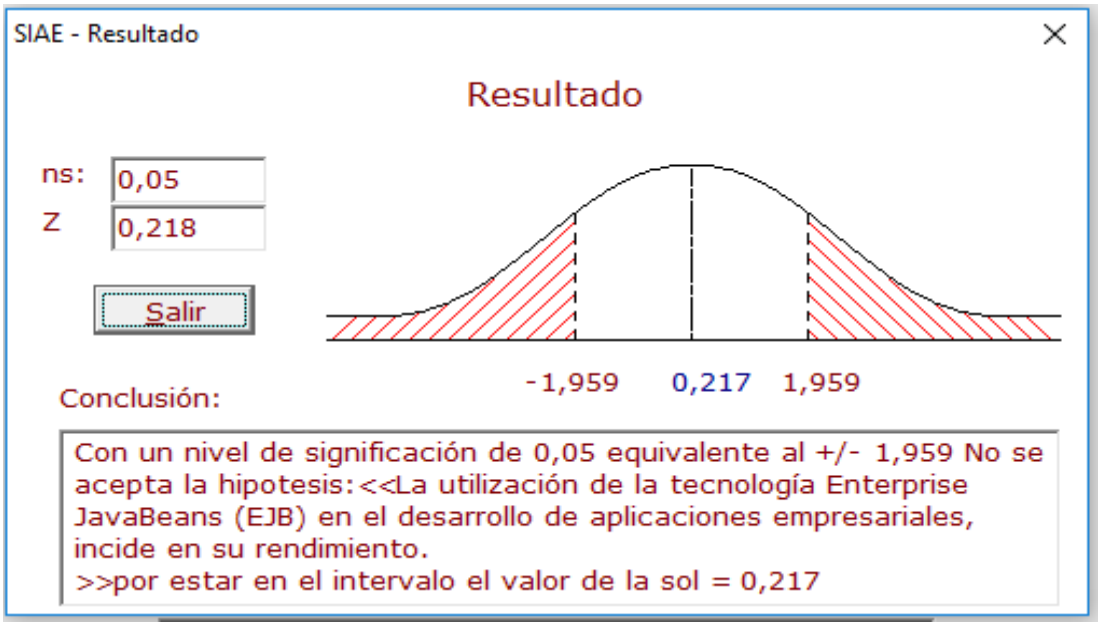


Ilustración 25: Indicador Memoria RAM

Fuente: Los Autores

Con un nivel de significación de 0,05 equivalente al +/- 1,959 no se acepta la hipótesis H1. En la cual la utilización de La Tecnología Enterprise JavaBeans (EJB) en el desarrollo de aplicaciones empresariales, no incide en su rendimiento **porque** sus valores medidos según la herramienta SIAE 2.0 no son significativos ya que se encuentra en el intervalo el valor de la solución en **0,217**, por cuanto le ubicamos con la misma ponderación.

Calificación

Rango de valores	**Valoración Cuantitativa**	**Valoración Cualitativa**	**Forma Gráfica**
Menor a 145, a 170	4	Excelente	★★★★
171 a 201	3	Bueno	★★★☆
202 a 226	2	Regular	★★☆☆
Mayor a 226	1	Malo	★☆☆☆

Tabla 23: Calificación del indicador uso de memoria RAM

Fuente: Los Autores

La tecnología EJB hace uso de 201 MB de memoria RAM, con lo que en base a la tabla de calificación para el presente indicador, se le asigna una calificación de regular, su equivalencia grafica es de 2 estrellas. Al igual que la programación tradicional consume 182,38 MB de memoria RAM, con lo cual se le asigna una calificación de Regular, su representación gráfica es de 2 estrellas, esto de acuerdo al análisis de diferencia significativa.

I2.Uso de Procesador

Este indicador mide el porcentaje de uso del procesador por la tecnología de servicios web. Al ejecutar las pruebas se obtiene resultados individuales, con los cuales se calcula la media o promedio, los errores relativos y porcentuales, desviación estándar e intervalos de confianza, con la finalidad de establecer datos estadísticos confiables que servirán a su vez en la evaluación del indicador. Las capturas de las pruebas ejecutadas sobre el presente indicador se pude apreciar en la parte de anexos y sus resultados de los dos primeros valores en la Tabla 24.

Prototipo	Prueba	x_0	(x)	$\lvert x_0$-x$\rvert$	$\frac{(\lvert x_0 - x\rvert)}{x_0}$	$\bar{x}$	($\bar{x}$-x)	($\bar{x}$-x)²
Tecnología EJB	1	84,3	84	0,3	0,0036	**84,0**	0,0	0
	2	80,6	81	0,6	0,0074		3,0	9
Programación Tradicional	1	46,3	46	0,3	0,0065	**33,58**	-12,42	154,26
	2	45,1	45	0,1	0,0022		-11,42	130,42

Tabla 24: Resultados de las pruebas para el uso de procesador

Fuente: Los Autores

Cálculo de errores para resultados de las pruebas en los prototipos

$$\varepsilon_a = \frac{(\sum_{i=1}^{n}(|x_0 - x|))}{n}$$

$$\varepsilon_r = \frac{\left(\sum_{i=1}^{n} \frac{(|x_0 - x|)}{x_0}\right)}{n}$$

$$\varepsilon_p = \frac{\left(\sum_{i=1}^{n} \frac{(|x_0 - x|)}{x_0}\right)}{n} \times 100\%$$

Para las pruebas con la tecnología EJB el error absoluto, relativo y relativo porcentual con respecto a los resultados son:

$\mathcal{E}_{a(ejb)}=\frac{4,6}{10}$=0,46 %CPU

$\mathcal{E}_{r(ejb)}=\frac{0,0548}{10}$=0,0055%CPU

$\mathcal{E}_{r(ejb)}=\frac{0,0548}{10}$=0,548%

Para la prueba con programación tradicional el error absoluto, relativo y relativo porcentual con respecto a los resultados son:

$\mathcal{E}_{a(pt)}=\frac{4,8}{10}$=0,48 %CPU

$\mathcal{E}_{r(pt)}=\frac{0,1429}{10}$=0,0143 %CPU

$\mathcal{E}_{r(pt)}=\frac{0,1429}{10}$=1,429%

Cálculo de la desviación estándar:

$$S = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$

El valor de la desviación estándar para la programación con La Tecnología EJB de servicios web es:

$S_{(ejb)}=\frac{32}{9}$=3,56

El valor de la desviación estándar para la programación tradicional:

$$S_{(pt)}=\frac{874,24}{9}=97,14$$

Cálculo del intervalo de confianza:

$$IC = \bar{x} \pm Z\left(\frac{S}{\sqrt{n-1}}\right)$$

Los intervalos de confianza para la programación con La Tecnología EJB son:

$$IC_{(ejb)} = 84,0 \pm 1.96(\frac{3,56}{\sqrt{9}})$$

$$IC_{(ejb)} = 84,0 \pm 2,32$$

Los intervalos de confianza para la programación tradicional

$$IC_{(pt)} = 33,6 \pm 1.96(\frac{97,14}{\sqrt{9}})$$

$$IC_{(pt)} = 33,6 \pm 63,46$$

De esta forma el resumen de los resultados en Megabytes de las pruebas realizadas a los prototipos se aprecian en la Tabla Siguiente.

Tecnología	**Frecuencia**	**Promedio**	**Error Relativo**	**Porcentaje Error**	$S_{\bar{X}}$	IC
Tecnología EJB	10	84	0,0055	0,5479 %	3,56	[86,28;81,64]
Programación Tradicional.	10	33,58	0,0143	1,43%	97,14	[97,04;-29,88]

Tabla 25: Resumen de los resultados de la medición del uso de procesador

Fuente: Los Autores

Interpretación de Resultados

La tecnología EJB presenta un mayor uso del procesador al atender 2173 peticiones con el 84% del CPU. A diferencia de la programación tradicional que responde haciendo uso de 33,58% del CPU. Debido a que el indicador es un recurso de hardware los rangos de valores son inversos.

Análisis de diferencia significativa

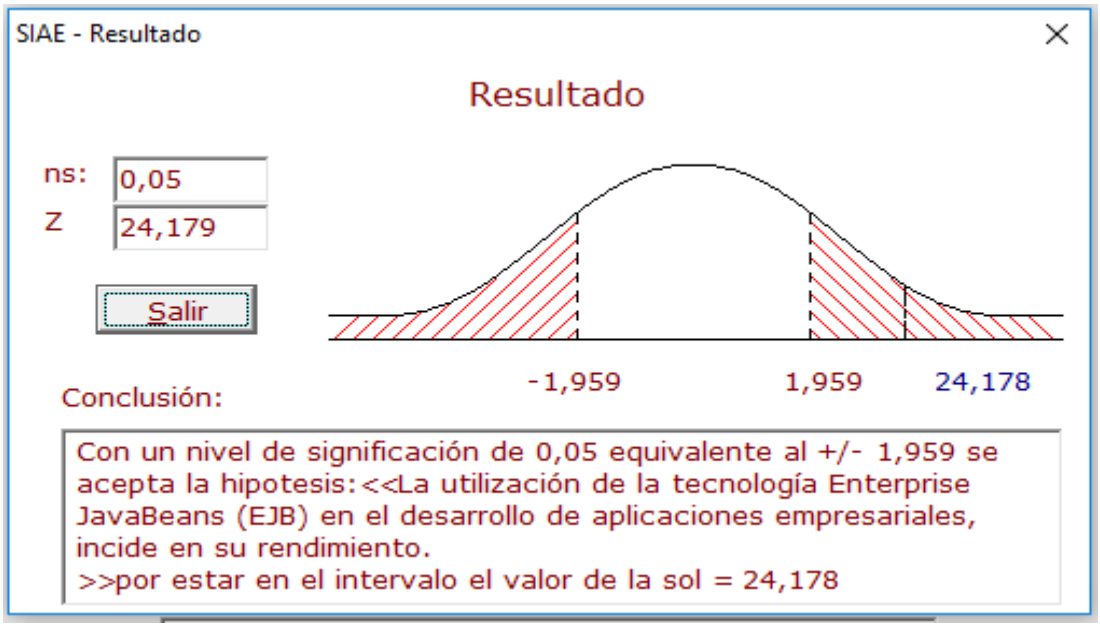


Ilustración 26: Análisis significativo del uso del procesador

Fuente: Los Autores

Con un nivel de significación de 0,05 equivalente al +/- 1,959 se acepta la hipótesis H1. En la cual la utilización de La Tecnología Enterprise JavaBeans (EJB) en el desarrollo de aplicaciones empresariales, incide en su rendimiento **porque** sus valores medidos según la herramienta SIAE 2.0 son significativos ya que se encuentra en el intervalo el valor de la solución en **24,178**, por cuanto le ubicamos con diferente ponderación.

Calificación

Rango de valores	**Valoración Cuantitativa**	**Valoración Cualitativa**	**Forma Gráfica**
Menor a 25	4	Excelente	★★★★
25 a 50	3	Bueno	★★★☆

51 a 75	2	Regular	
Mayor a 75	1	Malo	

Tabla 26: Calificación del indicador uso de procesador

Fuente: Los Autores

La tecnología EJB consume un 84% del procesador con lo cual y en base a la tabla de calificación se le asigna una calificación de Malo, su equivalencia gráfica es de 1 estrellas. La programación tradicional al tener un consumo de 33,58% de consumo del procesador recibe una calificación de Bueno, su equivalencia gráfica de 3 estrellas.

3.9.3.1. Evaluación de resultados

Para evaluar el parámetro uso de hardware se utiliza las calificaciones obtenidas por los indicadores en las pruebas realizadas y de las fórmulas (1, 2 y 3) previamente definidas en la metodología de investigación.

Los resultados que se obtenga a partir del cálculo, se presentarán en forma numérica y gráfica, para después realizar la interpretación de los mismos La calificación máxima del parámetro de comparación se establece mediante la suma de los valores máximos de calificación de cada indicador. Demostrando que:

Es así que:

$$C_{max}=\sum_{i=0}^{n} Vi=4+4=8$$

La calificación numérica para la tecnología de programación EJB en el parámetro uso de hardware, se calcula mediante la sumatoria de las calificaciones de los indicadores obtenidas en las pruebas. De esta forma:

$$C_{(ejb)}=\sum_{i=0}^{n} Vi=2+1=3$$

La calificación numérica para la tecnología de programación tradicional en el parámetro uso de hardware, se calcula mediante la sumatoria de las calificaciones de los indicadores obtenidas en las pruebas. De esta forma:

$$C_{(pt)}=\sum_{i=0}^{n} Vi=2+3=5$$

El nivel de cumplimiento de la tecnología EJB expresado en porcentajes es igual a la división de la calificación obtenida por este entre la calificación máxima del parámetro y todo esto multiplicado por el cien por ciento.

$$P_{(ejb)}=\left(\frac{C_{(ejb)}}{C_{max}}\right) X100\% = \frac{3}{8} x100\%=37,5\%$$

De igual forma el nivel de cumplimiento de la programación tradicional expresado en porcentajes es igual a la división de la calificación obtenida por este entre la calificación máxima del parámetro y todo esto multiplicado por el cien por ciento.

$$P_{(pt)}=\left(\frac{C_{(pt)}}{C_{max}}\right) X100\% = \frac{5}{8} x100\%=62,5\%$$

Los resultados del cumplimiento de las tecnologías de programación se expresan en la figura 27.

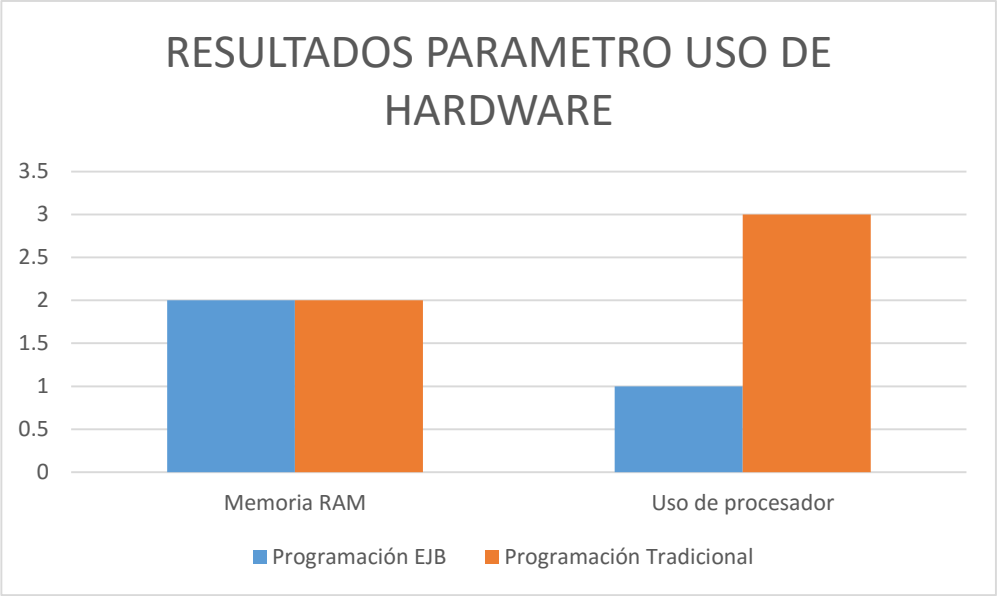


Ilustración 27: Resultados parámetro Uso de Hardware

Fuente: Los Autores

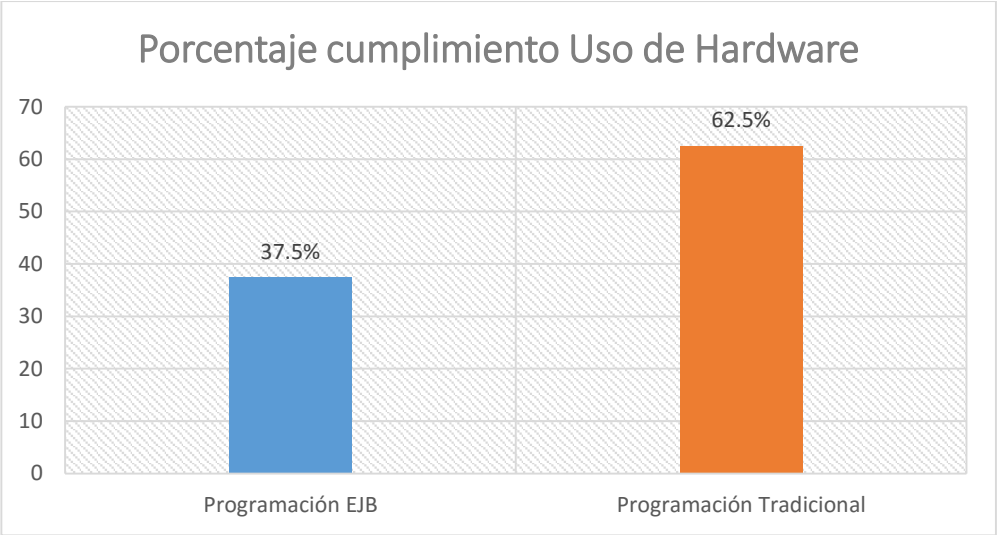


Ilustración 28: Porcentaje cumplimiento Uso de Hardware entre

Fuente: Los Autores

3.9.3.2. Interpretación de resultados

De acuerdo a los niveles de cumplimiento que permite establecer el equivalente cualitativo de acuerdo a porcentajes, se valorara los resultados obtenidos por cada tecnología de desarrollo web.

Los resultados obtenidos en el parámetro uso de hardware revelan que la tecnología EJB posee un nivel de cumplimiento del 37,5% de los indicadores establecidos, lo que equivale a Regular. Mientras que la programación tradicional cumple con el 62,5% de los indicadores establecidos, lo que equivale a Buena.

3.10. Demostración de la Hipótesis

Se realiza en la tabla número 28 un resumen de las ponderaciones para cada indicador, tomando como referencia y en base a la tabla número 3, donde se define el nivel de cumplimiento para las tecnologías de desarrollo a nivel de rendimiento.

Parámetro e Indicador		Tecnología de Programación EJB	Tecnología de Programación Tradicional	Calificación Máxima	%Programación EJB	%Programación Tradicional
Peticiones del usuario	**I1.Número de Peticiones Soportadas**	**4**	**3**	**4**	87,5%	50%
	I2.Tiempo de ejecución de solicitud	**2**	**2**	**4**		
Carga de Usuarios	**I1.Número de usuarios concurrentes**	**3**	**3**	**4**	75%	75%
	I2.Número de usuarios en espera	**3**	**3**	**4**		
Uso de Hardware	I1.Memoria RAM	2	1	4	37,5%	62,5%
	I2.Uso de Procesador	**2**	**3**	**4**		

Tabla 27: Resumen de los indicadores

Fuente: Los Autores

Las calificaciones en porcentajes de cada parámetro de comparación empleado en la presente investigación se observa en la Figura siguiente.

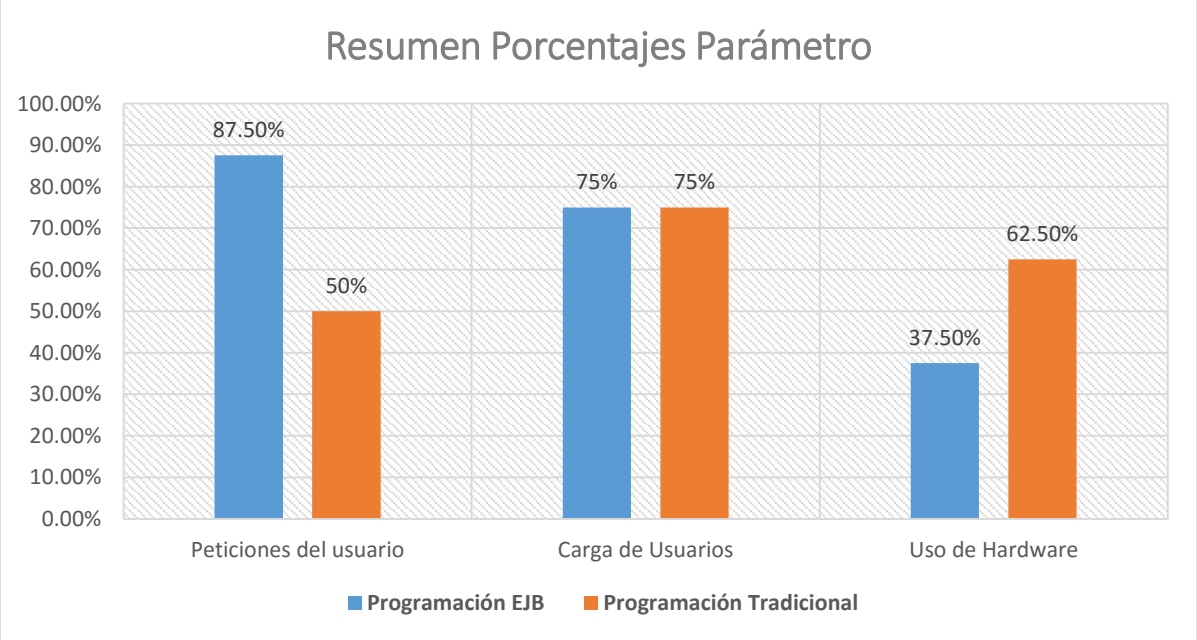


Ilustración 29: Resumen porcentajes parámetros

Fuente: Los Autores

En base a resultados obtenidos y a la Figura número 34 anterior se interpreta:

- La tecnología de programación EJB provee una ventaja en Peticiones del usuario con el 87,5% sobre la programación tradicional que cumple con el 50%.
- La tecnología de programación EJB y la programación tradicional cumplen con similar calificación de acuerdo al Análisis de diferencia significativo que cumplen con el 75%.
- La tecnología de programación EJB requiere una mayor cantidad de uso de hardware cumpliendo con el 37,5% en comparación a la programación tradicional que cumple con el 62,5% de las expectativas en los indicadores asignados.

Considerando los parámetros establecidos en la Tabla Número 3. El análisis se presenta en la Tabla 29 con un resumen de las calificaciones obtenidas por cada tecnología de programación web.

Tecnología	**Peticiones del usuario**	**Carga de usuarios**	**Uso de Hardware**	**Total**	**Porcentaje de Cumplimiento**
Programación EJB	4	3	2	9	75,00%
Programación tradicional	2	3	3	8	66,66%
Calificación Máxima	4	4	4	12	100%

Tabla 28: Resumen de calificaciones de las tecnologías en cada parámetro.

Fuente: Los Autores

Las calificaciones totales de las tecnologías de programación en porcentajes se ilustran en la figura 31.

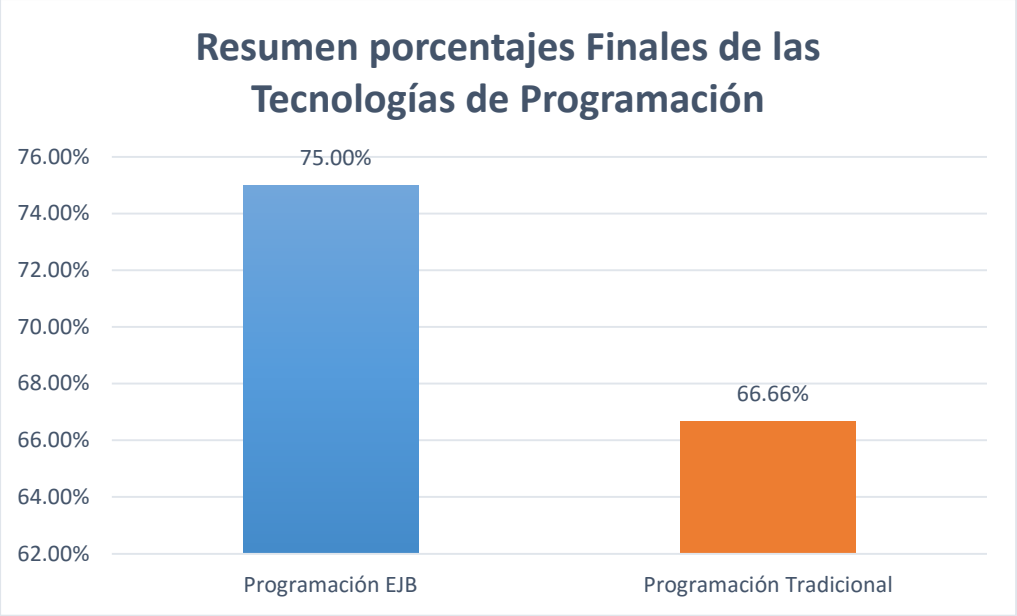


Ilustración 30: Resumen porcentajes Finales de las Tecnologías de Programación

Fuente: Los Autores

3.11. Comprobación de la Hipótesis

En base a los resultados del análisis realizado, su interpretación y haciendo uso de la estadística descriptiva y el análisis de diferencia significativa, se comprueba la hipótesis H1 planteada para la presente tesis, mediante la cual se puede afirmar que la Tecnología Enterprise Java Beans EJB incide en el rendimiento de aplicaciones empresariales cumpliendo con el **75,00%** de los parámetros establecidos para el análisis en comparación al **66,66%** de cumplimiento de la Programación Tradicional. Dando una mejora del **8,34%** en favor de la Tecnología de desarrollo EJB.

CAPITULO IV

IMPLEMENTACIÓN DE GLU - SISTEMA DE GESTIÓN DE LABORATORIOS DE LA UNIVERSIDAD NACIONAL DE CHIMBORAZO.

En el capítulo anterior de la presente investigación se demostró cuál de las tecnologías de desarrollo de Aplicaciones Empresariales ofrece mejores resultados en cuanto a rendimiento, dicha tecnología es Enterprise Java Beans EJB. Por esta razón, la aplicación implementada como caso aplicativo se centrará en la utilización de dicha tecnología para su desarrollo donde se utilizara la metodología RUP de la que se destaca las fases esenciales que contribuyen a la culminación del proyecto.

4.1. Estudio de viabilidad

4.1.1. Antecedentes

La falta de automatización dentro de la gestión de Laboratorios de la Universidad, tanto en equipos, salas de internet y laboratorios en conjunto con los estudiantes para la emisión de un certificado, genera un problema para la Universidad Nacional de Chimborazo, porque no se cuenta con un Sistema de Gestión de Laboratorios que nos permita tener la información actualizada y en línea.

4.1.2. Descripción del problema

En la actualidad, la Universidad Nacional de Chimborazo en cada una de sus unidades académicas dispone de laboratorios, salas multimedia, salas de internet, talleres; para el desarrollo de actividades de apoyo académico, las unidades académicas conforme a los perfiles de las carreras proveen de las infraestructuras, recursos e insumos para su respectiva investigación, los procesos de gestión y de control de los servicios y usos de los laboratorios se los realiza en forma manual, no existe un estándar institucional, razón por la cual cada funcionario organiza la información conforme a sus necesidades y basadas en la experiencia en su puesto de trabajo. En la cual se aplicara el análisis de los EJB aplicado al desarrollo de un Sistema de Gestión Y Control De Laboratorios en la Universidad Nacional de Chimborazo.

La web constituye un medio fundamental para que la gestión de ingresos, egresos y préstamos pueda permanecer en constante actualización del uso de laboratorios y equipos de los mismos.

4.1.3. Requerimientos del sistema

Previo al desarrollo del Sistema se estableció los siguientes requerimientos básicos con lo que deberá contar el Sistema de Gestión de Laboratorios de la Universidad Nacional de Chimborazo.

- Automatización de la información de los Laboratorios de la UNACH.
- Ingresos de equipos a los laboratorios
- Egresos de equipos de los laboratorios
- Prestamos de los equipos por parte de los estudiantes
- Devoluciones de los equipos a los laboratorios por parte de los estudiantes
- Certificado de no adeudar en la Institución
- Usuarios del sistema por roles.
- Permitir la obtención del Certificado de no adeudar.

De acuerdo a los requerimientos expuestos, el sistema se delimitó en Gestionar los Laboratorios de carácter académico que contará básicamente con los siguientes módulos establecidos:

- MÓDULO DE ADMINISTRACIÓN DE USUARIOS Y SEGURIDAD
  Este módulo permitirá definir usuarios, roles y permisos para los administradores y usuarios del sistema. Brindando la posibilidad de establecer diferentes niveles de acceso a la información contenida en el Sistema.

- MÓDULO DE INGRESOS
  Este módulo permitirá definir con detalle el ingreso de equipos a los laboratorios de la UNACH.

- MÓDULO DE EGRESOS
  Este módulo permitirá definir con detalle el egreso (dar de baja) de equipos de los laboratorios.

- MÓDULO DE PRESTAMOS
  Este módulo permitirá registrar el préstamo de equipos y laboratorios.

- MÓDULO DE REPORTES

Permitirá la visualización de información detallada para el Estudiante y para el Administrador, además de la impresión del respectivo Certificado.

A continuación se presenta el plan tentativo y su duración en semanas que se llevará a cabo para el Desarrollo del Sistema de Gestión de Laboratorios.

4.1.4. Plan de desarrollo

Se planteó la duración en semanas por cada módulo a desarrollar, con su respectiva iteración del Sistema de Gestión de equipos de laboratorio de la Universidad Nacional de Chimborazo.

Equivalencia según el tiempo de duración: 1 Alta, 2 Media, 3 Baja.

HISTORIA DE USUARIO	DURACIÓN EN SEMANAS	ITERACION
1)Módulo de administración de usuarios y seguridad	2	3
2)Módulo de Ingresos	3	2
3)Módulo de Egresos	4	1
4)Módulo de prestamos	4	1
5)Módulo de reportes	3	2

Tabla 29: Pla de Desarrollo

Fuente: Los Autores

4.2. Análisis

4.2.1. Planificación del proyecto

Esta planificación del proyecto se realizó después del estudio del problema y los requerimientos, mediante la representación de las historias se efectuó la planificación inicial la cual fue variando en el transcurso de la misma cambiando y mejorando las historias en base a concepción del problema.

4.2.2. Historias de Usuarios

Las historias de usuarios tiene como propósito ver las necesidades del sistema por lo tanto se realizarán descripciones cortas y escritas en el lenguaje del usuario sin terminología,

detallando el tiempo que conllevara la implementación, así como la estimación del riesgo de dicha historia de usuario.

Cada historia de usuario se fracciona en actividades planificables y medibles para su realización. La realización de este plan debe tener muy en cuenta la prioridad, riesgo, esfuerzo e iteración de los usuarios para satisfacerles en mayor medida posible. Resaltamos el riesgo por la seguridad en el Módulo de administración de usuarios y seguridad en la tabla número 31.

Nº	NOMBRE	PRIORIDAD	RIESGO	ESFUERZO	ITERACION
1	Módulo de administración de usuarios y seguridad	Baja	Alto	Bajo	3
2	Módulo de Ingresos	Media	Medio	Medio	2
3	Módulo de Egresos	Alta	Alto	Alto	1
4	Módulo de préstamos	Alta	Alto	Alto	1
5	Módulo de reportes	Media	Medio	Medio	2

Tabla 30: Tabla de iteraciones

Fuente: Los Autores

HISTORIA DE USUARIO	
Número: 1	**Nombre:** Módulo de administración de usuarios y seguridad.
Usuario: Súper Administrador	
Modificación de historia Número: 01	**Iteración: 3**
Programador responsable/s: Richar Buenaño y Angel Moyón	
Descripción: Permite acceder a la aplicación una vez registrado correctamente y poder asignar los permisos según su rol (Administrador de Laboratorio y Estudiante).	
Observaciones: En la administración de usuarios podemos ver un listado de todos los que forman parte del sistema Gestión de Laboratorios UNACH.	

Tabla 31: Historia Gestión de Administración de Roles y Usuarios

Fuente: Los Autores

HISTORIA DE USUARIO	
Número: 2	**Nombre: Gestión de Ingresos**
Usuario: Administrador y Estudiante	
Modificación de historia Número: 01	**Iteración: 2**
Programador responsable/s: Richar Buenaño y Angel Moyón	
Descripción: Permite el ingreso de información, de los equipos a los laboratorios.	
Observaciones: Los equipos provienen de una Bodega. Y se registran según el motivo de ingreso.	

Tabla 32: Historia Gestión de Ingresos

Fuente: Los Autores

HISTORIA DE USUARIO	
Número: 3	**Nombre: Gestión de Egresos**
Usuario: Administrador y Estudiante	
Modificación de historia Número: 01	**Iteración: 1**
Programador responsable/s: Richar Buenaño y Angel Moyón	
Descripción: Permite registrar información de los equipos que se dan de baja por parte de los Laboratorios.	
Observaciones: Los equipos son egresados (dar de baja).	

Tabla 33: Gestión de Egresos

Fuente: Los Autores

HISTORIA DE USUARIO	
Número: 4	**Nombre: Gestión de Prestamos**
Usuario: Administrador y Estudiante	
Modificación de historia Número: 01	**Iteración: 1**
Programador responsable/s: Richar Buenaño y Angel Moyón	
Descripción: El presente módulo registra el préstamo y devolución de equipos que realiza el estudiante al Administrador del Laboratorio.	

Observaciones: Mantienen un historial por el lado de ingresos y egresos de los préstamos y devoluciones.

Tabla 34: Historia Gestión de Préstamos

Fuente: Los Autores

HISTORIA DE USUARIO	
Número: 5	**Nombre: Módulo de Reportes**
Usuario: Administrador y Estudiante	
Modificación de historia Número: 01	**Iteración: 2**
Programador responsable/s: Richar Buenaño y Angel Moyón	
Descripción: El presente módulo muestra el historial de los préstamos, devoluciones, ingresos y egresos de equipos.	
Observaciones: El certificado de NO ADEUDAR se emite en este módulo.	

Tabla 35: Historia Gestión de Reportes

Fuente: Los Autores

4.2.4. Integrantes y roles

Con la colaboración del Director del proyecto, los miembros, los usuarios y desarrolladores, se constituirá el equipo delegado de la implementación del sistema web.
Esto involucrara que los diseños deberán ser sencillos y claros, los usuarios dispondrán de versiones de prueba del software para que puedan participar en el proceso de desarrollo mediante sugerencias y aportaciones, dicho equipo de trabajo se ve ilustrado en la Tabla 37 siguiente definiendo Integrantes y Roles.

Miembro	Grupo	Roles	Metodología
Richar Buenaño	Tesista	Rastreador, Testeador, Programador	RUP
Angel Moyón	Tesista	Rastreador, Testeador, Programador	
Ing. Diego Palacios		Consultor	

Tabla 36: Integrantes y Roles

Fuente: Los Autores

4.2.5. Prototipos de la aplicación

Para el usuario deben ser, muy importantes las interfaces de usuario ya que de esto dependerá el entendimiento claro y rápido por parte del usuario al comenzar a interactuar con el sistema. Se pretende que la interfaz del usuario sea amigable, sencilla y funcional con un alto grado de perspicacia, por tal razón se desarrollaron los prototipos generales del sistema. A continuación, se realizará una breve descripción del proceso primordial.

- En la Ilustración número 36 se muestra el prototipo de inicio de sesión de los usuarios.

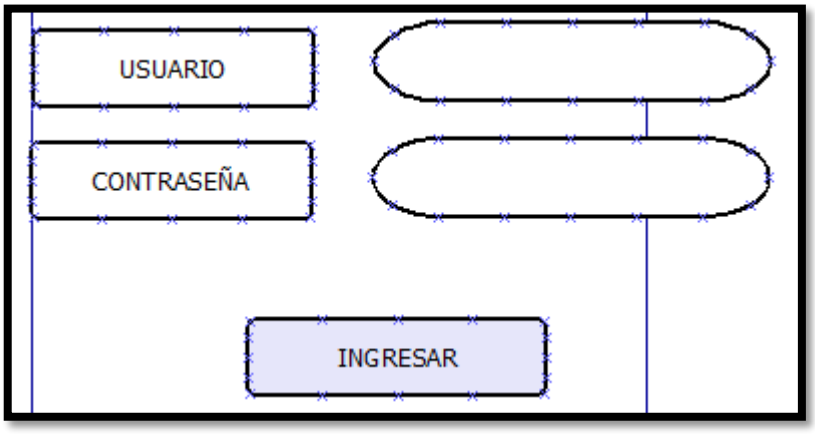


Ilustración 31: Prototipo Login

Fuente: Los Autores

- En la figura 37 se muestra la página principal del sistema

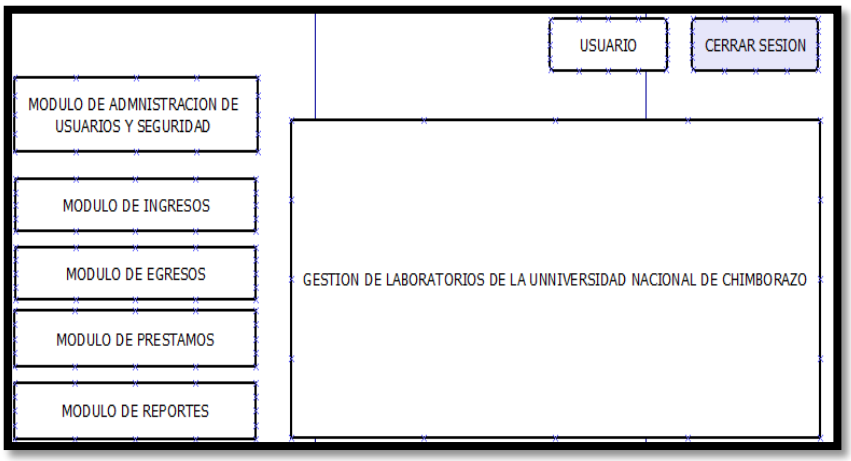


Ilustración 32: Prototipo Página Principal

Fuente: Los Autores

- En la figura 38 se muestra el módulo de administración de usuarios y seguridad.

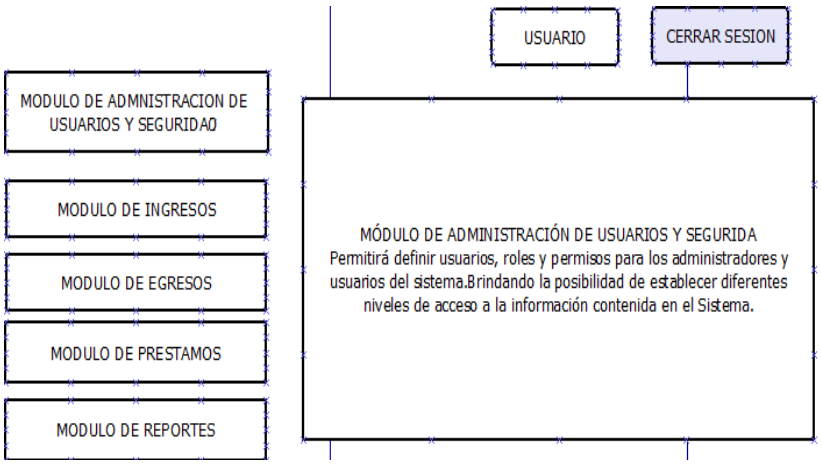


Ilustración 33: Módulo de administración de usuarios y seguridad.

Fuente: Los Autores

- En la figura 39 se muestra el módulo de ingresos.

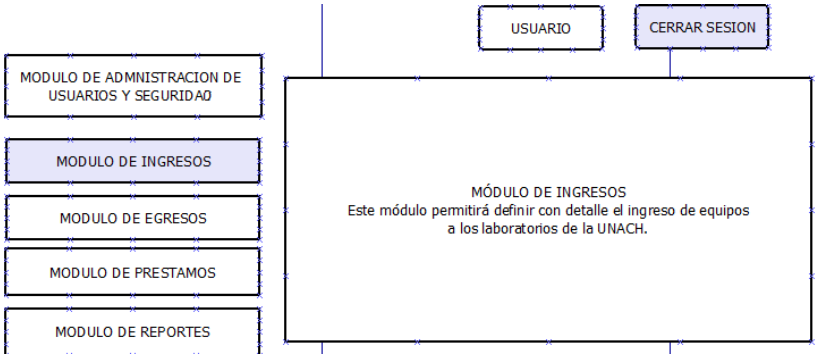


Ilustración 34: Prototipo del Módulo de Ingresos

Fuente: Los Autores

- En la figura 40 se muestra el módulo de egresos.

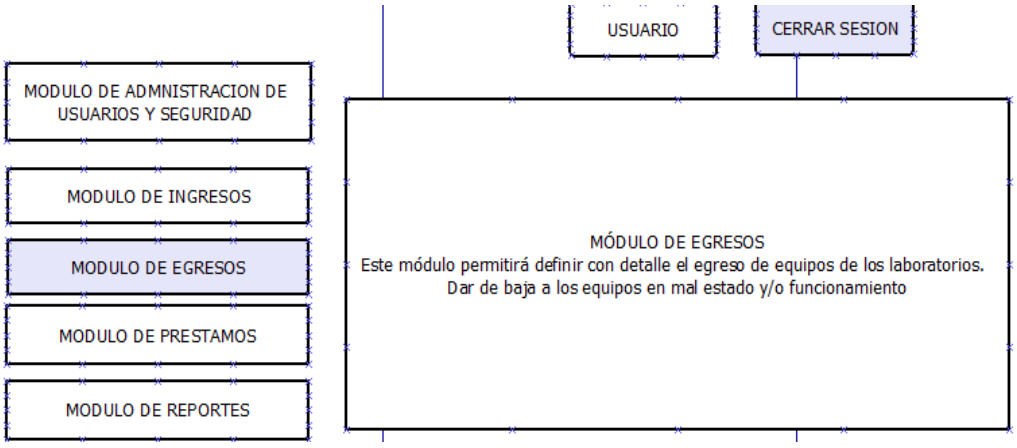


Ilustración 35: Prototipo del Módulo de Egresos

Fuente: Los Autores

- En la figura 41 se muestra el módulo de préstamos.

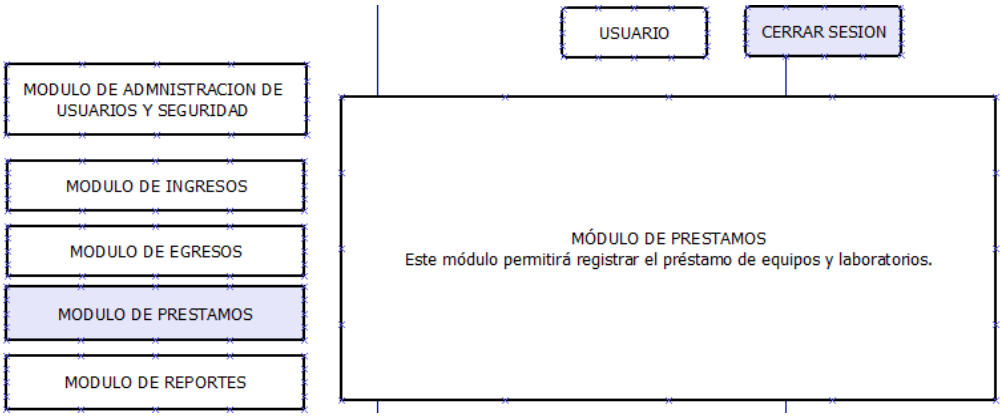


Ilustración 36: Prototipo del Módulo de Préstamos

Fuente: Los Autores

- En la figura 42 se muestra el módulo de Reportes.

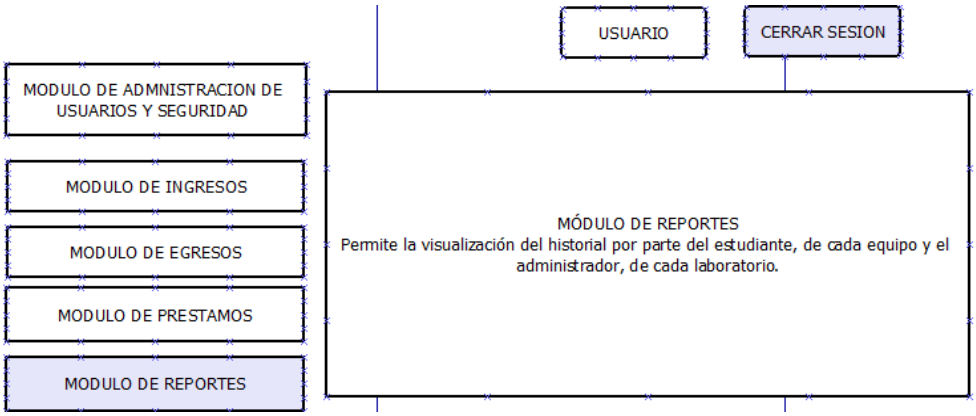


Ilustración 37: Prototipo del Módulo de Reportes

Fuente: Los Autores

4.2.6. Flujo de trabajo

Las actividades del sistema fueron divididas en varios procesos que serán reflejados mediante flujos de trabajo.

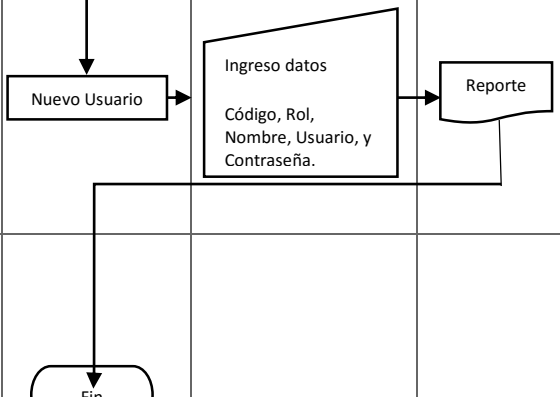
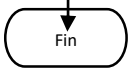
PROCESO NUEVO USUARIO						
Proceso	**Actividad**	**Flujograma**	**IN**	**OUT**	**Responsable**	**Observación**
	Inicio					
1	Actividad				SuperAdministrador	
	Fin					

Tabla 37: Definición del proceso de nuevo usuario

Fuente: Los Autores

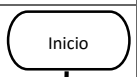
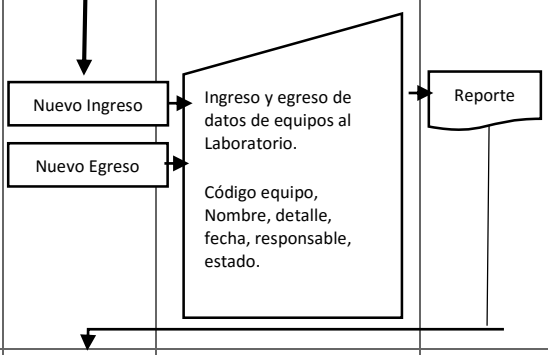
PROCESO GESTIÓN DE LABORATORIO INGRESOS Y EGRESOS EQUIPOS						
Proceso	**Actividad**	**Flujograma**	**IN**	**OUT**	**Responsable**	**Observación**
	Inicio					
1	Actividad				Administradores de Laboratorios	
2	Fin					

Tabla 38: Proceso de gestión de Ingresos y Egresos de Equipos a los Laboratorios

Fuente: Los Autores

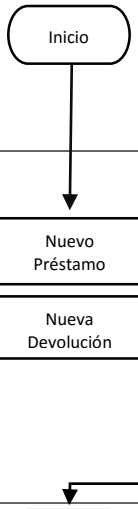
PROCESO GESTIÓN DE PRESTAMOS Y DEVOLUCIONES DE EQUIPOS						
	Actividad	**Flujograma**	**IN**	**OUT**	**Responsable**	**Observación**
	Inicio					
1	Actividad				Administrador y Estudiante	
	Fin					

Tabla 39: Proceso de gestión Préstamos y devolución de Equipos o los Laboratorios

Fuente: Los Autores

4.2.7. Herramientas de desarrollo.

Para la presente implementación del Sistema de Gestión de Laboratorios se utilizará las siguientes tecnologías y herramientas.

HERRAMIENTA	CONCEPTO	VERSIÓN UTILIZADA
pgAdmin 	Es un sistema de gestión de bases de datos relacional	Versión 9.3
NETBEANS 	Entorno de desarrollo integrado Java	Netbeans IDE 8.0.2

COMPONENTE PRIMEFACES 	Componentes visuales para JSF	PrimeFaces 5.0
SERVIDOR GLASSFISH 	Servidor de aplicaciones que implementa la plataforma JEE5.	GlassFish 4.0

Tabla 40: Herramientas utilizadas en el desarrollo

Fuente: Los Autores

4.3. Diseño

4.3.1. Listado de requerimientos del sistema y del usuario

REQUISITOS	SISTEMA DE GESTIÓN DE LABORATORIOS UNACH
REQUISITOS DEL USUARIO	• Usuarios finales del sistema (estudiantes). • Administradores de Laboratorios • Diseñadores del sistema • La aplicación permitirá realizar préstamos y devoluciones de equipos, llevando una gestión a nivel de laboratorios y componentes. • La aplicación permitirá emitir comprobantes de ingresos, egresos y de no adeudar ningún equipo en los Laboratorios de la UNACH. • El diseño de la interfaz debe ser similar en un porcentaje a la plataforma de la institución. • Controlar los reportes básicos • Detectar las fallas y sus causas, así como las acciones correctivas en el momento de su desarrollo.
REQUISITOS DEL SISTEMA	• El estudiante tendrá que registrarse en la aplicación con sus datos personales más el código estudiantil. • Accederá a una sesión con un perfil estudiante, de acuerdo a su carrera y facultad se desplegará los laboratorios en los que podrá realizar sus préstamos. • Ingresará al menú préstamos, entonces seleccionará los equipos que desea pedir para sus estudios. • Introducirá todos los datos y seleccionará realizar préstamo.

	• El sistema almacenará los datos del nuevo préstamo. • Usuarios finales del sistema (estudiantes). • Administradores de Laboratorios • Desarrolladores de software. • La base de datos será en postgres. • El software debe ser escrito en una tecnología que nos permita integrar con otras. • El sistema debe permitir al rol secretaría consultar acerca de los préstamos y devoluciones de los equipos, para la comprobación del certificado.

Tabla 41: Listado de Requerimientos

Fuente: Los Autores

4.3.2. Diagrama de clases

En el desarrollo del proyecto se utilizó la herramienta de mapeo con Netbeans en el que podemos realizar un mapeo de la base de datos con las clases del sistema, de esta manera se aplica el modelo de Persistencia con EJB.

Al realizar este mapeo, da como resultado que cada tabla de la base de datos se convierte en una clase, con su respectivo controlador, de esta manera el mismo diagrama de base de datos se convierte en el diagrama de clases, cada clase es un objeto de la base de datos, a continuación se muestra cada clase con sus funcionalidades.

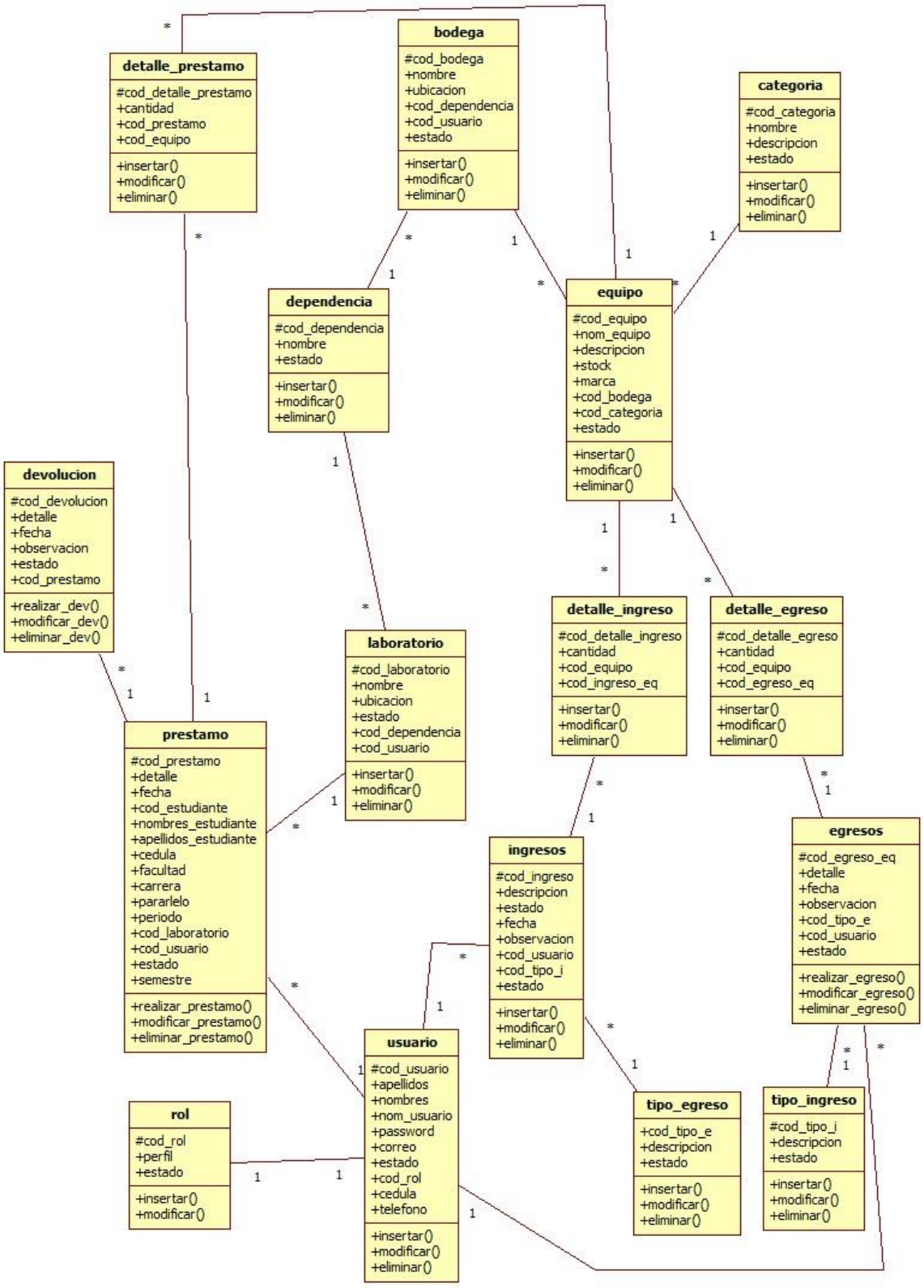


Ilustración 38: Diagrama de clases

Fuente: Los Autores

4.3.2. Diagrama de casos de uso

- **A continuación se presenta el diagrama de Ingresos de equipos**

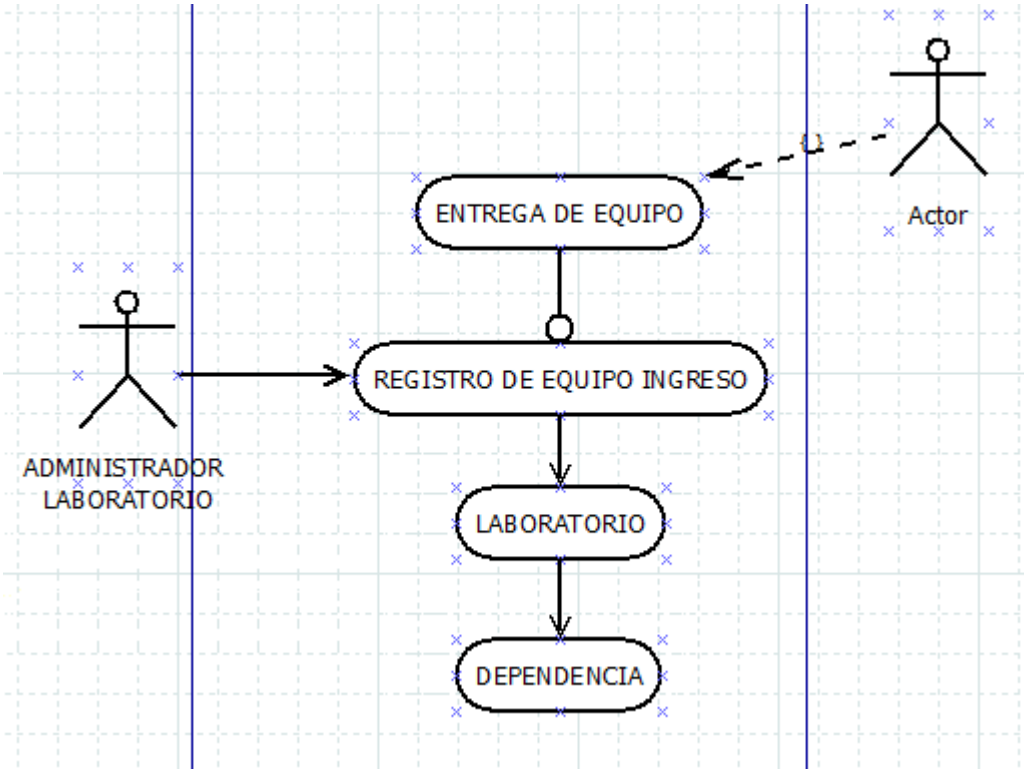


Ilustración 39: Caso de Uso Ingreso de equipos

Fuente: Los Autores

- **A continuación se presenta el diagrama de Egresos de equipos**

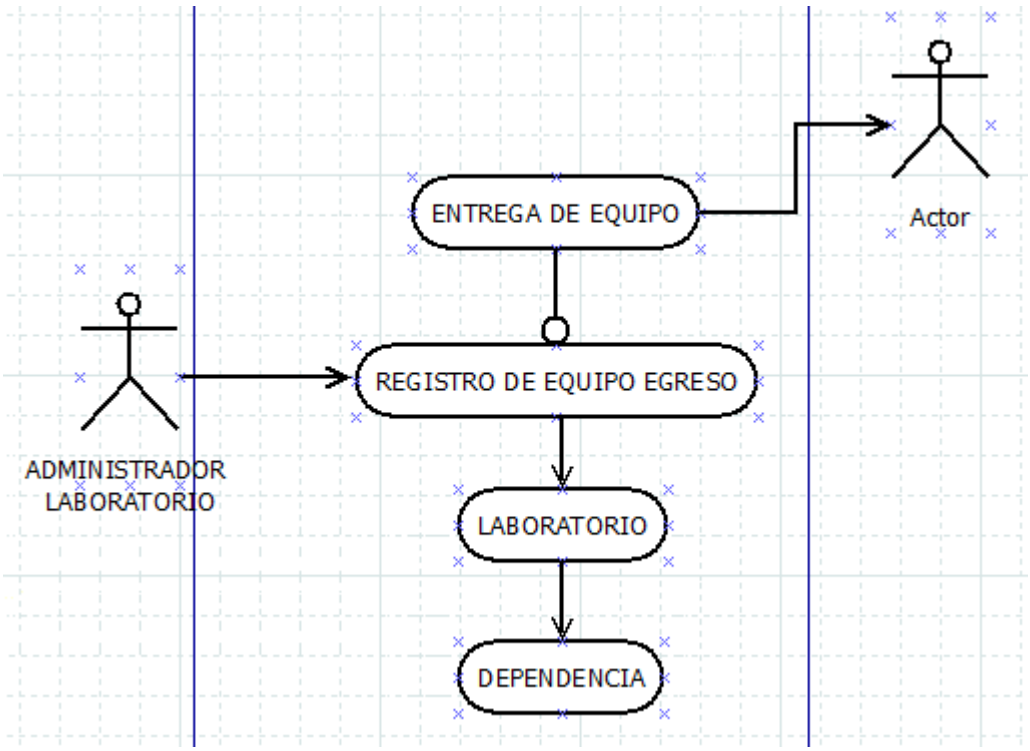


Ilustración 40: Caso de Uso Egreso de equipos

Fuente: Los Autores

- **Caso de uso préstamos de equipos**

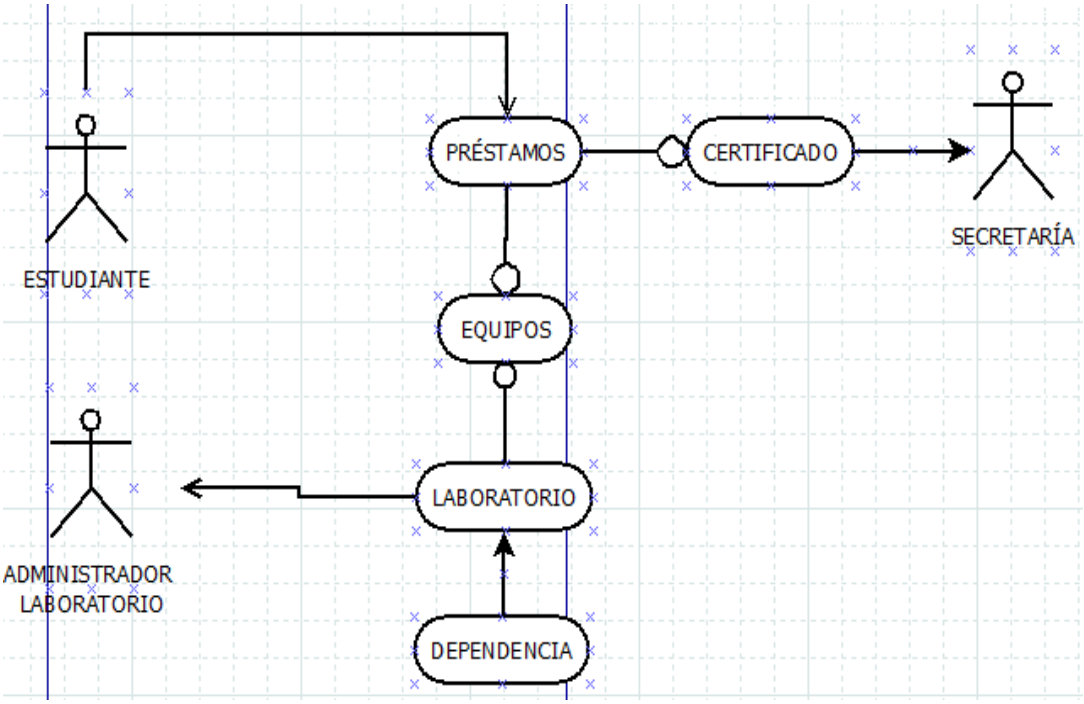


Ilustración 41: Caso de Uso Préstamos

Fuente: Los Autores

- **Caso de Uso devolución de equipos**

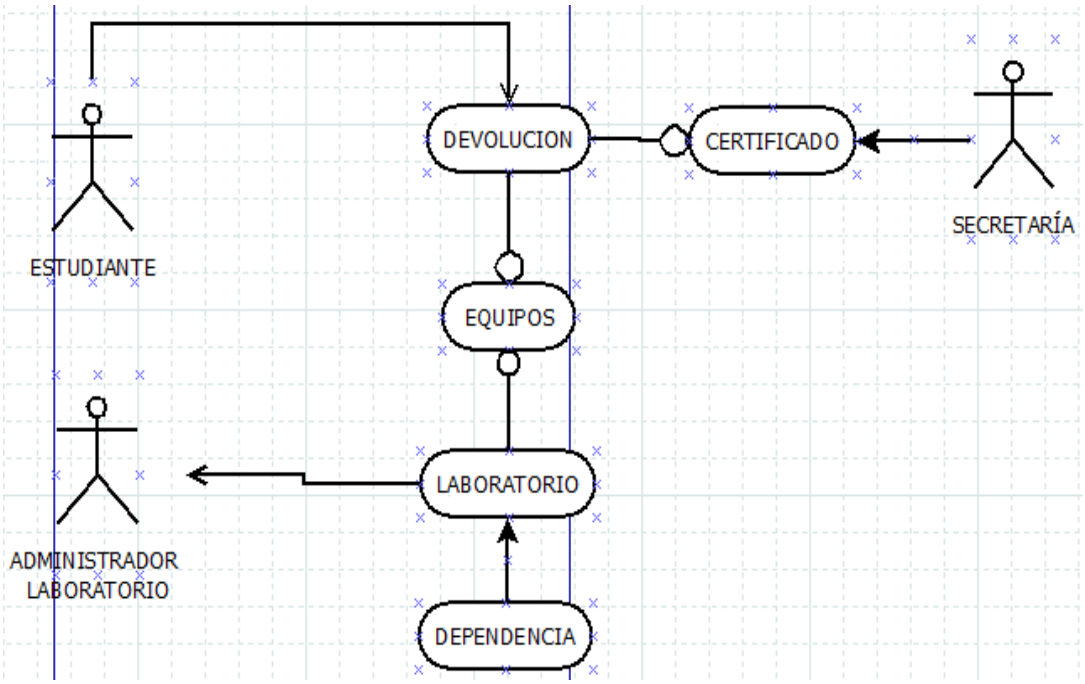


Ilustración 42: Caso de Uso Devolución

Fuente: Los Autores

- **Caso de uso para el usuario (ACCESO AL SISTEMA)**

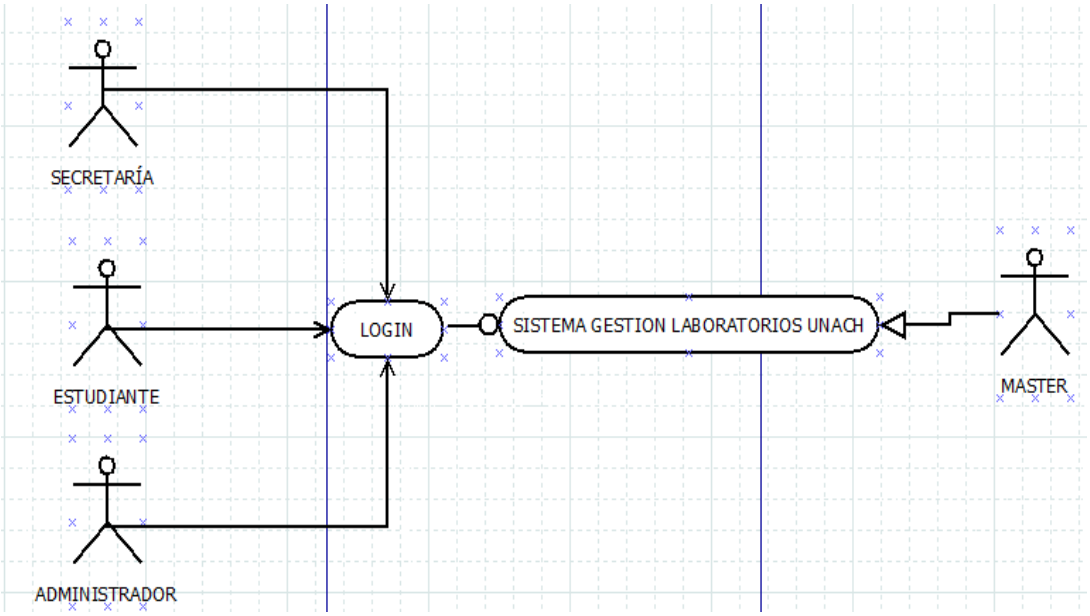


Ilustración 43: Caso de Uso usuarios

Fuente: Los Autores

4.3.2. Arquitectura y módulos de la aplicación

Arquitectura de la aplicación

En la siguiente imagen se muestra la arquitectura de la aplicación.

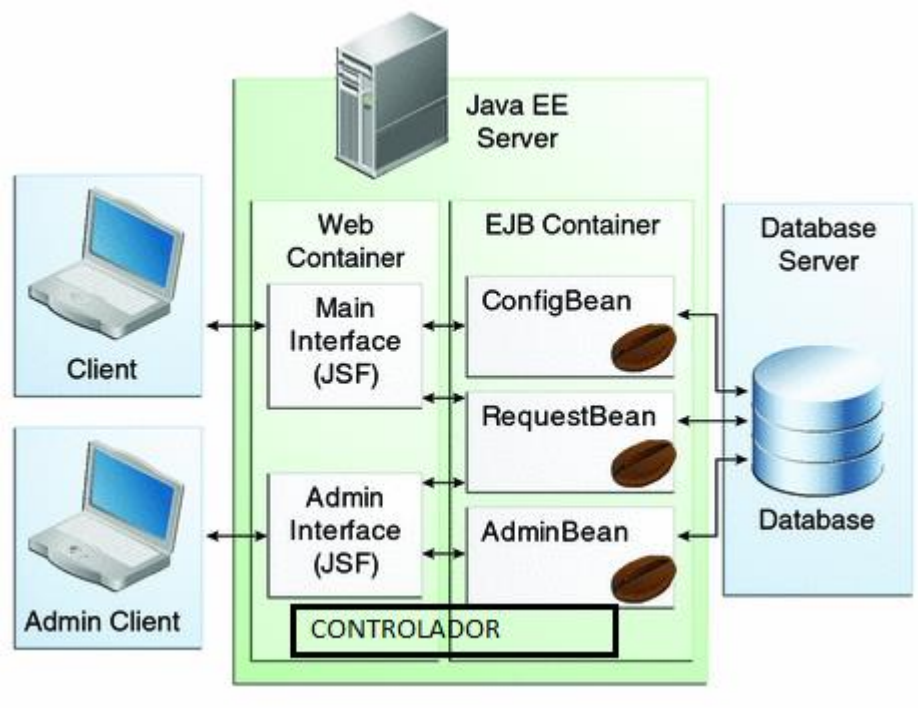


Ilustración 44: Arquitectura de la aplicación

Fuente: Los Autores

El archivo de biblioteca JAR es útil para permitir que las clases de entidad y clases de ayuda para ser reutilizados por otras aplicaciones, como por ejemplo una aplicación cliente Java.

El Sistema de Gestión de Laboratorios UNACH utiliza las siguientes características de Java EE de plataforma:

- Entidades Java Persistence API
    - API Java para JavaBeans Validación (Bean Validation) anotaciones en las entidades de verificación de datos
- Beans de la aplicación
    - Local, sin interfaz-vista de sesión y Singleton Beans
- La tecnología JavaServer Faces, utilizando Facelets para la interfaz web

- Plantillas
- Componentes de material compuesto
- Restricciones de seguridad en la interfaz administrativa
- Convertidores personalizados para las clases de entidad utilizados en los componentes de interfaz de usuario

El Sistema de Gestión de Laboratorios UNACH dispone de cuatro interfaces de usuario principales, tanto envasados dentro de un único archivo WAR:

- La interfaz principal, para estudiantes, administradores, Súper Administrador
- La interfaz administrativa utilizada por los administradores de los laboratorios para gestionar estudiantes con sus préstamos y devoluciones.

Módulos de la aplicación

La aplicación Sistema de Gestión de Laboratorios Unach, está compuesta por los siguientes módulos que se menciona a continuación en la Imagen 44.

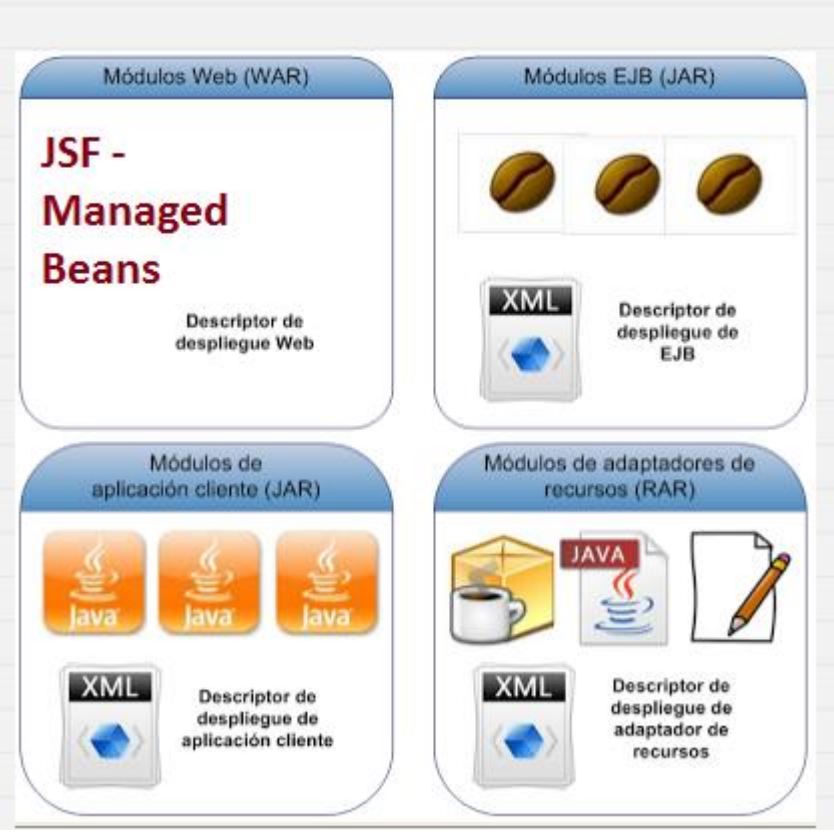


Ilustración 45: Arquitectura de la aplicación (Archivos)

Fuente: Los Autores

- Módulos Web que contiene Java Server Faces JSF- Managed Beans , imágenes, etc. y además un descriptor de despliegue Web.xml empaquetados en archivos WAR
- Módulos EJB, que contienen las clases de los beans empresariales y un descriptor de despliegue de EJB. Son empaquetados en archivos JAR.
- Módulos de aplicación cliente. Contienen clases que van a desplegarse en el cliente y un descriptor de despliegue de aplicación cliente. Son empaquetados en archivos JAR.
- Módulos de adaptadores de recursos, los cuáles contienen interfaces Java, librerías nativas, etc. junto al descriptor de despliegue de adaptador de recursos. Son empaquetados en archivos JAR pero con extensión .rar (resource adapter archive).

4.3.2. Diseño conceptual

A continuación, en la presente ilustración se muestra cómo se utiliza los servicios web para el sistema de Gestión de Laboratorios UNACH, con el usuario estratégico (PC Customer).

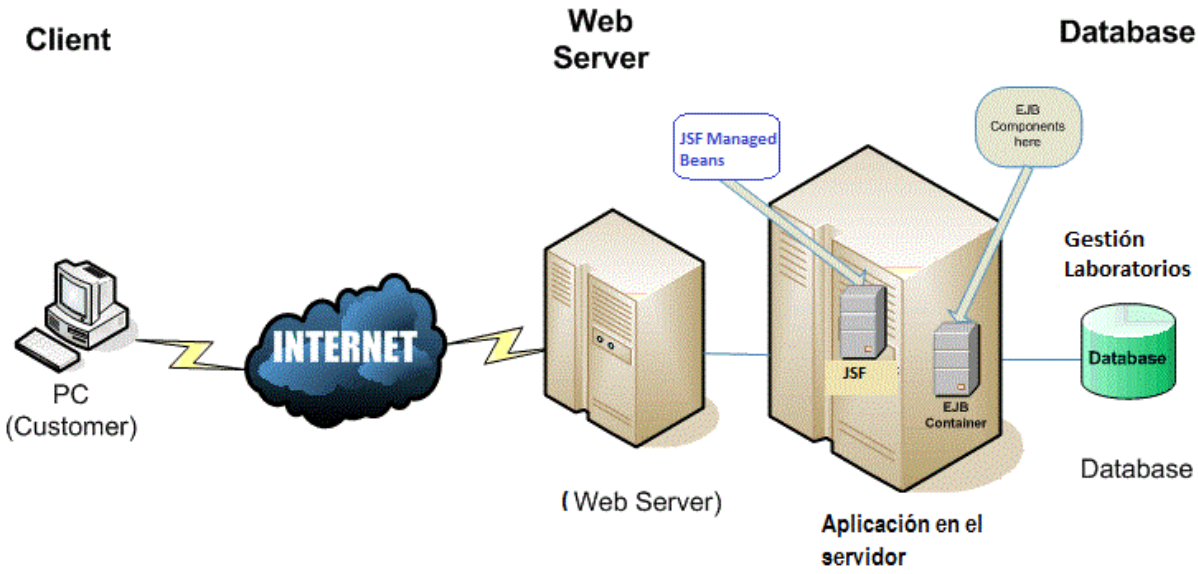


Ilustración 46: Diseño Conceptual

Fuente: Los Autores

4.3.1. Base de datos

La Base de datos se realizó en Postgres, está conformado por 16 tablas para la realización de los módulos del Sistema de Gestión de Laboratorios UNACH.

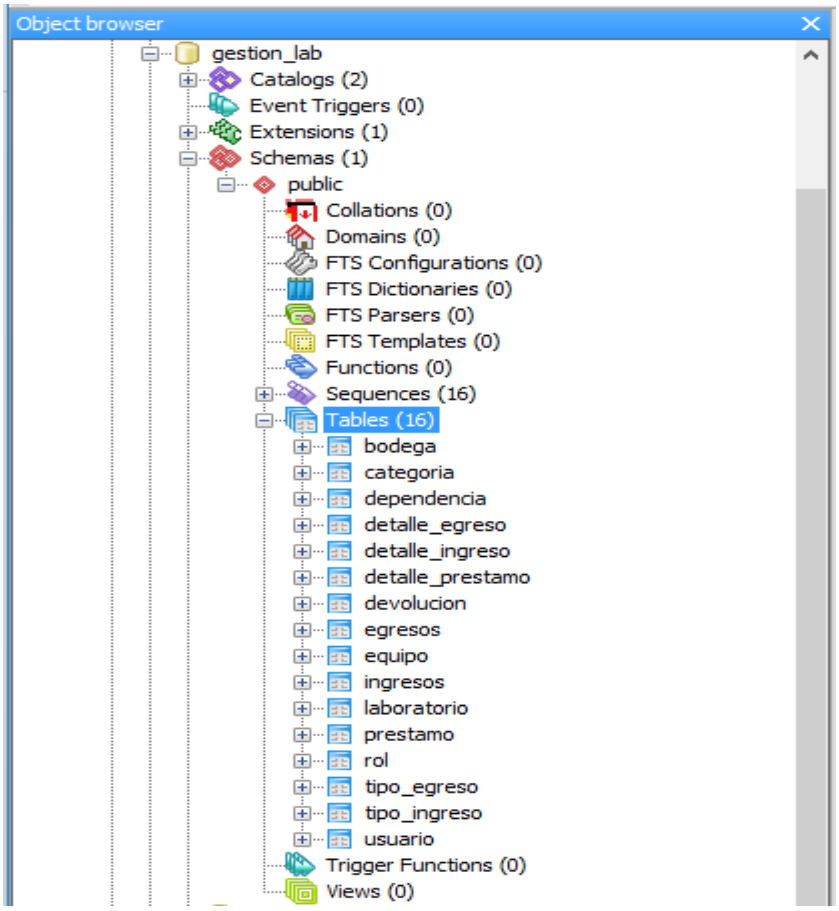


Ilustración 47: Tablas que componen la base de datos gestión_lab

Fuente: Los Autores

4.3.2. Modelo relacional de la Base de datos

El presente modelo relacional está realizado en la Herramienta DIA.

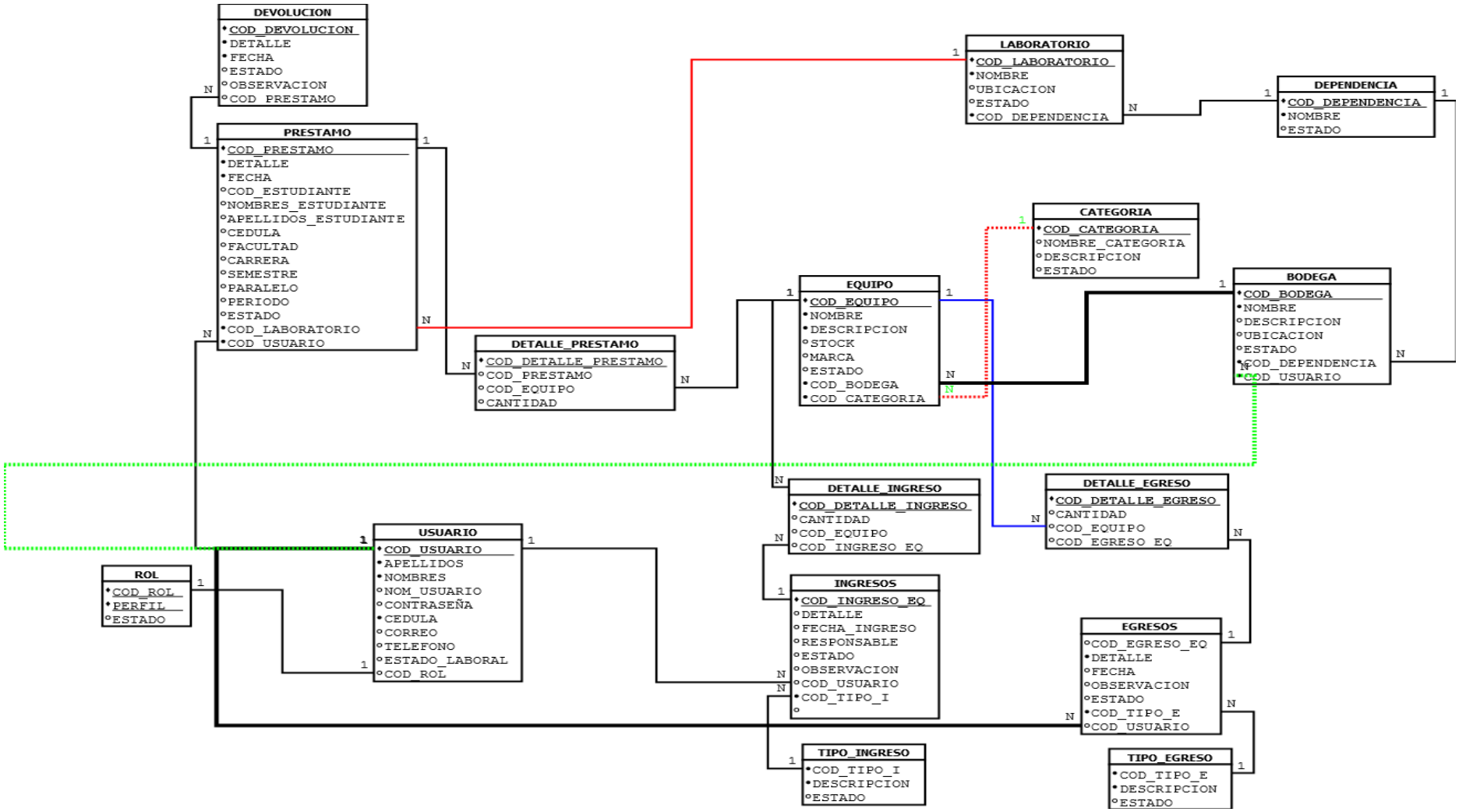


Ilustración 48: Modelo relacional de la base de datos gestión_lab.

Fuente: Los Autores

4.3.3. Diccionario de datos

El Diccionario de datos permite guardar la estructura de la base de datos, es decir se define como se almacena y accede a la información.

- bodega: En esta tabla se almacena los datos de las bodegas de donde provienen los equipos.

NOMBRE DE LA COLUMNA	TIPO DE DATO	CLAVE PRIMARIA	VALORES NULOS	AUTO INCREMENTAL
cod_bodega	Serial	SI	NO	SI
Nombre	Varchar	NO	NO	NO
Descripción	Varchar	NO	NO	NO
Ubicación	Varchar	NO	NO	NO
Estado	Varchar	NO	NO	NO
cod_dependencia	Integer	NO	NO	NO
cod_usuario	Integer	NO	NO	NO

Tabla 42: Descripción de la tabla bodega

Fuente: Los Autores

- categoria: En esta tabla se almacena los datos de las categorías de los equipos.

NOMBRE DE LA COLUMNA	TIPO DE DATO	CLAVE PRIMARIA	VALORES NULOS	AUTO INCREMENTAL
cod_categoria	Serial	SI	NO	SI
nombre_categoria	Varchar	NO	NO	NO
descripción	Varchar	NO	NO	NO
Estado	Varchar	NO	NO	NO

Tabla 43: Descripción de la tabla categoria

Fuente: Los Autores

- dependencia: En esta tabla se almacena el nombre de las unidades académicas y de estudio.

NOMBRE DE LA COLUMNA	TIPO DE DATO	CLAVE PRIMARIA	VALORES NULOS	AUTO INCREMENTAL
cod_dependencia	serial	SI	NO	SI
Nombre	Varchar	NO	NO	NO

Estado	Varchar	NO	NO	NO

Tabla 44: Descripción de la tabla dependencia

Fuente: Los Autores

- detalle_egreso: Esta tabla intermedia entre equipo y egresos almacena la cantidad y las claves foráneas de las tablas antes mencionadas.

NOMBRE DE LA COLUMNA	TIPO DE DATO	CLAVE PRIMARIA	VALORES NULOS	AUTO INCREMENTAL
cod_detalle_egreso	Serial	SI	NO	SI
Cantidad	Integer	NO	NO	NO
cod_equipo	Varchar	NO	NO	NO
cod_egreso_eq	Varchar	NO	NO	NO

Tabla 45: Descripción de la tabla detalle_egreso

Fuente: Los Autores

- detalle_ingreso: Esta tabla intermedia entre equipo e ingresos almacena la cantidad y las claves foráneas de las tablas antes mencionadas.

NOMBRE DE LA COLUMNA	TIPO DE DATO	CLAVE PRIMARIA	VALORES NULOS	AUTO INCREMENTAL
cod_detalle_ingreso	Serial	SI	NO	SI
Cantidad	Integer	NO	NO	NO
cod_equipo	Integer	NO	NO	NO
cod_ingreso_eq	Integer	NO	NO	NO

Tabla 46: Descripción de la tabla detalle_ingreso

Fuente: Los Autores

- detalle_prestamo: Esta tabla intermedia entre equipo y prestamo almacena la cantidad y las claves foráneas de las tablas antes mencionadas.

NOMBRE DE LA COLUMNA	TIPO DE DATO	CLAVE PRIMARIA	VALORES NULOS	AUTO INCREMENTAL
cod_detalle_prestamo	Serial	SI	NO	SI
cod_prestamo	Integer	NO	NO	NO

cod_equipo	Integer	NO	NO	NO
Cantidad	Integer	NO	NO	NO

Tabla 47: Descripción de la tabla detalle_prestamo

Fuente: Los Autores

- devolucion: En esta tabla almacenamos datos de los equipos que son devueltos por parte de los estudiantes.

NOMBRE DE LA COLUMNA	TIPO DE DATO	CLAVE PRIMARIA	VALORES NULOS	AUTO INCREMENTAL
cod_devolucion	Serial	SI	NO	SI
Detalle	Varchar	NO	NO	NO
Fecha	timestamp	NO	NO	NO
Estado	Varchar	NO	NO	NO
observacion	Varchar	NO	NO	NO
cod_prestamo	Integer	NO	NO	NO

Tabla 48: Descripción de la tabla devolución

Fuente: Los Autores

- egresos: En esta tabla almacenamos información de los equipos dados de baja por parte de los Administradores de los Laboratorios.

NOMBRE DE LA COLUMNA	TIPO DE DATO	CLAVE PRIMARIA	VALORES NULOS	AUTO INCREMENTAL
cod_egreso_eq	Serial	SI	NO	SI
Detalle	Varchar	NO	NO	NO
Fecha	Timestamp	NO	NO	NO
Observacion	Varchar	NO	NO	NO
Estado	Varchar	NO	NO	NO
cod_tipo_e	Integer	NO	NO	NO
cod_usuario	Integer	NO	NO	NO

Tabla 49: Descripción de la tabla egresos

Fuente: Los Autores

- equipo: En esta tabla almacenamos información de los equipos.

NOMBRE DE LA COLUMNA	TIPO DE DATO	CLAVE PRIMARIA	VALORES NULOS	AUTO INCREMENTAL
cod_equipo	Serial	SI	NO	SI
Nombre	Varchar	NO	NO	NO
descripción	Varchar	NO	NO	NO
observacion	Varchar	NO	NO	NO
Stock	Integer	NO	NO	NO
Marca	Varchar	NO	NO	NO
Estado	Varchar	NO	NO	NO
cod_bodega	Integer	NO	NO	NO
cod_categoria	Integer	NO	NO	NO

Tabla 50: Descripción de la tabla equipo

Fuente: Los Autores

- ingresos: Almacenamos información de los ingresos por parte de los equipos a cada Laboratorio.

NOMBRE DE LA COLUMNA	TIPO DE DATO	CLAVE PRIMARIA	VALORES NULOS	AUTO INCREMENTAL
cod_ingreso_eq	Serial	SI	NO	SI
Detalle	Varchar	NO	NO	NO
Fecha	timestamp	NO	NO	NO
Observacion	Varchar	NO	NO	NO
Estado	Varchar	NO	NO	NO
cod_tipo_i	Integer	NO	NO	NO
cod_usuario	Integer	NO	NO	NO

Tabla 51: Descripción de la tabla ingresos

Fuente: Los Autores

- laboratorio: Es la tabla que permite almacenar información de los laboratorios.

NOMBRE DE LA COLUMNA	TIPO DE DATO	CLAVE PRIMARIA	VALORES NULOS	AUTO INCREMENTAL
cod_laboratorio	Serial	SI	NO	SI
Nombre	Varchar	NO	NO	NO
Ubicación	Varchar	NO	NO	NO
Estado	Varchar	NO	NO	NO

cod_dependencia	Integer	NO	NO	NO

Tabla 52: Descripción de la tabla laboratorio

Fuente: Los Autores

- prestamo: Permite almacenar información de los préstamos realizados a los estudiantes.

NOMBRE DE LA COLUMNA	TIPO DE DATO	CLAVE PRIMARIA	VALORES NULOS	AUTO INCREMENTAL
cod_prestamo	Serial	SI	NO	SI
detalle	Varchar	NO	NO	NO
Fecha	timestamp	NO	NO	NO
cod_estudiante	Varchar	NO	NO	NO
nombres_estudiante	Varchar	NO	NO	NO
apellidos_estudiante	Varchar	NO	NO	NO
Cedula	Varchar	NO	NO	NO
facultad	Varchar	NO	NO	NO
carrera	Varchar	NO	NO	NO
semestre	Varchar	NO	NO	NO
paralelo	Varchar	NO	NO	NO
periodo	Varchar	NO	NO	NO
estado	Varchar	NO	NO	NO
cod_laboratorio	Integer	NO	NO	NO
cod_usuario	Integer	NO	NO	NO

Tabla 53: Descripción de la tabla prestamo

Fuente: Los Autores

- rol: Es la tabla que permite almacenar datos de los roles que administrarán el sistema.

NOMBRE DE LA COLUMNA	TIPO DE DATO	CLAVE PRIMARIA	VALORES NULOS	AUTO INCREMENTAL
cod_rol	Serial	SI	NO	SI
perfil	Varchar	NO	NO	NO
estado	Varchar	NO	NO	NO

Tabla 54: Descripción de la tabla rol

Fuente: Los Autores

- tipo_ingreso: Permite almacenar el motivo del ingreso de cada uno de los equipos.

NOMBRE DE LA COLUMNA	TIPO DE DATO	CLAVE PRIMARIA	VALORES NULOS	AUTO INCREMENTAL
cod_tipo_i	Serial	SI	NO	SI
descripcion	Varchar	NO	NO	NO
estado	Varchar	NO	NO	NO

Tabla 55: Descripción de la tabla tipo_ingreso

Fuente: Los Autores

- tipo_egreso: Permite almacenar el motivo del egreso (dar de baja) de cada uno de los equipos.

NOMBRE DE LA COLUMNA	TIPO DE DATO	CLAVE PRIMARIA	VALORES NULOS	AUTO INCREMENTAL
cod_tipo_e	Serial	SI	NO	SI
Descripcion	Varchar	NO	NO	NO
Estado	Varchar	NO	NO	NO

Tabla 56: Descripción de la tabla tipo_egreso

Fuente: Los Autores

- usuario: Nos permite almacenar datos de los usuarios del Sistema.

NOMBRE DE LA COLUMNA	TIPO DE DATO	CLAVE PRIMARIA	VALORES NULOS	AUTO INCREMENTAL
cod_usuario	Serial	SI	NO	SI
apellidos	Varchar	NO	NO	NO
nombres	Varchar	NO	NO	NO
nom_usuario	Varchar	NO	NO	NO
contraseña	Integer	NO	NO	NO
cedula	Numeric	NO	NO	NO
correo	Varchar	NO	NO	NO
teléfono	Numeric	NO	NO	NO
Estado	Varchar	NO	NO	NO
cod_rol	Integer	NO	NO	NO

Tabla 57: Descripción de la tabla usuario

Fuente: Los Autores

- facultad: Nos permite almacenar datos de los usuarios del Sistema.

NOMBRE DE LA COLUMNA	TIPO DE DATO	CLAVE PRIMARIA	VALORES NULOS	AUTO INCREMENTAL
cod_facultad	Serial	SI	NO	SI
Descripcion	Varchar	NO	NO	NO
Estado	Varchar	NO	NO	NO

Tabla 58: Descripción de la tabla facultad

Fuente: Los Autores

- carrera: Permite almacenar el nombre de todas las carreras de la UNACH.

NOMBRE DE LA COLUMNA	TIPO DE DATO	CLAVE PRIMARIA	VALORES NULOS	AUTO INCREMENTAL
cod_carrera	Serial	SI	NO	SI
Descripcion	Varchar	NO	NO	NO
Estado	Varchar	NO	NO	NO

Tabla 59: Descripción de la tabla carrera

Fuente: Los Autores

- semestre: Nos permite almacenar la información de los semestres.

NOMBRE DE LA COLUMNA	TIPO DE DATO	CLAVE PRIMARIA	VALORES NULOS	AUTO INCREMENTAL
cod_semestre	Serial	SI	NO	SI
Descripcion	Varchar	NO	NO	NO
Estado	Varchar	NO	NO	NO

Tabla 60: Descripción de la tabla semestre

Fuente: Los Autores

4.3.4. Interfaces de usuario finales

Con la representación detallada de los procesos de gestión de laboratorios de carácter académico y de estudio tales como: ingresos, egresos y préstamos, devoluciones y reportes. Con la información de los diagramas de procesos podemos definir las interfaces de usuario finales.

Interfaz 1: Control de Acceso a Usuarios y Seguridad.

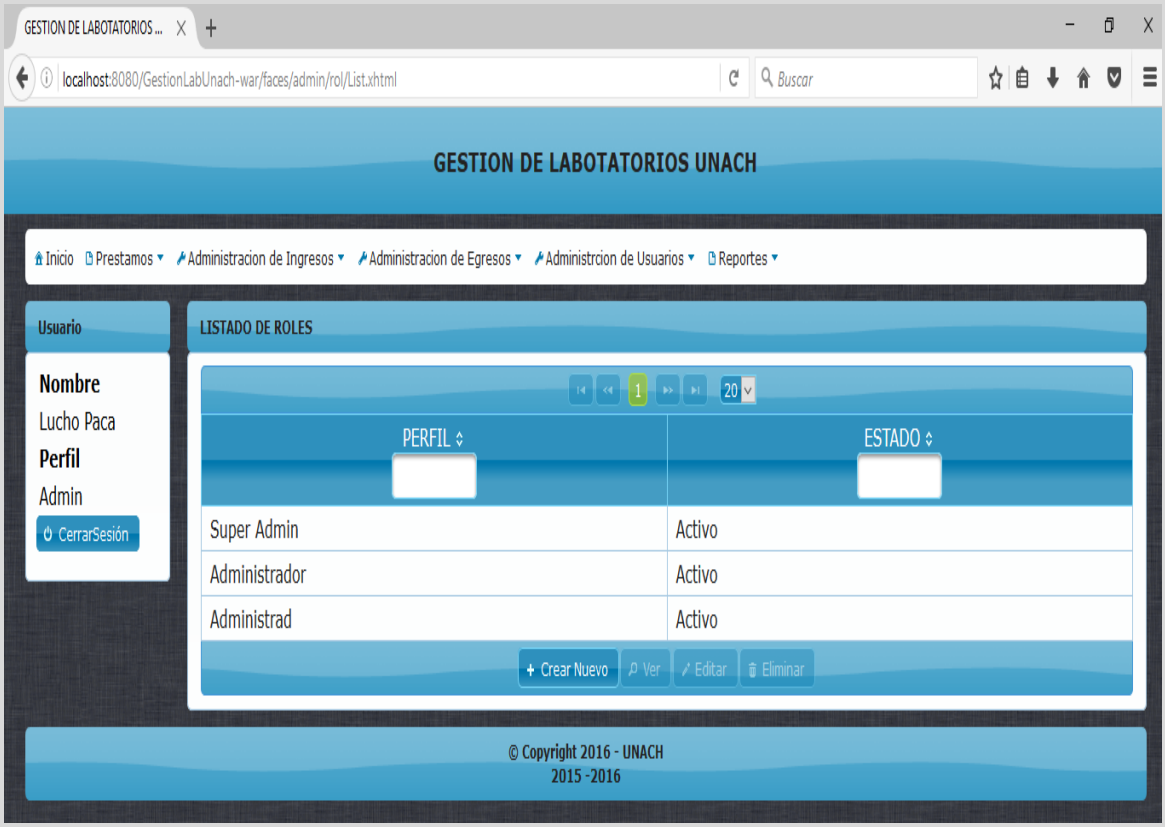
Ilustración 49: Control de Acceso a Usuarios

Fuente: Los Autores

Interfaz 2: Módulo de administración de usuarios y seguridad.

HISTORIA DE USUARIO	
Numero:2	**Usuario:** Administrador
Nombre historia: Módulo de administración de usuarios y seguridad.	
Riesgo en desarrollo: **Alto**	**Iteración asignada:** 3
Esfuerzo: Alto	Interfaz: 2
Programador responsable: Richar Buenaño y Angel Moyón	
Descripción: Posterior a iniciar el sistema se requiere el usuario y la contraseña para poder acceder a los módulos de acuerdo al rol de usuario.	
Observaciones: Hay 3 roles de usuarios: Súper Administrador, Administrador de Laboratorio y Estudiante, con distintos menús y permisos de acceso dependiendo de las funciones de los usuarios.	

Módulo de administración de usuarios y seguridad



Listado de roles, creación.

Tabla 61: Control de Acceso a Usuarios

Fuente: Los Autores

Interfaz 3: Módulo de Gestión de Ingresos

HISTORIA DE USUARIO	
Número:3	**Usuario:** Súper Administrador **y** Administrador del Laboratorio
Nombre historia: Módulo de Gestión de Ingresos	
Riesgo en desarrollo: **Medio**	**Iteración asignada:** 2
Esfuerzo: Medio	Interfaz: 2
Programador responsable: Richar Buenaño y Angel Moyón	
Descripción: Módulo para la inserción de información de los equipos que ingresan a los laboratorios.	
Observaciones: Por medio de este módulo podemos ingresar equipos desde las diferentes bodegas y la razón.	

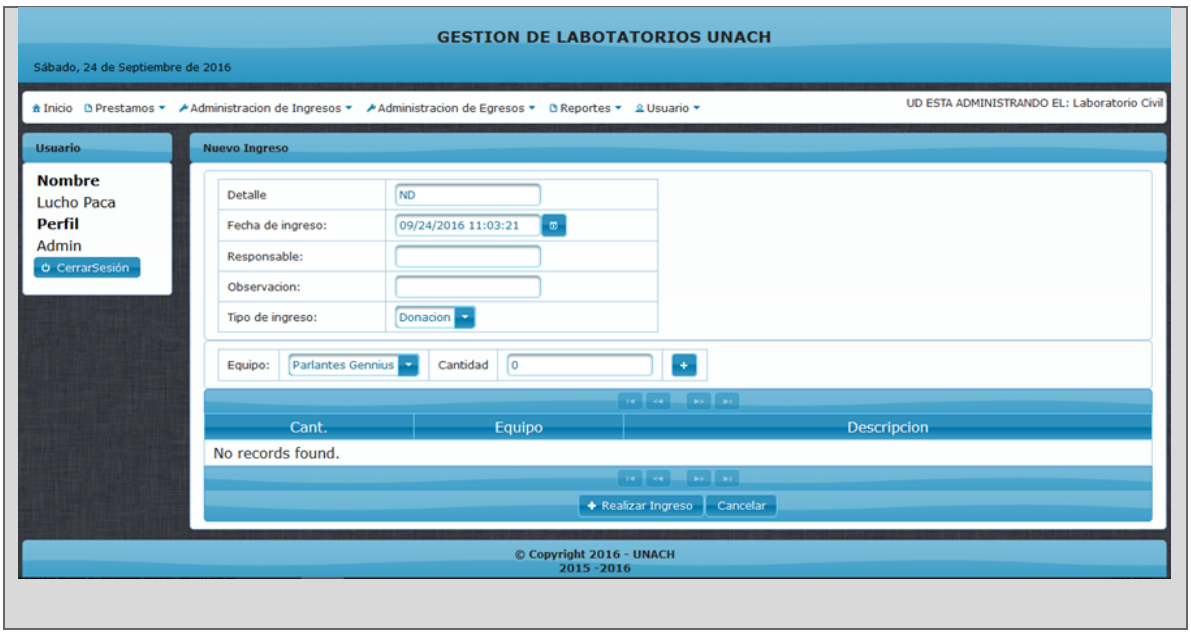


Tabla 62: Control de Ingresos de equipos

Fuente: Los Autores

Interfaz 4: Módulo de Gestión de Egresos

HISTORIA DE USUARIO	
Número:4	**Usuario:** Súper Administrador **y** Administrador del Laboratorio.
Nombre historia: Módulo de Gestión de Egresos	
Prioridad en negocio: Alto	**Riesgo en desarrollo:** Alto
Esfuerzo: Alto	**Iteración asignada:** 1
Programador responsable: Richar Buenaño y Angel Moyón	
Descripción: Módulo para la inserción de información de los equipos que egresan de los laboratorios.	
Observaciones: Por medio de este módulo podemos dar de baja a los equipos en mal estado o que ya hayan cumplido su ciclo de vida útil.	

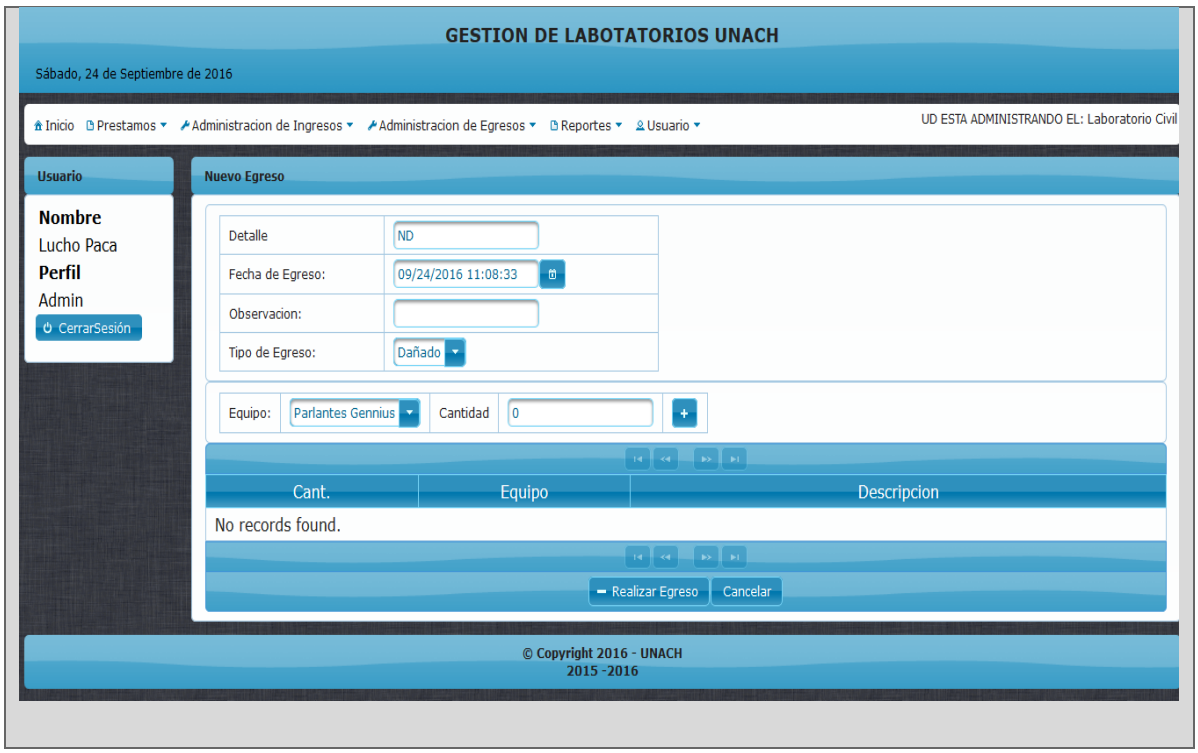


Tabla 63: Control de Egresos de equipos

Fuente: Los Autores

Interfaz 5: Módulo de Gestión de Préstamos

HISTORIA DE USUARIO	
Número:5	**Usuario:** Súper Administrador **y** Administrador
Nombre historia: Módulo de Gestión de préstamos	
Prioridad en negocio: Alto	**Riesgo en desarrollo:** Alto
Esfuerzo: Alto	**Iteración asignada:** 1
Programador responsable: Richar Buenaño y Angel Moyón	
Descripción: Módulo para la inserción de información del préstamo, que realizan los estudiantes.	
Observaciones: Registro de préstamos	

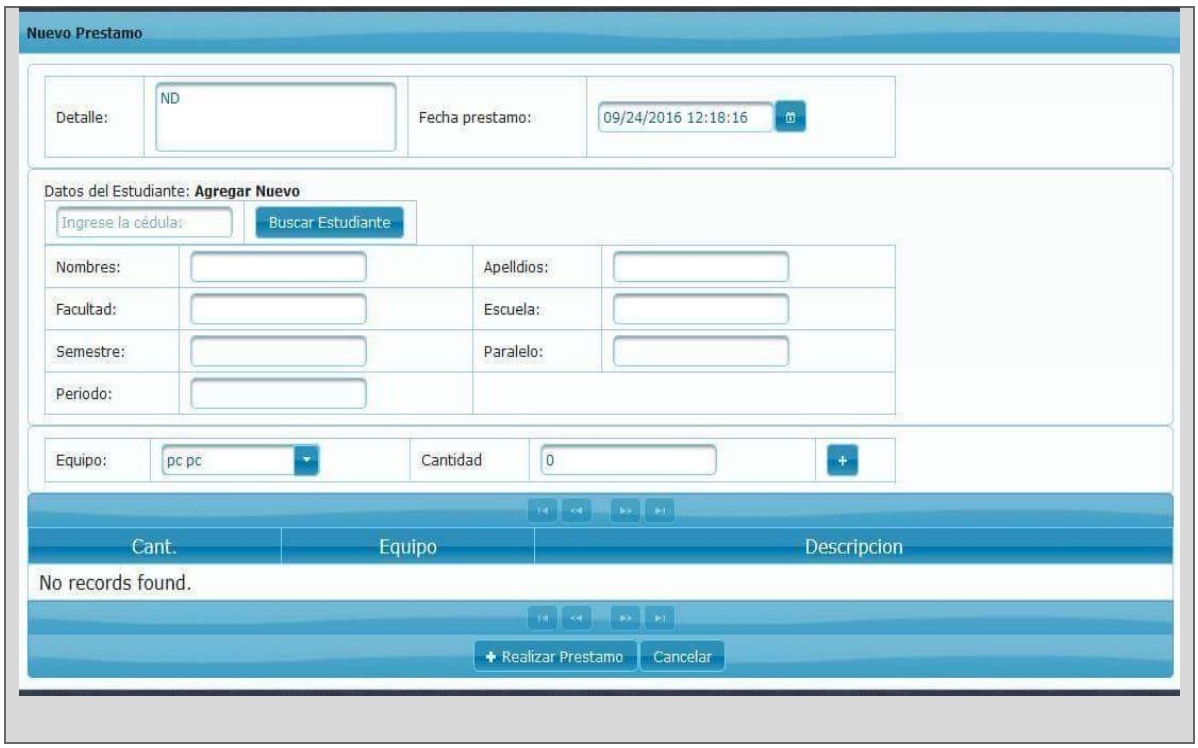


Tabla 64: Control de Préstamos de los equipos

Fuente: Los Autores

Interfaz 6: Módulo de Gestión de Reportes

HISTORIA DE USUARIO	
Número:5	**Usuario:** Súper Administrador, Administrador y Estudiante.
Nombre historia: Módulo de reportes	
Prioridad en negocio: Medio	**Riesgo en desarrollo:** Medio
Esfuerzo: Medio	**Iteración asignada:** 2
Programador responsable: Richar Buenaño y Angel Moyón	
Descripción: Permitirá la visualización de los préstamos y devoluciones que han realizado los estudiantes.	
Observaciones: El listado se da por Laboratorio.	

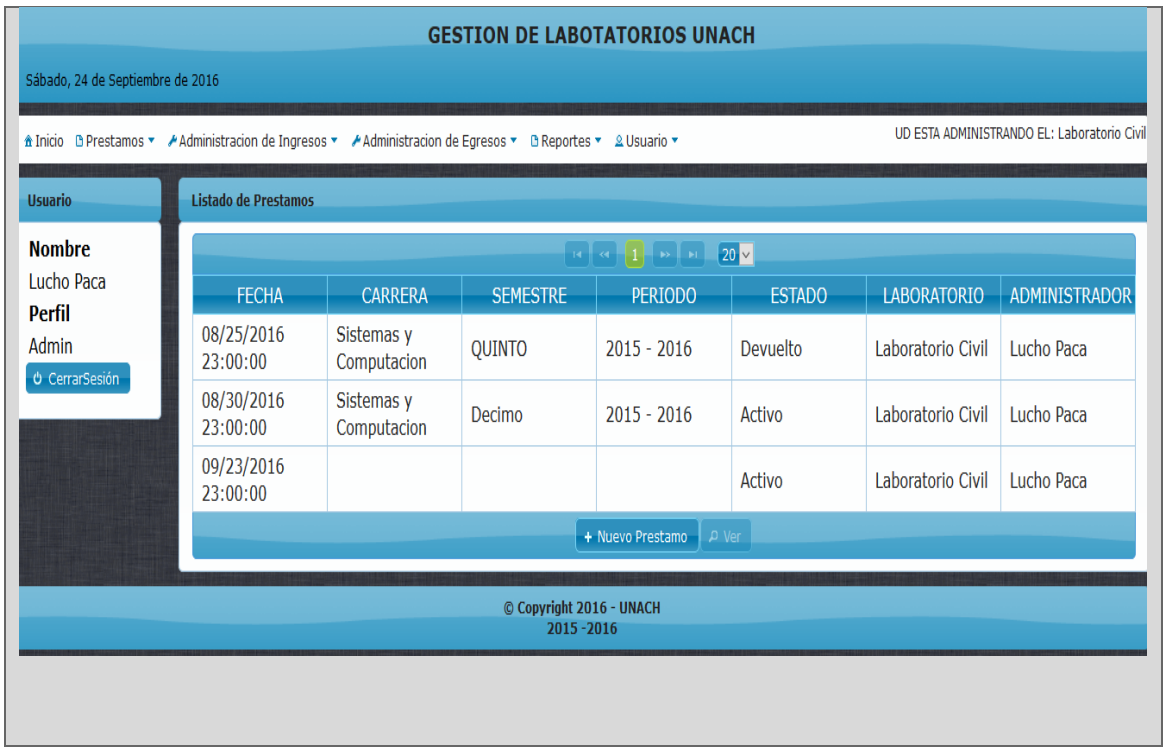


Tabla 65: Módulo de Ingresos de equipos

Fuente: Los Autores

4.3.5. Código fuente

Se anexado a este documento el código fuente de la clase Facade del sistema que se desarrolla debido a que en este código se demuestra la utilidad de esta clase principal para crear la persistencia de los datos y las entidades automáticamente desde la base de datos. Las demás entidades poseen una estructura similar. (Ver anexo 7).

4.4. Implementación

4.4.1. Funcionalidad del sistema

En esta sección se da a conocer las funcionalidades del Sistema de Control de gestión de Laboratorios de la Universidad Nacional de Chimborazo, en la misma se han definido las características principales de la aplicación tomando capturas de pantalla de las principales funcionalidades de la misma y explicando su funcionamiento. Este documento no pretende reemplazar a un manual de usuario, ni tampoco a un documento de ingeniería de software; sino más bien servir como una mera demostración de las funcionalidades principales del sistema que se ha implementado. Para referirse a documentación técnica de la aplicación, se deben consultar los anexos de este documento.

- INTERFAZ PRINCIPAL DE LA APLICACIÓN.

Ilustración 50: Página principal de la aplicación

Fuente: Los Autores

Esta página está enfocada a brindar información general de los módulos en funcionamiento.

- INTERFAZ INICIO DE SESIÓN.

Ilustración 51: Página de inicio de sesión

Fuente: Los Autores

Esta página le permite al usuario acceder a sistema GLU con sus credenciales, para esto, se le solicitará un nombre de usuario y una contraseña registradas, o se le permitirá crear una nueva cuenta. Dependiendo del tipo de cuenta que se haya creado el rol y los permisos que tenga esta, el usuario podrá administrar la información a la que su rol le dé acceso.

- INTERFAZ DE INICIO DE ESTUDIANTE

Ilustración 52: Página de inicio de estudiante

Fuente: Los Autores

Por medio de este perfil el estudiante podrá revisar sus préstamos vigentes y todo su historial.

- INTERFAZ DE INICIO DE ADMINISTRADOR DE LABORATORIO

Ilustración 53: Página de inicio del Administrador

Fuente: Los Autores

Por medio de esta vista y perfil el administrador podrá realizar préstamos y ver el historial del laboratorio asignado.

- INTERFAZ DE INICIO DE SUPER ADMINISTRADOR

Ilustración 54: Página de inicio de Súper Administrador

Fuente: Los Autores

Por medio de esta página el Súper Administrador registra y valida la información para la emisión del certificado.

- INTERFAZ DE INGRESOS

Ilustración 55: Interfaz de ingreso

Fuente: Los Autores

Por medio de esta página el Administrador registra los equipos que pasan a formar parte del Laboratorio.

- INTERFAZ DE EGRESOS

Ilustración 56: Página de egresos

Fuente: Los Autores

Por medio de esta página el Administrador registra y valida la información del egreso de los equipos y el motivo.

- INTERFAZ DE PRESTAMOS

Ilustración 57: Página de préstamos

Fuente: Los Autores

Por medio de esta página el estudiante realiza sus préstamos y devoluciones.

- MENÚ DE REPORTES

Ilustración 58: Página de préstamos

Fuente: Los Autores

Por medio de esta página y de acuerdo al perfil podemos ver los reportes de nuestro historial con el sistema Gestión de Laboratorios UNACH.

4.5. Pruebas

Las pruebas son muy trascendentales y son creadas a partir de las historias de los usuarios, el usuario debe especificar los aspectos que se van a probar cuando una historia de usuario ha sido correctamente realizada, esta prueba de usuario puede tener una o más pruebas de aprobación, las que sean ineludibles para garantizar el correcto funcionamiento.

A continuación, se muestra las pruebas realizadas a cada una de las historias de los usuarios del sistema con su respectiva tabla de pruebas.

Prueba 1

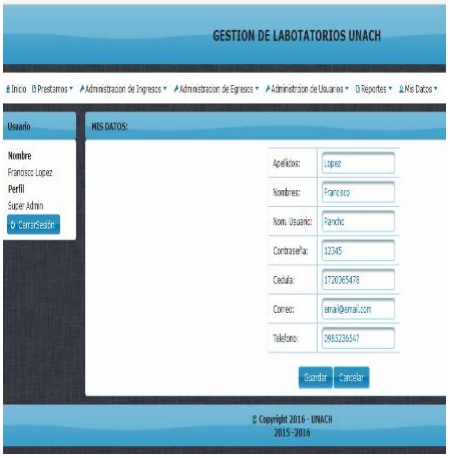
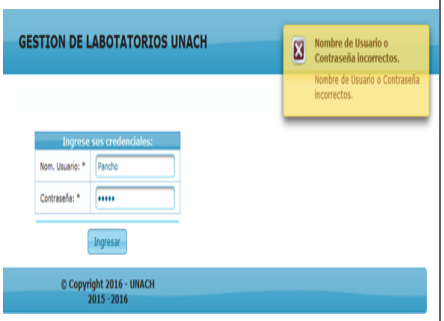
Fecha	Descripción	Autores
08/08/2016	Pruebas	Angel Moyón y Richar Buenaño.
Módulo de administración de usuarios y seguridad.		
Descripción	Hay tres tipos de usuarios: Administrador, Súper Administrador y Estudiante.	
Condiciones de Ejecución.	Cada uno de los usuarios mencionados debe constar en la base de datos y tener asignado un rol y permisos de acceso a los módulos y menús dependiendo de las funciones que le corresponden.	
Entrada	✓ El usuario ingresa su usuario y contraseña ✓ El proceso de control de acceso a Usuarios finaliza.	
Resultado Esperado	Después de ingresar su usuario y su contraseña, debe mostrarse automáticamente la página de inicio del sistema con los menús asignados para cada tipo de usuario.	
Evaluación de la Prueba	**Exitosa** 	**Fallida** 

Tabla 66: Prueba 1

Fuente: Los Autores

Prueba 2

Fecha	Descripción	Autores
08/08/2016	Pruebas	Angel Moyón y Richar Buenaño.
Módulo de Gestión de Ingresos		
Descripción	Hay un tipo de usuario: Administrador	

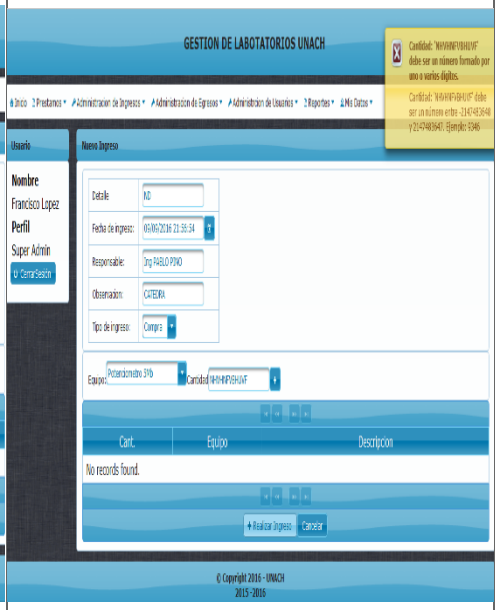
Condiciones de Ejecución.	Cada uno de los usuarios mencionados debe constar en la base de datos y tener asignado un rol y permisos de acceso a los módulos y menús dependiendo de las funciones que le corresponden.	
Entrada	✓ El Administrador del laboratorio realiza el ingreso de equipos. ✓ Administrador gestiona toda la información acerca de los equipos.	
Resultado Esperado	El administrador realiza el registro e ingreso de los equipos con su respectiva información.	
Evaluación de la Prueba	**Exitosa**	**Fallida**
		

Tabla 67: Prueba 2

Fuente: Los Autores

Prueba 3

Fecha	Descripción	Autores
08/08/2016	Pruebas	Angel Moyón y Richar Buenaño.
Módulo de Egresos		
Descripción	Hay un tipo de usuario: Administrador.	
Condiciones de Ejecución.	El usuario mencionado debe constar en la base de datos y tener asignado el rol y permisos de acceso a los módulos y menús dependiendo de las funciones que le corresponden.	
Entrada	✓ Administrador permite por medio del sistema web realizar el egreso de los equipos desde los laboratorios (dar de baja) con toda la información correspondiente.	

Resultado Esperado	Registro de los equipos que salieron de formar parte en algún laboratorio de la UNACH.	
Evaluación de la Prueba	**Exitosa** 	**Fallida** 

Tabla 68: Prueba 3

Fuente: Los Autores

Prueba 4

Fecha	Descripción	Autores
08/08/2016	Pruebas	Angel Moyón y Richar Buenaño.
Módulo de Préstamos.		
Descripción	Hay dos tipos de usuarios: Administrador y Estudiantes.	
Condiciones de Ejecución	Administrador y Estudiante debe constar en la base de datos del sistema para poder generar un reporte.	
Entrada	✓ El Administrador y estudiante interactúan en la gestión de equipos.	
Resultado Esperado	Que los Administradores y estudiantes realicen sus préstamos y devoluciones por medio del sistema web.	
Evaluación de la Prueba	**Exitosa**	**Fallida** En la parte de préstamos cargaremos como las pruebas fallidas se dan por los tipos de datos

Tabla 69: Prueba 4

Fuente: Los Autores

Prueba 5

Fecha	Descripción	Autores
08/08/2016	Pruebas	Angel Moyón y Richar Buenaño.
Módulo de Reportes		
Descripción	El Estudiante ingresa al sistema para administrar sus préstamos.	
Condiciones de Ejecución	Estudiante debe constar en la base de datos del sistema para poder ingresar al sistema.	
Entrada	✓ El estudiante una vez que ingresa al sistema elige su perfil y su reporte.	
Resultado Esperado	Que el estudiante obtenga mediante la aplicación web su certificado de NO ADEUDAR.	

Evaluación de la Prueba	**Exitosa**  UNIVERSIDAD NACIONAL DE CHIMBORAZO CERTIFICADO PARA TRAMITE DE FIN DE CARRERA El (la) señor (a): **Moyón Aulla Angel Estuardo** Cédula de Identidad No: **0603794231** Alumno (a) de la Facultad de: **Ingenieria** Carrera: **Sistemas y Computación** No adeuda materiales, componentes o reparación en los equipos de los Laboratorios, de la Universidad Nacional de Chimborazo La presente certificación, tiene validez de 30 días calendario a partir de su emision Fecha generada: sábado 10 septiembre 2016 RIOBAMBA-ECUADOR

Tabla 70: Prueba 5

Fuente: Los Autores

CONCLUSIONES

- El estudio de las características de la tecnología EJB permitió determinar que la tecnología de programación EJB posee un mayor nivel de rendimiento con un 75,00% frente a la Programación Tradicional que obtuvo un 66,66%, dando como resultado una diferencia del 8,34%.

- De acuerdo a los resultados del análisis se llegó a obtener en peticiones del usuario una diferencia del 37,5%, en el parámetro carga de usuarios se obtuvo una diferencia significativa del 0% y en el uso de hardware la tecnología EJB decrece en un 25%, comprobándose que la tecnología de desarrollo EJB incide en el rendimiento con una diferencia significativa del 8,34%, de acuerdo al análisis de estadística descriptiva y el sistema SIAE.

- La metodología RUP permitió desarrollar la aplicación, por medio de los requisitos del usuario, diagramas de clases, historias de usuarios, diagramas de casos de uso y diseños técnicos y conceptuales.

- En el tiempo de desarrollo se encontró con el mapeo de información en donde para nosotros fue muy difícil encontrar una parte específica del mapeo de la base de datos, debiendo recurrir a los mensajes del servidor GlassFish en el caso que fuese error y en un caso diferente por búsqueda en cada archivo con palabras claves.

- Se eligió la Tecnología de Programación Enterprise JavaBeans EJB, para desarrollar la aplicación empresarial JEE con incidencia en el rendimiento utilizando la información de los laboratorios, con lo que se logró integrar el estudio de las tecnologías de codificación en un caso aplicativo real, en beneficio de la Universidad Nacional de Chimborazo.

RECOMENDACIONES

- El IDE de Netbeans no es muy recomendable, ya que consume más recursos de nuestros equipos en comparación con otros IDE.

- Se dice que desde el punto de vista del negocio es siempre recomendable utilizar este tipo de frameworks. Primero, porque al momento de implementar la aplicación, se enfoca en lo que concierne solo a la lógica de negocios, que es lo que tu manejas a la perfección y ya no te preocupas por implementar código que concierne a la lógica de manejo de transacciones, seguridad, etc.

- Se recomienda utilizar la metodología RUP en el diseño y construcción de aplicaciones empresariales JEE, porque nos permite tener un completo análisis y llevar una documentación eficiente.

- Se invita para futuras investigaciones realizar un análisis de la Tecnología EJB con respecto a otros factores como es el caso de seguridad y productividad.

BIBLIOGRAFÍA

Acosta, M. Y. (2013). ESTUDIO DE PATRONES DE DISEÑO EN PLATAFORMA JAVA ENTERPRISE. Ibarra: UNIVERSIDAD TÉCNICA DEL NORTE.

Aumaille, B. (2002). *Desarrollo de Aplicaciones Web J2EE* . Eni Ediciones.

Ed Roman, Rima Patel Sriganesh, Gerald Brose. (2007). *Mastering Enterprise JavaBeans™, Third Edition.* Michigan: University Of Michigan.

Eric Armstrong, Jennifer Ball, Stephanie Bodoff, Debbie Carson, Ian Evans, Maydene Fisher, Dale Green, Kim Haase, Eric Jendrock. (2003). *The J2EE™ 1.4 Tutorial.* California: Sun Microsystems.

Ferguson, J. S. (2007). *JSF Jumpstart.* Wellington, New Zealand: Wakaleo Consulting Limited.

GÁLVEZ ROJAS, S., & ORTEGA DÍAZ, L. (2003). *Java 2 Micro Edition J2ME.* Málaga: Sun Microsystems.

Gelernter, B. (2011). *Oracle Fusion Middleware Programming Enterprise JavaBeans, Version 3.0, for Oracle WebLogic Server, 11g.* Washington DC: Copyright © 2007, 2011, Oracle and/or its affiliates. .

Hall, M. (2007). *Core Servlets and JavaServer Pages.* Washington: © Prentice Hall and Sun Microsystems. .

Hernández, L. A. (2006). *Programación Orientada a Objetos en Java.* México.

JBoss_Inc., J. G. (2005). *The JBoss 4 Application Server Guide.* JBoss Group Inc.

Linda DeMichiel - Sun Microsystems, Michael Keith - Oracle Corporation. (2006). *JSR 220: Enterprise JavaBeans.* California, USA: Copyright 2006 SUN MICROSYSTEMS, INC.

Matena, V., & Hapner, M. (2008). *Enterprise JavaBeans.* California: Sun Microsystems Inc.

Mulder, A. (2007). *Apache Gerónimo, Enterprise Java Development and deployment.* Pearson Addison Wesley Prof.

Óscar Belmonte, Carlos Granell Canut, María del Carmen Erdozain Navarro. (2012). *DESARROLLO DE PROYECTOS INFORMATICOS, CON TECNOLOGÍA JAVA.* Castelló: Universitat Jaume I.

PATIÑO LUIS, G. (2014). *"ANÁLISIS COMPARATIVO DE METRO Y AXIS2 PARA EL.* Riobamba.

Roldán, Á. (15 de Octubre de 2015). *http://www.ciberaula.com/.* Obtenido de http://www.ciberaula.com/: http://www.ciberaula.com/curso/java/que_es/

Sánchez, J. (2004). *Manual completo de programación en Java.* Palencia.

Villacrés, O. C. (2011). *ESTUDIO DE LA ARQUITECTURA DE LOS COMPONENTES EJB (ENTERPRISE JAVABEANS).* Riobamba: ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO.

Yerovi, M. L. (2013). *ESTUDIO DE PATRONES DE DISEÑO EN PLATAFORMA JAVA ENTERPRISE.* Ibarra: UNIVERSIDAD TÉCNICA DEL NORTE.

Yerovi, M. L. (2013). *ESTUDIO DE PATRONES DE DISEÑO EN PLATAFORMA JAVA ENTERPRISE* . Ibarra: UNIVERSIDAD TÉCNICA DEL NORTE.

ANEXOS

Anexo 1: Tabla de valores de resultados número de peticiones soportadas

Prototipo	Prueba	x_0	(x)	$\lvert x_0\text{-}x\rvert$	$\frac{(\lvert x_0 - x\rvert)}{x_0}$	$\bar{x}$	$(\bar{x}\text{-}x)$	$(\bar{x}\text{-}x)^2$
Tecnología EJB	1	68	68	0,0	0,0000	70,9	2,9	8,41
	2	72,8	73	0,8	0,0110		-2,1	4,41
	3	81,6	82	0,6	0,0074		-11,1	123,21
	4	74,5	75	0,5	0,0067		-4,1	16,81
	5	73,7	74	0,7	0,0095		-3,1	9,61
	6	68,6	69	0,6	0,0087		1,9	3,61
	7	67,7	68	0,7	0,0103		2,9	8,41
	8	78,4	78	0,4	0,0051		-7,1	50,41
	9	63,3	63	0,3	0,0047		7,9	62,41
	10	60,4	60	0,4	0,0066		10,9	118,81
Programación Tradicional	1	47,8	48	0,8	0,0167	52,95	4,95	24,50
	2	52,4	52	0,4	0,0076		0,95	0,9025
	3	58,04	58	0,0	0,0007		-5,05	25,5025
	4	55,6	56	0,6	0,0108		-3,05	9,3025
	5	16,1	16	0,1	0,0062		36,95	1365,303
	6	60,6	61	0,6	0,0099		-8,05	64,8025
	7	61,2	61	0,2	0,0033		-8,05	64,8025
	8	59,7	60	0,7	0,0117		-7,05	49,7025
	9	58,1	58	0,1	0,0010		-5,05	25,5025
	10	60	60	0,0	0,0000		-7,05	49,7025

Anexo 2: Tabla de valores de resultados tiempo de ejecución de solicitud

Prototipo	**Prueba**	x_0	**(x)**	$\lvert x_0\text{-}x\rvert$	$\frac{(\lvert x_0 - x\rvert)}{x_0}$	$\bar{x}$	$(\bar{x}\text{-}x)$	$(\bar{x}\text{-}x)^2$
Tecnología EJB	**1**	**10,7**	**11**	**0,8**	**0,0713**	**11,1**	**0,10**	**0,01**
	2	**7,80**	**8**	**0,8**	**0,1024**		**3,1**	**9,61**
	3	**7,84**	**8**	**0,8**	**0,1068**		**3,3**	**10,72**
	4	**10,10**	**10**	**0,1**	**0,0098**		**1,1**	**1,21**
	5	**11,8**	**12**	**0,8**	**0,0646**		**-0,9**	**0,81**
	6	**11,3**	**11**	**0,3**	**0,0288**		**0,10**	**0,01**
	7	**11,9**	**12**	**0,9**	**0,0764**		**-0,9**	**0,81**
	8	**8,8**	**9**	**0,8**	**0,0950**		**2,1**	**4,41**
	9	**13,8**	**14**	**0,8**	**0,0569**		**-2,9**	**8,41**
	10	**16,99**	**17**	**1,0**	**0,0583**		**-5,9**	**34,56**
Programación Tradicional	**1**	**29,18**	**29**	**0,2**	**0,0062**	**30,1**	**1,15**	**1,32**
	2	**24,17**	**24**	**0,2**	**0,0070**		**6,15**	**37,82**
	3	**19,13**	**19**	**0,1**	**0,0067**		**11,15**	**124,32**
	4	**25,81**	**26**	**0,8**	**0,0314**		**4,15**	**17,22**
	5	**105,14**	**105**	**0,1**	**0,0013**		**-74,85**	**5602,52**
	6	**18,28**	**18**	**0,3**	**0,0153**		**12,15**	**147,62**

	7	**16,53**	**17**	**0,5**	**0,0318**		**13,15**	**172,92**
	8	**23,85**	**24**	**0,8**	**0,0356**		**6,15**	**37,82**
	9	**17,12**	**17**	**0,1**	**0,0067**		**13,15**	**172,92**
	10	**22,35**	**22**	**0,4**	**0,0157**		**8,15**	**66,42**

Anexo 3: Tabla de valores de resultados número de usuarios concurrentes

Prototipo	**Prueba**	x_0	**(x)**	$\lvert x_0$**-x**$\rvert$	$\frac{(\lvert x_0 - x\rvert)}{x_0}$	$\bar{x}$	$(\bar{x}$**-x)**	$(\bar{x}$**-x)²**
Tecnología EJB	**1**	**1284**	**1284**	**0,0**	**0,0000**	**1323**	**39,0**	**1519,44**
	2	**689,93**	**689,9**	**0,9**	**0,0013**		**633,1**	**400790,3**
	3	**925,05**	**925,1**	**0,0**	**0,0001**		**397,9**	**158324,4**
	4	**1350,1**	**1350**	**0,1**	**0,0001**		**-27,0**	**729**
	5	**1604,11**	**1604**	**0,1**	**0,0001**		**-281,0**	**78961**
	6	**1423,97**	**1424**	**1,0**	**0,0007**		**-101,0**	**10192,92**
	7	**1518,93**	**1519**	**0,9**	**0,0006**		**-196,0**	**38416**
	8	**1136,04**	**1136**	**0,0**	**0,0000**		**187,0**	**34957,78**
	9	**1820,97**	**1821**	**1,0**	**0,0005**		**-498,0**	**247964,2**
	10	**1477**	**1477**	**1,0**	**0,0007**		**-154,0**	**23709,84**
Programación Tradicional	**1**	**476,1**	**476**	**0,1**	**0,0002**	**806,53**	**330,53**	**109250,1**
	2	**837,04**	**837**	**0,0**	**0,0000**		**-30,47**	**928,4209**
	3	**777,93**	**778**	**0,9**	**0,0012**		**28,53**	**813,9609**
	4	**533,91**	**534**	**0,9**	**0,0017**		**272,53**	**74272,6**

	5	**134,94**	**135**	**0,9**	**0,0070**		**671,53**	**450952,5**
	6	**948,08**	**948**	**0,1**	**0,0001**		**-141,47**	**20013,76**
	7	**1041,1**	**1041**	**0,1**	**0,0001**		**-234,47**	**54976,18**
	8	**978,07**	**978**	**0,1**	**0,0001**		**-171,47**	**29401,96**
	9	**1111,1**	**1111**	**0,0**	**0,0000**		**-304,47**	**92701,98**
	10	**1227,1**	**1227**	**0,1**	**0,0001**		**-420,47**	**176795**

Anexo 4: Tabla de valores de los resultados número de usuarios en espera.

Prototipo	Prueba	x_0	(x)	$\lvert x_0$-x$\rvert$	$\frac{(\lvert x_0 - x \rvert)}{x_0}$	$\bar{x}$	($\bar{x}$-x)	($\bar{x}$-x)²
Tecnología EJB	1	888,97	889	1,0	0,0011	850	-39,0	1519,44
	2	1483,1	1483	0,1	0,0000		-633,0	400689
	3	1248	1248	1,0	0,0008		-398,0	158372,2
	4	822,92	823	0,9	0,0011		27,0	729
	5	568,9	569	0,9	0,0016		281,0	78961
	6	749,0	749	0,0	0,0000		196,0	38416
	7	654,1	654	0,1	0,0001		195,9	38384,65
	8	1036,96	1037	1,0	0,0009		-187,0	34957,78
	9	352,03	352	0,0	0,0001		498,0	247964,2
	10	696,01	696	0,0	0,0000		154,0	23709,84
Programación Tradicional	1	1696,90	1697	0,9	0,0005	1366,51	-330,49	109223,6
	2	1336	1336	1,0	0,0007		30,51	930,8601
	3	1395,07	1395	0,1	0,0001		-28,49	811,6801
	4	1639,49	1639	0,5	0,0003		-272,49	74250,8
	5	2038,06	2038	0,1	0,0000		-671,49	450898,8
	6	1224,92	1225	0,9	0,0008		141,51	20025,08
	7	1131,9	1132	0,9	0,0008		234,51	54994,94

	8	1194,93	1195	0,9	0,0008		171,51	29415,68
	9	1062	1062	1,0	0,0009		304,51	92726,34
	10	945,91	946	0,9	0,0010		420,51	176828,7

Anexo 5: Resultados de las pruebas para el uso de memoria RAM

Prototipo	Prueba	x_0	(x)	$\lvert x_0\text{-}x\rvert$	$\frac{(\lvert x_0 - x\rvert)}{x_0}$	$\bar{x}$	$(\bar{x}\text{-}x)$	$(\bar{x}\text{-}x)^2$
Tecnología EJB	1	246,25	246	0,3	0,0010	201,1	-44,9	2016,01
	2	115,24	115	0,2	0,0021		86,1	7413,21
	3	224,81	225	0,8	0,0036		-23,9	571,21
	4	113,27	113	0,3	0,0024		88,1	7761,61
	5	242,9	243	0,9	0,0037		-41,9	1755,61
	6	255,4	255	0,4	0,0015		-53,9	2905,21
	7	122,7	123	0,7	0,0053		78,1	6099,61
	8	283,1	283	0,1	0,0004		-81,9	6707,61
	9	194,3	194	0,3	0,0013		7,1	50,41
	10	212,81	213	0,8	0,0038		-11,9	141,61
Programación Tradicional	1	167,63	168	0,6	0,0038	182,38	14,38	206,78
	2	171,34	171	0,3	0,0020		11,38	129,50
	3	176,63	177	0,6	0,0036		5,38	28,94

	4	**183,02**	**183**	**0,0**	**0,0001**		**-0,62**	**0,38**
	5	**155,66**	**156**	**0,7**	**0,0042**		**26,38**	**695,90**
	6	**202,60**	**203**	**0,6**	**0,0029**		**-20,62**	**425,18**
	7	**154,34**	**154**	**0,3**	**0,0022**		**28,38**	**805,42**
	8	**195,54**	**196**	**0,5**	**0,0028**		**-13,62**	**185,50**
	9	**178,22**	**178**	**0,2**	**0,0012**		**4,38**	**19,18**
	10	**238,87**	**239**	**0,9**	**0,0036**		**-56,62**	**3205,82**

Anexo 6: Resultados de las pruebas para el uso de procesador CPU

Prototipo	Prueba	x_0	(x)	$\lvert x_0$-x$\rvert$	$\frac{(\lvert x_0 - x \rvert)}{x_0}$	$\bar{x}$	($\bar{x}$-x)	($\bar{x}$-x)²
Tecnología EJB	1	84,3	84	0,3	0,0036	84,0	0,0	0
	2	80,6	81	0,6	0,0074		3,0	9
	3	82,1	82	0,1	0,0012		2,0	4
	4	84,1	84	0,1	0,0012		0,0	0
	5	85,1	85	0,1	0,0012		-1,0	1
	6	86,8	87	0,8	0,0092		-3,0	9
	7	85,7	86	0,7	0,0082		-2,0	4
	8	83,9	84	0,9	0,0107		0,0	0
	9	84,6	85	0,6	0,0071		-1,0	1
	10	82,4	82	0,4	0,0049		2,0	4
Programación Tradicional	1	46,3	46	0,3	0,0065	33,58	-12,42	154,26
	2	45,1	45	0,1	0,0022		-11,42	130,42
	3	24,5	25	0,5	0,0204		8,58	73,62
	4	47,5	48	0,5	0,0105		-14,42	207,94
	5	23,7	24	0,7	0,0295		9,58	91,78
	6	28,3	28	0,3	0,0106		5,58	31,14
	7	39,6	40	0,6	0,0152		-6,42	41,22
	8	28,4	28	0,4	0,0141		5,58	31,14
	9	28,7	29	0,7	0,0244		4,58	20,98
	10	23,7	24	0,7	0,0295		9,58	91,78

Anexo 7: com.gestionLab.Modelos, AbstractFacade

```
package com.gestionLab.Modelos;

import java.util.List;

import java.util.Map;

import javax.persistence.EntityManager;

import javax.persistence.NoResultException;

import javax.persistence.Query;

public abstract class AbstractFacade<T> {

   private Class<T> entityClass;

   public AbstractFacade(Class<T> entityClass) {

      this.entityClass = entityClass;

   }

   protected abstract EntityManager getEntityManager();

   public void create(T entity) {

      getEntityManager().persist(entity);

   }

   public void edit(T entity) {

      getEntityManager().merge(entity);

   }

   public void remove(T entity) {


getEntityManager().remove(getEntityManager().merge(entity));

   }

   public T find(Object id) {

      return getEntityManager().find(entityClass, id);

   }

   public List<T> findAll() {

      javax.persistence.criteria.CriteriaQuery cq = getEntityManager().getCriteriaBuilder().createQuery();

      cq.select(cq.from(entityClass));

      return getEntityManager().createQuery(cq).getResultList();

   }

   public List<T> findRange(int[] range) {

      javax.persistence.criteria.CriteriaQuery cq = getEntityManager().getCriteriaBuilder().createQuery();

      cq.select(cq.from(entityClass));

      javax.persistence.Query q = getEntityManager().createQuery(cq);

      q.setMaxResults(range[1] - range[0] + 1);

      q.setFirstResult(range[0]);

      return q.getResultList();

   }

   public int count() {

      javax.persistence.criteria.CriteriaQuery cq = getEntityManager().getCriteriaBuilder().createQuery();

      javax.persistence.criteria.Root<T> rt = cq.from(entityClass);


cq.select(getEntityManager().getCriteriaBuilder().count(rt));

      javax.persistence.Query q = getEntityManager().createQuery(cq);

      return ((Long) q.getSingleResult()).intValue();

   }

   // AGREGADO

   protected T findOneResult(String namedQuery, Map<String, Object> parameters) {

      T result = null;

      try {
```

```
        Query query = getEntityManager().createNamedQuery(namedQuery);

        // Method that will populate parameters if they are passed not null and empty

        if (parameters != null && !parameters.isEmpty()) {

            populateQueryParameters(query, parameters);

        }

        result = (T) query.getSingleResult();

    } catch (NoResultException e) {

        System.out.println("No result found for named query: " + namedQuery);

    } catch (Exception e) {

        System.out.println("Error while running query: " + e.getMessage());

        e.printStackTrace();

    }

    return result;

  }

    private void populateQueryParameters(Query query, Map<String, Object> parameters) {

    for (Map.Entry<String, Object> entry : parameters.entrySet()) {

        query.setParameter(entry.getKey(), entry.getValue());

    }  }

    protected List<T> findAllResults(String namedQuery, Map<String, Object> parameters) {

    List<T> result = null;

    try {

        Query query = getEntityManager().createNamedQuery(namedQuery);

        // Method that will populate parameters if they are passed not null and empty

        if (parameters != null && !parameters.isEmpty()) {

            populateQueryParameters(query, parameters);

        }

        result = (List<T>) query.getResultList();

    } catch (NoResultException e) {

        System.out.println("No result found for named query: " + namedQuery);

    } catch (Exception e) {

        System.out.println("Error while running query: " + e.getMessage());

        e.printStackTrace();

    }

    return result;

  }}
```

Anexo 8: GestionLabUnach-war, Controlador Prestamos

```
package com.gestionLab.Controlador;

import com.gestionLab.Entidades.Prestamo;

import com.gestionLab.Controlador.util.JsfUtil;

import com.gestionLab.Controlador.util.JsfUtil.PersistAction;

import com.gestionLab.Entidades.DetallePrestamo;

import com.gestionLab.Entidades.Equipo;

import com.gestionLab.Entidades.Usuario;

import com.gestionLab.Modelos.DetallePrestamoFacade;

import com.gestionLab.Modelos.PrestamoFacade;

import java.io.Serializable;

import java.util.ArrayList;

import java.util.List;

import java.util.ResourceBundle;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.ejb.EJB;

import javax.ejb.EJBException;

import javax.faces.bean.ManagedBean;

import javax.faces.bean.SessionScoped;

import javax.faces.component.UIComponent;

import javax.faces.context.FacesContext;

import javax.faces.convert.Converter;

import javax.faces.convert.FacesConverter;

@ManagedBean(name = "prestamoController")

@SessionScoped

public class PrestamoController implements Serializable {

    @EJB

    private DetallePrestamoFacade detallePrestamoFacade;

    @EJB

    private com.gestionLab.Modelos.PrestamoFacade ejbFacade;

    private List<Prestamo> items = null;

    private Prestamo selected;

    public PrestamoController() {

    }

    public Prestamo getSelected() {

        return selected;

    }

    public void setSelected(Prestamo selected) {

        this.selected = selected;

    }

    protected void setEmbeddableKeys() {

    }

    protected void initializeEmbeddableKey() {

    }

    private PrestamoFacade getFacade() {

        return ejbFacade;

    }

    public Prestamo prepareCreate() {

        selected = new Prestamo();

        initializeEmbeddableKey();

        return selected;

    }

    public void create() {

        persist(PersistAction.CREATE, ResourceBundle.getBundle("/Recursos").getString("PrestamoCreated"));
```

```
        if (!JsfUtil.isValidationFailed()) {

            items = null;    // Invalidate list of items to trigger re-query.

        }    }

    public void update() {

        persist(PersistAction.UPDATE, ResourceBundle.getBundle("/Recursos").getString("PrestamoUpdated"));

    }

    public void destroy() {

        persist(PersistAction.DELETE, ResourceBundle.getBundle("/Recursos").getString("PrestamoDeleted"));

        if (!JsfUtil.isValidationFailed()) {

            selected = null; // Remove selection

            items = null;    // Invalidate list of items to trigger re-query.

        }    }

    public List<Prestamo> getItems() {

        if (items == null) {

            items = getFacade().findAll();

        }

        return items;

    }

    ////TABLA DETALLE PRESTAMO

    private List<DetallePrestamo> lstEqpPrestamo = new ArrayList();

    private int cantidad;

    private Equipo objetoEquipo = new Equipo();

    public List<DetallePrestamo> getLstEqpPrestamo() {

        return lstEqpPrestamo;

    }

    public void setLstEqpPrestamo(List<DetallePrestamo> lstEqpPrestamo) {

        this.lstEqpPrestamo = lstEqpPrestamo;

    }

    public int getCantidad() {

        return cantidad;

    }

    public void setCantidad(int cantidad) {

        this.cantidad = cantidad;

    }

    public Equipo getObjetoEquipo() {

        return objetoEquipo;

    }

    public void setObjetoEquipo(Equipo objetoEquipo) {

        this.objetoEquipo = objetoEquipo;

    }

    public void agregar() {

        DetallePrestamo det = new DetallePrestamo();

        det.setCantidad(cantidad);

        det.setCodEquipo(objetoEquipo);

        this.lstEqpPrestamo.add(det);

    }    ////////////////////////

    private List<DetallePrestamo> listaDetallePrestamo;

    public void listar() {

        try {

            int codPrestamo = selected.getCodPrestamo();

            System.out.println("coigo usuario para lista egreso: " + codPrestamo);
```

```
        listaDetallePrestamo = detallePrestamoFacade.listaDetallePrestamo(codPrestamo);
      } catch (Exception e) {
         System.out.println(e.getMessage());
      }   }
   public List<DetallePrestamo> getListaDetallePrestamo() {
      return listaDetallePrestamo;
   }
   public void setListaDetallePrestamo(List<DetallePrestamo> listaDetallePrestamo) {
      this.listaDetallePrestamo = listaDetallePrestamo;
   }
  ////////////////////////////////////
   private void persist(PersistAction persistAction, String successMessage) {
      if (selected != null) {
         setEmbeddableKeys();
         try {
            if (persistAction != PersistAction.DELETE) {
               getFacade().edit(selected);
            } else {
               getFacade().remove(selected);
            }
JsfUtil.addSuccessMessage(successMessage);
         } catch (EJBException ex) {
            String msg = "";
            Throwable cause = ex.getCause();
            if (cause != null) {
               msg = cause.getLocalizedMessage();
            }
            if (msg.length() > 0) {
               JsfUtil.addErrorMessage(msg);
            } else {
               JsfUtil.addErrorMessage(ex, ResourceBundle.getBundle("/Recursos").getString("PersistenceErrorOccured"));
            }
         } catch (Exception ex) {
Logger.getLogger(this.getClass().getName()).log(Level.SEVERE, null, ex);
            JsfUtil.addErrorMessage(ex, ResourceBundle.getBundle("/Recursos").getString("PersistenceErrorOccured"));
         }      }   }
   public List<Prestamo> getItemsAvailableSelectMany() {
      return getFacade().findAll();
   }
   public List<Prestamo> getItemsAvailableSelectOne() {
      return getFacade().findAll();
   }
   @FacesConverter(forClass = Prestamo.class)
   public static class PrestamoControllerConverter implements Converter {
      @Override
      public Object getAsObject(FacesContext facesContext, UIComponent component, String value) {
         if (value == null || value.length() == 0) {
            return null;
         }
```

```
        PrestamoController controller = (PrestamoController) facesContext.getApplication().getELResolver().

getValue(facesContext.getELContext(), null, "prestamoController");

        return controller.getFacade().find(getKey(value));

    }

    java.lang.Integer getKey(String value) {

        java.lang.Integer key;

        key = Integer.valueOf(value);

        return key;

    }

    String getStringKey(java.lang.Integer value) {

        StringBuilder sb = new StringBuilder();

        sb.append(value);

        return sb.toString();

    }

    @Override

    public String getAsString(FacesContext facesContext, UIComponent component, Object object) {

        if (object == null) {

            return null;

        }

        if (object instanceof Prestamo) {

            Prestamo o = (Prestamo) object;

            return getStringKey(o.getCodPrestamo());

        } else {

Logger.getLogger(this.getClass().getName()).log( Level.SEVERE, "object {0} is of type {1}; expected type: {2}", new Object[]{object, object.getClass().getName(), Prestamo.class.getName()});

            return null;

        }    }  }

  ////////////////////////////////////////

  private List<Prestamo> listaPrestamoUsuario;

  public List<Prestamo> getListaPrestamoUsuario() {

      if (listaPrestamoUsuario == null) {

        Usuario us = (Usuario) FacesContext.getCurrentInstance().getExternalContext().getSessionMap().get("usuario");

        System.out.println("coigo usuario para lista egreso: " + us.getCodUsuario());

        listaPrestamoUsuario = ejbFacade.listarPrestamoUsuario(us.getCodUsuario());

      }

      return listaPrestamoUsuario;

  }

  public void setListaPrestamoUsuario(List<Prestamo> listaPrestamoUsuario) {

      this.listaPrestamoUsuario = listaPrestamoUsuario;

  }}
```

Anexo 9: GestionLabUnach-war, Entidad Préstamo

```
package com.gestionLab.DAO;

import com.gestionLab.Entidades.Laboratorio;

import com.gestionLab.Entidades.Usuario;

import java.io.Serializable;

import java.util.Calendar;

import java.util.Date;

public class cPrestamo implements Serializable{

   private int cod_prestamo;

   private String detalle;

   private Date fecha;

   private String cod_estudiante;

   private String nombres_estudiante;

   private String apellidos_estudiante;

   private String cedula;

   private String facultad;

   private String carrera;

   private String semestre;

   private String pararlelo;

   private String periodo;

   private Laboratorio laboratorio;

   private Usuario usuario;

   private String estado;

   public cPrestamo() {

      java.util.Date d = Calendar.getInstance().getTime();

      java.sql.Date sqlDate = new java.sql.Date(d.getTime());

      this.cod_prestamo = 0;

      this.detalle = "ND";

      this.fecha = sqlDate;

      this.cod_estudiante = "";

      this.nombres_estudiante = "";

      this.apellidos_estudiante = "";

      this.cedula = "";

      this.facultad = "";

      this.carrera = "";

      this.semestre = "";

      this.pararlelo = "";

      this.periodo = "";

      this.laboratorio = new Laboratorio();

      this.usuario = new Usuario();

      this.estado = "ACTIVO";

   }  public cPrestamo(String detalle, Date fecha, String cod_estudiante, String nombres_estudiante, String apellidos_estudiante, String cedula, String facultad, String carrera, String semestre, String pararlelo, String periodo, Laboratorio laboratorio, Usuario usuario, String estado) {

      this.detalle = detalle;

      this.fecha = fecha;

      this.cod_estudiante = cod_estudiante;

      this.nombres_estudiante = nombres_estudiante;

      this.apellidos_estudiante = apellidos_estudiante;

      this.cedula = cedula;

      this.facultad = facultad;

      this.carrera = carrera;

      this.semestre = semestre;

      this.pararlelo = pararlelo;

      this.periodo = periodo;

      this.laboratorio = laboratorio;

      this.usuario = usuario;

      this.estado = estado;
```

```
    }     public int getCod_prestamo() {
        return cod_prestamo;
    }     public void setCod_prestamo(int cod_prestamo) {
        this.cod_prestamo = cod_prestamo;
    }    public String getDetalle() {
        return detalle;
    }    public void setDetalle(String detalle) {
        this.detalle = detalle;
    }    public Date getFecha() {
        return fecha;
    }    public void setFecha(Date fecha) {
        this.fecha = fecha;
    }    public String getCod_estudiante() {
        return cod_estudiante;
    }    public void setCod_estudiante(String cod_estudiante) {
        this.cod_estudiante = cod_estudiante;
    }    public String getNombres_estudiante() {
        return nombres_estudiante;
    }    public void setNombres_estudiante(String nombres_estudiante) {    this.nombres_estudiante = nombres_estudiante;
    }    public String getApellidos_estudiante() {
        return apellidos_estudiante;
    }    public void setApellidos_estudiante(String apellidos_estudiante) {
this.apellidos_estudiante = apellidos_estudiante;
    }    public String getCedula() {
        return cedula;
    }    public void setCedula(String cedula) {
this.cedula = cedula;
    }    public String getFacultad() {
        return facultad;
    }    public void setFacultad(String facultad) {
        this.facultad = facultad;
    }    public String getCarrera() {
        return carrera;
    }    public void setCarrera(String carrera) {
        this.carrera = carrera;
    }    public String getSemestre() {
        return semestre;
    }    public void setSemestre(String semestre) {
        this.semestre = semestre;
    }    public String getPararlelo() {
        return pararlelo;
    }    public void setPararlelo(String pararlelo) {
        this.pararlelo = pararlelo;
    }    public String getPeriodo() {
        return periodo;
    }    public void setPeriodo(String periodo) {
        this.periodo = periodo;
    }    public Laboratorio getLaboratorio() {
        return laboratorio;
    }    public void setLaboratorio(Laboratorio laboratorio) {    this.laboratorio = laboratorio;
    }    public Usuario getUsuario() {
        return usuario;
    }    public void setUsuario(Usuario usuario) {
        this.usuario = usuario;
    }    public String getEstado() {
        return estado;
    }    public void setEstado(String estado) {
        this.estado = estado;
    }}
```

Anexo 10: GestionLabUnach-ejb , Modelo Préstamo

```
package com.gestionLab.Modelos;

import com.gestionLab.Entidades.Prestamo;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import javax.ejb.Stateless;

import javax.persistence.EntityManager;

import javax.persistence.PersistenceContext;

@Stateless

public class PrestamoFacade extends AbstractFacade<Prestamo> {

    @PersistenceContext(unitName = "GestionLabUnach-ejbPU")

    private EntityManager em;

    @Override

    protected EntityManager getEntityManager() {

        return em;

    }

    public PrestamoFacade() {

        super(Prestamo.class);

    }

        public List<Prestamo> listarPrestamoUsuario(int cod_usuario) {

        System.out.println("coigo usuario: " + cod_usuario);

        List<Prestamo> lista;

        Map<String, Object> parameters = new HashMap<String, Object>();

        parameters.put("cod_usuario", cod_usuario);

        lista = findAllResults("Prestamo.ByUser", parameters);

        return lista;

    }}
```

Anexo 11: Certificado

UNIVERSIDAD NACIONAL DE CHIMBORAZO

CERTIFICADO PARA TRAMITE DE FIN DE CARRERA

El (la) señor (a): **Moyón Aulla Angel Estuardo**

Cédula de Identidad No: **0603794231**

Alumno (a) de la Facultad de: **Ingenieria**

Carrera: **Sistemas y Computación**

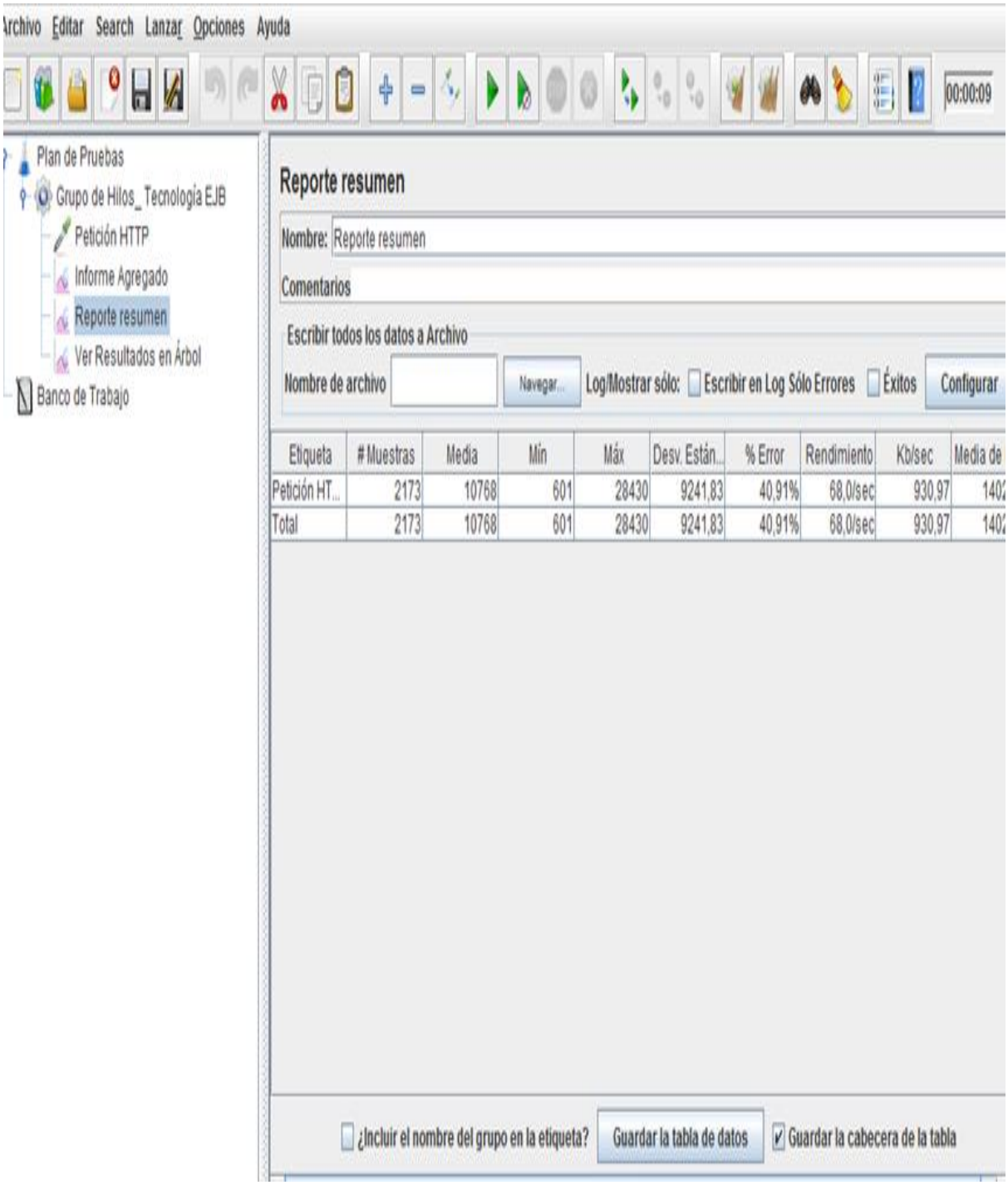
No adeuda materiales, componentes o reparación en los equipos de los Laboratorios, de la Universidad Nacional de Chimborazo

La presente certificación, tiene validez de 30 días calendario a partir de su emision

Fecha generada: sábado 10 septiembre 2016

RIOBAMBA-ECUADOR

Anexo 12: Pruebas JMeter-EJB



Anexo 13: Prueba JMeter-Programación tradicional