



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

“Trabajo de grado previo a la obtención del  
Título de Ingeniero en Sistemas y Computación.”

**TRABAJO DE GRADUACIÓN**

**Título del Proyecto**

“ANÁLISIS COMPARATIVO DE LA PRODUCTIVIDAD EN EL  
DESARROLLO DE APLICACIONES WEB UTILIZANDO LAS  
TECNOLOGÍAS JDBC Y JPA. CASO APLICATIVO: SISTEMA DE GESTIÓN  
DE EVENTOS DE LA UNIVERSIDAD NACIONAL DE CHIMBORAZO”.

**AUTOR (ES):**

Tatiana Elizabeth Molina Atiencia

Cesar Eduardo Mantilla Rivera

**Director:** Ing. Diego Palacios

Riobamba – Ecuador

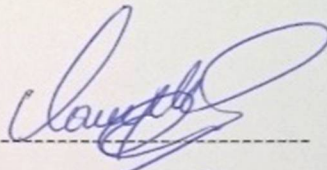
2016

Los miembros del Tribunal de Graduación del proyecto de investigación de título: **“ANÁLISIS COMPARATIVO DE LA PRODUCTIVIDAD EN EL DESARROLLO DE APLICACIONES WEB UTILIZANDO LAS TECNOLOGÍAS JDBC Y JPA. CASO APLICATIVO: SISTEMA DE GESTIÓN DE EVENTOS DE LA UNIVERSIDAD NACIONAL DE CHIMBORAZO”**, presentado por: Tatiana Elizabeth Molina Atiencia y Cesar Eduardo Mantilla Rivera, dirigido por el Ing. Diego Palacios.

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

Para constancia de lo expuesto firman:

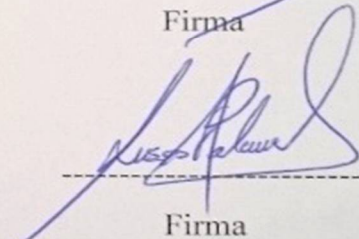
Ing. Danny Velasco  
Presidente del Tribunal Firma



---

Firma

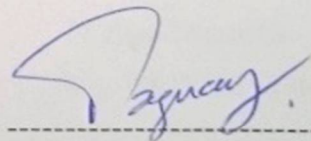
Ing. Diego Palacios  
Director del Tribunal Firma



---

Firma

Ing. Paul Paguay  
Miembro del Tribunal Firma



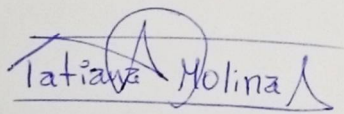
---

Firma

## AUTORÍA DE LA INVESTIGACIÓN

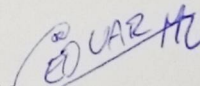
La responsabilidad del contenido de este Proyecto de Graduación, corresponde exclusivamente a: Tatiana Molina y Cesar Mantilla, autores del proyecto de investigación, al Ing. Diego Palacios, Director de Tesis; y el patrimonio intelectual de la misma a la Universidad Nacional de Chimborazo.

Riobamba, 10 de Marzo del 2016.



Tatiana Molina

C.I. 060341028-3



César Mantilla

C.I. 060375161-1

## **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.

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. Paul Paguay** por su paciencia y ayuda incondicional.

## **DEDICATORIA**

Dedicamos este estudio de tesis a nuestros hijos quienes son nuestro pilar fundamental y el motor de lucha para seguir adelante, a nuestros padres, 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.

Tatiana Molina y Eduardo Mantilla

## ÍNDICE DE ABREVIATURAS

UNACH: Universidad Nacional de Chimborazo  
ICITS: Instituto de Ciencia, Innovación, Tecnología y Saberes  
IBM: International Business Machines Corp.  
API: Interfaz de programación de aplicaciones.  
XML: Lenguaje de Etiquetado Extensible.  
HTML: Lenguaje de Marcas de Hipertexto.  
CSS: Cascading Style Sheets.  
API: Application Programming Interface  
JPA: Java Persistence API  
JDBC: Java Data Base Connection  
PHP: Hypertext Preprocessor.  
TICs: Tecnologías de la Información y la Comunicación.  
ID: identificación única.  
API\_KEY: Código de Acceso para una cuenta.  
URL: Uniform Resource Locator.  
JAVA SE: Java Standard Edition.  
JAVA EE: Java Enterprise Edition.  
POJO: Plain Old Java Object  
EJB: Enterprise Java Bean  
JPQL: Java Persistence Query Language  
JDO: Java Data Object  
JVM: Java Virtual Machine  
RUP: Rational Unificate Process

## ÍNDICE GENERAL

<b>CAPITULO I</b> .....	<b>2</b>
<b>MARCO REFERENCIAL</b> .....	<b>2</b>
<b>1.1. TÍTULO DEL PROYECTO</b> .....	<b>2</b>
<b>1.2. PROBLEMATIZACIÓN</b> .....	<b>2</b>
1.2.1. IDENTIFICACIÓN Y DESCRIPCIÓN DEL PROBLEMA.....	3
1.2.2. ANÁLISIS CRÍTICO .....	4
1.2.3. PROGNOSIS.....	5
1.2.4. DELIMITACIÓN .....	5
1.2.5. FORMULACIÓN DEL PROBLEMA.....	6
1.2.6. HIPÓTESIS.....	6
1.2.7. IDENTIFICACIÓN DE VARIABLES .....	6
<b>1.3. OBJETIVOS</b> .....	<b>6</b>
1.3.1. OBJETIVO GENERAL.....	6
1.3.2. OBJETIVOS ESPECÍFICOS .....	6
<b>CAPÍTULO II</b> .....	<b>7</b>
<b>FUNDAMENTACIÓN TEÓRICA</b> .....	<b>7</b>
<b>2.1. DEFINICIÓN DE PRODUCTIVIDAD</b> .....	<b>7</b>
2.1.1. LAS APLICACIONES WEB.....	8
2.1.2. METODOLOGÍA DE DESARROLLO DE APLICACIONES WEB.....	9
2.1.3. PRODUCTIVIDAD EN EL DESARROLLO DE APLICACIONES WEB .....	10
<b>2.2 JAVA</b> .....	<b>11</b>
2.2.1. PERSISTENCIA DE OBJETOS.....	11
2.2.2. MÉTODOS DE ALMACENAMIENTO DE OBJETOS PERSISTENTES .....	12

2.2.3 LAS BASES DE DATOS ORIENTADAS A OBJETOS .....	13
2.2.4. LAS BASES DE DATOS RELACIONALES .....	13
<b>2.3. MAPEO OBJETO RELACIONAL.....</b>	<b>15</b>
2.3.1. COMPONENTES.....	16
2.3.2. MAPEO DE OBJETOS EN LOS SISTEMAS MANEJADORES DE BASE DE DATOS RELACIONALES .....	17
2.3.3. HERRAMIENTAS ORM .....	20
<b>2.4. JPA (JAVA PERSISTENCE API) .....</b>	<b>21</b>
2.4.1. ARQUITECTURA DE JPA.....	23
2.4.2. MAPEO OBJETO RELACIONAL CON JPA .....	26
2.4.3 LAS ENTIDADES DE JPA.....	26
2.4.4. LENGUAJE DE CONSULTA PARA JPA .....	27
2.4.5. TRANSACCIONALIDAD .....	28
2.4.6. VENTAJAS DE JPA .....	29
2.4.7. DESVENTAJAS DE JPA.....	29
<b>2.5. JDBC (JAVA DATABASE CONNECTIVITY) .....</b>	<b>30</b>
2.5.1. ARQUITECTURA DE JDBC.....	31
2.5.2. JDBC COMPONENTE INDISPENSABLE PARA LOS ORM.....	33
2.5.3. CONTROLADORES JDBC .....	34
2.5.4. CONEXIÓN A LA BASE DE DATOS.....	37
2.5.5. ACCESO A LA BASE DE DATOS .....	38
2.5.6. LENGUAJE DE CONSULTA PARA JDBC .....	39
2.5.7. TRANSACCIONALIDAD .....	40
2.5.8. VENTAJAS DE JDBC .....	40
2.5.9. DESVENTAJAS DE JDBC .....	41
<b>2.6. METODOLOGÍA RUP .....</b>	<b>41</b>



<b>CAPITULO III .....</b>	<b>42</b>
<b>ANÁLISIS COMPARATIVO DE LA PRODUCTIVIDAD UTILIZANDO JPA Y JDBC .....</b>	<b>42</b>
<b>3.1. NIVEL DE CONOCIMIENTO DE LOS INVESTIGADORES EN LAS TECNOLOGÍAS ANALIZADAS .....</b>	<b>42</b>
<b>3.2. METODOLOGÍA DE INVESTIGACIÓN PARA LA EVALUACIÓN DE LA PRODUCTIVIDAD.....</b>	<b>42</b>
3.2.1. CONSTRUCCIÓN DE LOS PROTOTIPOS.....	43
3.2.2. PARÁMETROS DE EVALUACIÓN DE PRODUCTIVIDAD. ....	44
3.2.3. UTILIZACIÓN DE ESTADÍSTICA DESCRIPTIVA.....	47
3.2.4. GENERACIÓN DE LOS RESULTADOS .....	47
<b>3.3. OBTENCIÓN DE LOS RESULTADOS .....</b>	<b>50</b>
3.3.1. NÚMERO DE LÍNEAS DE CÓDIGO.....	50
3.3.2. NÚMERO DE HORAS DE PROGRAMACIÓN EMPLEADAS.....	51
3.3.3. NÚMERO DE FUNCIONES EN EL SISTEMA MANEJADOR DE BASES DE DATOS .....	52
3.3.4 NÚMERO DE LÍNEAS DE CÓDIGO PROGRAMADAS MANUALMENTE.....	53
<b>3.4. ANÁLISIS E INTERPRETACIÓN DE LOS RESULTADOS DEL ESTUDIO .....</b>	<b>54</b>
3.4.1. HIPÓTESIS A DEMOSTRAR .....	54
3.4.2. CONTRASTE DE LOS RESULTADOS OBTENIDOS DE LOS ANÁLISIS INDIVIDUALES DE LOS 4 PARÁMETROS MEDIDOS. ....	54
3.4.3. RESUMEN DE LOS RESULTADOS POR PARÁMETROS .....	56
3.4.4. APLICACIÓN DE LAS ENCUESTAS .....	61
3.4.5. RECOLECCION DE DATOS DE LAS ENCUESTAS REALIZADAS	61
3.4.6. DEMOSTRACIÓN DE LA HIPÓTESIS .....	66

<b>CAPÍTULO IV.....</b>	<b>68</b>
<b>IMPLEMENTACIÓN DE CEU - SISTEMA DE GESTIÓN DE EVENTOS DE LA UNIVERSIDAD NACIONAL DE CHIMBORAZO.....</b>	<b>68</b>
<b>4.1 ESTUDIO DE VIABILIDAD .....</b>	<b>68</b>
4.1.1. ANTECEDENTES .....	68
4.1.2. DESCRIPCIÓN DEL PROBLEMA.....	68
4.1.3. REQUERIMIENTOS DEL SISTEMA .....	69
4.1.4. PLAN DE DESARROLLO.....	70
<b>4.2. ANÁLISIS .....</b>	<b>70</b>
4.2.1. PLANIFICACIÓN DEL PROYECTO.....	70
4.2.2. INTEGRANTES Y ROLES.....	70
4.2.3. PROTOTIPOS.....	71
4.2.7. FLUJOS DE TRABAJO .....	74
4.2.3. HERRAMIENTAS DE DESARROLLO .....	75
<b>4.3. DISEÑO.....</b>	<b>76</b>
4.3.1. BASE DE DATOS .....	76
4.2.2. DICCIONARIO DE DATOS.....	77
4.2.3. PROTOTIPOS INTERFACES DE USUARIO FINALES.....	82
4.2.5. CÓDIGO FUENTE .....	86
<b>4.4. IMPLEMENTACIÓN.....</b>	<b>87</b>
4.4.1. INSTALACIÓN .....	87
5.4.2. FUNCIONALIDAD DEL SISTEMA .....	90
<b>4.5. PRUEBAS.....</b>	<b>99</b>
CONCLUSIONES .....	105
RECOMENDACIONES .....	106
BIBLIOGRAFÍA .....	107

## ÍNDICE DE ILUSTRACIONES

<b>Ilustración 1:</b> Ilustración del modelo relacional.....	14
<b>Ilustración 2:</b> Arquitectura JPA .....	21
<b>Ilustración 3:</b> Relación entre elementos de JPA.....	22
<b>Ilustración 4:</b> Contexto de persistencia .....	24
<b>Ilustración 5:</b> Unidad de persistencia en Netbeans .....	25
<b>Ilustración 6:</b> Interacción de Componentes .....	33
<b>Ilustración 7:</b> Formatos.....	36
<b>Ilustración 8:</b> Diagrama relacional de la base de datos del prototipo .....	43
<b>Ilustración 9:</b> Comparación del número de líneas de código utilizadas con JDBC Vs JPA.....	57
<b>Ilustración 10:</b> Cantidad de líneas de código utilizadas .....	57
<b>Ilustración 11:</b> Comparación del número de horas empleadas en el desarrollo ..	58
<b>Ilustración 12:</b> Cantidad de horas de programación empleadas.....	58
<b>Ilustración 13:</b> Comparación del número de las funciones en el SMBD .....	59
<b>Ilustración 14:</b> Cantidad de funciones utilizadas en el SMBD .....	59
<b>Ilustración 15:</b> Comparación del número de líneas de código programadas.....	60
<b>Ilustración 16:</b> Cantidad de líneas de código programadas manualmente .....	60
<b>Ilustración 17:</b> Nivel de conocimiento y experiencia en desarrollo web .....	61
<b>Ilustración 18:</b> Preferencia de tecnología en aplicaciones Java Web.....	62
<b>Ilustración 19:</b> Razones para utilizar las tecnologías antes mencionadas .....	62
<b>Ilustración 20:</b> Menor tiempo de desarrollo de aplicaciones .....	63
<b>Ilustración 21:</b> Menor cantidad de líneas de código.....	63
<b>Ilustración 22:</b> Mayor cantidad de documentación de ayuda .....	64
<b>Ilustración 23:</b> Menos consultas en el sistema de Gestión de Base de datos .....	64
<b>Ilustración 24:</b> Mayor compatibilidad en navegadores y sistemas operativos ....	65
<b>Ilustración 25:</b> Promedio de productividad medida .....	67
<b>Ilustración 26:</b> Prototipo Login .....	71
<b>Ilustración 27:</b> Prototipo Página Principal.....	72
<b>Ilustración 28:</b> Prototipo Administración de Usuarios y Seguridad .....	72
<b>Ilustración 29:</b> Prototipo Administración de Eventos .....	73

<b>Ilustración 30:</b> Prototipo del Módulo de Información.....	73
<b>Ilustración 31:</b> Prototipo del Módulo de Reportes .....	73
<b>Ilustración 32:</b> Tablas de la Base de Datos CEU .....	76
<b>Ilustración 33:</b> Control de Acceso a Usuarios.....	82
<b>Ilustración 34:</b> Creación del proyecto .....	87
<b>Ilustración 35:</b> Selección del tipo de proyecto .....	87
<b>Ilustración 36:</b> Configurando nombre del proyecto .....	88
<b>Ilustración 37:</b> Configuración del servidor .....	88
<b>Ilustración 38:</b> Con figuración del framework.....	89
<b>Ilustración 39:</b> Activación de los componentes .....	89
<b>Ilustración 40:</b> Resultado de la configuración del proyecto .....	90
<b>Ilustración 41:</b> Página principal de la aplicación .....	91
<b>Ilustración 42:</b> Sección de noticias.....	91
<b>Ilustración 43:</b> Página principal de eventos 1 .....	92
<b>Ilustración 44:</b> Página principal de eventos 2 .....	92
<b>Ilustración 45:</b> Página principal de un evento.....	93
<b>Ilustración 46:</b> Página de inicio de sesión.....	93
<b>Ilustración 47:</b> Página de inicio de usuarios .....	94
<b>Ilustración 48:</b> Página de administración de concursos.....	95
<b>Ilustración 49:</b> Página de conferencias .....	95
<b>Ilustración 50:</b> Página de eventos sugeridos .....	96
<b>Ilustración 51:</b> Página de generación de certificados .....	96
<b>Ilustración 52:</b> Administración de datos de los usuarios .....	97
<b>Ilustración 53:</b> Página de calificación del jurado .....	97
<b>Ilustración 54:</b> Página de Administración.....	98

## ÍNDICE DE TABLAS

<b>Tabla 1:</b> Parámetros de evaluación .....	44
<b>Tabla 2:</b> Descripción de las variables de la fórmula de productividad aplicada ..	49
<b>Tabla 3:</b> Número de líneas de código utilizadas en JDBC.....	50
<b>Tabla 4:</b> Número de líneas de código utilizadas en JPA.....	50
<b>Tabla 5:</b> Horas de programación empleadas en JDBC .....	51
<b>Tabla 6:</b> Horas de programación empleadas en JPA .....	51
<b>Tabla 7:</b> Funciones en el SMDB con JDBC.....	52
<b>Tabla 8:</b> Funciones en el SMDB con JPA.....	52
<b>Tabla 9:</b> Líneas de código programadas manualmente con JDBC.....	53
<b>Tabla 10:</b> Líneas de código programadas manualmente con JPA.....	53
<b>Tabla 11:</b> Resumen de resultados de la evaluación .....	54
<b>Tabla 12:</b> Comparación de los parámetros medidos de JDBC Vs JPA .....	56
<b>Tabla 13:</b> Promedio de la productividad medida.....	66
<b>Tabla 14:</b> Plan de Desarrollo .....	70
<b>Tabla 15:</b> Integrantes y Roles .....	71
<b>Tabla 16:</b> Definición del proceso de nuevo usuario .....	74
<b>Tabla 17:</b> Proceso de gestión del evento .....	74
<b>Tabla 18:</b> Herramientas utilizadas en el desarrollo .....	75
<b>Tabla 19:</b> Descripción de la tabla Archivo.....	77
<b>Tabla 20:</b> Descripción de la tabla Archivo_concurso.....	77
<b>Tabla 21:</b> Descripción de la tabla archivo_inscripción.....	77
<b>Tabla 22:</b> Descripción de la tabla Concurso.....	78
<b>Tabla 23:</b> Descripción de la tabla Conferencia .....	78
<b>Tabla 24:</b> Descripción de la tabla Evento.....	79
<b>Tabla 25:</b> Descripción de la tabla Ganador .....	79
<b>Tabla 26:</b> Descripción de la tabla Inscripción.....	80
<b>Tabla 27:</b> Descripción de la tabla Participante.....	80
<b>Tabla 28:</b> Descripción de la tabla Postulante .....	80
<b>Tabla 29:</b> Descripción de la tabla Rol.....	81
<b>Tabla 30:</b> Descripción de la tabla Rol_usuario .....	81

<b>Tabla 31:</b> Descripción de la tabla Unidad .....	81
<b>Tabla 32:</b> Descripción de la tabla Usuario .....	82
<b>Tabla 33:</b> Prototipo del Módulo de Administración a Usuarios y Seguridad .....	83
<b>Tabla 34:</b> Prototipo del Módulo de Gestión de Eventos.....	84
<b>Tabla 35:</b> Prototipo del Módulo de Información de Eventos.....	85
<b>Tabla 36:</b> Prototipo del Módulo de Reportes .....	86
<b>Tabla 37:</b> Prueba 1 .....	99
<b>Tabla 38:</b> Prueba 2 .....	100
<b>Tabla 39:</b> Prueba 3 .....	101
<b>Tabla 40:</b> Prueba 4.....	102
<b>Tabla 41:</b> Prueba 5 .....	103
<b>Tabla 42:</b> Prueba 6 .....	104

## **RESUMEN**

El presente trabajo investigativo tiene como objetivo realizar un análisis comparativo de la productividad en el desarrollo de Aplicaciones Web utilizando las tecnologías JDBC y JPA, para demostrar cuál de estas ofrece mayores beneficios en el desarrollo del Sistema de Control de Eventos para la Universidad Nacional de Chimborazo.

La investigación citada consta de IV capítulos, en el Capítulo I, inicia con un marco referencial para contextualizar el problema de investigación, el alcance que se le dará al mismo y los objetivos que se pretenden obtener con el estudio.

El Capítulo II es el sustento teórico de la investigación, en este se recopila y analiza la información de los frameworks de desarrollo estudiados, JPA y JDBC, para definir los parámetros que influyen en la productividad del desarrollo de Aplicaciones Web basadas en Java.

El Capítulo III describe la metodología utilizada y el estudio que se llevó a cabo para demostrar cuantitativamente cuál de las dos tecnologías es más productiva, obteniendo datos estadísticos de los parámetros definidos para la demostración de la hipótesis planteada.

El Capítulo IV se detalla la implementación del Sistema de Gestión de Eventos, donde se utilizará la tecnología JPA demostrando la productividad de la misma en el desarrollo de aplicaciones Web.

Se concluye la investigación comprobando que JPA es más productivo en el desarrollo de aplicaciones web en comparación con JDBC, ya que utiliza un menor número de líneas de código y optimiza el tiempo de desarrollo.

Es altamente recomendable la utilización de JPA cuando se pretende desarrollar un sistema con muchas clases y se posee poco tiempo para su desarrollo.



Lic. Maritza Larrea

09 de marzo de 2016

## SUMMARY

This research work has as aim make a comparative analysis of productivity in developing Web applications using JDBC and JPA technologies, to demonstrate which of these offers greater benefits in the development of System Event Control for the National University of Chimborazo.

The research consists on IV chapters, Chapter I, begins with a framework to contextualize the research problem, the scope given to it and the objectives to obtain with the study.

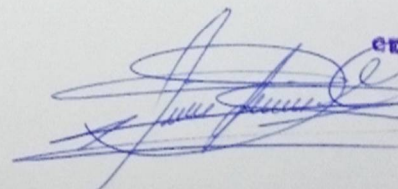
Chapter II is the research theoretical basis it collects and analyzes information development frameworks studied, JPA and JDBC, to define the parameters that influence the development productivity Web applications based on Java.

Chapter III describes the methodology and the study carried out to demonstrate which of the two technologies is more productive, obtaining statistical data of the parameters defined for the demonstration of the hypothesis.

Chapter IV, it shows the implementation Event Management System, where the JPA technology will used to demonstrate the productivity of the same in the development of Web applications is detailed.

The investigation concluded checking that JPA is more productive in the development of web applications compared to JDBC because it uses fewer lines of code and optimizes development time.

It is highly recommended the use of JPA when trying to develop a system with many classes and has little time to develop.

  
CENTRO DE IDIOMAS  
FACULTAD DE INGENIERIA  
UNIVERSIDAD NACIONAL DE CHIMBORAZO  
TAMAYO-ECUADOR  
COORDINACION



## INTRODUCCIÓN

Hoy en día la presencia de diferentes arquitecturas y tecnologías para el desarrollo de aplicaciones web, dificulta a los desarrolladores elegir una metodología adecuada. La constante evolución de la tecnología, el acceso a la información por medios informáticos inducen a escoger una metodología adecuada que permita resolver problemas.

Cada tecnología que se utilice para el desarrollo de Aplicaciones Web posee sus propias ventajas y desventajas. Algunas de ellas permiten generar aplicaciones web en tiempos records, optimizando tiempo y esfuerzo. Otras en cambio, están destinadas a que el desarrollador escriba todo el código necesario para que funcione, limitando la productividad del mismo.

Por esta razón es necesario que se elija una tecnología acorde a lo que se espera del producto final optimizando siempre la productividad en el ambiente de desarrollo.

El presente trabajo de investigación se enfoca al análisis de la tecnología JPA Y JDBC, que permitirá conocer los aspectos relacionados con el desarrollo de aplicaciones web basadas en Java, a la vez aportar con ideas, trabajos, sugerencias a través de una investigación completa acerca de la productividad utilizando dichas tecnologías.

# **CAPITULO I**

## **MARCO REFERENCIAL**

### **1.1. TÍTULO DEL PROYECTO**

Análisis comparativo de la productividad en el desarrollo de Aplicaciones Web utilizando las tecnologías JDBC y JPA.

Caso aplicativo: Sistema de Gestión de Eventos de la Universidad Nacional de Chimborazo.

### **1.2. PROBLEMATIZACIÓN**

Desarrollar una Aplicación Web es una tarea extensa y que, dependiendo del alcance del sistema, de los recursos que se utilizarán y del equipo de desarrolladores puede tornarse en una complicada labor. Cada fase del desarrollo debe ser planificada y ejecutada correctamente para que los resultados al momento del despliegue de la aplicación sean los esperados. En este ámbito hay muchas dudas que deben resolverse, ¿qué personas se involucrarán en el desarrollo?, ¿con qué presupuesto se cuenta?, ¿qué objetivos se pretenden alcanzar?, ¿cuánto tiempo hay disponible?, ¿qué tecnología se utilizará para el desarrollo?, etcétera. Precisamente estas dos últimas preguntas claves son en las que pretende enfocarse el estudio que se realizará debido a que el tiempo y la plataforma de desarrollo que se utilice para codificar la aplicación van de la mano. Algunas tecnologías permiten acelerar el desarrollo de la aplicación, sin embargo, pueden sacrificar la estabilidad o el rendimiento de la aplicación final y conseguir una aplicación que se haya desarrollado en menos pero que no ofrezca los resultados esperados en el nivel esperado. Por eso la pregunta a resolver es la siguiente:

¿Qué tecnología de desarrollo de aplicaciones web se debe elegir cuando se pretende priorizar la productividad?

### **1.2.1. IDENTIFICACIÓN Y DESCRIPCIÓN DEL PROBLEMA**

La definición de productividad de software más extendida se basa en el número de líneas de código fuente producidas por cada uno del personal de desarrollo de un proyecto en relación al tiempo de esfuerzo. Es decir, se evalúa la cantidad de código que un desarrollador ha generado para la aplicación.

Esta definición conlleva algunos otros interrogantes: ¿Se es más productivo mientras más código se escriba?, ¿Se debe priorizar la cantidad de código en una aplicación antes que el tiempo de desarrollo que tome generarlo?, ¿El obtener una aplicación estable, completa y funcional es parte de la productividad?, ¿Se puede optimizar el tiempo de desarrollo al máximo manteniendo la calidad del producto final?

Todos los programadores de software se encuentran con estas interrogantes a la hora de planificar el desarrollo de una nueva aplicación y tienen que encontrar las respuestas que más se adapten a la realidad de su emprendimiento.

Existen varias opciones cuando de hablar de productividad de desarrollo de software se trata. Se pueden citar algunos ejemplos:

- Hibernate
- JPA (Java Persistence API)
- JDBC (Java Database Connectivity)
- Oracle TopLink
- MyBatis
- GORM (Graphical Object Relationship Modeller)
- EclipseLink

Cada una de las tecnologías anteriormente descritas posee sus propias ventajas y desventajas. Algunas de ellas permiten generar aplicaciones web en tiempos records, sin embargo, personalizarlas, adaptarlas al entorno, optimizar el rendimiento o desplegarlas, pueden generar en muchos casos todo el trabajo que se ahorró en el desarrollo. Otras en cambio, están destinadas a que el desarrollador escriba todo el código necesario para que funcione, limitando la productividad del mismo.

Por esta razón es imperativo que se elija una tecnología acorde a lo que se espera del producto final optimizando siempre la productividad en el ambiente de desarrollo.

En la actualidad, en la Universidad Nacional de Chimborazo se realizan eventos permanentemente de carácter científico, social o deportivo. Esta institución carece de un medio mediante el cual se pueda gestionar e informar de estos eventos realizados y a realizarse dentro de la Universidad, esto se debe a que la UNACH adolece la falta de un Sistema de Gestión de Eventos para interactuar teniendo una información completa acerca de los eventos a realizarse y de esta manera poder asistir a los mismos.

La web constituye un medio fundamental para que la escuela pueda permanecer en constante información y futuros cambios en donde se pueda publicar los Eventos, tanto para los estudiantes, docentes y administrativos de la Universidad.

### **1.2.2. ANÁLISIS CRÍTICO**

La falta de conocimiento dentro del personal de la Universidad, tanto docentes, estudiantes y administrativos acerca de los eventos a realizarse dentro de ésta, genera un problema para la Universidad porque no se cuenta un Sistema de Gestión de Eventos que nos permita estar informados de los Eventos Académicos y los puntos tales como:

- Personal al cual está dirigido el Evento Académico
- Tipo de Evento a realizarse tales como:
  - Concursos
  - Congresos
  - Ponencias
  - Lugar y disponibilidad del mismo para el evento
  - Fecha y hora del evento a realizarse
  - Tema del evento a realizarse
  - Inscripciones del Evento
  - Resultados del Evento en el caso de Concursos
  - 
  -

### **1.2.3. PROGNOSIS**

El presente trabajo investigativo pretende convertirse en un referente que permitirá conocer los aspectos relacionados con el desarrollo de aplicaciones web basadas en Java utilizando tecnologías JPA y JDBC; a la vez aportar con ideas, trabajos, sugerencias a través de una investigación completa acerca de la productividad utilizando dichas tecnologías en aplicaciones web. Además, se aplicará la tecnología que se determina más adecuada para suplir la necesidad de un Sistema que nos permita gestionar los eventos de la Universidad Nacional de Chimborazo.

### **1.2.4. DELIMITACIÓN**

Este proyecto evaluará la productividad que ofrecen las tecnologías JPA y JDBC en el desarrollo de aplicaciones web para determinar cuál es la más adecuada según indicadores posteriormente descritos; para Implementar en la Universidad Nacional de Chimborazo un Sistema de Gestión de Eventos que será el inicio de la comunicación entre los estudiantes, docentes y administrativos de la Universidad organizando y participando de los Eventos que se desarrollen dentro de la misma. El sistema se delimitará en Gestionar los Eventos de carácter académico y contará básicamente con los siguientes módulos:

- **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 ADMINISTRACIÓN DE EVENTOS**

Permitirá administrar la información de cada uno de los eventos de diferente índole que se realicen en la UNACH, informando a los usuarios de eventos próximos y permitiendo la participación de éstos en dichos eventos.

- **MÓDULO DE INFORMACIÓN**

Difusión y Promoción del Evento a realizarse

- **MÓDULO DE REPORTES**

Permitirá la visualización de los resultados en el caso de Concursos y además permitirá la impresión de Certificados

### **1.2.5. FORMULACIÓN DEL PROBLEMA**

¿Cuál de las dos tecnologías más utilizadas en el desarrollo de aplicaciones web, JPA y JDBC ofrece mayores beneficios en función de la productividad?

### **1.2.6. HIPÓTESIS**

La tecnología JPA es la más adecuada con respecto a la productividad para el desarrollo de aplicaciones Web con Java en comparación que JDBC.

### **1.2.7. IDENTIFICACIÓN DE VARIABLES**

- VARIABLE DEPENDIENTE

Productividad en el desarrollo de aplicaciones web

- VARIABLE INDEPENDIENTE

La tecnología JPA y JDBC.

## **1.3.OBJETIVOS**

### **1.3.1. OBJETIVO GENERAL**

Realizar un análisis comparativo de la productividad en el desarrollo de Aplicaciones Web utilizando las tecnologías JDBC y JPA.

### **1.3.2. OBJETIVOS ESPECÍFICOS**

- Analizar el escenario del problema en cuanto a la Gestión de Eventos de la UNACH.
- 
- Comparar las tecnologías JDBC y JPA con respecto a la productividad para demostrar cuál de estas ofrece mejores resultados.
- 
- Implementar el Sistema de Gestión de Eventos para la UNACH.

## **CAPÍTULO II**

### **FUNDAMENTACIÓN TEÓRICA**

#### **2.1. DEFINICIÓN DE PRODUCTIVIDAD**

Según (Tangen, 2005), la definición de productividad varía dependiendo del enfoque que se le brinde al término, entonces, este enfoque puede ser:

- En tecnología: relación entre ratios de salidas y entradas utilizadas
- En ingeniería: la relación entre la salida actual y la potencial de un proceso
- En economía: la eficiencia de la asignación de recursos.

Según (Low & Jeffery, 1990), en el ámbito de la informática se denota gran importancia en la cantidad de líneas de código que posee una aplicación para definir su productividad, según la norma IEEE 1045-1992 la productividad es la relación de una primitiva de salida (líneas de código, puntos función o documentos) y su correspondiente primitiva de entrada (esfuerzo, tiempo) para desarrollar software.

La norma ISO 9126-4 mide la productividad basando el producto obtenido con los tipos de recursos empleados:

- Productividad Humana (Efectividad/Esfuerzo)
- Productividad Temporal (Efectividad/Tiempo)
- Productividad Económica (Efectividad/Coste)

Según (Payá Martín, 2016), la productividad en IS es comúnmente medida utilizando una medida tecnológica basada en un ratio entre el tamaño del producto desarrollado y el esfuerzo requerido para producirlo (MacCormack, Kemerer, Cusumano, & Crandall, 2003), por ejemplo las líneas de código por unidad de tiempo (SLOC/t) (Maxwell, 1996) o alguna variante de puntos función por unidad de tiempo (PF/t) (Low & Jeffery, 1990). En esta línea, la norma IEEE 1045-1992 define la productividad como la relación de una primitiva de salida (líneas de

código, puntos función o documentos) y su correspondiente primitiva de entrada (esfuerzo, tiempo) para desarrollar software. Por otro lado, la norma ISO 9126-4 define la productividad basándose en factores de calidad como la capacidad del producto software para permitir a los usuarios emplear cantidades de recursos adecuados en relación con la efectividad alcanzada en un contexto de uso específico. Esta norma está centrada en la calidad, el usuario final y el contexto de uso, por lo que la definición de la productividad gira en torno a estos tres conceptos. En resumen, el número de líneas de código para materializar la funcionalidad necesaria, se configura como un indicador del esfuerzo de desarrollo que permite la comparación entre cada una de las alternativas consideradas. Menos líneas de código, significa menor tiempo de desarrollo, mayor facilidad de mantenimiento, menos costes de desarrollo y consecuentemente mayor productividad.

### **2.1.1. LAS APLICACIONES WEB**

Según (Lujan Mora, 2002), un sitio web es un conjunto de páginas web relacionadas entre sí. Se entiende por página web tanto el fichero que contiene el código HTML como todos los recursos que se emplean en la página (imágenes, sonidos, código JavaScript, entre otros).

En todo sitio web se suelen distinguir dos páginas especiales: la página inicial (o página de entrada) y la página principal (o página menú). La página inicial, conocida como splash page en inglés, es la primera página que un usuario ve al visitar un sitio web. Normalmente, la página inicial se emplea para promocionar la compañía u organización a la que pertenece el sitio web, o para dar a conocer un producto o servicio particular (por ejemplo, para promocionar unos productos en oferta). También se suele emplear para informar al usuario de los requisitos (tipo y versión de navegador, resolución mínima) necesarios para visualizar correctamente el resto de páginas del sitio web. A menudo, la página inicial es la más vistosa del sitio web, ya que tiene el objetivo de atraer y atrapar al visitante. La mayoría de las páginas iniciales poseen las siguientes características:

- Poco texto, pero muchas imágenes, gráficos animados, sonidos o incluso vídeos.



- Algunas pasan automáticamente a la página principal, pero en otras el usuario tiene que pulsar en un enlace para cargar la página principal.

En algunos casos la página inicial se convierte en un tunel de entrada: una presentación que dura bastante tiempo (más de 15 segundos), que suele estar realizada con múltiples páginas o con una sola página que emplea tecnología multimedia (como Macromedia Flash). En estos casos, suele existir un enlace para evitar el tunel de entrada y saltar directamente a la página principal.

Además de usarse como tarjeta de presentación, la página inicial también se puede emplear para disminuir el tiempo necesario para cargar las páginas posteriores.

Mientras el usuario está visualizando la página inicial, las imágenes que se emplearán en las siguientes páginas se pueden cargar en la memoria caché del navegador mediante un proceso en segundo plano del que el usuario no es consciente.

### **2.1.2. METODOLOGÍA DE DESARROLLO DE APLICACIONES WEB**

Según (Lujan Mora, 2002), No existe en la actualidad una metodología de desarrollo de sitios web ampliamente aceptada. Sin embargo, una posible metodología es la que se presenta a continuación.

Algunas de las fases de esta metodología se pueden realizar en paralelo o no acabar hasta el final del desarrollo del sitio web:

- Se estudian los requisitos y especificaciones del sitio web: cuál es el contenido del sitio web, qué se pretende conseguir, a quién se destina y número de visitas previsto, qué inversión se desea realizar, de cuánto tiempo se dispone.
- A partir de los requisitos se decide la arquitectura y tecnología del sitio web: empleo de un servidor web propio o alojamiento (hospedaje) en un servidor alquilado, ancho de banda de la comunicación del servidor web con Internet, páginas estáticas o tecnología de generación dinámica de páginas como (ASP, CGI, entre otras.), además datos almacenados en ficheros o en un servidor de bases de datos.
- A continuación se diseña la estructura lógica o de navegación del sitio web: página inicial, página principal, empleo de marcos, los menús,

división en secciones, relación entre las distintas secciones, página de novedades, entre otros.

- Se define la estructura física, que puede ser igual a la lógica o totalmente independiente.
- Se crean los contenidos del sitio web. Si se emplea una base de datos, se realiza la carga de datos.
- Se realiza el diseño gráfico y ergonómico: colores, montaje, tipografía, botones de navegación, logotipos y demás elementos gráficos, entre otros
- Se crean las páginas estáticas y los elementos multimedia.
- Desarrollo de los scripts y páginas dinámicas.
- Por último, se verifica el correcto funcionamiento del sitio web: se comprueba la conexión con la base de datos, se verifica que no existan enlaces rotos, se confirma que todos los recursos empleados (imágenes, ficheros con código script entre otros), se encuentran en el sitio web y en su lugar correspondiente. Además, se comprueba el sitio web con distintos navegadores para asegurar su compatibilidad. También se realizan pruebas de carga para evaluar el rendimiento.
- Puesta en marcha

### **2.1.3. PRODUCTIVIDAD EN EL DESARROLLO DE APLICACIONES WEB**

Según (Monteagudo, 2014), cuando tenemos que desarrollar una aplicación web tenemos que analizar muy bien la tecnología que vamos a utilizar y conocer si dicha tecnología nos va a permitir el desarrollo del proyecto de forma satisfactoria. Para desarrollar el proyecto exitosamente conviene que dicha tecnología nos proporcione ciertas características, tales como productividad, funcionalidad y escalabilidad, entre otras.

Hoy en día **JavaScript** es un lenguaje que nos proporciona un amplio abanico de posibilidades, y existen muchas herramientas dentro del entorno de JavaScript que nos permiten tener acceso a las características que previamente he mencionado y que nos ayudarán a que el desarrollo de nuestros proyectos sean exitosos.

## **2.2 JAVA**

Según (Saney, Primer Programa Java, 2001), Java es un lenguaje de programación. Este lenguaje está definido por una especificación, es decir, un documento que describe las funcionalidades y sintaxis del lenguaje. EL lenguaje, así descrito, es comprensible por in programador. Para poder escribir los programas, la especificación le basta al programador. Sin embargo, estos programas no podrán ser ejecutados si no existe un método de traducirlos a na versión comprensible por un ordenador. Para ello necesitaos una implementación de cierto número de utilidades que nos permitirán efectuar esta traducción y ejecutar los programas.

Una particularidad de Java es que este lenguaje utiliza una máquina virtual. El lenguaje comprensible para el programador no es traducido al lenguaje específico del procesador que debe ejecutar el programa, pero sí que es traducido le lenguaje de un procesador virtual, esta traducción es traducida por un compilador. El código resultante es llamado bytecode.

### **2.2.1. PERSISTENCIA DE OBJETOS**

Sehún (Bauer & King, 2004) , los métodos en Java se utilizan para realizar muchas funciones, entre ellas instanciar las clases o crear objetos, luego de que un determinado método cree o instancie un objeto, este debe ser útil incluso después de que dicho método haya terminado de cumplir con sus instrucciones. Si el objeto “persiste” o se mantiene vivo aún después del método, se dice que este objeto es persistente y se puede hablar de persistencia de datos.

El objeto debe seguir existiendo para que pueda ser utilizado en una determinada acción por un programador, es decir, pueda ser consultado, eliminado, modificado y transferido.

“La persistencia de objetos significa que los objetos individuales pueden sobrevivir al proceso de la aplicación; pueden ser guardados a un almacén de datos y ser reconstruidos más tarde.

La persistencia implica que el objeto pueda ser accedido desde un lugar donde haya sido guardado.

## 2.2.2. MÉTODOS DE ALMACENAMIENTO DE OBJETOS PERSISTENTES

Según (Sperko, 2003) , un objeto debe ser almacenado en algún lugar para poder ser accedido posteriormente al método que lo instancia. Para esto, se necesita un “almacén de datos” que permita guardar el objeto y accederlo. Se han detallado a continuación, las herramientas que proveen esa característica al objeto.

- La serialización de objetos en java

La serialización es una técnica no propia de Java, se utiliza en la mayoría de los lenguajes de programación. También se conoce como marshalling en inglés y busca que el objeto creado a partir de una clase se convierta en un flujo de bits con la finalidad de almacenarlo en algún tipo de memoria, ya sea una base de datos o un archivo xml, para poder acceder y manejarlo como el programador determine.

Una vez que el objeto ha sido serializado no sólo guarda al objeto en sí, sino también una secuencia de bits que le sirven al objeto como metadato<sup>1</sup>, en el que además del objeto mismo se almacena el estado para que este pueda ser recordado.

Ventajas de la serialización

- Es más fácil de implementar.
- Es más rápido el acceso a los datos.
- No se necesita implementar un complejo sistema manejador de base de datos.

Desventajas de la serialización

- No soporta que los datos sean accedidos simultáneamente desde varios usuarios.
- Se utilizan para bancos de datos relativamente pequeños.
- Existen soluciones actuales de sistemas manejadores de bases de datos muy completos.
- Los sistemas manejadores de bases de datos son ampliamente más utilizados.

---

<sup>1</sup> **Metadato:** Información que se asocia a un dato para describirlo.

### **2.2.3 LAS BASES DE DATOS ORIENTADAS A OBJETOS**

Según (Bauer & King, 2004), para hablar de bases de datos orientadas a objetos es necesario que primero se entienda lo que es la persistencia transparente.

La persistencia transparente permite a un programador acceder directamente a los datos de la base de datos, sin la necesidad de que estos datos sean traducidos o convertidos en algo más para poder ser utilizados ni la implementación de un lenguaje de consulta de datos.

Entonces las bases de datos orientadas a objetos proporcionan métodos para que esta persistencia transparente de la que se habla sea posible y se puedan manejar directamente los objetos.

- Ventajas de las bases de datos orientadas a objetos

Los objetos se pueden manejar directamente sin implementar lenguajes de consulta. En estas bases de datos, los objetos a los que hacen referencia los objetos persistentes también alcanzan la persistencia.

- Desventajas de las bases de datos orientadas a objetos

Los sistemas manejadores de bases de datos orientados a objetos son escasos.

Las bases de datos relacionales están más extendidas lo que implica que existe más soporte e información sobre ellas.

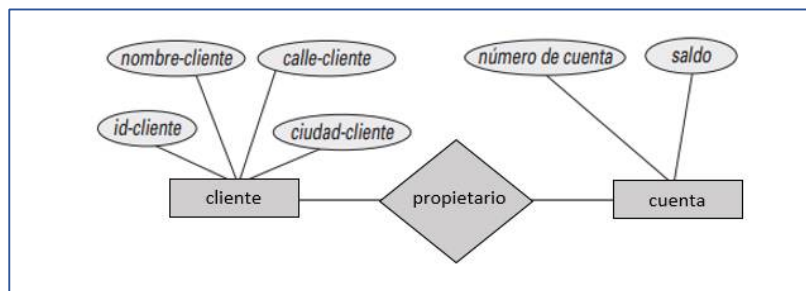
### **2.2.4. LAS BASES DE DATOS RELACIONALES**

Según (Keith & Schincariol, 2006) estas bases de datos implementan un modelo relacional para su funcionamiento, a continuación, más adelante se define lo que esto significa.

El modelo relacional es aquel que modela la información basándose en relaciones definidas en conjuntos finitos de valores llamados dominios.

Los modelos relacionales se basan en la forma en la que se relacionan entre sí las instancias que contiene la base de datos. Entonces, se necesita entender claramente lo que es una “relación”.

- Relación en una base de datos



**Ilustración 1:** Ilustración del modelo relacional  
**Fuente:** (Keith & Schincariol, 2006)

Según (Silberschatz, Korth, & Sudarshan, 2002), una de las formas más sencillas de definir una relación es como una asociación entre varias entidades. Es decir, la descripción de algo que une o asocia a las entidades entre sí es llamada relación. En la Ilustración 1 se puede apreciar dos entidades Cliente y Cuenta que están asociadas con la relación propietario. Esto significa que un cliente puede ser propietario de una cuenta, o viceversa. Sin embargo, para definir el número de cuentas a las que un cliente puede ser un propietario o el número de propietarios que puede tener una misma cuenta, se debe colocar un número máximo en la relación por cada una de las entidades que la componen, este “número” se llama restricción de la relación y es una correspondencia de cardinales que indica el número de entidades con las que cada entidad se puede relacionar. Estas restricciones pueden ser de varios tipos:

- Uno a uno
- Uno a varios
- Varios a uno
- Varios a varios
- Restricción uno a uno

Este tipo de restricción indica que cada instancia u objeto de cada entidad involucrada se relaciona con una sola instancia u objeto de la otra entidad.

Tomando como ejemplo la ilustración 1 se tendría que un cliente puede ser propietario de una sola cuenta y que una cuenta puede ser tener un solo propietario. En la práctica este tipo de relaciones son poco comunes de implementar.

- Restricción uno a varios

Este tipo de restricción indica que un objeto de la primera entidad se relaciona con uno o varios objetos de la segunda entidad, pero que la segunda entidad sólo se relaciona con un objeto de la primera.

Tomando como ejemplo la ilustración 1 se tendría que un cliente puede ser propietario de por lo menos una cuenta y que una cuenta puede tener un solo propietario.

- Restricción varios a uno

Este tipo de restricción indica que un objeto de la primera entidad se relaciona con un solo objeto de la segunda entidad, pero que un objeto de la segunda entidad se relaciona con uno o varios de la primera.

Tomando como ejemplo la ilustración 1 se tendría que un cliente puede ser propietario de solo de una cuenta y que una cuenta puede tener varios propietarios.

- Restricción varios a varios

En este tipo de relación cada objeto de cada entidad se relaciona con uno o varios objetos de la otra.

Tomando como ejemplo la ilustración 1 se tendría que un cliente puede ser propietario de varias cuentas y que una cuenta puede tener varios propietarios.

A pesar de que en las relaciones cada entidad se representa en la base de datos con una tabla, para la implementación de las restricciones varios a varios se deberán implementar una tabla adicional.

**@NamedNaviteQueries:** Especifica varias queries SQL.

### **2.3. MAPEO OBJETO RELACIONAL**

Según (Doroshenki y Romanenko, 2005), la mayor parte de las aplicaciones orientadas a objetos precisan de la implementación de mecanismos que permitan a los objetos mantenerse vivos tras la finalización del proceso que les dio vida. El problema de la persistencia de objetos Java ha dado lugar a utilizar alternativas que proporcionan al programador una manera sencilla, automática y transparente basada en bases de datos relacionales, de incluir esa funcionalidad en sus desarrollos denominada, ORM (Mapeo Objeto Relacional).

Según (Doroshenki y Romanenko, 2005), el mapeo objeto/relacional permite formar el mapeo de una base de datos relacional, generando en la aplicación orientada a objetos clases que son directamente la estructura de la base de datos que se posee. Es decir, en la aplicación se puede tener una base de datos orientada a objetos virtuales sobre la base de datos relacional. Esta característica permite aplicar conceptos de orientación a objetos como herencia y polimorfismo, a los datos almacenados de forma relacional.

En este capítulo discutiremos el concepto de ORM, componentes, principales requerimientos para un buen framework ORM y herramientas.

El mapeo Objeto/Relacional es “la persistencia automatizada y transparente de las tablas en una Base de Datos relacional, usando metadatos que definen el mapeo entre los objetos y la Base de Datos”.

En esencia, ORM transforma datos de una representación en otra.

Por tanto, ORM es una técnica que se utiliza para poder ligar las bases de datos y los conceptos de orientación a objetos creando “bases de datos virtuales”, es decir la aplicación desde dentro utiliza frameworks<sup>2</sup>, los mismos que son intermediarios entre la base de datos relacional y la aplicación totalmente orientada a objetos.

Se considera ORM a cualquier capa de persistencia que proporcione autogeneración de SQL a través de metadatos (generalmente XML). No se considera ORM a una capa de persistencia creada por el propio desarrollador.

### 2.3.1. COMPONENTES

Según (Bauer & King, 2004), una solución ORM está formada por cuatro partes:

- Una API que posibilite la realización de operaciones de creación, actualización y borrado sobre objetos de clases persistentes.
- Un lenguaje o API para poder especificar consultas sobre dichas clases.
- Una opción para especificar mapeo de metadatos.
- Una técnica para que la implementación del ORM pueda llevar a cabo búsquedas, asociaciones u otras funciones de optimización.

---

<sup>2</sup>**Framework:** Estructura de soporte definida, en la cual otro proyecto de software puede ser organizado y desarrollado. <sup>6</sup>**O/R:** Objeto Relacional.



### **2.3.2. MAPEO DE OBJETOS EN LOS SISTEMAS MANEJADORES DE BASE DE DATOS RELACIONALES**

Según (Sánchez, 2016), antes de hablar del mapeo de base de datos se debe aclarar lo que es un sistema manejador de base de datos relacionales. Estos sistemas deben cumplir con las 12 reglas de Codd (Edgar Frank Codd, 19 de agosto de 1923 - 18 de abril de 2003) para sistemas manejadores de bases de datos para que se puedan concebir como relacional. Las reglas se detallan a continuación:

Regla 0: El sistema debe ser relacional, base de datos y administrador de sistema. Ese sistema debe utilizar sus facilidades relacionales exclusivamente para manejar la base de datos.

Regla 1: La regla de la información, toda la información en la base de datos es representada unidireccionalmente, por valores en posiciones de las columnas dentro de filas de tablas.

Regla 2: la regla del acceso garantizado, todos los datos deben ser accesibles sin ambigüedad. Esta regla es esencialmente una nueva exposición del requisito fundamental para las llaves primarias. Dice que cada valor escalar individual en la base de datos debe ser lógicamente direccionable especificando el nombre de la tabla, la columna que lo contiene y la llave primaria.

Regla 3: Tratamiento sistemático de valores nulos, el sistema de gestión de base de datos debe permitir que haya campos nulos. Debe tener una representación de la "información que falta y de la información inaplicable" que es sistemática, distinto de todos los valores regulares.

Regla 4: catálogo dinámico en línea basado en el modelo relacional, el sistema debe soportar un catálogo en línea, el catálogo relacional debe ser accesible a los usuarios autorizados.

Regla 5: la regla comprensiva del sublenguaje de los datos, el sistema debe soportar por lo menos un lenguaje relacional que:

- Tenga una sintaxis lineal.
- Puede ser utilizado de manera interactiva.
- Soporte operaciones de definición de datos, operaciones de manipulación de datos (actualización, así como la recuperación), seguridad e integridad y operaciones de administración de transacciones.

Regla 6: regla de actualización, todas las vistas que son teóricamente actualizables deben ser actualizables por el sistema.

Regla 7: alto nivel de inserción, actualización y borrado, permitiendo el sistema realizar manipulación de datos de alto nivel, es decir, sobre conjuntos de tuplas. Esto significa que los datos no solo se pueden recuperar de una base de datos relacional de filas múltiples y/o de tablas múltiples, sino también pueden realizarse inserciones, actualización y borrados sobre varias tuplas y/o tablas al mismo tiempo (no sólo sobre registros individuales).

Regla 8: independencia física de los datos, los programas de aplicación y actividades del terminal permanecen inalterados a nivel lógico cuando quiera que se realicen cambios en las representaciones de almacenamiento o métodos de acceso.

Regla 9: independencia lógica de los datos, los cambios al nivel lógico (tablas, columnas, filas, etc.) no deben requerir un cambio a una solicitud basada en la estructura. La independencia de datos lógica es más difícil de lograr que la independencia física de datos.

Regla 10: independencia de la integridad, las limitaciones de la integridad se deben especificar por separado de los programas de la aplicación y se almacenan en la base de datos. Debe ser posible cambiar esas limitaciones sin afectar innecesariamente las aplicaciones existentes.

Regla 11: independencia de la distribución, la distribución de las porciones de la base de datos a las varias localizaciones debe ser invisible a los usuarios de la base de datos. Los usos existentes deben continuar funcionando con éxito:

Cuando una versión distribuida del SGBD se introdujo por primera vez

Cuando se distribuyen los datos existentes se redistribuyen en todo el sistema.

Regla 12: la regla de la no subversión, si el sistema proporciona una interfaz de bajo nivel de registro, a parte de una interfaz relacional, que esa interfaz de bajo nivel no se pueda utilizar para subvertir el sistema, por ejemplo: sin pasar por seguridad relacional o limitación de integridad. Esto es debido a que existen sistemas anteriormente no relacionales que añadieron una interfaz relacional, pero con la interfaz nativa existe la posibilidad de trabajar no relacionamente.

Según (Sánchez, 2016), es muy común encontrar aplicaciones orientadas a objetos manipulando datos en bases de datos relacionales, mediante el uso de técnicas de mapeo por metadatos.

Dentro de las consideraciones a tener en cuenta encontramos:

Un atributo o propiedad podría mapearse a cero o más columnas en una tabla de un RDBMS.

- Algunos atributos o propiedades de los objetos no son persistentes (calculados por la aplicación).
- Algunos atributos de un objeto son también objetos (Cliente --> Dirección) y esto refleja una asociación entre dos clases que deben tener sus propios atributos mapeados.
- En principio identificaremos dos tipos de metadatos de mapping: property mapping y relationship mapping.
- Property mapping: Mapping que describe la forma de persistir una propiedad de un objeto.
- Relationship mapping: Mapping que describe la forma de persistir una relación (asociación, agregación, o composición) entre dos o más objetos.
- El mapping más simple es un *property mapping* de un atributo simple a una columna de su mismo tipo.

Por otra parte, para soportar la conversión entre los dos modelos es necesario incorporar al modelo OO (orientado a objetos) lo que se conoce como información shadow. Esto es, cualquier dato que necesiten mantener los objetos para poder persistirse, como por ejemplo identificación de las propiedades que corresponden a la clave primaria en la base de datos, marcas que permitan el control de intentos de modificación concurrente de los datos en la base (timestamps o números de versión de los datos), atributos especiales que indiquen si el objeto ya fue persistido (para determinar si se usa INSERT o UPDATE, entre otros

Según el autor “Los metadatos que configuran el mapeo pueden llegar a ser muy complejos y es necesario comprender muy bien que se está configurando y las implicaciones de usar diferentes alternativas”

### **2.3.3. HERRAMIENTAS ORM**

Según (Deitel & Deitel, 2004), hay múltiples alternativas para los programadores de Java cuando se pretende trabajar con mapeadores O/R. Existen tres organizaciones o comunidades que están implicadas en el mundo de la persistencia O/R de Java de forma activa: organizaciones basadas en el estándar, comunidades open source y grupos comerciales.

Dependiendo de que tanto se apoyen en ORM, podemos clasificar las aplicaciones en las siguientes categorías:

**Relacional pura:** Toda la aplicación, incluyendo la interfaz de usuario, está diseñada en base al modelo relacional. Es una aproximación válida para aplicaciones simples donde no se va a reutilizar gran cantidad de código, aunque tiene serios contratiempos como la mantenibilidad o la portabilidad.

**Mapeo de objetos ligero:** En esta aproximación las entidades se representan como clases que después son mapeadas manualmente en las tablas relacionales. Se oculta el código de acceso a la Base de Datos con los patrones de diseño más utilizados. Es una aproximación con mucha aceptación, y muy válida cuando no tenemos muchas entidades.

**Mapeo de objetos medio:** La aplicación se diseña en base a un modelo de objetos. Las sentencias SQL se generan en tiempo de ejecución a través de un framework o generadores de código. Las asociaciones entre objetos son manejadas por el mecanismo de persistencia y es posible refinar el acceso a datos a través de expresiones en un lenguaje orientado a objetos. Este nivel es adecuado para aplicaciones de complejidad media, donde es importante la portabilidad hacia diferentes RDBMS. Usualmente no usan procedimientos almacenados.

**Mapeo de objetos completo:** Se soporta modelado de objetos complejo con composición, herencia, polimorfismo, entre otros. La capa de persistencia implementa persistencia transparente, esto es la capacidad de manipular los datos almacenados en la RDBMS directamente a través de objetos, sin tener que implementar alguna interfaz especial o heredar una clase. Se utilizan técnicas eficientes para la obtención y cacheo de información, las cuales son transparentes hacia la aplicación. Este nivel de funcionalidad es muy difícil de lograr a través de una solución in-house, ya que requiere meses o tal vez años de desarrollo.

## 2.4. JPA (JAVA PERSISTENCE API)

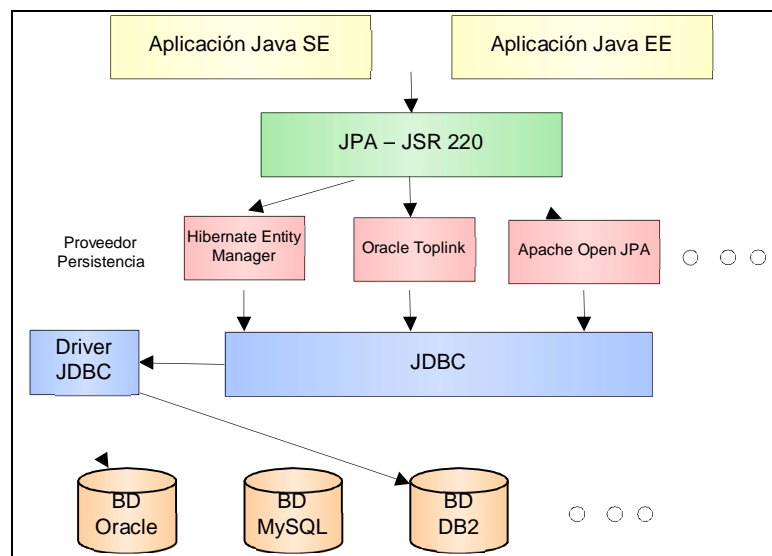
Según (Groussard, 2010), Java Persistence API, ampliamente conocida por sus siglas en inglés JPA, es una herramienta que provee a los programadores de Java de capacidades para realizar un mapeo objeto relacional para administrar datos relacionales de aplicaciones.

JPA ha sido ideada para utilizar todas las facilidades que provee la orientación a objetos al conectarse a un sistema manejador de base de datos para esto utiliza el patrón de mapeo objeto-relacional

La API de persistencia Java es el estándar de transformación objeto/relacional que permite a los desarrolladores Java manejar datos relacionales en las aplicaciones Java mediante anotaciones o con descriptores XML.

Según (Keith & Schincariol, 2006), para utilizar JPA, es necesario elegir un proveedor de persistencia el cual se ocupa de lo relacionado con la carga y almacenamiento de los datos, cuando refrescar cada instancia y de la sincronización entre los objetos.

La arquitectura de JPA, en alto nivel, se muestra en la Ilustración N° 2.



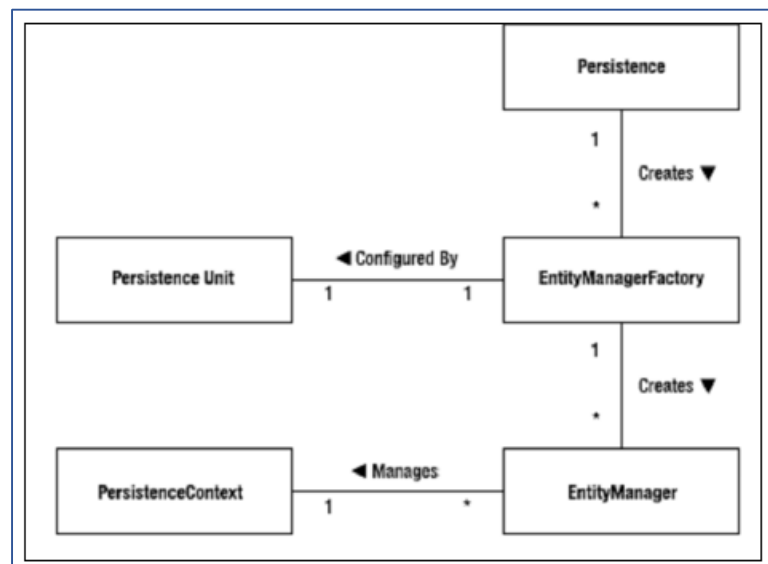
**Ilustración 2:** Arquitectura JPA  
**Fuente.** (Keith & Schincariol, 2006)

Según (Keith & Schincariol, 2006), cualquier aplicación Java (SE o EE) que utiliza JPA no está vinculada a los proveedores de persistencia (incluso si son de código abierto) como Hibernate o el TopLink. En lugar de ello, la aplicación sólo utiliza una especificación estándar de JCP (Java Community Process).

Para facilitar la persistencia JPA se basa en los siguientes elementos:

- Entidades
- Contexto de Persistencia
- Unidad de Persistencia
- Administrador de Entidades

En la Ilustración N° 3 que plantea el autor muestra la relación entre estos elementos.



**Ilustración 3:** Relación entre elementos de JPA  
**Fuente:** (Keith & Schincariol, 2006)

La figura muestra que para cada unidad de persistencia hay un EntityManagerFactory y que muchos Entity Managers pueden ser creados para un solo EntityManagerFactory. En tanto que muchos Entity Managers pueden apuntar al mismo contexto de persistencia.

### **2.4.1. ARQUITECTURA DE JPA**

Según (Álvarez Caules, 2014), la persistencia de datos es la posibilidad de mantener los datos entre ejecuciones de la aplicación. La persistencia es vital en las aplicaciones empresariales debido a la necesidad de acceder a bases de datos relacionales. Las aplicaciones desarrolladas para este entorno deben gestionar ellas mismas la persistencia o utilizar soluciones de terceros para manejar las actualizaciones y recuperaciones de las bases de datos con persistencia. JPA (Java Persistence API) proporciona un mecanismo para gestionar la persistencia y la correlación y funciones relacionales de objetos para las especificaciones EJB 3.0 y EJB 3.1.

La especificación JPA define la correlación relacional de objetos internamente, en lugar de basarse en implementaciones de correlación específicas del proveedor. JPA se basa en el modelo de programación Java que se aplica a los entornos Java EE, pero JPA puede funcionar dentro de un entorno Java SE para probar funciones de la aplicación

JPA representa una simplificación del modelo de programación de persistencia. La especificación JPA define explícitamente la correlación relacional de objetos, en lugar de basarse en implementaciones de correlación específicas del proveedor. JPA crea un estándar para la importante tarea de la correlación relacional de objetos mediante la utilización de anotaciones o XML para correlacionar objetos con una o más tablas de una base de datos. Para simplificar aún más el modelo de programación de persistencia:

La API EntityManager puede persistir, actualizar, recuperar o eliminar objetos de una base de datos

La API EntityManager y los metadatos de correlación relacional de objetos manejan la mayoría de las operaciones de base de datos sin que sea necesario que se escriba código JDBC o SQL para mantener la persistencia.

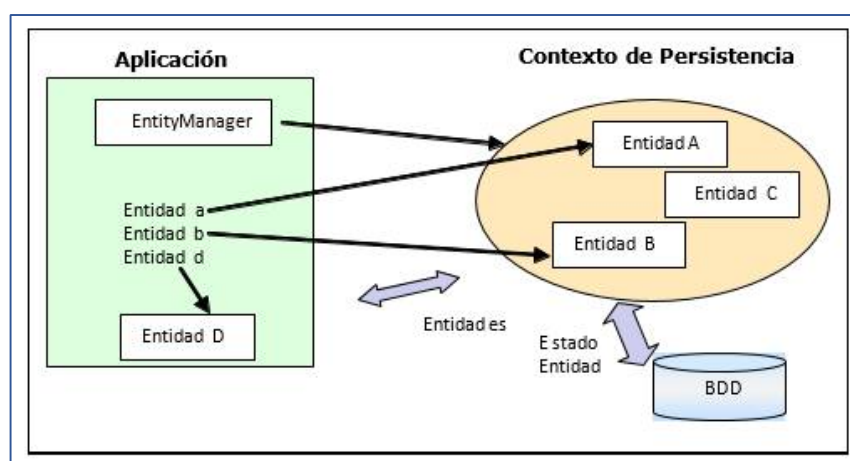
JPA proporciona un lenguaje de consulta, ampliando el lenguaje de consulta EJB independiente (también denominado JPQL), que se puede utilizar para recuperar objetos sin escribir consultas SQL específicas de la base de datos con la que está trabajando.

JPA está diseñado para funcionar dentro y fuera de un contenedor Java Enterprise Edition (Java EE). Cuando se ejecuta JPA dentro de un contenedor, las aplicaciones pueden utilizar el contenedor para gestionar el contexto de persistencia. Si no hay ningún contenedor para gestionar JPA, la aplicación debe manejar ella misma la gestión del contexto de persistencia. Las aplicaciones diseñadas para la persistencia gestionada por contenedor no requieren tanta implementación de código para manejar la persistencia, pero estas aplicaciones no se pueden utilizar fuera de un contenedor. Las aplicaciones que gestionan su propia persistencia pueden funcionar en un entorno de contenedor o en un entorno Java SE.

#### 2.4.1.1. CONTEXTO DE PERSISTENCIA EN JPA

Según (Keith & Schincariol, 2006), un contexto de persistencia es un conjunto de instancias de entidades en las que para cualquier entidad hay únicamente una instancia. En el contexto de persistencia, las instancias de las entidades y sus ciclos de vida son administrados. Al decir que una instancia de entidad es administrada significa que está en el contexto de persistencia y que debe ser representada por un EntityManager.

Cada contexto de persistencia es asociado con una unidad de persistencia, restringiendo las clases de las instancias administradas al conjunto definido por la unidad de persistencia.



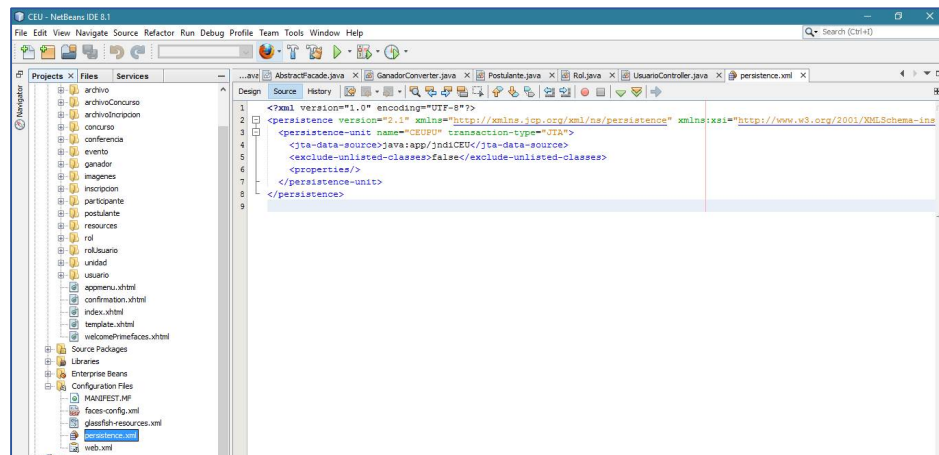
**Ilustración 4:** Contexto de persistencia  
**Fuente:** (Keith & Schincariol, 2006)



### 2.4.1.2. UNIDAD DE PERSISTENCIA

Según (Keith & Schincariol, 2006), la Unidad de Persistencia consiste en la declaración de entidades que serán mapeadas a una base de datos relacional. Es definida por el archivo persistence.xml.

Este archivo de configuración es donde se definen los contextos de persistencia de la aplicación. Se debe situar dentro del directorio META-INF.



**Ilustración 5:** Unidad de persistencia en Netbeans  
**Fuente:** Tatiana Molina, César Mantilla

Los elementos más importantes del archivo son los siguientes:

**persistence-unit name:** Especifica un nombre para el contexto o persistente. Si únicamente se especifica uno, no habrá; que incluir su nombre cuando se recupere el EntityManager (con la anotación @PersistenceContext o @PersistenceUnit).

**transaction-type:** El valor de este elemento es JTA o RESOURCE-LOCAL. El tipo de transacción por omisión es RESOURCE-LOCAL para aplicaciones Java SE. Una transacción tipo JTA significa que el administrador de entidades participa en la transacción.

**provider:** Especifica el nombre del proveedor de persistencia. En el ejemplo se muestra la implementación de GlassFish<sup>3</sup>, Toplink Essentials.

**class:** Lista los nombres de las entidades que son parte de la unidad de persistencia.

**properties:** Se especifica el tipo de base de datos a utilizar en entornos Java SE si no es posible usar JNDI. Las propiedades de la conexión a la base de datos incluyen el nombre de usuario y contraseña para la conexión, la cadena de la conexión (URL) y el nombre de la clase del driver. La propiedad namespace javax.persistence está

<sup>3</sup> **GlassFish:** Proyecto open source de SUN.

reservada para propiedades definidas por la especificación. Las opciones de las especificaciones y propiedades del proveedor deben ser usadas para evitar conflictos con la especificación. El archivo `persistence.xml` mostrado usa la implementación GlassFish. Sus propiedades de la especificación del proveedor son el namespace `toplink` y no son parte de su propia especificación. Los proveedores de persistencia ignorarán cualquier otra propiedad que no estén en sus especificaciones o que no sean parte de sus propiedades de las especificaciones del proveedor.

#### **2.4.2. MAPEO OBJETO RELACIONAL CON JPA**

Según (Coad & Yourdon, 1991), en la actualidad existen muchos paradigmas para programar aplicaciones web. Uno de los más comunes es el paradigma de orientación a objetos. Este se aplica en las capas de las aplicaciones sea cual sea el lenguaje o framework que se utilice.

Para desarrollar un sistema moderno, sin duda el paradigma más utilizado es el de la orientación a objetos. Pero a la hora de modelar las necesidades de persistencia de datos comerciales, el paradigma que se impone es el de base de datos relacionales (RDBMS). Esta diferencia de enfoques hace que un framework de persistencia u ORM sea un componente crítico de la arquitectura de una aplicación.

En los últimos años, numerosos frameworks de persistencia han evolucionado para simplificar la transición del modelo de objetos al relacional y viceversa. Elegir uno que se ajuste a sus requerimientos no es una tarea trivial. Es por ello que destacaremos uno de los que pretende ser el estándar definitivo, Java Persistence API.

#### **2.4.3 LAS ENTIDADES DE JPA**

Según (Oracle Corporation, 2013), una entidad en un objeto ligero de la persistencia. Si se compara con un sistema manejador de base de datos una entidad representa a una tabla de la base. Cada instancia de la entidad representa entonces a una fila en dicha tabla.

Lo primero que un programador debe determinar al comenzar la programación es la capa de entidades o clases, más conocida en inglés como Entity Classes, que básicamente define los tipos de todos los objetos que se manejarán en la aplicación.

#### 2.4.4. LENGUAJE DE CONSULTA PARA JPA

Según (Serna, 2011), el Java Persistence Query Language (JPQL) es usado para definir queries para las entidades y su estado persistente. Esto le permite al desarrollador especificar la semántica de los queries de manera portable, independiente de una base de datos.

Java Persistence Query Language es una extensión de EJB QL; al igual que EJB QL, es un lenguaje al estilo de SQL.

Una sentencia JPQL puede ser un select, un update o un delete. Cualquiera de ellas puede ser construida dinámicamente o puede ser estáticamente definida con metadatos XML o anotaciones (@NamedQuery, @NamedNativeQuery). Además, pueden tener parámetros definidos por nombre o por posición.

Hay que resaltar que el EntityManager es fábrica para objetos Query CreateNamedQuery, createQuery, createNativeQuery; métodos para controlar el máximo de resultados, la paginación, y modo de vaciado (flush).

- SINTÁXIS

Una consulta de select tiene 6 elementos: SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY.

SELECT y FROM son requeridos los demás son opcionales. La forma general de una consulta de select es:

```
QL_statement ::= select_clause from_clause  
[where_clause][groupby_clause][having_clause][orderby_clause]
```

Las consultas de delete o update tienen la siguiente forma:

```
update_statement ::= update_clause [where_clause] delete_statement ::=  
delete_clause [where_clause]
```

La where\_clause tiene el mismo significado que en el caso del SELECT.

## 2.4.5. TRANSACCIONALIDAD

Según (Serna, 2011) , la transacción es la unidad de trabajo de JPA. Cuando se cierra una transacción, JPA ve cual es el estado de las entidades y realiza las operaciones necesarias sobre base de datos para mantener la coherencia entre las entidades y la base de datos. Hay que tener claro que esto pasa si o si (a menos de nos preocupemos expresamente de que no pase como marcando la entidad como “readOnly” o haciendo un “clear()” en el EntityManager, pero esto no es el caso que nos ocupa ya que en la mayoría de los casos no será nuestra intención). Muchas veces que monitorizamos las operaciones que realiza JPA sobre base de datos al finalizar la transacción vemos operaciones (“insert”, “update”, “delete”) que se ejecutan y no sabemos el porqué. El motivo es que hemos modificado una entidad y no nos hemos dado cuenta. Las entidades solo se utilizan para obtener datos y modificarlos en base de datos, nunca se pasan a un framework, a presentación ni nada por el estilo. Si modificamos una propiedad es a sabiendas de que se va a modificar.

Las propagaciones más comunes son:

- REQUIRE\_NEW. Crea la transacción y si ya existe la cierra y la abre de Nuevo.
- REQUIRED. Crea la transacción y si ya existe sigue con la misma.
- NOT\_SUPPORTED. Sin transacción.

La anotación “@Transactional” no es nativa de JAVA (no esta en JPA y no está en la implementación de JPA que estás utilizando) si no que es de SPRING que será el encargado de gestionar el ciclo de vida de la transacción. Que esto no te preocupe porque es muy intuitivo pero obviamente tendrás que cargar JPA desde la configuración de SPRING.

La idea es muy simple “SPRING se hace cargo de todo” solo tienes que crear los BEANS que tiene que utilizar.que son los siguientes:

- Crea un dataSource con
- Instancia el bean de gestion de entityManager
- Instancia los gestores de anotaciones
- Crea un gestor de transacciones
- Instancia el gestor de transacciones con <tx:annotation-driven/>.

Según (Serna, 2011) , el flujo del programa es lineal. Cuando abres una transacción y si abres otra transacción antes de cerrar la anterior crees que tienes dos transacciones, pero no es así. Si abres una transacción antes de cerrar EXPLICITAMENTE la anterior, será el “EntityManager” el que tome la decisión (dependiendo de lo que hayas echo anteriormente) de si hace “commit()” y abre otra transacción, de si hace “rollback()” y abre otra transacción o de si te devuelve la misma transacción que estabas utilizando.

#### **2.4.6. VENTAJAS DE JPA**

- Nos permite desarrollar mucho más rápido.
- Permite trabajar con la base de datos por medio de entidades en vez de Querys.
- Nos ofrece un paradigma 100% orientado a objetos.
- Elimina errores en tiempo de ejecución.
- Mejora el mantenimiento del software.
- Integra conceptos de muchas infraestructuras existentes como Hibernate, Toplink y JDO.
- Se puede usar tanto en entornos Java SE, así como en Java EE.
- Permite que diferentes proveedores de persistencia se puedan usar sin afectar el código del entity.
- Evita tener que capturar información de la pantalla y construir nuestras sentencias SQL. Esto, a diferencia de un API como JDBC, permite pensar más en objetos que en tablas y como acceder a los datos en ellas.

#### **2.4.7. DESVENTAJAS DE JPA**

- Se limita principalmente a la comunidad Java, pero es posible que el API de persistencia aparezca en las otras plataformas en el futuro.
- No ofrece toda la funcionalidad que ofrecería tirar consultas nativas.
- El performance es mucho más bajo que realizar las consultas por JBDC.
- Puede representar una curva de aprendizaje más grande.
- La curva de aprendizaje puede ser un poco compleja. (Blancarte, 2014)

## 2.5. JDBC (JAVA DATABASE CONNECTIVITY)

Según (Ángel Esteban, 2000) , JDBC es un API incluido dentro del lenguaje Java para el acceso a bases de datos. Consiste en un conjunto de clases e interfaces escritos en Java que ofrecen un completo API para la programación de bases de datos, por lo tanto es la una solución 100% Java que permite el acceso a bases de datos.

Es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede, utilizando el dialecto SQL del modelo de base de datos que se utilice.

Según (Pech-May, 2010) , el API JDBC se presenta como una colección de interfaces Java y métodos de gestión de manejadores de conexión hacia cada modelo específico de base de datos. Un manejador de conexiones hacia un modelo de base de datos en particular es un conjunto de clases que implementan las interfaces Java y que utilizan los métodos de registro para declarar los tipos de localizadores a base de datos (URL) que pueden manejar. Para utilizar una base de datos particular, el usuario ejecuta su programa junto con la biblioteca de conexión apropiada al modelo de su base de datos, y accede a ella estableciendo una conexión; para ello provee el localizador a la base de datos y los parámetros de conexión específicos. A partir de allí puede realizar cualquier tipo de tarea con la base de datos a la que tenga permiso: consulta, actualización, creación, modificación y borrado de tablas, ejecución de procedimientos almacenados en la base de datos, etc.

Java DataBase Connectivity es el API<sup>4</sup> de Java que accede a la base de datos a través de ejecuciones de sentencias SQL. (Bauer & King, 2004)

No obstante, es necesario gestionar explícitamente los valores de los campos y su proyección en tablas de una base de datos relacional.

Por tanto:

Hay que tratar con dos modelos de datos, lenguajes y paradigmas de acceso a los datos, muy diferentes (Java y SQL).

---

<sup>4</sup> **API**: Interfaz de Programación de Aplicaciones. Representa una interfaz de comunicación entre componentes software. [http://es.wikipedia.org/wiki/Application\\_Programming\\_Interface](http://es.wikipedia.org/wiki/Application_Programming_Interface)

El esfuerzo necesario para implementar el mapping entre el modelo relacional y el modelo de objetos es demasiado grande:

- Muchos desarrolladores nunca llegan a definir un modelo de objetos para sus datos.
- Se limitan a escribir código Java procedural para manipular las tablas de la base de datos relacional subyacente.
- Se pierden los beneficios y ventajas del desarrollo orientado a objetos.

Esta forma de trabajo es cada vez menos frecuente por el alto costo de desarrollo y mantenimiento.

### **2.5.1. ARQUITECTURA DE JDBC**

En este apartado se introduce una clase muy importante para el acceso a datos a través de Java, se trata de la clase DriverManager.

La columna vertebral de JDBC es el Driver Manager (gestor de drivers) que se encuentra representado por la clase `java.sql.DriverManager`. El gestor de drivers es pequeño y simple y su función primordial es la de seleccionar el driver adecuado para conectar la aplicación o applet con una base de datos determinada, y acto seguido desaparece (el proceso que sigue el DriverManager se explicará con más detalle en siguientes temas en el apartado correspondiente). (Ángel Esteban, 2000) Se puede considerar que JDBC ofrece dos conjuntos de clases e interfaces bien diferenciados, aquellas de más alto nivel que serán utilizados por los programadores de aplicaciones para el acceso a bases de datos, y otras de más bajo nivel enfocadas hacia los programadores de drivers que permiten la conexión a una base de datos. En el presente curso nos vamos a centrar en el primer subconjunto, el de más alto nivel, aunque se comentará algunos puntos de la parte del API de más bajo nivel. (Ángel Esteban, 2000)

Sin embargo, en función de la localización de la base de datos, el driver, la aplicación y el protocolo de comunicación usado, nos podemos encontrar distintos escenarios que accedan a Base de Datos a través de JDBC:

- Aplicaciones standalone.
- Applets comunicando con un servidor Web.

- Aplicaciones y applets comunicando con una base de datos a través de un puente JDBC/ODBC.
- Aplicaciones accediendo a recursos remotos usando mecanismos como Java RMI.

Todos ellos se pueden agrupar en dos tipos distintos de arquitecturas:

#### **2.5.1.1. ARQUITECTURA JDBC EN DOS CAPAS.**

La aplicación que accede a la base de datos reside en el mismo lugar que el driver de la base de datos. El driver accederá al servidor donde corra el motor de base de datos.

En este caso, será el driver el encargado de manejar la comunicación a través de la red.

En el ejemplo, una aplicación java corriendo en una máquina cliente que usa el driver también local. Toda la comunicación a través de la red con la base de datos será manejada por el driver de forma transparente a la aplicación Java.

#### **2.5.1.2. ARQUITECTURA JDBC EN TRES CAPAS.**

Según (Ángel Esteban, 2000) en el modelo de tres capas (three-tier), los comandos son enviados a una capa intermedia (middle-tier) de servicios, la cual enviará sentencias SQL a la base de datos. La base de datos procesa las sentencias y devuelve los resultados a la capa intermedia que se los enviará al usuario.

Este modelo es bastante interesante, ya que la aplicación intermedia no poseerá las restricciones de seguridad de los applets y dejará más libertad al programador; otra ventaja del uso del modelo en tres capas, es que el usuario puede utilizar una API de más alto nivel, y por lo tanto más sencilla de manejar, que será traducida por la capa intermedia a las llamadas apropiadas, en este caso utilizando el API de JDBC. JDBC provee un mecanismo para permitir nombrar las bases de datos, para que los programadores puedan especificar a qué base de datos desean conectarse. Este sistema debe tener las siguientes características:

Drivers de diferentes tipos pueden tener diferentes formas de nombrar las bases de datos.

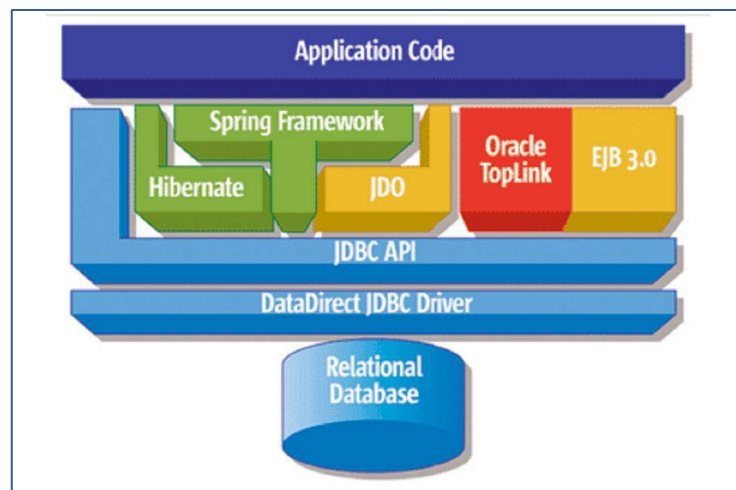


- Este sistema de nombrado debe ser capaz de acoger diversos parámetros de configuración de la red.
- Debería ser capaz de acoger cierto nivel de indirección, para que los nombres pudiesen ser resueltos de alguna manera (DNS) (Visconti, 2010)

### 2.5.2. JDBC COMPONENTE INDISPENSABLE PARA LOS ORM

Según (Reyes Freire, 2016) sin tener en cuenta la solución de mapeo objeto/relacional que se vaya a utilizar para comunicarse con la base de datos relacional, todos ellos dependen de JDBC. Teniendo en cuenta que la mayor parte de las aplicaciones se comunican con bases de datos relacionales, es fundamental considerar cada uno de los niveles del software (desde el código del programa hasta la fuente de datos) para asegurar que el diseño de persistencia objeto/relacional sea óptimo.

Tal y como se verá más adelante, cada una de las soluciones de mapeo objeto/relacional tiene una dependencia particular en el driver JDBC para poder comunicarse con la base de datos de una forma eficiente. Si el driver JDBC que va a participar en la comunicación no es óptimo, la posible gran eficiencia de cualquier Framework quedará debilitada. Por tanto, elegir el driver JDBC que mejor se adapte a la aplicación es esencial a la hora de construir un sistema eficiente en el que participe una solución de mapeo objeto/relacional.



**Ilustración 6:** Interacción de Componentes  
**Fuente:** (Carmen Gonzales, 2016)

La figura muestra una representación de los diferentes mecanismos o soluciones de mapeo objeto/relacional y cómo se relacionan con el código de la aplicación y con

los recursos de datos relacionados. Esto muestra claramente la función crítica que desempeña el driver JDBC puesto que está situado en la base de cada uno de los Frameworks.

Según (Reyes Freire, 2016), la eficiencia del driver JDBC tiene importantes consecuencias en el comportamiento de las aplicaciones. Cada mecanismo de mapeo objeto/relacional es completamente dependiente del driver, sin tener en cuenta el diseño de la API del Framework que esté expuesta al código fuente de la aplicación.

Como los mecanismos de mapeo objeto/relacional generan llamadas eficientes para acceder a la base de datos, mucha gente defiende que la importancia del driver JDBC se ha visto reducida. Sin embargo, como en cualquier arquitectura, la totalidad de eficiencia en una aplicación siempre estará afectada por el nivel más débil del sistema.

Independientemente del código JDBC generado, los mecanismos de mapeo objeto/relacional son incapaces de controlar cómo los drivers interactúan con la base de datos. Entonces la eficiencia de la aplicación depende en gran parte de la habilidad que tenga el driver del nivel JDBC para mover todos los datos manejados entre la aplicación y la base de datos.

Aunque hay múltiples factores que considerar a la hora de elegir un driver JDBC, seleccionar el mejor driver JDBC posible basándose en comportamiento, escalabilidad y fiabilidad es la clave para obtener el máximo beneficio de cualquier aplicación basada en un Framework de mapeo objeto/relacional.

### **2.5.3. CONTROLADORES JDBC**

Según (Mateu C. , 2004), a pesar de las muchas similitudes entre los diferentes SGBD, sus lenguajes, prestaciones, etc., los protocolos de comunicación que se deben emplear para acceder a ellos varían totalmente de unos a otros. Por eso, para comunicarnos con los diferentes SGBD desde JDBC deberemos emplear un controlador (un Driver) que nos aisle de las peculiaridades del SGBD y de su protocolo de comunicaciones.

Según (Páez Martínez, 2006), existen diversos tipos de controladores de JDBC, que clasificaremos según el siguiente esquema:

- Controladores de tipo 1. Bridging Drivers.

Los controladores son los que traducen las llamadas de JDBC a llamadas de algún otro lenguaje de acceso a SGBD (por ejemplo, ODBC). Se usan en aquellas situaciones en las que no disponemos de un driver JDBC más adecuado. Implica instalar en la máquina cliente el controlador que nos permite traducir las llamadas JDBC. El más conocido es el driver JDBC-ODBC que actúa de puente entre JDBC y ODBC.

Ventajas: Buena forma de aprender JDBC. También puede ser buena idea usarlo, en sistemas donde cada máquina cliente tenga ya instalado los drivers ODBC. También es posible que sea la única forma de acceder a ciertos motores de Bases de Datos.

Inconvenientes: No es buena idea usar esta solución para aplicaciones que exijan un gran rendimiento, ya que la transformación JDBC-ODBC es costosa. Tampoco es buena solución para aplicaciones con alto nivel de escalabilidad.

- Controladores de tipo 2. Native API Partly Java Drivers.

Son controladores que usan el API de Java JNI (Java native interface) para presentar una interfase Java a un controlador binario nativo del SGBD. Su uso, igual que los de tipo 1, implica instalar el controlador nativo en la máquina cliente. Suelen tener un rendimiento mejor que los controladores escritos en Java completamente, aunque un error de funcionamiento de la parte nativa del controlador puede causar problemas en la máquina virtual de Java.

Ventajas: Mejor rendimiento que el anterior. Quizá puede ser buena solución para entornos controlados como intranets. Ejemplo OCI oracle.

Inconvenientes: Principalmente la escalabilidad, ya que estos drivers exigen que en la máquina cliente librerías del cliente de la Base de Datos.

- Controladores de tipo 3. Net-protocol All-Java Drivers.

Controladores escritos en Java que definen un protocolo de comunicaciones que interactúa con un programa de middleware que, a su vez, interacciona con un SGBD. El protocolo de comunicaciones con el middleware es un protocolo de red independiente del SGBD y el programa de middleware debe ser capaz de comunicar los clientes con diversas bases de datos. El inconveniente de esta opción estriba en

que debemos tener un nivel más de comunicación y un programa más (el iddleware).

Ventajas: Buena solución cuando necesitamos acceder a Bases de Datos distintas y se quiere usar un único driver JDBC para acceder a las mismas. Al residir la traducción en el servidor del middleware, los clientes no necesitan librerías específicas, tan solo el driver.

Inconvenientes: La desventaja principal reside en la configuración del servidor donde se encuentra el middleware. Necesitará librerías específicas para cada motor de base de datos distinto.

- Controladores de tipo 4. Native-protocol All-Java Drivers.

Son los controladores más usados en accesos de tipo intranet (los usados generalmente en aplicaciones web). Son controladores escritos totalmente en Java, que traducen las llamadas JDBC al protocolo de comunicaciones propio del SGBD. No requieren ninguna instalación adicional ni ningún programa extra.

Ventajas: 100 % portable. Buen rendimiento. El cliente sólo necesita el driver.

Inconvenientes: Al ser independiente de la plataforma, no aprovecha las características específicas del S.O.

Casi todos los SGBD modernos disponen ya de un controlador JDBC (especialmente de tipo 4).

Según (Mateu C. , 2004), los controladores JDBC para identificar una conexión concreta a una base de datos utilizan un formato de dirección de tipo URL (Universal Resource Locator). Esta dirección suele ser de la forma:

`jdbc:controlador:basededatos`

Algunos de los formatos más usados son:

PostgreSQL	<code>jdbc:postgresql://127.0.0.1:5432/basedatos</code>
Oracle	<code>jdbc:oracle:oci8:@DBHOST</code>
JDBC-ODBC	<code>jdbc:odbc:dsn;opcionesodbc</code>
MySQL	<code>jdbc:mysql://localhost/ basedatos?user=jose&amp;password=pepe</code>
SAP DB	<code>jdbc:sapdb://localhost/basedatos</code>

**Ilustración 7:** Formatos  
**Autor:** (Mateu C. , 2004)

Podemos observar que PostgreSQL especifica la dirección IP del servidor, así como el puerto (5432) y el nombre de la base de datos.

Oracle, por otro lado, especifica un sub-controlador (oci8) y un nombre de base de datos que sigue los definidos por Oracle TNS. Podemos ver en los ejemplos de direcciones de conexión que, a pesar de ser diferentes, todas siguen un patrón muy similar (especialmente PostgreSQL, MySQL y SAP DB).

#### **2.5.4. CONEXIÓN A LA BASE DE DATOS.**

Según (Mateu C. , 2004) el método simple de conexión a una base de datos nos proporcionará un objeto de tipo Connection que encapsulará una conexión simple. Podemos tener en cada aplicación tantas conexiones como nos permitan los recursos del sistema (especialmente los del SGBD) y mantener conexiones a diferentes SGBD.

Para obtener una Connection emplearemos el método DriverManager.getConnection (). Nunca instanciamos un objeto de tipo Connection directamente.

Connection con =

```
DriverManager.getConnection (“url”, “usuario”, “password”);
```

Pasamos tres parámetros a getConnection, la dirección de la base de datos en el formato visto anteriormente, el usuario y la contraseña.

Para las bases de datos en las que no es necesario el usuario y la contraseña, los dejaremos en blanco. Cuando llamamos este método, JDBC pregunta a cada controlador registrado si soporta la URL que le pasamos y en caso afirmativo nos devuelve un objeto.

Cuando un Connection ya no vaya a ser usado más, debemos cerrarlo explícitamente con close () para no ocupar recursos. Es especialmente importante liberar las conexiones a bases de datos, ya que constituyen un recurso muy costoso. JDBC, a partir de la versión 2.0, proporciona además un mecanismo para pooling de conexiones, permitiéndonos contar con un bloque de conexiones preestablecidas que, además, se reutilizan de uso en uso.

### 2.5.5. ACCESO A LA BASE DE DATOS

Según (Mateu C. , 2004), una vez tenemos un objeto Connection, podemos empezar a usarlo para ejecutar comandos SQL en la base de datos. Disponemos de tres tipos básicos de sentencias SQL en JDBC:

PreparedStatement: Representa una sentencia precompilada de SQL: que ofrece mejores prestaciones que las sentencias básicas.

CallableStatement: Representa una llamada a un procedimiento almacenado de SQL.

- SENTENCIAS BÁSICAS

Para obtener un objeto Statement utilizaremos el método createStatement del objeto Connection:

```
Statement sent=con.createStatement ();
```

Según (Mateu C. , 2004), disponemos también de otro método, executeUpdate, para ejecutar sentencias que no retornen resultados, por ejemplo UPDATE o DELETE. executeUpdate retorna un entero que nos indica qué número de filas se ha visto afectado por el comando SQL enviado.

Para los casos en que no sabemos a priori si una sentencia retornará una tabla de resultados (como executeQuery) o un número de filas afectadas (como executeUpdate), disponemos finalmente de un método, execute, más genérico. Execute devuelve true si hay un ResultSet asociado a una sentencia y false si dicha sentencia retorna un entero. En el primer caso, podemos recoger el ResultSet resultante con getResultSet, mientras que en el segundo, mediante getUpdateCount podemos recoger el número de filas afectadas.

Es necesario recordar que un Statement representa una única sentencia SQL y, por lo tanto, si hacemos una llamada a execute, executeQuery o executeUpdate los ResultSet asociados a ese Statement se cierran y liberan. Por lo tanto, es muy importante haber acabado de procesar los ResultSet antes de lanzar cualquier otro comando SQL.

Para cerrar un Statement, disponemos de un método close.

A pesar de que al cerrar la Connection también se cierran los Statement asociados, es mucho mejor cerrarlos explícitamente para poder liberar antes los recursos ocupados.

## 2.5.6. LENGUAJE DE CONSULTA PARA JDBC

**API A NIVEL SQL.** Según (Ángel Esteban, 2000), JDBC es un API de bajo nivel, es decir, que está orientado a permitir ejecutar comandos SQL directamente, y procesar los resultados obtenidos. Cada propietario de base de datos implementa un driver JDBC que podemos utilizar en nuestra aplicación java. Normalmente habrá un driver por versión y tipo de base de datos, puesto que no siempre se cumple la compatibilidad hacia versiones anteriores.

**COMPATIBLE CON SQL:** Cada motor de Base de Datos implementa una amplia variedad de comandos SQL, y muchos de ellos no tienen por qué ser compatibles con el resto de motores de Base de Datos. JDBC, para solventar este problema de incompatibilidad, ha tomado la siguiente posición.

JDBC permite que cualquier Comando SQL pueda ser pasado al driver directamente, con lo que una aplicación Java puede hacer uso de toda la funcionalidad que provea el motor de Base de Datos, con el riesgo de que esto pueda producir errores o no en función del motor de Base de Datos.

Con el objetivo de conseguir que un driver sea compatible con SQL (SQL compliant), se obliga a que al menos, el driver cumpla el Estándar ANSI SQL 92.

JDBC debe ser utilizable sobre cualquier otro API de acceso a Bases de Datos, o más en particular ODBC (Open Database Connectivity)

JDBC debe proveer un interfaz homogéneo al resto de APIs de Java.

JDBC debe ser un API simple, y desde ahí, ir creciendo.

JDBC debe ser fuertemente tipado, y siempre que sea posible de manera estática, es decir, en tiempo de compilación, para evitar errores en tiempo de ejecución.

JDBC debe mantener los casos comunes de acceso a Base de Datos lo más sencillo posible:

- Mantener la sencillez en los casos más comunes (SELECT, INSERT, DELETE y UPDATE).
- Hacer realizables los casos menos comunes: Invocación de procedimientos almacenados.
- Crear múltiples métodos para múltiple funcionalidad. JDBC ha preferido incluir gran cantidad de métodos, en lugar de hacer métodos complejos con gran cantidad de parámetros.

### 2.5.7. TRANSACCIONALIDAD

Si hay una propiedad que distingue una base de datos de un sistema de archivos, esa propiedad es la capacidad de soportar transacciones. Si está escribiendo en un archivo y el sistema operativo cae, es probable que el archivo se corrompa. Si está escribiendo en un archivo de base de datos, utilizando correctamente las transacciones, se asegura que, o bien el proceso se completará con éxito, o bien la base de datos volverá al estado en el que se encontraba antes de comenzar a escribir en ella.

Cuando múltiples instrucciones son ejecutadas en una única transacción, todas las operaciones pueden ser realizadas (convertidas en permanentes en la base de datos) o descartadas (es decir, se deshacen los cambios aplicados a la base de datos).

Cuando se crea un objeto Connection, éste está configurado para realizar automáticamente cada transacción. Esto significa que cada vez que se ejecuta una instrucción, se realiza en la base de datos y no puede ser deshecha. Los siguientes métodos en la interfaz.

Connection son utilizados para gestionar las transacciones en la base de datos:

```
void setAutoCommit(boolean autoCommit) throws SQLException
```

```
void commit() throws SQLException
```

```
void rollback() throws SQLException
```

Para iniciar una transacción, invocamos setAutoCommit(false). Esto nos otorga el control sobre lo que se realiza y cuándo se realiza. Una llamada al método commit() realizará todas las instrucciones emitidas desde la última vez que se invocó el método commit(). Por el contrario, una llamada rollback() deshará todos los cambios realizados desde el último commit(). Sin embargo, una vez se ha emitido una instrucción commit(), esas transacciones no pueden deshacerse con rollback().

### 2.5.8. VENTAJAS DE JDBC

- Ofrece un performance superior ya que es la forma más directa de mandar instrucciones la base de datos.
- Permite explotar al máximo las funcionalidades de la base de datos.



### **2.5.9. DESVENTAJAS DE JDBC**

- El mantenimiento es mucho más costoso.
- Introduce muchos errores en tiempo de ejecución.
- El desarrollo es mucho más lento. (Blancarte, 2014)

### **2.6. METODOLOGÍA RUP**

Según (Carrillo, 2011), RUP (Proceso Unificado de Rational - Proceso Unificado de Desarrollo de Software): Es un proceso que de manera ordenada define las tareas y quién de los miembros del equipo de desarrollo las hará. Es una guía para usar UML.

UML es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software.

A pesar de la importancia que tiene hacer una buena Ingeniería del Software para la calidad final de un producto esto se obvia o no siguen los pasos adecuados y muchos consideran entonces que la realización de la ingeniería del software es una pérdida de tiempo.

Esta metodología es el fruto de varios años de trabajo de un colectivo de autores con reconocido prestigio internacional por sus trabajos en este campo. Es posible atribuir a dicha metodología resultados positivos en su aplicación y se considera acertada la decisión de explicarla en las asignaturas dirigidas a estudiantes de la enseñanza superior pertenecientes a las carreras del perfil informático. No obstante se requiere que el estudiante se apropie de determinadas habilidades para la aplicación eficiente y racional de esta metodología que de hecho incluye el tránsito por varios flujos de trabajo y diferentes fases dentro de cada flujo, resultando algo denso para iniciar a los estudiantes en el estudio de la ingeniería del software.

Siempre serán escasos los esfuerzos que el profesor realice para garantizar la calidad del proceso enseñanza aprendizaje de estos temas y no siempre son suficientes los medios auxiliares que se pongan a disposición del estudiantado para el desarrollo de su aprendizaje y que puedan ser usados por el profesor en sus actividades presenciales así como por los propios estudiantes en su trabajo individual.

## **CAPITULO III**

### **ANÁLISIS COMPARATIVO DE LA PRODUCTIVIDAD UTILIZANDO JPA Y JDBC**

En este capítulo se realizará un análisis de la productividad de las herramientas JPA y JDBC; para determinar cuál de ellas ofrece mayores capacidades en cuanto a productividad se refiere. Se utilizará estadística descriptiva para demostrar y presentar los resultados del análisis de los datos obtenidos en la medición de la productividad.

#### **3.1. NIVEL DE CONOCIMIENTO DE LOS INVESTIGADORES EN LAS TECNOLOGÍAS ANALIZADAS**

Con la intención de que el estudio que se propone sea lo más equitativo posible, los investigadores están capacitados con el mismo nivel de conocimientos y experiencia en desarrollo en ambas herramientas analizadas; tanto en JPA como en JDBC. Esto asegurará que el estudio refleje resultados más óptimos comparando las mismas capacidades de cada herramienta y que las conclusiones del estudio sean apoyadas por la experiencia de desarrollo de los investigadores.

#### **3.2. METODOLOGÍA DE INVESTIGACIÓN PARA LA EVALUACIÓN DE LA PRODUCTIVIDAD.**

Con el estudio realizado de la tecnología JPA Y JDBC, se procede a realizar un análisis comparativo entre las mencionadas tecnologías para determinar cuál de ellas es la mejor opción con respecto a la productividad.

El proceso aplicado para el análisis comparativo es el siguiente:

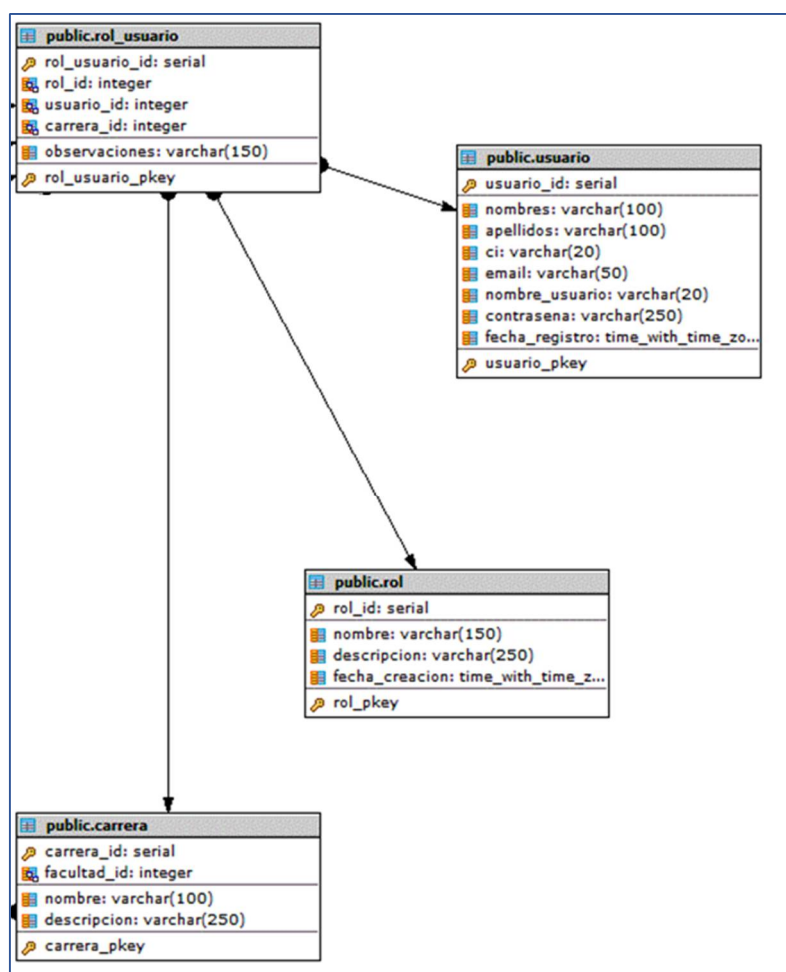
### 3.2.1. CONSTRUCCIÓN DE LOS PROTOTIPOS

Para la demostración de la hipótesis de este estudio se han realizado dos módulos de gestión de usuarios el primero utilizando la tecnología JPA y el segundo utilizando la tecnología JDBC, dichos módulos que cuentan con las siguientes funciones básicas:

- Inserción
- Eliminación
- Selección
- Edición

El estudio ha evaluado las capas bajas e intermedias de cada aplicación según los parámetros que se detallan más adelante en este documento.

En la Ilustración 8, se muestra el modelo entidad relación del módulo desarrollado.



**Ilustración 8:** Diagrama relacional de la base de datos del prototipo  
**Fuente:** Tatiana Molina, César Mantilla

### 3.2.2. PARÁMETROS DE EVALUACIÓN DE PRODUCTIVIDAD.

En el ámbito de la informática se denota gran importancia en la cantidad de líneas de código que posee una aplicación para definir su productividad, según la norma IEEE 1045-1992 la productividad es la relación de una primitiva de salida (líneas de código, puntos función o documentos) y su correspondiente primitiva de entrada (esfuerzo, tiempo) para desarrollar software.

A continuación, se detallan los parámetros utilizados en esta investigación con los que se ha medido la productividad en el desarrollo de aplicaciones.

**Tabla 1:** Parámetros de evaluación

<b>PARÁMETRO</b>	<b>DETALLE</b>
<b>NÚMERO DE LÍNEAS DE CÓDIGO UTILIZADAS</b>	<p>En este parámetro se consideran el número de líneas de código que se emplean en todo el proceso de desarrollo de la aplicación web.</p> <p>Se ha tomado en consideración el código de las capas bajas e intermedias de la aplicación.</p> <p>Se empleará un número positivo entero para evaluar este parámetro.</p> <p>Un número menor de líneas de código denota más productividad.</p>
<b>NÚMERO DE HORAS DE PROGRAMACIÓN EMPLEADAS</b>	<p>Este parámetro evalúa el número de horas de codificación en la programación. Se ha definido en horas y se tiene como base que el día laborable de codificación posee 8 horas.</p> <p>Se especifica el total de horas utilizadas en las capas superiores, inferiores y en el desarrollo de la base de datos.</p> <p>Se empleará un número positivo entero para evaluar este parámetro.</p> <p>Un número menor de horas de programación empleadas denota más productividad.</p>

<p style="text-align: center;"><b>NÚMERO DE FUNCIONES EN EL SISTEMA MANEJADOR DE BASES DE DATOS</b></p>	<p>Especifica el número de funciones que se realizan en la base de datos para operaciones de inserción, extracción y modificación de los datos.</p> <p>Se empleará un número positivo entero para evaluar este parámetro.</p> <p>Un número menor de funciones realizadas en la base de datos denota más productividad.</p>
<p style="text-align: center;"><b>NÚMERO DE LÍNEAS DE CÓDIGO PROGRAMADAS MANUALMENTE</b></p>	<p>Especifica el número de líneas de código que el programador digita para completar la aplicación.</p> <p>Se empleará un número positivo entero para evaluar este parámetro.</p> <p>Un número menor de líneas de código programadas manualmente denota más productividad.</p>

**Fuente:** Tatiana Molina, César Mantilla.

- **CUANTIFICACIÓN DEL NÚMERO DE LÍNEAS DE CÓDIGO EMPLEADAS.**

En el proceso de cuantificación de las líneas de código para la aplicación desarrollada se utiliza la herramienta LinesOfCodeWichtel; esta herramienta freeware permite el conteo de líneas de código exceptuando las líneas en blanco y los comentarios en la aplicación, por lo que se puede verificar con exactitud la cantidad de líneas útiles en la página.

Según su sitio oficial (Berl, 2015) esta herramienta es compatible con la mayoría de lenguajes de programación más utilizados como C++, C#, C, JAVA, JSP, BASIC, HTML, XML, CSS y PHP. A demás, cuenta con opciones de configuración como la cantidad de caracteres que hacen una línea en blanco, los tipos de comentarios, el guardado de preferencias, el idioma (sólo inglés y alemán), entre otros.

En esta medición se cuentan las líneas de código de todas las capas de la aplicación, con excepción de la capa de presentación (o de las vistas); ya que esta investigación

está enfocada en la persistencia de los datos y en la administración de los mismos, más que en la interfaz gráfica con la que sean presentados.

- **CUANTIFICACIÓN DEL NÚMERO DE HORAS DE PROGRAMACIÓN EMPLEADAS.**

En este caso, se cuentan las horas de codificación totales de la aplicación con JPA y con JDBC. Al igual que en el caso anterior, se evalúa el tiempo de desarrollo para todas las capas de la aplicación con excepción de las vistas y en este caso, también se considera el tiempo utilizado para la codificación de las tablas, los campos, las relaciones y las funciones de la base de datos que utilizará la aplicación.

En síntesis, este parámetro evaluará todo el tiempo que los desarrolladores han empleado en la implementación total del módulo de administración de usuario que se utiliza como referencia para la medición del estudio.

Esta cantidad será expresada en un número entero positivo de programación de horas reloj. Los programadores han empleado una jornada laboral de 8 horas diarias y 40 horas a la semana, las mismas que serán cuantificadas de forma manual.

- **CUANTIFICACIÓN DEL NÚMERO DE FUNCIONES EN EL SISTEMA MANEJADOR DE BASES DE DATOS.**

Este parámetro evalúa únicamente los procedimientos y funciones realizados en el sistema manejador de base de datos (pgAdmin en el estudio realizado, tanto para JDBC como para JPA), sin obtener datos del resto de la aplicación.

En este sistema manejador de base de datos se contarán el número de funciones realizadas para trabajar con los datos y permitir las operaciones básicas de:

- Búsqueda
- Inserción
- Actualización
- Eliminación

Se ha establecido en este caso que las tablas, campos y relaciones de la base de datos necesarias para realizar estas operaciones ya se han creado y no se realiza ninguna medición del proceso de creación de los mismos en este parámetro.

- **CUANTIFICACIÓN DEL NÚMERO DE LÍNEAS DE CÓDIGO PROGRAMADAS MANUALMENTE**

Este parámetro evalúa únicamente el número de líneas de código realizadas manualmente por los programadores para la implementación de la aplicación.

Para definir este parámetro se ha contado manualmente el número de líneas de código que el programador agregó en ambos casos, tanto con JPA como con JDBC; para completar las funciones básicas de CRUD que utiliza el módulo estudiado de la aplicación para realizar la persistencia de los datos, en este caso:

- Búsqueda
- Inserción
- Actualización
- Eliminación

### **3.2.3. UTILIZACIÓN DE ESTADÍSTICA DESCRIPTIVA**

En la demostración de la hipótesis de esta investigación se utiliza estadística descriptiva, con la cual se analiza, describe y representa los resultados recolectados en el proceso de medición de los parámetros mencionados en la sección anterior. Se aprovechan las capacidades de los gráficos estadísticos y de otras herramientas para plasmar las características evaluadas en esta investigación para la búsqueda posterior de conclusiones.

### **3.2.4. GENERACIÓN DE LOS RESULTADOS**

Para la obtención de los porcentajes individuales de los parámetros:

Una vez que sean cuantificados los resultados se procede a evaluar las dos herramientas en los cuatro parámetros a considerar, utilizando estadística descriptiva y basándose en el siguiente proceso que ha sido ideado por los investigadores de este proyecto:

- Elección del mayor número de los resultados.

Este número indica cuál de las dos herramientas utiliza más líneas de código, más horas de programación o más funciones en la base de datos. Es decir; mide la cantidad de esfuerzo mayor para los resultados. Evidentemente en estos casos el mayor número implica menos productividad, ya que mientras más tiempo se

dedique al desarrollo de la aplicación, más código se escriba o más funciones se desarrollen, menos productiva es la herramienta.

- Comparación de las cantidades con una regla de tres.

La herramienta que posea mayor cantidad en los resultados (mayor esfuerzo en el desarrollo) equivale al 100% y se halla el porcentaje que posee la herramienta con menores resultados sobre la otra, con una regla de 3, el porcentaje de la segunda herramienta deberá ser menor que la anterior.

- Inversión de los porcentajes.

Debido a que en el paso anterior se hallan porcentajes que indican la cantidad de esfuerzo en la programación, se deben plasmar estas cantidades en función de la productividad y no de esfuerzo; por lo tanto, se invierten los números con el único fin de graficar los resultados de una manera más didáctica.

El proceso anterior puede graficarse con el siguiente escenario:

Luego de cuantificarse las líneas de código se obtiene los siguientes resultados:

JPA=20, JDBC=100.

Entonces primero debe hallarse el valor superior que este caso sería JDBC con 100 líneas de código.

Posteriormente se realiza la regla de 3 para comprobar qué porcentaje tiene un valor inferior sobre el mayor, de la siguiente manera:

$$\frac{\text{número mayor de líneas de código}}{100\%} = \frac{\text{número menor de líneas de código}}{x}$$
$$\frac{100}{100\%} = \frac{20}{x}$$
$$x = \frac{20 \times 100\%}{100}$$
$$x = 20\%$$

Esto indica que JPA realiza sólo un 20% del esfuerzo que realiza JDBC con la codificación de las líneas de código, por lo que implicaría que JPA es 80% más productiva que JDBC en este parámetro.

Para la obtención de los porcentajes totales de la productividad:



Cuando se han calculado los valores de porcentajes de cada uno de los parámetros que medirá este estudio, se proceden a calcular los valores generales con un promedio total de los parámetros, para obtener un promedio de accesibilidad final.

$$Productividad = \frac{nlc + nhp + nfb + nlcp}{4}$$

Esta fórmula devolverá el promedio total de productividad. En donde:

**Tabla 2:** Descripción de las variables de la fórmula de productividad aplicada

ABREVIACIÓN	SIGNIFICADO
<b>nlc</b>	Porcentaje calculado de productividad en número de líneas de código
<b>nhp</b>	Porcentaje calculado de productividad en número de horas de programación
<b>nfb</b>	Porcentaje calculado de productividad en número de funciones en la base de datos
<b>nlcp</b>	Porcentaje del número de líneas de código programadas manualmente

**Fuente:** Tatiana Molina, César Mantilla.

La suma y promedio de los tres parámetros no necesariamente implica que los tres tengan el mismo peso/importancia dentro del desarrollo de una aplicación, podría darse el caso de que las líneas de código tengan un peso inferior al del tiempo programación; sin embargo, se ha evaluado a los tres de la misma manera ya que este estudio no pretende demostrar cuál de los tres influye más en la productividad, sino cuál de las dos herramientas elegidas, JPA y JDBC, poseen mayor productividad.

### 3.3. OBTENCIÓN DE LOS RESULTADOS

Una vez definida la metodología de evaluación de los parámetros que se analizarán y las herramientas que se utilizarán se obtienen los datos para JPA y JDBC.

#### 3.3.1. NÚMERO DE LÍNEAS DE CÓDIGO

- APLICANDO JDBC

En la Tabla 3 se obtiene los resultados del número de líneas de código de la aplicación de JDBC:

**Tabla 3:** Número de líneas de código utilizadas en JDBC

Sección	Número de líneas de código
Acceso a Datos	129
Entity Clases	606
Lógica de Negocios	1224
<b>TOTALES</b>	<b>1959</b>

**Fuente:** Tatiana Molina, César Mantilla.

- 
- Observación:

Debido a que se ha aplicado JDBC con el enfoque tradicional para desarrollar la aplicación, las líneas de código aquí especificadas son escritas casi en su totalidad por los desarrolladores, sin la utilización de ninguna herramienta, a diferencia de con JPA como se explica en la sección siguiente.

- APLICANDO JPA

En la Tabla 4 se obtiene los resultados del número de líneas de código de la aplicación de JPA.

**Tabla 4:** Número de líneas de código utilizadas en JPA

Sección	Número de líneas de código
Entity Clases	238
Converters	329
Controllers	87
<b>TOTALES</b>	<b>654</b>

**Fuente:** Tatiana Molina, César Mantilla.

- Observación:

Se ha empleado herramientas de generación automática de código para la implementación de esta aplicación; esto implica que las líneas de código especificadas en esta sección no han sido necesariamente escritas en su totalidad por los programadores. Esto puede provocar un margen de error leve en esta sección.

### 3.3.2. NÚMERO DE HORAS DE PROGRAMACIÓN EMPLEADAS

- APLICANDO JDBC

En la Tabla 5 se obtiene los resultados del número de horas de programación de la aplicación de JDBC:

**Tabla 5:** Horas de programación empleadas en JDBC

Sección	Número de horas	Equivalente en días laborables
Base de Datos	320	40
Acceso a Datos	8	1
Entity Clases	224	28
Lógica de Negocios	360	45
<b>TOTALES</b>	<b>912</b>	<b>114</b>

Fuente: Tatiana Molina, César Mantilla.

- APLICANDO JPA

En la siguiente tabla se obtiene los resultados del número de horas de programación de la aplicación de JPA:

**Tabla 6:** Horas de programación empleadas en JPA

Sección	Número de horas	Equivalente en días laborables
Base de Datos	160	20
Entity Clases	8	1
Converters	40	5
Controllers	40	5
<b>TOTALES</b>	<b>248</b>	<b>31</b>

Fuente: Tatiana Molina, César Mantilla.

### 3.3.3. NÚMERO DE FUNCIONES EN EL SISTEMA MANEJADOR DE BASES DE DATOS

- APLICANDO JDBC

En la siguiente tabla se obtiene los resultados del número de funciones utilizando JDBC.

**Tabla 7:** Funciones en el SMBD con JDBC

Tabla	Funciones	Número de Funciones
<b>Usuario</b>	Usuario_listar Usario_id Usuario_eliminar Usuario_modificar Usuario_insertar	5
<b>Rol_Usuario</b>	Rol_Usuario_listar Rol_Usuario_id Rol_Usuario_eliminar Rol_Usuario_modificar Rol_Usuario_insertar	5
<b>Rol</b>	Rol_Usuario_listar Rol_Usuario_id Rol_Usuario_eliminar Rol_Usuario_modificar Rol_Usuario_insertar	5
<b>Unidad</b>	Rol_Usuario_listar Rol_Usuario_id Rol_Usuario_eliminar Rol_Usuario_modificar Rol_Usuario_insertar	5
<b>TOTAL DE FUNCIONES UTILIZADAS</b>		20

Fuente: Tatiana Molina, César Mantilla.

- APLICANDO JPA

A continuacion se obtiene los resultados del número de funciones utilizando JPA:

**Tabla 8:** Funciones en el SMBD con JPA

Tabla	Funciones	Número de Funciones
<b>Usuario</b>	(Ninguna)	0
<b>Rol_Usuario</b>	(Ninguna)	0
<b>Rol</b>	(Ninguna)	0
<b>Unidad</b>	(Ninguna)	0
<b>TOTAL DE FUNCIONES UTILIZADAS</b>		0

Fuente: Tatiana Molina, César Mantilla.

### 3.3.4 NÚMERO DE LÍNEAS DE CÓDIGO PROGRAMADAS MANUALMENTE

- APLICANDO JDBC

En la Tabla 9 se obtiene los resultados Número de Líneas de Código Programadas Manualmente de la aplicación de JDBC:

**Tabla 9:** Líneas de código programadas manualmente con JDBC

Función del CRUD de las 4 Tablas	Número de Líneas de Código Programadas Manualmente
<b>Inserción</b>	243
<b>Selección</b>	328
<b>Eliminación</b>	106
<b>Actualización</b>	356
<b>TOTAL DE LÍNEAS PROGRAMADAS</b>	1033

Fuente: Tatiana Molina, César Mantilla.

- APLICANDO JPA

En la Tabla 10 se obtiene los resultados Número de Líneas de Código Programadas Manualmente aplicación de JPA:

**Tabla 10:** Líneas de código programadas manualmente con JPA

Función del CRUD de las 4 Tablas	Número de Líneas de Código Programadas Manualmente
<b>Inserción</b>	0
<b>Selección</b>	0
<b>Eliminación</b>	0
<b>Actualización</b>	0
<b>TOTAL DE LÍNEAS PROGRAMADAS</b>	0

Fuente: Tatiana Molina, César Mantilla.

### 3.4. ANÁLISIS E INTERPRETACIÓN DE LOS RESULTADOS DEL ESTUDIO

#### 3.4.1. HIPÓTESIS A DEMOSTRAR

La tecnología JPA es la más adecuada con respecto a la productividad para el desarrollo de aplicaciones Web con Java en comparación que JDBC.

#### 3.4.2. CONTRASTE DE LOS RESULTADOS OBTENIDOS DE LOS ANÁLISIS INDIVIDUALES DE LOS 4 PARÁMETROS MEDIDOS.

- Resultados individuales de los parámetros medidos

Los valores totales obtenidos en las mediciones anteriores de cada uno de los parámetros evaluados están resumidos en la tabla 11:

**Tabla 11:** Resumen de resultados de la evaluación

RESUMEN DE LOS RESULTADOS		
PARÁMETRO	JDBC	JPA
NÚMERO DE LÍNEAS DE CÓDIGO UTILIZADAS	1959	654
NÚMERO DE HORAS DE PROGRAMACIÓN EMPLEADAS	912	248
NÚMERO DE FUNCIONES EN EL SISTEMA MANEJADOR DE BASES DE DATOS	20	0
NÚMERO DE LÍNEAS DE CÓDIGO PROGRAMADAS MANUALMENTE	1033	0

**Fuente:** Tatiana Molina, César Mantilla.

Aplicando el proceso que ha sido detallado en la sección 3.1.3 de este documento se obtienen los porcentajes de comparación de productividad siguientes:

- PARA LAS LÍNEAS DE CÓDIGO

$$\frac{\text{número mayor de líneas de código}}{100\%} = \frac{\text{número menor de líneas de código}}{x}$$

$$\frac{1959}{100\%} = \frac{654}{x}$$

$$x = \frac{654 \times 100\%}{1959}$$

$$x = 33,38\%$$

Los resultados demuestran que JPA obtiene un 33,38% de líneas de código comparado con JDBC.

- PARA LAS HORAS DE PROGRAMACIÓN:

- 

$$\frac{\text{número mayor de horas}}{100\%} = \frac{\text{número menor de horas}}{x}$$

$$\frac{912}{100\%} = \frac{248}{x}$$

$$x = \frac{248 \times 100\%}{912}$$

$$x = 27,19\%$$

Los resultados demuestran que JPA utiliza un 27,19% de horas de programación comparado con JDBC.

- PARA LAS FUNCIONES EN EL MANEJADOR DE BASE DE DATOS:

- 

$$\frac{\text{número mayor de funciones}}{100\%} = \frac{\text{número menor de funciones}}{x}$$

$$\frac{20}{100\%} = \frac{0}{x}$$

$$x = \frac{0 \times 100\%}{20}$$

$$x = 0\%$$

Se obtiene que JPA utiliza un 0% de funciones codificadas en la base de datos comparado con JDBC.

- PARA LAS LÍNEAS DE CÓDIGO PROGRAMADAS MANUALMENTE:

- 

$$\frac{\text{número mayor de líneas de código}}{100\%} = \frac{\text{número menor de líneas de código}}{x}$$

$$\frac{1033}{100\%} = \frac{0}{x}$$

$$x = \frac{0 \times 100\%}{1033}$$

$$x = 0\%$$

Se obtiene que JPA utiliza un 0% de líneas de código programadas manualmente en comparación con JDBC.

### 3.4.3. RESUMEN DE LOS RESULTADOS POR PARÁMETROS

Los resultados de esfuerzo de desarrollo, hallados de cada uno de los parámetros, se pueden resumir en términos de productividad en la tabla 12.

**Tabla 12:** Comparación de los parámetros medidos de JDBC Vs JPA

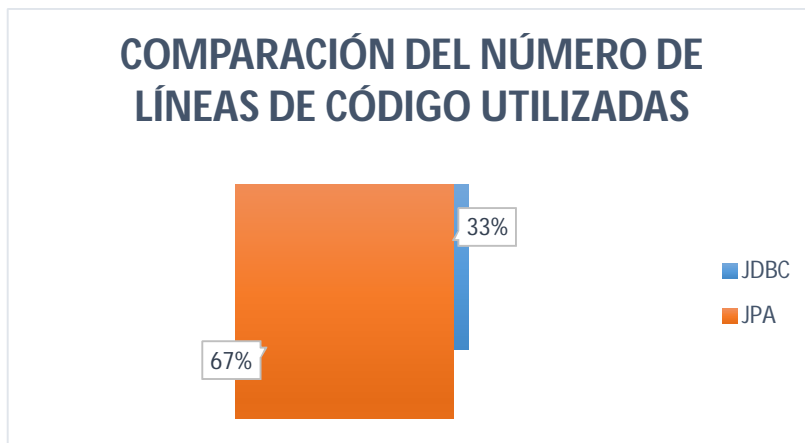
COMPARACIÓN DE PRODUCTIVIDAD POR PARÁMETROS		
PARÁMETRO	JDBC	JPA
NÚMERO DE LÍNEAS DE CÓDIGO UTILIZADAS	33,38%	66,62%
NÚMERO DE HORAS DE PROGRAMACIÓN EMPLEADAS	27,19%	72,81%
NÚMERO DE FUNCIONES EN EL SISTEMA MANEJADOR DE BASES DE DATOS	0%	100%
NÚMERO DE LÍNEAS DE CÓDIGO PROGRAMADAS MANUALMENTE	0%	100%

Fuente: Tatiana Molina, César Mantilla.



- COMPARACIÓN DE LAS LÍNEAS DE CÓDIGO

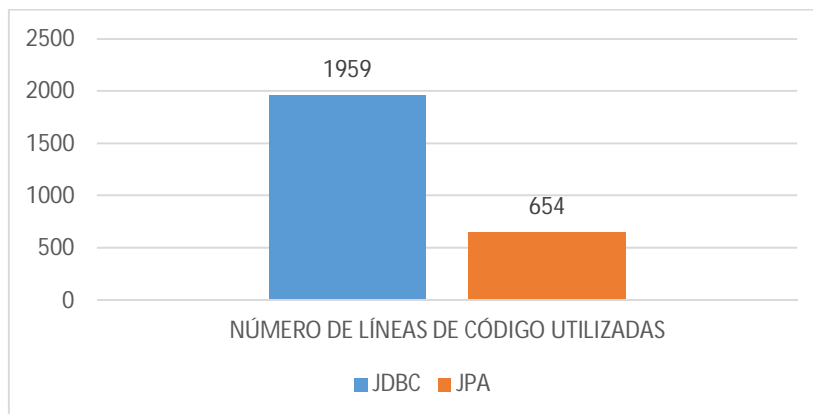
En esta sección se analizan los resultados del número de líneas de código presentes en la aplicación.



**Ilustración 9:** Comparación del número de líneas de código utilizadas con JDBC Vs JPA

**Fuente:** Tatiana Molina, César Mantilla.

En el gráfico se puede observar los porcentajes que ocupan cada herramienta en líneas de código. JDBC obtiene un 33% mientras que JPA el 66%, esto implica que JPA es 3 veces más productivo que JDBC en este parámetro de evaluación, debido a que JDBC posee tres veces más código escrito que JPA. Sin embargo, debe señalarse que los resultados de JPA de productividad en este parámetro podrían ser incluso superiores, debido a que con la utilización de herramientas de generación de código se genera código automáticamente que no es escrito por el programador y que no representa esfuerzo para el mismo.



**Ilustración 10:** Cantidad de líneas de código utilizadas

**Fuente:** Tatiana Molina, César Mantilla.

- COMPARACIÓN DE LAS HORAS DE PROGRAMACIÓN EMPLEADAS

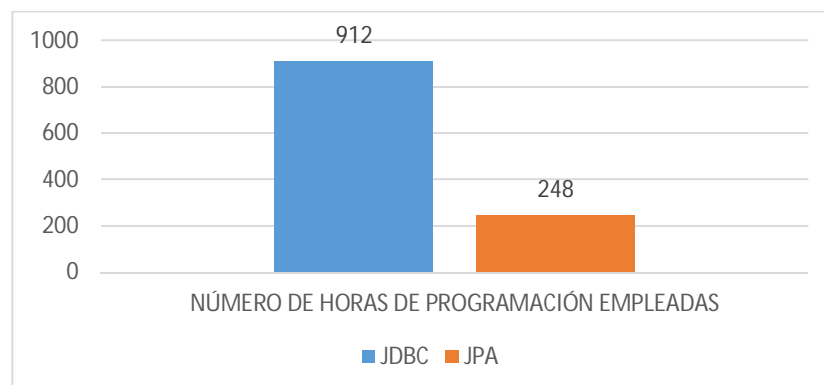
En esta sección se analizan los resultados del número de horas de desarrollo obtenidos



**Ilustración 11:** Comparación del número de horas empleadas en el desarrollo  
**Fuente:** Tatiana Molina, César Mantilla.

En el gráfico se puede observar los porcentajes que ocupan cada herramienta en horas de programación. JDBC obtiene un 27% mientras que JPA el 73%, esto implica que JPA es casi 4 veces más productivo que JDBC en este parámetro de evaluación, debido a que JDBC posee 3,67 veces más horas de programación que JPA.

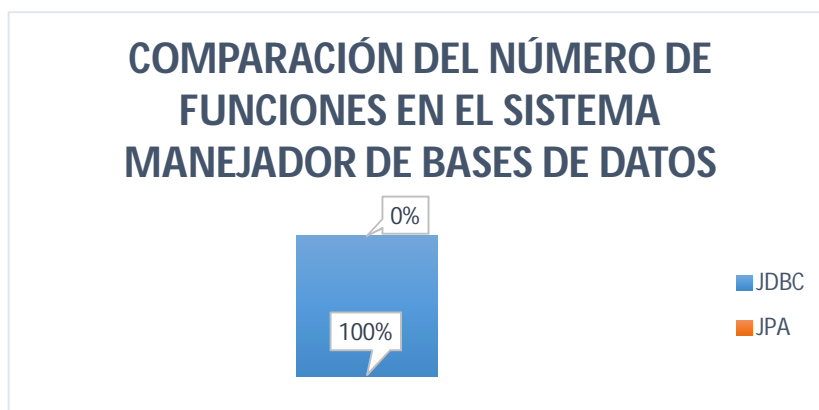
A diferencia del parámetro medido anteriormente la productividad podría medirse mayormente en términos de horas de programación, ya que el tiempo de desarrollo abarca a todos los procesos que se realizaron para tener la aplicación funcionando, por lo que estos datos deberían darnos una idea clara de los resultados generales de la productividad.



**Ilustración 12:** Cantidad de horas de programación empleadas  
**Fuente:** Tatiana Molina, César Mantilla.

Es este gráfico se puede comparar la cantidad de horas de desarrollo presentes en la aplicación en cada una de las dos herramientas seleccionadas.

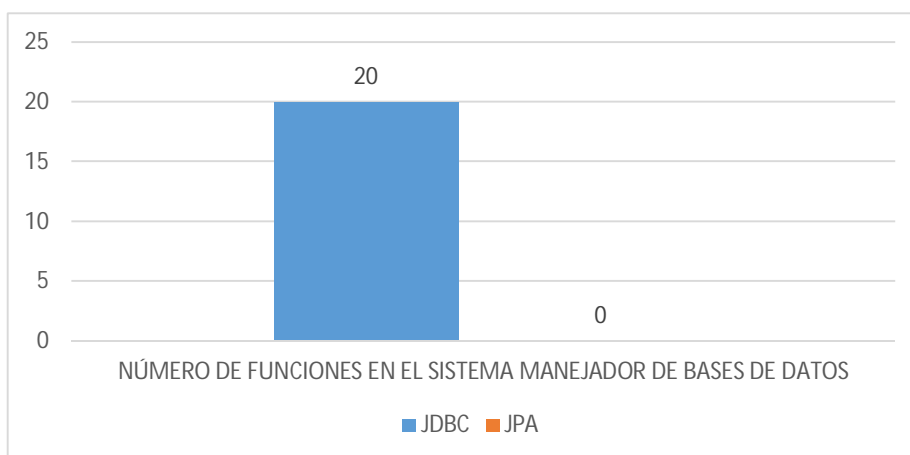
- COMPARACIÓN DE LAS FUNCIONES EN LA BASE DE DATOS



**Ilustración 13:** Comparación del número de las funciones en el SMBD  
**Fuente:** Tatiana Molina, César Mantilla.

En el gráfico se puede observar los porcentajes que ocupan cada herramienta en funciones codificadas en la base de datos. JPA obtiene un 100% de productividad mientras que JDBC el 0%, en este parámetro es evidente que mientras en JPA no se realizan funciones en la base de datos, JDBC realiza 20.

Este parámetro se puede analizar de una manera diferente a los anteriores, ya que no se puede especificar con un número cuántas veces es superior JPA a JDBC, sólo es posible demostrar la cantidad de trabajo que es desarrollado en JDBC y que no es necesario en JPA. Este parámetro es en el que más se evidencia la productividad de JPA.



**Ilustración 14:** Cantidad de funciones utilizadas en el SMBD  
**Fuente:** Tatiana Molina, César Mantilla.

Es este gráfico se puede comparar la cantidad de funciones desarrolladas en la base de datos para la aplicación en cada una de las dos herramientas seleccionadas.

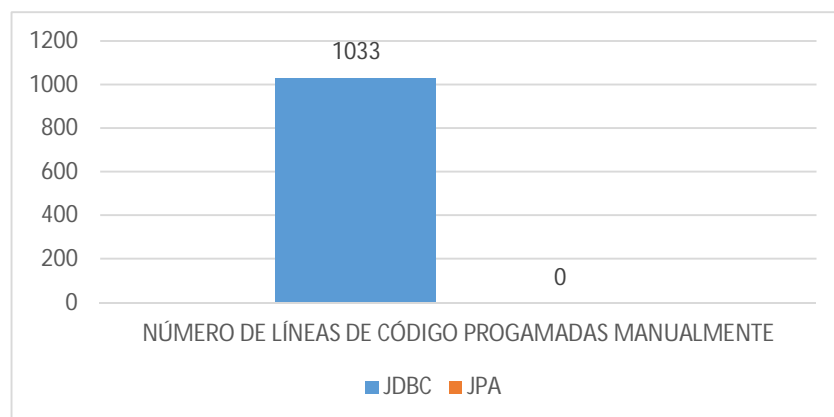
- COMPARACIÓN DEL NÚMERO DE LÍNEAS DE CÓDIGO PROGRAMADAS MANUALMENTE



**Ilustración 15:** Comparación del número de líneas de código programadas manualmente  
**Fuente:** Tatiana Molina, César Mantilla.

En el gráfico se puede observar los porcentajes que ocupan cada herramienta en líneas de código programadas manualmente. JDBC obtiene un 0% de productividad mientras que JPA el 100%, en este parámetro es evidente que mientras en JPA no se codifican líneas manualmente para realizar la persistencia con un CRUD básico, JDBC realiza 1033.

En este parámetro se debe tomar en cuenta que el framework realiza solamente las funciones básicas de cada tabla y en el módulo estudiado se han tomado en cuenta solamente dichas funciones CRUD. Sin embargo, en una aplicación web completa necesariamente se tendrá que agregar código que será desarrollado por los programadores, aunque se utilice JPA.



**Ilustración 16:** Cantidad de líneas de código programadas manualmente  
**Fuente:** Tatiana Molina, César Mantilla.

En este gráfico se puede comparar la cantidad de líneas de código programadas manualmente para la aplicación en cada una de las dos herramientas seleccionadas.

### 3.4.4. APLICACIÓN DE LAS ENCUESTAS

Para ampliar la fundamentación de la presente investigación se consideró realizar encuestas a expertos en el desarrollo de las aplicaciones web, las encuestas se lo realizaron a varios programadores de la Universidad, lo cual permitió obtener datos en base a los indicadores de la tecnología JPA y JDBC.

Para la aplicación de las Encuestas se utilizó la herramienta Google Forms, a través de esta herramienta podemos generar formularios online sin tener conocimiento de código, y acceder rápidamente a los resultados usando nuestra cuenta de Google, en forma de una planilla también de Google Docs, siendo una herramienta muy útil para planificar eventos, crear encuestas, crear pruebas o recopilar información.

### 3.4.5. RECOLECCION DE DATOS DE LAS ENCUESTAS REALIZADAS

Las encuestas se realizaron en total a 28 expertos en programación y se obtuvo los siguientes resultados para cada una de las preguntas:

1.- ¿Cómo describiría su nivel de conocimientos y experiencia en el desarrollo de aplicaciones Java Web?

- BAJO
- INTERMEDIO
- AVANZADO



**Ilustración 17:** Nivel de conocimiento y experiencia en desarrollo web  
**Fuente:** Tatiana Molina, César Mantilla.

- Este gráfico demuestra que en su mayoría las personas encuestadas tienen conocimientos avanzados e intermedios en el Desarrollo de Aplicaciones Java Web.

2. ¿Cuál de las siguientes tecnologías prefiere para desarrollar aplicaciones Java Web?

- JPA
- JDBC

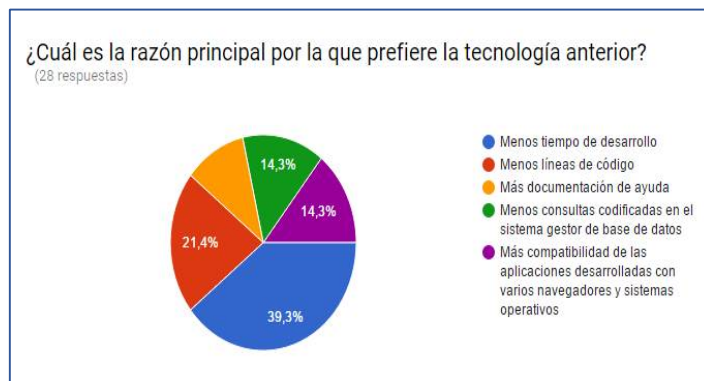


**Ilustración 18:** Preferencia de tecnología en aplicaciones Java Web  
**Fuente:** Tatiana Molina, César Mantilla.

- El gráfico demuestra que de 28 encuestados, 20 programadores prefieren utilizar la tecnología JPA para el desarrollo de aplicaciones Java Web mientras que solo 8 programadores optan por utilizar JDBC

3. ¿Cuál es la razón principal por la que prefiere la tecnología anterior?

- Menos tiempo de desarrollo
- Menos tiempo de código
- Más documentación de ayuda
- Menos consultas codificadas en el sistema Gestor de Base de Datos

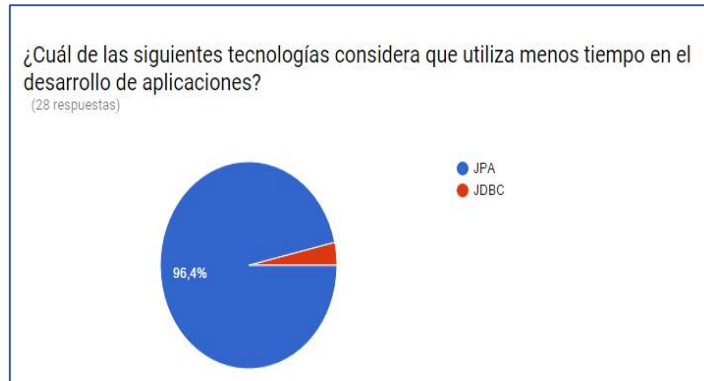


**Ilustración 19:** Razones para utilizar las tecnologías antes mencionadas  
**Fuente:** Tatiana Molina, César Mantilla.

- En el gráfico se puede visualizar que prefieren la Tecnología JPA, por el menos tiempo que se invierte en la programación.

4. ¿Cuál de las siguientes tecnologías considera que utiliza menos tiempo en el desarrollo de aplicaciones?

- JPA
- JDBC

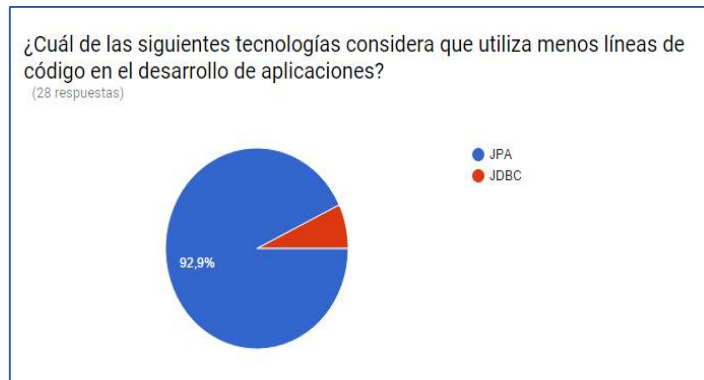


**Ilustración 20:** Menor tiempo de desarrollo de aplicaciones  
**Fuente:** Tatiana Molina, César Mantilla.

- El gráfico anterior arroja un total de 27 programadores que opina que JPA utiliza menos tiempo en el Desarrollo de Aplicaciones Web.

5. ¿Cuál de las siguientes tecnologías considera que utiliza menos líneas de código en el desarrollo de aplicaciones?

- JPA
- JDBC



**Ilustración 21:** Menor cantidad de líneas de código  
**Fuente:** Tatiana Molina, César Mantilla.

- El gráfico anterior arroja un total de 26 programadores que opina que JPA emplea menos líneas de código en el Desarrollo de Aplicaciones Web.

6. ¿Cuál de las siguientes tecnologías considera que ofrece más documentación de ayuda?

- JPA
- JDBC

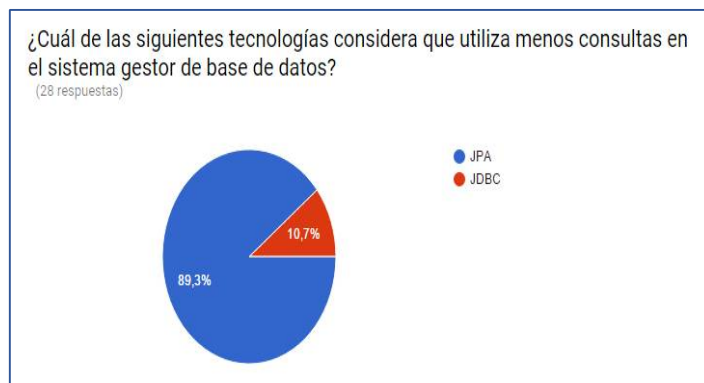


**Ilustración 22:** Mayor cantidad de documentación de ayuda  
**Fuente:** Tatiana Molina, César Mantilla.

- El gráfico anterior arroja un total de 15 programadores que opina que JPA ofrece más documentación de ayuda para los programadores.

7. ¿Cuál de las siguientes tecnologías considera que utiliza menos consultas en el sistema gestor de base de datos?

- JPA
- JDBC



**Ilustración 23:** Menos consultas en el sistema de Gestión de Base de datos  
**Fuente:** Tatiana Molina, César Mantilla.

- El gráfico anterior arroja un total de 25 programadores que opina que la Tecnología JPA utiliza menos consultas al Sistema Gestor de Base de Datos.



8. ¿Cuál de las siguientes tecnologías considera que ofrece más compatibilidad en navegadores y sistemas operativos?

- JPA
- JDBC



**Ilustración 24:** Mayor compatibilidad en navegadores y sistemas operativos  
**Fuente:** Tatiana Molina, César Mantilla.

- En esta pregunta se observa que 15 de 28 programadores opinan que la tecnología JPA prevalece sobre la tecnología JDBC en cuanto a la compatibilidad con los navegadores.

El resultado final de las encuestas realizadas a los expertos en el desarrollo de aplicaciones web, se obtuvo los siguientes datos.

- Se determinó que JPA brinda más beneficios en cuanto a la productividad en comparación con JDBC enfocado a las líneas de código y tiempo empleado.
- Además se determinó que JPA ofrece más compatibilidad en navegadores y sistemas operativos

### 3.4.6. DEMOSTRACIÓN DE LA HIPÓTESIS

Con los resultados hallados de cada uno de los parámetros se establece un promedio total de productividad medida como se trata en la sección 3.1.3 de este documento.

- Para JDBC:

$$Productividad = \frac{nlc + nhp + nfb + nlcp}{4}$$

$$Productividad = \frac{33,38\% + 27,19\% + 0\% + 0\%}{4}$$

$$Productividad = \frac{60,57\%}{4}$$

$$Productividad = 15,14\%$$

Se puede verificar de esta manera que el porcentaje promedio de productividad de JDBC con respecto a JPA es del 15,14%.

- Para JPA:

$$Productividad = \frac{nlc + nhp + nfb + nlcp}{4}$$

$$Productividad = \frac{66,72\% + 72,81\% + 100\% + 100\%}{4}$$

$$Productividad = \frac{339,43\%}{4}$$

$$Productividad = 84,86\%$$

Se puede verificar de esta manera que el porcentaje promedio de productividad de JPA con respecto a JDBC es del 84,86%.

Los resultados finales de productividad en cada herramienta utilizada se pueden visualizar en la tabla 13.

**Tabla 13:** Promedio de la productividad medida

PROMEDIO TOTAL DE LA PRODUCTIVIDAD MEDIDA		
	JDBC	JPA
TOTAL	15,14%	84,86%

**Fuente:** Tatiana Molina, César Mantilla.

Representando esos resultados de una manera gráfica se obtiene:



**Ilustración 25:** Promedio de productividad medida  
**Fuente:** Tatiana Molina, César Mantilla.

Por lo que, basado en la obtención de valores y el análisis posterior de los datos de los parámetros de cantidad de líneas de código, cantidad de horas de desarrollo utilizadas y cantidad de funciones en la base de base de datos, se demuestra claramente que el JPA brinda un 84,86% de productividad en estos parámetros comparado con JDBC que en promedio obtiene un 15,14%.

El análisis desarrollado ha permitido demostrar cuál de las dos herramientas posee mayores capacidades de productividad por lo que la hipótesis se cumple en su totalidad.

## **CAPÍTULO IV**

### **IMPLEMENTACIÓN DE CEU - SISTEMA DE GESTIÓN DE EVENTOS DE LA UNIVERSIDAD NACIONAL DE CHIMBORAZO**

En el capítulo anterior de esta investigación se demostró cuál de los frameworks analizados de desarrollo de Aplicaciones Web ofrece mejores resultados en cuanto a productividad, dicho framework es JPA. Por esta razón, la aplicación implementada como caso aplicativo se centrará en la utilización de dicha herramienta para su desarrollo donde se utilizara la metodología RUP<sup>5</sup> 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 conocimiento dentro del personal de la Universidad, tanto docentes, estudiantes y administrativos acerca de los Eventos a realizarse dentro de ésta, genera un problema para la Universidad porque no se cuenta un Sistema de Gestión de Eventos que nos permita estar informados de los Eventos Académicos que se realicen.

##### **4.1.2. DESCRIPCIÓN DEL PROBLEMA**

En la actualidad, en la Universidad Nacional de Chimborazo se realizan eventos permanentemente de carácter académicos. Esta institución carece de un medio mediante el cual se pueda gestionar e informar de estos eventos realizados y a realizarse dentro de la Universidad, donde se permita interactuar y de esta manera facilitar una información completa acerca de los eventos que se llevaran a cabo y de esta manera poder asistir a los mismos.

---

<sup>5</sup> RUP: Rational Unificate Process. Metodología de desarrollo de Software.

La web constituye un medio fundamental para que la escuela pueda permanecer en constante información y futuros cambios en donde se pueda publicar los Eventos, tanto para los estudiantes, docentes y administrativos de la Universidad.

#### **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 Eventos de la Universidad Nacional de Chimborazo.

- Promoción y Difusión de los Eventos Académicos de la UNACH.
- Información de los Eventos tales como:
  - Concursos
  - Congresos
  - Ponencias
- Módulo de Inscripción a los Eventos.
- Resultados de la Calificación obtenida por el Jurado calificador.
- Permitir la obtención de Certificados en el caso de los Concursos.

De acuerdo a los requerimientos expuestos, el sistema se delimitará en Gestionar los Eventos 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 ADMINISTRACIÓN DE EVENTOS**

- Permitirá la Difusión y Promoción del Evento a realizarse mediante la página Web

- **MÓDULO DE INFORMACIÓN**

Permitirá administrar la información de cada uno de los eventos de diferente índole que se realicen en la UNACH, informando a los usuarios de eventos próximos y permitiendo la participación de éstos en dichos eventos.

- **MÓDULO DE REPORTES**

Permitirá la visualización de Resultados de las calificaciones además de la impresión de los respectivos Certificados

#### 4.1.4. PLAN DE DESARROLLO

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

**Tabla 14:** Plan de Desarrollo

<b>HISTORIA DE USUARIO</b>	<b>DURACIÓN EN SEMANAS</b>
Módulo de administración de usuarios y seguridad	1
Módulo de administración de eventos	2
Módulo de información	2
Módulo de reportes	1

**Fuente:** Tatiana Molina y Eduardo Mantilla

## 4.2. ANÁLISIS

### 4.2.1. PLANIFICACIÓN DEL PROYECTO

Esta planificación del proyecto se realizó tras el 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. INTEGRANTES Y ROLES

Con la participación del Director del proyecto, los miembros, los usuarios y desarrolladores, se formará el equipo encargado de la implementación del sistema. Esto implicara 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 15 definiendo Integrantes y Roles.

**Tabla 15:** Integrantes y Roles

Miembro	Grupo	Roles XP	Metodología
Tatiana Molina	Tesista	Rastreador, Testeador, Programador	RUP
Eduardo Mantilla	Tesista	Rastreador, Testeador, Programador	
Ing. Diego Palacios		Consultor	

**Fuente:** Tatiana Molina, César Mantilla.

### 4.2.3. PROTOTIPOS

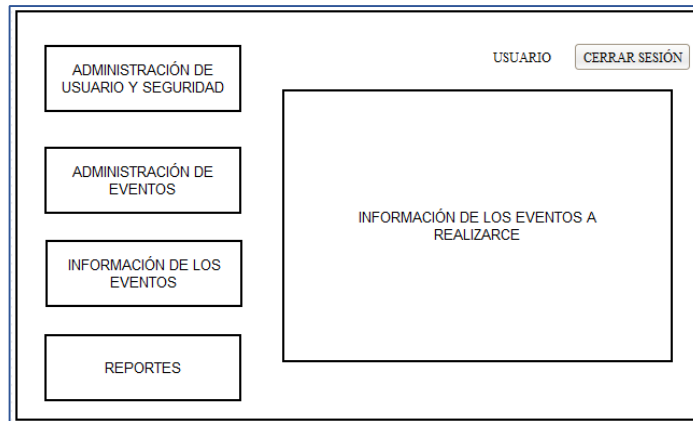
Las interfaces de usuario son las más importantes ya que de esto dependerá el entendimiento fácil y rápido por parte del usuario al comenzar a manejar el sistema. Se pretende que la interfaz del usuario sea amigable, sencilla y funcional con un alto grado de comprensión, por tal razón se crearon los prototipos generales del sistema. A continuación, se realizará una breve descripción del proceso principal.

- En la Ilustración 26 se muestra el prototipo de inicio de sesión de los usuarios

El prototipo de login muestra un formulario con dos campos de entrada de texto. El primer campo está etiquetado como 'USUARIO' y el segundo como 'CONTRASEÑA'. Debajo de estos campos se encuentra un botón rectangular con el texto 'INGRESAR'.

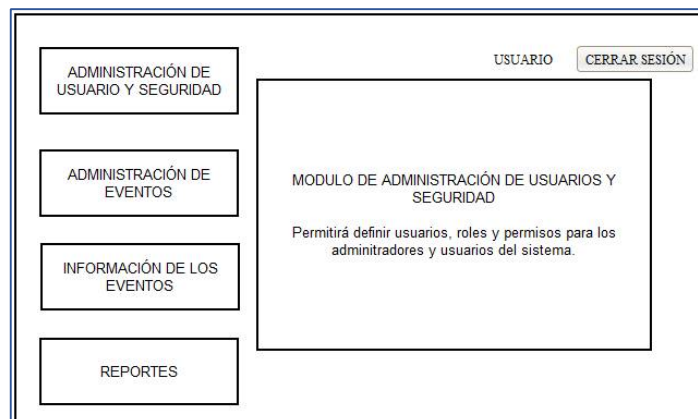
**Ilustración 26:** Prototipo Login  
**Fuente:** Tatiana Molina, César Mantilla.

- En la Ilustración 27 se muestra la página principal del sistema



**Ilustración 27:** Prototipo Página Principal  
**Fuente:** Tatiana Molina, César Mantilla.

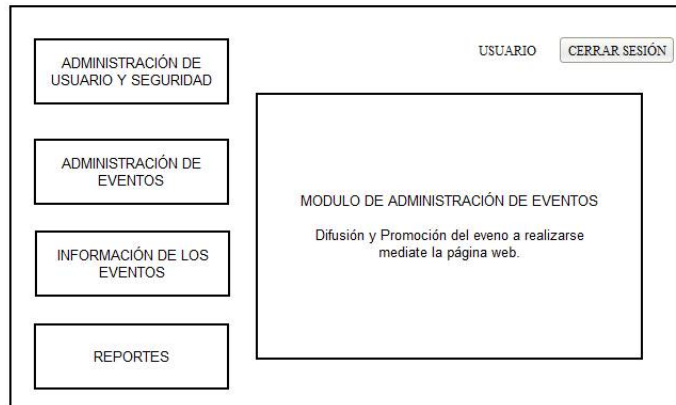
- En la Ilustración 28 se muestra el módulo de administración de usuarios y seguridad.



**Ilustración 28:** Prototipo Administración de Usuarios y Seguridad  
**Fuente:** Tatiana Molina, César Mantilla.

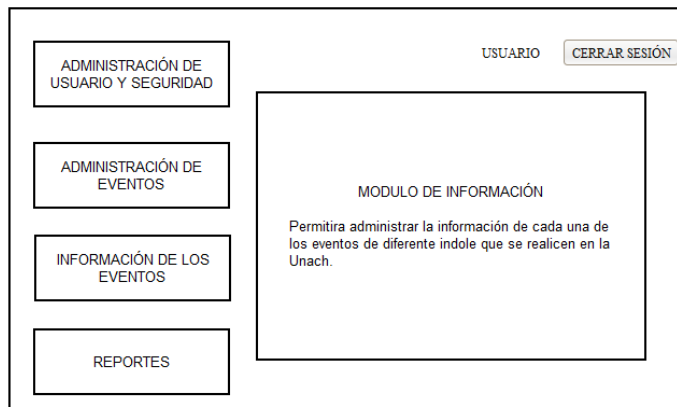
- En la Ilustración 29 se muestra el módulo de administración de eventos.





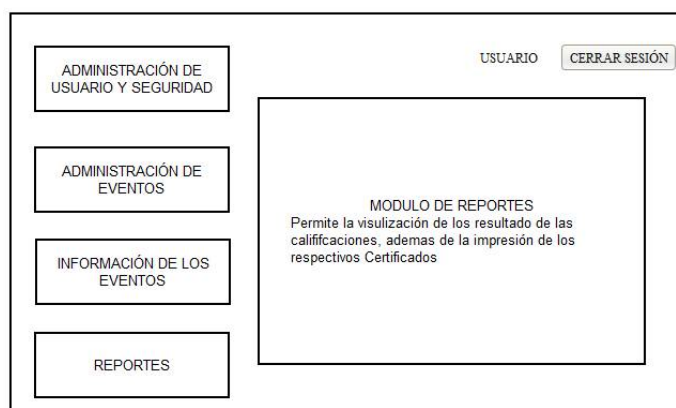
**Ilustración 29:** Prototipo Administración de Eventos  
**Fuente:** Tatiana Molina, César Mantilla.

- En la Ilustración 30 se muestra el módulo de información de los eventos.



**Ilustración 30:** Prototipo del Módulo de Información  
**Fuente:** Tatiana Molina, César Mantilla.

- En la Ilustración 31 se muestra el módulo de reportes.

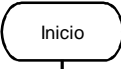
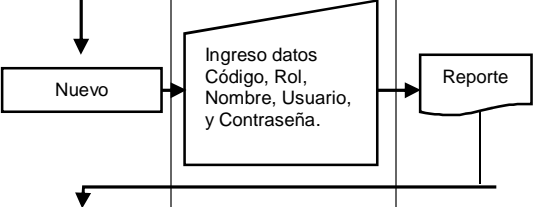
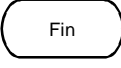


**Ilustración 31:** Prototipo del Módulo de Reportes  
**Fuente:** Tatiana Molina, César Mantilla.

#### 4.2.7. FLUJOS DE TRABAJO

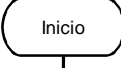
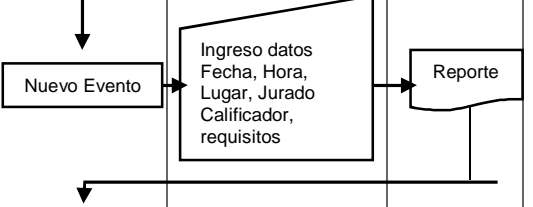
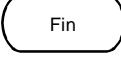
Las actividades del sistema fueron divididas en varios procesos que serán reflejados mediante flujos de trabajo.

**Tabla 16:** Definición del proceso de nuevo usuario

PROCESO NUEVO USUARIO						
	Actividad	Flujograma	IN	OUT	Responsable	Observación
	Inicio					
1	Actividad				Administrador	
	Fin					

Fuente: Tatiana Molina, César Mantilla

**Tabla 17:** Proceso de gestión del evento






PROCESO GESTIÓN DE EVENTO						
	Actividad	Flujograma	IN	OUT	Responsable	Observación
	Inicio					
1	Actividad				Secretaria	
	Fin					

Fuente: Tatiana Molina, César Mantilla

### 4.2.3. HERRAMIENTAS DE DESARROLLO

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

**Tabla 18:** Herramientas utilizadas en el desarrollo

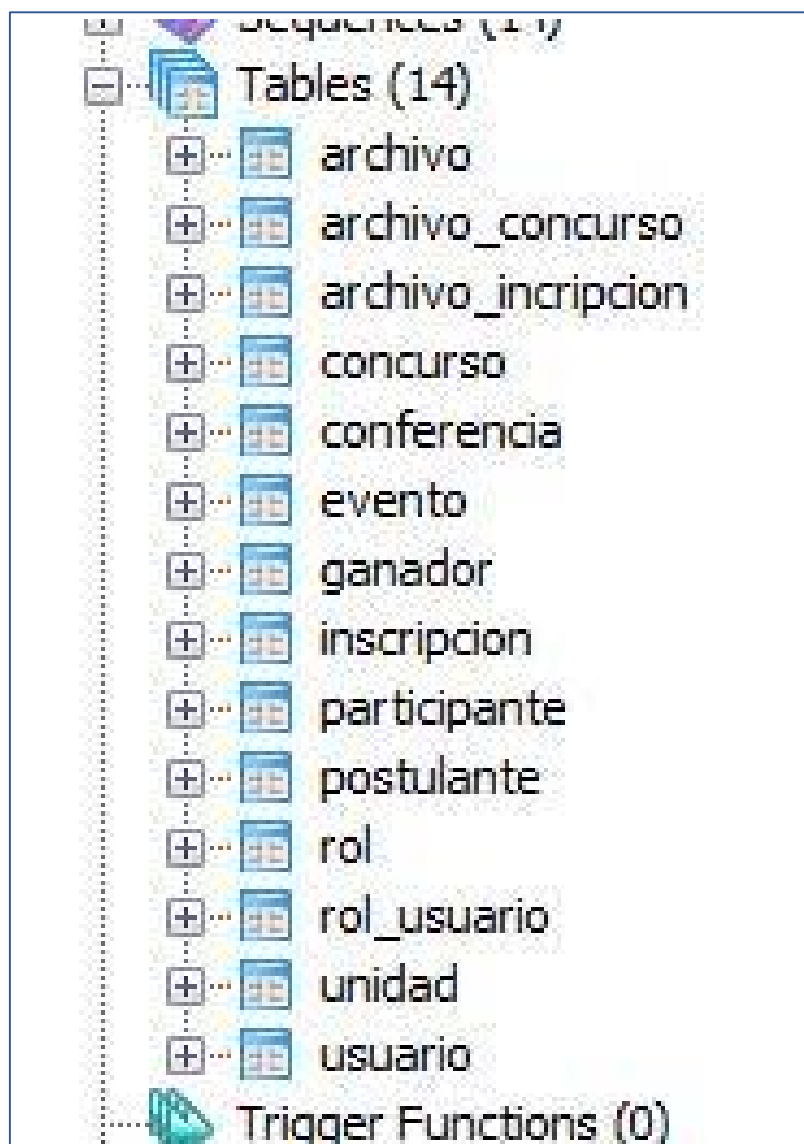
HERRAMIENTA	CONCEPTO	VERSIÓN UTILIZADA
<p><b>pgAdmin</b></p>  <p>PostgreSQL</p>	Es un sistema de gestión de bases de datos relacional	9.5.1
<p><b>NETBEANS</b></p> 	Entorno de desarrollo integrado	NetbeanIDE 8.1
<p><b>COMPONENTE PRIMEFACES</b></p> 	Componentes visuales para JSF	PrimeFaces 5.0
<p><b>JPA</b></p> 	Un mapeo de objeto O/R (ORM)	JPA 2.1
<p><b>SERVIDOR GLASSFISH</b></p> 	Servidor de aplicaciones que implementa la plataforma JavaEE5.	GlassFish 4.0

Fuente: Tatiana Molina, César Mantilla.

## 4.3. DISEÑO

### 4.3.1. BASE DE DATOS

En la base de datos está conformado por 14 tablas para la realización de los módulos del Sistema de Gestión de Eventos.



**Ilustración 32:** Tablas de la Base de Datos CEU

**Fuente:** Tatiana Molina, César Mantilla

#### 4.2.2. 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.

- NOMBRE DE LA TABLA: **archivo**, en esta tabla se almacena la los tipos de archivos que manejará el sistema.

**Tabla 19:** Descripción de la tabla Archivo

NOMBRE DE LA COLUMNA	TIPO DE DATO	CLAVE PRIMARIA	VALORES NULOS	AUTO INCREMENTAL
archivo_id	int	SI	NO	SI
Nombre	varchar	NO	NO	NO
Descripción	varchar	NO	NO	NO
fecha_creacion	timestamp	NO	NO	NO

Fuente: Tatiana Molina, César Mantilla

- NOMBRE DE LA TABLA: **archivo\_concurso**, esta tabla intermedia entre concurso y archivo almacena los archivos de los concursos.

**Tabla 20:** Descripción de la tabla Archivo\_concurso

NOMBRE DE LA COLUMNA	TIPO DE DATO	CLAVE PRIMARIA	VALORES NULOS	AUTO INCREMENTAL
archivo_concurso_id	Int	SI	NO	SI
Archivo	Int	NO	NO	NO
Concurso	Int	NO	NO	NO
Fecha	timestamp	NO	NO	NO

Fuente: Tatiana Molina, César Mantilla

- NOMBRE DE LA TABLA: **archivo\_inscripcion**, esta tabla intermedia entre inscripcion y archivo almacena los archivos de las inscripciones.

**Tabla 21:** Descripción de la tabla archivo\_inscripción

NOMBRE DE LA COLUMNA	TIPO DE DATO	CLAVE PRIMARIA	VALORES NULOS	AUTO INCREMENTAL
archivo_inscripcion_id	Int	SI	NO	SI
archivo	Int	NO	NO	NO
inscripcion	Int	NO	NO	NO
ruta	varchar	NO	NO	NO
Fecha_creacion	timestamp	NO	NO	NO

Fuente: Tatiana Molina, César Mantilla

- NOMBRE DE LA TABLA: **concurso**, es la tabla que permite almacenar datos de concursos.

**Tabla 22:** Descripción de la tabla Concurso

NOMBRE DE LA COLUMNA	TIPO DE DATO	CLAVE PRIMARIA	VALORES NULOS	AUTO INCREMENTAL
<b>concurso_id</b>	Int	SI	NO	SI
<b>Titulo</b>	Varchar	NO	NO	NO
<b>Descripcion</b>	Int	NO	NO	NO
<b>evento_id</b>	Int	NO	NO	NO
<b>Participantes</b>	Int	NO	NO	NO
<b>Lugar</b>	Varchar	NO	NO	NO
<b>Fecha</b>	Timestamp	NO	NO	NO
<b>duracion_minutos</b>	Int	NO	NO	NO
<b>Bases</b>	Varchar	NO	NO	NO
<b>Observaciones</b>	Varchar	NO	NO	NO

**Fuente:** Tatiana Molina, César Mantilla

- NOMBRE DE LA TABLA: **conferencia**, es la tabla que permite almacenar datos de las conferencias.

**Tabla 23:** Descripción de la tabla Conferencia

NOMBRE DE LA COLUMNA	TIPO DE DATO	CLAVE PRIMARIA	VALORES NULOS	AUTO INCREMENTAL
<b>conferencia_id</b>	int	SI	NO	SI
<b>título</b>	varchar	NO	NO	NO
<b>descripcion</b>	varchar	NO	NO	NO
<b>evento_id</b>	Int	NO	NO	NO
<b>expositor_principal</b>	Int	NO	NO	NO
<b>expositor_secundario</b>	Int	NO	NO	NO
<b>numero_maximo_asi</b>	Int	NO	NO	NO
<b>lugar</b>	Varchar	NO	NO	NO
<b>fecha</b>	timestamp	NO	NO	NO
<b>duracion_minutos</b>	Int	NO	NO	NO

**Fuente:** Tatiana Molina, César Mantilla

- NOMBRE DE LA TABLA: **evento**, es la tabla que permite almacenar datos de eventos.

**Tabla 24:** Descripción de la tabla Evento

NOMBRE DE LA COLUMNA	TIPO DE DATO	CLAVE PRIMARIA	VALORES NULOS	AUTO INCREMENTAL
<b>evento_id</b>	Int	SI	NO	SI
<b>título</b>	Varchar	NO	NO	NO
<b>descripcion</b>	Varchar	NO	NO	NO
<b>contacto_responsable</b>	int	NO	NO	NO
<b>fecha_inicio</b>	Timestamp	NO	NO	NO
<b>fecha_fin</b>	Timestamp	NO	NO	NO
<b>informacion_invitacion</b>	Varchar	NO	NO	NO
<b>informacion_lectores</b>	Varchar	NO	NO	NO
<b>informacion_autores</b>	Varchar	NO	NO	NO
<b>requisitos character</b>	Varchar	NO	NO	NO
<b>unidad_organizadora</b>	Int	NO	NO	NO
<b>observaciones</b>	Varchar	NO	NO	NO
<b>fecha_creacion</b>	Timestamp	NO	NO	NO

**Fuente:** Tatiana Molina, César Mantilla

- NOMBRE DE LA TABLA: **ganador**, en esta tabla se almacenará información del ganador del evento.

**Tabla 25:** Descripción de la tabla Ganador

NOMBRE DE LA COLUMNA	TIPO DE DATO	CLAVE PRIMARIA	VALORES NULOS	AUTO INCREMENTAL
<b>ganador_id</b>	int	SI	NO	SI
<b>concurso</b>	int	NO	NO	NO
<b>participante</b>	int	NO	NO	NO
<b>observaciones</b>	varchar	NO	NO	NO

**Fuente:** Tatiana Molina, César Mantilla

- NOMBRE DE LA TABLA: inscripción, es la tabla que almacenará los datos de las inscripciones a los eventos.

**Tabla 26:** Descripción de la tabla Inscripción

NOMBRE DE LA COLUMNA	TIPO DE DATO	CLAVE PRIMARIA	VALORES NULOS	AUTO INCREMENTAL
<b>inscripcion_id</b>	int	SI	NO	SI
<b>Conferencia</b>	Int	NO	NO	NO
<b>Estado</b>	varchar	NO	NO	NO
<b>Fecha</b>	Timestamp	NO	NO	NO
<b>asistente</b>	Int	NO	NO	NO

**Fuente:** Tatiana Molina, César Mantilla

- NOMBRE DE LA TABLA: participante, es la tabla se guarda la información de los participantes de los concursos.

**Tabla 27:** Descripción de la tabla Participante

NOMBRE DE LA COLUMNA	TIPO DE DATO	CLAVE PRIMARIA	VALORES NULOS	AUTO INCREMENTAL
<b>participante_id</b>	Int	SI	NO	SI
<b>Concurso</b>	Int	NO	NO	NO
<b>Fecha</b>	timestamp	NO	NO	NO
<b>Concursante</b>	Int	NO	NO	NO

**Fuente:** Tatiana Molina, César Mantilla

- NOMBRE DE LA TABLA: postulante, almacena datos de los postulantes a evento a realizarse en la UNACH.

**Tabla 28:** Descripción de la tabla Postulante

NOMBRE DE LA COLUMNA	TIPO DE DATO	CLAVE PRIMARIA	VALORES NULOS	AUTO INCREMENTAL
<b>postulante_id</b>	int	SI	NO	SI
<b>Evento</b>	Int	NO	NO	NO
<b>Postulante</b>	Int	NO	NO	NO
<b>Estado</b>	Varchar	NO	NO	NO
<b>Fecha</b>	timestamp	NO	NO	NO
<b>Observaciones</b>	Varchar	NO	NO	NO
<b>Archive</b>	Varchar	NO	NO	NO
<b>tipo_archivo</b>	Int	NO	NO	NO

**Fuente:** Tatiana Molina, César Mantilla



- NOMBRE DE LA TABLA: rol, es la tabla que permite almacenar datos de los roles que administrará el sistema.

**Tabla 29:** Descripción de la tabla Rol

NOMBRE DE LA COLUMNA	TIPO DE DATO	CLAVE PRIMARIA	VALORES NULOS	AUTO INCREMENTAL
<b>rol_id</b>	int	SI	NO	SI
<b>Nombre</b>	varchar	NO	NO	NO
<b>Descripcion</b>	varchar	NO	NO	NO
<b>fecha_creacion</b>	timestamp	NO	NO	NO

**Fuente:** Tatiana Molina, César Mantilla

- NOMBRE DE LA TABLA: rol\_usuario, esta tabla intermedia define la relación entre el usuario, el rol que se le asigna y la unidad a la que el usuario pertenece.

**Tabla 30:** Descripción de la tabla Rol\_usuario

NOMBRE DE LA COLUMNA	TIPO DE DATO	CLAVE PRIMARIA	VALORES NULOS	AUTO INCREMENTAL
<b>rol_usuario_id</b>	int	SI	NO	SI
<b>rol_id</b>	int	NO	NO	NO
<b>usuario_id</b>	int	NO	NO	NO
<b>unidad_id</b>	int	NO	NO	NO
<b>observaciones</b>	varchar	NO	NO	NO

**Fuente:** Tatiana Molina, César Mantilla

- NOMBRE DE LA TABLA: unidad, es la tabla que almacena los datos de la Unidad que organice el Evento crea los eventos.

**Tabla 31:** Descripción de la tabla Unidad

NOMBRE DE LA COLUMNA	TIPO DE DATO	CLAVE PRIMARIA	VALORES NULOS	AUTO INCREMENTAL
<b>unidad_id</b>	int	SI	NO	SI
<b>unidad_principal</b>	varchar	NO	NO	NO
<b>unidad_secundaria</b>	varchar	NO	NO	NO
<b>Descripcion</b>	varchar	NO	NO	NO
<b>unach</b>	boolean	NO	NO	NO

**Fuente:** Tatiana Molina, César Mantilla

- **NOMBRE DE LA TABLA:** usuario, nos permite almacenar datos de los usuarios del Sistema.

**Tabla 32:** Descripción de la tabla Usuario

NOMBRE DE LA COLUMNA	TIPO DE DATO	CLAVE PRIMARIA	VALORES NULOS	AUTO INCREMENTAL
usuario_id	Int	SI	NO	SI
Nombres	varchar	NO	NO	NO
Apellidos	Varchar	NO	NO	NO
Ci	Varchar	NO	NO	NO
Email	Varchar	NO	NO	NO
nombre_usuario	Varchar	NO	NO	NO
Contraseña	varchar	NO	NO	NO
fecha_registro	Timestamp	NO	NO	NO

**Fuente:** Tatiana Molina, César Mantilla

#### 4.2.3. PROTOTIPOS INTERFACES DE USUARIO FINALES

Con la descripción detallada de las historias de los procesos de gestión de eventos de carácter académico tales como: concursos, congresos y ponencias, con la información de los diagramas de procesos podemos definir las interfaces de usuario finales, las cuales serán implantadas en el sistema.



##### **Prototipo 1:** Control de Acceso a Usuarios y Seguridad

**Ilustración 33:** Control de Acceso a Usuarios

**Fuente:** Tatiana Molina, César Mantilla

**Prototipo 2:** Modulo de administración de usuarios y seguridad.

**Tabla 33:** Prototipo del Módulo de Administración a Usuarios y Seguridad

<b>HISTORIA DE USUARIO</b>	
<b>Numero:1</b>	<b>Usuario:</b> Administrador
<b>Nombre historia:</b> Modulo de administración de usuarios y seguridad.	
<b>Prioridad en negocio: Alta</b>	<b>Riesgo en desarrollo:</b> Alto
<b>Esfuerzo: Alto</b>	<b>Iteración asignada: 1</b>
<b>Programador responsable:</b> Tatiana Molina y Eduardo Mantilla.	
<b>Descripción:</b> Antes de iniciar el sistema se requiere el usuario y la contraseña para poder acceder a los módulos de acuerdo al rol de usuario.	
<b>Observaciones:</b> Hay cinco roles de usuarios: Administrador, Secretaria, Jurado, Participante y Asistente, con distintos menús y permisos de acceso dependiendo de las funciones de los usuarios.	
	
<p><b>Módulo de administración de usuarios y seguridad</b></p> 	

**Fuente:** Tatiana Molina, César Mantilla

**Prototipo 3:** Modulo de Gestión de Eventos

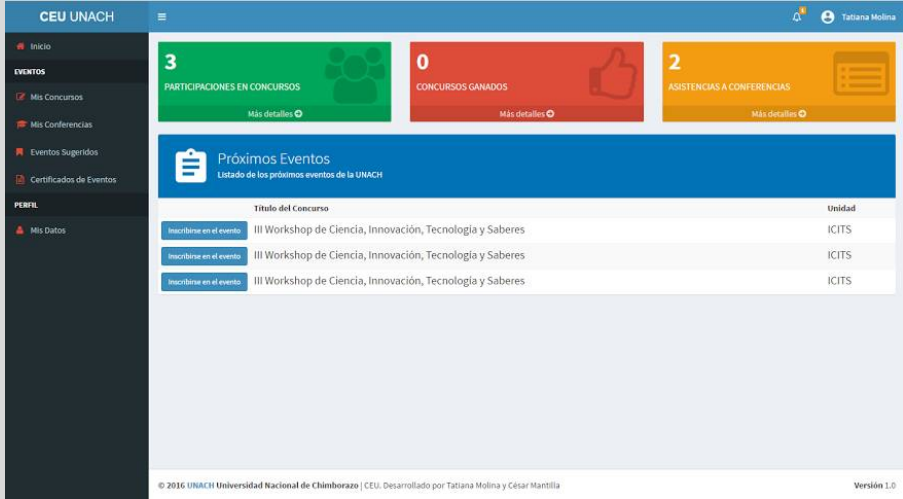
**Tabla 34:** Prototipo del Módulo de Gestión de Eventos

<b>HISTORIA DE USUARIO</b>	
<b>Numero:3</b>	<b>Usuario:</b> Administrador y Secretaria.
<b>Nombre historia:</b> Módulo de Gestión de Eventos	
<b>Prioridad en negocio: Alto</b>	<b>Riesgo en desarrollo:</b> Alto
<b>Esfuerzo: Alto</b>	<b>Iteración asignada:</b> 1
<b>Programador responsable:</b> Tatiana Molina y Eduardo Mantilla.	
<b>Descripción:</b> Difusión y promoción del Evento a realizarse mediante la página web.	
<b>Observaciones:</b> Difusión y promoción del evento se llevará a cabo de medios digitales	
	

**Fuente:** Tatiana Molina, César Mantilla

**Prototipo 4:** Modulo de Información de Eventos

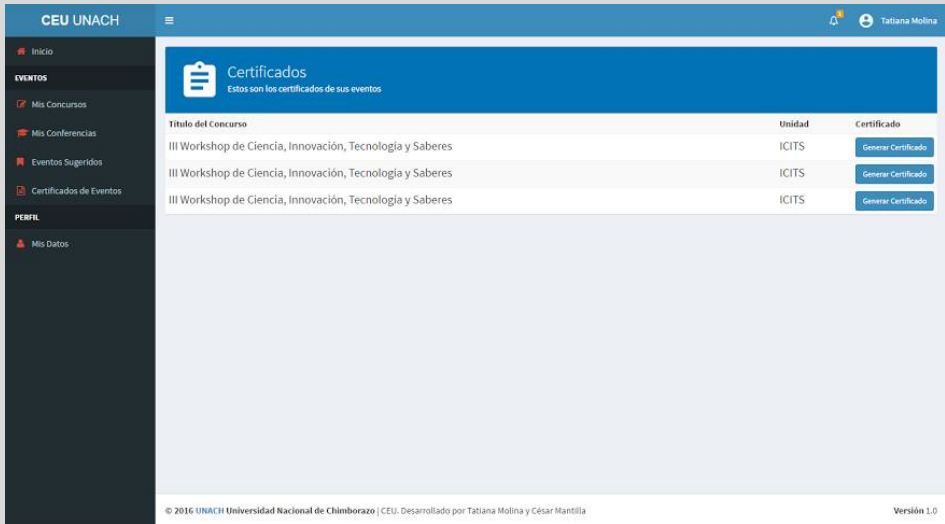
**Tabla 35:** Prototipo del Módulo de Información de Eventos

HISTORIA DE USUARIO	
<b>Numero:</b> 4	<b>Usuario:</b> Administrador
<b>Nombre historia:</b> Módulo de información de Eventos	
<b>Prioridad en negocio:</b> Alto	<b>Riesgo en desarrollo:</b> Alto
<b>Esfuerzo:</b> Alto	<b>Iteración asignada:</b> 1
<b>Programador responsable:</b> Tatiana Molina y Eduardo Mantilla.	
<b>Descripción:</b> Permitirá administrar la información de cada uno de los eventos de diferente índole que se realicen en la Unach, informando a los usuarios de eventos próximos y permitiendo la participación de estos en dichos eventos.	
<b>Observaciones:</b> Cada usuario que crea el evento es el único responsable de dicha información.	
 <p>The screenshot displays the CEU UNACH user interface. At the top, there's a navigation bar with the logo and user name 'Tatiana Molina'. Below it, a dashboard shows three key metrics: '3 PARTICIPACIONES EN CONCURSOS', '0 CONCURSOS GANADOS', and '2 ASISTENCIAS A CONFERENCIAS'. A section titled 'Próximos Eventos' lists upcoming events, including 'III Workshop de Ciencia, Innovación, Tecnología y Saberes' at ICITS. The footer contains copyright information for 2016 UNACH and version 1.0.</p>	

**Fuente:** Tatiana Molina, César Mantilla

## Prototipo 5: Modulo de Reportes

**Tabla 36:** Prototipo del Módulo de Reportes

HISTORIA DE USUARIO	
<b>Numero:5</b>	<b>Usuario:</b> Jurado, Participante, Asistente y Secretaria
<b>Nombre historia:</b> Módulo de reportes	
<b>Prioridad en negocio: Medio</b>	<b>Riesgo en desarrollo:</b> Medio
<b>Esfuerzo: Medio</b>	<b>Iteración asignada: 2</b>
<b>Programador responsable:</b> Tatiana Molina y Eduardo Mantilla.	
<b>Descripción:</b> Permitirá la visualización de resultado de las calificaciones además de la impresión de los respetivos certificados.	
<b>Observaciones:</b> Una vez exportado los reportes en pdf se pueden imprimir.	
	

**Fuente:** Tatiana Molina, César Mantilla

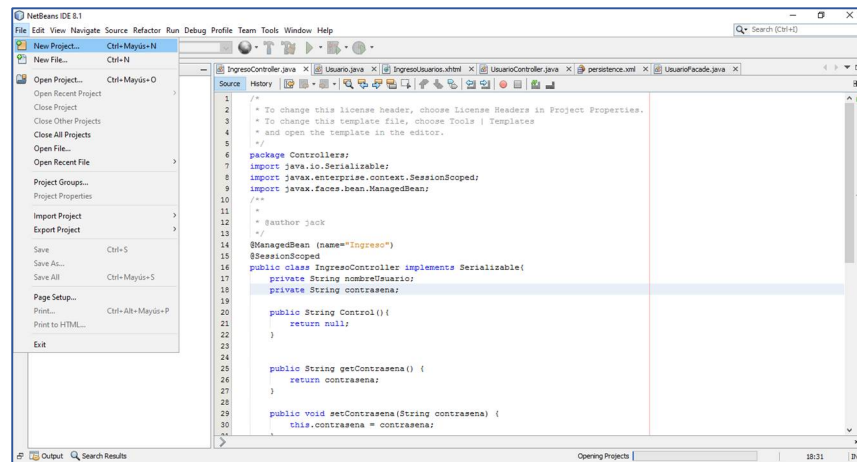
### 4.2.5. CÓDIGO FUENTE

Se ha anexado a este documento el código fuente de la clase Eventos del sistema que se desarrolla debido a que en este código se demuestra la utilidad de la herramienta JPA 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 6)

## 4.4. IMPLEMENTACIÓN

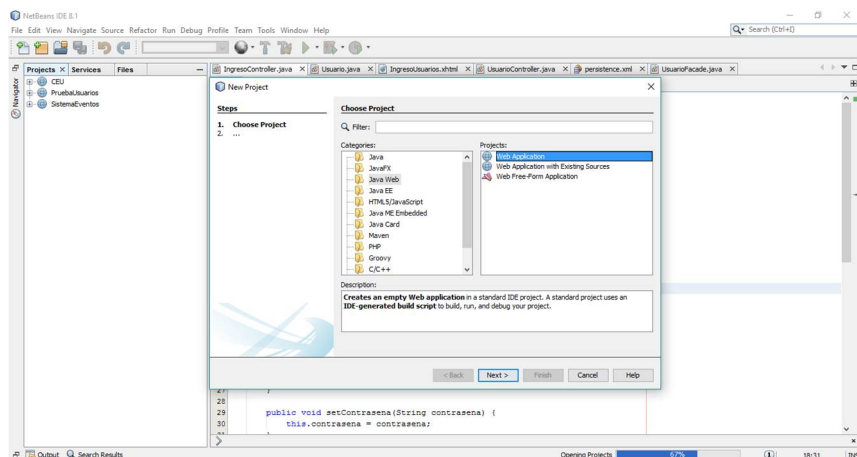
### 4.4.1. INSTALACIÓN

En esta sección se detalla la aplicación que ha sido desarrollada para el Sistema de Gestión de Eventos de la Universidad Nacional de Chimborazoy su respectiva instalación



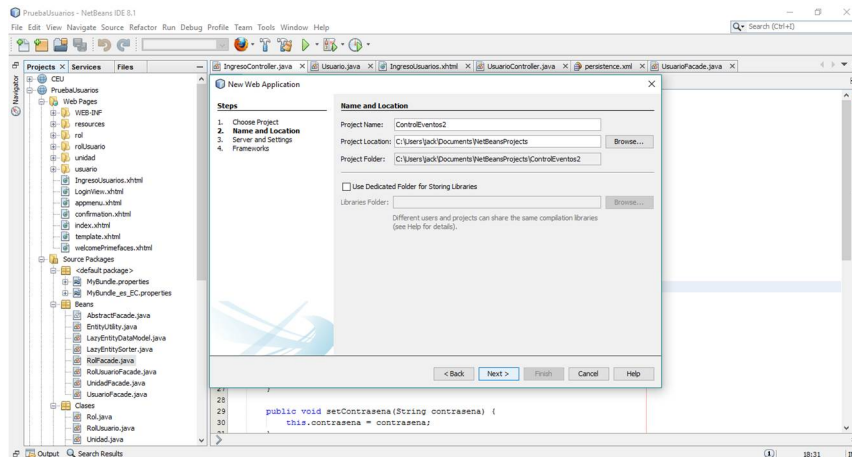
**Ilustración 34:** Creación del proyecto  
**Fuente:** Tatiana Molina, César Mantilla.

Se procede a seleccionar el menú de “Nuevo proyecto” del menú principal de Netbeans.



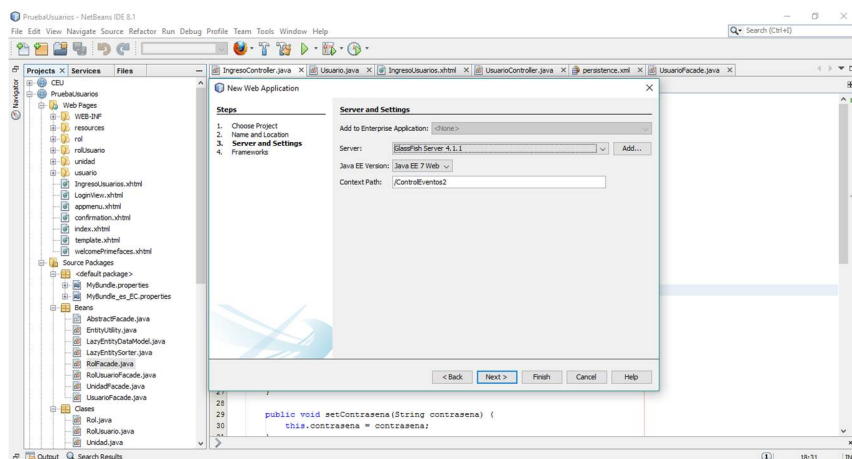
**Ilustración 35:** Selección del tipo de proyecto  
**Fuente:** Tatiana Molina, César Mantilla.

Se selecciona el tipo de proyecto. En el caso de estudio se utilizará Web Application (aplicación web) que se encuentra dentro de la categoría Java Web. Se presiona el botón “Siguiente” a continuación.



**Ilustración 36:** Configurando nombre del proyecto  
**Fuente:** Tatiana Molina, César Mantilla.

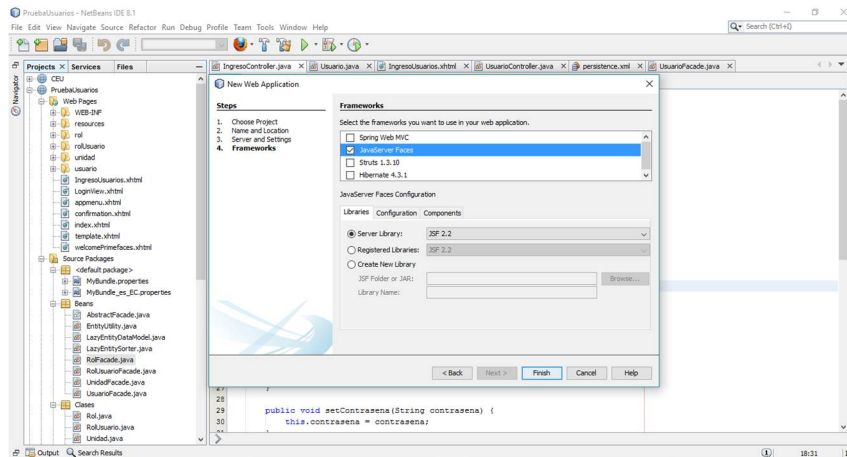
Posteriormente se procede a colocar el nombre que desea darse a toda la solución, la ubicación del proyecto y la carpeta donde se almacenarán los archivos. En este caso se puede apreciar que se coloca el nombre de “ControlEventos2” a la aplicación a desarrollarse. Se presiona el botón “Siguiente” a continuación.



**Ilustración 37:** Configuración del servidor  
**Fuente:** Tatiana Molina, César Mantilla.

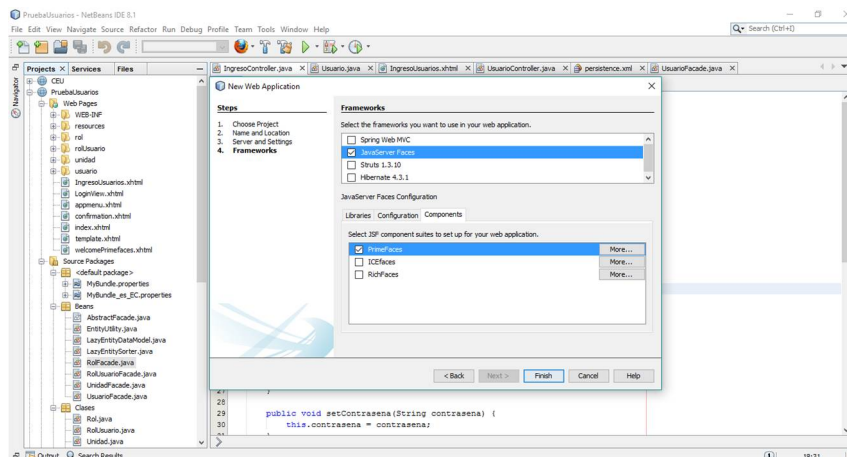
Posteriormente se selecciona el Servidor que utilizará la aplicación, este será Glassfish en su versión 4.1.1 y la versión del Java EE que será la 7. Se presiona el botón “Siguiente”.





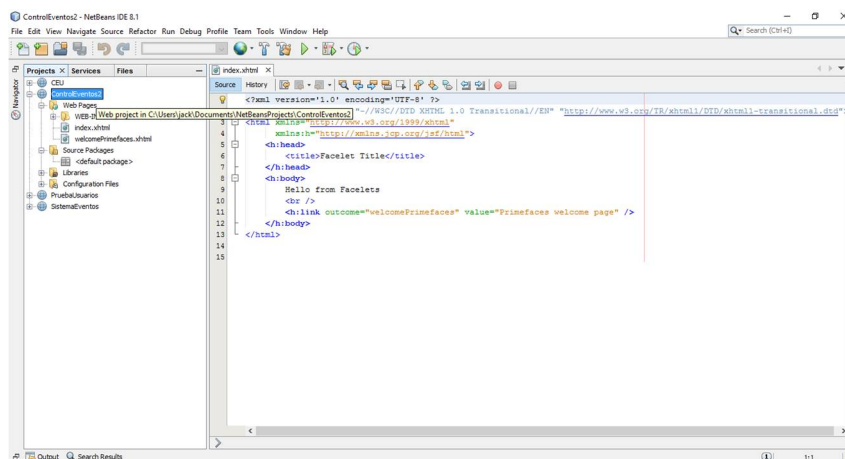
**Ilustración 38:** Con figuración del framework  
**Fuente:** Tatiana Molina, César Mantilla.

El framework a utilizarse se selecciona en la siguiente pantalla del asistente de creación. Java Server Faces será en este caso el framework que soportará las interfaces de usuarios, se tiene que asegurar que se mantenga seleccionada la correcta versión del servidor en la parte inferior de la ventana, es decir, JSF 2.2.



**Ilustración 39:** Activación de los componentes  
**Fuente:** Tatiana Molina, César Mantilla.

En la misma ventana de selección del framework se deben seleccionar los componentes que lo implementarán seleccionando la pestaña de componentes. Se puede apreciar la presencia de los siguientes componentes: Primefaces, ICEfaces y Richfaces. Se seleccionará Primefaces que es la opción por defecto activando la casilla de verificación que lo contiene y se presiona finalizar.



**Ilustración 40:** Resultado de la configuración del proyecto  
**Fuente:** Tatiana Molina, César Mantilla.

Una vez terminada la generación del proyecto se pueden apreciar los siguientes archivos:

Index.html

Este archivo contiene la página principal con un texto base “Hello from Facelets” y un vínculo a la siguiente página “welcomePrimefaces”. Evidentemente esta página tendrá que modificarse por la aplicación en el desarrollo de la misma para plasmar todas las operaciones que esta contenga.

welcomePrimefaces.xhtml

En esta página se puede apreciar información básica sobre el componente Primefaces.

## 5.4.2. FUNCIONALIDAD DEL SISTEMA

En esta sección se definirán las funcionalidades del Sistema de Control de Eventos 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 intenta 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.

- PÁGINA PRINCIPAL DE LA APLICACIÓN.



**Ilustración 41:** Página principal de la aplicación  
**Fuente:** Tatiana Molina, César Mantilla

Esta página está enfocada a brindar información general de los eventos, por lo tanto, no es necesario que el usuario ingrese al sistema para ver dicha información. Cuando el usuario ingresa a la dirección del sistema esta página plantea un resumen de todos los eventos que se realizarán en la Universidad Nacional de Chimborazo. Esta página cuenta con 3 secciones.



**Ilustración 42:** Sección de noticias  
**Fuente:** Tatiana Molina, César Mantilla

La primera sección es el encabezado de la página, que contiene un Carrusel de noticias en el cual se muestran noticias referentes a los eventos realizados en la UNACH. Cuando el usuario realiza un clic sobre algún elemento del Carrusel este le muestra la página correspondiente con información sobre la noticia.



**Ilustración 43:** Página principal de eventos 1  
**Fuente:** Tatiana Molina, César Mantilla

La sección intermedia de la página contiene los eventos que se realizarán en las universidades clasificadas según la unidad que los organiza, el usuario puede realizar un clic sobre el evento para ver una descripción corta del mismo y tendrá acceso a toda la información del evento a través de un botón de más información.



**Ilustración 44:** Página principal de eventos 2  
**Fuente:** Tatiana Molina, César Mantilla

Por último, la página posee en su última sección un listado de todas las unidades que poseen eventos vigentes en la Universidad Nacional de Chimborazo, esto le brinda al usuario la facilidad de encontrar un evento según la unidad que lo organice.

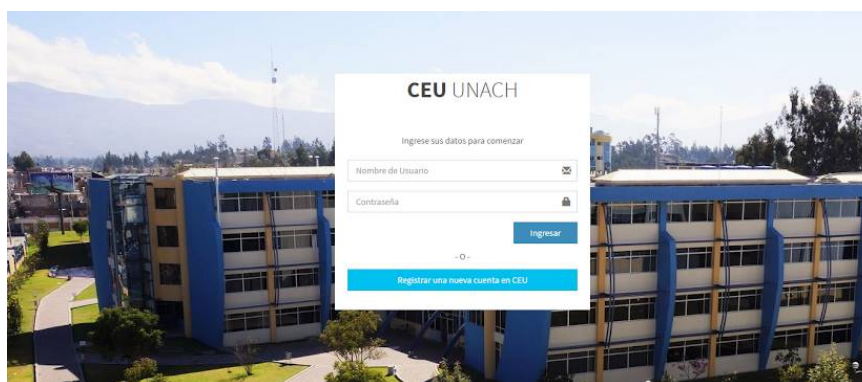
- PÁGINA DE EVENTO.



**Ilustración 45:** Página principal de un evento  
**Fuente:** Tatiana Molina, César Mantilla

Cuando el usuario accede a un evento se muestra la página del mismo. En esta página, se puede observar el título del evento y la unidad que lo organiza en el encabezado de la página, seguido de información del evento como la fecha, participantes, entre otros. Esta página cuenta con un botón que le permite al usuario seleccionar el evento para participar si se encuentra interesado en el mismo siempre y cuando acceda al sistema.

- PÁGINA DE INICIO DE SESIÓN.

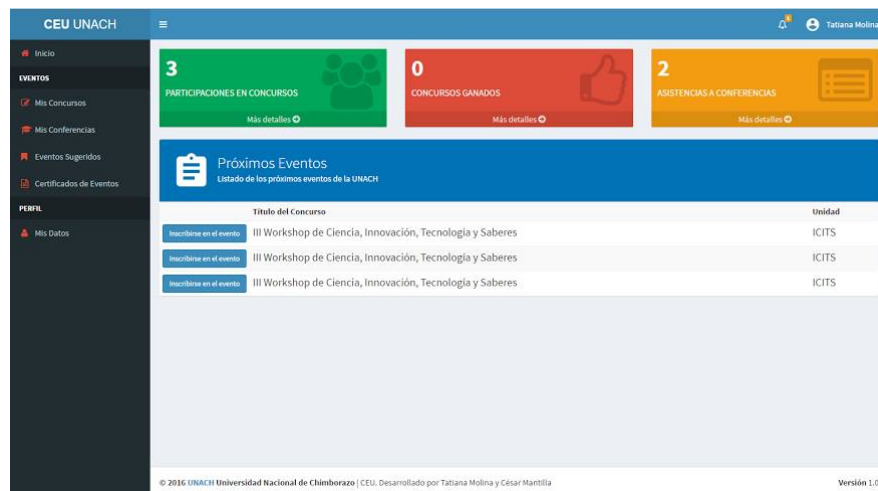


**Ilustración 46:** Página de inicio de sesión  
**Fuente:** Tatiana Molina, César Mantilla

Esta página le permite al usuario acceder a sistema CEU 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.

- **PÁGINA DE INICIO DE USUARIO.**

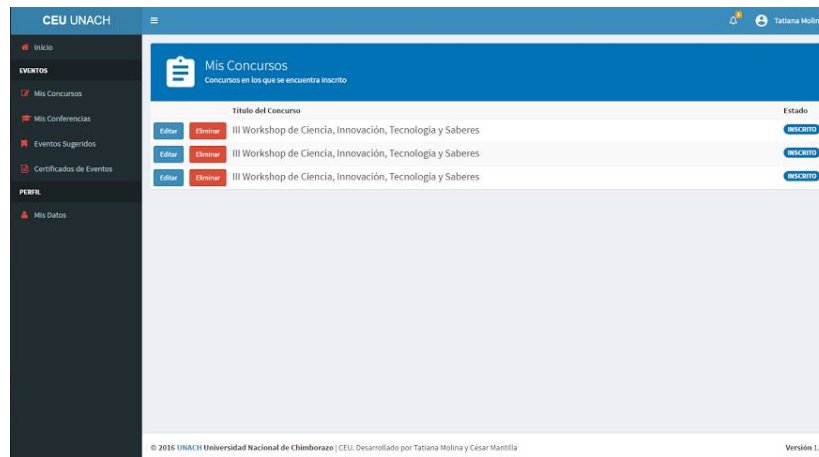


**Ilustración 47:** Página de inicio de usuarios  
**Fuente:** Tatiana Molina, César Mantilla

Esta página es la principal que se le muestra al usuario participante de una conferencia o concursante; en esta página el usuario puede ver unas breves estadísticas sobre el número de participaciones que ha tenido en concursos y conferencias y en cuántas de estas ha sido ganador. Además, se le muestra un breve listado de los siguientes eventos que se llevarán a cabo en la universidad para que él puede acceder a información sobre estos.

En la esquina superior derecha de la página se encuentra el nombre de usuario ingresó en la sesión y la opción de cerrar sesión. En el menú lateral izquierdo se encuentran varias opciones a las que podrá acceder y que le llevarán a otras páginas de administración y que se analizarán a continuación.

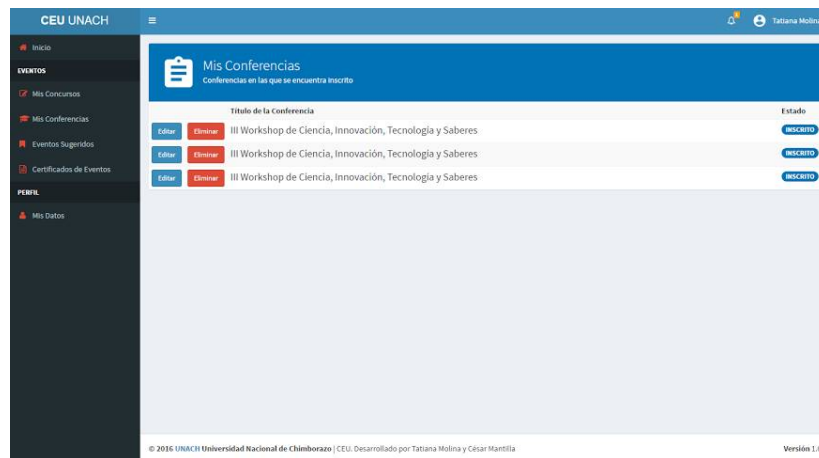
- **PÁGINA MIS CONCURSOS.**



**Ilustración 48:** Página de administración de concursos  
**Fuente:** Tatiana Molina, César Mantilla

Desde esta página el usuario tiene acceso a todos los concursos en los que ha participado y le permite administrar los mismos; brindándole así, la posibilidad de eliminar o editar un concurso en el que previamente debe estar inscrito.

- **PÁGINA MIS CONFERENCIAS.**

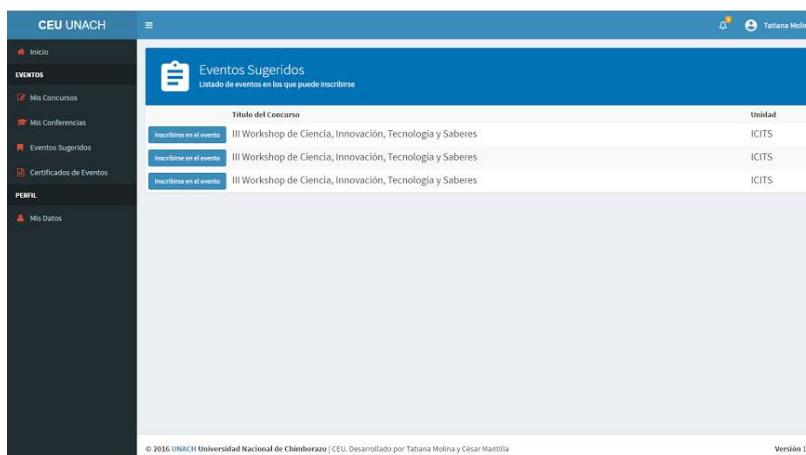


**Ilustración 49:** Página de conferencias  
**Fuente:** Tatiana Molina, César Mantilla

Esta es la página de administración de conferencias del usuario al igual que en la página de concursos desde ésta puede administrar la información de todas las conferencias en la que el usuario se ha inscrito, permitiéndole eliminar y editar esta información.



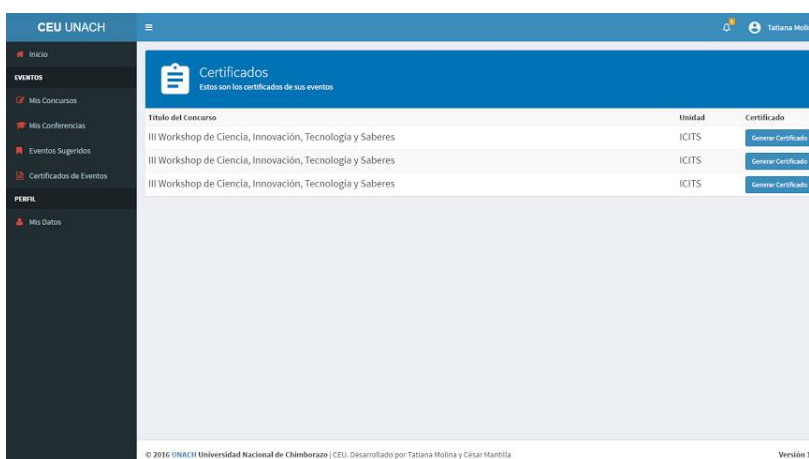
- **PÁGINA EVENTOS SUGERIDOS.**



**Ilustración 50:** Página de eventos sugeridos  
**Fuente:** Tatiana Molina, César Mantilla

Esta es la página de eventos sugeridos, el usuario recibirá invitaciones de eventos a los que el creador de un evento crea que el usuario debe asistir. Esta página le facilita al usuario la búsqueda y participación de eventos ya que recibirá automáticamente invitaciones a eventos que estén acordes a su carrera, unidad, o que simplemente sean de su interés. Esta página le brindará las opciones de inscribirse al evento a través de un botón de participación que se encuentra en cada ítem del listado.

- **PÁGINA DE CERTIFICADOS.**



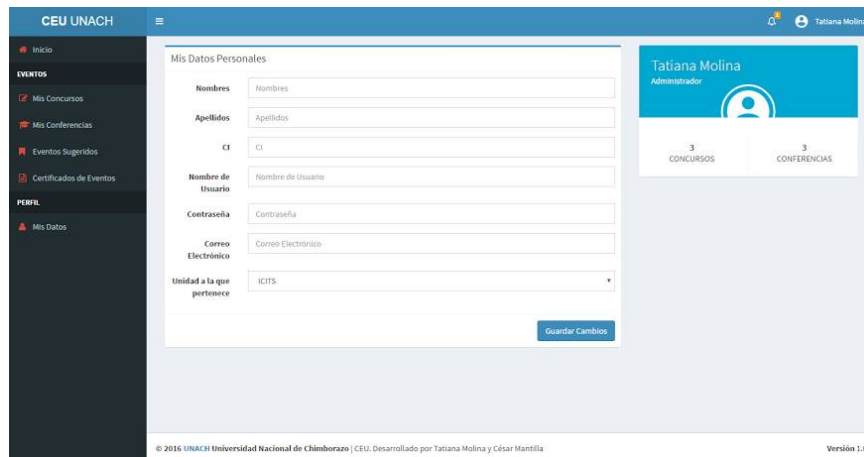
**Ilustración 51:** Página de generación de certificados  
**Fuente:** Tatiana Molina, César Mantilla



En la página mis certificados el usuario puede acceder a un listado de todos los certificados de los eventos en los que ha participado facilitándole de esta manera encontrar y descargar en un solo lugar toda información de certificados.

Estos certificados han sido generados con herramientas especializadas para reportes como Jasper Reports.

- **PÁGINA DE DATOS PERSONALES.**

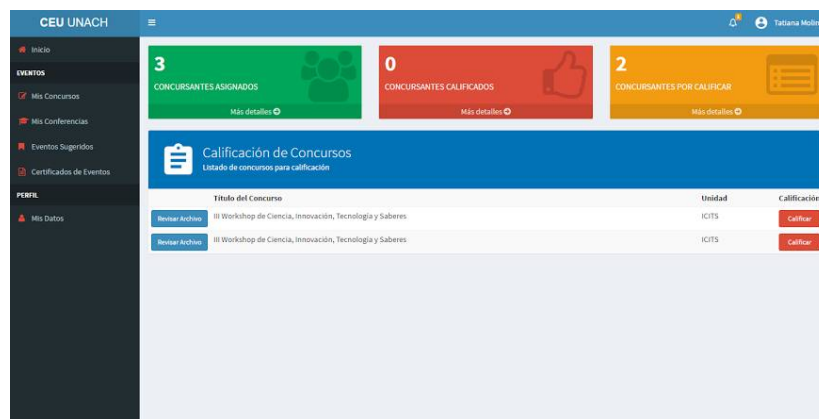


**Ilustración 52:** Administración de datos de los usuarios

**Fuente:** Tatiana Molina, César Mantilla

A través de esta página el usuario puede administrar su información personal, sus nombres de usuario, su cédula identidad, la unidad a la que pertenece, entre otros. La información que el usuario provea en esta página será utilizada para generar sus reportes y para sus inscripciones en cada uno de los eventos en los que participe.

- **PÁGINA DE CALIFICACIÓN DE JURADO.**

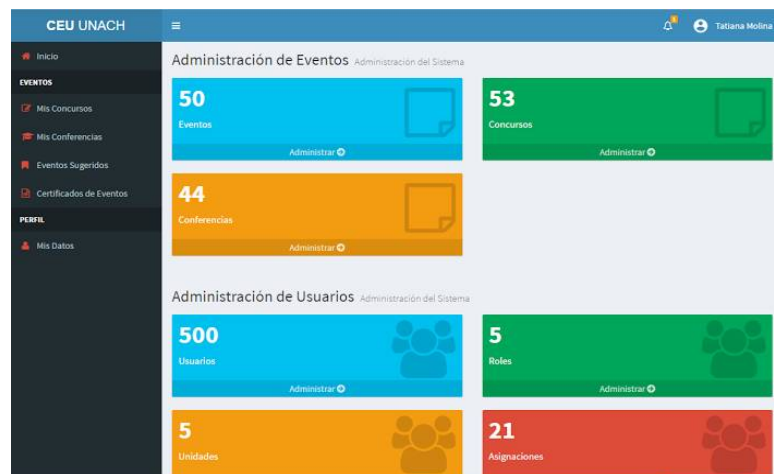


**Ilustración 53:** Página de calificación del jurado

**Fuente:** Tatiana Molina, César Mantilla

Está en la página de calificación de concurso. A esta página sólo pueden acceder usuarios que tengan el rol de Jurado y además sólo podrán acceder para calificar a los concursantes que se les hayan sido asignados por un administrador con más permiso cómo una secretaria o el administrador del sistema. El Jurado visualizará un encabezado con breves estadísticas sobre la cantidad de concursantes que se les han asignado, los concursantes que ha calificado y los que faltan por calificar; a continuación, podrá visualizar un listado con dichos concursantes y archivos de cada uno de ellos que tendrá que revisar para otorgar una calificación.

- **PÁGINAS DE ADMINISTRACIÓN.**



**Ilustración 54:** Página de Administración  
**Fuente:** Tatiana Molina, César Mantilla

Estas son las páginas de administración de sistema. Estas páginas poseen toda la información que administra sistema sin restricción alguna. Desde estas páginas se puede acceder a las funciones de selección, eliminación y edición de los datos; además de capacidades de generación de reportes. A estas páginas sólo puede acceder el administrador del sistema con todos los permisos de usuario ya que este tendrá incluso la libertad de visualizar y editar la información que los demás usuarios hayan ingresado.

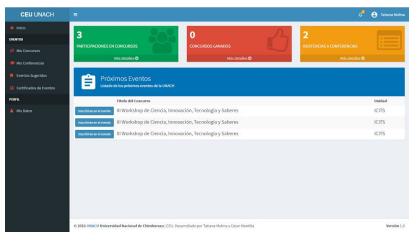
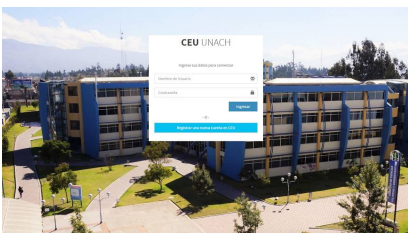
## 4.5. PRUEBAS

Las pruebas son muy importantes 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 aceptación, las que sean necesarias 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

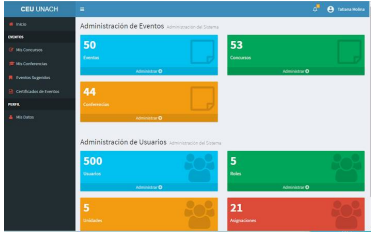

**Tabla 37:** Prueba 1

Fecha	Descripción	Autores
15/01/2016	Pruebas	Tatiana Molina y Eduardo Mantilla.
<b>Módulo de administración de usuarios y seguridad.</b>		
<b>Descripción</b>	Hay cinco tipos de usuarios: Administrador, Secretaria, Jurado, Participante y Asistente.	
<b>Condiciones de Ejecución.</b>	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.	
<b>Entrada</b>	<ul style="list-style-type: none"> <li>▪ El usuario ingresa su usuario y contraseña</li> <li>▪ El proceso de control de acceso a Usuarios finaliza.</li> </ul>	
<b>Resultado Esperado</b>	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.	
<b>Evaluación de la Prueba</b>	<b>Exitosa</b> 	<b>Fallida</b> 

**Fuente:** Tatiana Molina, César Mantilla

▪ PRUEBA 2



**Tabla 38:** Prueba 2

Fecha	Descripción	Autores
15/01/2016	Pruebas	Tatiana Molina y Eduardo Mantilla
<b>Módulo de Gestión de Eventos</b>		
<b>Descripción</b>	Hay dos tipos de usuarios: Administrador, Secretaria.	
<b>Condiciones de Ejecución.</b>	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.	
<b>Entrada</b>	<ul style="list-style-type: none"> <li>▪ Secretaria ingresa toda la información correspondiente al evento a realizarse.</li> <li>▪ Administrador gestiona toda la información acerca de los eventos.</li> </ul>	
<b>Resultado Esperado</b>	Secretaria registra el evento a realizarse con toda la información que se solicita en el sistema.	
<b>Evaluación de la Prueba</b>	<b>Exitosa</b>	<b>Fallida</b>
		

**Fuente:** Tatiana Molina, César Mantilla

▪ PRUEBA 3

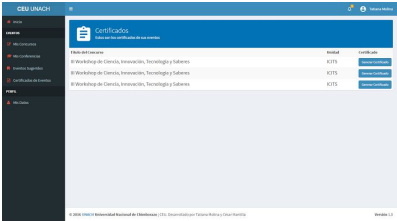

**Tabla 39:** Prueba 3

Fecha	Descripción	Autores	
15/01/2016	Pruebas	Tatiana Molina y Eduardo Mantilla	
Módulo de Información			
<b>Descripción</b>	Hay uno tipo de usuario: Administrador.		
<b>Condiciones de Ejecución.</b>	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.		
<b>Entrada</b>	<ul style="list-style-type: none"> <li>Administrador permite por medio del sistema web la difusión y promoción del evento a realizarse.</li> </ul>		
<b>Resultado Esperado</b>	Dar a conocer los eventos a realizarse en la UNACH.		
<b>Evaluación de la Prueba</b>	<b>Exitosa</b>	<b>Fallida</b>	
			

**Fuente:** Tatiana Molina, César Mantilla

▪ PRUEBA 4

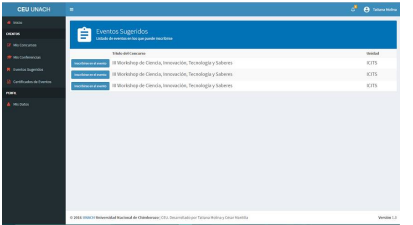

**Tabla 40:** Prueba 4

Fecha	Descripción	Autores
15/01/2016	Pruebas	Tatiana Molina y Eduardo Mantilla
<b>Módulo de Reportes</b>		
<b>Descripción</b>	El Administrador, Secretaria, Participante, Asistente y Jurado depende de la actividad que requiera podrán exportar un reporte.	
<b>Condiciones de Ejecución</b>	Administrador, Secretaria, Participante, Asistente y Jurado debe constar en la base de datos del sistema para poder generar un reporte.	
<b>Entrada</b>	<ul style="list-style-type: none"> <li>El Administrador, Secretaria, Participante, Asistente y Jurado una vez que ingresa al sistema escoge la opción de descargar pdf.</li> </ul>	
<b>Resultado Esperado</b>	Que los participantes de un concurso obtengan mediante la página web sus certificados	
<b>Evaluación de la Prueba</b>	<b>Exitosa</b>	<b>Fallida</b>
		

**Fuente:** Tatiana Molina, César Mantilla

▪ PRUEBA 5

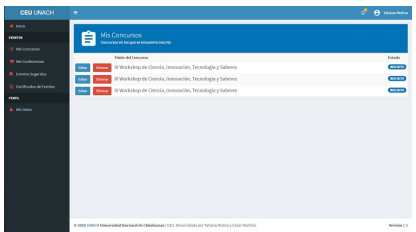

Tabla 41: Prueba 5

Fecha	Descripción	Autores
15/01/2016	Pruebas	Tatiana Molina y Eduardo Mantilla
<b>Módulo de Gestión de Eventos</b>		
<b>Descripción</b>	El Participante y Asistente ingresan al sistema para verificar que se les notifique de eventos sugeridos.	
<b>Condiciones de Ejecución</b>	Participante y Asistente debe constar en la base de datos del sistema para poder ingresar al sistema.	
<b>Entrada</b>	<ul style="list-style-type: none"> <li>El Participante y Asistente una vez que ingresa al sistema elige el menú eventos sugeridos.</li> </ul>	
<b>Resultado Esperado</b>	Que los participantes de un concurso y asistentes de conferencias obtengan mediante la página web a sugerencia de eventos.	
<b>Evaluación de la Prueba</b>	<b>Exitosa</b>	<b>Fallida</b>
		

Fuente: Tatiana Molina, César Mantilla

▪ PRUEBA 6

**Tabla 42:** Prueba 6

Fecha	Descripción	Autores
15/01/2016	Pruebas	Tatiana Molina y Eduardo Mantilla
<b>Módulo de Gestión de Eventos</b>		
<b>Descripción</b>	El Participante ingresa al sistema para administrar los concursos a los que están inscritos.	
<b>Condiciones de Ejecución</b>	Participante debe constar en la base de datos del sistema para poder ingresar al sistema.	
<b>Entrada</b>	<ul style="list-style-type: none"> <li>▪ El Participante una vez que ingresa al sistema elige el menú mis concursos.</li> </ul>	
<b>Resultado Esperado</b>	Que los participantes de un concurso obtengan mediante la página web la opción de ver los eventos a los que están inscritos.	
<b>Evaluación de la Prueba</b>	<b>Exitosa</b>	<b>Fallida</b>
		

**Fuente:** Tatiana Molina, César Mantilla



## CONCLUSIONES

A través del estudio realizado se puede concluir lo siguiente:

- JPA mejora significativamente la productividad comparado con la generación manual de persistencia que ofrece JDBC. Sin embargo, se debe tomar en cuenta que la persistencia generada automáticamente ingresa en la aplicación código que el programador no ha generado y que deberá aprender a usar, lo que implica una curva de aprendizaje que no en todos los casos podría ser leve. También se debe entender que, aunque se implemente JPA, se debe en algunos casos implementar también de manera conjunta JDBC para consultas complejas que no se ven integradas en la persistencia automática.
- Los resultados del estudio demuestran que JPA es un 84,86% más productivo en desarrollo de aplicaciones web en comparación con JDBC, en los apartados que se detallan a continuación.
- JPA implementa un menor número de líneas de código 654, en comparación con 1959 que utiliza JDBC, esto indica una productividad comparada del 66,62%. Un menor número de líneas de código indica que hay menos cantidad de instrucciones que ejecutar, por lo que la ejecución es más rápida; también, se debe señalar que estas líneas no son en su mayoría escritas por el programador, sino generadas automáticamente.
- JPA también utiliza un menor número de tiempo de desarrollo 248 horas, en comparación con JDBC que utiliza 912, durante todo el desarrollo del proyecto. Lo que implica una productividad de 72, 81%. Este parámetro es el más importante en cuanto a la medición de la productividad, ya que determina que el producto final se ha realizado en menos tiempo.
- JPA emplea un menor número de funciones en la base de datos. Se emplearon 0 funciones en JPA y 20 en JDBC. Este parámetro medido demuestra el poder de JPA sobre JDBC en relación al trabajo que se tiene que realizar en el sistema gestor de base de datos.
- JDBC permite el manejo directo y transparente de las clases en Java. Si bien es cierto, esto facilita que el programador tenga un contacto más amplio e inteligible de cada una de sus líneas de código, también implica que la

productividad de la aplicación se vea afectada ya que el programador tiene que escribir todo lo que quiera programar, además, se verá en la obligación de implementar consultas en el sistema manejador de base de datos y manejar a profundidad un lenguaje de consulta como SQL o MySQL.

## **RECOMENDACIONES**

- Si lo que se busca es mejorar al máximo la productividad, no basta con implementar una herramienta de generación de persistencia automática, se debe planificar el tiempo adicional que llevará manejar esta herramienta para adaptarla a las necesidades del proyecto.
- Es altamente recomendable la utilización de JPA cuando se pretende desarrollar un sistema con muchas clases y se posee poco tiempo para su desarrollo. Además que el equipo de desarrollo debe tener un nivel de conocimientos intermedio-avanzado y estar integrado por una cantidad pequeña de personas.
- JDBC puede ser muy útil en ambiente de producción, cuando se poseen pocas clases para implementar y un nivel de conocimientos de programación básico; además, el aprendizaje de JDBC es imprescindible para todo desarrollador web de Java, ya que en este se entienden y aplican conceptos básicos necesarios del paradigma de programación y por último la utilización de JPA no necesariamente evita que se emplee JDBC en algún momento en la aplicación.

## BIBLIOGRAFÍA

- (Álvarez Caules, C. (10 de Septiembre de 2014). *JEE, JPA*. Obtenido de <http://www.ibm.com/>:  
[http://www.ibm.com/support/knowledgecenter/SSAW57\\_8.5.5/com.ibm.wabsphere.nd.doc/ae/cejb\\_persistence.html?lang=es](http://www.ibm.com/support/knowledgecenter/SSAW57_8.5.5/com.ibm.wabsphere.nd.doc/ae/cejb_persistence.html?lang=es)
- Ángel Esteban. (2000). Acceso a Base de Datos con Java JDBC 2.0. En Á. Esteban, *Acceso a Base de Datos con Java JDBC 2.0* (pág. 217). Madrid: Grupo Eidos.
- Bauer, C., & King, G. (2004). *Practical Object/Relational Mapping*. Manning Publications.
- Berl, A. (12 de Julio de 2015). *LinesOfCodeWichtel's Homepage*. Obtenido de LinesOfCodeWichtel's Homepage: <http://www.andreas-berl.de/linesofcodewichtel/>
- Blancarte, O. (15 de Julio de 2014). *www.oscarblancarteblog.com*. Obtenido de PA, Hibernate VS JDBC: <http://www.oscarblancarteblog.com/2014/07/15/jpa-hibernate-vs-jdbc/>
- Carmen Gonzales. (21 de 01 de 2016). <http://java.sys-con.com/>. Obtenido de <http://java.sys-con.com/node/140123>
- Carrillo, A. (2011). *Metodología RUP de Ingeniería de Software*. España: Academia Española.
- Coad, P., & Yourdon, E. (1991). *Object-oriented Design*. Prentice-Hall International.
- Colque, D. C., & Valdivia, R. P. (2006). Integración de tecnologías en una plataforma J2EE dirigida por modelos. *Ingeniare: Revista Chilena de Ingeniería*, 47-57.

- Deitel, H. M., & Deitel, P. J. (2004). *Java. How to Program*. Prentice-Hall.
- Ferri Pérez, F. (25 de Febrero de 2011). *tutoriales*. Obtenido de <http://www.adictosaltrabajo.com/tutoriales/tutorial-basico-jdbc/>
- García Ruiz, F. (2010). *Estudio, Comparativa y Aplicación Práctica de Metodologías de Desarrollo de Aplicaciones Web en Java*. Cataluña, España: Universitat Oberta de Catalunya.
- Groussard, T. (2010). *Java Enterprise Edition. Desarrollo de Aplicaciones Web con JEE*. Editions ENI.
- J., A. A. (s.f.). *Midiendo la Productividad del Desarrollo de Aplicaciones*. New York, USA: IBM Corporation.
- Keith, M., & Schincariol, M. (2006). *Pro EJB 3: Java Persistence API*. Apress.
- Lujan Mora, S. (2002). *Programación de Aplicaciones Web*. España: Editorial Club Universitario.
- Mateu, C. (2004). Desarrollo de Aplicaciones Web. En C. Mateu, *Desarrollo de aplicaciones web* (pág. 377). Catalunya: Fundació para la Universitat Oberta de Catalunya.
- Monteagudo, J. L. (13 de Agosto de 2014). <http://www.jlmonteagudo.com>. Obtenido de <http://www.jlmonteagudo.com>: <http://www.jlmonteagudo.com/2014/06/desarrollo-de-aplicaciones-web-modernas-mean-angularjs/>
- Oracle Corporation. (Enero de 2013). *Entities-The Java EE 6 Tutorial*. Obtenido de The Java EE 6 Tutorial : <http://docs.oracle.com/javaee/6/tutorial/doc/docinfo.html>
- Páez Martínez, F. J. (04 de Mayo de 2006). <http://www.adictosaltrabajo.com/>. Obtenido de Introducción a JDBC: <http://www.adictosaltrabajo.com/tutoriales/introjdb/>

- Payá Martín, A. (26 de 01 de 2016). *http://www.comillas.edu/*. Obtenido de <http://www.iit.upcomillas.es/pfc/resumenes/450955e7368ca.pdf>
- Pech May, F. A. (2010). *Desarrollo de Aplicaciones web con JPA, EJB, JSF y PrimeFaces*. Tabasco, México: Instituto Tecnológico Superior de los Ríos .
- programacionextrema. (28 de 01 de 2016). *http://programacionextrema.tripod.com*. Obtenido de <http://programacionextrema.tripod.com/fases.htm>
- Reyes Freire, T. A. (26 de 01 de 2016). *http://www.utn.edu.ec/*. Obtenido de <http://repositorio.utn.edu.ec/bitstream/123456789/571/1/Tesis.pdf>
- Sánchez, J. (15 de 1 de 2016). *Principios sobre bases de datos relacionales*. Obtenido de Principios sobre bases de datos relacionales: <http://cursa.ihmc.us/rid=1H73QYLH3-6LFRCX-JT6/bdrelacional.pdf>
- Saney, N. (2001). Su Primer Programa Java. En N. Saney, *Su Primer Programa Java* (pág. 112). S.A. MARCOMBO.
- Serna, C. N. (2011). *Tutorial Avanzado de JPA*. Mexico.
- Silberschatz, A., Korth, H. F., & Sudarshan, S. (2002). *Fundamentos de bases de datos*. McGraw-Hill.
- Sperko, R. (2003). *Java Persistence for Relational Databases*. Apress.
- Taylor, A. (1998). *JDBC Developer's Resource*. New York: Prentice-Hall, Inc.
- Visconti, M. (2010). Ingeniería de Software Avanzada. *Documento Digital*. Valparaiso, Chile: Universidad Técnica Federico Santa María.
- Yang, D. (2010). *Java Persistence with JPA*. The ACM Digital Library.

## GLOSARIO DE TÉRMINOS

**SISTEMA OPERATIVO.-** Conjunto de órdenes y programas que controlan los procesos básicos de una computadora y permiten el funcionamiento de otros programas.

**SOFTWARE.-** Conjunto de programas y rutinas que permiten a la computadora realizar determinadas tareas.

**LENGUAJES DE PROGRAMACIÓN.-** Es un lenguaje formal diseñado para realizar procesos que pueden ser llevados a cabo por máquinas como las computadoras.

**JAVA.-** Es un lenguaje de programación orientado a objetos que se popularizó a partir del lanzamiento de su primera versión comercial de amplia difusión.

**IBM.-** Es una empresa multinacional estadounidense de tecnología, que fabrica y comercializa hardware y software para computadoras.

**FRAMEWORK.-** Es un entorno o ambiente de trabajo para desarrollo; dependiendo del lenguaje normalmente integra componentes que facilitan el desarrollo de aplicaciones como el soporte de programa, bibliotecas, plantillas y más.

**API.-** Es el conjunto de subrutinas, funciones y procedimientos, que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

**JPA.-** Es un framework del lenguaje de programación Java que maneja datos relacionales en aplicaciones usando la Plataforma Java en sus ediciones Standard (Java SE) y Enterprise (Java EE).

**JDBC.-** Es una API que ofrece la posibilidad de ejecutar operaciones en las bases de datos utilizando lenguaje Java y abstrayendo la base de datos y el sistema operativo y evitando su dependencia

**MAPEO.-** Es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional como motor de persistencia.

**PARAMETROS:** Los indicadores son variables que intentan medir u objetivar en forma cuantitativa o cualitativa para así, poder respaldar para evaluar logros y metas.

La OMS los ha definido como "variables que sirven para medir los cambios".

**BASE DE DATOS.-** Una base de datos es una herramienta para recopilar y organizar información. En las bases de datos, se puede almacenar información sobre personas, productos, pedidos o cualquier otra cosa.

**INGENIERÍA DE SOFTWARE.-** Es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software, y el estudio de estos enfoques, es decir, la aplicación de la ingeniería al software.

**Visual Studio.-** Es un entorno de desarrollo integrado que nos permite a los desarrolladores crear aplicaciones (Web, Escritorio), Aplicaciones para SmartPhone, Pocket PC, servicios web y otras utilidades.

**ECLIPSE.-** Es una plataforma de software compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar lo que el proyecto

**NETBEANS.-** Es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java

**TOPLINK.-** Es un paquete de Mapeo objeto-relacional (ORM) para desarrolladores Java. Provee un marco de trabajo para almacenar objetos Java en una base de datos relacional, o convertir objetos Java a archivos XML.

**XML.-** Se trata de un lenguaje estándar que posee una Recomendación del World Wide Web Consortium.

**METADATOS.-** Son datos altamente estructurados que describen información, describen el contenido, la calidad, la condición y otras características de los datos. Es "Información sobre información" o "datos sobre los datos".

**ENTITYMANAGER.-** Es la interfaz principal de JPA utilizada para la persistencia de las aplicaciones.

**SERIALIZACIÓN.-** Es un mecanismo ampliamente usado para transportar objetos a través de una red, para hacer persistente un objeto en un archivo o base de datos, o para distribuir objetos idénticos a varias aplicaciones o localizaciones.

**CONTEXTUALIZAR:** Se denomina contextualizar al hecho de poner una circunstancia, hecho o discurso en relación con el entorno en que se generó.

**CUANTITATIVAMENTE:** se utiliza para referirse a la investigación o análisis que toma en cuenta variables medibles o cuantificables con el fin de establecer estadísticas.

# **ANEXOS**



## **ANEXO 1: REQUERIMIENTOS DEL SISTEMA**

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

- Promoción y Difusión de los Eventos Académicos de la UNACH.
- Información de los Eventos tales como:
  - Concursos
  - Congresos
  - Ponencias
- Módulo de Inscripción a los Eventos.
- Resultados de la Calificación obtenida por el Jurado calificador.
- Permitir la obtención de Certificados en el caso de los Concursos.

De acuerdo a los requerimientos expuestos, el sistema se delimitará en Gestionar los Eventos 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 ADMINISTRACIÓN DE EVENTOS**

- Permitirá la Difusión y Promoción del Evento a realizarse mediante la página Web

- **MÓDULO DE INFORMACIÓN**

Permitirá administrar la información de cada uno de los eventos de diferente índole que se realicen en la UNACH, informando a los usuarios de eventos próximos y permitiendo la participación de éstos en dichos eventos.

- **MÓDULO DE REPORTES**

Permitirá la visualización de Resultados de las calificaciones además de la impresión de los respectivos Certificados

## **ANEXO 2: HOJA DE ENCUESTAS A PROFESIONALES CON EXPERIENCIA EN EL DESARROLLO DE APLICACIONES JAVA.**

La siguiente encuesta está orientada a profesionales con experiencia en desarrollo de aplicaciones Java Web utilizando JPA y JDBC. Tiene como intención recabar información que demuestre cuál de las dos tecnologías antes mencionadas ofrece más productividad en el ambiente de desarrollo.

1.- ¿Cómo describiría su nivel de conocimientos y experiencia en el desarrollo de aplicaciones Java Web?

- BAJO
- INTERMEDIO
- AVANZADO

2. ¿Cuál de las siguientes tecnologías prefiere para desarrollar aplicaciones Java Web?

- JPA
- JDBC

3. ¿Cuál es la razón principal por la que prefiere la tecnología anterior?

- Menos tiempo de desarrollo
- Menos tiempo de código
- Más documentación de ayuda
- Menos consultas codificadas en el sistema Gestor de Base de Datos
- Más compatibilidad de las aplicaciones desarrolladas con varios navegadores y sistemas operativos

4. ¿Cuál de las siguientes tecnologías considera que utiliza menos tiempo en el desarrollo de aplicaciones?

- JPA
- JDBC

5. ¿Cuál de las siguientes tecnologías considera que utiliza menos líneas de código en el desarrollo de aplicaciones?

- JPA
- JDBC

6. ¿Cuál de las siguientes tecnologías considera que ofrece más documentación de ayuda?

- JPA
- JDBC

7. ¿Cuál de las siguientes tecnologías considera que utiliza menos consultas en el sistema gestor de base de datos?

- JPA
- JDBC

8. ¿Cuál de las siguientes tecnologías considera que ofrece más compatibilidad en navegadores y sistemas operativos?

- JPA
- JDBC

## ANEXO 3: USO DE LA HERRAMIENTA GOOGLE FORMS PARA LA APLICACIÓN DE LAS ENCUESTAS

### ANÁLISIS COMPARATIVO DE LA PRODUCTIVIDAD EN EL DESARROLLO DE APLICACIONES WEB UTILIZANDO LAS TECNOLOGÍAS JDBC Y JPA

La siguiente encuesta está orientada a profesionales con experiencia en desarrollo de aplicaciones Java Web utilizando JPA y JDBC. Tiene como intención recabar información que demuestre cuál de las dos tecnologías antes mencionadas ofrece más productividad en el ambiente de desarrollo. La encuesta tiene 8 preguntas puntuales de opción múltiple, todas son obligatorias. No le tomará más de 3 minutos. Gracias por su amable cooperación.

\*Obligatorio

Universidad Nacional de Chimborazo

Facultad de Ingeniería. Carrera de Ingeniería en Sistemas y Computación.  
Encuesta desarrollada por Tatiana Molina y Eduardo Mantilla.

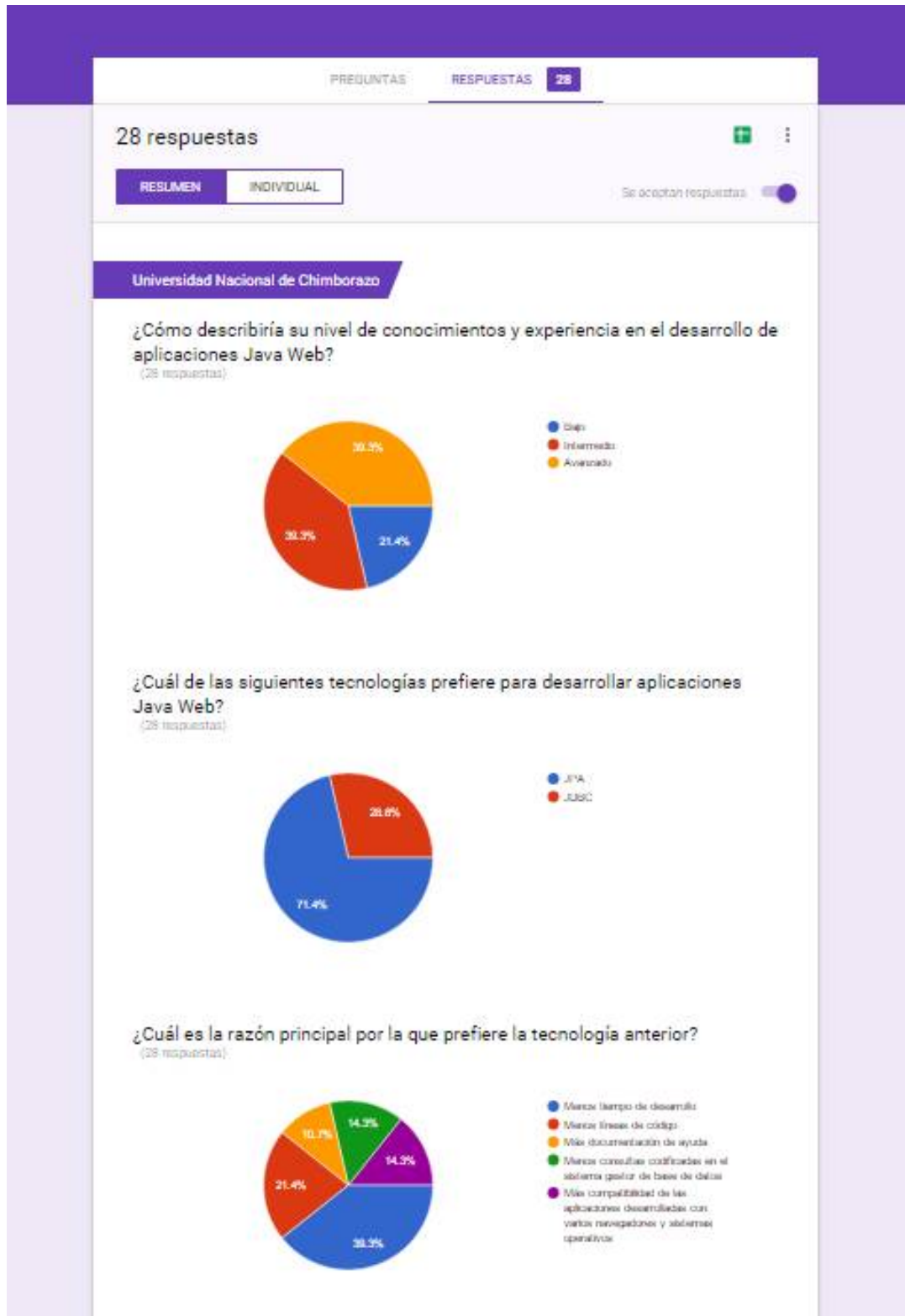
¿Cómo describiría su nivel de conocimientos y experiencia en el desarrollo de aplicaciones Java Web? \*

- Bajo
- Intermedio
- Avanzado

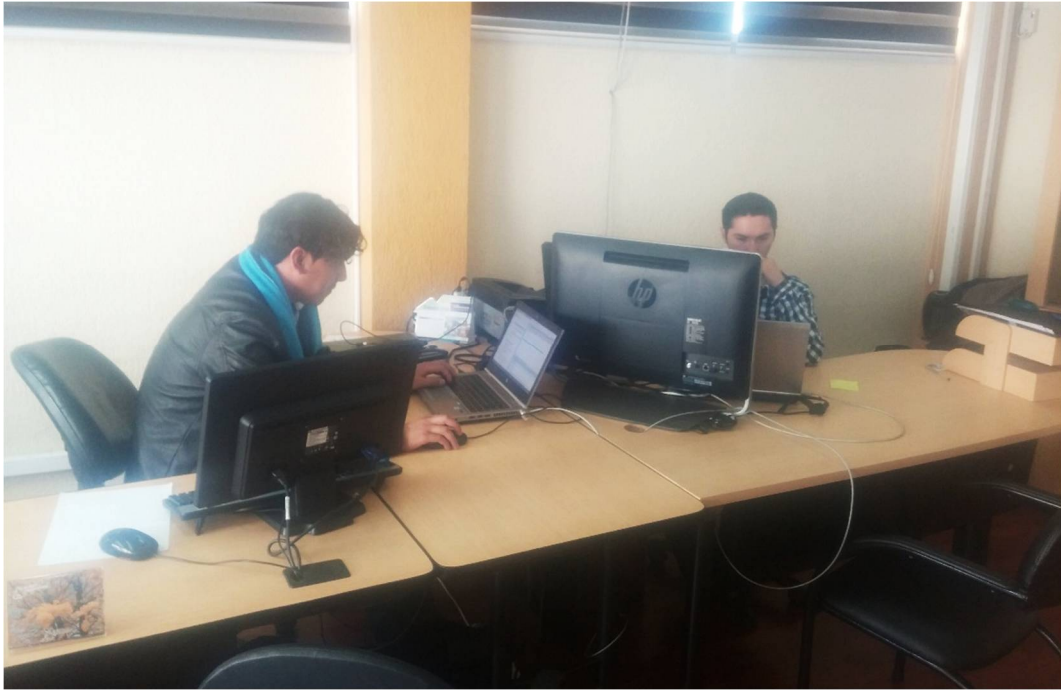
¿Cuál de las siguientes tecnologías prefiere para desarrollar aplicaciones Java Web? \*

- JPA
- JDBC

## ANEXO 4: RESULTADOS DE ENCUESTAS REALIZADAS A 28 PROFESIONALES CON EXPERIENCIA EN EL DESARROLLO DE APLICACIONES JAVA.



**ANEXO 5: FOTOGRAFÍAS DE PROFESIONALES APLICANDO LAS ENCUESTAS EN EL ICITS**



## ANEXO 6: CÓDIGO DE LA CLASE EVENTO MAPEADA DESDE LA BASE DE DATOS CON JPA

```
package Modelo;
import java.io.Serializable;
import java.util.Collection;
import java.util.Date;
import javax.persistence.Basic;
import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.OneToMany;
import javax.persistence.Table;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlTransient;
/**
 *
 * @author jack
 */
@Entity
@Table(name = "evento")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "Evento.findAll", query = "SELECT e FROM Evento e"),
    @NamedQuery(name = "Evento.findByEventoid", query = "SELECT e FROM Evento e
WHERE e.eventoid = :eventoid"),
```

```

    @NamedQuery(name = "Evento.findByNombre", query = "SELECT e FROM Evento e
WHERE e.nombre = :nombre"),
    @NamedQuery(name = "Evento.findByDescripcion", query = "SELECT e FROM
Evento e WHERE e.descripcion = :descripcion"),
    @NamedQuery(name = "Evento.findByUbicacion", query = "SELECT e FROM Evento
e WHERE e.ubicacion = :ubicacion"),
    @NamedQuery(name = "Evento.findByFechainicio", query = "SELECT e FROM
Evento e WHERE e.fechainicio = :fechainicio"),
    @NamedQuery(name = "Evento.findByFechafin", query = "SELECT e FROM Evento e
WHERE e.fechafin = :fechafin"),
    @NamedQuery(name = "Evento.findByResponsable", query = "SELECT e FROM
Evento e WHERE e.responsable = :responsable"),
    @NamedQuery(name = "Evento.findByTematica", query = "SELECT e FROM Evento
e WHERE e.tematica = :tematica"),
    @NamedQuery(name = "Evento.findByRequisitos", query = "SELECT e FROM Evento
e WHERE e.requisitos = :requisitos"),
    @NamedQuery(name = "Evento.findByObservaciones", query = "SELECT e FROM
Evento e WHERE e.observaciones = :observaciones"))
public class Evento implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "eventoid")
    private Integer eventoid;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 100)
    @Column(name = "nombre")
    private String nombre;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 250)
    @Column(name = "descripcion")
    private String descripcion;
    @Basic(optional = false)
    @NotNull
    @Size(min = 1, max = 150)

```



```

@Column(name = "ubicacion")
private String ubicacion;
@Basic(optional = false)
@NotNull
@Column(name = "fechainicio")
@Temporal(TemporalType.DATE)
private Date fechainicio;
@Basic(optional = false)
@NotNull
@Column(name = "fechafin")
@Temporal(TemporalType.DATE)
private Date fechafin;
@Basic(optional = false)
@NotNull
@Column(name = "responsable")
private int responsable;
@Basic(optional = false)
@NotNull
@Size(min = 1, max = 250)
@Column(name = "tematica")
private String tematica;
@Size(max = 250)
@Column(name = "requisitos")
private String requisitos;
@Size(max = 250)
@Column(name = "observaciones")
private String observaciones;
@JoinColumn(name = "unidadorganizadora", referencedColumnName = "unidadid")
@ManyToOne(optional = false)
private Unidad unidadorganizadora;
@OneToMany(cascade = CascadeType.ALL, mappedBy = "evento")
private Collection<Noticias> noticiasCollection;
@OneToMany(cascade = CascadeType.ALL, mappedBy = "eventoid")
private Collection<Conferencia> conferenciaCollection;
@OneToMany(cascade = CascadeType.ALL, mappedBy = "evento")
private Collection<Concurso> concursoCollection;

public Evento() {
}

```

```

public Evento(Integer eventoid) {
    this.eventoid = eventoid;
}

public Evento(Integer eventoid, String nombre, String descripcion, String ubicacion,
Date fechainicio, Date fechafin, int responsable, String tematica) {
    this.eventoid = eventoid;
    this.nombre = nombre;
    this.descripcion = descripcion;
    this.ubicacion = ubicacion;
    this.fechainicio = fechainicio;
    this.fechafin = fechafin;
    this.responsable = responsable;
    this.tematica = tematica;
}

public Integer getEventoid() {
    return eventoid;
}

public void setEventoid(Integer eventoid) {
    this.eventoid = eventoid;
}

public String getNombre() {
    return nombre;
}

public void setNombre(String nombre) {
    this.nombre = nombre;
}

public String getDescripcion() {
    return descripcion;
}

public void setDescripcion(String descripcion) {
    this.descripcion = descripcion;
}

```

```
}

public String getUbicacion() {
    return ubicacion;
}

public void setUbicacion(String ubicacion) {
    this.ubicacion = ubicacion;
}

public Date getFechainicio() {
    return fechainicio;
}

public void setFechainicio(Date fechainicio) {
    this.fechainicio = fechainicio;
}

public Date getFechafin() {
    return fechafin;
}

public void setFechafin(Date fechafin) {
    this.fechafin = fechafin;
}

public int getResponsable() {
    return responsable;
}

public void setResponsable(int responsable) {
    this.responsable = responsable;
}

public String getTematica() {
    return tematica;
}

public void setTematica(String tematica) {
```

```

        this.tematica = tematica;
    }

    public String getRequisitos() {
        return requisitos;
    }

    public void setRequisitos(String requisitos) {
        this.requisitos = requisitos;
    }

    public String getObservaciones() {
        return observaciones;
    }

    public void setObservaciones(String observaciones) {
        this.observaciones = observaciones;
    }

    public Unidad getUnidadorganizadora() {
        return unidadorganizadora;
    }

    public void setUnidadorganizadora(Unidad unidadorganizadora) {
        this.unidadorganizadora = unidadorganizadora;
    }

    @XmlTransient
    public Collection<Noticias> getNoticiasCollection() {
        return noticiasCollection;
    }

    public void setNoticiasCollection(Collection<Noticias> noticiasCollection) {
        this.noticiasCollection = noticiasCollection;
    }

    @XmlTransient
    public Collection<Conferencia> getConferenciaCollection() {
        return conferenciaCollection;
    }

```

```

}

public void setConferenciaCollection(Collection<Conferencia> conferenciaCollection) {
    this.conferenciaCollection = conferenciaCollection;
}

@XmlTransient
public Collection<Concurso> getConcursoCollection() {
    return concursoCollection;
}

public void setConcursoCollection(Collection<Concurso> concursoCollection) {
    this.concursoCollection = concursoCollection;
}

@Override
public int hashCode() {
    int hash = 0;
    hash += (eventoid != null ? eventoid.hashCode() : 0);
    return hash;
}

@Override
public boolean equals(Object object) {
    // TODO: Warning - this method won't work in the case the id fields are not set
    if (!(object instanceof Evento)) {
        return false;
    }
    Evento other = (Evento) object;
    if ((this.eventoid == null && other.eventoid != null) || (this.eventoid != null &&
!this.eventoid.equals(other.eventoid))) {
        return false;
    }
    return true;
}

@Override
public String toString() {
    return "Modelo.Evento[ eventoid=" + eventoid + " ]";
}
}

```