ESUELA SUPERIOR POLITECNICA DE CHIMBORAZO

FACULTAD DE INFOMATICA Y ELECTRONICA

ESCUELA DE INGENIERIA EN SISTEMAS

"ANÁLISIS DE LA TECNOLOGÍA TOPLINK, COMO FRAMEWORK DE PERSISTENCIA APLICADO AL SISTEMA DE EVALUACIÓN DE LAS CARRERAS DE LA FACULTAD DE INFORMÁTICA Y ELECTRÓNICA CON FINES DE ACREDITACIÓN"

TESIS DE GRADO

Previa a la obtención del título de

INGENIERO EN SISTEMAS INFORMATICOS

Presentado por:

JENNY PAULINA NAULA LEMA

WALTER IVÁN MAIQUIZA TITUAÑA

RIOBAMBA – ECUADOR

2012

AGRADECIMIENTO

Agradezco a Dios por sucuidado, provisión, sabiduria y amor incondicional, a mis padres por brindarme su apoyo y comprensión durante todo este tiempo, a mis hermanas por compartir momentos alegres y tristes en mi vida, y a todos los que conforman Cruzada Estudiantil y Profesional para Cristo por marcar mi vida y enseñarme a ver las cosas de una manera diferente poniendo en primer lugar a Dios en mi vida.

Jenny Paulina Naula Lema

Agradezco con un profundo sentimiento de amor a toda mi familia en especial amis padres por haber brindado su apoyo y comprensión durante todo este tiempo, sin decaer ante los fracasos y adversidades, a mis hermanos por compartir gratos momentos en mi vida, y a todos aquellos que conforman la cruzada estudiantil.

Walter Iván MaiquizaTituaña

DEDICATORIA

Dedico este trabajo primeramente a Dios por ser mi guía y darme la sabiduría necesaria, a mis padres, hermanas y Cruzada Estudiantil que han sido los pilares fundamentales para terminar este sueño, gracias por otorgarme su apoyo incondicional al estar junto a mí en momentos felices y tristes.

Jenny Paulina Naula Lema

El presente trabajo dedico infinitamente a Dios por ser la luz que guía mi camino, a mis padres que han sido el principal apoyo para cumplir este sueño anhelado. Tambien dedico a todos mis compañeros y profesores que siempre me apoyaron incondicionalmente para seguir adelante, haciendo de mí una persona de bien

Walter Iván MaiquizaTituaña

FIRMAS DE RESPONSABLES Y NOTA

NOMBRES	FIRMAS	FECHA
Ing. Iván Menes **DECANO DE LA FACULTAD** **DE INFORMÁTICA Y ELECTRÓNICA**	----------	----------
Ing. Raúl Rosero **DIRECTOR DE LA ESCUELA** **DE INGENIERÍA EN SISTEMAS**	----------	----------
Ing. Jorge Huilca **DIRECTOR DE TESIS**	----------	----------
Ing. Raúl Rosero **MIEMBRO DE TESIS**	----------	----------
Tlgo. Carlos Rodríguez **Dir. Dpto. CENTRO DOCUMENTACIÓN**	----------	----------

NOTA DE LA TESIS ----------

"Nosotros, Jenny Paulina Naula Lema y Walter Iván Maiquiza Tituaña, somos los responsables del contenido, ideas y resultados planteados en el presente proyecto de tesis, y el patrimonio intelectual del mismo pertenece a la Escuela Superior Politécnica Chimborazo".

Jenny Paulina Naula Lema	**Walter Iván Maiquiza Tituaña**

INDICE GENERAL

INDICE FIGURAS

INDICE TABLAS

INTRODUCCION

La manipulación de datos en Programación Orientada a Objetos (OOP) involucra la manipulación de Objetos con propiedades simples y asociaciones de agregación, composición, herencia, etc. a otros objetos (datos). Pero además los objetos definen comportamiento mediante métodos que manipulan su estado interno además de relaciones de herencia y la propiedad de polimorfismo.

Por su parte los RDBMS manipulan información escalar (int, string, etc.) organizada en tablas y no definen comportamiento ni relaciones de herencia asociado a las entidades de datos (registros en las tablas de la base de datos). Es el programador quien debe convertir, si no cuenta con un framework adecuado, objetos en datos tabulares y viceversa, lo cual puede resultar en una tarea muy compleja.

Esto ha llevado al diseño de los llamados frameworks de persistencia Objeto/Relacional basados en las técnicas conocidas como Mapeo Objeto/Relacional (ORM por la sigla en inglés de Object Relational Mapping).

Las técnicas de ORM permiten automatizar procesos que trasladan objetos a formas almacenables en tablas y viceversa, preservando los atributos de los objetos. Para esto se basan en la utilización de metadatos de "mapping" que especifican la información necesaria para que un framework de persistencia ORM pueda efectuar de forma automática la conversión de datos entre el sistema relacional y el sistema orientado a objetos.

CAPITULO I

MARCO REFERENCIAL

1.1. Antecedentes

La tendencia actual es trabajar con programación orientada a objetos y bases de datos relacionales probablemente estas dos formas de paradigmas son presentadas de formas diferentes.

El modelo relacional en los gestores de base de datos trata mediante relaciones, así como el manejo de tuplas o registros son puramente matemáticos. El nombre de modelo relacional viene de la estrecha relación que existe entre el elemento básico de este modelo, y el concepto matemático de relación.

Por otro lado la programación orientada a objetos es un paradigma que utiliza objetos como elementos fundamentales en la construcción de la solución.

TopLink proporciona un mecanismo altamente flexible y productivo para almacenar los objetos. TopLink ofrece a desarrolladores funcionamiento excelente y opción de reutilización de código, trabajando con cualquier base de datos relacional, cualquier

Servidor del uso, cualquier conjunto de herramientas y proceso del desarrollo, y cualquier arquitectura de J2EE.

TopLink es un miembro de la familia del Middleware de la fusión de Oracle de los productos, que traen mayor agilidad, una toma de decisión mejor, y coste y riesgo reducidos.

OracleTopLinkconstruyeaplicacionesde alto rendimiento que almacena persistencia de datos orientadosa objetosenunabase de datosrelacional. Con éxitotransformalos datosorientadosa objetosencualquiera delos datosrelacionalesoelementosde marcado extensible (XML).

Es así que se presenta TopLink, como un framework orientado a la persistencia solucionando sistemas web mediante su mapeo a las bases de datos relacionales.

¿Qué es la persistencia?

La persistencia proviene del Latín *persistere* que significa Durar por largo tiempo, en el ámbito informático la mayoría de los programas informáticos actuales necesitan preservar los datos para su posterior uso, esto es más frecuente en el uso de base de datos relacionales.

ORM ayuda a reducir la llamada diferencia de impedancia Objeto-Relacional, el abismo entre el modelo orientado a objetos de una aplicación Java bien diseñada (basada en el modelado de cosas y conceptos del mundo real) y el modelo relacional de un esquema de base de datos (basado en aproximaciones matemáticas para almacenar datos).

Este abismo es sorprendentemente ancho, una aproximación es el simple mapeo de una clase a una tabla.

El ORM lo realiza un marco de trabajo de persistencia, que sabe como consultar la base de datos para recuperar objetos Java, y cómo persistir dichos objetos en su representación en las tablas y columnas de la base de datos. Los mapeos se definen en metadatos, típicamente ficheros XML.

Con ORM, la automatización se extiende no sólo al mapeo de objetos a filas y columnas de la base de datos, sino también a la detección de objetos dañados y la escritura en la base de

datos con el menor número de actualizaciones SQL, requerimientos que son apropiados para la mayoría de las aplicaciones y se pueden cumplir más fácilmente utilizando marcos de trabajo.

La Facultad de Informática y Electrónica ha visto como una necesidad de promover la calidad en todas sus funciones, para lo cual es necesario realizar una medición y verificación de los diferentes indicadores que proporcionen los resultados exactos en el momento que las carreras de la facultad son evaluadas.

En vista de estas necesidades surge el planteamiento de un sistema informático de evaluación que se construye en base a indicadores para lo cual se empleará la herramienta seleccionada a fin de dar solución al mismo.

La autoevaluación es el proceso de evaluación orientado a la mejora de la calidad, y llevado a cabo por las propias instituciones o programas educativos con la participación de sus actores sociales, es decir, estudiantes, egresados, docentes, administrativos, autoridades, padres de familia, y grupos de interés.

1.2. Justificación

1.2.1. Justificación Teórica

El modelo relacional trata con relaciones, tuplas y conjuntos y es muy matemático por naturaleza. Sin embargo, el paradigma orientado a objetos trata con objetos, sus atributos y relaciones entre objetos. Cuando se quiere hacer que los objetos sean persistentes utilizando para ello una base de datos relacional, uno se da cuenta de que hay una desavenencia entre estos dos paradigmas, la también llamada diferencia objeto-relacional (object – relational gap"). Un mapeador objeto-relacional (ORM) nos ayudará a evitar esta diferencia.

TopLink es una herramienta para la plataforma Java que facilita el mapeo de atributos entre una base de datos relacional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) que permiten establecer estas relaciones.

TopLink es una herramienta ORM completa que ha conseguido en un tiempo record una excelente reputación en la comunidad de desarrollo posicionándose claramente como el producto OpenSource líder en este campo gracias a sus prestaciones, buena documentación y estabilidad.

De tal manera que TopLink permite al desarrollador ser más productivo, reduciendo costos y aumentando ganancias, proporcionando bajos niveles de vulnerabilidad a los sistemas. Ya que existen estudios que demuestran que el 35% del código de una aplicación se produce como consecuencia del mapeado (correspondencia) entre los datos de la aplicación y el almacén de datos.

1.2.2. Justificación Metodológica

La importancia del desarrollo de una guía metodológica radica en los lineamientos que ésta ofrecerá para la implementación de futuras aplicaciones, lo cual además de reducir el tiempo requerido, establecerá un estándar de desarrollo orientando y fortaleciendo el trabajo y la gestión.

Microsoft Solution Framework (MSF), es un modelo de procesos que combina dos modelos muy comunes en proyectos de desarrollo, el modelo en cascada y el modelo en espiral.

Consta de cinco etapas, en las cuales se generan entregables concretos que ayudan a resolver los requerimientos del cliente de una manera objetiva. Las cinco etapas de MSF son:

El “envisionamiento” del proyecto, en donde todo el equipo va a tener una idea clara, pero general de los objetivos, infraestructura, tecnología, arquitectura, roles y riesgos todos estos necesarios para el buen desenvolvimiento del proyecto.

La planificación del proyecto, es la etapa donde se levantarán los requerimientos específicos del cliente, tomando en cuenta que MSF no es cerrado, es decir que permite cambios dentro del proyecto, incluso en la etapa de desarrollo.

El desarrollo del proyecto, que es la etapa donde propiamente se genera el código necesario para generar un producto funcional para el cliente.

La fase de estabilización, que es donde se prueba al proyecto en ambientes simulados, que deben ser los más parecidos a los ambientes reales, y se corrigen, en caso de haber, los errores del aplicativo.

Y por último la implementación, que es la fase donde el proyecto será puesto en el ambiente de producción, para que cumpla el fin por el cual fue creado.

Para entender el modelo MSF, es importante entender a claridad el modelo en cascada, y el modelo en espiral, así como las ventajas y desventajas de estos, luego se podrá visualizar de qué manera MSF junta las mejores prácticas de ambos para dar como resultado un modelo altamente práctico.

MSF involucra todo un framework guía, que indica paso a paso las mejores prácticas para el desarrollo rápido y fiable de un proyecto de desarrollo de software. Recomendado, para proyectos pequeños y medianos, y que va desde el cómo levantar los requerimientos, hasta la puesta en producción del proyecto.

1.2.3. Justificación Aplicativa

El motivo principal para el desarrollo de este proyecto surge de las necesidades de automatizar el proceso de evaluación de las carreras de la FIE con fines de acreditación, de esta manera es posible dar una solución informática aplicando la tecnología TopLink.

La solución contempla los siguientes Módulos de Gestión:

- Módulos de autenticación de usuarios administradores y clientes definidos por la institución, basado en roles, lo que permitirá desplegar los diferentes procesos descritos en los siguientes puntos.
- Módulo de gestión de Indicadores

  Este módulo permitirá realizar tareas como ingreso, modificaciones, actualizaciones o eliminaciones de cada indicador.
- Módulo de gestión de Usuarios que proporcionan información de Indicadores.

  Este módulo permitirá al Administrador realizar tareas como ingreso, modificación, actualización o eliminación de cada usuario.

- Módulo de reportes con información relevante para la toma de decisiones.
  Este módulo contendrá los resultados de las evaluaciones realizadas a las carreras
- Módulo de presentación de estadísticas con resultados de evaluación de las carreras.

Este módulo contendrá información resumida estadísticamente para la toma de decisiones por parte de las autoridades pertinentes.

1.3. Objetivos

1.3.1. Objetivo General

Analizar la tecnología TopLink, como framework de persistencia, y aplicar en el desarrollo del sistema de Evaluación de las carreras de la Facultad de Informática y Electrónica con fines de acreditación.

1.3.2. Objetivos Específicos

- Analizar la herramienta de persistencia TopLink de Java basabas en ORM bajo un IDE con sus características, ventajas y desventajas.
- Analizar el manejo de objetos, clases, transacciones y seguridades como parámetros principales.
- Implementar el Sistema de Evaluación de carrearas de la FIE con fines de acreditación usando TopLink como Framework de persistencia.
- Evaluar a TopLink como Framework de persistencia.

1.4. Hipótesis

El uso del Framework de persistencia TopLink permitirá construir aplicaciones web con bajos niveles de vulnerabilidad a ataques externos.

CAPITULO II

2. ORM (MAPEO OBJETO RELACIONAL) – PERSISTENCIA DE OBJETOS

2.1. ORM (MAPEO OBJETO RELACIONAL)

2.1.1. Introducción

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).En este capítulo se presenta los fundamentos, problema y la alternativa de solución común para la persistencia de Objetos en base de datos relacionales.

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

2.1.2. ¿Qué es ORM?

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”. [1]

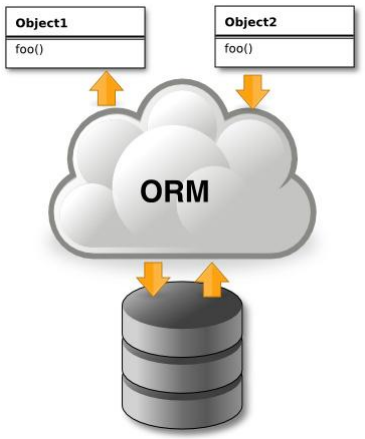


Figura II.1. Mapeo Objeto Relacional

Fuente: http://www.tecnoretales.com/programacion/que-es-doctrine-orm/

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 losconceptos de orientación a objetos creando ”bases de datos virtuales”, es decir laaplicación desde dentro utiliza frameworks5, los mismos que son intermediarios entre labase de datos relacional y la aplicación totalmente orientada a objetos. [11]

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.

Framework es un esquema (un esqueleto, un patrón) para el desarrollo y/o la implementación de una aplicación. Sí, es una definición muy genérica, pero también puede serlo un *framework*: El paradigma MVC (Model-View-Controller) dice poco más que "separa en tu aplicación la gestión de los datos, las operaciones, y la presentación". En el otro extremo, otros *frameworks* pueden llegar al detalle de definir los nombres de ficheros, su estructura, las convenciones de programación, etc.

Los frameworks se están utilizando frecuentemente para el desarrollo ágil de aplicaciones informáticas y sufren cambios constantes a fin de proporcionar mayor ayuda al programador y sobre todo brindar una presentación más amigable de la información.

En periodos cortos de tiempo se crea una versión mejorada de frameworks, es así que dotNET framework, está sufriendo constantes cambios, comenzó con el framework 1.0, continuó con el framework 2.0, y actualmente posee el ORCASframework. En el mismo ámbito aunque no tan desarrollado, publicitado y conocido sehallan, el Cakephp, seagull, zend, kumbia, entre otros que agilitan la programación en PHP, que buscan ganar un sitio entre las aplicaciones de nivel inferior "requerimientos de aplicación o programa base".

Definitivamente, el desarrollo continuo de frameworks es una carrera interminable contra la tenaz velocidad de los requerimientos software de seguridad, presentación, productividad, costo, entre otros aspectos; de tal manera que, mancomunadamente, debe existir interacción entre usuario – gestor – programador, para lograr el éxito de la tecnología.

2.1.3. Importancia de un Framework

Se ha definido la importancia del framework según algunos autores, que indican lo siguiente:

- **Modularidad y reducción de la complejidad**. La aplicación esta formada por subsistemas **especializados** en distintos aspectos fundamentales de toda aplicación (persistencia, presentación, manejo de log, etc.)
- **Fortaleza al cambio**. Los módulos pueden ser evolucionados o cambiados conservando la arquitectura global de la aplicación.
- **Documentación**. La documentación del framework promueve el uso correcto del mismo y disminuye el esfuerzo necesario para el mantenimiento.
- **Estructura.** El desarrollo basado en frameworks establece una estructura sobre la cual las aplicaciones pueden ser construidas, liberando al desarrollador de tomar el 100% de las decisiones de diseño.
- **Distribución de funciones.** Permite paralelizar el trabajo de desarrollo, ya que la solución puede desarrollarse como un conjunto de piezas independientes que encajarán en el framework usado.

- **Eficiencia.** El desarrollador puede concentrarse en los requerimientos funciónales de la aplicación.
- **Aplicaciones ricas.** Posibilidad de dar más funcionalidad a los usuarios de la aplicación.
- **Buen manejo de base de datos del que puede crear cada desarrollador:** Ya es un hecho que se tiene poco tiempo para desarrollar toda una aplicación, así que pensar en crear un buen manejador para las bases de datos propias resulta inconveniente. Pero los creadores de Web frameworks ya se encargaron de esto por ejemplo los modernos frameworks MVC. Hoy en día los frameworks hacen muy bien este trabajo, y cada día mejoran aún más.
- **El código se ejecuta en varios tipos de plataforma:** Los desarrolladores de frameworks son unos verdaderos genios, y además de algún modo cuentan con la capacidad de probar sus frameworks en varios tipos de plataforma y asegurarse de que se ejecuten correctamente en cada una de ellas.
- **Seguridad de la aplicación:** esta es la razón más importante para destacarla, desafortunadamente, la seguridad en el proceso de desarrollo de una aplicación aún no es algo que se considere en toda la importancia que tiene. En las universidades aún no es común que enseñen a desarrollar software seguro, y el nivel de seguridad con el que se maneja la información en muchas organizaciones es francamente paupérrimo. Es por esto que la gran mayoría de los Web frameworks traen incorporados mecanismos de seguridad, en la forma de procedimiento de sanitización y validación de datos que permiten filtrar todo lo que es enviado dentro de la aplicación y que generalmente sirve como vector de ataque.

2.1.4. Mapeando objetos RDBMS

Es muy común encontrar aplicaciones orientadas a objetos manipulando datos en bases de datos relacionales, mediante el uso de técnicas de mapeo por metadatos. [9].

Dentro de las consideraciones a tener en cuenta encontramos:

- Un atributo o propiedad podría mapearse a cero o más columnas en una tabla de unRDBMS.
- 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 yrelationship 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 desu mismo tipo.

Pero, excepto para casos triviales, no siempre es tan sencillo como mapear una clase a una tabla en una relación uno a uno donde cada propiedad de la clase corresponde a un campo de una tabla en la base de datos.

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 extra (agregado al modelo OO original y sin significado para el negocio) 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), etc.

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.

2.1.4.1. Asociaciones

Cuando queremos mapear relaciones o colecciones entre entidades tenemos que utilizar asociaciones. Estas asociaciones pueden ser unidireccionales y bidireccionales. Asociaciones unidireccionales pueden entenderse cuando un objeto tiene una referencia a otro objeto. Asociaciones bidireccionales pueden entenderse cuando ambos objetos mantienen referencias al objeto contrario. Además del concepto de dirección, existe otro llamado cardinalidad, que determina cuantos objetos puede haber en cada extremo de la asociación.

Existen tres tipos de asociaciones entre objetos:

- Asociación
- Agregación
- Composición

Por ahora las trataremos igual, aunque existen diferencias en el manejo de las restricciones. Las cuales se pueden clasificar de acuerdo a dos categorías ortogonales:

En base a la multiplicidad:

- One-to-one
- One-to-many (many-to-one según desde donde se lea).
- Many-to-many

En base a la dirección:

- Unidireccionales
- Bidireccional

En la base de datos las relaciones se mantienen mediante el uso de Foreign Keys:

- One-to-one: FK implementada en una de las tablas.
- One-to-many: FK desde la “one table” a la “many table”
- Many-to-many: Es necesario incorporar una tabla asociativa (o de relación)

En cuanto a la direccionalidad, todas las relaciones en la base de datos relacional son efectivamente bidireccionales.

2.1.4.1.1. Asociaciones Unidireccionales

Para entender bien el concepto de unidireccionalidad veamos un ejemplo de una asociación unidireccional:

```
@Entity
public class Cliente {
@Id
@GeneratedValue
private Long id;
@OneToOne
private Direccion direccion;
// Getters y setters
}
```

```
@Entity
public class Direccion {
@Id
GeneratedValue
private Long id;
private String calle;
private String ciudad;
private String pais;
private Integer codigoPostal;
// Getters y setters
}
```

Como puedes ver en el ejemplo de arriba, cada entidad Cliente mantiene una referencia a una entidad Dirección. Esta relación es de tipo uno-a-uno (one-to-one) unidireccional (puesto que UNA entidad Cliente tiene UNA referencia a Dirección y por tanto es anotada con @OneToOne. Cliente es el dueño de la relación de manera implícita, y por tanto cada entidad de este tipo contendrá por defecto una columna adicional en la base de datos que será utilizada para acceder al objeto Dirección. Esta columna es una clave foránea (foreign key), un concepto muy importante en bases de datos relaciones. Cuando JPA realice el mapeo de esta relación, cada entidad será almacenada en su propia tabla, añadiendo ala tabla que contiene la entidad que es dueña de la relación la clave foránea para acceder a la dirección del cliente. Recuerda que JPA utiliza configuración por defecto para realizar el mapeo, pero podemos personalizar como se realizará dependiendo de nuestras necesidades. Para configurar la columna que contendrá la clave foránea podemos usar la anotación @JoinColumn:

```
@OneToOne
@JoinColumn(name = "direccion_fk")
private Direccion direccion;
```

Otro tipo de asociación muy típica es uno-a-muchos (one-to-many) unidireccional.
Veamos un ejemplo:

```
@Entity
public class Cliente {
@Id
@GeneratedValue
private Long id;
@OneToMany
private List direcciones;
// Getters y setters}
```

En este caso, cada entidad de tipo Cliente mantiene una lista de direcciones, medianteuna propiedad List anotada con @OneToMany. En este caso, en vez de utilizar una clave foránea en Cliente, JPA utilizara por defecto una tabla de unión (join table). Una tabla de unión permite que ambos lados de la asociación tengan una clave foránea que es almacenada en una tercera tabla con dos columnas. Como siempre, podemos configurar el mapeo a nuestras necesidades mediante metadatos (anotaciones o xml). Veamoscomo:

```
@OneToMany
@JoinTable(name = ...,
joinColumn = @JoinColumn(name = ...),
inverseJoinColumn = @JoinColumn(name = ...))
private List direcciones;
```

```
@Entity
@Table(name = "ROLES")
@NamedQueries({
    @NamedQuery(name = "Roles.findAll", query = "SELECT r FROM Roles r"),
    @NamedQuery(name = "Roles.findByRolcodigo", query = "SELECT r FROM Roles r WHERE r.rolcodigo = :rolcodi
    @NamedQuery(name = "Roles.findByRoldescripcion", query = "SELECT r FROM Roles r WHERE r.roldescripcion
    @NamedQuery(name = "Roles.findByRoltipo", query = "SELECT r FROM Roles r WHERE r.roltipo = :roltipo")})
public class Roles implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @Column(name = "ROLCODIGO")
    private Short rolcodigo;
    @Column(name = "ROLDESCRIPCION")
    private String roldescripcion;
    @Column(name = "ROLTIPO")
    private Short roltipo;
    @OneToMany(cascade = CascadeType.ALL, mappedBy = "roles", fetch = FetchType.LAZY)
    private List<Usuario> usuarioList;
```

Figura II.2. Ejemplo práctico de Asociacion Unidireccional en SEACE

2.1.4.1.2. Asociaciones Bidireccionales

En las asociaciones bidireccionales, ambos extremos de la relación mantienen una referencia al otro. En este caso, el dueño de la relación debe ser especificado explícitamente

para que JPA pueda realizar el mapeo de manera adecuada. Veamos un ejemplo de bidireccionalidad en una relación uno-a-uno:

```
@Entity
public class Mujer {
@Id
@GeneratedValue
private Long id;
@OneToOne
private Marido marido;
// Getters y setters}}
```

```
@Entity
public class Marido {
@Id
@GeneratedValue
private Long id;
@OneToOne(mappedBy = "marido")
private Mujer mujer;
}
```

En el caso de arriba, Mujer es la dueña de la relación. Puesto que la relación es bidireccional, ambos lados de la relación deben estar anotados con @OneToOne, pero ahora uno de ellos debe indicar que la parte contraria es dueña de la relación añadiendo el atributo mappedBy. El valor de este atributo es el nombre de la propiedad asociada en la entidad que es dueña de la relación. El atributo mappedBy puede ser usado en relaciones de tipo @OneToOne, @OneToMany y @ManyToMany. Solamente el cuarto tipo de relación, @ManyToOne no acepta este atributo.

2.1.4.1.3. Lectura Temprana y Lectura Demorada de Asociaciones

Las asociaciones son parte del mapeo relacional y por tanto también son afectadas por este concepto. Eltipo de lectura por defecto para las relaciones uno-a-uno y muchos-a-uno es temprana (eager). El tipo de lectura para las dos clases de relaciones restantes, uno-a-muchos ymuchos-a-muchos es demorada (lazy). Por supuesto, estos valores pueden sercambiados:

```
@OneToMany(fetch = FetchType.LAZY)
private List pedidos;
```

Se debe ser consciente del impacto en el rendimientode la aplicación que puede tener una configuración errónea en el tipo de lectura, ya queen las asociaciones se pueden ver involucrados muchos objetos, y cargarlos de maneratemprana puede ser además de innecesario, inadecuado.

```
private static final long serialVersionUID = 1L;
@Id
@Basic(optional = false)
@Column(name = "ROLCODIGO")
private Short rolcodigo;
@Column(name = "ROLDESCRIPCION")
private String roldescripcion;
@Column(name = "ROLTIPO")
private Short roltipo;
@OneToMany(cascade = CascadeType.ALL, mappedBy = "roles", fetch = FetchType.LAZY)
private List<Usuario> usuarioList;
```

Figura II.3. Ejemplo práctico de Lectura Temprana y Lectura Demorada de Asociaciones.

Fuente: SEACE

2.1.5. Diferentes técnicas de Mapeo

En el mercado existe una gran variedad de herramientas y productos ORM. Conozcamos algunas de las técnicas más utilizadas por estas herramientas. [8]

Mapeo directo sin identidad: El desarrollador crea clase con atributos de cuyo nombre se derivan las columnas de las tablas en la base de datos (viceversa). Las herramientas examinan estas clases y generan los queries correspondientes. No se mantiene identidad basada en la llave primaria. Esto significa que es posible que en memoria existan dos objetos diferentes que correspondan a la misma hilera de una tabla. Si ambos objetos son actualizados, entonces se perderán los cambios en alguno de ellos. Esto implica que la lógica de la aplicación esté al pendiente de que esto no suceda.

Mapeo directo con identidad: Similar al anterior, pero se mantiene un índice (típicamente un hash table) de todos los registros residentes en memoria. De esta manera se asegura que no puedan existir dos objetos que correspondan a la misma hilera de una tabla. El inconveniente de esta estrategia es que la capa ORM debe responsabilizarse de administrar el ciclo de vida de los objetos, ya que la lógica de la aplicación no tendrá forma de saber cuando se deba eliminar un objeto compartido. Esto puede presentar complicaciones dependiendo del lenguaje de programación utilizado y los mecanismos que provea para administrar el ciclo de vida de los objetos.

Generación a partir de XML: Consiste en generar código a partir de definiciones escritas en XML. Es recomendable que el código generado no sea modificado posteriormente.

2.1.6. Optimizaciones de Rendimiento

A nivel de la base de datos puede ser necesario cambiar el esquema, usualmente desnormalizando porciones del mismo, cambiar los tipos de las columnas clave (numéricos son más efectivos que los strings), reducir la cantidad de columnas que componen una clave, introducir índices o incluso introducir procedimientos almacenados que trasladen a la base ciertos procesos.

También puede ser necesario agregar cachés en memoria que minimicen la cantidad de accesos a la base de datos a los mínimos necesarios.

En cuanto a la optimización de los mapeos siempre hay que tener en cuenta que tengo:

- Maneras de mapear herencia.
- Maneras de mapear relaciones one to one.
- Maneras de mapear atributos de clase.

Además, hay que tener en cuenta que cada vez que se cambia la estrategia de mapping puede ser necesario cambiar el esquema de objetos, el esquema de base de datos o ambos. Por lo que la recomendación es elegir muy bien las estrategias a utilizar.

2.1.7. Ventajas

- **Rapidez en el desarrollo.** La mayoría de las herramientas actuales permiten la creación del modelo por medio del esquema de la base de datos, leyendo el esquema, nos crea el modelo adecuado.
- **Abstracción de la base de datos.** Al utilizar un sistema ORM, lo que conseguidos es separarnos totalmente del sistema de Base de datos que utilicemos, y así si en un futuro debemos de cambiar de motor de bases de datos, tendremos la seguridad de que este cambio no nos afectará a nuestro sistema, siendo el cambio mas sencillo.
- **Reutilización.** Nos permite utilizar los métodos de un objeto de datos desde distintas zonas de la aplicación, incluso desde aplicaciones distintas.
- **Seguridad.** Los ORM suelen implementar sistemas para evitar tipos de ataques como pueden ser los SQL injections.

- **Mantenimiento del código.** Nos facilita el mantenimiento del código debido a la correcta ordenación de la capa de datos, haciendo que el mantenimiento del código sea mucho más sencillo.
- **Lenguaje propio para realizar las consultas.** Estos sistemas de mapeo traen su propio lenguaje para hacer las consultas, lo que hace que los usuarios dejen de utilizar las sentencias SQL para que pasen a utilizar el lenguaje propio de cada herramienta.

2.1.8. Desventajas

Tiempo utilizado en el aprendizaje. Este tipo de herramientas suelen ser complejas por lo que su correcta utilización lleva un tiempo que hay que emplear en ver el funcionamiento correcto y ver todo el partido que se le puede sacar.

Aplicaciones algo más lentas. Esto es debido a que todas las consultas que se hagan sobre la base de datos, el sistema primero deberá de transformarlas al lenguaje propio de la herramienta, luego leer los registros y por último crear los objetos.

2.2. PERSISTENCIA DE OBJETOS

2.2.1. Introducción

Cuando abordamos el desarrollo de una aplicación específicamente una orientada a objetos, uno de los primeros requerimientos que debemos resolver es la integración con una base de datos relacional para guardar, actualizar y recuperar la información que utiliza nuestra aplicación.

2.2.2. Persistencia de Objetos

Podemos encontrar diferentes definiciones del término persistencia, según distintos puntos de vista y autores. Veamos dos que con más claridad y sencillez, concretan el concepto de persistencia de objetos.

La primera definición dice así: “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.” [1]

La otra definición dice: Se llama persistencia de objetos a su capacidad para guardarse y recuperarse desde un medio de almacenamiento. [12]

En definitiva podemos decir que la persistencia de objetos es la capacidad que tienen los objetos de sobrevivir al proceso que los creó; permitiendo al programador almacenar, transferir, y recuperar el estado de los objetos.

2.2.3. Requerimientos de la Capa de Persistencia

Una capa de persistencia encapsula el comportamiento necesario para persistir objetos.

Es decir: leer, escribir y borrar objetos en el almacenamiento persistente (base de datos).

Un buen framework de persistencia debería proveer de forma relativamente sencilla:

Soporte para diversos tipos de mecanismos de persistencia (archivos planos, RDBMS, OODBMS, etc.)

- Encapsulamiento total del mecanismo de persistencia.
- Acciones sobre múltiples objetos (asociaciones, búsquedas, etc.).
- Transacciones: planas o anidadas, locales o distribuidas.
- Extensibilidad: permitir agregar nuevas clases (entidades) de forma sencilla.
- Soportar la manipulación automática de identificadores de objetos (OIDs).
- Cursores que permitan la lectura incremental de objetos desde la base de datos.
- Soporte para proxies que habiliten las técnicas de "lazy loading".
- Registros (Record Sets) para operaciones de bajo nivel.
- Soporte para múltiples arquitecturas (Desktop, Enterprise).
- Soporte para diferentes versiones de una base de datos o diferentes proveedores en forma transparente a la aplicación.
- Soporte para el manejo eficiente de múltiples conexiones a diferentes BD.
- Soporte para drivers de base de datos nativos y no nativos.
- Permitir ejecutar consultas SQL si es necesario
- Proveer un lenguaje para consultas Orientadas a Objetos, más naturales al modelo de información de la aplicación, y transformación automática de éstas a SQL al momento de ejecutar la consulta.

2.2.4. Métodos de Persistencia de Objetos

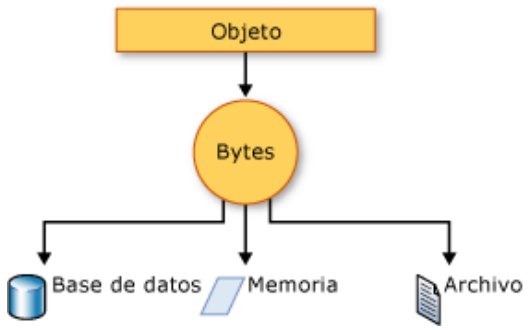
En la actualidad podemos identificar tres formas usuales de persistir objetos: [2]

- Serialización.
- Bases de Datos Orientadas a Objetos (ODBMS).
- Bases de Datos Relacionales.

2.2.4.1. Serialización

La serialización es el proceso de convertir un objeto en una secuencia de bytes para conservarlo en memoria, una base de datos o un archivo. Su propósito principal es guardar el estado de un objeto para poder crearlo de nuevo cuando se necesita. El proceso inverso se denomina deserialización. [15]

El siguiente ejemplo muestra el proceso total de serialización.



FiguraII .4. Funcionamiento de Serialización.
Fuente:http://msdn2.microsoft.com/es-es/library/ms233836(VS.80).aspx

El objeto se serializa en una secuencia que, además de los datos, contiene información sobre el tipo de objeto, como la versión, referencia cultural y nombre de ensamblado. Esa secuencia se puede almacenar en una base de datos, un archivo o en memoria.

La serialización permite al desarrollador guardar el estado de un objeto y volver a crearlo cuando es necesario, y proporcionar almacenamiento de objetos e intercambio de datos.

A través de la serialización, un desarrollador puede realizar acciones como enviar un objeto a una aplicación remota por medio de un servicio Web, pasar un objeto de un dominio a

otro, pasar un objeto a través de un firewall como una cadena XML1 o mantener la seguridad o información específica del usuario entre aplicaciones.

Sin embargo no soporta transacciones, consultas o acceso compartido a los datos entre usuarios múltiples y se la utiliza sólo para proporcionar persistencia en aplicaciones simples o en entornos empotrados que no pueden gestionar una base de datos de forma eficiente.

Es evidente que, dada la tecnología actual, la serialización como método de persistencia es insuficiente para la alta concurrencia web y aplicaciones empresariales.

2.2.5. Impedancia Objeto - Relacional

"Impedancia Objeto-Relacional", se define como un conjunto de dificultades técnicas que surgen cuando una base de datos relacional se usa en conjunto con un programa escrito bajo el paradigma de la Orientación a Objetos. [LIB03]

La siguiente figura muestra algunas diferencias.

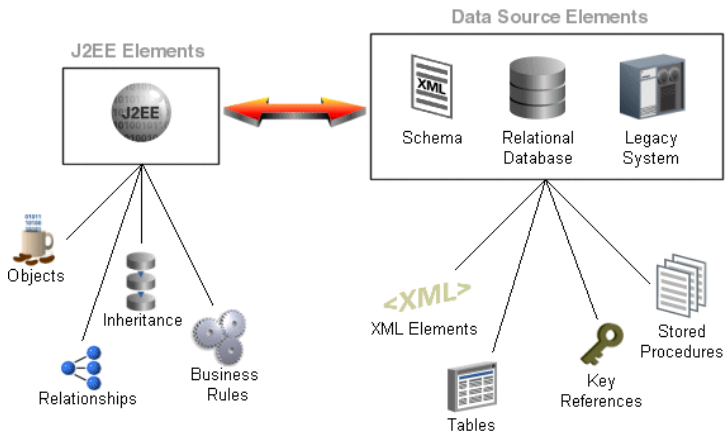


Figura II.5. Impedancia Objeto-Relacional.

Fuente: http://download.oracle.com/docs/cd/B32110_01/web.1013/b28218/undtl.htm

Un ejemplo claro de esta impedancia se observa en el hecho que en el mundo de la programación orientada a objetos, se tiene un claro sentido de la pertenencia, a cada objeto

le pertenecen sus correspondientes atributos; por ejemplo para el objeto Agenda Telefónica podríamos especificar como atributos a una colección de objetos llamados “persona”, en la que a cada persona le corresponde su correspondiente atributo “teléfono”, al transformar esto hacia el mundo relacional se ocuparía más de una tabla para almacenar la información, este simple hecho, hace notar que las tablas del modelo relacional son inconscientes de cómo están relacionadas con otras tablas a un nivel fundamental, puesto que aún cuando posean constraints para definir sus relaciones, para reconstruir el objeto originalmente persistido se debe construir un query, y dicho query debe especificar explícitamente como se relacionan las tablas entre sí, con esto se demuestra además que el lenguaje SQL a pesar de los constraints se mantiene inconsciente de las relaciones que a nivel de objeto poseen las tablas entre ellas.

Así como lo anteriormente expuesto se pueden enumerar distintos problemas que surgen entre los dos modelos:

Reglas de Acceso: En el modelo relacional los atributos pueden ser accesados y/o modificados a través de operadores relacionales predefinidos, mientras que en el modelo orientado a objetos, se permite que cada clase defina la forma en que serán alterados los atributos así como la interface que ocupará para ello.

Ataduras del Esquema: Los objetos del modelo de la POO, no deben seguir ningún esquema en cuanto a que atributos deben o pueden tener, puesto que son definidos por el programador, mientras que las tablas deben seguir el esquema entidad-relación.

Identificador único: Las llaves primarias de una fila tienen generalmente una forma de poder representarse como texto visible, mientras que los objetos no requieren un identificador único externamente visible.

Estructura vs Comportamiento: La orientación a objetos se concentra primordialmente en asegurar que la estructura del programa sea razonable (entendible, extensible, reusable, segura, etc), mientras que los sistemas relacionales ponen el énfasis en tipo de comportamiento que el sistema tendrá una vez en producción (eficiencia, adaptabilidad,

rapidez, etc.). Los métodos de la POO asumen que el principal usuario del código orientado a objetos y sus beneficios es el desarrollador de aplicaciones, mientras que el modelo relacional enfatiza que la forma en que los usuarios finales perciben el comportamiento del sistema es mucho más importante.

De todo esto surge la necesidad de utilizar algún mecanismo para integrar la información contenida en nuestros objetos con los datos almacenados en la base de datos relacional.
Esto típicamente se logra a través de una capa de traducción objeto – relacional.
A continuación se explica la arquitectura de una aplicación y una de las alternativas más comunes para la transformación de objetos a bases de datos relacionales.

2.2.6. Capa de Persistencia

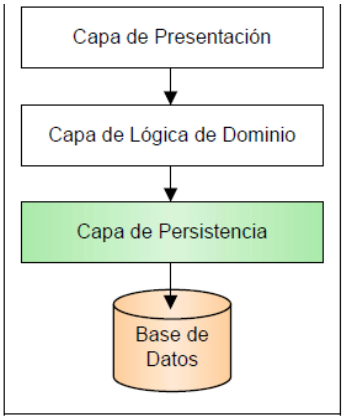


Figura II.6. La Capa de Persistencia.

Esta capa brinda servicios para sincronizar la capa de lógica con un medio de almacenamiento. Para esto se deben identificar los objetos en memoria que deben sobrevivir a la ejecución del programa teniendo así una persistencia de largo plazo.
Dependiendo del tipo de aplicación existen distintos requisitos para esta capa. Por ejemplo en una aplicación de diseño CAD generalmente cuando se edita un documento se trae un gran conjunto de información a memoria desde el medio de almacenamiento ya que traer subconjuntos de información haría que la aplicación tenga un tiempo de respuesta pobre. En cambio en una aplicación empresarial generalmente las operaciones se concentran sólo en un grafo limitado de objetos, por lo que traer toda la información disponible es un gasto

inaceptable. Para esto debe existir un especial interés en la optimización entre la cantidad de información que se trae a memoria y la realmente utilizada.

2.3. JPA /Toplink

2.3.1. Introducción

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.

2.3.2. Java Persistence API (JPA)

2.3.2.1. Definición

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. [LIB05]

Está construida alrededor de tres áreas principales:

- API de persistencia Java.
- Metadatos de transformación objeto/relacional (Anotaciones).
- Lenguaje de consulta.

2.3.2.2. Arquitectura

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. [6]

La arquitectura de JPA, en alto nivel, se muestra en la siguiente figura. (Apache Open JPA/TOPLINK)

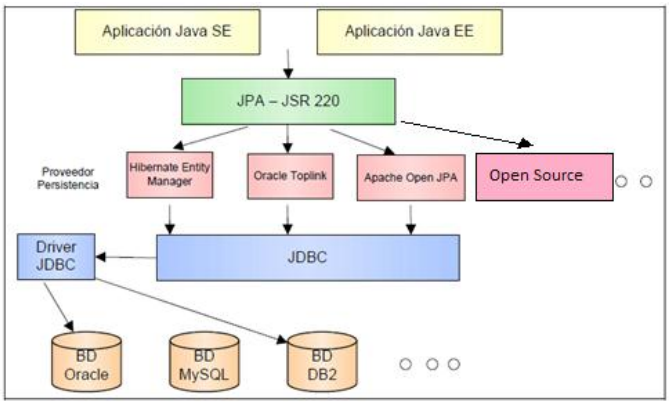


Figura II.7. Arquitectura externa JPA.
Fuente: http://blog.marcomendes.com/category/tecnologias-java/j2ee/

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). [3]

2.3.2.3. Elementos

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

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

La siguiente figura muestra la relación entre estos elementos.

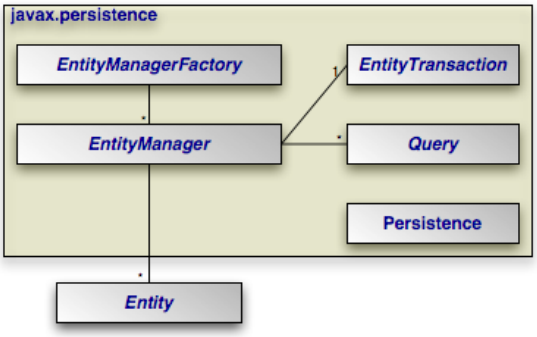


Figura II.8. Relación entre elementos de JPA.

Fuente: Java Persistence for Relational Databases.Richard Sperko Apress © 2003.

La figura muestra que para cada unidad de persistencia hay un EntitityManagerFactory y que muchos Entity Managers pueden ser creados para un solo EntitityManagerFactory. En tanto que muchos Entity Managers pueden apuntar al mismo contexto de persistencia. [LIB05]

2.3.2.3.1. Entidades

Una entidad es un objeto persistente en la aplicación. [3]

- Típicamente representa una tabla de una base de datos relacional.
- Cada instancia de una entidad corresponde normalmente a una fila de la tabla.
- La parte principal de la entidad es la clase Entity, aunque se pueden usar clases auxiliares.

Requerimientos para la clase Entity

- La clase debe llevar la anotación **javax.persistence.Entity**.
- Tener un constructor por omisión público o protegido. Sin argumentos.
- La clase, los métodos o las variables de instancia persistentes no deben ser declarados final.
- Si una instancia de un entity es pasada por valor como un objeto debe implementar la interfaz java.io.Serializable.

- Los entities pueden extender tanto clases entity como cualquier otra, incluso clases no entity pueden extender clases entity.
- Las variables persistentes deben ser declaradas private o protected y solamente pueden ser accedidas a través de los correspondientes métodos de acceso: getters o setters. [DOC01]

Un entity debe declarar una llave primaria. Esto se hace marcando el campo con la anotación @Id. Al igual que en las bases de datos relacionales, la llave primaria hace del entity un objeto único. Cuando se requiere de una llave compuesta se puede usar una clase como llave primaria compuesta. Para este tipo de claves, se utilizan las anotaciones javax.persistence.EmbededId y javax.persistence.IdClass.

Los entities no pueden ser accedidos directamente, deben ser desplegados y usados localmente desde JSE12 o en un contenedor (desde beans de sesión, o de mensajería, o incluso desde cualquier componente web). De cualquier manera, el código cliente debe primero recuperar una instancia particular del entity desde el contexto de persistencia o crear una y adicionarla a dicho contexto. [6]

Para definir los datos de una tabla de base de datos, utilizamos la anotación **@Table** a nivel de la declaración de la clase, esta anotación nos permite definir el nombre de tabla con la que se está mapeando la clase, el esquema, el catálogo, constraints de unicidad. Si no se utiliza esta anotación y se utiliza únicamente la anotación Entity, el nombre de la tabla corresponderá al nombre de la clase.

```
@Entity
@Table(name="CRITERIO")
public class  CRITERIO  implements Serializable {      }
```

La anotación **Table** contiene los atributos de catalog y schema, en caso de que necesitemos definirlos. Usted también puede definir constraints de unicidad utilizando la anotación @UniqueConstraint en conjunción con la anotación @Table

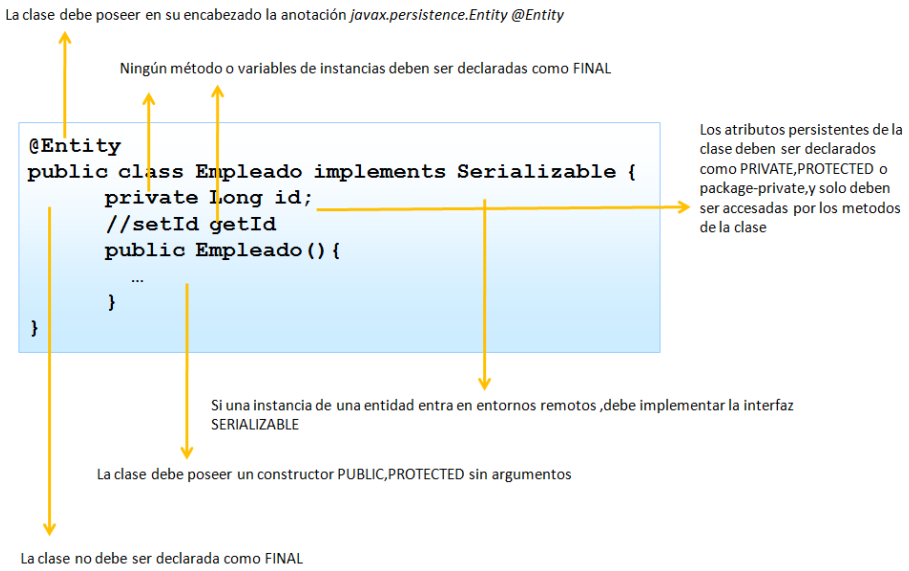


Figura II.9.@Entity

El estado persistente de una entidad puede ser accesible a través de variables de instancia a la entidad o bien a través de las propiedades de estilo de JavaBean (Setters and Getters). Los campos o propiedades pueden tener asociados los siguientes tipos Java:

- Tipos primitivos de Java (int,long,double,etc)
- java.lang.String
- Otro tipo de objeto serializable, incluyendo:
  - Wrappers de tipos primitivos en Java (Integer,Long,Double,etc)
  - java.math.BigInteger
  - java.math.BigDecimal
  - java.util.Date
  - java.util.Calendar
  - java.sql.Date
  - java.sql.Time

- java.sql.TimeStamp
- User-defined serializable types
- byte [ ]
- Byte [ ]
- char [ ]
- Character [ ]
- Tipos enumerados (Enumeration)
- Otras entidades y/o colecciones de entidades

Las entidades podrán utilizar campos persistentes o propiedades persistentes.

Si las anotaciones de mapeo se aplican a las instancias de las entidades, la entidad utiliza campos persistentes, En cambio, si se aplican a los métodos getters de la entidad, se utilizarán propiedades persistentes. Hay que tener en cuenta que no es posible aplicar anotaciones tanto a campos como a propiedades en una misma entidad.

Ciclo de vida de una Entidad

Engloba dos aspectos: la relación entre el objeto entidad y su contexto a persistir y por otro lado la sincronización de su estado con la base de datos. Para realizar estas operaciones la entidad puede encontrarse en cualquiera de estos cuatro estados: [5]

- **New:** Nueva instancia de la Entidad en memoria sin que aún le sea asignado su contexto persistente almacenado en la tabla de la base de datos.
- **Managed:** La Entidad dispone de contenido asociado con el de la tabla de la base de datos debido a que se utilizó el método.
- **Persist ():** Los cambios que se produzcan en la Entidad se podrán sincronizar con los de la base de datos llamando al método **flush ()**.
- **Detached:** La Entidad se ha quedado sin su contenido persistente. Es necesario utilizar el método **merge ()** para actualizarla.

- **Removed:** Estado después de llamarse al método **remove ()** y el contenido de la Entidad será; eliminado de la base de datos.

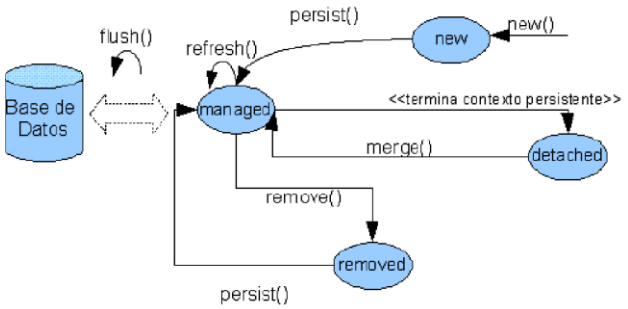


Figura II.10. Ciclo de vida de entidades perisistente.

Fuente: http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=AnotacionesEJB3

2.3.2.4. Identidad

Además, todas las entidades tienen que poseer una identidad que las diferencie del resto, por lo que deben contener una propiedad (preferiblemente que admita un valor null) marcada con la anotación de identidad @Id. Existen formas mas complejas de identidad (@EmbeddedId, @IdClass) que no vamos a explicar para mantener los ejemplos sencillos. Esta identidad va a ser gestionada por el proveedor de persistencia, así que será el quien le asigne un valor la primera vez que almacene la entidad en la base de datos. Para ello, le añadimos la anotacion @GeneratedValue.

2.3.2.5. Lectura temprana y lectura demorada

Una configuración importante es si los objetos asociados a otro se leen automáticamente al leer el mismo. Cargar un objeto y todos sus objetos asociados automáticamente en memoria puede llegar a impactar negativamente en el rendimiento del sistema. Para esto se usan técnicas de **lazy loading** que consisten en cargar los objetos asociados a demanda a medida que son solicitados. Si una asociación nunca es solicitada entonces nunca se lee de la base de datos. [9]

Otra consideración importante tiene que ver con el almacenamiento de colecciones secuenciales de objetos asociados a un objeto padre. En este caso es necesario almacenar en

la base la información (por ejemplo un ordinal numérico) que permita recuperar estos objetos en la secuencia correcta.

Existen dos momentos:

- **LAZY:** se cargan en el momento que se necesiten.
- **EAGER:** se cargan al cargar el objeto que las asocia.

Ya sea Lazy o Eager se carga en el momento que se decide cargarlos como:

- Batch prefetch
- Subselect
- Eager JOIN
- Eager Select

Lazy Load con Proxies:

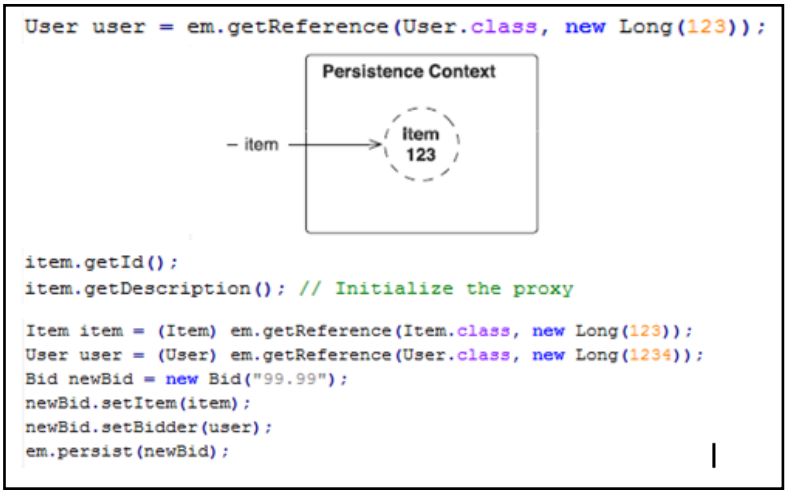


Figura II.11. Lazy Load con Proxies.

Lazy Load con Asociaciones:

Las asociaciones son proxies

-ToOne: Se cargan al referenciarlos

-ToMany: Al acceder a cualquier método de la colección

```
Item item = (Item) em.find(Item.class, new Long(123));
```

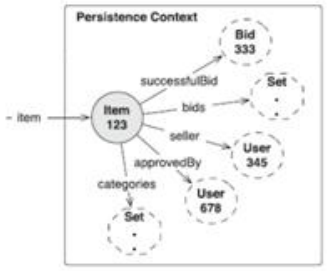


Figura II.12. Lazy Load con Asociaciones.

Posibilidades de configuración Lazy:

- En JPA por defecto:

  LAZY: asociaciones –ToMany

  EAGER: asociaciones –ToOne

- Posibilidades:
  - ✓ Desactivar lazy para la clase
    - Todas las cargas de esa clase serán siempre eager.
    - Ajuste demasiado grueso
  - ✓ ToOne lazy `@ManyToOne(fetch = FetchType.LAZY)`
  - ✓ ToMany eager `@OneToMany(fetch = FetchType.EAGER)`

Desactivar la carga Lazy de una clase

```
@Entity
@Table(name = "USERS")
@org.hibernate.annotations.Proxy(lazy = false)
public class User { ... }
```

```
User user = em.getReference(User.class, new Long(123));
```

Ya no carga proxy

Al cargar el Item se cargan todos los usuarios relacionados por cualquier asociación

```
Item item = em.find(Item.class, new Long(123));
```

```
@Entity
@Table(name = "PERMISO")
@NamedQueries({
    @NamedQuery(name = "Permiso.findAll", query = "SELECT p FROM Permiso p"),
    @NamedQuery(name = "Permiso.findByCodigoper", query = "SELECT p FROM Permiso p WHERE p.codigoper = :codigo
    @NamedQuery(name = "Permiso.findByDescripcionper", query = "SELECT p FROM Permiso p WHERE p.descripcionper
public class Permiso implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @Column(name = "CODIGOPER")
    private Short codigoper;
    @Basic(optional = false)
    @Column(name = "DESCRIPCIONPER")
    private String descripcionper;
    @OneToMany(cascade = CascadeType.ALL, mappedBy = "permiso", fetch = FetchType.LAZY)
    private List<UsuarioPermiso> usuarioPermisoList;
```

Figura II.13. Ejemplo práctico de asociación en SEACE

2.3.2.6. Tipos enumerados

JPA puede manejar todos los tipos básicos de datos de Java, incluidos los tiposenumerados o enum:

```
@Enumerated
private Genero genero;
```

La configuración por defecto de JPA mapeara cada valor ordinal de un tipo enumeradoa una columna de tipo numérico en la base de datos. Por ejemplo, siguiendo el ejemploanterior podemos crear un tipo enum que describa el tipo de género de una película:

```
public enum Genero {
TERROR,
DRAMA,
COMEDIA,
ACCION,}
```

2.3.2.7. Transient

Ciertas propiedades de una entidad pueden no representar su estado. Por ejemplo, imaginemos que tenemos una entidad que representa a una persona:

```
@Entity
public class Persona {
@Id
@GeneratedValue
private Long id;
private String nombre;
private String apellidos
private Date fechaNacimiento;
private int edad;// getters y setters}
```

Podemos considerar que la propiedad edad no representa el estado de Persona, ya que(manteniendo las cosas simples) por si sola no cambiara su valor cada año y por tantopuede contener un valor erróneo. Ya que su valor puede ser calculado gracias a lapropiedad fechaNacimiento, no vamos a almacenar la propiedad edad en la base dedatos, si no a calcularlo cada vez que lo necesitemos. Para indicar al proveedor depersistencia que ignore una propiedad cuando realice el mapeo, usamos la anotación

```
@Transient:
@Transient
private int edad;
```

Ahora, para obtener el valor de edad utilizamos su método getter:

```
public int getEdad() {
// calcular la edad y devolverla
}
```

2.3.2.8. Tipo de Acceso

JPA permite dos tipos de acceso: acceso avariables (FIELD) y acceso a propiedad (PROPERTY). El tipo de acceso estápredeterminado por donde situemos las anotaciones de mapeo. Si las anotaciones están en las variables que conforman la clase,

estaremosindicando a JPA que debe hacer acceso a variables. Si por el contrario definimos lasanotaciones en los métodos getter correspondientes, estaremos indicando un acceso apropiedad. A efectos prácticos, no existe diferencia alguna entre ambas opciones, másallá de gustos personales y de organización de código. Sin embargo, en determinadasocasiones debemos ser consecuentes con el método que elijamos, de manera que nomezclemos tipos de acceso por defecto para evitar que JPA pueda actuar de formaerrónea. Por ejemplo, una clase insertable con acceso por defecto (no definidoexplícitamente) heredara el tipo de acceso de la entidad que la contiene, de manera quesi la clase insertable está utilizando un tipo de acceso diferente debemos indicarloexplícitamente para evitar que JPA utilice el tipo de acceso erróneo. Vamos a ver un ejemplo simple donde se muestra este problema:

```
@Embeddable
public class Insertable {
private int variable;
@Column(name =
"VALOR_DOBLE")
public int getVariable() {
return variable * 2;
}
public void setVariable(int variable) {
this.variable = variable;
}}
```

```
@Entity
public class Entidad
@Id
@GeneratedValue
private Long id;
@Embedded
private Insertable insertable;
}
```

La clase Insertable define un tipo de acceso a propiedad, ya que las anotaciones seencuentran en los métodos getter, y queremos que se acceda a través de estos métodos alvalor de las variables para poder procesar previamente su valor (multiplicar por dos).

2.3.2.9. Contexto de Persistencia

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. [LIB05]

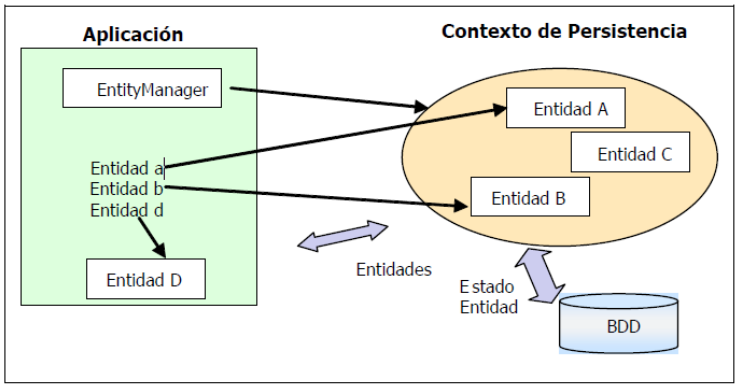


Figura II.14 Contexto de Persistencia

2.3.2.10. Unidad de Persistencia

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.

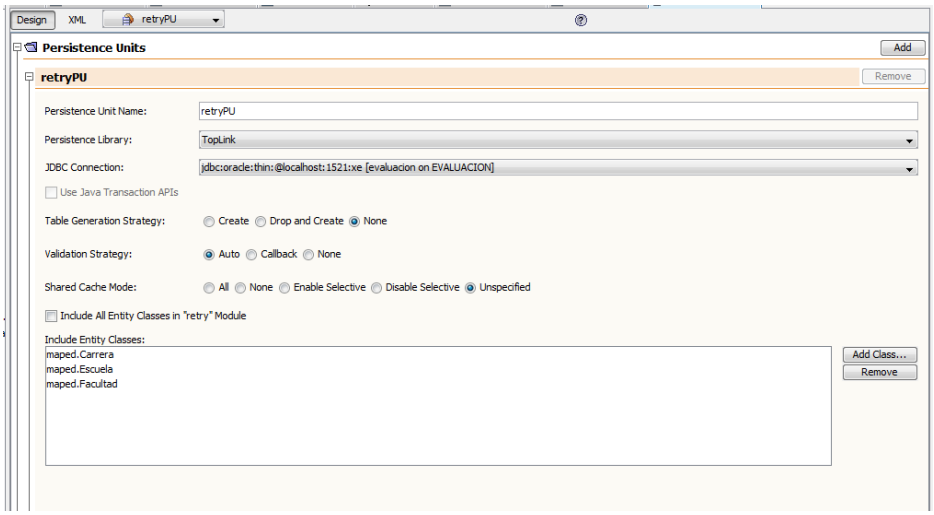


Figura II.15 Unidad de persistencia IDE (Netbeans 6.9.1)

Fuente: SEACE

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.

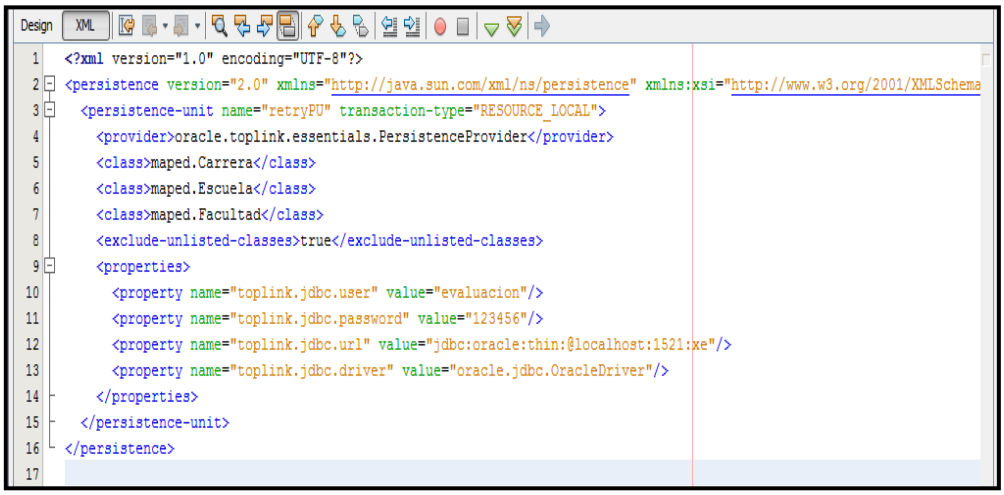


Figura II .16. Unidad de persistencia
Fuente: SEACE

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 transacciónntipo JTA significa que el administrador de entidades participa en la transacción.

provider: Especifica el nombre del proveedor de persistencia. En el ejemplo se muéstrala implementación de GlassFish13, 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 esposible usar JNDI. Las propiedades de la conexión a la base de datos incluyen el

nombrede usuario y contraseña para la conexión, la cadena de la conexión (URL) y el nombre dela clase del driver. [18]

2.3.2.11. Administrador de Entidades (Entity Manager)

Esta interfaz es en la que se apoya la API de persistencia y la que se encarga del mapeo} entre una tabla relacional y su objeto Java. Proporciona métodos para manejar la persistencia de una entidad, permite añadir, eliminar, actualizar y consultar así como manejar su ciclo de vida. Sus métodos más importantes son: [5]

- **Persist (Object entity):** Almacena el objeto entity en la base de datos.
- **Merge (T entity)**: Actualiza las modificaciones en la entidad devolviendo la lista resultante.
- **Remove (Object entity):** Elimina la entidad.
- **Find (Class<T> entity, Object primaryKey):** busca la entidad a través de su clave primaria.
- **Flush ():** Sincroniza las entidades con el contenido de la base de datos.
- **Refresh (Object entity):** Refresca el estado de la entidad con su contenido en la base de datos.
- **createQuery (String query):** Crea una query utilizando el lenguaje JPQL.
- **createNativeQuery ():** Crea una query utilizando el lenguaje SQL.
- **isOpe ():** Comprueba si está; abierto el EntityManager.
- **Close ():** Cierra el EntityManager.

```
public int insertRol(Short cod, String descri, Short tipo)
{
        int flag = -1;
        EntityManagerFactory emf = Persistence.createEntityManagerFactory("corePU");
        EntityManager em = emf.createEntityManager();
        EntityTransaction et = em.getTransaction();
    try
    {
        et.begin();

                    Roles obj = new Roles();
                    obj.setRolcodigo(cod);
                    obj.setRoldescripcion(descri);
                    obj.setRoltipo(tipo);

                      em.persist(obj);

                      flag = 0;
```

Figura II .17. Ejemplo práctico de Persist
Fuente: SEACE

```
public int editRol(Short cod, String descri, Short tipo)
{
    int flag = -1;
        EntityManagerFactory emf = Persistence.createEntityManagerFactory("corePU");
        EntityManager em = emf.createEntityManager();
        EntityTransaction et = em.getTransaction();
    try
    {
        //iniciar transaccio  para la busqueda
        if(!descri.equalsIgnoreCase(null) && !descri.equals(""))
        {
        et.begin();
        //busacr al objeto por la clave primaria
        Roles objrol = em.find(Roles.class, cod);
        et.commit();
        if(objrol != null)
```

Figura II.18. Ejemplo práctico de Find
Fuente: SEACE

```
        //busacr al objeto por la clave primaria
        Roles objrol = em.find(Roles.class, cod);
        et.commit();
        if(objrol != null)
        {
            //iniciar transaccio  para la actualizacion
            et.begin();
            //asignar valores a objeto para modificar
            objrol.setRoldescripcion(descri);
            objrol.setRoltipo(tipo);
            em.merge(objrol);
            //sincronizar al objeto con los valores actuales
            em.flush();
           // em.refresh(objrol);
            flag = 0;
            et.commit();
```

Figura II.19. Ejemplo práctico de Update
Fuente: SEACE

```
public int deleteRol(Short cod)
{
    int flag = -1;
        EntityManagerFactory emf = Persistence.createEntityManagerFactory("corePU");
        EntityManager em = emf.createEntityManager();
        EntityTransaction et = em.getTransaction();
    try
    {
       /* if(cod >= 0)
        {*/
            et.begin();
            Roles obj = em.find(Roles.class,cod);
            et.commit();
            if(obj != null)
            {
                et.begin();
                em.remove(obj);
```

Figura II.20. Ejemplo práctico de Delete
Fuente: SEACE

Podemos obtener una referencia al EntityManager a través de la anotación **@PersistenceContext**. El contenedor nos proporciona el contexto de persistencia mediante inyección por lo que no tendremos que preocuparnos de su creación y destrucción.

```
@PersistenceContext
EntityManager entityManager;
```

Otra forma de obtener un EntityManager es a través de la factoría **EntityManagerFactory** con el nombre del contexto de persistencia configurado en el persistence.xml.

```
public int insertPermiso(Short cod, String desc)
{
    int flag = -1;
    EntityManagerFactory emf = Persistence.createEntityManagerFactory("corePU");
    EntityManager em = emf.createEntityManager();
    EntityTransaction et = em.getTransaction();
    try
```

Figura II .21. Ejemplo práctico EntityManagerFactory
Fuente: SEACE

En el método createEntityManagerFactory de la clase Persistence se debe pasar el nombre del contexto definido en el persistence.xml. [5]

2.3.2.11.1. EntityManager.getReference (class, object)

Este método es similar al método **find()**, toma como argumentos el nombre de la clase y el valor de la llave primaria.

Pero a diferencia del método **find()**, que retorna null sin no encuentra la entidad, este método nos lanzará una EntityNotFoundException.

Otra diferencia es que la entidad extraída por este método, algunos datos de la entidad pueden presentar carga perezosa (lazy loading) al ser accedidos la primera vez.

```
Long pk=1;
Try{
Persona persona=EntityManager.getReference(Persona.class,pk);
}
catch(EntityNotFoundException ex){...}
```

2.3.2.11.2. Persistiendo una entidad

EntityManager.persist (Entidad).

Entidades son simples pojos (Plain old Java objets) hasta que entran en estado managed (manejado) y luego hechos persistentes por un EntityManager.

La siguiente linea de codigo nos muestra una entidad persistente y manejada.

```
Permiso obj = new Permiso();
obj.setCodigoper(cod);
obj.setDescripcionper(desc);
et.commit();
//  if(obj != null)
//  {
      et.begin();
      em.persist(obj);
      em.flush();
      et.commit();
```

Cuando el método **persist ()** es llamado,el motor de persistencia revisará por la existencia del objeto con la ayuda de su identificador único(usualmente representada en la forma de llave primaria).Si un objeto duplicado es encontrado, entonces se lanzará una RunTime exception, **EntityExistsException.**

2.3.2.11.3. Flush y Refresh

EntityManager.flush ()

Sincroniza los datos de una entidad y los hace persistentes en la base de datos.

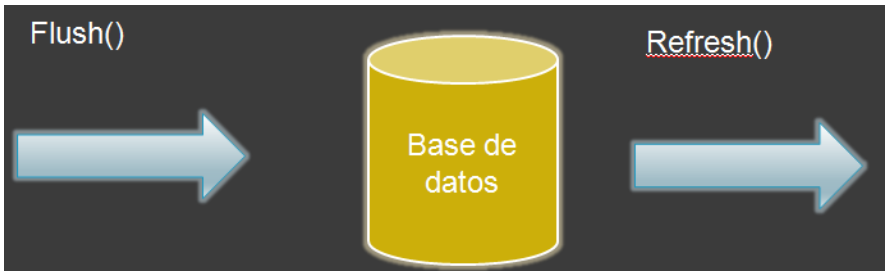


Figura II.22. Flush y Refresh

EntityManager.refresh (entidad)

Contrario al flush (), este método vuelve a cargar los datos originales de la base de datos a la entidad.

2.3.2.12. Anotaciones

Una anotación o metadato proporciona un recurso adicional al elemento de código al que va asociado en el momento de la compilación. Cuando la máquina virtual de Java (JVM) ejecuta la clase busca estos metadatos y determina el comportamiento a seguir con el código al que va unido la anotación. [5]

A continuación se describen las principales anotaciones.

@Entity: Indica que es una Entidad.
@Table: Específica la tabla principal relacionada con la entidad.
@SecondaryTable: Especifica una tabla secundaria relacionada con la entidad si éste englobara a más de una. Tiene los mismos atributos que @Table.

@SecondaryTables: Indica otras tablas asociadas a la entidad.

@UniqueConstraints: Específica que una única restricción se incluya para la tabla principal y la secundaria.

@Column: Especifica una columna de la tabla a mapear con un campo de la entidad.
@JoinColumn: Especifica una campo de la tabla que es foreign key de otra tabla definiendo la relación del lado propietario.
@JoinColumns: Anotación para agrupar varias JoinColumn.

@Id: Indica la clave primaria de la tabla.

@GeneratedValue: Asociado con la clave primaria, indica que ésta se debe generar porejemplo con una secuencia de la base de datos.
@SequenceGenerator: Define un generador de claves primarias utilizado junto con laanotación @GeneratedValue. Se debe especificar la secuencia en la entidad junto a laclave primaria.
@TableGenerator: Define una tabla de claves primarias generadas. Se debe especificaren la anotación @GeneratedValue con strategy = GenerationType.TABLE.
@AttributeOverride: Indica que sobrescriba el campo con el de la base de datos asociado.
@AttributeOverrides: Mapeo de varios campos.

@EmbeddedId: Se utiliza para formar la clave primaria con múltiples campos.

@IdClass: Se aplica en la clase entidad para especificar una composición de la clave primaria mapeada a varios campos o propiedades de a entidad.

@Transient: Indica que el campo no se debe persistir.

@Version: Se utiliza a la hora de persistir la entidad en base de datos para identificar lasentidades según su versión. Se actualiza automáticamente cuando el objeto es mapeadoen la base de datos.
@Basic: Mapeo por defecto para tipos básicos: tipos primitivos, wrappers de los tiposprimitivos, String, BigInteger, BigDecimal, Date, Calendar, Time, Timestamp, byte[],Byte[], char[], Character[], enumerados y cualquier otra clase serializable.

@OneToOne: (1:1) Indica que un campo está; en relación con otro (Ej: dentro de una empresa, un empleado tiene un contrato de trabajo).
@ManyToOne: (N: 1) Indica que un campo está; asociado con varios campos de otra entidad (ej: las cuentas que posee un banco o los empleados que pertenecen a un departamento).
@OneToMany: (1: N) Asocia varios campos con uno (Ej: varios departamentos de un empresa).

@ManyToMany: (N: M) Asociación de varios campos con otros con multiplicidad muchosa-muchos (Ej: relaciones entre empresas).
@Lob: Se utiliza junto con la anotación @Basic para indicar que un campo se debepersistir como un campo de texto largo si la base de datos soporta este tipo.

@Temporal: Se utiliza junto con la anotación @Basic para especificar que un campo fecha debe guardarse con el tipo java.util.Date o java.util.Calendar. Si no se especificara ninguno de estos por defecto se utiliza java.util.Timestamp.

@Enumerated: Se utiliza junto con la anotación @Basic e indica que el campo es un tipo enumerado (STRING), por defecto ORDINAL.

@JoinTable: Se utiliza en el mapeo de una relación ManyToMany o en una relación unidireccional OneToMany.
@MapKey: Específica la clave de una clase de tipo java.util.Map.

@OrderBy: Indica el orden de los elementos de una colección por un ítem específico de forma ascendente o descendente.

@Inheritance: Define la forma de herencia de una jerarquía de clases entidad, es decir la relación entre las tablas relacionales con las entidades.

@NamedQuery: Especifica el nombre del objeto query utilizado junto a EntityManager.
@NamedQueries: Específica varias queries como la anterior.

@NamedNativeQuery: Específica el nombre de una query SQL normal.

@NamedNaviteQueries: Específica varias queries SQL.

2.3.2.13. Relaciones entre entidades

2.3.2.13.1. Relaciones de multiplicidad

Hay cuatro tipos de multiplicidades en las relaciones entre entidades:

- Uno a uno.

- Uno a muchos.
- Muchos a uno.
- Mucho a Muchos.

Uno a uno: en esta relación cada instancia de un entity está relacionada a una únicainstancia de otro entity. Se identifica la relación con la anotación: javax.persistence.OneToOne.

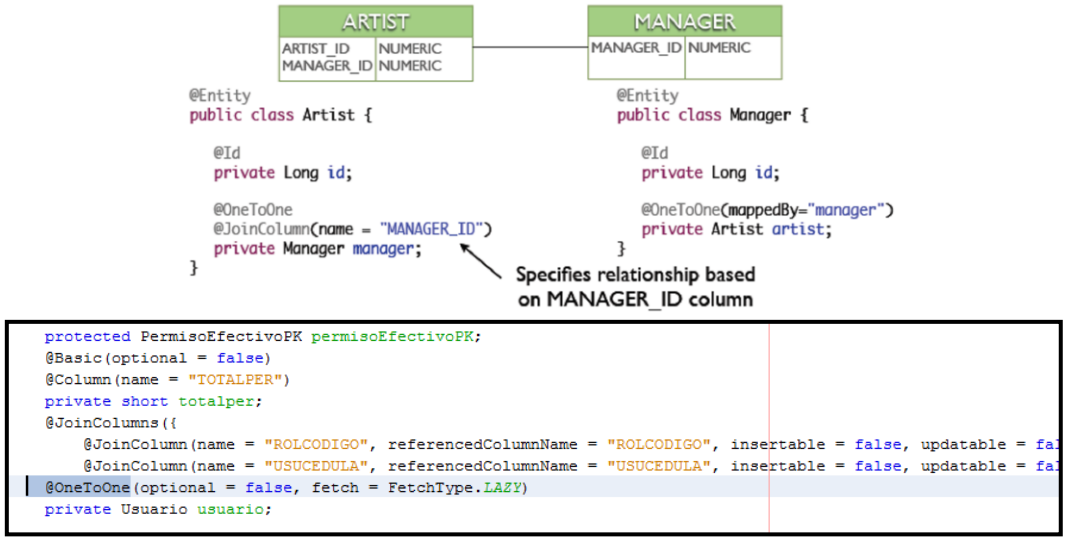


Figura II.23. Ejemplo de @ OneToOne.

Fuente: SEACE

Uno a muchos: una instancia de un entity puede estar relacionada con múltiples instancias de otros entities. La anotación javax.persistence.OneToMany caracteriza esta situación.

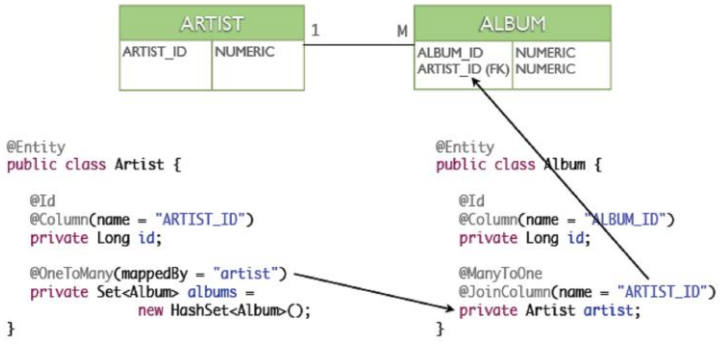
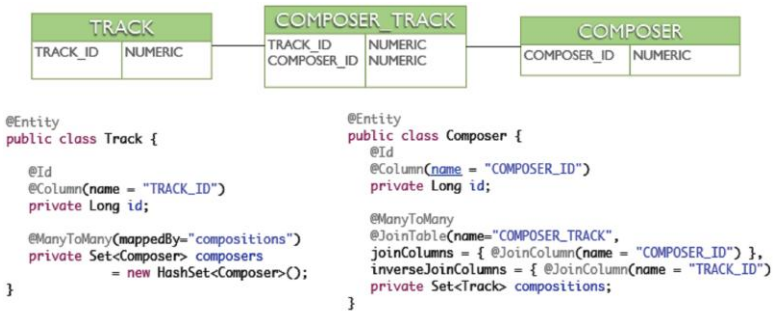


Figura II.24 Ejemplo de @ OneToMany.

Fuente: http://www.intertech-inc.com/resource/usergroup/Exploring%20the%20JPA.pdf

Muchos a uno: múltiples instancias de un entity pueden estar relacionadas con una única instancia de otro entity. Es la opuesta a la relación uno a muchos. Se usa la anotación javax.persistence.ManyToOne.

Muchos a muchos: instancias de un entity pueden estar relacionadas con múltiples instancias de otros entities. Se identifica mediante la anotación javax.persistence.ManyToMany. [14]




```
@Id
@Basic(optional = false)
@Column(name = "ROLCODIGO")
private Short rolcodigo;
@Column(name = "ROLDESCRIPCION")
private String roldescripcion;
@Column(name = "ROLTIPO")
private Short roltipo;
@OneToMany(cascade = CascadeType.ALL, mappedBy = "roles", fetch = FetchType.LAZY)
private List<Usuario> usuarioList;

public Roles() {
}
```

Figura II.25 Ejemplo de @ OneToMany.

Fuente: SEACE

2.3.2.14. Lenguaje de Consulta (JPQL)

JPQL (Java Persistence Query Language)

El Java Persistence Query Language (JPQL) es un lenguaje de consulta independiente de plataforma orientado a objetos definidos como parte de la especificación Java Persistence API.

JPQL se utiliza para hacer consultas en entidades almacenadas en bases de datos

relacionales. Está fuertemente inspirado en SQL, y sus consultas se asemejan a la sintaxis de las consultas SQL, solo que operan contra objetos entidad JPA en lugar de hacerlo directamente con las tablas de base de datos.

Con EntityManager estamos limitados a realizar consultas en la base de datos proporcionando la identidad de la entidad que deseamos obtener, y solo podemos obtener una entidad por cada consulta que realicemos. JPQL nos permite realizar consultas en base a multitud de criterios, como por ejemplo propiedades de la entidad almacenada o condiciones booleanas, y obtener más de un objeto por consulta.

2.3.2.14.1. Ejecución de Sentencias JPQL

JPQL es integrado a través de implementaciones de la interface Query. Dichasimplementaciones se obtienen a través de nuestro querido amigo EntityManagermediante diversos métodos de factoría. Los tres mástípicamente usados (y los queexplicaremos aquí) son:

```
createQuery (String jpql)
createNamedQuery (String name)
createNativeQuery (String sql)
```

Veamos cómo crear un objeto Query y realizar una consulta a la base de datos:

```
import java.util.LinkedList;
import javax.persistence.Query;

@Entity
@Table(name = "USUARIO")
@NamedQueries({
    @NamedQuery(name = "Usuario.findAll", query = "SELECT u FROM Usuario u"),
    @NamedQuery(name = "Usuario.findByRolcodigo", query = "SELECT u FROM Usuario u WHERE u.usuarioPK.rolcodigo = :
    @NamedQuery(name = "Usuario.findByUsucedula", query = "SELECT u FROM Usuario u WHERE u.usuarioPK.usucedula = :
    @NamedQuery(name = "Usuario.findByUsunombre", query = "SELECT u FROM Usuario u WHERE u.usunombre = :usunombre"
    @NamedQuery(name = "Usuario.findByUsuusuario", query = "SELECT u FROM Usuario u WHERE u.usuusuario = :usuusua
    @NamedQuery(name = "Usuario.findByUsuclave", query = "SELECT u FROM Usuario u WHERE u.usuclave = :usuclave"),
    @NamedQuery(name = "Usuario.findByUsuestado", query = "SELECT u FROM Usuario u WHERE u.usuestado = :usuestado"
    @NamedQuery(name = "Usuario.findByCedrol", query = "SELECT u FROM Usuario u WHERE u.usuarioPK.rolcodigo =:rol
public class Usuario implements Serializable {
    private static final long serialVersionUID = 1L;
```

Figura II.26. Ejemplo práctico de sentencias JPQL.

Fuente: SEACE

En el ejemplo anterior obtenemos una implementación de Query mediante el métodocreateQuery (String) de EntityManager, al cual le pasamos una sentencia JPQL

enforma de cadena de texto. Con el objeto Query ya inicializado, podemos realizar laconsulta a la base de datos llamando a su métodogetResultList() el cual devuelve unobjeto List con todas las entidades alcanzadas por la sentencia JPQL. Esta sentencia esuna sentencia dinámica, ya que es generada cada vez que se ejecuta. Utilizando unacolección parametrizada (List<Pelicula>) evitamos tener que hacer ningún tipo decasting dentro del bloque for-each.

2.3.2.14.2. Consulta con Nombre (Estáticas)

Las consultas con nombre son diferentes de las sentencias dinámicas que hemos vistohasta ahora en el sentido en que no pueden cambiar: son leídas y transformadas ensentencias SQL cuando el programa arranca en lugar de cada vez que son ejecutadas.

Este comportamiento estático las hace mas eficientes y por tanto ofrecen un mejorrendimiento. Las consultas con nombre son definidas mediante metadatos (recuerda quelos metadatos se definen mediante anotaciones o configuración XML) definidos en laspropias entidades. Veamos un ejemplo:

```
@Entity
@NamedQuery(name="buscarTodos", query="SELECT p FROM Pelicula p")
public class Pelicula { ... }
```

El código anterior define una consulta con nombre a través de la anotación@NamedQuery. Esta anotación necesita dos atributos: name que define el nombre de laconsulta, y query que define la sentencia JPQL a ejecutar. El nombre de la consultadebe ser único dentro de su unidad de persistencia, de manera que no puede existir otraentidad definiendo una consulta con nombre que se llame buscarTodos. Para evitar quepodamos modificar por error la sentencia, es una buena idea utilizar una constantedefinida dentro de la propia entidad como nombre de la consulta:

```
@Entity
@NamedQuery(name=Pelicula.BUSCAR_TODOS, query="SELECT p FROM
Pelicula
p")
public class Pelicula {
public static final String BUSCAR_TODOS = "Pelicula.buscarTodos";....}
```

Tanto de la primera manera, como de esta ultima, una vez definida una consulta connombre podemos ejecutarla desde nuestra aplicación mediante el segundo metodo de la lista del punto 4.7: createNamedQuery():

```
Query query = em.createNamedQuery(Pelicula.BUSCAR_TODOS);
List<Pelicula> resultados = query.getResultList();
```

createNamedQuery() requiere un parámetro de tipo String que contenga el nombre dela consulta (el cual hemos definido en @NamedQuery(name=...). Una vez obtenida lacolección de resultados, podemos trabajar con ellos de la manera habitual.

2.3.2.14.3. Consultas Nativas SQL

El tercer y último tipo de consultas que vamos a ver requiere una sentencia SQL enlugar de una JPQL:

```
String sql = "SELECT * FROM PELICULA";
Query query = em.createNativeQuery(sql);
```

Las consultas nativas SQL pueden ser definidas de manera estática como las consultascon nombre, obteniendo los mismos beneficios de eficiencia y rendimiento. Para ello, necesitamos utilizar de nuevo metadatos:

```
@Entity
@NamedNativeQuery(name=Pelicula.BUSCAR_TODOS, query="SELECT
* FROM
PELICULA")
public class Pelicula {
public static final String BUSCAR_TODOS = "Pelicula.buscarTodos";
...}
```

Tanto las consultas con nombre como las consultas nativas SQL estáticas son muy útilespara definir consultas inmutables (por ejemplo buscar todas las instancias de unaentidad en la base de datos, como hemos hecho en los ejemplos anteriores), aquellas

quese mantienen entre distintas ejecuciones del programa y que son usadas frecuentemente,etc.

Ejemplos JPQL

• Queries que navegan a entidades relacionadas. [7]

```
SELECT DISTINCT p FROM Player p, IN(p.teams) t
SELECT DISTINCT p FROM Player p JOIN p.teams t
SELECT DISTINCT p FROM Player p WHERE p.team IS NOT EMPTY
```

• Queries que navegan usando relaciones con valores sencillos.

```
SELECT t FROM Team t JOIN t.league l WHERE l.sport = 'soccer' OR l.sport ='football'
```

• Filtrado en los campos relacionados.

```
SELECT DISTINCT p FROM Player p, IN (p.teams) t WHERE t.league.sport = :sport
```

• LIKE

```
SELECT p FROM Player p WHERE p.name LIKE 'Mich%'
```

• IS NULL

```
SELECT t FROM Team t WHERE t.league IS NULL
```

• IS EMPTY

```
SELECT p FROM Player p WHERE p.teams IS EMPTY
```

• BETWEEN

```
SELECT DISTINCT p FROM Player p WHERE p.salary BETWEEN :lowerSalary AND :higherSalary
```

• Operadores de comparación.

```
SELECT DISTINCT p1 FROM Player p1, Player p2 WHERE p1.salary > p2.salary AND p2.name =  : name
```

CREANDO CONSULTAS

- **Consulta Dinámica**

```
public Album findById(Long id) {
String jpql = "select distinct a from Album a left join fetch a.artist art
"
+ "left join fetch art.genre left join fetch a.tracks where a.id = :id"
Query query = getEntityManager().createQuery(jpql);
query.setParameter("id", id);
return (Album) query.getSingleResult();
}
```

- **Consulta Estática**

```
@NamedQuery(name="artist.all",
query="select distinct a from Artist a left join fetch a.albums")
public List<Artist> findAll() {
Query query = getEntityManager().createNamedQuery("artist.all");
return query.getResultList()
}
```

2.3.2.15. Transaccionalidad

Una aplicación empresarial típica accede y almacena información en uno o más bases de datos. Como esta información es crítica para las operaciones de negocios, debe ser precisa, actual, y confiable. Las transacciones controlan el acceso concurrente a los datos por parte de múltiples programas y en evento en que ocurra una falla del sistema, las transacciones aseguran que después de la recuperación los datos estén en un estado consistente.

Una transacción es una unidad indivisible de trabajo que puede finalizar con un commit o con un rollback.

Los métodos **begin ()**, **commit ()** y **rollback ()** demarcan los límites de la transacción.

Las implementaciones JPA realizan automáticamente rollback si cualquier excepción ocurre durante el proceso commit.

Se ha incluido la siguiente línea:

<persistence-unit name="introduccionJPA" transactiontype="

RESOURCE_LOCAL">

En él se indica el nombre de la unidad de transacción, el cual deberemos proporcionar a

EntityManager para informarle de los parámetros necesarios para realizar el mapeo para un conjunto de entidades a gestionar (en nuestro caso solo Película), y el tipo de transacción que utilizaremos al manejar este conjunto de entidades, RESOURCE_LOCAL.

El concepto de transacción nos permite realizar varias operaciones como si fueran una sola, de manera que o bien todas o ninguna de ellas son completadas. Esto es muy importante para garantizar la integridad de los datos que queremos persistir.

Imaginemos que estamos persistiendo una entidad que contiene una lista de entidades en su interior (una asociación uno-a-muchos). Si cualquiera de las entidades de esta lista no pudiera ser persistida por algún tipo de error, no deseamos que el resto de entidades de la lista, así como la entidad que las contiene, sean persistidas tampoco. Si permitiéramos que esto ocurriera, nuestros datos no reflejarían su estado real y completo, y ni dichos datos ni nuestro programa seria fiable. JPA nos permite configurar en cada unidad de persistencia el tipo de transacción, que puede ser:

- Manejada por la aplicación (RESOURCE_LOCAL)
- Manejada por el contenedor (JTA)

Nosotros hemos decidido manejar las transacciones desde la aplicación, y lo haremos en nuestro código a través de la interface EntityTransaction. Aunque en una aplicación real utilizaríamos transacciones manejadas por el contenedor, para mantener los ejemplos simples (no necesitamos instalar y configurar un contenedor) y comprender mejor como funcionan las manejaremos desde nuestro código.

2.3.2.16. Estado de una entidad

Para JPA, una entidad puede estar en dos estados diferentes: managed (gestionada) y de tached (separada). Cuando persistimos una entidad (como veremos en el punto 3.4), automáticamente se convierte en una entidad gestionada. Todos los cambios que efectuemos en ella dentro del contexto de una transacción se verán reflejados también en la base de datos.

El segundo estado en el que puede estar una entidad es separado, en el cual los cambios realizados en la entidad no están sincronizados con la base de datos. Una entidad está en estado separado antes de ser persistida por primera vez y cuando es separada del contexto de persistencia

2.3.2.17. Métodos Callback

Unaentidad puede estar en dos estados diferentes: gestionada y separada. Los métodos callback son métodos que se ejecutan cuando ciertos eventos ocurren en el ciclo de vidade una entidad. Estos eventos se clasifican en cuatro categorías:

- Eventos de persistencia (método callback asociado anotado con @PrePersists y@PostPersist)
- Eventos de actualización (método callback asociado anotado con @PreUpdate y@PostUpdate)
- Eventos de borrado (método callback asociado anotado con @PreRemove y@PostRemove)
- Eventos de carga (método callback asociado anotado con @PostLoad)

Las anotaciones superiores pueden marcar un método de negocio, el cual será ejecutadocuando el evento correspondiente suceda:

```
@Entity
public class Pelicula {
...
@PrePersist
@PreUpdate
private void validar() {
// Validarparámetros antes de persistir y actualizar laentidad}}
```

Los métodos callback deben seguir algunas reglas para que el código funcione:

- No deben ser métodos estáticos ni finales
- Cada anotación de ciclo de vida puede aparecer una y solo una vez en cada entidad(no puede anotar dos o mas métodos)
- No pueden lanzar excepciones de tipo checked
- No pueden invocar métodos de las clases EntityManager y Query

Cuando existe herencia entre entidades, los métodos @Pre/PostXxx de la superclase soninvocados antes que los de la subclase. Así mismo y como vimos antes, los eventos de -ciclo de vida se propagan en cascada cuando existen asociaciones.

2.3.2.18. Clases Listener

Cuando necesitas aplicar un mismo método callback a varias entidades, es preferible extraer este método de la entidad en una clase externa, que podrá ser aplicada a variasentidades, reduciendo así la duplicación de código. Estas clases son clases listener:

```
public class ValidadorListener {
@PrePersist
@PreUpdate
public void validar(Pelicula pelicula) {
// validar parámetros antes de persistir y actualizar la entidad
}
}
```

Una vez definida la clase listener, debemos indicar a cada entidad que clases listenerspuede utilizar:

```
@Entity
@EntityListeners(ValidadorListener.class)
public class Pelicula { ... }
```

Si deseamos que una clase listener pueda ser aplicada amás de un tipo de entidad podemos utilizar una superclase o interface común a dichasentidades, o una referencia a Object como parámetro del método callback:

```
public class ValidadorListener {
@PrePersist
@PreUpdate

public void validar(Object obj) {
// validar parámetros antes de persistir y actualizar
laentidad}}
```

La anotación @EntityListeners puede hacer referencia a varias clases listener, lascuales deben ser incluidas entre llaves y separadas por comas. Cuando una entidad esasociada con varias clases listener, los métodos callback de dichas clases que haganreferencia a un mismo evento serán invocados en el orden en que fueron declaradas loslisteners. Así mismo, los métodos listener serán invocados previamente a los métodoscallback que hagan referencia al mismo tipo de evento y que existan como métodos propios (no externos) de la entidad:

```
@Entity
@EntityListeners({
ValidadorListener.class,
OtroListener.class})
public class Pelicula {
...
@PrePersist
@PreUpdate
private void validar() {
// validar parametros antes de persistir y actualizar la
entidad}}
```

En el ejemplo anterior, los métodos @PrePersist y @PreUpdate se ejecutaran en esteorden: ValidatorListener, OtroListener, Pelicula.

2.3.2.19. Ventajas de JPA

La Java Persistence API constituye un nuevo concepto de programación que:

- Integra conceptos de muchas infraestructuras existentes como Hibernate, Toplink yJDO.
- Se puede usar tanto en entornos Java SE, así como en Java EE.
- Permite que diferentes proveedores de persistencia se puedan usar sin afectar elcódigo del entity.
- Evita tener que capturar información de la pantalla y construir nuestras sentencias SQL, ya que mediante una configuración xml y anotaciones sobre el código fuentepodemos mapear directamente de objetos en memoria a tablas en bases de datos.
- Esto, a diferencia de un API como JDBC, permite pensar más en objetos que en tablasy como acceder a los datos en ellas.
- Simplicidad: Una única clase para declarar la persistencia (con la ayuda deanotaciones).
- Facilidad de aprendizaje.
- Transparencia: Las clases a persistir son simples POJOs.

- No hay restricciones con respecto a relaciones entre objetos (herencia, polimorfismo).

2.3.2.20. 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.

CAPITULO III

3. MARCO METODOLÓGICO

3.1. OPERACIONALIZACIÓN METODOLÓGICA DE LA VARIABLE INDEPENDIENTE

Tabla III.I. Operacionalización Variable Independiente.

HIPOTESIS	VARIABLE	INDICADOR	PARAMETRO	TÉCNICA	FUENTES DE VERIFICACIÓN
El uso del **Framework de persistencia** TopLink permitirá construir aplicaciones web con bajos niveles de vulnerabilidad a ataques externos.	Framework de Persistencia	Conexiónes (100)	1. Trabajo con conexiones (60)	Auditoría Base Oracle para Conexiones	SQL Plus de Oracle Monitor de salida de Oracle mediante ejecución de sentencias SQL.
			2. Librerías Utilizadas (40)	Revisión de Estándares	Definición de Estándares J2E
		Operaciones CRUD (100)	1. Métodos CRUD (60)	Auditoria de la Base de Datos	Monitor de salida de Oracle mediante ejecución de sentencias SQL.
			2. Líneas de Código (20)	Conteo de líneas de código (**Practiline Source**)	Anexo 4
			3. Uso de Procedimeintos Almacenados	Observación directa	Anexo 5

3.2. OPERACIONALIZACIÓN METODOLÓGICA DE LA VARIABLE DEPENDIENTE

Tabla III.II. Operacionalización Variable Dependiente.

HIPOTESIS	**VARIABLE**	**INDICADOR**	**PARAMETRO**	**TÉCNICA**	**FUENTES DE VERIFICACIÓN**
El uso del Framework de persistencia TopLink permitirá construir aplicaciones web con **bajos niveles de vulnerabilidad** a ataques externos.	Niveles de vulnerabilidad	Seguridades de la Información (100)	1. Integridad (40)	SQL Injection	Software propietario **Netsparker Community**
			2. Confidencialidad(20)	Manipulación de entradas **(Boolean SQL**)	
			3. Disponibilidad(40)	Ejecución de comandos (**Cross - Site ScriptingXSS**)	

CAPITULO IV

4. PRUEBAS Y ANÁLISIS DE RESULTADOS

Para realizar las pruebas y el análisis del Sistema "SEACE" se comparó el módulo con persistencia en este caso Usuarios y el módulo Responsable sin persistencia, tomando en cuenta que el sistema cuenta varios módulos para su completo funcionamiento, de este modo se pudo ir comparando cada uno de las técnicas utilizadas, con los diferentes indicadores y parámetros planteados en la operacionalización de la hipótesis.

El procesamiento de la información se lo realizó basándose en los indicadores de las dos variables: independiente y dependiente y a su vez se consideró cada uno de los parámetros que conforman cada indicador.

Para dar valor a cada uno de los indicadores se utilizó una media ponderada de los respectivos parámetros de la siguiente manera.

Para la cuantificación de cada parámetros se basó en ciertas alternativas o escalas que van de uno a cuatro niveles de acuerdo a la aplicabilidad de cada ámbito del parámetro.

4.1. VARIABLE INDEPENDIENTE: Framework de persistencia

4.1.1. VALIDACIÓN DE TÉCNICAS A UTILIZAR

4.1.2. INDICADOR No. 1: Conexiones

Desde una perspectiva de programación, hay dos clases principales responsables para el establecimiento de una conexión con una base de datos. La primera clase es Driver Manager, que es una de las clases que realmente proporciona el API JDBC. Driver Manager es responsable de manejar un almacén de drivers registrados, esencialmente abstrayendo los detalles del uso de un driver para que el programador no tenga que tratar con ellos directamente. La segunda clase es la clase real del Driver JDBC. Estas son proporcionadas por vendedores independientes. La clase Driver JDBC es la responsable de establecer la conexión con la base de datos y de manejar todas las comunicaciones con la base de datos.

El parámetro se considera 1 se considera que debe llevar un valor de 60 puntos sobre 100 debido a su importancia, ya que las conexiones es una parte esencial en los sistemas web.
El parámetro 2 de damos la valoración de 40 puntos sobre 100.

Tabla IV.I. Alternativas de evaluación para la variable independiente

N.	ALTERNATIVAS DE EVALUACIÓN			
1	ALTA	SIEMPRE	TOTALMENTE	SI
2	MEDIANA	FRECUNTEMENTE	BASTANTE	
3	BAJA	A VECES	POCO	
4	MUY BAJA	NUNCA	NADA	NO

Ponderación:

Tabla IV.II. Ponderaciones del indicador 1 Variable Independiente

INDICADOR N° 1.Conexiones		
N	**PARAMETROS**	**PESO**
1	Trabajo con Conexiones	60
2	Librerías utilizadas	40

Altenativas de evaluación elegidas

Tabla IV.III. Alternativas de evaluación para indicador 1 variable independiente, parámetro 1

ALTERNATIVAS DE EVALUACIÓN			
N.	**Calificador**	**Tiempo (miliseg.)**	**Intervalo**
3	Alta	8.31 - 11.00	20
2	Mediana	4.31 – 8.30	20
1	Baja	0.10 – 4.30	20

Tabla IV.IV. Alternativas de evaluación para indicador 1 variable independiente, parámetro 2

ALTERNATIVAS DE EVALUACIÓN		
N|°	**Calificador**	**Peso**
4	Siempre	10
3	Frecuentemente	10
2	A Veces	10
1	Nunca	10

No Uso de Persistencia

Parámetro: Trabajo con conexiones

TablaIV.V. Promedio parámetro 1 del indicador 1 variable Independiente Sin Persistencia

Fuente: Anexo 1

Técnica: Auditoria de Base de datos Oracle					
Operación	**No**	**Calificador**	**Intervalo**	**Promedio**	**Total**
Crear	4	Alta	7.31 - 11.00	9.91	20
Actualizar	4	Alta	7.31 - 11.00	9.92	20
Eliminar	4	Alta	7.31 - 11.00	9.91	20
Promedio General				**9.91**	**20**

Parámetro: Librerías Utilizadas

TablaIV.VI. Promedio parámetro 2 del indicador 1 variable Independiente Sin Persistencia

Fuente: Anexo 1

Técnica: Estándar de Java			
Operación	**No**	**Calificador**	**Total**
ojdbc	4	Siempre	10
odbc	2	A veces	10
Toplink-essentials.jar	1	Nunca	10
Toplink-essentials-agent.jar	1	Nunca	10
Promedio General			**10**

Uso de Persistencia

Parámetro: Trabajo con conexiones

TablaIV.VII. Promedio parámetro 1 del indicador 1 variable Independiente Con Persistencia

Fuente: Anexo 1

Técnica: Auditoria de Base de datos Oracle					
Operación	**No**	**Calificador**	**Intervalo**	**Promedio**	**Total**
Crear	4	Alta	7.31 - 11.00	10.25	20
Actualizar	4	Alta	7.31 - 11.00	10.26	20
Eliminar	4	Alta	7.31 - 11.00	10.18	20
Promedio General				**10.23**	**60**

Parámetro: Librerías Utilizadas

TablaIV.VIII. Promedio parámetro 2 del indicador 1 variable Independiente Sin Persistencia

Fuente: Anexo 1

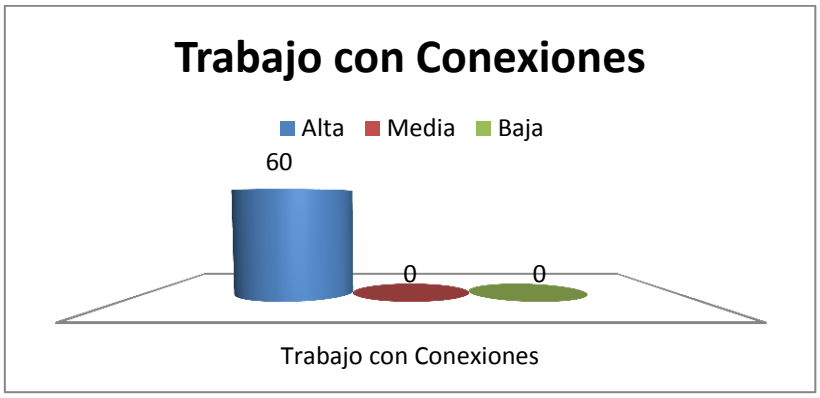
Técnica: Auditoria de Base de datos			
Operación	**No**	**Calificador**	**Total**
ojdbc	3	Frecuentemente	10
odbc	1	Nunca	10
toplink-essentials.jar	4	Siempre	10
toplink-essentials-agent.jar	4	Siempre	10
Promedio General			**10**

RESUMEN DE LAS VALORACIONES DE LA VARIABLE INDEPENDIENTE

No Uso de Persistencia

TablaIV.IX. Resumen de Evaluación parámetro 1, indicador 1 variable Independiente Sin Persistencia

N0.	60	60%
	Trabajo con Conexiones	**Total**
3	60	60
2	0	0
1	0	0
Promedio	60	60



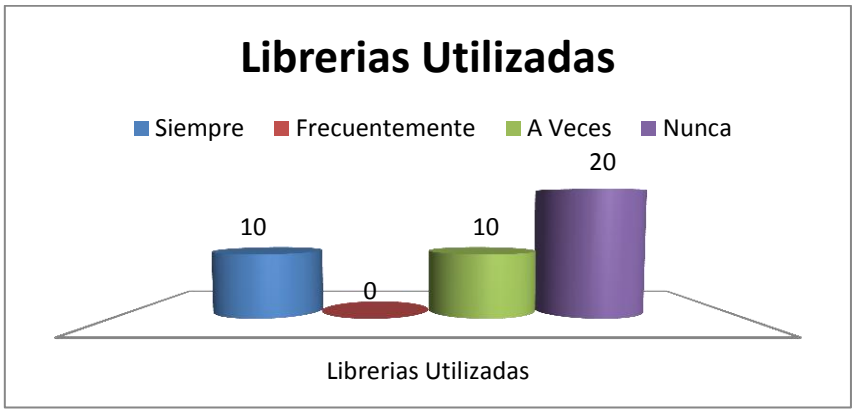
FiguraIV.27. Trabajo con conexiones sin uso de persistencia

Análisis

Como se puede ver el parámetro Trabajo con conexiones tiene un porcentaje alto con un 60 % en el calificador Alta mientras en los demás tiene valor 0%. Lo cual nos indica que el trabajo con conexiones sin el uso de persistencia tiene un corto periodo de tiempo de duración de la persistencia, para realizar todas las acciones a la base de datos desde la aplicación. .

Tabla III.X Resumen de Evaluación parámetro 2, indicador 1 variable Independiente Sin Persistencia

N0.	40	40%
	Librerías Utilizadas	**Total**
4	10	10
3	0	0
2	10	10
1	20	20
Promedio	40	40



FiguraIV.28. Librerías Utilizadas sin uso de persistencia

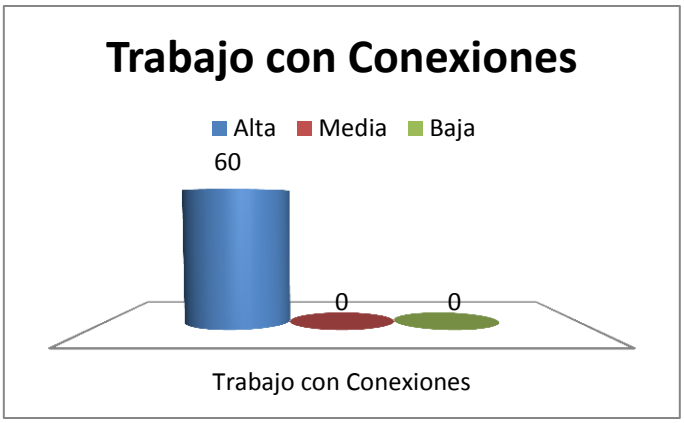
Análisis

Como se puede ver el parámetro Librerías Utilizadas tiene un 10 % en el calificador Siempre; en el calificador A veces tiene un 10 % y en el calificador Nunca posee un valor de 20 % lo cual nos indica que la librerìa del JDBC no depende del proveedor de persistencia siendo de esta forma independiente para realizar sus acciones hacia la base.

Con Uso de Persistencia

TablaIV.XI. Resumen de Evaluación parámetro 1 indicador 1 variable Independiente con Persistencia

N0.	60	100%
	Trabajo con Conexiones	**Total**
4	60	60
3	0	0
2	0	0
Promedio	60	60



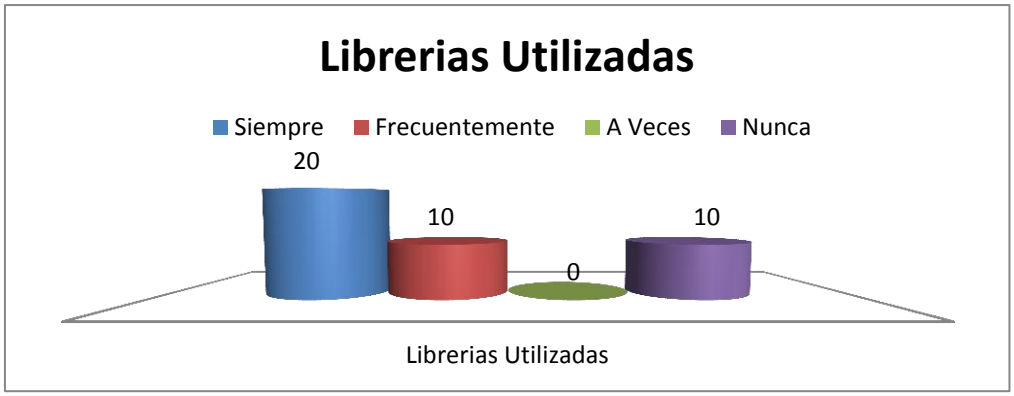
FiguraIV.29. Trabajo con conexiones con uso de persistencia

Análisis

Como se puede ver el parámetro Trabajo con conexiones tiene un porcentaje alto, con un 60%en el calificador Alta mientras los demás calificadores tienen el porcentaje 0%.Lo cual nos indica que el trabajo que realiza el proveedor de persistencia a través de sus conexiones para realizar las acciones hacia la base de datos, manteniendo sus conexiones persistentes con largos tiempos de duración ya que el proveedor de persistencia depende del JDBC.

Tabla IV.XII. Resumen de Evaluación parámetro 2 indicador 1 variable Independiente con Persistencia

N0.	40	100%
	Librerías Utilizadas	**Total**
4	20	20
3	10	10
2	0	0
1	10	10
Promedio	40	40



FiguraIV.30. Librerías Utilizadas Con uso de persistencia

Análisis

Como se puede ver el parámetro Librerías Utilizadas tiene un 20 % en el calificador Siempre; en el calificador Frecuentemente tiene un 10 % y en el calificador Nunca posee un valor de 10 %. Lo cual nos indica que es indispensable utilizar las librerías en el momento en que se este usando el proveedor de persistencia para realizar las acciones.

TablaIV.XIII. Resumen de promedios indicador 1 variable Independiente con y sin uso de persistencia

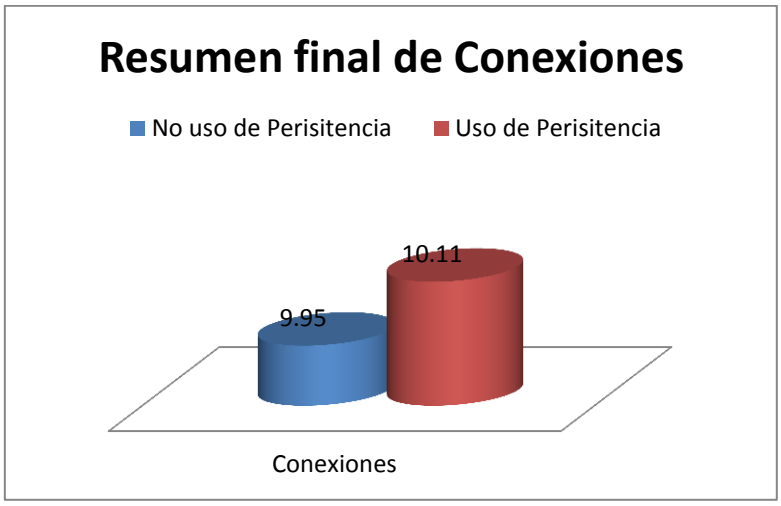
Parámetro	**No uso de Persistencia**	**Uso de Persistencia**
Trabajo con Conexiones	9.91	10.23
Librerías Utilizadas	10	10
Total	19.91	20.23
Promedio	9.95	10.11

FiguraIV.31. Librerías Utilizadas Con uso de persistencia

Análisis

Para el parámetro Trabajo con Conexiones mediante la programación tradicional se tiene un porcentaje del 9.91 % mientras al hacer uso de persistencia se tiene un porcentaje de 10.23%, esto significa que el proveedor de persistencia por naturaleza mantiene muchas conexiones abiertas para realizar sus acciónes las mismas que son reutilizadas varias veces, de esta forma optimizando recusros de memoria y procesamiento.

En el parámetro Librerías Utilizadas se refleja una igualdad de porcentajes, la razón es que para poder iniciar una conexión tanto en el uso de persistencia y en el no uso, se requiere de un driver y un conjunto de librerías para poder establecer una conexión es decir que están relacionadas las librerías con los Drivers de conexión a la base.



FiguraIV.32. Resumen Final del indicador 1 Variable Independiente Conexiones

Análisis

En esta figura se puede ver notoriamente que con el uso de persistencia aumenta el valor de Conexiones con un porcentaje de 10.11% mientras que sin uso de persistencia se tiene un porcentaje de 9.95 % ya que al utilizar persistencia mantiene muchas conexiones abiertas para realizar una acción, mediante la cual se optimiza recursos, procesamiento y se mejora el rendimiento.

4.1.3. INDICADOR N. 2: Operaciones CRUD

CRUD es el acrónimo de Crear, Obtener, Actualizar y Borrar (del original en inglés: Create, Read, Update and Delete).

Es usado para referirse a las funciones básicas en bases de datos o la capa de persistencia en un sistema de software. Las operaciones CRUD son importantes ya que con ellas podemos manipular la base de datos según nuestra necesidad.

El parámetro 1 se considera que debe llevar un valor de 60 puntos sobre 100 debido a su importancia, ya que con este parámetro podemos manipular la base de datos.

El Parámetro 2 y 3 se le han dado una valoración de 20.

Ponderación:

Tabla IV.XIV. Ponderaciones del indicador 2

INDICADOR N° 2. Operaciones CRUD		
N	**PARAMETROS**	**VALORACIÓN**
1	Métodos CRUD	60
2	Líneas de Código	20
3	Procedimientos Almacenados	20

Alternativas de Evaluación

Tabla IV.XV. Alternativas de evaluación para indicador 2 variable independiente, parámetro 1

ALTERNATIVAS DE EVALUACIÓN			
N.	Calificador	Tiempo (miliseg.)	**Intervalo**
4	Alta	7.31 - 11.00	46-60
2	Mediana	6.31 – 7.30	31-45
3	Baja	5.01 – 6.30	16-30
1	Muy baja	0.10 – 5.00	1-15
Total			60

Tabla IV.XVI. Alternativas de evaluación para indicador 2 variable independiente, parámetro 2

ALTERNATIVAS DE EVALUACIÓN			
N|°	**Calificador**	**Líneas de Código**	**Intervalo**
4	Alta	501 - 600	16-20
3	Mediana	401 –500	11-15
2	Baja	301 – 400	6-10
1	Muy baja	200 – 300	1-5
Total			20

Tabla IV.XVII. Alternativas de evaluación para indicador 2 variable independiente, parámetro 3

ALTERNATIVAS DE EVALUACIÓN		
N|°	**Calificador**	**Peso**
4	Si	20
1	No	0
Total		20

VALORACION IDICADOR: Operaciones CRUD

No Uso de Persistencia

Parámetro: Métodos CRUD

TablaIV.XVIII Promedio parámetro 1 del indicador 2 variable Independiente Sin Persistencia

Fuente: Anexo 2

Técnica: Auditoria de Base de datos					
Operación	**No**	**Calificador**	**Intervalo**	**Promedio**	**Valor Real**
Crear	4	Alta	7.31 - 11.00	9.15	49.90
Actualizar	4	Alta	7.31 - 11.00	9.42	51.38
Eliminar	4	Alta	7.31 - 11.00	9.41	51.32
Promedio General				**9.32**	50.83

Parámetro: Líneas de Código

TablaIV.XIX. Promedio parámetro 2 del indicador 2 variable Independiente Sin Persistencia

Fuente: Anexo 2

Técnica: Conteo de líneas de Código				
Archivo Java	**N°**	**Calificador**	**Valor**	**Valor real**
Responsable	1	Muy Baja	214	3.56

Parámetro: Procedimiento Almacenado

Tabla IV.XX. Promedio parámetro 3 del indicador 2 variable Independiente Sin Persistencia

Fuente: Anexo 3

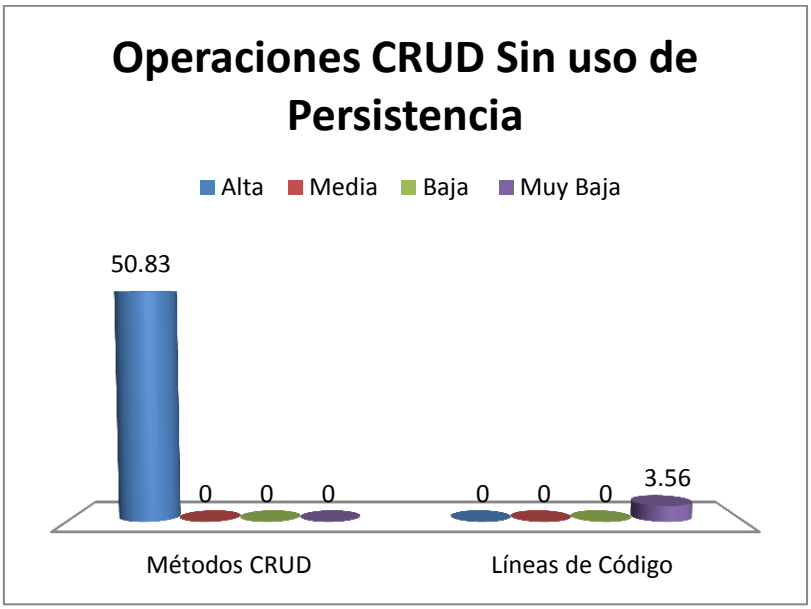
Técnica: Observación Código IDE NetBeans			
	N°	**Calificador**	**Valor**
Utiliza Procedimientos almacenados	4	Si	20
Total			20

RESUMEN DE LAS VALORACIONES DE LA VARIABLE INDEPENDIENTE

No Uso de Persistencia

TablaIV.XXI. Resumen de Evaluación parámetro 1, 2 indicador 2 variable Independiente Sin Persistencia

N0.	60	20	80%
	Métodos CRUD	**Líneas de Código**	**Total**
4	50.83	0	50.83
3	0	0	0
2	0	0	0
1	0	3.56	3.56
Promedio	50.83	3.56	54.39



FiguraIV.33. Operaciones CRUD parámetro 1,2 sin uso de persistencia

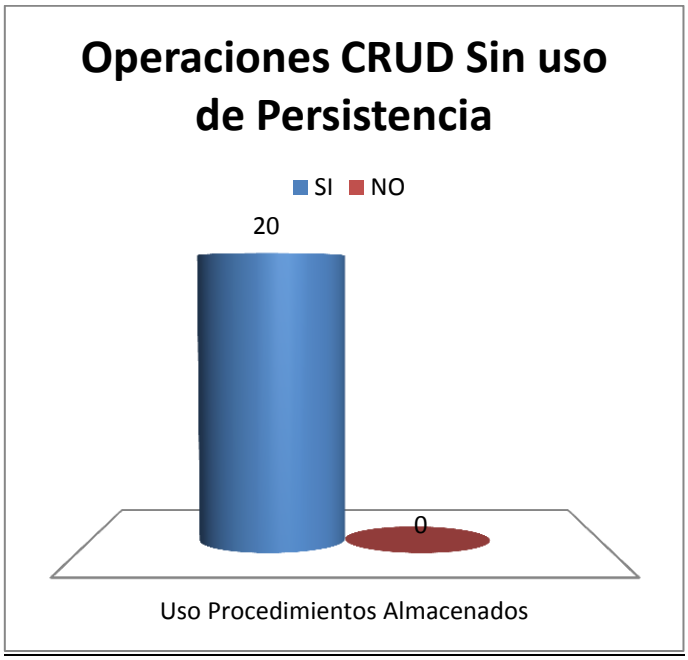
Análisis

Como se puede ver el parámetro Métodos CRUD tiene un nivel alto con un porcentaje de 50 % lo cual nos indica que al utilizar la programación tradicional el tiempo en realizar las operaciones CRUD es considerablemente alto.

El parámetro Lìneas de Còdigo se lo midió basándose en el código necesario para realizar las operaciones CRUD, en la gráfica podemos observar que tiene un porcentaje del 3.56% y se ubica en Muy baja lo que nos indica que al utilizar la programación tradicional no requiere de un proceso especìfico.

TablaIV.XXII. Resumen de Evaluación parámetro 3 indicador 2 variable Independiente Sin Persistencia

N0.	20	20%
	Uso Procedimientos Almacenados	**Total**
4	20	20
3	0	0
2	0	0
1	0	0
Promedio	20	20



FiguraIV.34. Operaciones CRUD parámetro 3 sin uso de persistencia

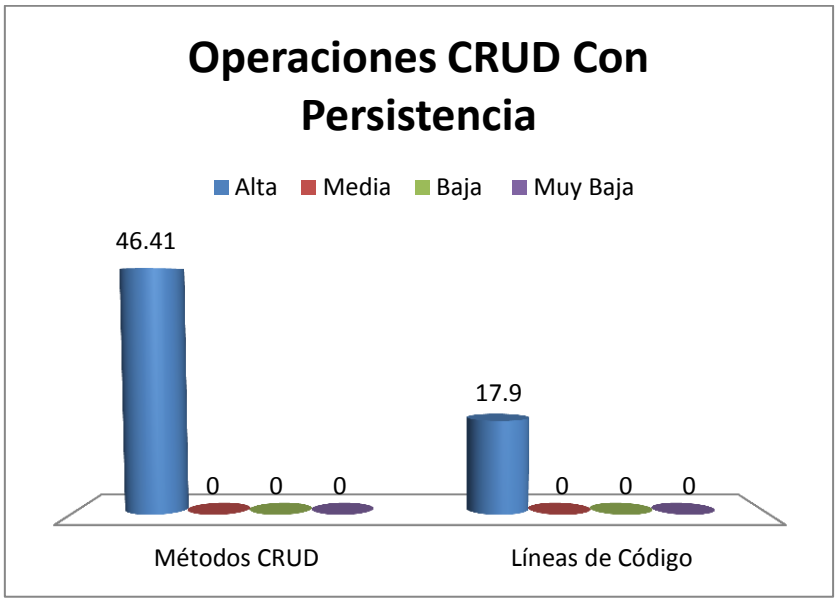
Análisis

El Parámetro Uso Procedimientos Almacenados tiene un porcentaje del 20 % de su totalidad lo cual nos indica que al utilizar la programación tradicional se usan procedimientos almacenados para ocultar la operación que se realice en la base de datos simplificando al desarrollador la escritura del lenguaje SQL.

Uso de Persistencia

Tabla IV.XXIII. Resumen de Evaluación parámetro1, 2 indicador 2 variable Independiente Con Persistencia

N0.	60	20	80%
	Métodos CRUD	**Líneas de Código**	**Total**
4	46.41	17.9	64.31
3	0	0	0
2	0	0	0
1	0	0	0
Promedio	46.41	17.9	64.31



FiguraIV.35. Operaciones CRUD parámetro 1,2 con uso de persistencia

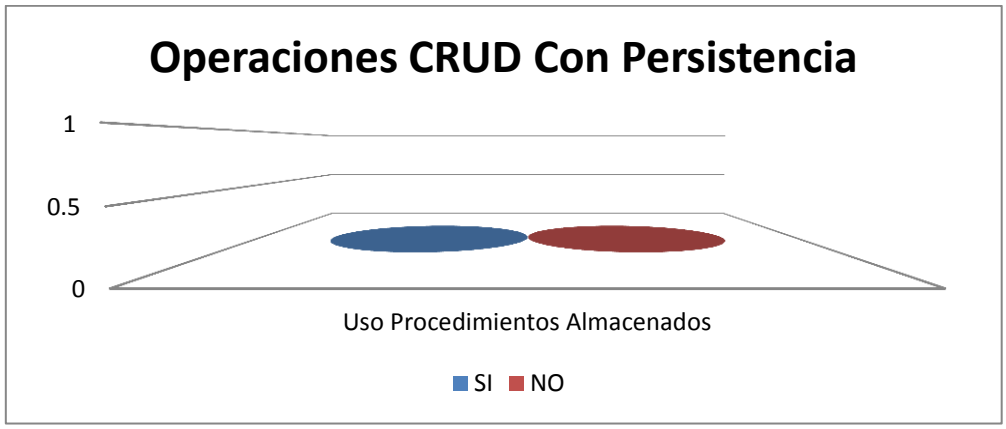
Análisis

El parámetro Métodos CRUD tiene un nivel alto con un porcentaje del 45 % lo que nos indica que al utilizar el proveedor de persistencia el tiempo que se demora en realizar las operaciones CRUD es menor en relación a la rpogramación tradicional.

El parámetro Lìneas de Còdigo se lo midió basándonos en el código necesario para realizar las operaciones CRUD, en este caso el código es genrado por el API de persistencia en base a una estrategia de mapeo, en la gráfica podemos observar que tiene un nivel alto con un porcentaje del 17% superando al porcentaje obtenido con la programación tradicional.

Tabla IV.XXIV. Resumen de Evaluación parámetro 3 indicador 2 variable Independiente Con Persistencia

20	20%
Utiliza Procedimientos Almacenados	**Total**
0	0
0	0
0	0
0	0
0	0



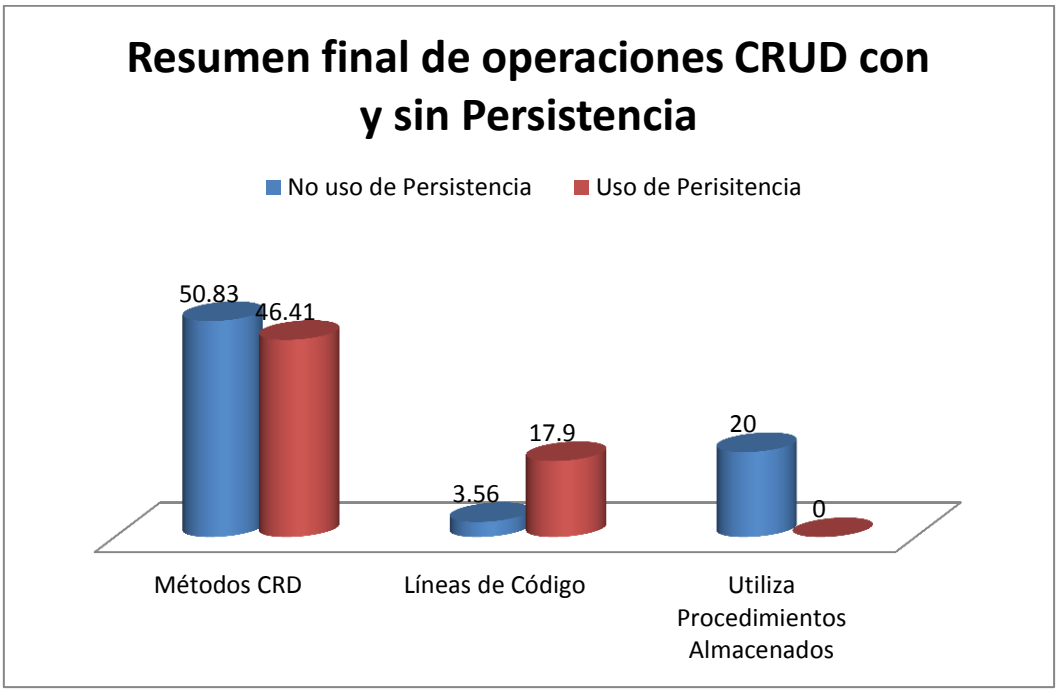
FiguraIV.36. Operaciones CRUD parámetro 1,2 con uso de persistencia

Análisis

El Parámetro Uso de Procedimientos Almacenados tiene un porcentaje del 0% el mismo que nos indica que el framework de persitencia JPA/TopLink no admite el uso de procedimientos almacenados ya que utiliza su propia técnica para realizar las operaciones CRUD.

TablaIV.XXV. Resumen de promedios indicador 2 variable Independiente con y sin uso de persistencia

Parámetro	**No uso de Persistencia**	**Uso de Persistencia**
Métodos CRUD	50.83	46.41
Líneas de Código	3.56	17.9
Utiliza Procedimientos Almacenados	20	0
Total	74.39	64.31
Promedio	24.79	21.43



FiguraIV.37. Resumen Final de Operaciones CRUD con y sin uso de persistencia

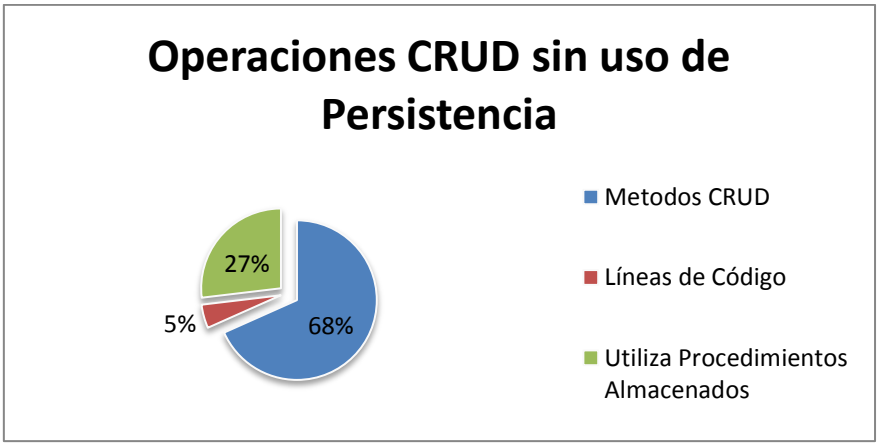
Análisis

Mediante el parámetro Métodos CRUD se puede determinar el tiempo promedio de ejecución de una operación, con la programación tradicional se tiene un porcentaje del 50% lo cual nos indica que tiene una duración alta, mientras que con el proveedor de persistencia se tiene un porcentaje de 46%, reduciendo tiempos de ejecución de un 4%.

Para el parámetro Líneas de Código a través del desarrollo tradicional se tiene un porcentaje del 3% de código escrito para el funcionamiento de una acción, mientras que al hacer uso del proveedor de persistencia tiene un porcentaje alto con el 17% de código generado para el funcionamiento de una acción, esté código se genera a través de técnicas para optimización de recursos.

Para el parámetro Procedimientos Almacenados mediante el desarrollo tradicional se tiene un porcentaje alto del 20% ya que su uso es altamente significativo para realizar acciones en la base de Datos, mientras que el framework de persistencia JPA/TopLink no admite el uso de procedimientos almacenados, en lugar de esto hace uso de sus propias interfaces a

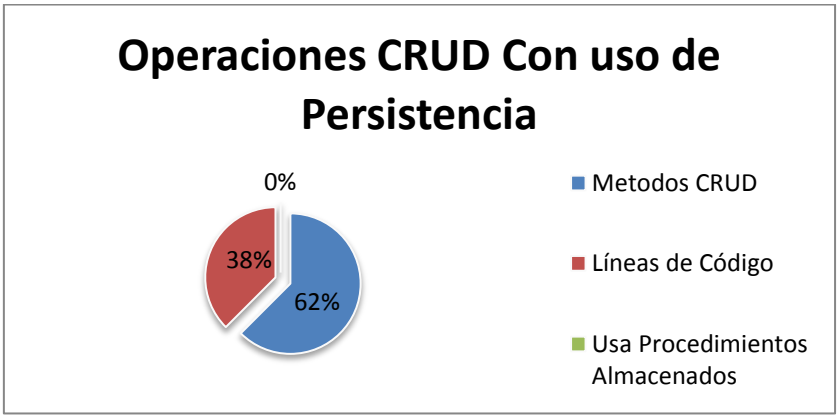
través del código generado por el ORM que facilita la interacción del modelo con la Base de Datos.



FiguraIV.38. Resumen final de Operaciones CRUD sin uso persistencia

Análisis

En la figura se puede ver que el parámetro Métodos CRUD tiene un 68% , las Líneas de Código tiene un 5%, y el Uso de Procedimientos Almacenados con el 27% los mismos que representan las ventajas en líneas de Código pero la desventaja en tiempo y rendimiento al no Utilizar persistencia.



FiguraIV.39. Resumen final de Operaciones CRUD con uso de persistencia

Análisis

En la figura se puede ver que el parámetro Métodos CRUD tiene un porcentaje del 62% lo que nos indica que el tiempo que se demora en realizar las operaciones CRUD utilizando persistencia es menor al al tiempo que se demora al utilizar la programación tradicional; las Líneas de Código tiene un porcentaje del 38% ya que al Utilizar Persistencia se genera código necesario que el programador lo omite, y el Uso de Procedimientos Almacenados tiene un porcentaje del 0% lo cual indica que el framework de persitencia JPA/TopLink no admite el uso de procedimientos almacenados ya que utiliza su propia técnica para realizar las operaciones CRUD.

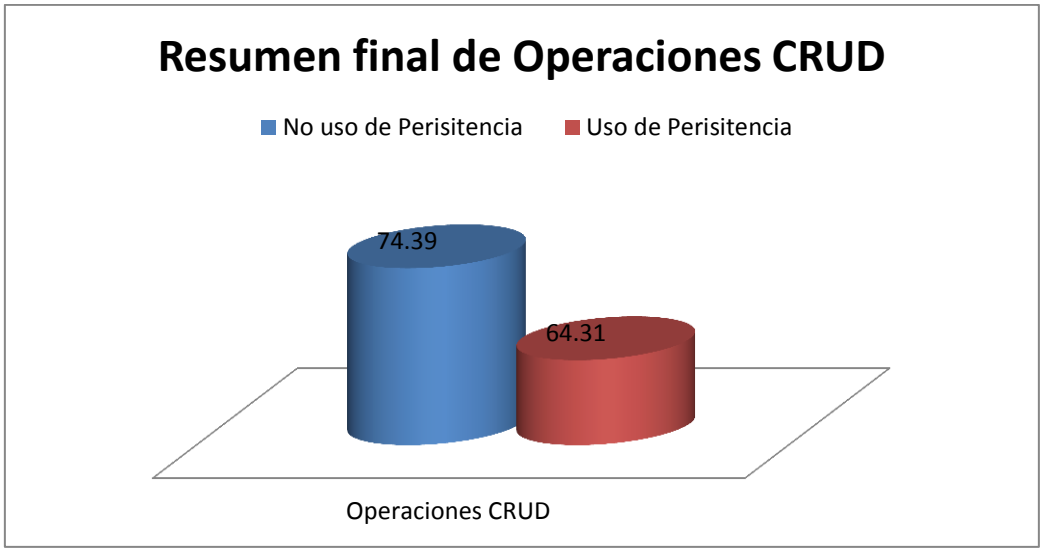


Figura IV.40. Resumen final de operaciones CRUD

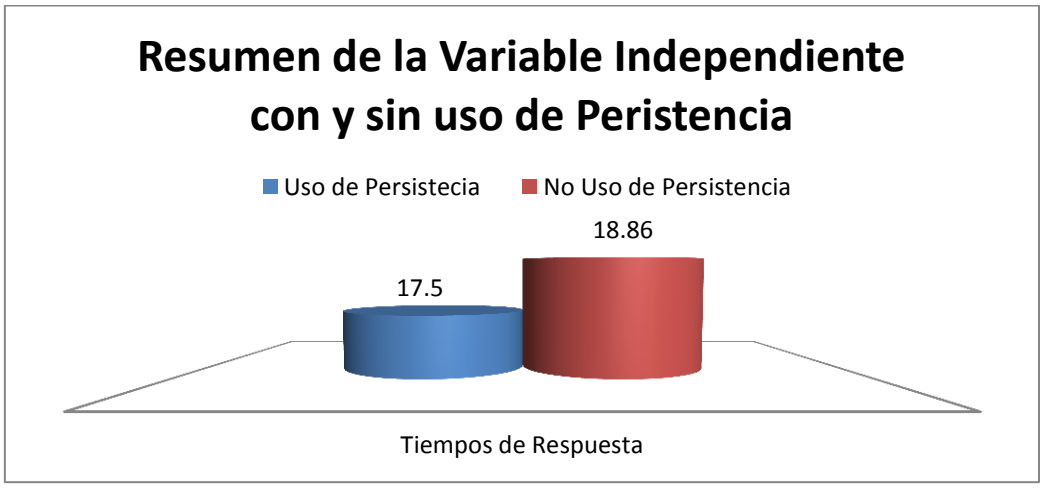
Análisis

En esta figura se puede ver notoriamente que con el uso de persistencia se disminuye el tiempo de respuesta de la Aplicación y a la vez se optimiza el rendimiento con un porcentaje de 64.31 % mientras que sin uso de persistencia se tiene un porcentaje de 74.39%.

Indicador	Parámetros	Uso de Persistencia	No Uso de persistencia
Conexiones	Trabajo con Conexiones	10.23	9.91
	Librerías Utilizadas	10	10
Operaciones CRUD	Métodos CRUD	46.41	50.83
	Líneas de Código	17.90	3.56
	Uso de Procedimientos almacenados	0	20
Total		87.54	94.3
Promedio		17.50	18.86

Tabla IV.XXVI. Resumen de Evaluación de la Variable Independiente

4.1.4. RESUMEN DE LA VARIABLE INDEPENDIENTE DE LOS INDICADORES 1 Y 2



FiguraIV.41. Resumen de la Variable Independiente

Análisis

En la siguiente tabla se pude ver que la variable independiente al utilizar el framework de persistencia para la construcción de aplicaciones web reduce los tiempos de respuesta optimizando recursos y procesamiento, incrementando la productividad de sus serivicons a través del sistema SEACE ya que el promedio es del 17.50 %, mientras que al usar la programación tradicional se obtiene un promedio del 18.86 % superando con el 1.36 % al No usar persistencia en tiempos de ejecución y uso de recursos.

4.2. DEMOSTRACION DE LA VARIABLE DEPENDIENTE

INDICADOR

Seguridad de la Información

La seguridad, en informática como en otras áreas, se basa en la protección de activos. Estos activos pueden ser elementos tan tangibles como un servidor o una base de datos, o pueden ser la reputación de una empresa. Generalmente podemos evaluar la seguridad de un activo en base a tres aspectos principales que no necesitan explicación: integridad, disponibilidad, confidencialidad.

PARAMETROS

Integridad

Para la Seguridad de la Información, la integridad es la propiedad que busca mantener los datos libres de modificaciones no autorizadas. (No es igual a integridad referencial en bases de datos.) La violación de integridad se presenta cuando un empleado, programa o proceso (por accidente o con mala intención) modifica o borra los datos importantes que son parte de la información, así mismo hace que su contenido permanezca inalterado a menos que sea modificado por personal autorizado, y esta modificación sea registrada, asegurando su precisión y confiabilidad. [16]

Confidencialidad

La confidencialidad es la propiedad de prevenir la divulgación de información a personas o sistemas no autorizados. Que nadie pueda leer la información excepto el destinatario. [4]

La pérdida de la confidencialidad de la información puede adoptar muchas formas. Cuando alguien mira por encima de su hombro, mientras usted tiene información confidencial en la pantalla, cuando se publica información privada, cuando un laptop con información sensible sobre una empresa es robado, cuando se divulga información confidencial a través del teléfono, etc. Todos estos casos pueden constituir una violación de la confidencialidad. [17]

4.2.1. VALIDACION DE LAS TECNICAS A UTILIZAR

SQL Injection

SQL Injection es una vulnerabilidad que afecta aplicaciones a nivel de base de datos. Dicha vulnerabilidad consiste en enviar instrucciones SQL adicionales a partir de parámetros entrada ingresados por el usuario.

Al "inyectar" el código **SQL** malicioso dentro de estos campos, el código “invasor” se ejecuta dentro del código SQL propio de la aplicación para alterar su funcionamiento normal, de acuerdo con el propósito del atacante.

SQL Injection es un problema de seguridad que debe ser tomado en cuenta por el programador para prevenirlo. La vulnerabilidad ocurre cuando la aplicación ejecuta una sentencia SQL que utiliza el valor de campos de entrada sin validarlos correctamente.

Permiten al atacante saltar restricciones de acceso, elevar privilegios del usuario, extracción de información de la base de datos, ejecución de comandos dentro del servidor. Incluso es posible destruir parte la base de datos de la aplicación (por ejemplo insertando una sentencia Drop Table).

Manipulación de Parámetros

Es un conjunto de técnicas que atacan la lógica de negocio de la aplicación, tomando ventaja del uso de campos ocultos o fijos para la transferencia de información sensible entre browser y servidor. En particular, tags ocultos en un form, parámetros anexados a la URL son fácilmente modificables por un atacante.

Ejecución de comandos

Las técnicas de manipulación de entrada vistas son sólo algunas que llevan a la posibilidad de ejecutar remotamente comandos del Sistema Operativo de la víctima.

Determinados caracteres especiales pueden ser interpretados por scripts de validación poco seguros como una instrucción al SO de esperar un comando arbitrario a continuación. En particular, el punto y coma o la barra | son en Unix caracteres que permiten encadenar comandos. Por tanto incluir en el campo de entrada un; seguido del comando que se desea ejecutar puede tener éxito si el mecanismo de validación no es lo bastante robusto.

Tabla IV.XXVII. Alternativas de evaluación para la variable independiente

N.	ALTERNATIVAS DE EVALUACIÓN			
1	ALTA	SIEMPRE	TOTALMENTE	SI
2	MEDIANA	FRECUNTEMENTE	BASTANTE	
3	BAJA	A VECES	POCO	
4	MUY BAJA	NUNCA	NADA	NO

Se asignó peso a cada uno de los índices que conforman un indicador, resultando de esta manera una calificación total por cada tipo de educación evaluada, sobre una valoración de 200.

Alternativas de Evaluación elegidas.

Tabla IV.XXVIII. Valoración del indicador 1 de la variable dependiente

INDICADOR N. 1. Seguridad de la Información		
N.	**PARAMETRO**	**VALORACIÓN**
1	Integridad	Sobre200 p.
2	Confidencialidad	Sobre200 p.
3	Disponibilidad	Sobre200 p.

TablaIV.XXIX Valoración del Parámetro 1, 2, 3 de la Indicador 1 variable dependiente

N.	ALTERNATIVAS DE EVALUACIÓN	
4	Alta	151-200
3	Mediana	101-150
2	Baja	51-100
1	Muy baja	1-50

4.2.2. VALORACION IDICADOR: Seguridad de la Información

No uso de Persistencia

Parámetro: Integridad

TablaIV.XXX. Promedio parámetro 1 del indicador 1 variable dependiente

Fuente Anexo 6

Técnica: SQL INJECTION				
Operación	**No**	**Calificador**	**Intervalo**	**Promedio Vulnerabilidad**
Crear	2	Baja	51-100	70
Actualizar	2	Baja	51-100	84
Eliminar	2	Baja	51-100	85
Listar	2	Baja	51-100	70
Promedio General				**77**

Parámetro: Confidencialidad

TablaIV.XXXI. Promedio parámetro 2 del indicador 1 variable dependiente

Fuente: Anexo 6

Técnica: Manipulación de Parámetros (Boolean SQL)				
Operación	**No**	**Calificador**	**Intervalo**	**Promedio Vulnerabilidad**
Crear	2	Baja	51-100	80
Actualizar	2	Baja	51-100	95
Eliminar	3	Mediana	101-150	105
Listar	2	Baja	51-100	80
Promedio General				**90**

Parámetro: Disponibilidad

TablaIV.XXXII. Promedio parámetro 3 del indicador 1 variable dependiente

Fuente Anexo 6

Técnica: Ejecución de comandos (Cross -Site ScriptingXSS)				
Operación	**No**	**Calificador**	**Intervalo**	**Promedio Vulnerabilidad**
Crear	4	Alta	151-200	159
Actualizar	4	Alta	151-200	174
Eliminar	4	Alta	151-200	174
Listar	4	Alta	151-200	159
Promedio General				**167**

Uso de Persistencia

Parámetro: Integridad

TablaIV.XXXIII. Promedio parámetro 1 del indicador 1 variable dependiente

Fuente Anexo

Técnica: SQL INJECTION				
Operación	**No**	**Calificador**	**Intervalo**	**Promedio Vulnerabilidad**
Crear	2	Baja	51-100	70
Actualizar	2	Baja	51-100	98
Eliminar	1	Muy baja	1-50	27
Listar	2	Baja	51-100	70
Promedio General				**66**

Parámetro: Confidencialidad

TablaIV.XXXIV. Promedio parámetro 2 del indicador 1 variable dependiente

Fuente: Anexo 6

Técnica: Manipulación de Parámetros (Boolean SQL)				
Operación	**No**	**Calificador**	**Intervalo**	**Promedio Vulnerabilidad**
Crear	2	Baja	51-100	80
Actualizar	3	Mediana	101-150	109
Eliminar	1	Muy baja	1-51	35
Listar	2	Baja	51-100	80
Promedio General				**76**

Parámetro: Disponibilidad

TablaIV.XXXV. Promedio parámetro 3 del indicador 1 variable dependiente

Fuente: Anexo 6

Técnica: Ejecución de comandos (Cross -Site ScriptingXSS)				
Operación	**No**	**Calificador**	**Intervalo**	**Promedio Vulnerabilidad**
Crear	4	Alta	151-200	159
Actualizar	4	Alta	151-200	190
Eliminar	3	Mediana	101-150	109
Listar	4	Alta	151-200	159
Promedio General				**154**

4.2.3. RESUMEN DE LAS VALORACIONES DE LA VARIABLE DEPENDIENTE

Tabla IV.XXXVI. Ponderacióndel indicador 1 variable independiente

N	PARAMETRO	PESO
1	Integridad	40
2	Confidencialidad	40
3	Disponibilidad	20

Tabla.IV.XXXVII. Alternativa de Evaluación del parámetro 1, 2 del indicador 1 variable dependiente

ALTERNATIVAS DE EVALUACIÓN			
N.	**Calificador**	**Intervalo**	**Peso**
4	Alta	151-200	10
3	Mediana	101-150	10
2	Baja	51-100	10
1	Muy baja	1-50	10

Tabla IV.XXXVIII. Alternativa de Evaluación del parámetro 3 del indicador 1 variable dependiente

ALTERNATIVAS DE EVALUACIÓN			
N.	**Calificador**	**Intervalo**	**Peso**
4	Alta	151-200	5
3	Mediana	101-150	5
2	Baja	51-100	5
1	Muy baja	1-50	5

No Uso de Persistencia

Tabla IV.XXXIX. Resumen de Evaluación del parámetro 1, 2, 3 del indicador 1 variable dependiente

N0.	40	40	20	100%
	Integridad	**Confidencialidad**	**Disponibilidad**	**Total**
4	0	40	0	40
3	0	0	5	5
2	40	0	15	55
1	0	0	0	0

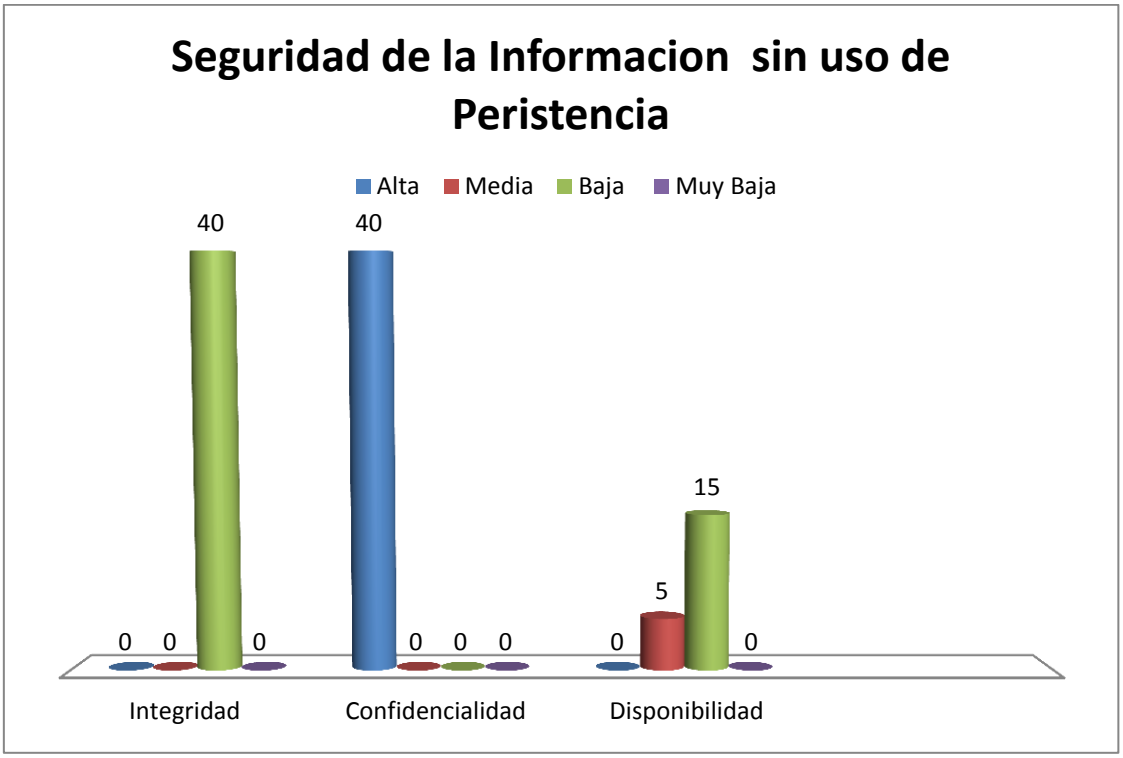


Figura IV.42. Seguridad de la información sin uso de persistencia

Análisis

Como se puede ver el parámetro Integridad tiene un nivel bajo con un 40 % en el calificador Baja mientras en los demás tiene valor 0% puediendo decir que la implementación no tiene los parámetros necesarios para el desarrollo

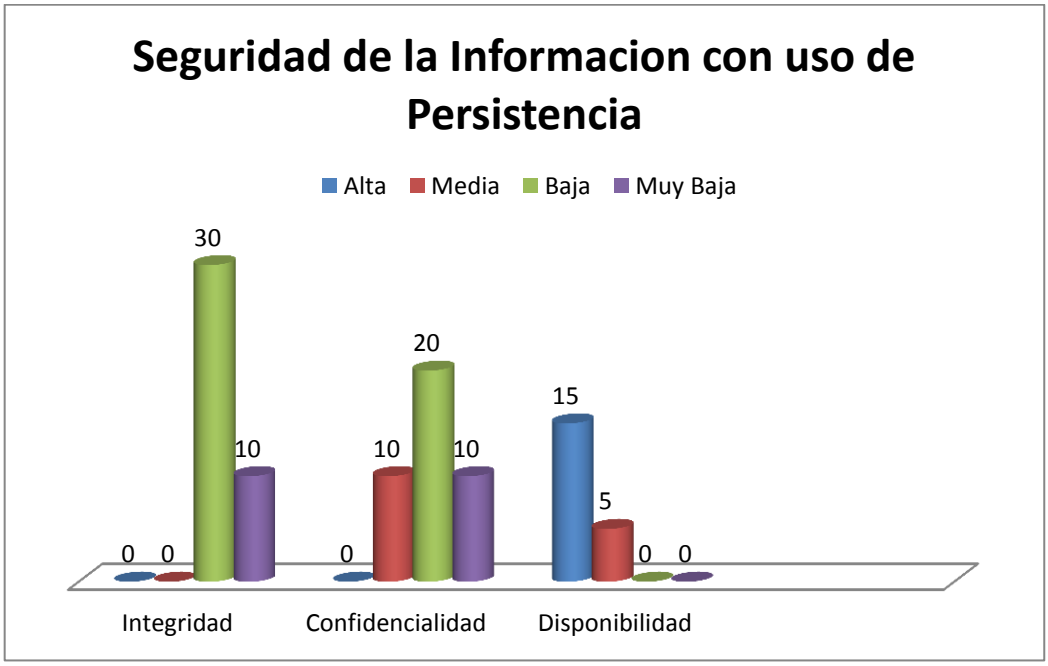
El parámetro Confidencialidad tiene un nivel alto de vulnerabilidad con un 40% en el calificador Alta lo que nos indica que la información puede ser manipulada durante la ejecución de la operación.

El Parámetro Disponibilidad tiene un nivel Media con un porcentaje de 15 % y con un valor significativo del 5% en el calificador bajo, pudiendo decir que la disponibilad de la información puede ser afectada con mediana dificultad por pate del atacante.

Uso de Persistencia

Tabla IV.XL Resumen de Evaluación del parámetro 1, 2, 3 del indicador 1 variable dependiente

N0.	40	40	20	100%
	Integridad	**Confidencialidad**	**Disponibilidad**	**Total**
4	0	0	15	15
3	0	10	5	15
2	30	20	0	50
1	10	10	0	20



FiguraIV.43. Seguridad de la información con uso de persistencia

Análisis

Como se puede ver en la figura el parámetro Integridad tiene un porcentaje de30 % en el calificador Baja y en calificador Muy bajo 10% y en los demás con valor 0%, se puede decir que la vulnerabilidad en la integridad es muy Baja.

El parámetro Confidencialidad en el calificador Media tiene un valor de 10%, en el calificador Baja 20% y en Muy Baja 10% por lo que se puede decir está Garantizada la confidencialidad de la Información,

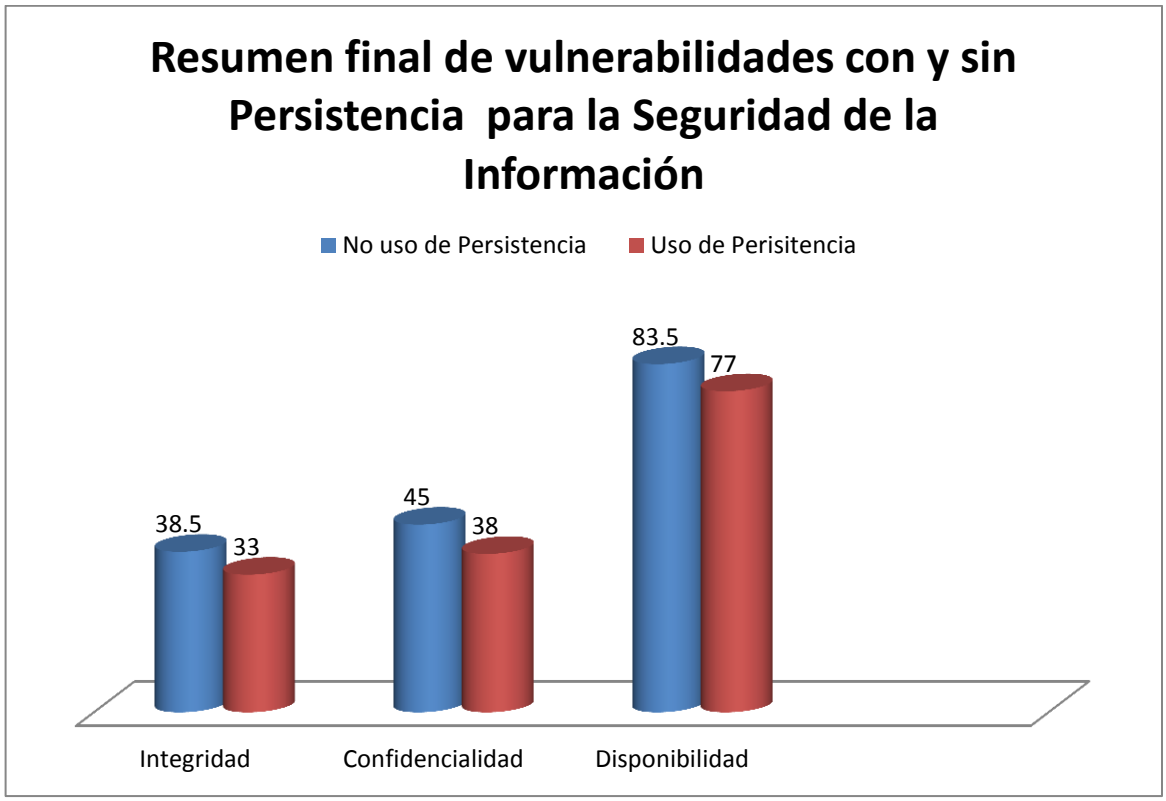
Finalmente el parámetro Disponibilidad tiene en el calificador Alta un 15% y en el calificador Mediana 5% por lo que se puede decir que la Disponibilidad de la Informaciòn puede ser afectada notoriamente significando pérdidas y dificultades de recuperación.

Mediante este análisis se puede determinar el nivel de vulnerabilidad de manera general es relativamente Baja, pudiendo hacer mención que se puede considerar como Baja la efectivización y ejecución de ataques a sistemas Web desarrollados con el Framework de persistencia

Para su valoración en la tabla se toma que el valor 200 es el 100%

Tabla IV.XLI. Resumen de promedios indicador 1 variable dependiente con y sin uso de persistencia

Parámetro	**No uso de Persistencia**	**Uso de Persistencia**
Integridad	38.5	33
Confidencialidad	45	38
Disponibilidad	83.5	77
Total	167	148
Promedio	55.67	49.33



FiguraIV.44. Resumen final de vulnerabilidad para la seguridad de la información

Análisis

Como se puede ver en la figura el parámetro Integridad sin uso de persistencia tiene un porcentaje del 38.5 % mientras que con el uso tiene un porcentaje del 33% teniendo una reducción del 5.5% al utilizar la persistencia pudiendo decir que la informacòn estrà protegida con un alto grado se seguridad.

En el parámetro Confidencialidad sin uso de persistencia tiene un porcentaje de 45% mientras que con el uso tiene un porcentaje de 38% teniendo una reducción del 7% al utilizar la persistencia manteniendo la confidencialidad de la información en el momento de realizar las acciones que afectràn la base de datos..

En el parámetro Disponibilidad sin uso de persistencia tiene un porcentaje del 83.5% mientras que con el uso tiene un porcentaje del 77% teniendo una reducción del 6.5% al utilizar el framework depersistencia puediendo decir que la disponibilidad de la

información no será afectada fácilmente ya que el Framework de persistencia implementa métodos de seguridad para evitar ataques a la disponibildiad de la información.

Finalmente se puede decir que con el uso de persistencia reduce notoriamente los niveles de vulnerabilidad en la seguridad de la información en un promedio de 6.33%, pudiendo decir el con el uso de persistencia se obtiene bajos niveles de vulnerabilidad en relación al desarrollo tradicional o al no uso de persistencia

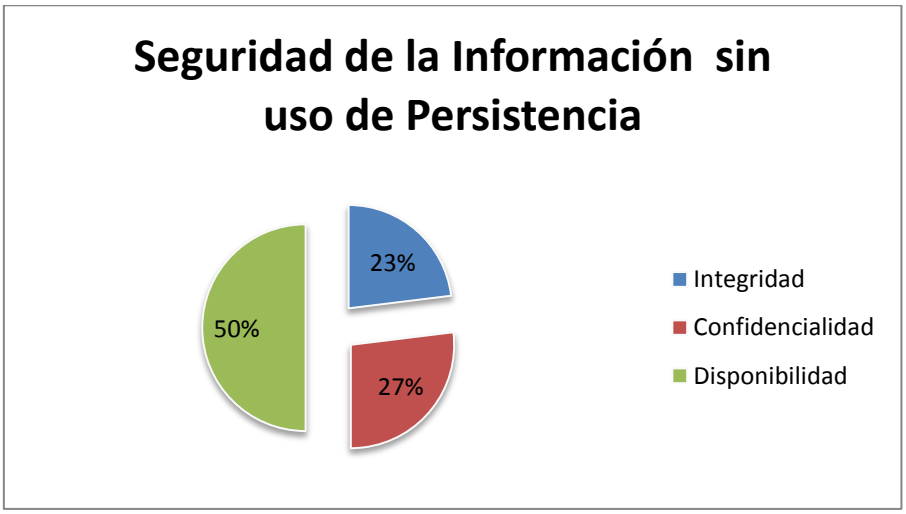


Figura IV.45. Resumen final de vulnerabilidad no uso persistencia

Análisis

En la figura se puede ver que la Integridad tiene un 23% y la confidencialidad tiene un 27%, y la disponibilidad con el 50% los mismos que representan el grado de vulnerabilidad significativo.

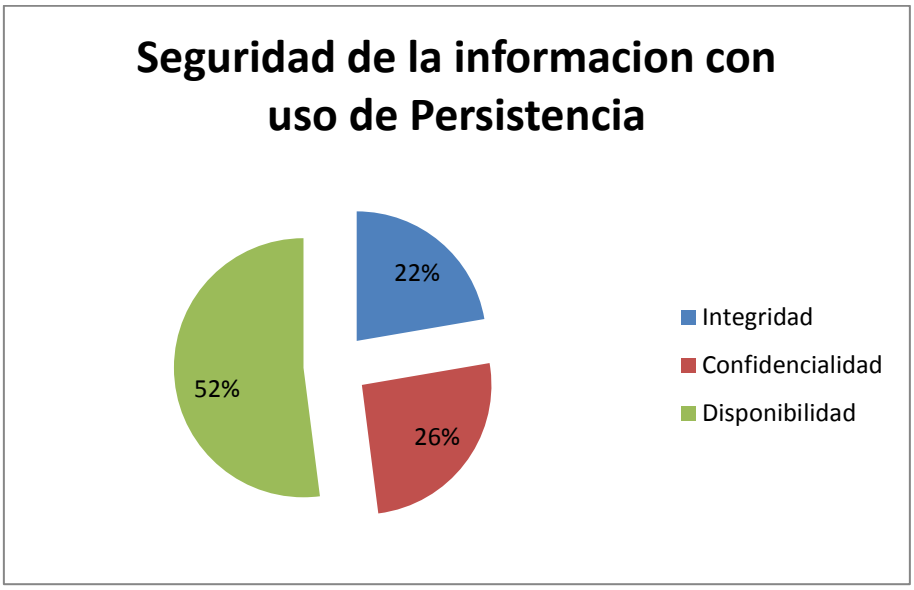


Figura IV.46. Resumen final de vulnerabilidad no uso persistencia

En la figura se puede ver que la Integridad tiene un 22% y la confidencialidad tiene un 26%, y la disponibilidad con el 52% los mismos que son menores en relación a los porcentajes obtenidos con el no uso de persistencia.

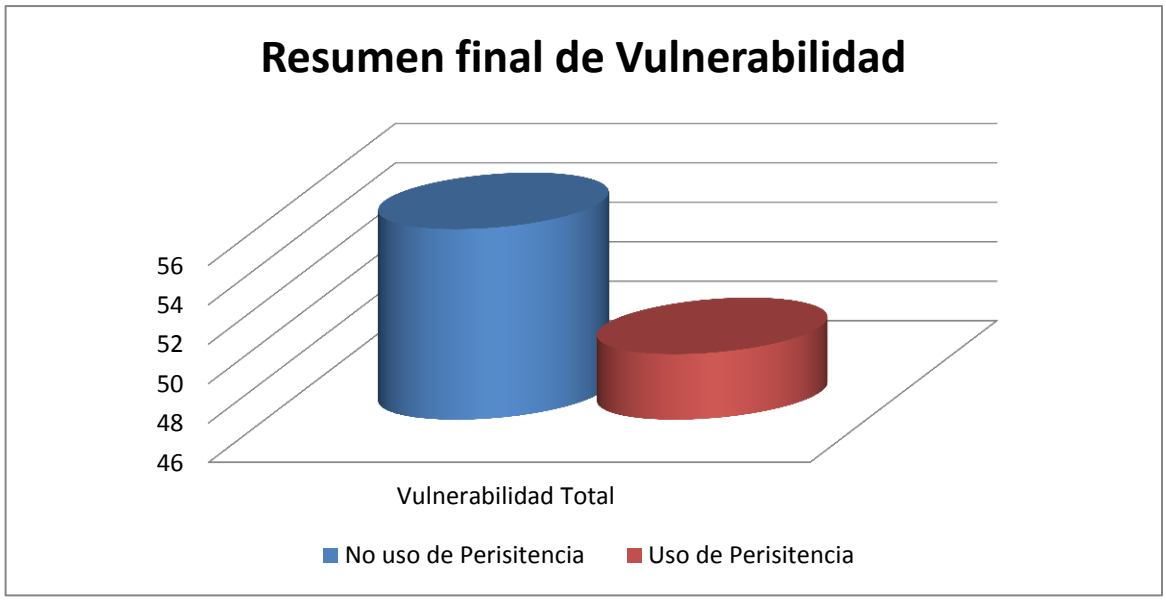


Figura IV.47. Resumen final de vulnerabilidad de seguridad de la Información

Análisis

En esta figura se puede ver notoriamente que con el uso de persistencia se tiene un porcentaje de vulnerabilidad del 49.33% mientras que con el no uso de persistencia se tiene el 55.67% por lo cual se puede decir que se reduce el nivel de vulnerabilidad a ataques externos con el uso de persistencia.

CAPITULO V

5. PRUEBAS DE LA HIPÓTESIS

5.1. TABLA DE CONTINGENCIA DE 6X2 CON LAS FRECUENCIAS OBSERVADAS

Tabla V.I. Tabla de contingencia de Frecuencias Observadas

	Framework de Persistencia TopLink			
VARIABLE DEPENDIENTE	***Parámetros***	***Uso de Persistencia***	***No Uso de Persistencia***	**Total**
Hi Bajos niveles de vulnerabilidad	Integridad	40	40	80
	Confidencialidad	30	0	30
	Disponibilidad	0	15	25
HoAltos niveles de vulnerabilidad	Integridad	0	0	0
	Confidencialidad	10	40	50
	Disponibilidad	20	5	25
Total		100	100	210

La Tabla V.II. Contiene las frecuencias esperadas, la cual constituye los valores que esperaríamos encontrar si las variables no estuvieran relacionadas. La ji cuadrada partirá del supuesto de "no relación entre las variables" y se evaluará si es cierto o no, analizando si sus frecuencias observadas son diferentes de lo que pudiera esperarse en caso de ausencia de correlación.

La frecuencia esperada de cada celda, se calcula mediante la siguiente fórmula aplicada a la tabla de frecuencias observadas.

$$fe = \frac{(total\_de\_fila)(total\_de\_columna)}{N}$$

Donde ***N*** es el número total de frecuencias observadas

Para la primera celda: *(Framework de persistencia TopLink/Bajos niveles de vulnerabilidad),* la frecuencia esperada sería:

$$fe = \frac{(80)(100)}{210} = 38.09$$

Para la Tabla V.I, la tabla de frecuencias esperadas sería la de la Tabla VII.

5.2. TABLA DE FRECUENCIAS ESPERADAS

Tabla V.II. Tabla de Frecuencias Esperadas

	Framework de Persistencia TopLink			
VARIABLE DEPENDIENTE	***Parámetros***	***Uso de Persistencia***	***No Uso de Persistencia***	**Total**
Hi Bajos niveles de vulnerabilidad	Integridad	38.09	38.09	80
	Confidencialidad	14.28	14.28	30
	Disponibilidad	11.90	11.90	25
Ho Altos niveles de vulnerabilidad	Integridad	0	0	0
	Confidencialidad	23.80	23.80	50
	Disponibilidad	11.90	11.90	25
Total		100	100	210

Una vez obtenidas las frecuencias esperadas, se aplica la siguiente fórmula de ji cuadrada:

$$\chi^2 = \sum \frac{(O-E)^2}{E}$$

Donde:

O es la frecuencia observada en cada celda

E es la frecuencia esperada en cada celda

En la Tabla3 se calcula el valor de χ^2

5.3. CÁLCULO DE χ^2

Tabla V.III. Tabla de Cálculo de χ^2

CELDA	O	E	0-E	$(O-E)^2$	$(O-E)^2/E$
Bajo/Integridad/Uso Persistencia	40	38.09	1.91	3.64	0.09
Bajo/Confidencialidad/Uso Persistencia	30	14.28	15.72	247.11	17.30
Bajo/Disponibilidad/Uso Persistencia	0	11.90	-11.90	141.61	11.90
Bajo/Integridad/No Uso Persistencia	40	38.09	1.91	3.64	0.09
Bajo/Confidencialidad/ No Uso Persistencia	0	14.28	-14.28	203.91	14.28
Bajo/Disponibilidad/No Uso Persistencia	15	11.90	3.1	9.61	0.80
Alto/Integridad/Uso Persistencia	0	0	0	0	0
Alto/Confidencialidad/ Uso Persistencia	10	23.80	-13.80	190.44	8
Alto/Disponibilidad/Uso Persistencia	20	11.90	8.1	65.61	5.51
Alto/Integridad/No Uso Persistencia	0	0	0	0	0
Alto/Confidencialidad/ No Uso Persistencia	40	23.80	16.20	262.44	11.02
Alto/Disponibilidad/No Uso Persistencia	5	11.90	-6.90	47.61	4
				χ^2	**72.99**

5.4. INTERPRETACIÓN

Para saber si el valor de χ^2 es o no significativo, se debe determinar los grados de libertad mediante la siguiente fórmula:

$$Gl = (r-1)(c-1)$$

Dónde:

r es el número de filas de la tabla de contingencia

c es el número de columnas de la tabla de contingencia

Por lo tanto:

$$Gl = (6-1)(2-1) = 5$$

De la tabla de distribución del χ^2 que se encuentra en el Anexo No. 3 y eligiendo como nivel de confianza $\alpha = 0{,}05$ se obtiene que $\chi^2 = 11.070$El valor de χ^2 calculado en esta investigación es de 72.99 que es muy superior al de la tabla de distribución, por lo que χ^2 resulta significativa y se acepta la hipótesis de investigación.

CAPITULO VI

6. IMPLEMENTACIÓN DEL SISTEMA DE EVALUACIÓN Y ACREDITACIÓN DE CARRERAS DE LA ESPOCH

6.1. Microsoft Solution Framework

6.1.1. Definición

El MSF es un modelo diseñado específicamente para crear productos de muy buena calidad, donde para cumplir este objetivo prima la comunicación tanto entre el equipo de desarrollo como entre ellos y los clientes.

6.1.2. Fases

El MSF está compuesto por las siguientes fases:

1. Visión
2. Planeación
3. Desarrollo
4. Estabilización
5. Instalación
6. Soporte

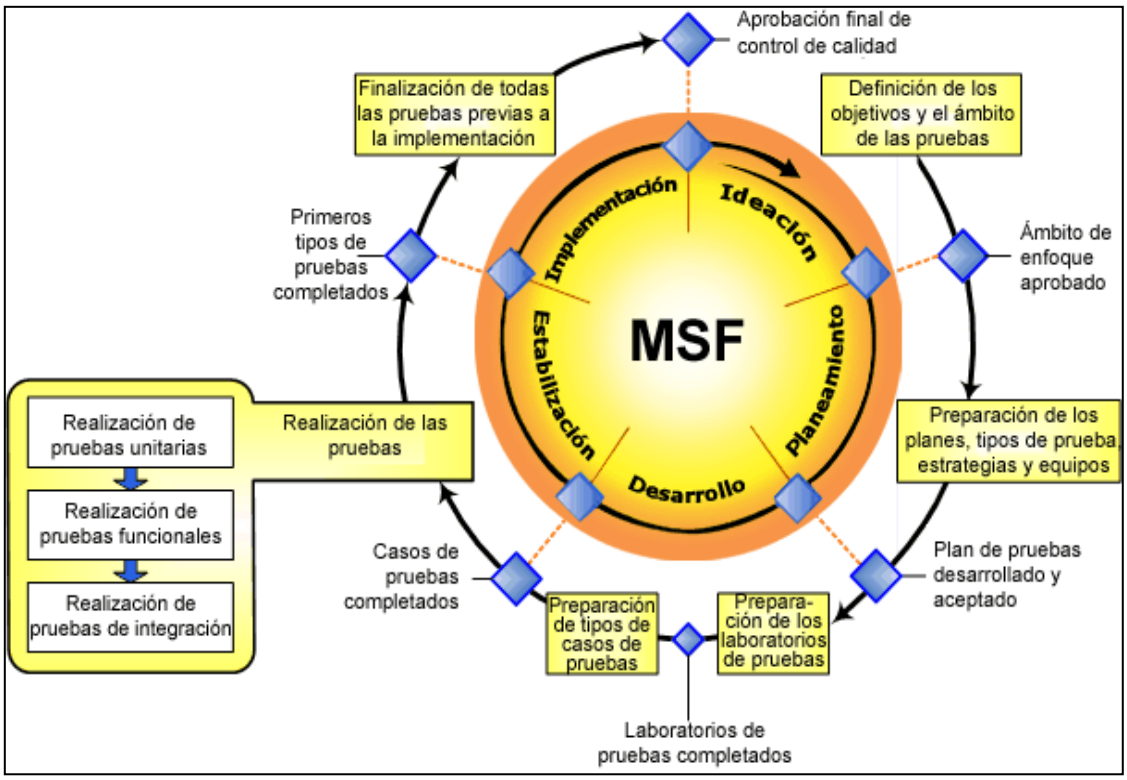


Figura VI.48. Microsoft Solution Framework

6.1.3. Ventajas

La ventaja principal ventaja es que al ser un modelo desarrollado por Microsoft se puede tener mayor soporte y mantenimiento, además la mayoría de los usuarios finales están más acostumbrados con este producto. Además sirve para grandes y pequeños proyectos.

Cabe recalcar que MSF no se parece al RUP en algunas definiciones (principalmente en la cuestión de los cambios).

- Incentiva al trabajo en equipo y a la colaboración.
- Es útil para proyectos de pequeña y gran escala.
- Crea una disciplina de análisis de riesgos que ayuda y evoluciona con el proyecto.
- Cuenta con plantillas que nos ayuda para el proceso de documentación.

6.1.4. Desventajas

La principal desventaja es que se torna un trabajo bastante largo, ya que para cada fase se debe documentar profundamente todo lo que se haga, pero no deja de ser un modelo que tiene buenos resultados.

6.2. Fase de Visión

Para el desarrollo eficiente del proyecto es importante obtener una visión del proyecto compartida, comunicada, extendida y alineada con los objetivos del negocio. Además identificar los beneficios, requerimientos funcionales, sus alcances y restricciones; y los riesgos inherentes al proceso

6.2.1. Definición del Problema

Al pasar los años la Escuela Superior Politécnica de Chimborazo a demostrado excelencia Académica y busca la calidad de la Educación Superior a través de la acreditación de carreras, para lo cual se requiere Evaluaciones de las carreras con el fin de ser acreditadas. Por tal motivo se requiere implementar un sistema Informático que permita automatizar el Proceso de evaluación, el cual proporcionará resultados exactos que ayudarán a la toma de decisiones para mejorar la Carreras Evaluadas y alcanzar el objetivo propuesto.

6.2.2. Visión del Proyecto

Desarrollo de un sistema que permita la evaluación de las carreras existentes en la ESPOCH con fines de acreditación.

Para proceder a evaluar cada carrera primero deberán tener asignadas carreras a las cuales evaluar, además se les otorgara una cuenta de usuario y su clave.

Los datos de los usurarios así como los resultados de la evaluación serán almacenados en la base de Datos Oracle 10g XE.

- **Módulo de Administración**

Este módulo consta de varios módulos y permitirá llevar un control de los datos necesarios para el sistema y está dividido en:

- Módulo Indicador
- Módulo Unidad Académica
- Módulo Cálculo Indicador
- Módulo GestíonResponsable
- Módulo Gestíon Responsable - Carrera
- Módulo Evaluar Indicador
- Módulo Responsable
- Módulo Responsable - carrera
- Módulo Hoja de Ruta y Evidencia
- Módulo Reportes

  Nota: Dentro de cada módulo se encuentra la opción de reportes.

- **Módulo de Seguridades**

Este módulo nos servirá para controlar la autenticación de nuestros clientes a la hora de utilizar nuestro Sistema.

- Módulo Roles
- Módulo Usuario
- Permisos
- Claves

6.3. Perfiles de Usuario

Al realizar un análisis acerca de los posibles usuarios del sistema se ha determinado los siguientes: El usuario SuperAdministrador (Una persona delegada de la comisión de Evaluación), el usuario Administrador (Decanos, Directores de Escuela, vicedecanos, personas delegadas por la Comisión de evaluación), el Usuario Normal (Secretarias, Personas delegadas por la comisión de evaluación)

Tabla VI.I. Perfiles de Usuario

No.	Nombre	Perfil	Tipo de Acceso	Descripción
1	SuperAdministrador	Usuario SuperAdministrador	Web	➢ Es el encargado de administrar el sistema totalmente. Tiene acceso sobre todas las facultades y carreras sin acepción.
2	Administrador	Usuario Administrador	Web	➢ Es el encargado de administrar el sistema de evaluar carreras pero solo de las que le hayan sido asignadas. ➢ Administración de reportes.
3	Normal	Usuario Normal	Web	➢ Posee ciertos permisos sobre las carreras que le corresponden no poseen control total. ➢ Acceso a varios reportes.

6.4. Ámbito del Proyecto

Desarrollo de una aplicación web para la Gestión y Administración de Evaluaciones de Carreras el mismo que presta el servicio Evaluaciones automáticas de Indicadores.

Este sistema permitirá la Gestión y Administración de los módulos incorporados que permiten realizar la configuración de una Evaluación.

6.4.1. Requerimientos Funcionales

Módulo indicador

Req1. El sistema deberá permitir gestionar datos de Criterios.

Req2. El sistema deberá permitir gestionar datos de Grupos.

Req3. El sistema deberá permitir gestionar datos de Subgrupos

Req4. El sistema deberá permitir gestionar datos de Indicador

Módulo Unidad Académica

Req5. El sistema deberá permitir gestionar datos de Facultad

Req6. El sistema deberá permitir gestionar datos de Escuela

Req7. El sistema deberá permitir gestionar datos de Carrera

Módulo Calculo Indicador

Req8. El sistema deberá permitir gestionar datos de las fórmulas.

Módulo Responsable

Req9. El sistema deberá permitir gestionar datos del Responsable.

Módulo Responsable – carrera

Req10. El sistema deberá permitir gestionar datos de Responsable - Carrera.

Módulo Evaluación Indicador

Req11. El sistema deberá permitir gestionar datos de Indicador Evaluado.

MóduloHoja de Ruta y Verificación

Req12. El sistema deberá permitir gestionar datos de Hoja de Ruta.

Req13. El sistema deberá permitir gestionar datos de Evidencia.

Módulo Seguridades

Req14. El sistema deberá permitir gestionar datos de Roles.

Req15. El sistema deberá permitir gestionar datos de los Usuarios.

Req16. El sistema deberá permitir gestionar datos de Permisos.

Req17. El sistema deberá permitir gestionar las claves.

Módulo de Reportes

Req 18. El sistema debe generar reportes.

- Listado de Facultades.
- Listado de Escuelas.
- Listado de Carreras.
- Listado de Responsables.
- Listado de Responsables por carrera.
- Listado de Criterios.
- Listado de Grupos.
- Listado de Subgrupos.
- Listado de Indicadores.
- Listado de Indicadores por Criterio.
- Listado de Fórmulas de Evaluación.
- Listado de Indicadores Evaluados por Carrera.
- Listado de Ruta de verificación de Indicador por Carrera.
- Listado de Evidencia de Indicador por Carrera.

6.4.2. Requerimientos No Funcionales

A continuación se muestran los requerimientos no funcionales

- Rendimiento
- Disponibilidad
- Seguridad
- Portabilidad
- Mantenibilidad
- Escalabilidad
- Reusabilidad
- Interfaces
- Usabilidad

6.5. Objetivos del Proyecto

6.5.1. Objetivos del Negocio

- Registrar a los diferentes Usuarios, registrando su cuenta.
- Llevar un control transparente de la evaluación de cada carrera
- Limitar el acceso autorizado solo a las personas autorizadas.
- Generación de reportes.

6.5.2. Objetivos del Diseño

- Garantizar un acceso rápido a la aplicación.
- Proteger contra el acceso de intrusos mediante la validación de cuentas.
- Proporcionar una Interfaz web amigable y de fácil manejo.

6.6. Riesgos

El riesgo implica cambios que pueden darse por cambios de opinión, de acciones, de lugares, es inevitable e implica incertidumbre y pérdida cuando el riesgo se ha convertido en problema.

Debemos analizar qué riesgos podrían hacer que nuestro proyecto fracasara, y se debe escoger adecuadamente que acciones pueden convertirse en riesgos para de esta manera lograr superarlos.

El propósito esencial de este proceso de análisis de riesgos va enfocado a prevenir muchos acontecimientos que pueden dificultar el desarrollo del "Sistema de Evaluación de Carreras de la ESPOCH".

Por lo cual debemos darle una gran importancia a este análisis tomando en cuenta todos los tipos de riesgos que se puede encontrar en el desarrollo del sistema al mismo tiempo gestionar aquellos riesgos con gran probabilidad de impacto.

6.6.1. Identificación del Riesgo

La identificación del riesgo es un intento sistemático para especificar las amenazas al plan de proyecto (estimaciones, planificación temporal, carga de recursos, etc.). Identificando los riesgos conocidos y predecibles, el gestor del proyecto da un paso adelante para evitarlos cuando sea posible y controlarlos cuando sea necesario.

Existen tres tipos de riesgo:

- Riesgo del proyecto.
- Riesgo técnico.
- Riesgo del negocio.

Tabla VI.II. Identificación de Riesgos

ID	DESCRIPCIÓN DEL RIESGO	CATEGORIA	CONSECUENCIA
R1	Los usuarios no definieron correctamente los requerimientos.	Proyecto	Retraso del proyecto, costo del proyecto.
R2	Cambios continuos de los requerimientos por parte del usuario	Negocio	Pérdida de recursos económicos.
R3	Falta de conocimiento por parte de los programadores del lenguaje de programación y desarrollo de software	Técnico	Amenazan la calidad Capacitación a los programadores Retraso del proyecto
R4	Costo final del proyecto sea exagerado	Proyecto	Retraso del proyecto, Costo del proyecto.
R5	Interfaces inadecuadas para validar las operaciones en el sistema.	Técnico	Amenazan la calidad del software y la implementación puede llegar a ser difícil.
R6	Mala relación de los miembros del equipo de desarrollo.	Proyecto	Retraso en el proyecto
R7	La información que se modifique no	Negocio	Provocan inconsistencia

	se actualice en la base de datos, del Sistema		en los datos

6.6.2. Análisis de Riesgos

Tabla VI.III. Valoración de riesgos

Rango de probabilidad	Descripción	Valor
1% - 33%	Baja	1
34% - 67%	Media	2
68% - 99%	Alta	3

Tabla VI.IV. Probabilidad

Identificación	Probabilidad		
	%	Valor	Probabilidad
R1	20	1	BAJA
R2	20	1	BAJA
R3	30	1	BAJA
R4	40	2	MEDIA
R5	20	1	BAJA
R6	50	1	MEDIA
R7	60	2	MEDIA

Determinación del Impacto

Tabla VI.V. Impacto del Riesgo

Impacto	Retraso	Impacto técnico	Costo	valor
Bajo	1 semana	Ligero efecto en el desarrollo del proyecto	< 1%	1
Moderado	2 semanas	Moderado efecto en el desarrollo del proyecto	< 5%	2
Alto	1 mes	Severo efecto en el desarrollo del proyecto	< 10%	3
Crítico	> 1 meses	Proyecto no puede ser culminado	> 20%	4

Tabla VI.VI. Riesgo - Impacto

Identificación	Impacto	
	Valor	Impacto
R1	3	ALTO
R2	3	ALTO
R3	1	BAJO
R4	3	ALTO
R5	1	BAJO
R6	2	MEDIA
R7	3	MEDIA

Determinación de la exposición al riesgo

Exposición al riesgo	Valor	color
Baja	1 o 2	
Media	3 o 4	
Alta	> 6	

Tabla VI.VII. Impacto – Probabilidad

Impacto / Probabilidad	Bajo =1	Moderado =2	Alto =3	Crítico =4
Alta = 3	3	6	9	12
Media = 2	2	4	6	8
Baja = 2	2	4	6	8

Tabla VI.VIII. Tabla total Riesgo

Identificación	Probabilidad			Impacto		Exposición al riesgo	
	%	Valor	Probabilidad	Valor	Impacto	Valor	Exposición
R1	20	1	BAJA	3	ALTO	3	MEDIA
R2	20	1	BAJA	3	ALTO	3	MEDIA
R3	30	1	BAJA	1	BAJO	1	BAJA
R4	40	2	MEDIA	3	ALTO	6	ALTA
R5	20	1	BAJA	1	BAJO	1	BAJA
R6	50	1	MEDIA	2	MODERADO	2	BAJA
R7	60	2	MEDIA	3	ALTO	6	MEDIA

Determinación de la prioridad del riesgo

Tabla VI.IX. Prioridades del riesgo

Identificación	Prioridad	Exposición
R4	1	6
R1	2	3
R2	2	3
R7	2	3
R6	3	2
R3	4	1
R5	4	1

6.6.3. Planeación y programación del Riesgo

Para mitigar el riesgo se utiliza la Hoja de Gestión de Riesgo

Tabla VI.X. Gestión del Riesgo 1

HOJA DE GESTIÓN DEL RIESGO			
ID DEL RIESGO: R1		**FECHA:**	
Probabilidad: Baja **Valor:** 1	**Impacto**: Alto **Valor:** 3	**Exposición:** Alta **Valor: 3**	**Prioridad:** 1
DESCRIPCIÓN: Los usuarios no definieron correctamente los requerimientos.			
REFINAMIENTO: **Causas:** ➢ No existió una comunicación adecuada entre el usuario y el responsable del proyecto ➢ El usuario no explicó correctamente las necesidades que posee. **Consecuencias:** ➢ Retraso del proyecto ➢ Incremento en el costo del proyecto			
REDUCCIÓN: ➢ Analizar las necesidades del usuario para establecer correctamente los requerimientos. ➢ Que exista una adecuada comunicación entre el cliente y el programador			
SUPERVISIÓN: ➢ Ponerse de acuerdo al inicio: el cliente y el responsable acerca de sus necesidades. ➢ Que el ambiente de comunicación sea el más propicio entre el cliente y el responsable del proyecto.			
GESTIÓN: ➢ Que una vez conocidas las necesidades del cliente se deberán poner de acuerdo el responsable del proyecto y cliente para establecer los requerimientos.			
ESTADO ACTUAL: Fase de reducción iniciada: Fase de supervisión iniciada: Gestionando el riesgo: 			
RESPONSABLES: Walter Maiquiza Jenny Naula			

Tabla VI.XI. Gestión del Riesgo 2

<table>
<tr><th colspan="4">HOJA DE GESTIÓN DEL RIESGO</th></tr>
<tr><td colspan="2">ID DEL RIESGO: R2</td><td colspan="2">FECHA:</td></tr>
<tr><td>Probabilidad: Baja
Valor: 1</td><td>Impacto: Alta
Valor: 3</td><td>Exposición: Moderado
Valor: 3</td><td>Prioridad: 1</td></tr>
<tr><td colspan="4">DESCRIPCIÓN: Cambios continuos de los requerimientos por parte del usuario</td></tr>
<tr><td colspan="4">REFINAMIENTO:

Causas:
➢ Falta de análisis de requerimientos por parte del usuario y desarrollador

Consecuencias:
➢ No cumplir con las expectativas del Sistema.
➢ Pérdida de recursos.</td></tr>
<tr><td colspan="4">REDUCCIÓN:

➢ Asignación de recursos para una correcta captación de los requerimientos</td></tr>
<tr><td colspan="4">SUPERVISIÓN:

➢ Seguimiento de las funcionalidades de los requerimientos planteados</td></tr>
<tr><td colspan="4">GESTIÓN:

➢ Desplegar los beneficios de un requerimiento gestionado.</td></tr>
<tr><td colspan="4">ESTADO ACTUAL:

Fase de reducción iniciada: ☑
Fase de supervisión iniciada: ☐
Gestionando el riesgo: ☐</td></tr>
<tr><td colspan="4">RESPONSABLE:
Walter Maiquiza
Jenny Naula</td></tr>
</table>

Tabla VI.XII. Gestión del riesgo 3

HOJA DE GESTIÓN DEL RIESGO			
ID DEL RIESGO: R3		**FECHA:**	
Probabilidad: Baja **Valor:** 1	**Impacto**: Bajo **Valor:** 1	**Exposición:** Baja **Valor:** 1	**Prioridad: 4**
DESCRIPCIÓN: Falta de conocimiento por parte de los programadores del lenguaje de programación y desarrollo de software			
REFINAMIENTO: **Causas:** ➢ Especialización en diferentes lenguajes de programación y desarrollo del software al que se utilizará para la elaboración del nuevo sistema ➢ Falta de investigación acerca del lenguaje de programación que se usará. **Consecuencias:** ➢ Amenazan la calidad ➢ Capacitación a los programadores ➢ Retraso del proyecto			
REDUCCIÓN: ➢ Capacitar a los programadores en el lenguaje de programación y desarrollo de software que se usará ➢ Promover la investigación de nuevos lenguajes de programación			
SUPERVISIÓN: ➢ Controlar que la capacitación a los programadores sea la adecuada			
GESTIÓN: ➢ Fomentar la especialización el lenguajes de programación			
ESTADO ACTUAL: Fase de reducción iniciada: Fase de supervisión iniciada: Gestionando el riesgo: 			
RESPONSABLE: Walter Maiquiza Jenny Naula			

Tabla VI.XIII. Gestión del riesgo 4

HOJA DE GESTIÓN DEL RIESGO			
ID DEL RIESGO: R4		**FECHA:**	
Probabilidad: Media **Valor:** 2	**Impacto**: Alto **Valor:** 3	**Exposición:** Alta **Valor:** 6	**Prioridad:** 1
DESCRIPCIÓN: Costo final del proyecto sea exagerado			
REFINAMIENTO: **Causas:** ➢ No se planifico de manera adecuada el presupuesto para el desarrollo del sistema. ➢ La culminación de cada etapa no fue a tiempo. **Consecuencias:** ➢ Retraso en la planificación. ➢ Aumento en el costo final del proyecto			
REDUCCIÓN: ➢ Culminar las etapas asignadas para cada miembro del grupo puntualmente. ➢ Realizar correctamente la planificación del presupuesto del proyecto			
SUPERVISIÓN: ➢ Revisar que la planificación presupuestaria sea adecuada para evitar inconvenientes. ➢ Verificar el trabajo del equipo de desarrollo para la culminación de cada etapa sea eficientemente.			
GESTIÓN: ➢ Exigir al equipo de trabajo que todas las etapas sean culminadas puntualmente.			
ESTADO ACTUAL: Fase de reducción iniciada: ☑ Fase de supervisión iniciada: ☐ Gestionando el riesgo: ☐			
RESPONSABLE: Walter Maiquiza Jenny Naula			

Tabla VI.XIV. Gestión del riesgo 5

HOJA DE GESTIÓN DEL RIESGO			
ID DEL RIESGO: R5		**FECHA:**	
Probabilidad: Baja **Valor:** 1	**Impacto**: Bajo **Valor:** 1	**Exposición:** Baja **Valor:** 2	**Prioridad:** 4
DESCRIPCIÓN: Interfaces inadecuadas para validar las operaciones en el sistema.			
REFINAMIENTO: **Causas:** ➢ El diseñador no elabore las interfaces con las datos necesarios para que el usuario se valide realice las operaciones básicas. **Consecuencias:** ➢ Amenazan la calidad del software y la implementación puede llegar a ser difícil.			
REDUCCIÓN: ➢ Diseñar correctamente las interfaces para la validar el acceso al sistema de Gestión			
SUPERVISIÓN: ➢ Revisar que el diseño de las interfaces sea el más óptimo para la Gestión			
GESTIÓN: ➢ Exigir al equipo de trabajo que las interfaces sean aprobadas unánimemente para garantizar la validación.			
ESTADO ACTUAL: Fase de reducción iniciada: ☑ Fase de supervisión iniciada: ☐ Gestionando el riesgo: ☐			
RESPONSABLE: Walter Maiquiza Jenny Naula			

Tabla VI.XV. Gestión del riesgo 6

HOJA DE GESTIÓN DEL RIESGO			
ID DEL RIESGO: R6		**FECHA:**	
Probabilidad: Baja **Valor:** 1	**Impacto**: Moderado **Valor:** 2	**Exposición:** Baja **Valor:** 2	**Prioridad:** 3
DESCRIPCIÓN: Mala relación de los miembros del equipo de desarrollo.			
REFINAMIENTO: **Causas:** ➢ Ambiente de trabajo tenso ➢ No exista una compenetración del equipo de trabajo **Consecuencias:** ➢ Retraso en el proyecto.			
REDUCCIÓN: ➢ Fomentar la unión para desarrollar eficientemente el proyecto			
SUPERVISIÓN: ➢ Actitud de los miembros del proyecto sea positiva ➢ La unión del equipo sea fuerte para garantizar el desarrollo del proyecto.			
GESTIÓN: ➢ Los integrantes se familiaricen rápidamente ➢ El gestor del proyecto converse con los miembros del proyecto para mantener la unión y la armonía entre el grupo			
ESTADO ACTUAL: Fase de reducción iniciada: ☑ Fase de supervisión iniciada: ☐ Gestionando el riesgo: ☐			
RESPONSABLE: Walter Maiquiza Jenny Naula			

Tabla VI.XVI. Gestión del riesgo 7

<table>
<tr><th colspan="4">HOJA DE GESTIÓN DEL RIESGO</th></tr>
<tr><td colspan="2">ID DEL RIESGO: R8</td><td colspan="2">FECHA:</td></tr>
<tr><td>Probabilidad: Media
Valor: 2</td><td>Impacto: Alto
Valor: 3</td><td>Exposición: Alta
Valor: 6</td><td>Prioridad: 1</td></tr>
<tr><td colspan="4">DESCRIPCIÓN: La información que se modifique no se actualice en la base de datos, del Sistema</td></tr>
<tr><td colspan="4">REFINAMIENTO:
Causas:
➢ No se cuenta con un método adecuado para facilitar la actualización automática.

Consecuencias:
➢ Provocan inconsistencia en los datos</td></tr>
<tr><td colspan="4">REDUCCIÓN:

➢ Implementar un método para la actualización de los datos cuando se haga una modificación de los mismos.</td></tr>
<tr><td colspan="4">SUPERVISIÓN:

➢ Revisar que el método implementado cumpla a cabalidad la tarea para el cual fue creado.</td></tr>
<tr><td colspan="4">GESTIÓN:
➢ Solicitar al equipo de trabajo que el método implementado sea el adecuado</td></tr>
<tr><td colspan="4">ESTADO ACTUAL:

Fase de reducción iniciada: ☑
Fase de supervisión iniciada: ☐
Gestionando el riesgo: ☐</td></tr>
<tr><td colspan="4">RESPONSABLE:
Walter Maiquiza
Jenny Naula</td></tr>
</table>

6.7. Planificación Inicial

6.7.1. Factibilidad

Factibilidad técnica

La factibilidad técnica ayuda a determinar si la propuesta puede ser implementada con el hardware, software y recurso humano disponible.

Para el desarrollo de la aplicación web SEACE se cuenta con casi todos los recursos hardware y software necesarios. A continuación se detalla el hardware, software existente, requerido así como también el personal técnico requerido para el desarrollo del mismo.

Hardware Existente

Hardware con el que se cuenta para el desarrollo de la aplicación es el siguiente:

Tabla VI.XVII Hardware existente

Cantidad	Descripción	Observaciones
2	Computadores	Desarrollo de la aplicación y documentación
1	Infraestructura de Red	Acceder al internet para consultar las dudas en el desarrollo de la aplicación y realizar las pruebas respectivas.

Hardware Requerido

Tabla VI.XVIII. Hardware requerido

Cantidad	Descripción	Observaciones
1	Impresora	Imprimir Informes
1	Setrvidor	

Software Existente

El Software que se necesita para el desarrollo del sistema es el siguiente:

Tabla VI.XIX Software Existente

Nombre	Descripción
XP profesional	Sistema Operativo
Enterprise Architect	Herramienta de diseño UML

Software Requerido

Tabla VI.XX. Software Requerido

Nombre	Descripción
NetBeans IDE 6.9.1.	Entorno de desarrollo
iReport	Para generar reportes
Apache	Servidor web de distribución libre.
Netsparker Community	Analiza Vulnerabilidades existentes en un sistema web
Practiline Source	Cuenta las líneas de Còdigo

Recurso Humano Requerido

Tabla VI.XXI. Recurso Humano Requerido

Función	Formación
Jefe de Proyecto	Ingeniería en Sistemas
Equipo de dessarrolladores	Estudiante de Ingeniería en Sistemas
Administrador de Base de Datos	Ingeniería en Sistemas
Diseñador	Estudiante de Ingeniería en Sistemas

Factibilidad Operativa

Recurso Humano

El recurso humano que participará en la operación del sistema son:

- **Usuarios Directos**

  Los usuarios directos a capacitar para el manejo del sistema son:

  Personal a capacitar

Tabla VI.XXII. Personal a Capacitar

Nombre	Función
U. Superadministrador	Persona delegada por la Comisión
U. Administrador	Director de la Escuela, Decanos, Vicedecanos
U. Normales	Secretarias, y personal delegado por la Comisión

Factibilidad Económica

El tiempo de duración del proyecto será de 8 meses.

Costos

- **Costos de desarrollo**

Costos Personal

Mensual Total

Jefe de Proyecto y Desarrollador	$800	$6400.00
Administrador de BD, Diseñador	$800	$6400.00
Costo Personal Total		**$12800.00**

- **Costo de hardware y software**

Costo Software

Internet	**$400,00**

- **Costos varios**

Suministros	$300,00
Alimentacion	$500.00
Papel A4	$10.00
Costo Personal Total	**$810.00**

- ***Costos Total de desarrollo*** **$14010,00**

Análisis costo-beneficio

Los beneficios que se podrá obtener con la utilización de este sistema son los siguientes:

Permitirá realizar la evaluación de las carreras en la ESPOCH de una manera rápida, ya que este proceso suele demandar de tiempo y recursos, mediante este sistema se podrán evaluar las carreras de manera sincronizada y con resultados exactos.

Se podrá realizar un mayor control de las carreras cuando estas estén fallando en el cumplimiento de criterios que son importantes para el avance de la Educación Superior.

Mediante el sistema se puede generar reportes que ayudarán a la toma de decisiones de las autoridades.

Se optimizará el tiempo en la elaboración de informes.

La utilización del sistema será fácil de utilizar y además los usuarios pueden ingresar desde cualquier lugar de donde se encuentren.

6.8. Fase de Planeación

Obtener un cronograma de trabajo que cumpla con lo especificado en la fase de visión, desarrollar los requerimientos funcionales y no funcionales, describir el escenario, diagramar casos de uso.

PLANEACIÓN

En esta fase se realiza la preparación de la especificación funcional, diseño conceptual.

Especificación Funcional

6.8.1. Diseño Conceptual

6.8.1.1. Requerimientos Funcionales

MODULO INDICADORES

6.8.1.1.1. Requerimiento Funcional 1, 2, 3, 4

6.8.1.1.1.1. Especificaciones

6.8.1.1.1.1.1. Introducción

El sistema deberá permitir gestionar datos de Criterios, Grupos, Subgrupos, Indicador.

6.8.1.1.1.1.2. Entrada

Fuentes de Entrada

Usuario

Contraseña

Rol

Frecuencia

Bajo demanda

Requisitos de control

Controla que los campos del formulario no estén vacíos u además que los datos sean correctos

Entradas válidas

Todos los campos sean válidos.

6.8.1.1.1.1.3. Procesos

1. El usuario ingresará Nombre de usuario y contraseña de autenticación.
2. El sistema validará la contraseña.

3. Si el dato es verdadero

3.1 Ingresará al sistema

Caso contrario

3.2 Mensaje de error

4. Ingreso, modificar, eliminar

5. Aceptar

6. Validación de los datos.

7. Si todos los campos están llenos y los datos son correctos

7.1 Actualización de la BD y mensaje satisfactorio.

Caso contrario

7.2 Mensaje de error

6.8.1.1.1.1.4. Salidas

Destino de las salidas

Muestra un listado nuevo con el dato ingresado cuando los datos ingresados son correctos o sin el dato elegido.

6.8.1.1.1.2. Interfaces de Hardware

- El monitor es el principal medio hardware de visualización, el cual se lo utilizará para mostrar cada uno de los procesos que se efectuarán.

6.8.1.1.1.3. Interfaces de software

- La herramienta de desarrollo que se utilizará es NetBeans IDE 6.9.1 para realizar cada una de las aplicaciones que sean necesarios para este requerimiento.
- La Base de Datos está implementada en Oracle 10G Express Edition.

6.8.1.1.1.4. Interfaces de Comunicación

El Sistema utilizará el protocolo TCP/IP para establecer la comunicación.

MÓDULO UNIDAD ACADÉMICA

6.8.1.1.2. Requerimiento Funcional 5, 6, 7

6.8.1.1.2.1. Especificaciones

6.8.1.1.2.1.1. Introducción

El sistema deberá permitir gestionar datos de Facultad, Escuela, Carrera.

6.8.1.1.2.1.2. Entrada

Fuentes de Entrada

Usuario

Contraseña

Rol

Frecuencia

Bajo demanda

Requisitos de control

Controla que los campos del formulario no estén vacíos u además que los datos sean correctos

Entradas válidas

Todos los campos sean válidos.

6.8.1.1.2.1.3. Procesos

1. El usuario ingresará Nombre de usuario y contraseña de autenticación.
2. El sistema validará la contraseña.
3. Si el dato es verdadero

   3.3 Ingresará al sistema

   Caso contrario

   3.4 Mensaje de error
4. Elije el Módulo Unidad Académica
5. Administrar
6. El usuario escoge si desea Ingresar, modificar o eliminar
7. Validación de los datos.
8. Si todos los campos están llenos y los datos son correctos

   8.1.Actualización de la BD y mensaje satisfactorio.

   Caso contrario

   8.2.Mensaje de error

8.2.1.1.1.1.1. Salidas

Destino de las salidas

Muestra un listado nuevo con el dato ingresado cuando los datos ingresados son correctos o sin el dato elegido.

8.2.1.1.1.2.Interfaces de Hardware

- El monitor es el principal medio hardware de visualización, el cual se lo utilizará para mostrar cada uno de los procesos que se efectuarán.

8.2.1.1.1.3.Interfaces de software

- La herramienta de desarrollo que se utilizará es NetBeans IDE 6.9.1 para realizar cada una de las aplicaciones que sean necesarios para este requerimiento.

- La Base de Datos está implementada en Oracle 10G Express Edition.

8.2.1.1.1.4.Interfaces de Comunicación

El Sistema utilizará el protocolo TCP/IP para establecer la comunicación.

MODULO CÁLCULO INDICADOR

6.8.1.1.3. Requerimiento Funcional 8

6.8.1.1.3.1. Especificaciones

6.8.1.1.3.1.1. Introducción

El sistema deberá permitir gestionar los datos las Fórmulas.

6.8.1.1.3.1.2. Entrada

Fuentes de Entrada

Usuario

Contraseña

Rol

Frecuencia

Bajo demanda

Requisitos de control

Controla que los campos del formulario no estén vacíos y además que los datos sean correctos

Entradas válidas

Todos los campos sean válidos.

6.8.1.1.3.1.3. Procesos

1. El usuario ingresará Nombre de usuario y contraseña de autenticación.
2. El sistema validará la contraseña.
3. Si el dato es verdadero
    a. Ingresará al sistema

   Caso contrario
    b. Mensaje de error
4. Elije el Módulo Cálculo Indicador
5. Administrar
6. El usuario escoge si desea Ingresar, modificar o eliminar
7. Validación de los datos.
8. Si todos los campos están llenos y los datos son correctos
    a. Actualización de la BD y mensaje satisfactorio.

   Caso contrario
    b. Mensaje de error

6.8.1.1.3.1.4. Salidas

Destino de las salidas

Se vuelve a cargar la página cuando los datos ingresados son correctos.

6.8.1.1.3.2. Interfaces de Hardware

- El monitor es el principal medio hardware de visualización, el cual se lo utilizará para mostrar cada uno de los procesos que se efectuarán.

6.8.1.1.3.3. Interfaces de software

- La herramienta de desarrollo que se utilizará es NetBeans IDE 6.9.1 para realizar cada una de las aplicaciones que sean necesarios para este requerimiento.

- La Base de Datos está implementada en Oracle 10G Express Edition.

6.8.1.1.3.4. Interfaces de Comunicación

El Sistema utilizará el protocolo TCP/IP para establecer la comunicación.

MODULO RESPONSABLE

6.8.1.1.4. Requerimiento Funcional 9

6.8.1.1.4.1. Especificaciones

6.8.1.1.4.1.1. Introducción

El sistema deberá permitir gestionar datos del Responsable.

6.8.1.1.4.1.2. Entrada

Fuentes de Entrada

Usuario

Contraseña

Rol

Frecuencia

Bajo demanda

Requisitos de control

Controla que los campos del formulario no estén vacíos u además que los datos sean correctos

Entradas válidas

Todos los campos sean válidos.

6.8.1.1.4.1.3. Procesos

1. El usuario ingresará Nombre de usuario y contraseña de autenticación.
2. El sistema validará la contraseña.
3. Si el dato es verdadero

   3.1 Ingresará al sistema

   Caso contrario

   3.2 Mensaje de error
4. Elije el Módulo Responsable
5. Administrar
6. El usuario escoge si desea Ingresar, modificar o eliminar
7. Validación de los datos.
8. Si todos los campos están llenos y los datos son correctos

   8.1. Actualización de la BD y mensaje satisfactorio.

   Caso contrario

   8.2 Mensaje de error

6.8.1.1.4.1.4. Salidas

Destino de las salidas

Se vuelve a cargar la página cuando los datos ingresados son correctos.

6.8.1.1.4.2. Interfaces de Hardware

- El monitor es el principal medio hardware de visualización, el cual se lo utilizará para mostrar cada uno de los procesos que se efectuarán.

6.8.1.1.4.3. Interfaces de software

- La herramienta de desarrollo que se utilizará es NetBeans IDE 6.9.1 para realizar cada una de las aplicaciones que sean necesarios para este requerimiento.

- La Base de Datos está implementada en Oracle 10G Express Edition.

6.8.1.1.4.4. Interfaces de Comunicación

El Sistema utilizará el protocolo TCP/IP para establecer la comunicación.

MÓDULO RESPONSABLE - CARRERA

6.8.1.1.5. Requerimiento Funcional 10

6.8.1.1.5.1. Especificaciones

6.8.1.1.5.1.1. Introducción

El sistema deberá permitir gestionar datos del Responsable.

6.8.1.1.5.1.2. Entrada

Fuentes de Entrada

Usuario

Contraseña

Rol

Frecuencia

Bajo demanda

Requisitos de control

Controla que los campos del formulario no estén vacíos u además que los datos sean correctos

Entradas válidas

Todos los campos sean válidos.

6.8.1.1.5.1.3. Procesos

1. El usuario ingresará Nombre de usuario y contraseña de autenticación.

2. El sistema validará la contraseña.

3. Si el dato es verdadero

3.5 Ingresará al sistema

Caso contrario

3.6 Mensaje de error

4. Elije el Módulo Responsable - Carrera

5. Administrar

6. El usuario escoge si desea Ingresar, modificar o eliminar

7. Validación de los datos.

8. Si todos los campos están llenos y los datos son correctos

8.1 Actualización de la BD y mensaje satisfactorio.

Caso contrario

8.2 Mensaje de error

6.8.1.1.5.1.4. Salidas

Destino de las salidas

Se vuelve a cargar la página cuando los datos ingresados son correctos.

6.8.1.1.5.2. Interfaces de Hardware

- El monitor es el principal medio hardware de visualización, el cual se lo utilizará para mostrar cada uno de los procesos que se efectuarán.

6.8.1.1.5.3. Interfaces de software

- La herramienta de desarrollo que se utilizará es NetBeans IDE 6.9.1 para realizar cada una de las aplicaciones que sean necesarios para este requerimiento.
- La Base de Datos está implementada en Oracle 10G Express Edition.

6.8.1.1.5.4. Interfaces de Comunicación

El Sistema utilizará el protocolo TCP/IP para establecer la comunicación.

MÓDULO INDICADOR EVALUADO

6.8.1.1.6. Requerimiento Funcional 11

6.8.1.1.6.1. Especificaciones

6.8.1.1.6.1.1. Introducción

El sistema deberá permitir gestionar datos del Indicador evaluado.

6.8.1.1.6.1.2. Entrada

Fuentes de Entrada

Usuario

Contraseña

Rol

Frecuencia

Bajo demanda

Requisitos de control

Controla que los campos del formulario no estén vacíos u además que los datos sean correctos

Entradas válidas

Todos los campos sean válidos.

6.8.1.1.6.1.3. Procesos

1. El usuario ingresará Nombre de usuario y contraseña de autenticación.

2. El sistema validará la contraseña.

3. Si el dato es verdadero

3.7 Ingresará al sistema

Caso contrario

3.8 Mensaje de error

4. Elije el Módulo Indicador Evaluado

5. Administrar

6. El usuario escoge si desea Ingresar, modificar o eliminar

7. Validación de los datos.

8. Si todos los campos están llenos y los datos son correctos

8.1 Actualización de la BD y mensaje satisfactorio.

Caso contrario

8.2Mensaje de error

6.8.1.1.6.1.4. Salidas

Destino de las salidas

Se vuelve a cargar la página cuando los datos ingresados son correctos.

6.8.1.1.6.2. Interfaces de Hardware

- El monitor es el principal medio hardware de visualización, el cual se lo utilizará para mostrar cada uno de los procesos que se efectuarán.

6.8.1.1.6.3. Interfaces de software

- La herramienta de desarrollo que se utilizará es NetBeans IDE 6.9.1 para realizar cada una de las aplicaciones que sean necesarios para este requerimiento.
- La Base de Datos está implementada en Oracle 10G Express Edition.

6.8.1.1.6.4. Interfaces de Comunicación

El Sistema utilizará el protocolo TCP/IP para establecer la comunicación.

MÓDULO HOJA DE RUTA Y VERIFICACIÓN

6.8.1.1.7. Requerimiento Funcional 12, 13

6.8.1.1.7.1. Especificaciones

6.8.1.1.7.1.1. Introducción

El sistema deberá permitir gestionar datos de Hoja de Ruta y Evidencia.

6.8.1.1.7.1.2. Entrada

Fuentes de Entrada

Usuario

Contraseña

Rol

Frecuencia

Bajo demanda

Requisitos de control

Controla que los campos del formulario no estén vacíos u además que los datos sean correctos

Entradas válidas

Todos los campos sean válidos.

6.8.1.1.7.1.3. Procesos

1. El usuario ingresará Nombre de usuario y contraseña de autenticación.
2. El sistema validará la contraseña.

3. Si el dato es verdadero

3.9 Ingresará al sistema

Caso contrario

3.10 Mensaje de error

4. Elije el módulo Hoja de Ruta y Verificación o el módulo Evidencia.

5. Administrar

6. El usuario escoge si desea Ingresar, eliminar o modificar

7. Validación de los datos.

8. Si todos los campos están llenos y los datos son correctos

8.1 Actualización de la BD y mensaje satisfactorio.

Caso contrario

8.2 Mensaje de error

6.8.1.1.7.1.4. Salidas

Destino de las salidas

Se vuelve a cargar la página cuando los datos ingresados son correctos.

6.8.1.1.7.2. Interfaces de Hardware

- El monitor es el principal medio hardware de visualización, el cual se lo utilizará para mostrar cada uno de los procesos que se efectuarán.

6.8.1.1.7.3. Interfaces de software

- La herramienta de desarrollo que se utilizará es NetBeans IDE 6.9.1 para realizar cada una de las aplicaciones que sean necesarios para este requerimiento.
- La Base de Datos está implementada en Oracle 10G Express Edition.

6.8.1.1.7.4. Interfaces de Comunicación

El Sistema utilizará el protocolo TCP/IP para establecer la comunicación.

MÓDULO HOJA DE RUTA Y VERIFICACIÓN

6.8.1.1.8. Requerimiento Funcional 14,15,16,17

6.8.1.1.8.1. Especificaciones

6.8.1.1.8.1.1. Introducción

El sistema deberá permitir gestionar datos de Roles, Usuarios, Permisos, y claves.

6.8.1.1.8.1.2. Entrada

Fuentes de Entrada

Usuario

Contraseña

Rol

Frecuencia

Bajo demanda

Requisitos de control

Controla que los campos del formulario no estén vacíos u además que los datos sean correctos

Entradas válidas

Todos los campos sean válidos.

6.8.1.1.8.1.3. Procesos

1. El usuario ingresará Nombre de usuario y contraseña de autenticación.
2. El sistema validará la contraseña.

3. Si el dato es verdadero

3.1 Ingresará al sistema

Caso contrario

3.2 Mensaje de error

4. Elije el módulo que desee.

5. El usuario escoge si desea Ingresar, eliminar o modificar

6. Validación de los datos.

7. Si todos los campos están llenos y los datos son correctos

7.1 Actualización de la BD y mensaje satisfactorio.

Caso contrario

7.2 Mensaje de error

6.8.1.1.8.1.4. Salidas

Destino de las salidas

Se vuelve a cargar la página cuando los datos ingresados son correctos.

6.8.1.1.8.2. Interfaces de Hardware

- El monitor es el principal medio hardware de visualización, el cual se lo utilizará para mostrar cada uno de los procesos que se efectuarán.

6.8.1.1.8.3. Interfaces de software

- La herramienta de desarrollo que se utilizará es NetBeans IDE 6.9.1 para realizar cada una de las aplicaciones que sean necesarios para este requerimiento.
- La Base de Datos está implementada en Oracle 10G Express Edition.

6.8.1.1.8.4. Interfaces de Comunicación

El Sistema utilizará el protocolo TCP/IP para establecer la comunicación.

6.8.1.2. Requerimientos No Funcionales

A continuación se muestran los requerimientos no funcionales con sus respectivas características:

6.8.1.2.1. Rendimiento

Tiempos de respuesta para acceder a la página de autenticación del sistema máximo de 30 segundos.

6.8.1.2.2. Disponibilidad

El sistema estará fuera de línea máximo una hora.
Empleo de sistemas de respaldo.

6.8.1.2.3. Seguridad

Para ingresar al sistema primero deberá el usuario autenticarse, y según los permisos que posea podrá acceder a la información

6.8.1.2.4. Portabilidad

Garantizar compatibilidad con otros sistemas operativos: Windows XP, Windows 2003, Windows Vista, Linux.

6.8.1.2.5. Mantenibilidad

Empleo del Modelo Microsoft Solution Framework.
Documentación del diseño y de la codificación de la solución.

6.8.1.2.6. Escalabilidad

Diseño de la arquitectura empleando módulos independientes.

6.8.1.2.7. Reusabilidad

Uso de estándares en los formatos para los datos.

6.8.1.2.8. Interfaces

Interfaces realizadas en Netbeans 6.9.1.

6.8.1.2.9. Usabilidad

Facilidad de uso.

6.8.1.3. Actores

SuperAdministrador

Es la persona que posee el control total de todo el Sistema, y es el encargado de otorgar los diferentes permisos a los demás Usuarios además él se encargará de asignar las personas responsables de evaluar cada carrera.

Administrador Normal

Serán los encargados de administrar en su totalidad el sistema desarrollado pero solo de las carreras que le ha sido asignada.

- Control Total
  - Lectura.
  - Escritura.
  - Actualización.
  - Gestión de reportes
    - Generar.
    - Imprimir.

- Visualizar.

Usuario Normal

Podrán acceder al sistema pero con restricciones por ejemplo tendrán permisos de:

- Solo Lectura.
- Solo Escritura.
- Lectura y Escritura.
- Visualizar Reportes en forma general.

6.8.1.3.1. Casos de Uso

CU - AUTENTICACIÓN

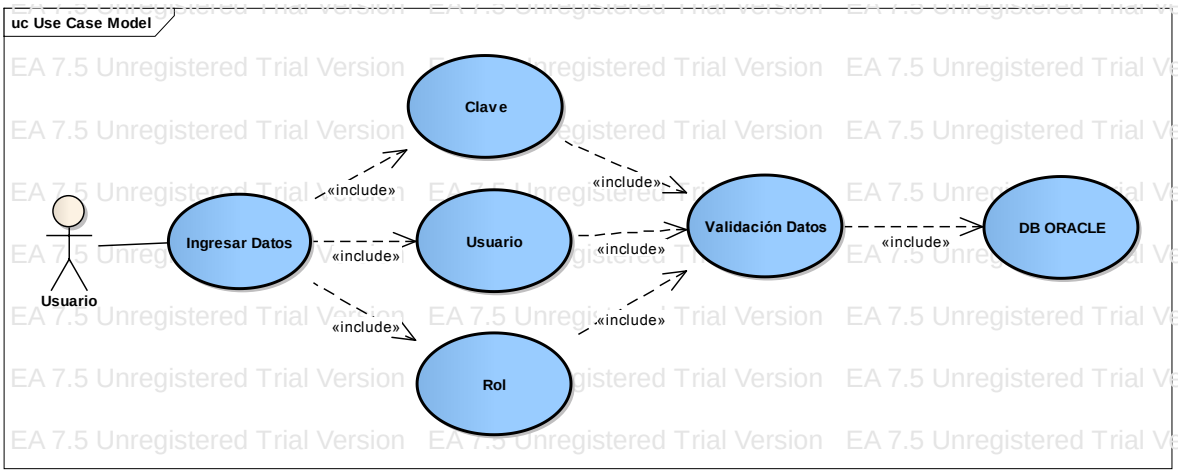


Figura VI.49. Caso de Uso Autenticación

CU - USUARIO ADMINISTRADOR

Este tipo de usuarios tendrá un control total sobre el sistema pero solo de las carreras de las cuales son responsables.

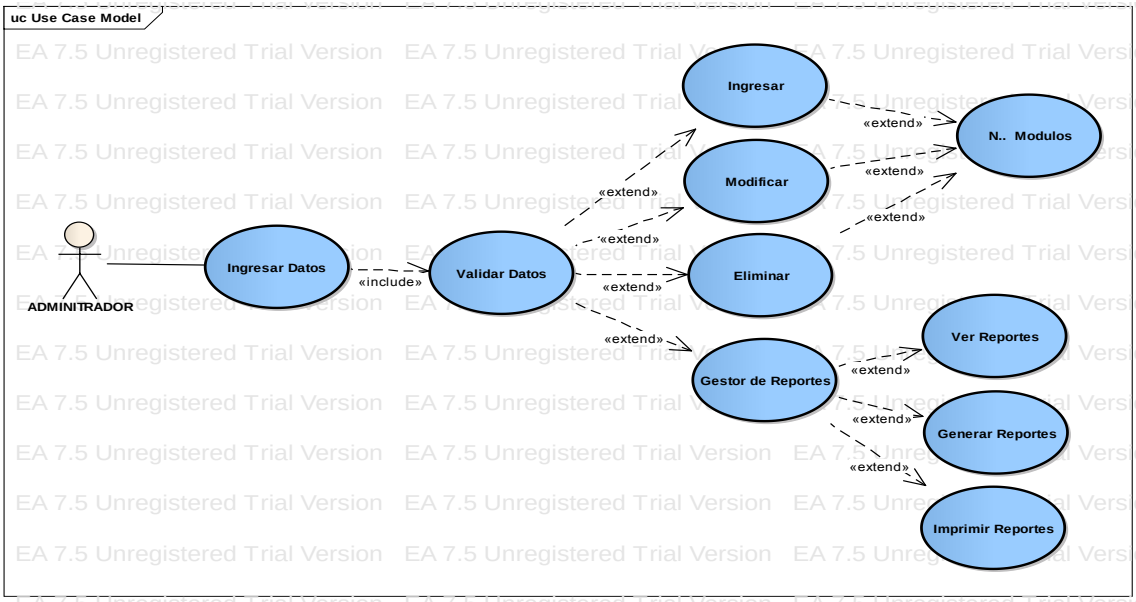


Figura VI.50. Caso de Uso Usuario Administrador Normal

CU - USUARIO NORMAL

El Usuario Normal podrá ingresar a la aplicación según los permisos que tenga, y no tendrá el control total sobre la aplicación.

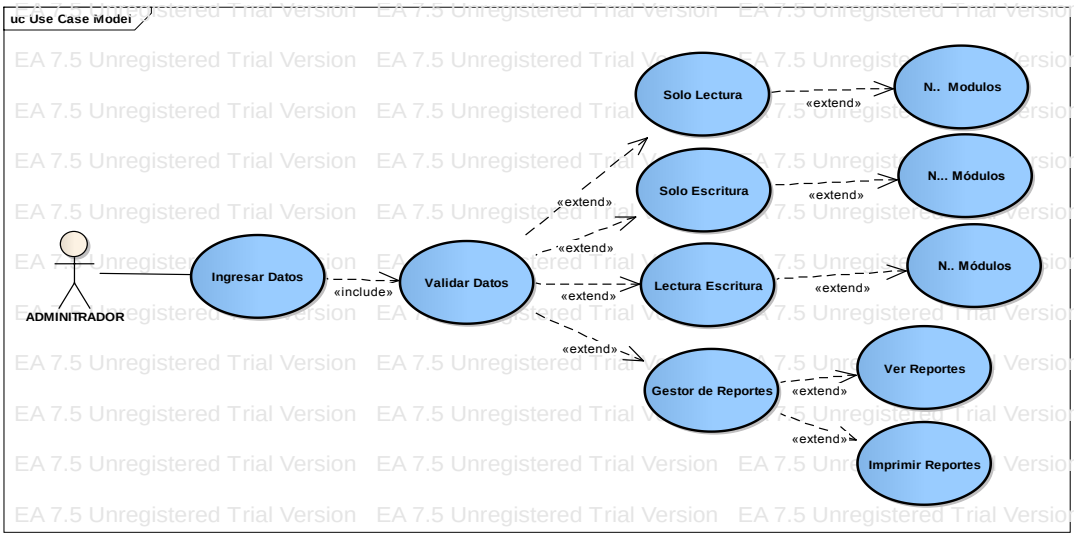


Figura VI.51. Caso de Uso Usuario Normal

6.8.1.3.2. Escenarios

AUTENTICACIÓN USUARIO SUPERADMINSTRADOR

- **Personas Alternativas**

Tabla VI.XXIII. Personas alternativas para SuperAdminitrador

PERSONA	DESVIACIONES (si es aplicable)
Persona delegada de la Comisión de evaluación	Ninguna

- **Descripción del Escenario**

El **SuperAdministrador** ingresará su Usuario, Clave en la página de autenticación. Este administrador tendrá el control total y absoluto del sistema. Será el encargado de asignar permisos, y asignar las carreras a las personas que serán responsables de la evaluación.

AUTENTICACIÓN USUARIOS ADMINSTRADORES

- **Personas Alternativas**

Tabla VI.XXIV. Personas alternativas para Administradores

PERSONA	DESVIACIONES (si es aplicable)
Director de Escuela	Ninguna
Decanos	Ninguna
Vicedecanos	Ninguna
Personas delegadas por la Comisión de evaluación	Ninguna

- **Descripción del Escenario**

El **Administrador** ingresará su Usuario, Clave y Rol (Administrador) en la página de autenticación. Si los datos son correctos se le presentará la página principal en la cual se les muestra el menú de Módulos de gestión. El Usuario

Administrador tienen control total sobre las carreras que le han sido asignadas es decir se le permite: Ingreso, Modificación. Eliminación, crear Reportes, ver Reportes e Imprimir reportes.

AUTENTICACIÓN USUARIOS NORMALES

- **Personas Alternativas**

Tabla VI.XXV. Personas Alternativas para Usuarios Normales

PERSONA	DESVIACIONES (si es aplicable)
Secretarias	Ninguna
Personas Elegidas por la Comisión de Evaluación	Ninguna

- **Descripción del Escenario**

La autenticación es un proceso realizado por los usuarios principales como son: Secretarias y ciertas personas elegidas pos la comisión de evaluación, es decir, usuarios que contribuirán con el proceso de evaluación de Carreras.

Inicia el proceso cuando los usuarios ingresan su Usuario y Clave y eligen su rol (normal), datos que son validados en la BD Oracle.

Si la información proporcionada es correcta le dará paso al uso de la aplicación y acceder con los permisos que se les haya otorgado, caso contrario se le mostrará nuevamente la página autenticación.

Para el ingreso de Usuario y Clave los usuarios deberán ser registrados por el SuperAdministrador o Administrador y les será otorgado

6.8.1.3.3. Glosario de Términos

Tabla VI.XXVI Glosario de Términos

TERMINO	DESCRIPCION
MSF	Microsoft Solution Framework, metodología empleada para el desarrollo de la aplicación.
UML	Lenguaje Unificado de Modelado, es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema.
Módulos	Distintas dependencias del sistema que brinda ciertos servicios a los usuarios.
Hardware	Las características físicas y tangibles del computador utilizado.
Clave	Es una serie secreta de caracteres que permite a un usuario tener acceso a un ordenador, programa, sitio.
Rendimiento	Capacidad de evolución y aprovechamiento del sistema.
Servidor	Un servidor es un ordenador de gran potencia, que se encarga de "prestar un servicio" a otros ordenadores que se conectan a el
Sistema Operativo	Es un programa informático que actúa de interfaz entre los dispositivos de hardware y el usuario. Es responsable de gestionar, coordinar las actividades y llevar a cabo el intercambio de recursos de un computador.
Base de Datos	Conjunto de datos organizados para su almacenamiento en la memoria de un ordenador o computadora, diseñado para facilitar su mantenimiento y acceso de una forma estándar. La información se organiza en campos y registros.
Usuario	Persona que utiliza el sistema
Administrador	Es la persona encargada de administrar en su totalidad el sistema, además puede generar reportes.
Reporte	El reporte es aquel documento que se utilizará cuando se quiera obtener o dar información de una determinada cuestión.

Metodología	Conjunto de pasos lógicos y ordenados a seguir para realizar una actividad, la metodología son micro actividades. Primero debemos tener un modelo para seguir una metodología.
Internet	conjunto descentralizado de redes de comunicación interconectadas que utilizan la familia de protocolosTCP/IP, garantizando que las redes físicas heterogéneas que la componen funcionen como una red lógica única, de alcance mundial.
Evaluación	Es la determinación sistemática del mérito, el valor y el significado de algo o alguien en función de unos criterios respecto a un conjunto de normas.

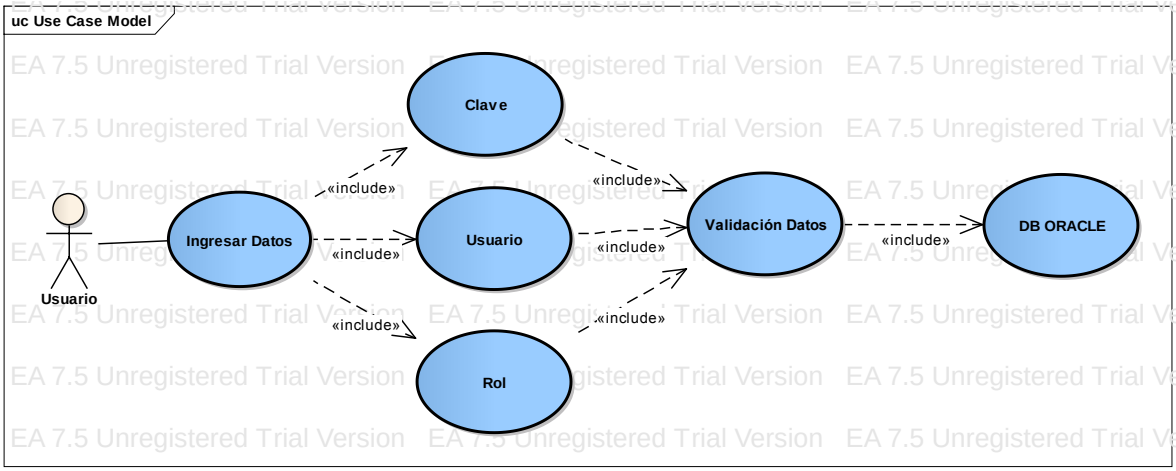
6.8.1.3.4. Refinar los Casos de Uso

CU - AUTENTICACIÓN DE USUARIOS

Tabla VI.XXVII. Autenticación de Usuarios

IDENTIFICAR CASO DE USO	CU – Autenticación
NOMBRE DEL CASO DE USO	Realizar autenticación de los clientes.
ACTORES	Directores de Escuela, Decanos, Vicedecanos, Secretarias
PROPÓSITO	Realizar el proceso de autenticación y validación de de los usuarios y a la vez otorgarles los permisos correspondientes.
VISIÓN GENERAL	Los usuarios acceden a la página de autenticación y deben ingresar su Usuario, Clave y su Rol (Administrador o Normal) respectivo al sistema, y así podrán acceder a la página Principal
TIPO	Primario, real y expandido
REFERENCIAS	Caso de uso : Autenticación
CURSO TÍPICO DE EVENTOS	
ACCIÓN DEL ACTOR	**RESPUESTA DEL SISTEMA**

1. Este caso de uso empieza cuando los clientes se autentican para hacer uso del Sistema de Evaluación de Carreras servicio de Internet contratado.	2. El Sistema muestra como página Inicial la página de autenticación que les permite a los usuarios ingresar sus datos
3. Los usuarios ingresan su Usuario, Clave y eligen su rol (Administrador , Normal)	4. Valida los datos ingresados
	5. El Sistema realiza una búsqueda de los datos ingresados, en la Base de Datos Oracle y los datos son correctos aparecerá la pagina Principal.
CURSOS ALTERNATIVOS	
Control de Campos Vacíos 4.1.- Ingreso de Usuario Cuando no se ingresa un Usuario, el sistema se vuelve a cargar con la misma página de autenticación 4.2.- Ingreso de Clave Cuando no se ingresa una Clave, el sistema se vuelve a cargar con la misma página de autenticación. 5.1. Si el Usuario está en la base de datos pero tiene un estado de bloqueado aparece una pantalla con el mensaje ”Usuario Bloqueado contáctese con el Administrador”	



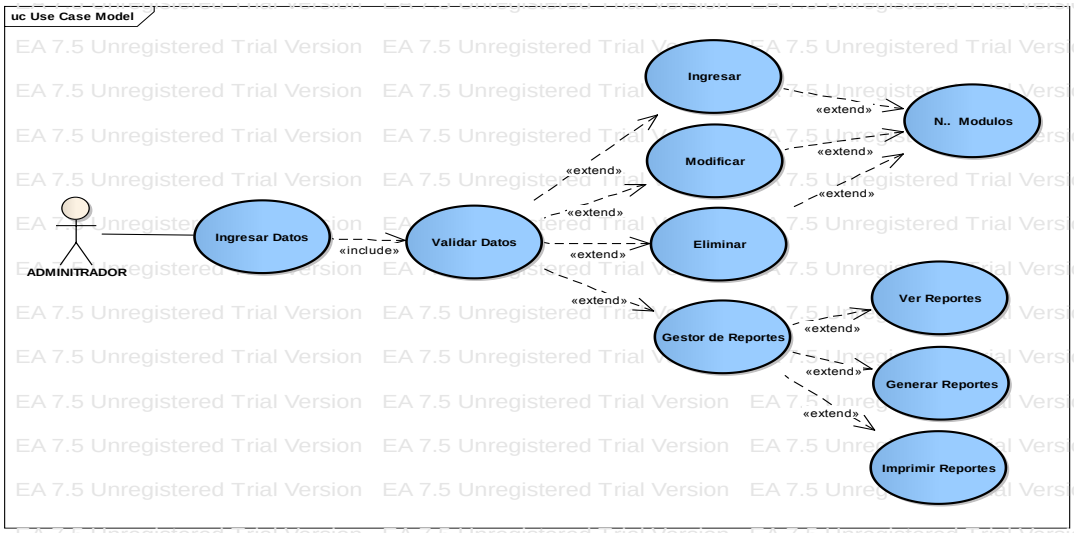
Figuna VI.52. CU- Autenticación

CU - USUARIO ADMINISTRADOR

Tabla VI.XXVIII. Usuario Administrador

IDENTIFICAR CASO DE USO	CU – Usuario Administrador
NOMBRE DEL CASO DE USO	Realizar administración del sistema
ACTORES	Directores de Escuela, Decanos , Vicedecanos y personas delegadas por la comisión de Evaluación
PROPÓSITO	Realizar la administración del proceso de Evaluación de las carreras Asignadas.
VISIÓN GENERAL	Los administradores deben autenticarse en el sistema ingresando su Usuario y Clave respectivo, así podrán acceder al sistema.
TIPO	Primario, real y expandido
REFERENCIAS	Caso de uso : Usuario Administrador
CURSO TÍPICO DE EVENTOS	
ACCIÓN DEL ACTOR	**RESPUESTA DEL SISTEMA**
1. Este caso de uso empieza cuando los Administradores desean evaluar a las carreras que les han sido asignadas para	2. El sistema cuenta con un control para que se autentiquen.

evaluar.	
3. Ingresan su Usuario y Clave	4. Valida los datos ingresados
	5. Al autenticarse podrán acceder al sistema y tener un control total de las carreras que les han sido asignadas.
CURSOS ALTERNATIVOS	
2.1.- El sistema no presenta pantalla para ingreso de datos	
Control de Campos Vacíos 4.1.- Ingreso de Usuario Cuando no se ingresa un Usuario, el sistema se vuelve a cargar con la misma página de autenticación 4.2.- Ingreso de Clave Cuando no se ingresa una Clave, el sistema se vuelve a cargar con la misma página de autenticación. 5.1. Si el Usuario está en la base de datos pero tiene un estado de bloqueado aparece una pantalla con el mensaje ”Usuario Bloqueado contáctese con el Administrador”	

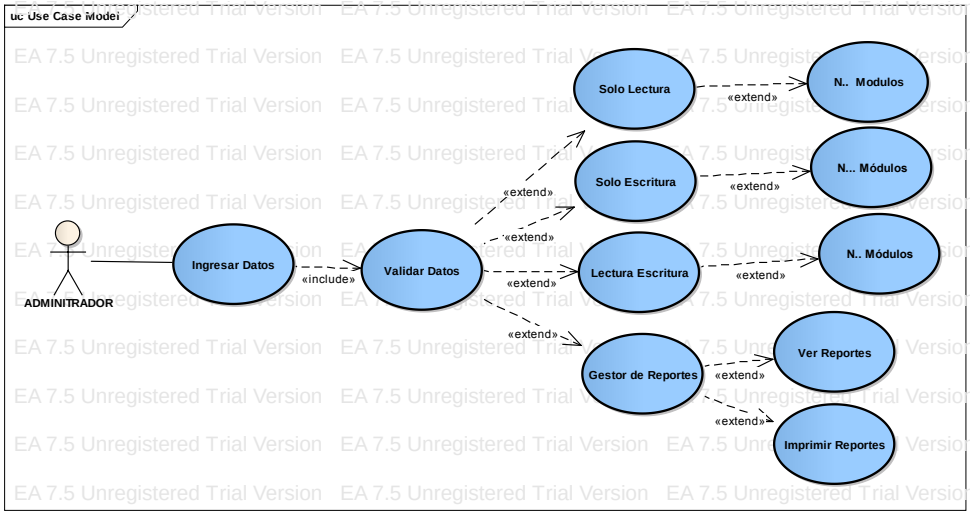


Figuna VI.53. CU- Usuarios Administrador

CU - USUARIO NORMAL

Tabla VI.XXIX Usuario Normal

IDENTIFICAR CASO DE USO	CU – Usuario Normal
NOMBRE DEL CASO DE USO	Evaluar las Carreras
ACTORES	Secretarias y ciertas personas elegidas por la Comisión de Evaluación.
PROPÓSITO	Realizar la el proceso de ingreso de información al sistema para el proceso de evaluación de carreras.
VISIÓN GENERAL	Los Usuarios deben autenticarse en el sistema ingresando su Usuario y Clave respectivo, así podrán acceder a la página principal.
TIPO	Primario, real y expandido
REFERENCIAS	Caso de uso : Usuario Normal
CURSO TÍPICO DE EVENTOS	
ACCIÓN DEL ACTOR	**RESPUESTA DEL SISTEMA**
1. Este caso de uso empieza cuando los usuarios deben ingresar información al sistema.	2. El sistema cuenta con un control para que se autentiquen.
3. Ingresan su Usuario y Clave	4. Valida los datos ingresados
	5. Al autenticarse podrán acceder al sistema y hacer uso de los permisos que les ha sido otorgados.
	6. Presenta una página en donde se encuentra el menú principal.
CURSOS ALTERNATIVOS	
Control de Campos Vacíos 4.1.- Ingreso de Usuario Cuando no se ingresa un Usuario, el sistema se vuelve a cargar con la misma página de autenticación 4.2.- Ingreso de Clave Cuando no se ingresa una Clave, el sistema se vuelve a cargar con la misma página de autenticación. 5.1. Si el Usuario está en la base de datos pero tiene un estado de bloqueado aparece una pantalla con el mensaje ”Usuario Bloqueado contáctese con el Administrador”	



Figuna VI.54. CU- Usuarios Administrador

6.8.2. Diseño Lógico

6.8.2.1. Tecnología a utilizar en el proyecto

Se analizaron las herramientas para el desarrollo del sistema y se escogió la plataforma de Desarrollo (IDE) de Java Netbeans 6.9.1, debido a la facilidad que esta herramienta nos brinda y las funcionalidades que esta posee.

El Sistema Gestor de Base de datos a utilizar es Oracle 10G EX.

Para la documentación del sistema se utilizó la herramienta UML MSF (Microsoft Solution Framework).

6.8.2.2. Diagramas de Secuencia

AUTENTICACIÓN DE USUARIOS

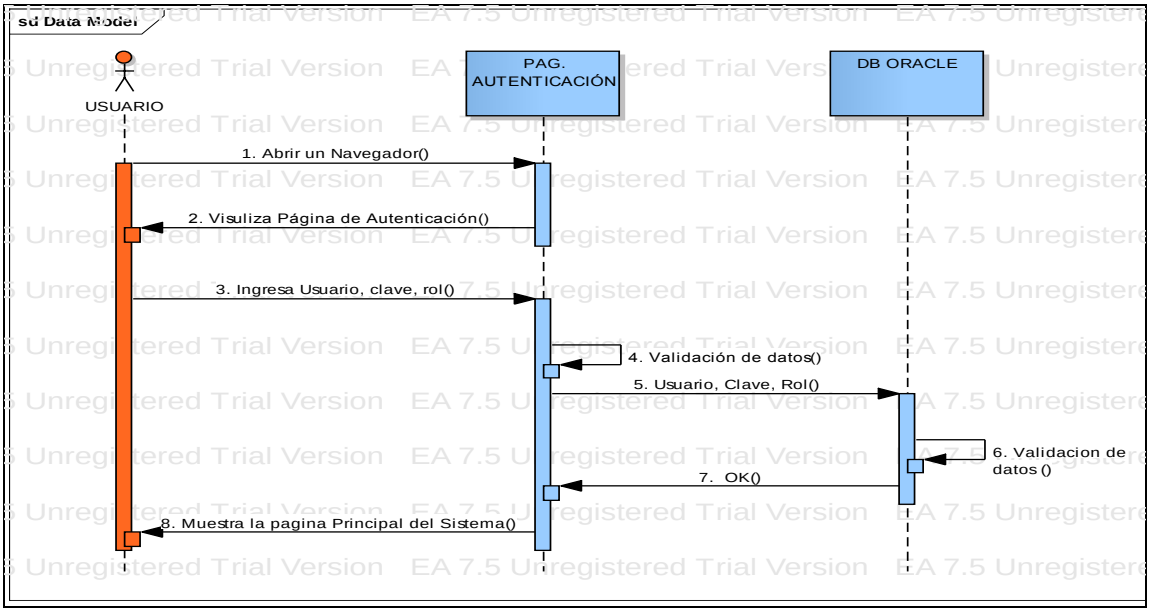


Figura VI.55. Diagrama de Secuencia Autenticación de Usuarios

Los siguientes Diagramas de secuencias son generales para todos los módulos existentes, en e este caso se realizara de Ingreso y Eliminación de Criterios.

INGRESAR CRITERIO

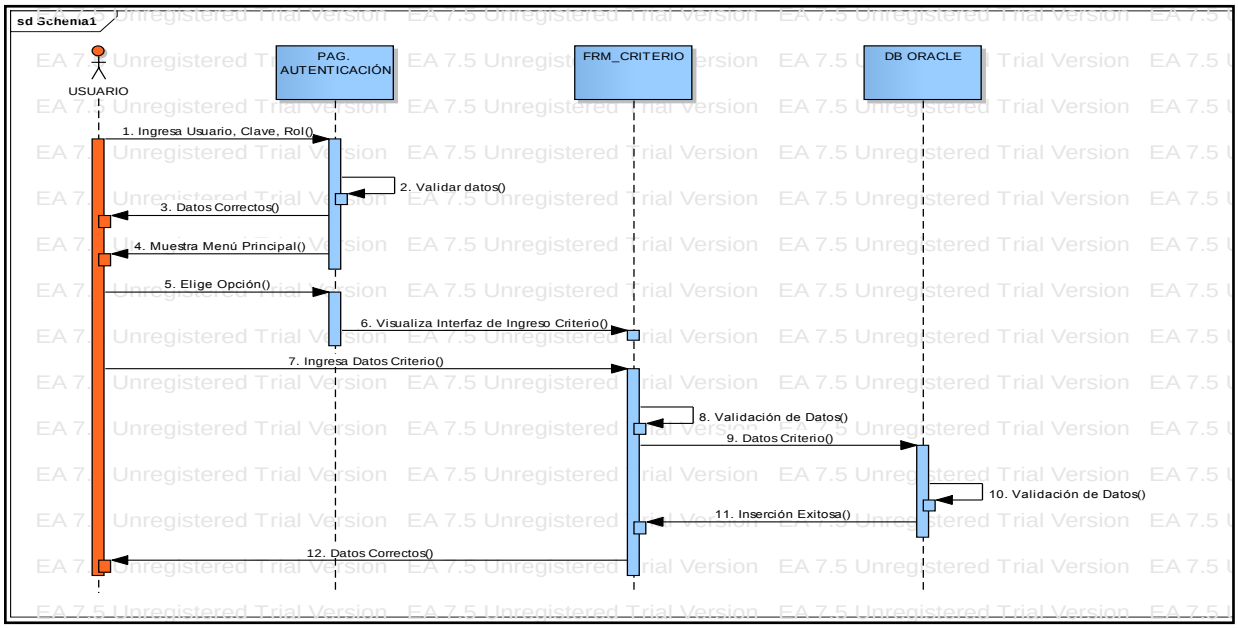


Figura VI.56. Diagrama de Secuencia Ingresar Criterio

ELIMINAR CRITERIO

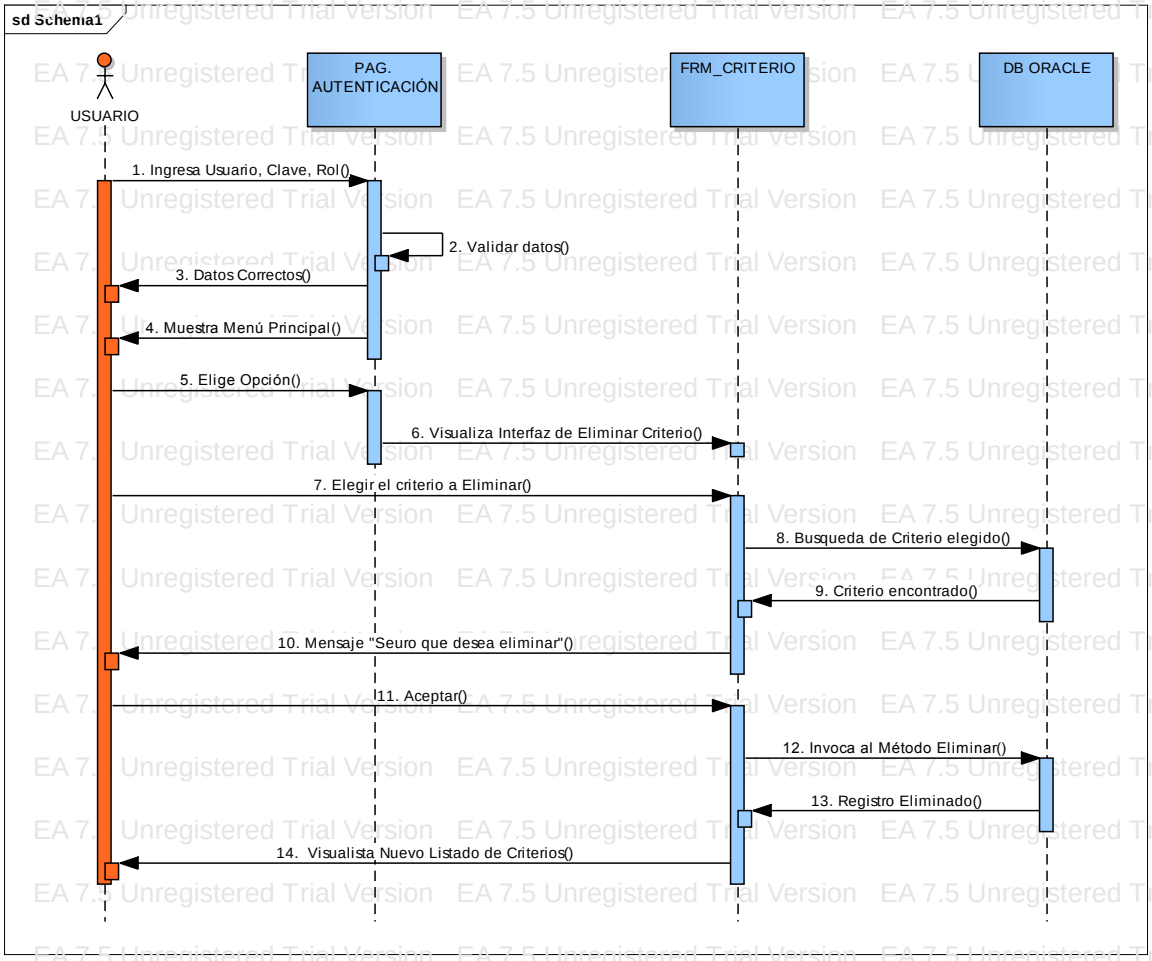


Figura VI.57. Diagrama de Secuencia Eliminar Criterio

6.8.2.3. Diagramas de Clase

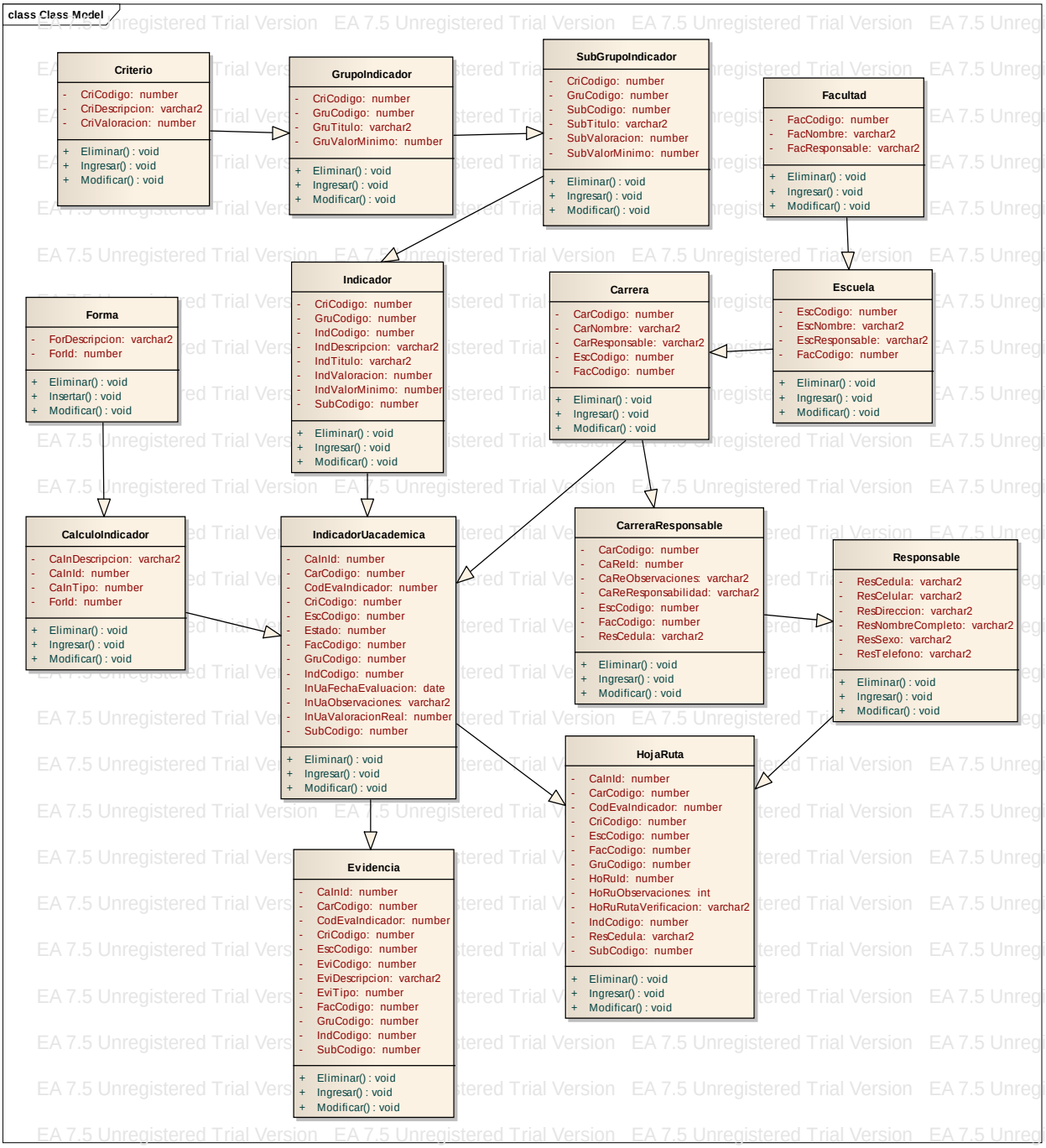


Figura VI.58. Diagrama de clases General

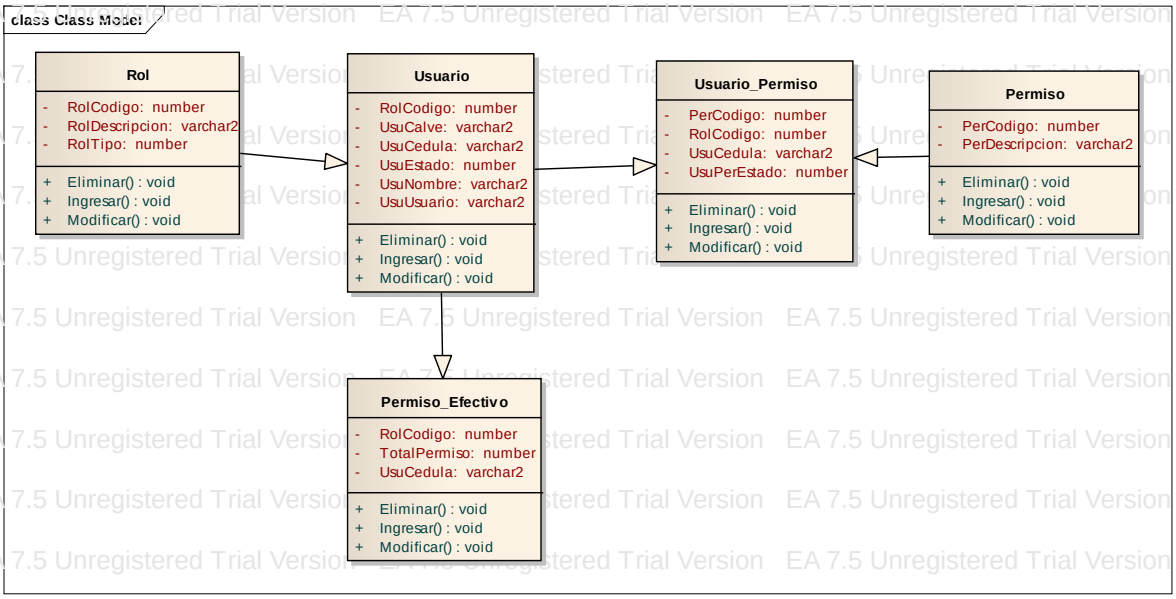


Figura VI.59. Diagrama de Clases Usuarios y Roles

6.8.2.4. Diseño de Interfaces de Usuario

- Autenticación de Usuarios.

Figura VI.60. Autenticación de Usuarios

- Página Principal del SuperAdministrador

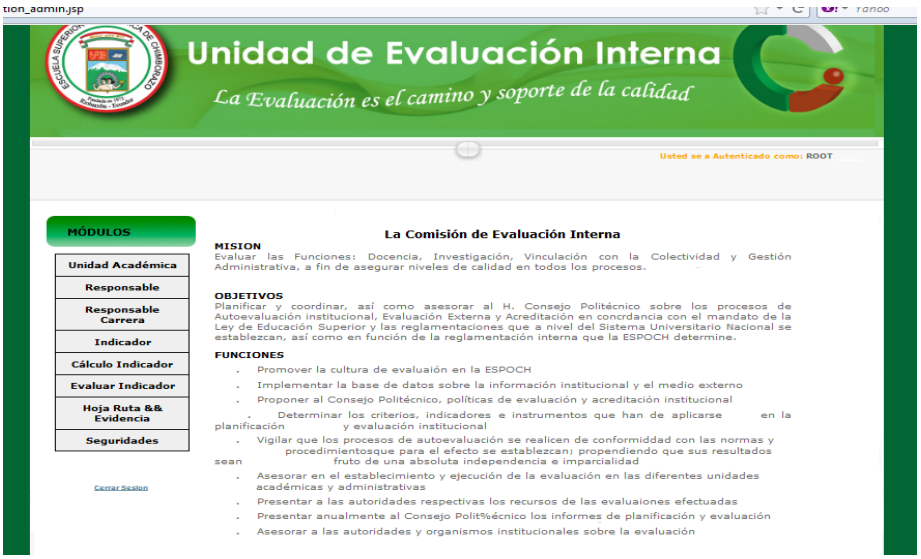


Figura VI.61.Página Principal de SuperAdministrador

- Gestión Unidad Académica

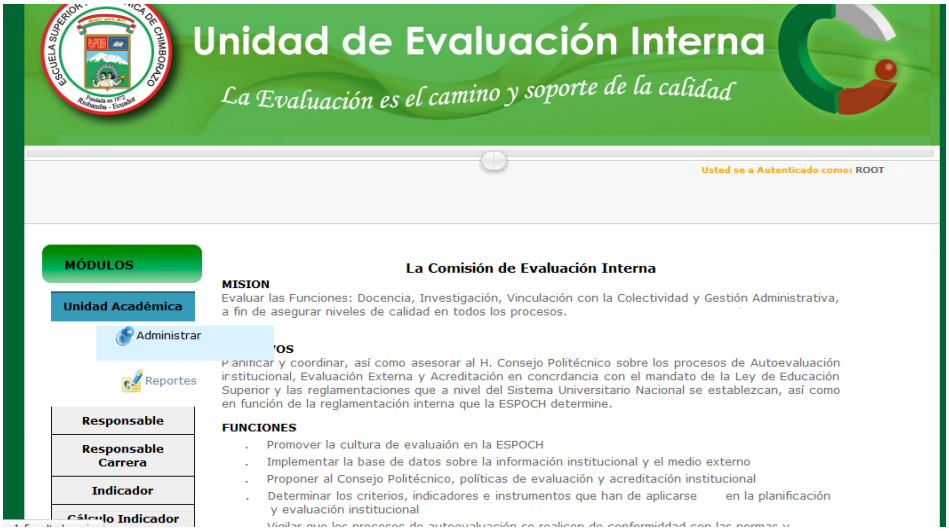


Figura VI.62. Gestión Unidad Académica

- Listado Facultades

Figura VI.63. Listado Facultades

- Ingresar Nueva Facultad

Figura VI.64. Ingresar Nueva Facultad

- Modificar Facultad

Figura VI.65. Modificar

Figura VI.66. Modificar Facultad

- Eliminar Facultad

Figura VI.67. Eliminar

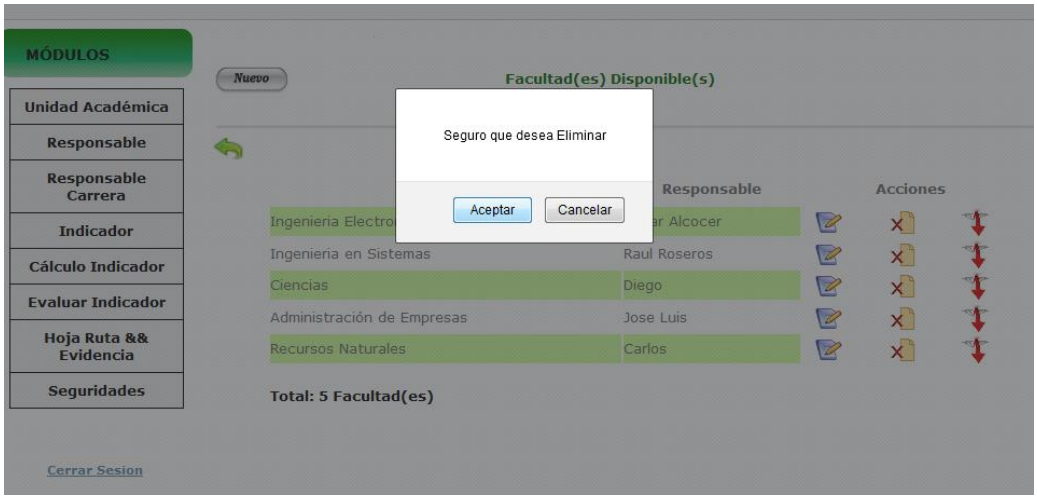


Figura VI.68. Eliminar Criterio

6.8.3. Diseño Físico

6.8.3.1. Diagrama de Actividades

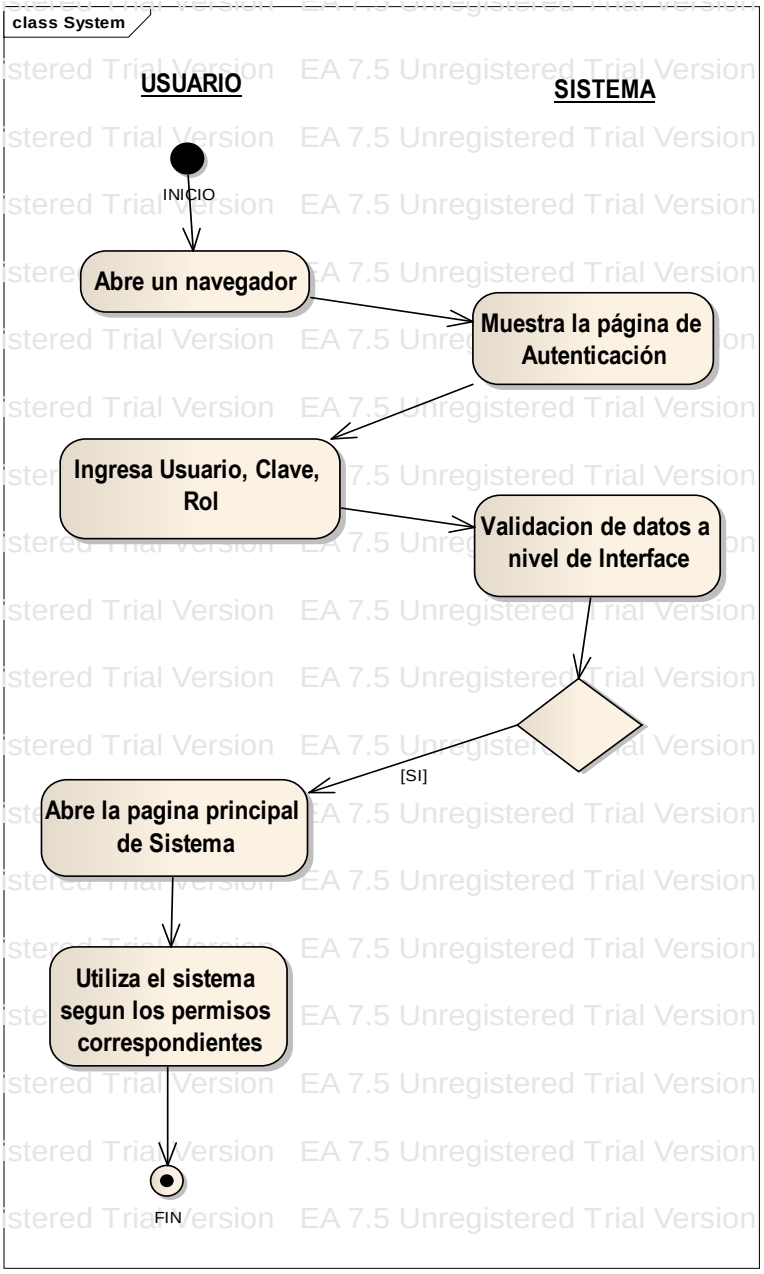


Figura VI.69. Diagrama de Actividades

6.8.3.2. Diagrama de Componentes

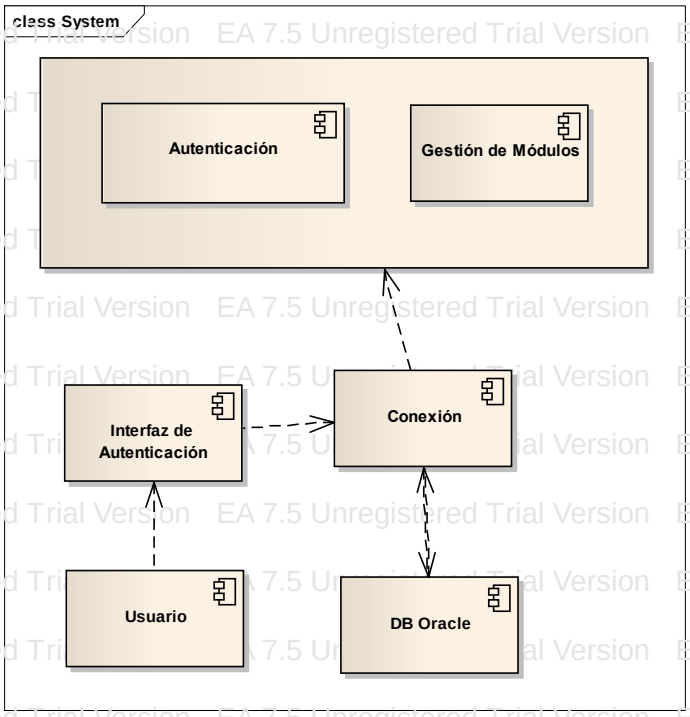


Figura VI.70. Diagrama de Componentes

6.8.3.3. Diagrama de Implementación

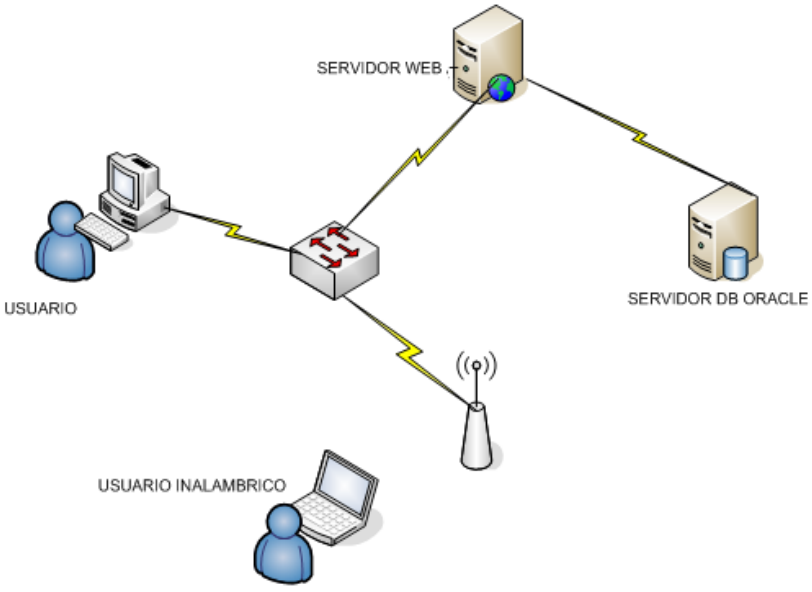
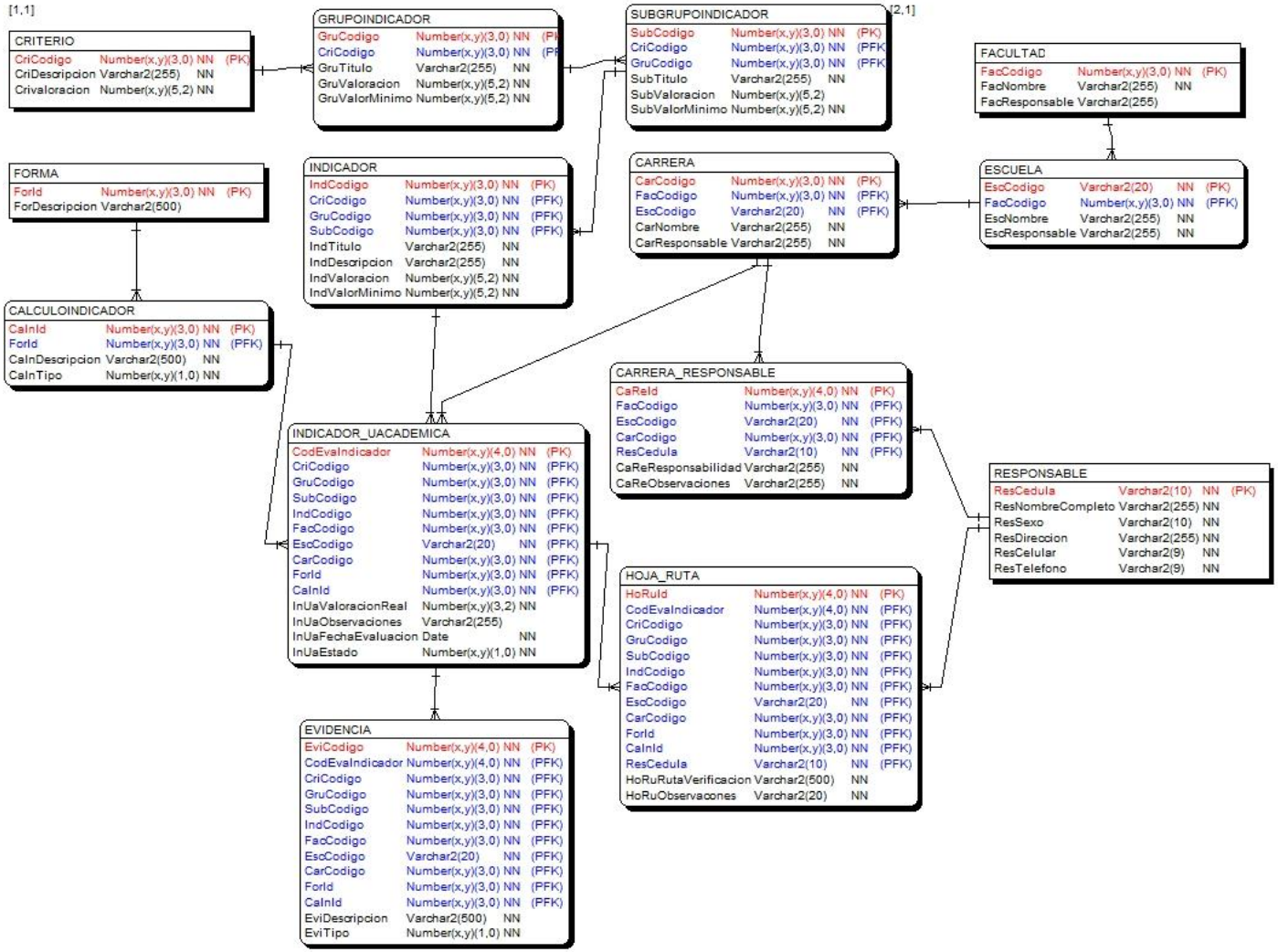


Figura .71. Diagrama de Implementación

6.8.3.4. Modelo Físico de la Base de Datos



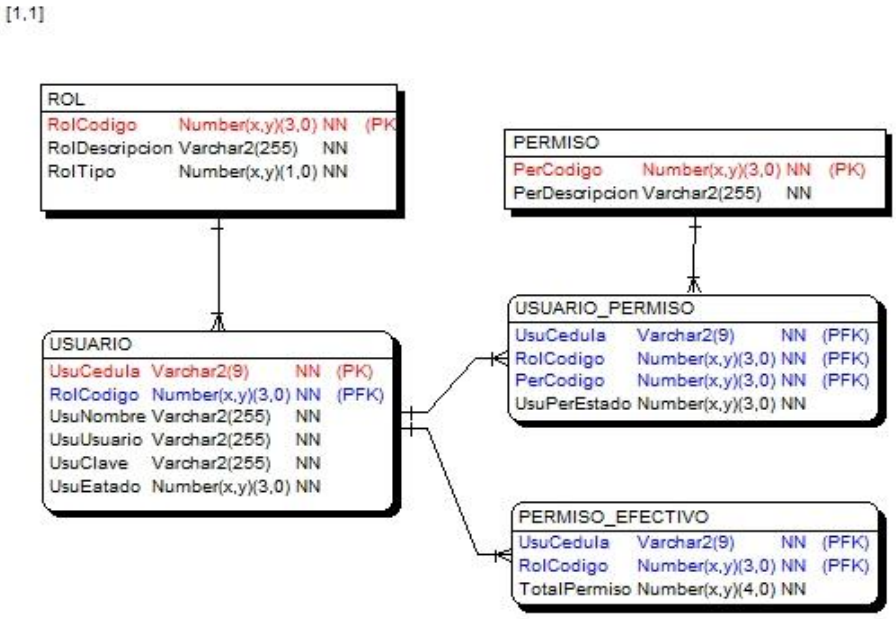


Figura VI.72. Diseño Físico de la Base de Datos

6.9. Fase de Desarrollo

Junto con la Fase de Planeación y de Estabilización se debe obtener versiones del producto para proporcionar al cliente una visión e idea clara de la aplicación desarrollada.

6.9.1. Nomenclatura y Estándares

Los estándares manejados en el desarrollo de los productos de la aplicación se describen a continuación:

Tabla VI.XXX. Nomenclatura y estándares

ARCHIVO	EXTENSIÓN	NOMENCLATURA
Programas en Java	.jsp	NombreDeArchivo.jsp
Imágenes	.ico	NombreDeImagen.ico

JAVA SERVER PAGES (JSP)

Es una es una tecnología Java que permite a los desarrolladores de software generar dinámicamente HTML, XML u otros tipos de documentos, en respuesta al requerimiento de un cliente web. Esta tecnología permite que códigos Java y ciertas otras acciones predefinidas, sean integrados en contenido estático. Los JSPs son compilados en forma de Java Servlets empleando un compilador JSP.

FICHEROS FUENTE JAVA

Cada fichero con código Java debe contener una sola clase pública o interfaz. Cuando se asocian clases privadas e interfaces con una clase pública, se pueden colocar en el mismo fichero fuente que la clase pública. La clase pública debería ser la primera clase o interface del fichero. Los ficheros fuente con código Java, deberían seguir el siguiente orden:

Comentarios iniciales: Todos los ficheros fuente deben comenzar con un comentario al estilo C que indique el programador, o programadores, la fecha, una nota de copyright y una pequeña descripción del propósito del programa. Por ejemplo:

```
/*
* Nombre de la clase
*
* Versión
*
* Copyright
*/
```

Sentencias import y package: Por ejemplo:

```
importjava.applet.Applet;
importjava.awt.*;
import.java.net.*;
```

Declaraciones de clase e interface: En la tabla siguiente se describen las diferentes partes de la declaración de una clase o interface, y en el orden que deben aparecer, tal como se muestra en el código de ejemplo.

Tabla VI.XXXI. Declaraciones

	Parte de la declaración	Notas
1	Comentario de documentación (/**...*/)	En la parte de Comentario de Documentación se indica la información que debe ir en este comentario
2	Sentencia class o interface	
3	Comentarios a la implementación de la clase/interface (/*...*/), en caso necesario	Este comentario debería contener cualquier información de alcance que no sea adecuada para estar situada en el comentario que permite generar posteriormente la documentación.
4	Variables de la clase (estáticas)	En primer lugar las variables public, luego las protected y finalmente las private.
5	Variables de instancia	Primero las públicas, luego las protegidas y las privadas.
6	Constructores	
7	Métodos	Estos métodos deben estar agrupados por funcionalidad, en vez de por el ámbito o accesibilidad. Por ejemplo, un método de una clase privada puede encontrarse entre dos métodos de instancia públicos. La finalidad es que el código sea más fácil de leer y entender.

6.9.2. Capa de Datos

6.9.2.1. Diccionario de Datos

Tabla VI.XXXII. Diccionario de Datos

Nombre Campo	**Tipo dato**	**longitud**	**Clave Primaria**	**Calculado**
TABLA: CRITERIO				
CriCodigo	Number	(3,0)	x	No

CriDescripción	Varchar2	225		No
CriValoracion	Number	(5,2)		No
TABLA: GRUPOINDICADOR				
GruCodigo	Number	(3,0)	x	No
CriCodigo	Number	(3,0)	x	
GruTitulo	Varchar2	255		No
GruValoración	Number	(5,2)		No
GruValorMinimo	Number	(5,2)		No
TABLA: SUBGRUPOINDICADOR				
SubCodigo	Number	(3,0)	x	No
GruCodigo	Number	(3,0)	x	No
CriCodigo	Number	(3,0)	x	
SubTitulo	Varchar2	255		No
SubValoracion	Number	(5,2)		No
SubValorMinimo	Number	(5,2)		No
TABLA: INDICADOR				
IndCodigo	Number	(3,0)	x	No
SubCodigo	Number	(3,0)	x	No
GruCodigo	Number	(3,0)	x	No
CriCodigo	Number	(3,0)	x	
IndTitulo	Varchar2	255		No

IndDescripcion	Varchar2	500		No
IndValoracion	Number	(5,2)		No
IndValorminimo	Number	(5,2)		No
TABLA: CALCULOINDICADOR				
CaInId	Number	(3,0)	x	No
ForId	Number	(3,0)	x	No
CaInDescripcion	Varchar2	500		No
CaInTipo	Number	(1,0)		No
TABLA: FORMA				
ForId	Number	(3,0)	x	No
ForDescripcion	Varchar2	500		No
TABLA: FACULTAD				
FacCodigo	Number	(3,0)	x	No
FacNombre	Varchar2	255		No
FacResponsable	Varchar2	255		No
TABLA: ESCUELA				
EscCodigo	Number	(3,0)	x	No
FacCodigo	Number	(3,0)	x	
EscNombre	Varchar2	100		No

EscResponsable	Varchar2	100		No
TABLA: CARRERA				
CarCodigo	Number	(3,0)	x	No
EscCodigo	Number	(3,0)	x	
FacCodigo	Number	(3,0)	x	
EscNombre	Varchar2	255		No
EscResponsable	Varchar2	255		No

6.9.2.2. Script de la Base de Datos

```
create or replace procedure sp_Insert_Criterios
(TxtDescripcion in varchar2,Valoracion in number)
is
begin
        insert into CRITERIO
                values(sec_criterio.NEXTVAL, TxtDescripcion , Valoracion);
end sp_Insert_Criterios;
create or replace procedure sp_Edit_Criterios
( cricodigo_v in out number,cridescripcion_v in out varchar2,crivaloracion_v in out number)
is
begin
        update CRITERIO
        set
cridescripcion          =cridescripcion_v,
        crivaloracion           =crivaloracion_v
                where(cricodigo = cricodigo_v);
end sp_Edit_Criterios;

create or replace procedure sp_Delete_Criterio(v_codcriterio in number)
is
begin
delete from criterio
where cricodigo = v_codcriterio;
end sp_Delete_Criterio;
```

SECUENCIAS

```
create sequence sec_criterio
increment by 1
start with 1;
create sequence sec_grupoindicador
increment by 1
start with 1;
```

ALTER TABLES REESTRUCTURACION

```
alter table USUARIO_PERMISO
add constraint  FK_user_permiuser foreign key (RolCodigo,UsuCedula)
        references USUARIO( RolCodigo,UsuCedula);
```

6.9.2.3. Implementación de la Base de Datos.

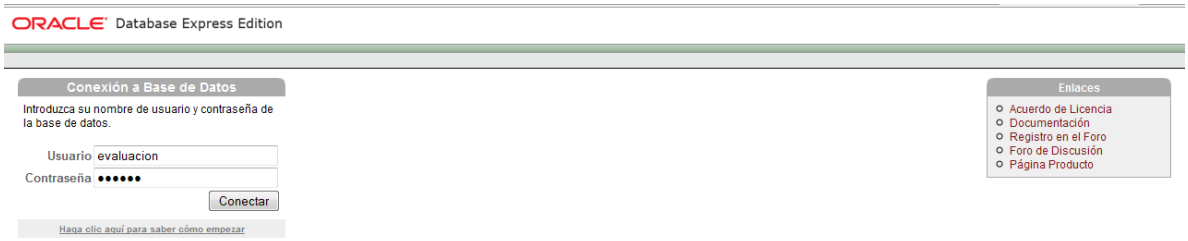


Figura VI.73. Base de Datos

Examinar Objetos de la Base de Datos

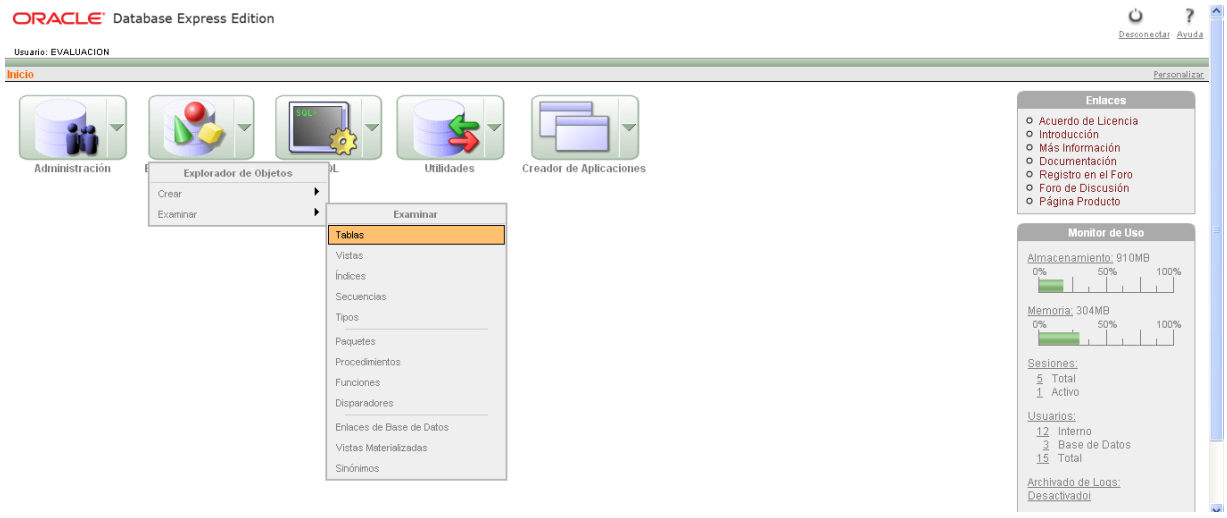


Figura VI.74. Examinar objetos de base de datos

Examinar todas las tablas de la base.

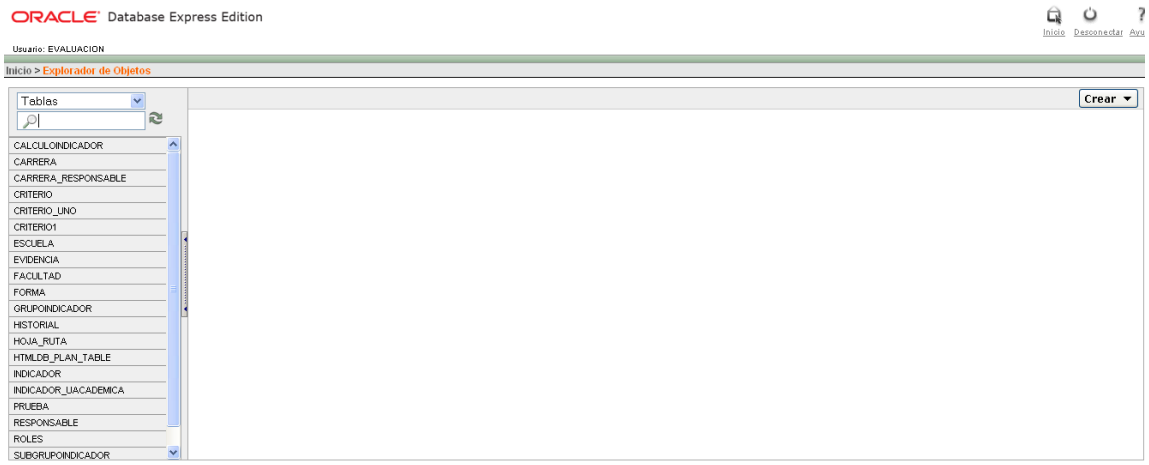


Figura VI.75. Tablas existentes en la DB

Estructura de una tabla.

Figura VI.76. Estructura de una tabla

Procedimientos almacenados

Figura VI.77. Procedimientos almacenados

6.10. Fase de Estabilización

Esta fase se encarga de validar la solución de la implementación. El desarrollo de la solución del proyecto ha sido completado y procederemos a realizar las pruebas.

6.10.1. Revisión General del Sistema

6.10.1.1. Código Fuente

El código fuente del Sistema de Evaluacion de Carreras de los podrá observar en el anexo 7

Documentación de Instalación

Ofrecemos los siguientes manuales para facilitar la utilización de la aplicación del sistema:

- Manual Técnico
- Manual de Usuario

6.11. Fase de Instalación

6.11.1. Tareas a realizar

La aplicación ha sido culminada y será entregada a la Comisión de Evaluación de la Escuela Superior Politécnica de Chimborazo.

Además se entregará la documentación técnica, manuales de usuario, con el propósito de facilitar la utilización del sistema.

CONCLUSIONES

1. El ORM JPA facilita la implementación de las clases de Acceso a Datos proporcionando la estructura básica para el funcionamiento de la capa Controlador y Vista.
2. JPA puede interactuar con varios proveedores de persistencia puesto que su arquitectura está diseñada para soportar varias formas de hacer persistencia con Frameworks dedicados a prestar estos tipos de servicios.
3. TopLink puede trabajar con varias unidades de persistencia los mismos que puede identificar a un grupo de clases, estas clases no pueden pertenecer a dos o más unidades de persistencia.
4. El proveedor de persistencia TopLink se encuentra implementado para trabajar en varios entornos de Desarrollo IDE como NetBeans, Eclipse, Oracle ADF.
5. El diseño del modelo de la Base de Datos puede ser relacional, Orientada a Objetos, archivos XML ya que de esto dependerá la forma en que se realice del mapeo de objetos, pasando de tablas a Clases y Objetos java.
6. El Framework de persistencia no admite el uso de procedimientos almacenados, ni sentencias SQL nativas Directas ya que maneja a través de código java el lenguaje SQL.
7. Para instanciar una clase a través de los métodos, el proveedor de persistencia realiza acciones dentro de una transacción para el aseguramiento de la acción.

8. JPA/TopLink para mantener la relación de las clases mapeadas, pasadas del modelo relacional crea una clase PK para mantener la relación el mismo que está vinculado con las clases participantes.
9. Del análisis realizado se puede concluir que el uso del Framework de persistencia "JPA/TopLink" reduce el nivel de vulnerabilidad muy significativo del 55.66% en el Desarrollo Tradicional al 49.33% con el uso de persistencia, y logrando cumplir el objetivo de obtener bajos niveles de vulnerabilidad.
10. El nivel de vulnerabilidad en el desarrollo con el uso de persistencia ha reducido en un 6.33% en la ejecución de ataques externos.
11. El uso del Framework de persistencia disminuye los tiempos de respuestas al realizar acciones que manipulan la información, teniendo un porcentaje del 17.50 % mediante el uso de persistencia y del 18.86% mediante la programación tradicional, proporcionando al usuario rapidez y seguridad en las operaciones.
12. La mejor forma de construir aplicaciones se basa en la integración de plataformas y tecnologías Heterogéneas combinando herramientas y metodologías de implementación.
13. El Framework de persistencia funciona en al menos dos Sistemas Operativos Diferentes como es Linux y Windows.

RECOMENDACIONES

1. Se recomienda el uso del Framework de persistencia TopLink mediante el uso del Sistema de Evaluación y Acreditación de Carreras de la FIE.
2. Hacer uso adecuado de los tipos de datos que son generados por el ORM JPA/TopLink para interactuar con la Base de Datos.
3. Para la construcción y el desarrollo de un sistema de Gestión se debe seguir un modelo o patrón de diseño que servirá de guía durante todas las fases de construcción la misma que permite construir software de calidad.
4. La técnica de mapeo y carga de la Base de Datos más común es la carga lenta (Lazy load**)** ya queproporciona optimización en el uso de recursos de memoria Caché de modo que el objeto instanciado queda temporalmente guardada en memoria Caché L1, L2, L3.
5. El manejo de transacciones en las acciones hacia la Base de datos debe ser manejada de forma automática por el contenedor EJB de JPA/TopLink ya que proporciona control equilibrado de acciones.
6. Una clase mapeada solo puede pertenecer a una unidad de persistencia, mientras que puede existir varias unidades de persistencia en un mismo proyecto.
7. En los sistemas de Gestión como un medio de seguridad se requiere de Autenticación e Identificación, así como la protección de los datos a nivel de Aplicación independiente de los protocolos de comunicación y de la seguridad del servidor web.
8. Los planes de aseguramiento de Datos garantizan la estabilidad y capacidad de reposición inmediata frente a fallos o borrados inesperados de la Base de Datos.

9. Mediante el uso del Framework de persistencia se implementa sistemas para evitar ataques tales como SQL Inyections.
10. El proveedor de persistencia utiliza su propio lenguaje de consultas, lo que hace que los usuarios dejen de utilizar las sentencias SQL Nativas.
11. Se sugiere el uso de la capa de persistencia ya que encapsula el comportamiento necesario para persistir objetos.
12. Se recomienda el uso de las Unidades de persistencia ya que este brinda los servicios de sincronización de la capa de lógica con el medio de almacenamiento.
13. Usar adecuadamente el archivo de configuración persistence.xml el mismo que define el contexto de persistencia de las clases persistentes instanciadas.
14. Utilizar el lenguaje SQL mediante el uso del JPQL ya que provee un lenguaje para consultas Orientadas a Objetos a través del código java.

RESUMEN

El Análisis de la Tecnología TopLink, como Framework de Persistencia aplicado a la Facultad de Informática y Electrónica de la ESPOCH a través del desarrollo del Sistema de Evaluación y Acreditación de Carreras de ESPOCH (SEACE).

La Investigación se realizó a través del método Analítico mediante comparaciones los mismos que se efectúo en igualdad de condiciones bajo la misma parametrizasión de las variables.

Para las pruebas del uso de persistencia y el no uso se utilizó un servidor web, auditoria de base de datos Oracle, Herramientas de terceros, como lenguaje de desarrollo JAVA y JSP, Netbeans como entorno de desarrollo, donde se seleccionó y categorizó parámetros de comparación como: trabajo con conexiones, librerías, métodos CRUD, líneas de código.

Los resultados obtenidos del análisis con la ayuda de técnica de ponderación asignándole un peso en la escala numérica de valoración de 10 y 5. Observando que en el no uso de persistencia tiene una porcentaje del 55.67% en el nivel de vulnerabilidad y con el uso de persistencia se obtuvo un porcentaje del 49.33% reduciendo con un porcentaje del 6.34% al utilizar el Framework de persistencia JPA/TopLink.

Mediante los resultados obtenidos se concluye que el uso del Framework de persistencia reduce el nivel de vulnerabilidad en la construcción aplicaciones web mejorando la seguridad de la aplicación manteniendo la integridad y disponibilidad de la información.

Se recomienda el uso de la persistencia ya que su uso disminuye las vulnerabilidades a ataques externos los mismos que garantizan un aseguramiento y robustez en la etapa de producción y utilización del Sistema web “SEACE”.

SUMMARY

TOPLINK TECHNOLOGY ANALISYS AS A PERSISTENCE FRAMEWORK APPLIED TO THE EVALUATION SYSTEM OF THE CARRIERS IN THE INFORMATICS AND ELECTRONICS FACULTY WITH THE PURPOSE OF ACREDDIT ATION.

It is important the persistence framework analysis even though the major of the informatic programs need to preserve data for later use. There is the need to automate the carriers evaluation process with the purpose of accreditation giving an informatic solution applying the Toplink technology reducing the vulnerability levels.

The objectives of this study were; To analyze the Toplink tool of Java based in ORM under an IDE with their features. To analyze the objects management, class, transactions and security as main parameters. To implement the evaluation system of the faculty careers with the purpose of accreditation using Toplink as a persistence Framework. MICROSOFT SOLUTION FRAMEWORK (MSF) persistence Framework is the approach used in this research; it is a model of processes combining two very common models in the development projects, the cascade model and the traditional model.

The results obtained in this research were; the no use of persistence has a percentage of 55.67% in the vulnerability level. And with the use of persistence with a percentage of 49.33% reducing the percentage to 6.34% using the persistence Framework JPA/Toplink. It is concluded that the persistence Framework use of reduces the level of vulnerability in the web page construction. It is recommended the usage of persistence decreasing the vulnerability and time responding in database operations.

Key words: Framework, persistence, vulnerability.

ANEXOS

Anexo 1

Sin Uso de Persistencia

Técnica: Auditoria de Base de datos										
OPERACION	**EQUIPO 1**		**EQUIPO 2**		**EQUIPO 3**		**EQUIPO 4**		**TOTAL**	**PROMEDIO**
	Calificador	**Valor**	**Calificador**	**Valor**	**Calificador**	**Valor**	**Calificador**	**Valor**		
Crear	Alta	10.16	Alta	10.16	Alta	10.35	Alta	10.34	41.01	10.25
Actualizar	Alta	10.17	Alta	10.17	Alta	10.35	Alta	10.35	41.04	10.26
Eliminar	Alta	10.06	Alta	10.05	Alta	10.30	Alta	10.33	40.74	10.18

Técnica: Auditoria de Base de datos					
Operación	**No**	**Calificador**	**Intervalo**	**Promedio**	**Valor Real**
Crear	4	Alta	7.31 - 11.00	9.91	49.90
Actualizar	4	Alta	7.31 - 11.00	9.92	51.38
Eliminar	4	Alta	7.31 - 11.00	9.91	51.32
Promedio General				**9.91**	50.83

Con uso de persistencia

Técnica: Auditoria de Base de datos					
Operación	**No**	**Calificador**	**Intervalo**	**Promedio**	**Valor Real**
Crear	4	Alta	7.31 - 11.00	10.25	49.90
Actualizar	4	Alta	7.31 - 11.00	10.26	51.38
Eliminar	4	Alta	7.31 - 11.00	10.18	51.32
Promedio General				**9.91**	50.83

Anexo 2

Parámetro: Librerías utilizadas

Definición JDBC

JDBC es un API (Application programming interface) que describe o define una librería estándar para acceso a fuentes de datos, principalmente orientado a Bases de Datos relacionales que usan SQL (Structured Query Language). JDBC no sólo provee un interfaz para acceso a motores de bases de datos, sino que también define una arquitectura estándar, para que los fabricantes puedan crear los drivers que permitan a las aplicaciones en java el acceso a los datos.

Sun define JDBC como: ***"The JDBC API is the industry standard for database-independent connectivity between the Java programming language and a wide range of databases."*** http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=introjdbc

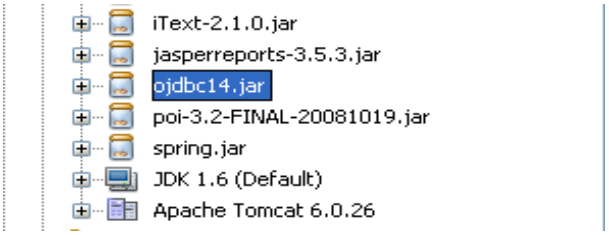


Figura: Librería ojdbc.jar

Fuente: SEACE

Interpretación

De acuerdo al estándar publicado por ***Microsoft System Sun*** el API jdbc está desarrollado para proveer conexiones desde aplicaciones JAVA independientemente del gestor de base de datos al que se desee conectar para lo cual se requiere de un driver específico.

Librerías para persistencia JPA/TopLink

En la tabla 2-1 de ***Oracle® TopLink Getting Started Guide 10g (10.1.3.1.0)*** encontramos la descripción de los archivos de Oracle TopLinkJAR instalados. Las que se utilizan para realizar conexiones persistentes en el sistema "SEACE" son:

JAR File	Contents
toplink-essentials.jar	This JAR contains TopLink Essentials, the open source JPA edition of TopLink. See "TopLink JPA" for more information, including how to obtain the source (toplink-essentials-src.zip).
toplink-essentials-agent.jar	This JAR contains the classes that TopLink uses to perform byte-code weaving on JPA entities to automatically enable features such as Value Holder indirection. You invoke toplink-essentials-agent.jar by adding -javaagent:toplink-essentials-agent.jar to your application or client JVM command line or by using the toplink.weaving TopLink JPA extension that you can define in a persistence.xml file. This JAR is optional and should never be placed on the classpath – only as part of -javaagent.

Tabla: JAR file usado para persistencia JPA/TopLink
Fuente: http://docs.oracle.com/cd/B32110_01/web.1013/b28217.pdf

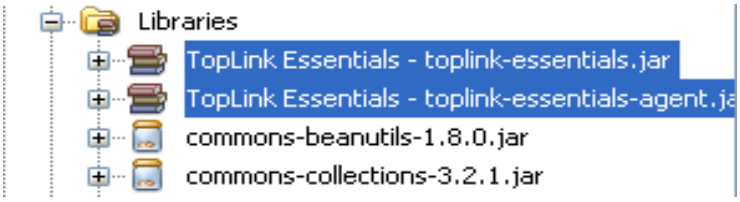


Figura. Librerías de TopLink.
Fuente: "SEACE"

Figura. TopLink Essentials
Fuente: http://mvnrepository.com/artifact/javax.persistence/toplink-essentials/1.0

Interpretación

Según la publicación de ***Oracle® TopLink Getting Started Guide 10g (10.1.3.1.0)*** hace referencia a las librerías necesarias para trabajar con persistencia en aplicaciones java, ya que los mismos están desarrolladas para trabajar con entidades de forma automática, para la cual se puede demostrar la utilización de las mismas

. **Anexo 3**

ALTERNATIVAS DE EVALUACIÓN			
N.	Calificador	Tiempo (miliseg.)	Intervalo
4	Alta	7.31 - 11.00	46-60
2	Mediana	6.31 – 7.30	31-45
3	Baja	5.01 – 6.30	16-30
1	Muy baja	0.10 – 5.00	1-15
Total			60

Indicador 2 Parámetro 2 : Métodos CRUD

Para evaluar este parámetro se han tomado dos archivos .jsp de similares características y realizamos las operaciones CRUD y tomamos el tiempo que se demora en realizar la acción a la base de datos.

Indicador: Operaciones CRUD

No Uso de Persistencia

Parámetro: Métodos CRUD

Técnica: Auditoria de Base de datos										
OPERACION	**EQUIPO 1**		**EQUIPO 2**		**EQUIPO 3**		**EQUIPO 4**		**TOTAL**	**PROMEDIO**
	Calificador	**Valor**	**Calificador**	**Valor**	**Calificador**	**Valor**	**Calificador**	**Valor**		
Crear	Mediana	06.34	Alta	10.37	Alta	10.43	Alta	10.46	36.60	9.15
Actualizar	Mediana	06.38	Alta	10.41	Alta	10.44	Alta	10.46	37.69	9.42
Eliminar	Mediana	06.37	Alta	10.37	Alta	10.44	Alta	10.46	37.64	9.41

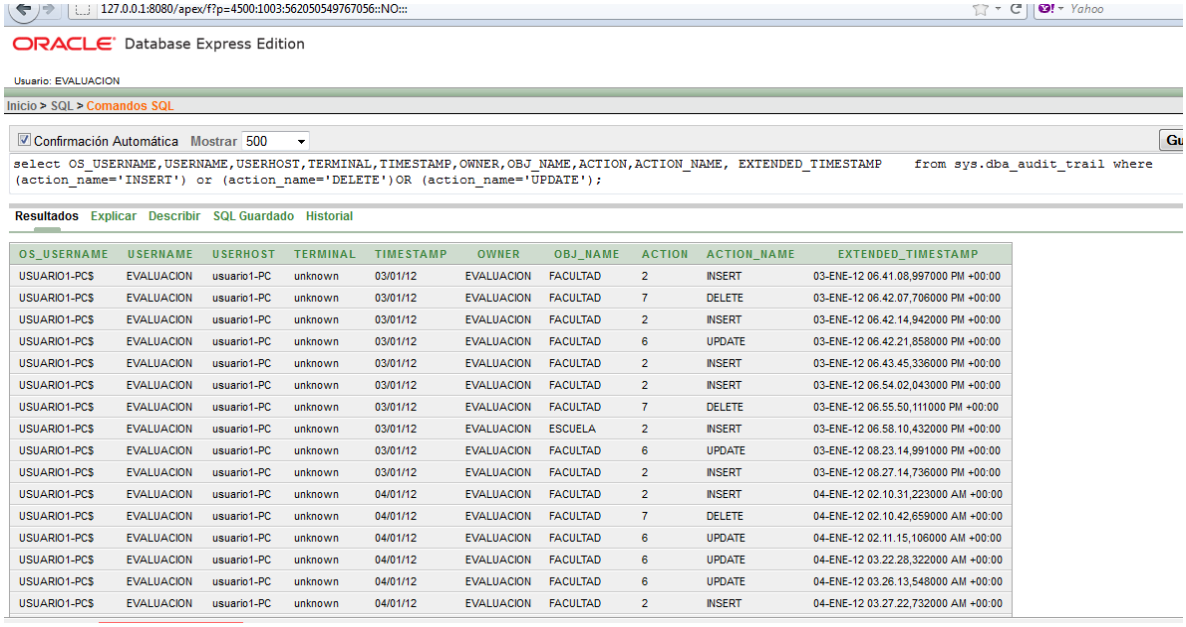
Técnica: Auditoria de Base de datos				
Operación	**No**	**Calificador**	**Intervalo**	**Promedio**
Crear	4	Alta	7.31 - 11.00	9.15
Actualizar	4	Alta	7.31 - 11.00	9.42
Eliminar	4	Alta	7.31 - 11.00	9.41
Promedio General				**9.32**

Uso de Persistencia

Parámetro: Métodos CRUD

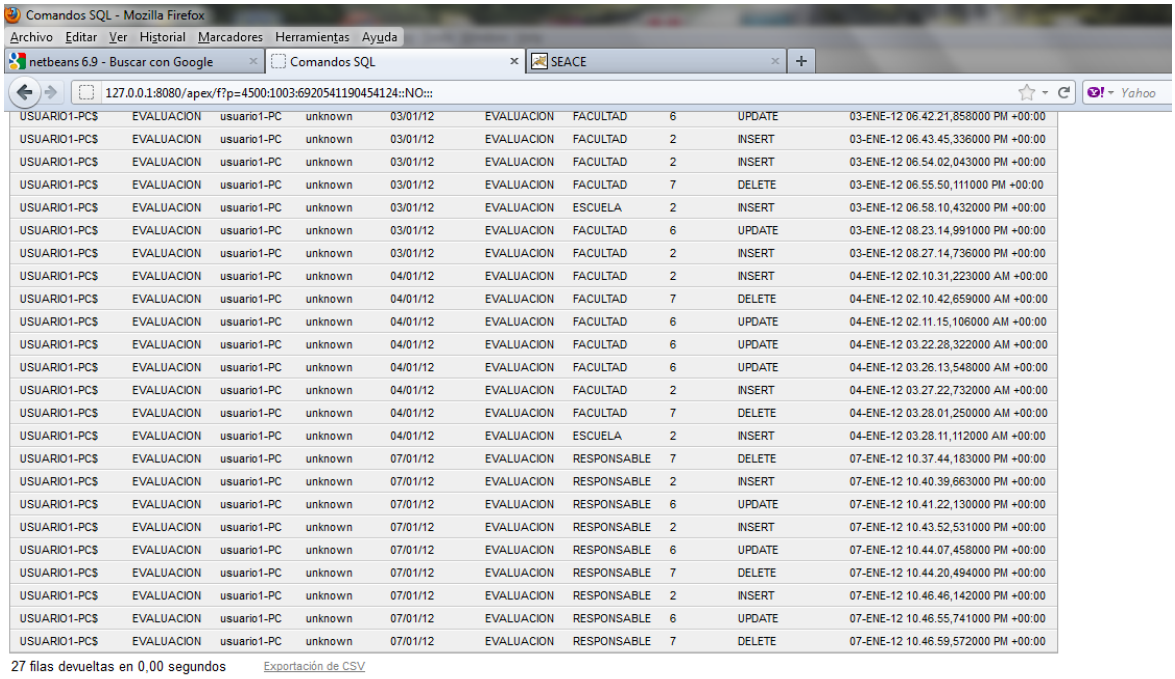
Técnica: Auditoria de Base de datos : Usuario										
OPERACION	**EQUIPO 1**		**EQUIPO 2**		**EQUIPO 3**		**EQUIPO 4**		**TOTAL**	**PROMEDIO**
	Calificador	**Valor**	**Calificador**	**Valor**	**Calificador**	**Valor**	**Calificador**	**Valor**		
Crear	Mediana	05.51	Alta	8.56	Alta	8.70	Alta	9.89	32.66	8.16
Actualizar	Alta	06.02	Alta	9.5	Alta	9.20	Alta	9.58	34.3	8.57
Eliminar	Alta	06.28	Alta	9.86	Alta	9.35	Alta	9.81	35.3	8.82

Técnica: Auditoria de Base de datos: Usuario				
Operación	**No**	**Calificador**	**Intervalo**	**Promedio Vulnerabilidad**
Crear	4	Alta	7.31 - 11.00	8.16
Actualizar	4	Alta	7.31 - 11.00	8.57
Eliminar	4	Alta	7.31 - 11.00	8.82
Promedio General				**8.51**



OS_USERNAME	USERNAME	USERHOST	TERMINAL	TIMESTAMP	OWNER	OBJ_NAME	ACTION	ACTION_NAME	EXTENDED_TIMESTAMP
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	03/01/12	EVALUACION	FACULTAD	2	INSERT	03-ENE-12 06.41.08,997000 PM +00:00
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	03/01/12	EVALUACION	FACULTAD	7	DELETE	03-ENE-12 06.42.07,706000 PM +00:00
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	03/01/12	EVALUACION	FACULTAD	2	INSERT	03-ENE-12 06.42.14,942000 PM +00:00
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	03/01/12	EVALUACION	FACULTAD	6	UPDATE	03-ENE-12 06.42.21,858000 PM +00:00
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	03/01/12	EVALUACION	FACULTAD	2	INSERT	03-ENE-12 06.43.45,336000 PM +00:00
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	03/01/12	EVALUACION	FACULTAD	2	INSERT	03-ENE-12 06.54.02,043000 PM +00:00
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	03/01/12	EVALUACION	FACULTAD	7	DELETE	03-ENE-12 06.55.50,111000 PM +00:00
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	03/01/12	EVALUACION	ESCUELA	2	INSERT	03-ENE-12 06.58.10,432000 PM +00:00
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	03/01/12	EVALUACION	FACULTAD	6	UPDATE	03-ENE-12 08.23.14,991000 PM +00:00
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	03/01/12	EVALUACION	FACULTAD	2	INSERT	03-ENE-12 08.27.14,736000 PM +00:00
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	04/01/12	EVALUACION	FACULTAD	2	INSERT	04-ENE-12 02.10.31,223000 AM +00:00
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	04/01/12	EVALUACION	FACULTAD	7	DELETE	04-ENE-12 02.10.42,659000 AM +00:00
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	04/01/12	EVALUACION	FACULTAD	6	UPDATE	04-ENE-12 02.11.15,106000 AM +00:00
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	04/01/12	EVALUACION	FACULTAD	6	UPDATE	04-ENE-12 03.22.28,322000 AM +00:00
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	04/01/12	EVALUACION	FACULTAD	6	UPDATE	04-ENE-12 03.26.13,548000 AM +00:00
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	04/01/12	EVALUACION	FACULTAD	2	INSERT	04-ENE-12 03.27.22,732000 AM +00:00

Figura: Captura de tiempos al realizar Operaciones CRUD



USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	03/01/12	EVALUACION	FACULTAD	6	UPDATE	03-ENE-12 06.42.21,858000 PM +00:00
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	03/01/12	EVALUACION	FACULTAD	2	INSERT	03-ENE-12 06.43.45,336000 PM +00:00
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	03/01/12	EVALUACION	FACULTAD	2	INSERT	03-ENE-12 06.54.02,043000 PM +00:00
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	03/01/12	EVALUACION	FACULTAD	7	DELETE	03-ENE-12 06.55.50,111000 PM +00:00
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	03/01/12	EVALUACION	ESCUELA	2	INSERT	03-ENE-12 06.58.10,432000 PM +00:00
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	03/01/12	EVALUACION	FACULTAD	6	UPDATE	03-ENE-12 08.23.14,991000 PM +00:00
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	03/01/12	EVALUACION	FACULTAD	2	INSERT	03-ENE-12 08.27.14,736000 PM +00:00
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	04/01/12	EVALUACION	FACULTAD	2	INSERT	04-ENE-12 02.10.31,223000 AM +00:00
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	04/01/12	EVALUACION	FACULTAD	7	DELETE	04-ENE-12 02.10.42,659000 AM +00:00
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	04/01/12	EVALUACION	FACULTAD	6	UPDATE	04-ENE-12 02.11.15,106000 AM +00:00
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	04/01/12	EVALUACION	FACULTAD	6	UPDATE	04-ENE-12 03.22.28,322000 AM +00:00
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	04/01/12	EVALUACION	FACULTAD	6	UPDATE	04-ENE-12 03.26.13,548000 AM +00:00
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	04/01/12	EVALUACION	FACULTAD	2	INSERT	04-ENE-12 03.27.22,732000 AM +00:00
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	04/01/12	EVALUACION	FACULTAD	7	DELETE	04-ENE-12 03.28.01,250000 AM +00:00
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	04/01/12	EVALUACION	ESCUELA	2	INSERT	04-ENE-12 03.28.11,112000 AM +00:00
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	07/01/12	EVALUACION	RESPONSABLE	7	DELETE	07-ENE-12 10.37.44,183000 PM +00:00
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	07/01/12	EVALUACION	RESPONSABLE	2	INSERT	07-ENE-12 10.40.39,663000 PM +00:00
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	07/01/12	EVALUACION	RESPONSABLE	6	UPDATE	07-ENE-12 10.41.22,130000 PM +00:00
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	07/01/12	EVALUACION	RESPONSABLE	2	INSERT	07-ENE-12 10.43.52,531000 PM +00:00
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	07/01/12	EVALUACION	RESPONSABLE	6	UPDATE	07-ENE-12 10.44.07,458000 PM +00:00
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	07/01/12	EVALUACION	RESPONSABLE	7	DELETE	07-ENE-12 10.44.20,494000 PM +00:00
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	07/01/12	EVALUACION	RESPONSABLE	2	INSERT	07-ENE-12 10.46.46,142000 PM +00:00
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	07/01/12	EVALUACION	RESPONSABLE	6	UPDATE	07-ENE-12 10.46.55,741000 PM +00:00
USUARIO1-PC$	EVALUACION	usuario1-PC	unknown	07/01/12	EVALUACION	RESPONSABLE	7	DELETE	07-ENE-12 10.46.59,572000 PM +00:00

27 filas devueltas en 0,00 segundos Exportación de CSV

Figura: Captura de tiempos al realizar Operaciones CRUD

Anexo 4

Indicador 2 Parámetro 2: Líneas de Código

Tabla: Alternativas de evaluación para indicador 1 variable independiente, parámetro 3

ALTERNATIVAS DE EVALUACIÓN		
N|°	**Calificador**	**Líneas de Código**
1	Alta	501 - 600
2	Mediana	401 –500
3	Baja	301 – 400
4	Muy baja	200 – 300

Para analizar este parámetro se ha utilizado el software ***Practiline Source Code Line Counter 1.1*** que nos permite contar en número de líneas de código que posee cada página java. Se ha elegido dos tablas que tienen las mismas características par hacer el conteo. De acuerdo a los resultados generados por este software se ha obtenido la siguiente tabla.

Parámetro: Líneas de Código

Sin Persistencia

Técnica: Mapeo de archivos .java					
Archivo Java	**LCD Fuente**	**LCD Comentario**	**LCD Blanco**	**LCD Mezcladas**	**LCD Total**
Responsable	214	0	24	0	238

Archivo Java	Calificador	Valor
Responsable	Muy Baja	214

Con Persistencia

Técnica: Mapeo de archivos .java						
Archivo Java	**Calificador**	**LCD Fuente**	**LCD Comentario**	**LCD Blanco**	**LCD Mezcladas**	**LCD Total**
Usuario – Usuario PK	Alta	537	22	57	0	616

Archivo Java	Calificador	Valor
Usuario – Usuario PK	Alta	537

Se tomo en cuenta Usuario y Responsable ya que poseen las mismas características.

Número de líneas de Código: Con Persistencia - Usuario

Tabla: Total líneas de Código Con Persistencia

File Name	Nominal Lines	Source Code Lines	Source Code Lines (%)	Comment Lines	Comment Lines (%)	Blank Lines	Blank Lines (%)	Mixed Lines	Mixed Lines (%)	Total Lines
G:\TESIS VERSION 11.0.0\core\src\java\Persistencia\Usuario.java	536	472	88,06 %	20	3,73 %	44	8,21 %	0	0,00 %	536
G:\TESIS VERSION 11.0.0\core\src\java\Persistencia\UsuarioPK.java	80	65	81,25 %	2	2,50 %	13	16,25 %	0	0,00 %	80
Total:	**616**	**537**	**87,18 %**	**22**	**3,57 %**	**57**	**9,25 %**	**0**	**0,00 %**	**616**

Número de líneas de Código: Sin Persistencia - Respomsable

Tabla: Total líneas de Código Sin Persistencia

File Name	Nominal Lines	Source Code Lines	Source Code Lines (%)	Comment Lines	Comment Lines (%)	Blank Lines	Blank Lines (%)	Mixed Lines	Mixed Lines (%)	Total Lines
G:\TESIS VERSION 11.0.0\core\src\java\Entidades\cResponsable.java	238	214	89,92 %	0	0,00 %	24	10,08 %	0	0,00 %	238
Total:	**238**	**214**	**89,92 %**	**0**	**0,00 %**	**24**	**10,08 %**	**0**	**0,00 %**	**238**

Mínimo de LCD Con persistencia

Tabla: Mínimo Líneas de Código Con Persistencia

File Name	Nominal Lines	Source Code Lines	Source Code Lines (%)	Comment Lines	Comment Lines (%)	Blank Lines	Blank Lines (%)	Mixed Lines	Mixed Lines (%)	Total Lines
G:\TESIS VERSION 11.0.0\core\src\java\Persistencia\Roles.java	334	274	82,04 %	20	5,99 %	40	11,98 %	0	0,00 %	334
Total:	**334**	**274**	**82,04 %**	**20**	**5,99 %**	**40**	**11,98 %**	**0**	**0,00 %**	**334**

Mínimo de LCD Sin persistencia

Tabla: Mínimo Líneas de Código Sin Persistencia

File Name	Nominal Lines	Source Code Lines	Source Code Lines (%)	Comment Lines	Comment Lines (%)	Blank Lines	Blank Lines (%)	Mixed Lines	Mixed Lines (%)	Total Lines
G:\TESIS VERSION 11.0.0\core\src\java\Entidades\cCriterio.java	258	222	86,05 %	4	1,55 %	32	12,40 %	0	0,00 %	258
Total:	**258**	**222**	**86,05 %**	**4**	**1,55 %**	**32**	**12,40 %**	**0**	**0,00 %**	**258**

Anexo 5:

Indicador 2 Parámetro 3: Procedimientos Almacenados

Para este indicador se tomaron dos tablas con similares características, la primera sin persistencia (Responsable) la segunda con persistencia (Usuarios)

Responsable:

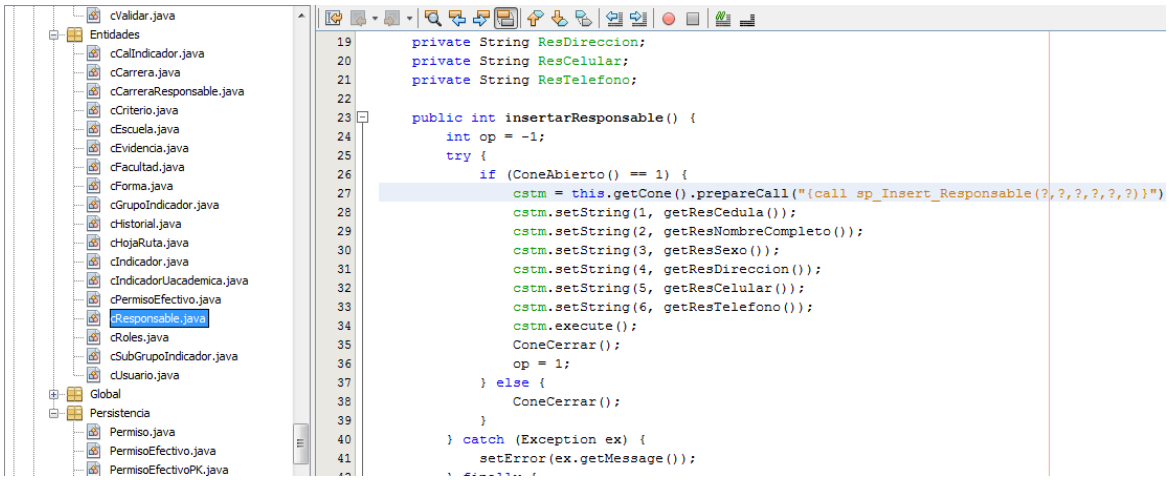


Figura: Llamada al procedimiento almacenado Insert Sin persistencia

En la figura de arriba nos muestra la manera como se realiza una operación de inserción de un responsable en la base de datos Sin persistencia. Se realiza la llamada al Procedimiento almacenado ***sp_Insert_Responsable***

Usuario:

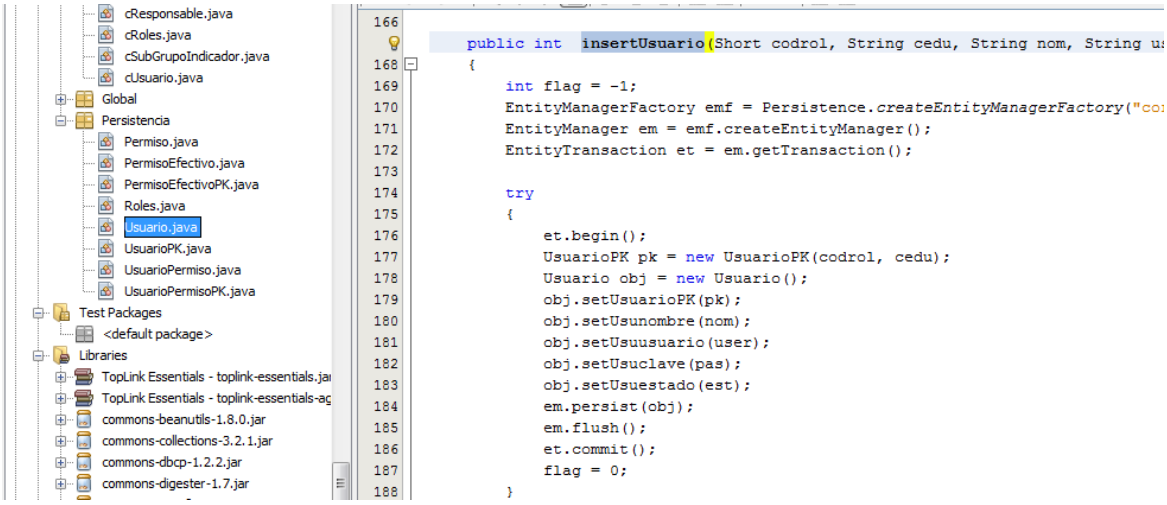


Figura: Insertar un nuevo Usuario con persistencia

En la figura anterior nos muestra la manera como se realiza una operación de inserción de un Usuario en la base de datos Con persistencia, en este caso no se hace llamadas a ningún procedimiento almacenado.

La manera como se realizan las operaciones en ambos casos son diferentes ya que al no utilizar persistencia no se utiliza ningún procedimiento almacenado y además se invocan librerías adicionales.

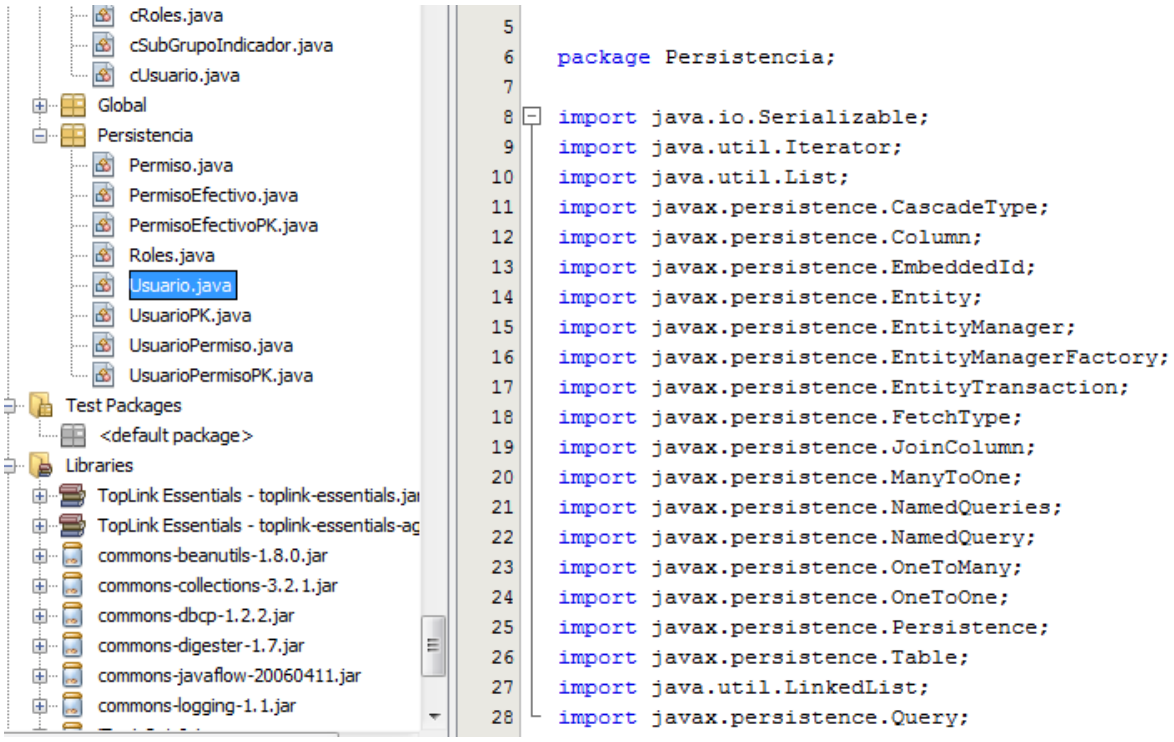


Figura: Librerías utilizadas con Persistencia

Anexo 6

Variable Dependiente

Tabla Alternativa de Evaluación variable dependiente

N.	ALTERNATIVAS DE EVALUACIÓN	
4	Alta	151-200
3	Mediana	101-150
2	Baja	51-100
1	Muy baja	1-50

Indicador: Seguridad de la Información

No uso de Persistencia

Parámetro: Integridad

Tabla: Recopilación de valores por parámetro Integridad variable dependiente

Técnica: SQL Inyection										
OPERACION	EQUIPO 1		EQUIPO 2		EQUIPO 3		EQUIPO 4		TOTAL	PROMEDIO
	Calificador	Valor	Calificador	Valor	Calificador	Valor	Calificador	Valor		
Crear	Baja	70	Baja	68	Baja	69	Baja	71	278	70
Actualizar	Baja	84	Baja	82	Baja	85	Baja	85	336	84
Eliminar	Baja	84	Baja	83	Baja	87	Baja	86	340	85
Listar	Baja	70	Baja	71	Baja	69	Baja	71	281	70

Parámetro: Confidencialidad

Tabla: Recopilación de valores por parámetro Confidencialidad variable dependiente

Técnica: Manipulación de Parámetros (Boolean SQL)										
OPERACION	**EQUIPO 1**		**EQUIPO 2**		**EQUIPO 3**		**EQUIPO 4**		**TOTAL**	**PROMEDIO**
	Calificador	**Valor**	**Calificador**	**Valor**	**Calificador**	**Valor**	**Calificador**	**Valor**		
Crear	Baja	80	Baja	80	Baja	79	Baja	81	320	80
Actualizar	Baja	96	Baja	95	Baja	93	Baja	97	381	95
Eliminar	Baja	96	Mediana	110	Mediana	105	Mediana	109	420	105
Listar	Baja	80	Baja	79	Baja	80	Baja	82	321	80

Parámetro: Disponibilidad

Tabla: Recopilación de valores por parámetro disponibilidad variable dependiente

Técnica: Ejecución de comandos (Cross-Site Scripting XSS)										
OPERACION	**EQUIPO 1**		**EQUIPO 2**		**EQUIPO 3**		**EQUIPO 4**		**TOTAL**	**PROMEDIO**
	Calificador	**Valor**	**Calificador**	**Valor**	**Calificador**	**Valor**	**Calificador**	**Valor**		
Crear	Alta	159	Alta	157	Alta	158	Alta	160	634	159
Actualizar	Alta	174	Alta	175	Alta	174	Alta	173	696	174
Eliminar	Alta	174	Alta	174	Alta	175	Alta	174	697	174
Listar	Alta	159	Alta	157	Alta	159	Alta	160	635	159

Uso de Persistencia

Parámetro: Integridad

Tabla: Recopilación de valores por parámetro Integridad variable dependiente

Técnica: SQL Inyection										
OPERACION	**EQUIPO 1**		**EQUIPO 2**		**EQUIPO 3**		**EQUIPO 4**		**TOTAL**	**PROMEDIO**
	Calificador	**Valor**	**Calificador**	**Valor**	**Calificador**	**Valor**	**Calificador**	**Valor**		
Crear	Baja	70	Baja	71	Baja	69	Baja	71	281	70
Actualizar	Baja	98	Baja	97	Baja	99	Baja	98	392	98
Eliminar	Muy baja	26	Muy baja	29	Muy baja	26	Muy baja	27	108	27
Listar	Baja	70	Baja	69	Baja	70	Baja	71	280	70

Parámetro: Confidencialidad

Tabla: Recopilación de valores por parámetro Confidencialidad variable dependiente

Técnica: Manipulación de Parámetros (Boolean SQL)										
OPERACION	**EQUIPO 1**		**EQUIPO 2**		**EQUIPO 3**		**EQUIPO 4**		**TOTAL**	**PROMEDIO**
	Calificador	**Valor**	**Calificador**	**Valor**	**Calificador**	**Valor**	**Calificador**	**Valor**		
Crear	Baja	80	Baja	81	Baja	80	Baja	80	321	80
Actualizar	Mediana	112	Mediana	109	Mediana	105	Mediana	109	435	109
Eliminar	Muy baja	35	Muy baja	35	Muy baja	36	Muy baja	35	141	35
Listar	Baja	80	Baja	79	Baja	82	Baja	81	322	80

Parámetro: Disponibilidad

Tabla: Recopilación de valores por parámetro disponibilidad variable dependiente

Técnica: Ejecución de comandos (Cross-Site Scripting XSS)										
OPERACION	**EQUIPO 1**		**EQUIPO 2**		**EQUIPO 3**		**EQUIPO 4**		**TOTAL**	**PROMEDIO**
	Calificador	**Valor**	**Calificador**	**Valor**	**Calificador**	**Valor**	**Calificador**	**Valor**		
Crear	Alta	159	Alta	158	Alta	160	Alta	160	637	159
Actualizar	Alta	189	Alta	190	Alta	189	Alta	190	758	190
Eliminar	Mediana	109	Mediana	107	Mediana	110	Mediana	109	435	109
Listar	Alta	159	Alta	160	Alta	159	Alta	159	637	159

PROCESO DE RECOPILACIÓN DE DATOS

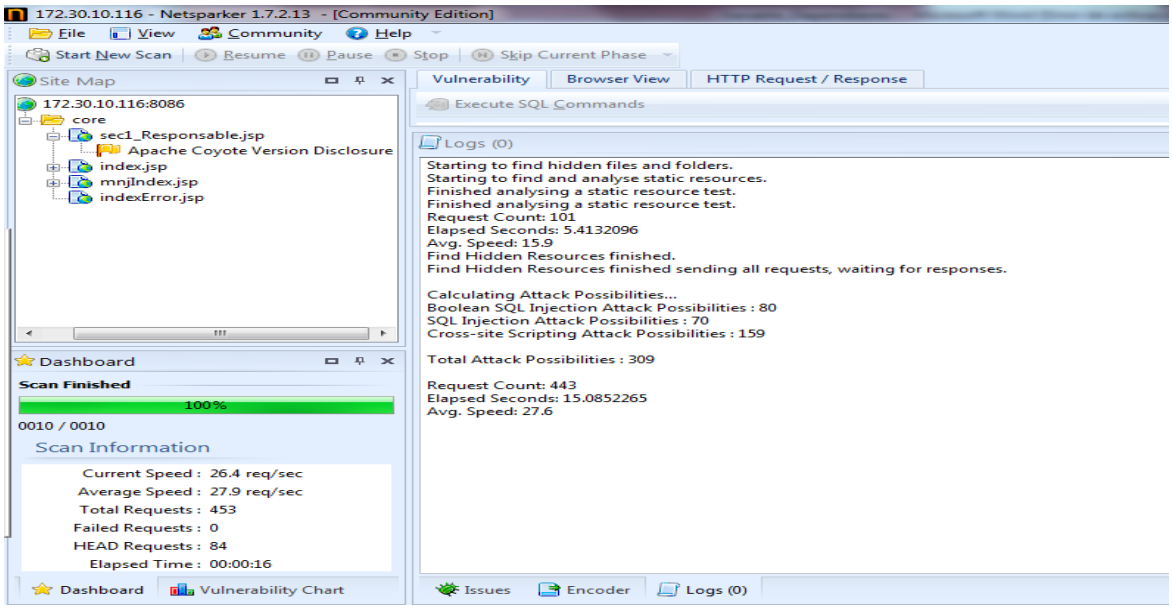
MODULO RESPONSABLE

No uso de Persistencia

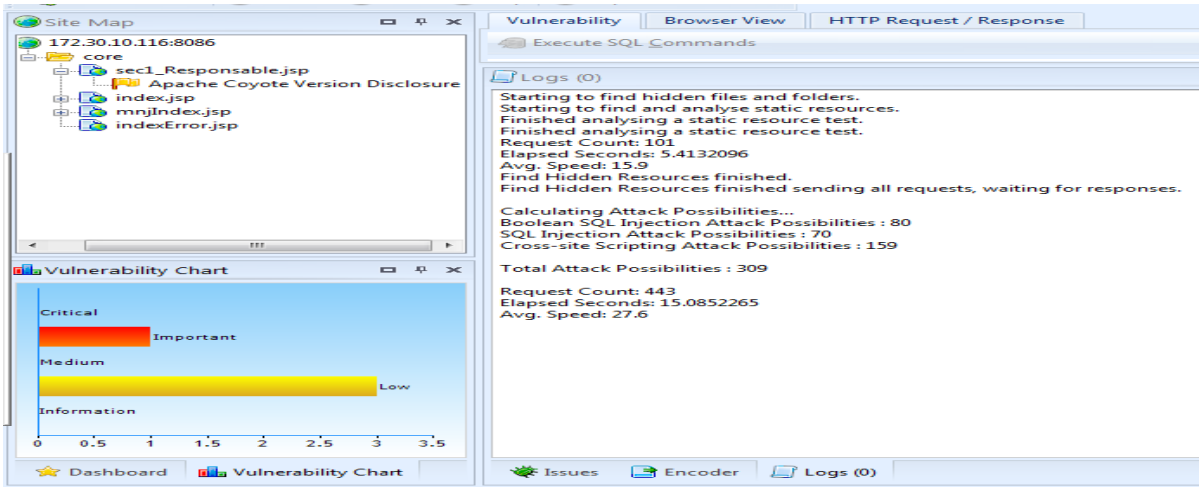
Descripción de proceso en el Equipo 1

Para capturar las salidas de Netsparker ingresamos la URL:**172.30.10.116:8086/core/sec1_Responsable.jsp**

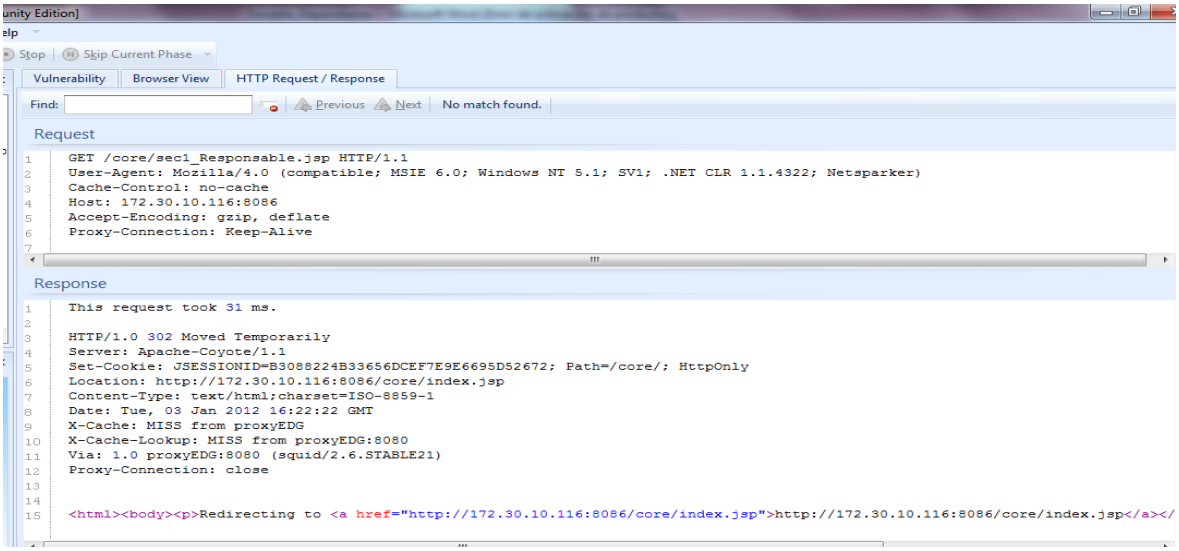
Seleccionamos el ítem **logs**, las mismas que proporcionan datos de las diferentes técnicas a evaluar.

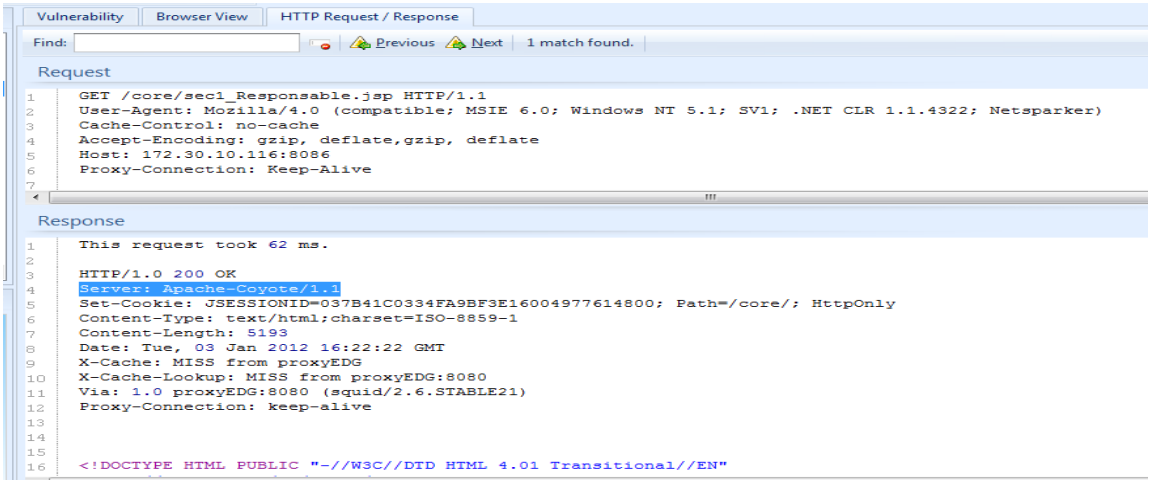


En la siguiente imagen se puede ver los gráficos estadísticos que muestran el nivel de vulnerabilidad.

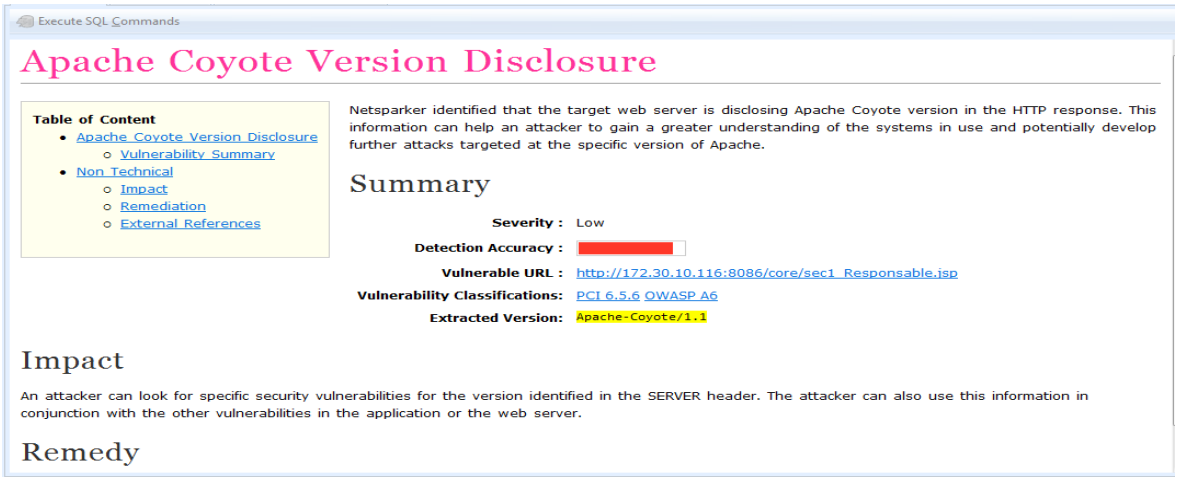


En esta imagen se visualiza la cabecera y la codificación utilizada al realizar la operación.



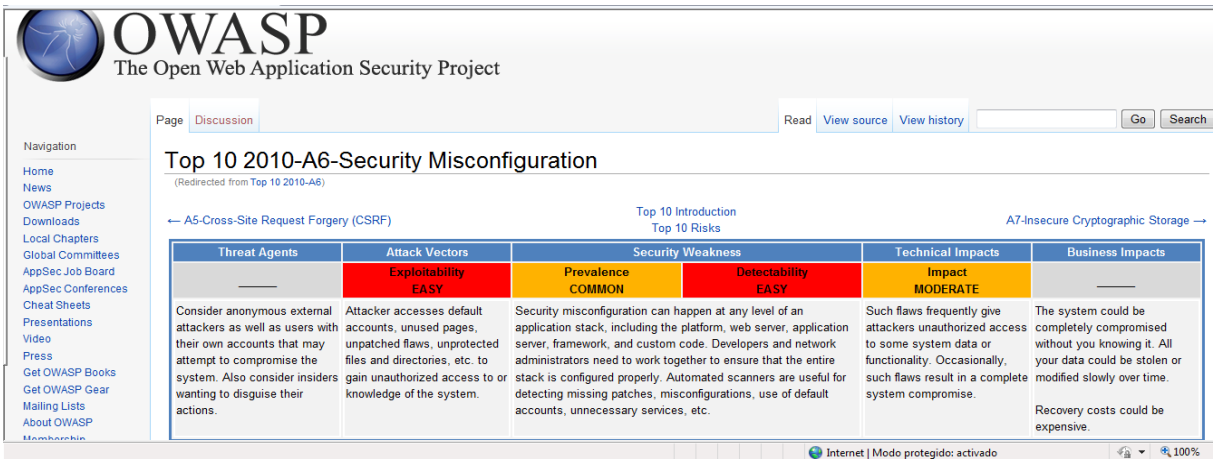


A continuación en el visor de salida de NetsparKer despliega un resumen el nivel de vulnerabilidad así como la URL analizada a profundidad toda su construcción.



Al presionar sobre el enlace de la clasificación de la vulnerabilidad desplegada por Netsparker.

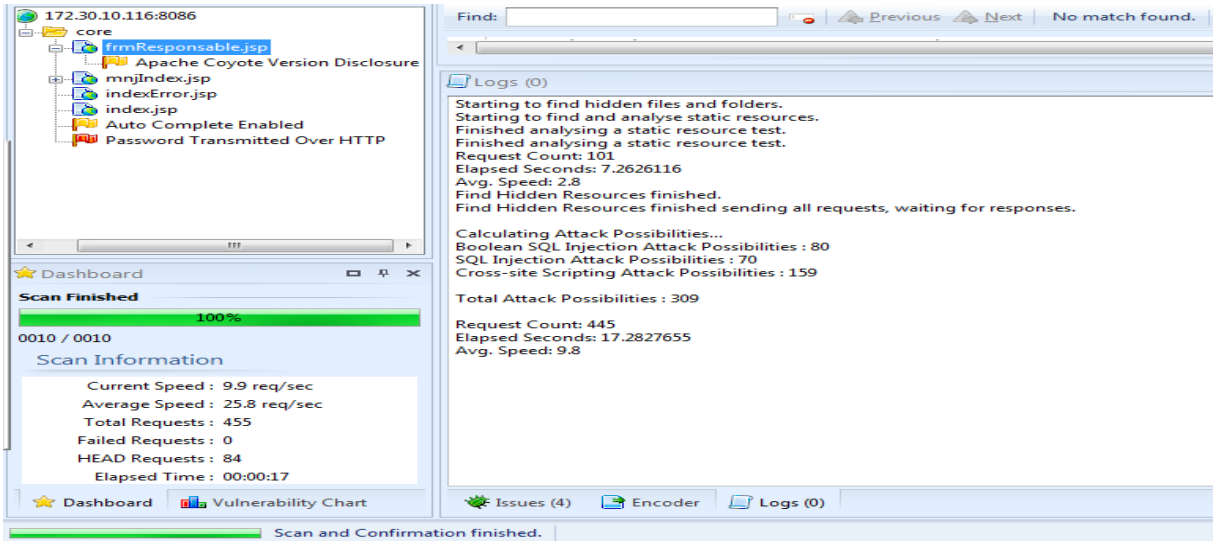
Llama a un sitio oficial de **OWASP** donde me indica información adicional que permite analizar a un más la vulnerabilidad.



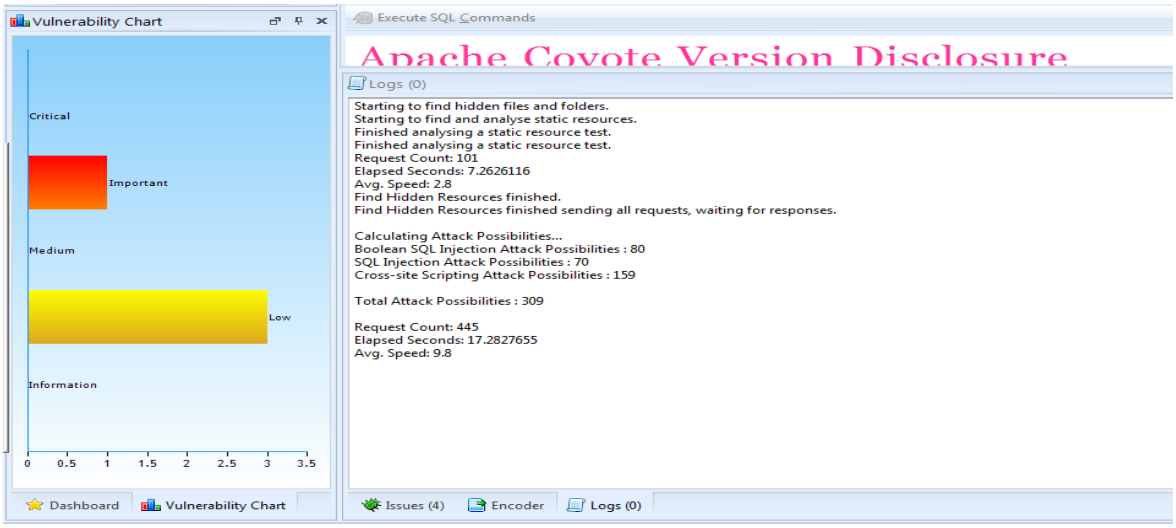
Fuente: https://www.owasp.org/index.php/Top_10_2010-A6

Operación Insertar mediante el formulario **frmResponsable**

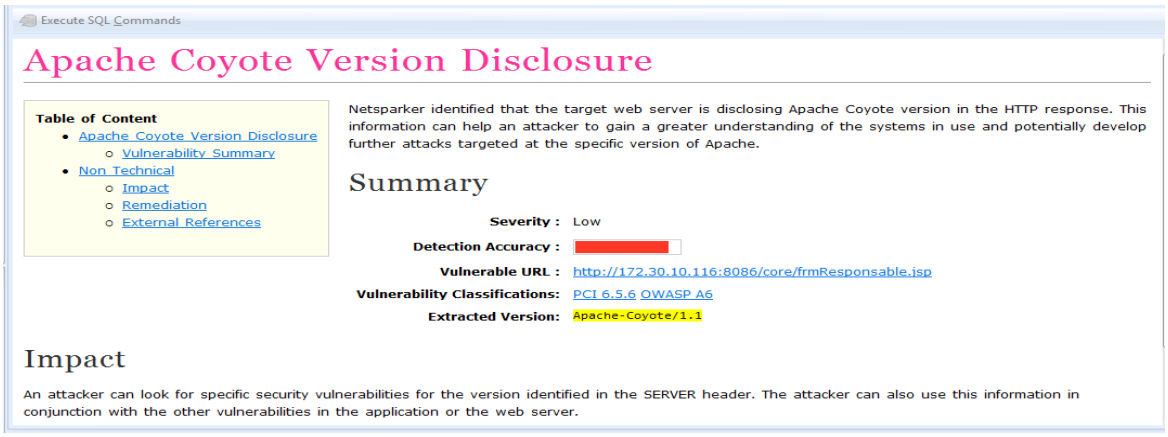
En esta imagen visualizamos la salida a través de **logs** que me proporciona datos sobre las diferentes vulnerabilidades de este formulario y toda su construcción.



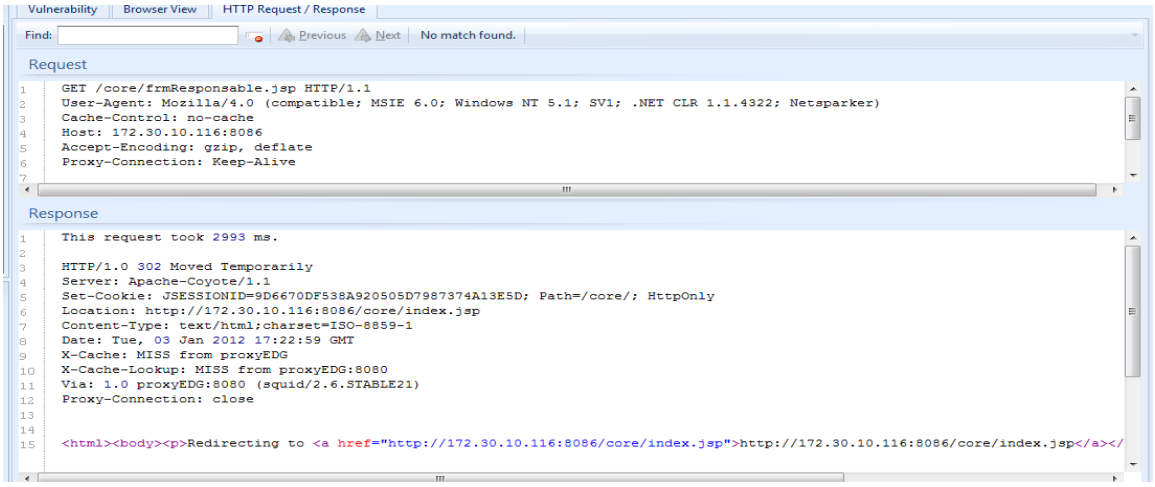
A continuación se visualiza graficas estadisticas sobre el nivel de vulnerabilidad de la URL asignada al analizador Netsparker.



En la siguiente imagen se puede ver el resumen de la vulnerabilidad proporcionada por el analizador el mismo que nos indica que el nivel es bajo



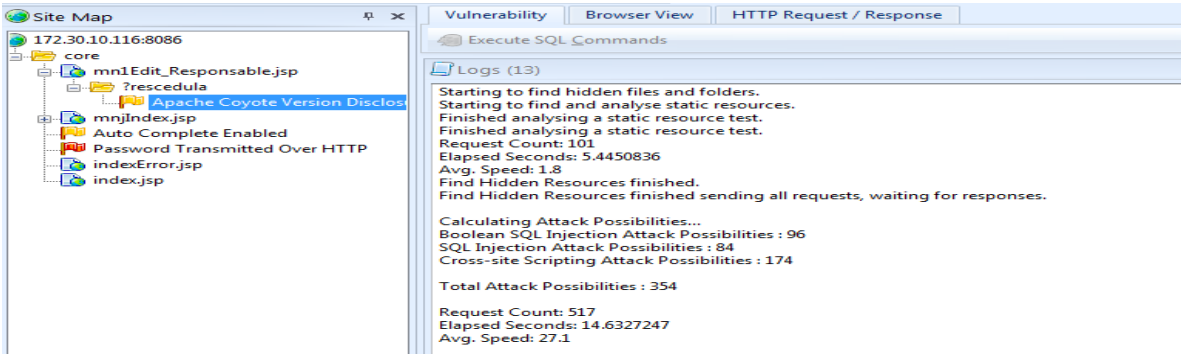
A continuación se analiza la cabecera y codificación que usa al realizar la operación



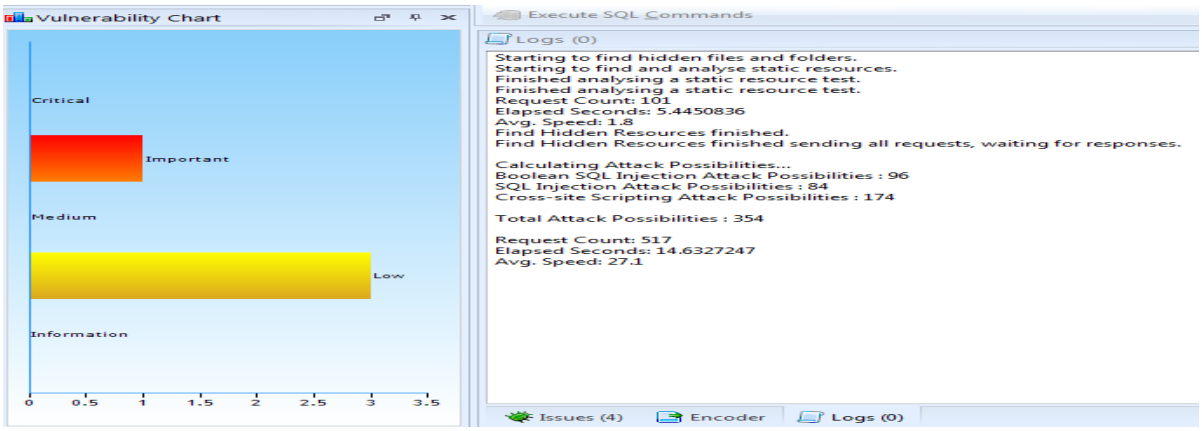
En esta imagen se despliega los puntos más vulnerables encontrados durante el análisis, los mismos que se puede diferenciar por su color.

Operación Actualizar mediante el formulario **mn1Edit_Responsable**

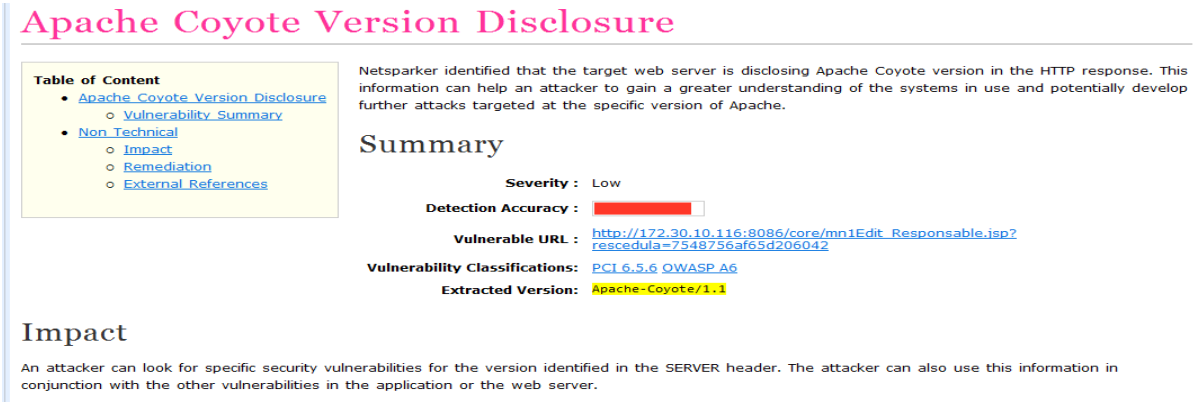
En esta imagen visualizamos la salida a través de **logs** que me proporciona datos sobre las diferentes vulnerabilidades de este formulario y toda su construcción.



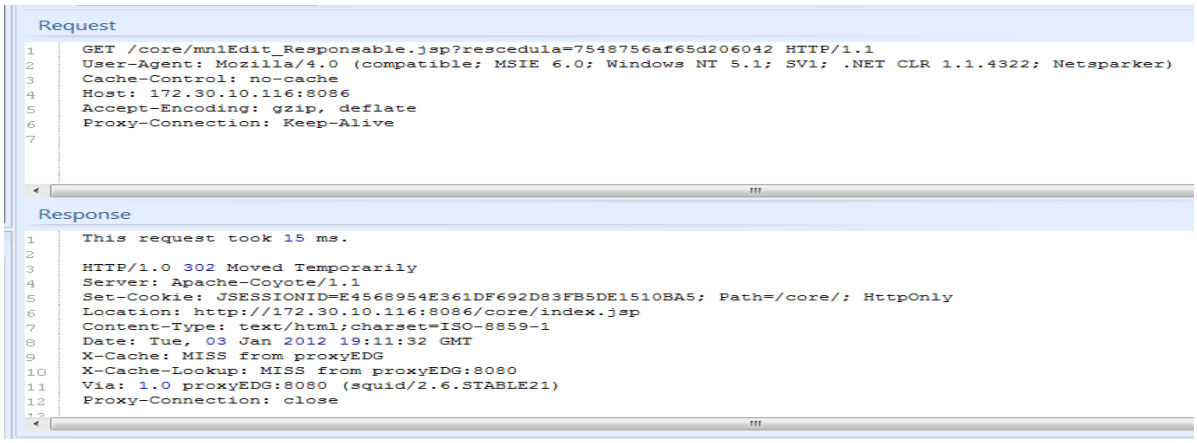
A continuación se visualiza graficas estadisticas sobre el nivel de vulnerabilidad de la URL asignada al analizador Netsparker.



En la siguiente imagen se puede ver el resumen de la vulnerabilidad proporcionada por el analizador el mismo que nos indica que el nivel es bajo

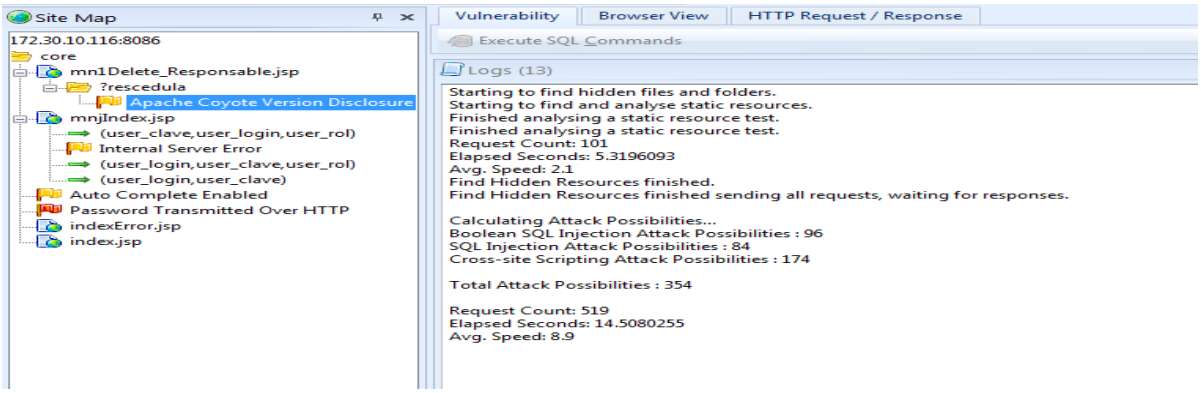


En esta imagen se visualiza la cabecera y la codificación utilizada al realizar la operación.

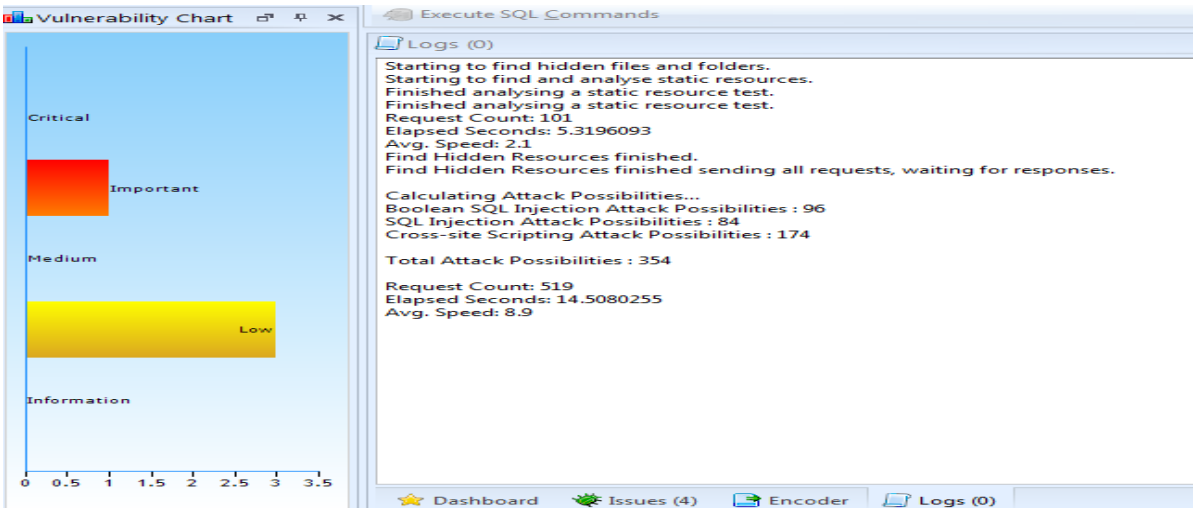


Operación Eliminar mediante el formulario **mn1Delete_Responsable**

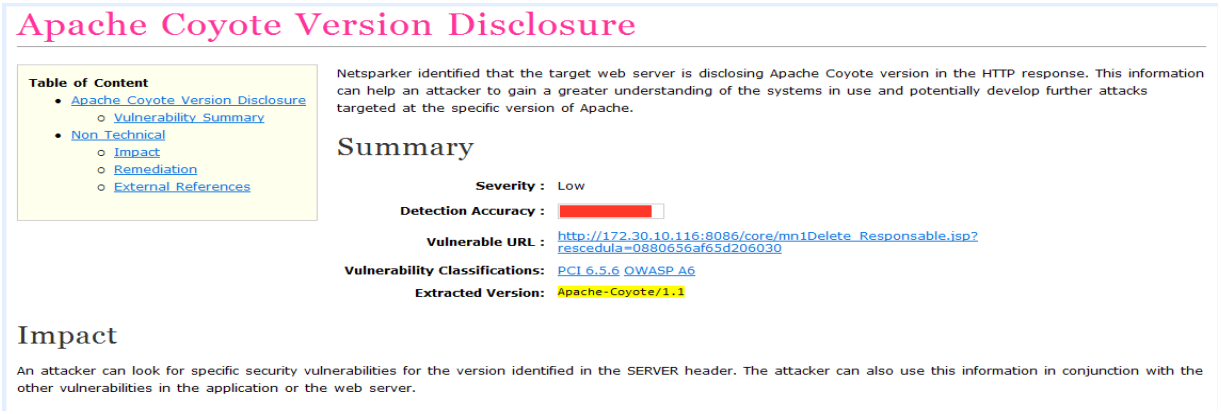
En esta imagen visualizamos la salida a través de **logs** que me proporciona datos sobre las diferentes vulnerabilidades de este formulario y toda su construcción.



A continuación se visualiza graficas estadisticas sobre el nivel de vulnerabilidad de la URL asignada al analizador Netsparker.



En la siguiente imagen se puede ver el resumen de la vulnerabilidad proporcionada por el analizador el mismo que nos indica que el nivel es bajo



MODULO USUARIO

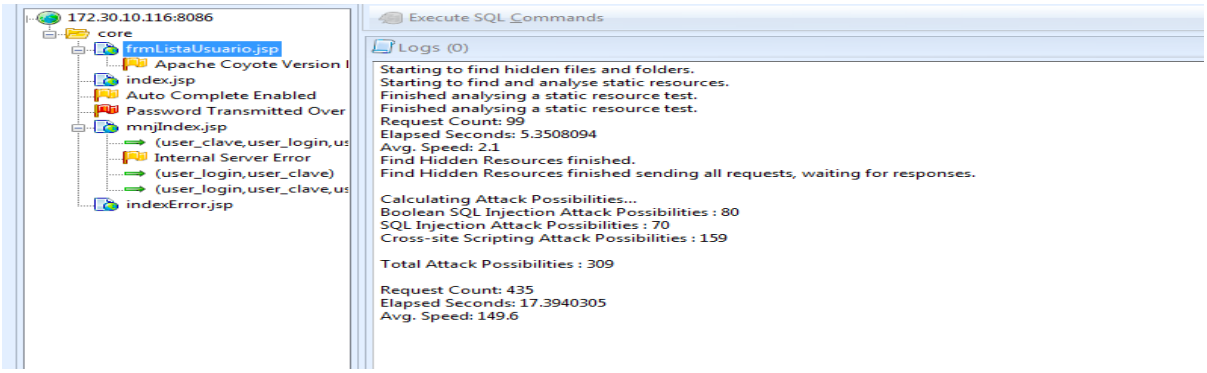
Uso de Persistencia

Descripción de proceso en el Equipo 1

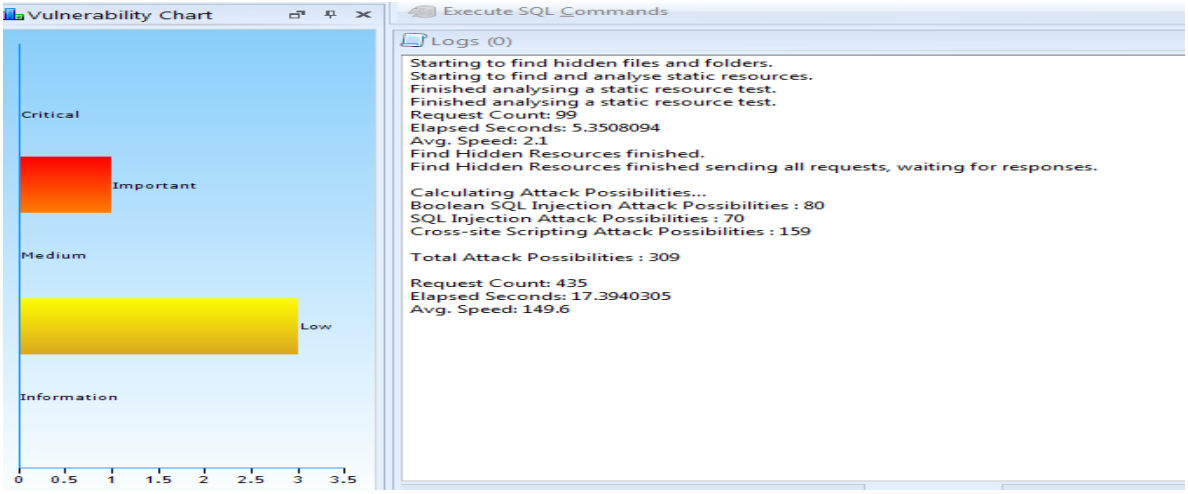
Para capturar las salidas de Netsparker ingresamos laURL:

172.30.10.116:8086/core/frmListaUsuario.jsp

Seleccionamos el ítem **logs**, las mismas que proporcionan datos de las diferentes técnicas a evaluar.

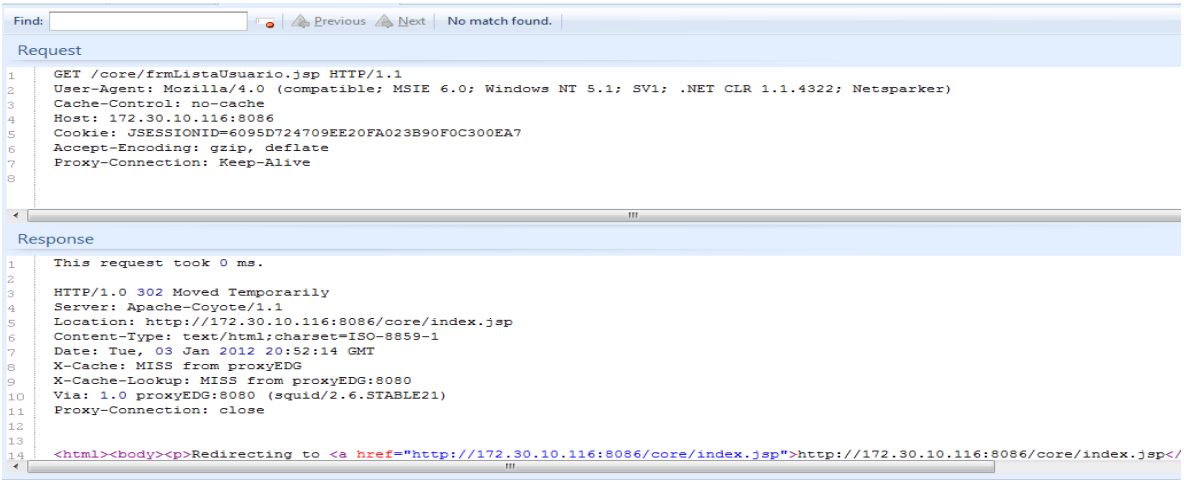


A continuación se visualiza graficas estadisticas sobre el nivel de vulnerabilidad de la URL asignada al analizador Netsparker.



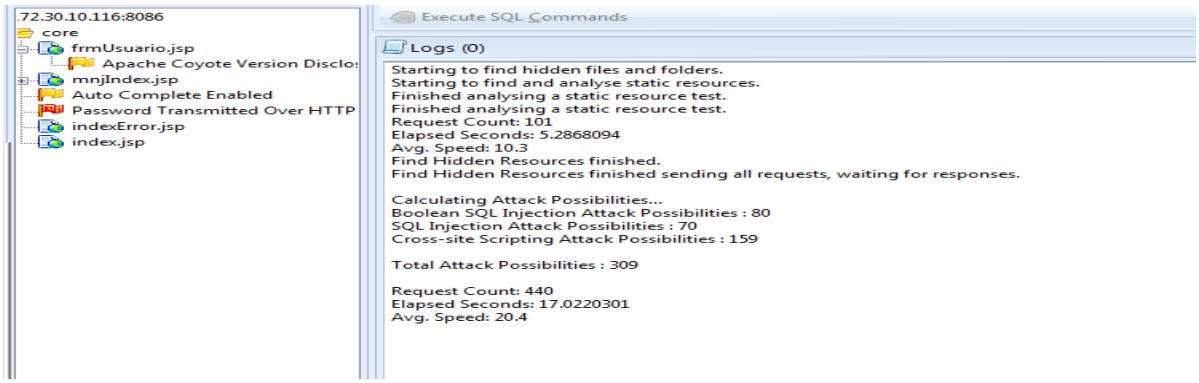
En la siguiente imagen se puede ver el resumen de la vulnerabilidad proporcionada por el analizador el mismo que nos indica que el nivel es bajo

En esta imagen se visualiza la cabecera y la codificación utilizada al realizar la operación.

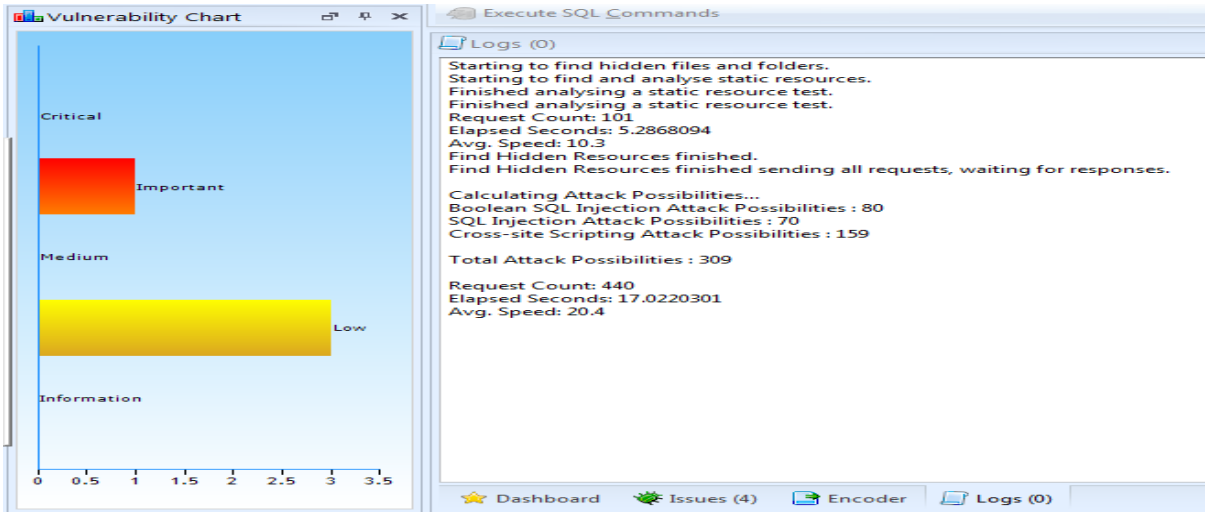


Operación Insertar mediante el formulario **frmUsuario**

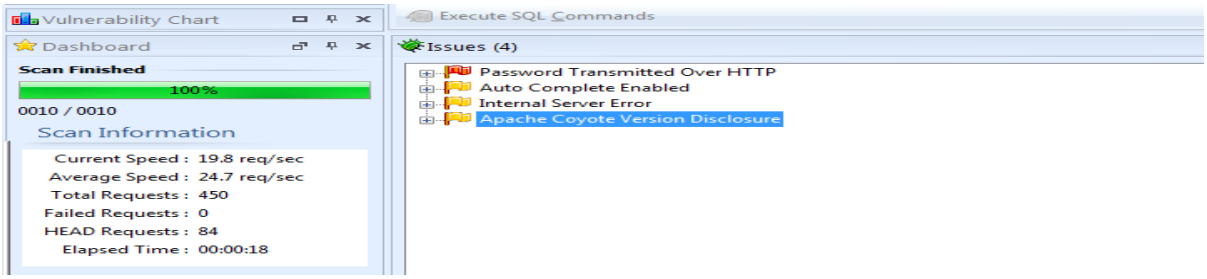
Seleccionamos el ítem **logs**, las mismas que proporcionan datos de las diferentes técnicas a evaluar.



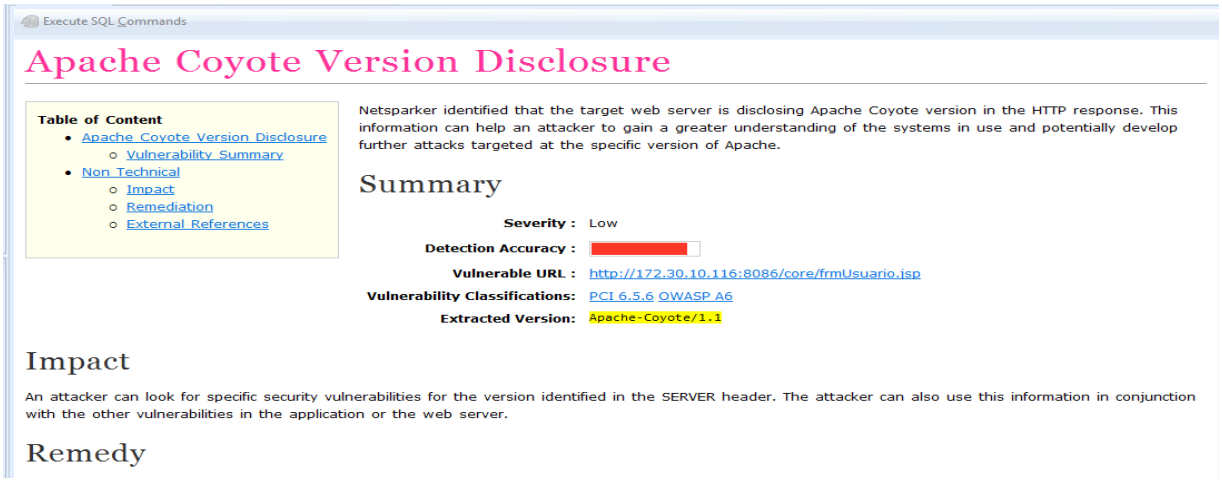
A continuación se visualiza graficas estadisticas sobre el nivel de vulnerabilidad de la URL asignada al analizador Netsparker.



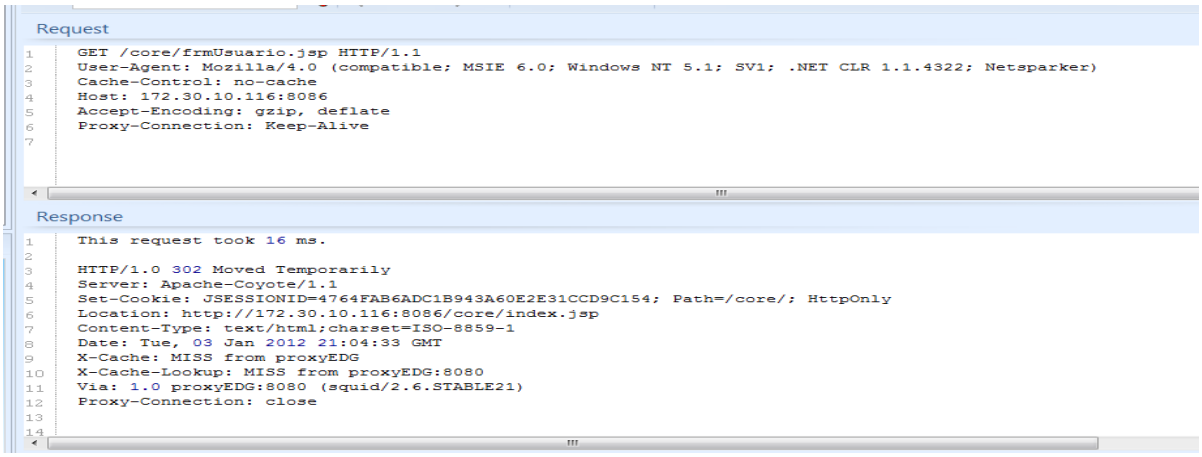
En esta imagen se despliega los puntos más vulnerables encontrados durante el análisis, los mismos que se puede diferenciar por su color.



En la siguiente imagen se puede ver el resumen de la vulnerabilidad proporcionada por el analizador el mismo que nos indica que el nivel es bajo.

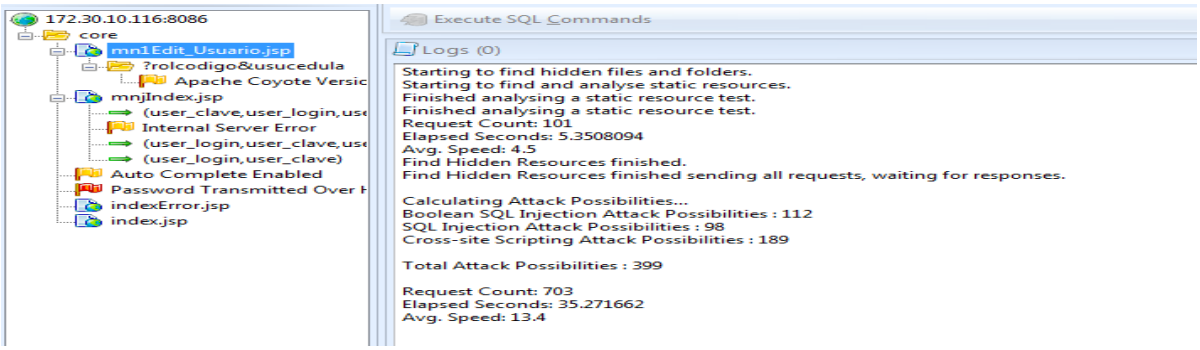


En esta imagen se visualiza la cabecera y la codificación utilizada al realizar la operación.

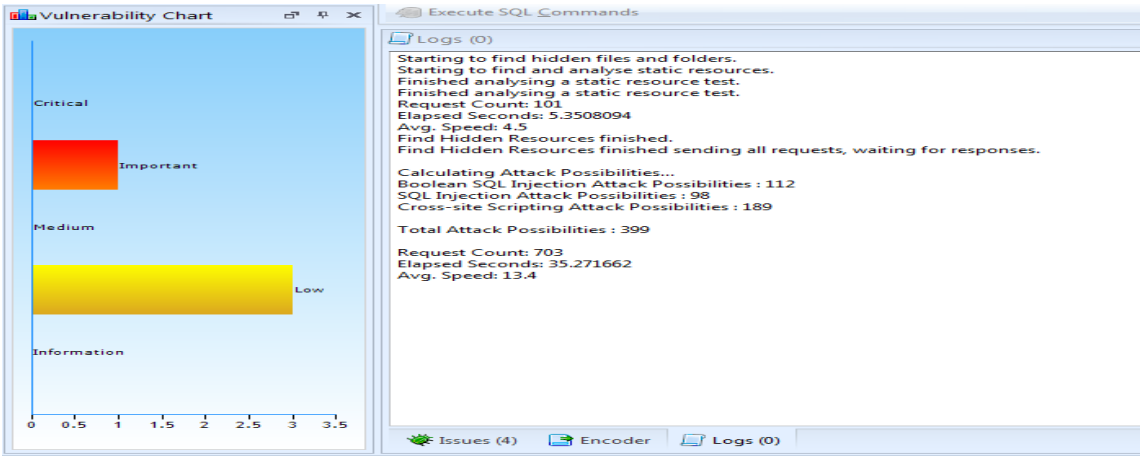


Operación Actualizar mediante el formulario **mn1Edit_Usuario**

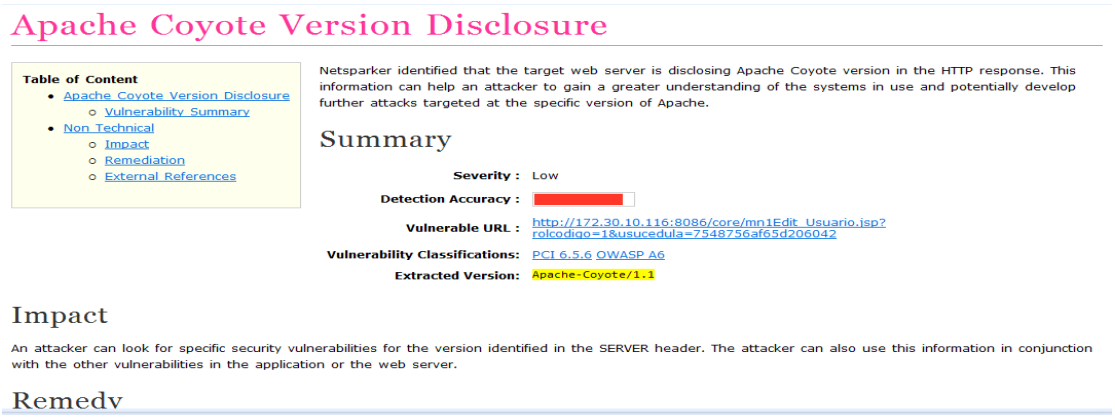
Seleccionamos el ítem **logs**, las mismas que proporcionan datos de las diferentes técnicas a evaluar.



A continuación se visualiza graficas estadisticas sobre el nivel de vulnerabilidad de la URL asignada al analizador Netsparker.



En la siguiente imagen se puede ver el resumen de la vulnerabilidad proporcionada por el analizador el mismo que nos indica que el nivel es bajo.



En esta imagen se visualiza la cabecera y la codificación utilizada al realizar la operación.

GLOSARIO

ORM

mapeo objeto-relacional técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programaciónorientado a objetos y el utilizado en una base de datos relacional, utilizando un motor de persistencia.

FRAMEWORK

Es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, con base a la cual otro proyecto de software puede ser más fácilmente organizado y desarrollado

RDBMS

Sistema de Gestión de Base de Datos Relacional

PERSISTENCIA

La acción de preservar la información de un objeto de forma permanente (guardar), pero a su vez también se refiere a poder recuperar la información del mismo (leer) para que pueda ser nuevamente utilizada.

JPA

Java Persistence API, Es un framework del lenguaje de programación Java que maneja datos relacionales en aplicaciones usando la Plataforma Java en sus ediciones Standard

POO

Programación Orientada a Objetos.

LISTENER

Es un proceso servidor que provee la conectividad de red con la base de datos Oracle. El listener está configurado para escuchar la conexión en un puerto específico en el servidor de base de datos.

SERIALIZACIÓN

Es el proceso de convertir un objeto en una secuencia de bytes para conservarlo en memoria, una base de datos o un archivo.

XML

eXtensible Markup Language ('lenguaje de marcas extensible")

TRANSACCIÓN

Controlan el acceso concurrente a los datos por parte de múltiples programas y en evento en que ocurra una falla del sistema, las transacciones aseguran que después de la recuperación los datos estén en un estado consistente.

AUDITORIA DE DB

Es el proceso que permite medir, asegurar, demostrar, monitorear y registrar los accesos a la información almacenada en las bases de datos.

PRACTILINE SOURCE

La capacidad de contar líneas de código, líneas de comentario y code/comment mezclado alinea en solos archivos así como directorios enteros

JPQL

Java Persistence Query Language (JPQL) es un lenguaje de consulta independiente de plataforma orientado a objetos definidos como parte de la especificación Java Persistence API

BIBLIOGRAFÍA

LIBROS

1. RICHARD SPERKO. Java Persistence for Relational Databases, 2003.
2. SCOTT W. AMBLER, Mapping objects to relational databases: O/R mapping in detail., 2006.

BIBLIOGRAFÍA DE INTERNET

3. Arquitectura de JPA
   http://blog.marcomendes.com/category/tecnologias-java/j2ee
4. Confidencialidad

   http://es.wikipedia.org/wiki/Seguridad_de_la_informaci%C3%B3n#Confidencialidad
5. Entidades
   http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=AnotacionesEB3
6. JPA
   http://sophia.javeriana.edu.co/~javila/pregrado/arquitectura/JPA.pdf
7. JPA
   http://www.fing.edu.uy/inco/cursos/tsi/TSI2/teorico/Clase5a-JPA.pdf112
8. Mapeo
   http://www.softwareguru.com.mx/downloads/SG-200502.pdf
9. Mapeando objetos RDBMS
   http://pizarropablo.googlepages.com/ORM-ObjectRelationalMapping-PizarroP.pdf
10. Objetos
    http://www.db4o.com/espanol/db4o%20Whitepaper%20%20Bases%20de%20Objetos.pdf
11. ORM
    http://msevents.microsoft.com/CUI/WebCastEventDetails.aspx?EventID=1032309114&EventCategory=5&culture=es-MX&CountryCode=MX

12. Persistencia de Objetos
http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=PersistenciaJava
13. Persistencia
http://es.wikipedia.org/wiki/Persistencia_de_objetos
14. Relaciones entre Entidades
http://www.intertech-inc.com/resource/usergroup/Exploring/20JPA.pdf
15. Serialización
http://msdn2.microsoft.com/es-es/library/ms233836(VS.80).aspx
16. Seguridad de la Información
http://es.wikipedia.org/wiki/Seguridad_de_la_informaci%C3%B3n#Integridad
17. Seguridad
http://www.usc.es/~catelmed/2005/materialAsignatura/SeguridadEnLaTransmisionDeDatos.pdf
18. Unidad de Persistencia
http://java.sun.com/developer/technicalArticles/J2SE/Desktop/persistenceapi/?fed=JSC