UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Desarrollo de un sitio web para consultas de libros de la bibliografía de las materias de la carrera de Ingeniería en Computación

TESIS PROFESIONAL

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACIÓN

P R E S E N T A :

ORLANDO SÁNCHEZ PÉREZ

DIRECTORA DE TESIS:

DRA. ANA MARÍA VÁZQUEZ VARGAS

CIUDAD UNIVERSITARIA, MÉXICO, D.F., SEPTIEMBRE 2010

Universidad Nacional Autónoma de México
UNAM
Biblioteca Central
Dirección General de Bibliotecas de la UNAM

UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

AGRADECIMIENTOS

Agradezco a mi mamá, aunque no hay palabras que puedan describir el amor y el soporte que eres en mi vida, siempre estaré agradecido, gracias mamá por todo y por tanto.

A Selene el amor de mi vida y mi mejor amiga, por su apoyo y paciencia.

A mi sobrino Juan Diego, por ser como mi hermano.

A Nufi por sus consejos y todo su apoyo gracias.

A la UNAM, en especial a la Facultad de Ingeniería por ser el lugar donde se forjan a grandes hombres y mujeres de mi México.

A todos los profesores de la Facultad de Ingeniería, por su compromiso y enseñanzas gracias.

A la Doctora Ana María Vázquez Vargas, mi Profesora y asesora de tesis, por siempre brindar una sonrisa a sus alumnos y siempre impulsarme a dar lo mejor de mi durante la elaboración de la tesis gracias.

Al jurado designado Ing. Maricela Castañeda Perdomo, M.I Norma Elva Chávez Rodríguez, M.I Aurelio Adolfo Millán Nájera y al Ing. Orlando Zaldívar Zamorategui. Por sus valiosas aportaciones para mejorar la presente tesis.

A mis Amigos de la carrera Adrian, Aldebarán, Alejandro, Arnulfo, Daniel, Enrique, Gil, Jesús, Marco, Mauricio, Raúl, Rigoberto, Rodolfo y Ulises por todos los momentos compartidos siempre están en mis recuerdos.

A Dios que siempre ve por todos.

Índice

Índice de imágenes

I Introducción

El desarrollo de la tecnología, ha revolucionado el concepto de diversos medios de consulta de información, transformándola y dando lugar a nuevos conceptos como son los catálogos digitales. Son muchos los retos a los que se enfrentan diversas comunidades, como son las de marketing, bibliotecas, museos, librerías, cartografía, etc. Todo tipo de información se puede catalogar y clasificar; por lo tanto, cualquier usuario de la información puede hacer uso de un catálogo digital debido a que la nueva era digital ha modificado la forma de acceso a la misma. De acuerdo al tipo de información se requiere un nuevo modelo para realizar la búsqueda, organización y distribución de la información, así como los servicios, funciones y relaciones que se establecen con el usuario. El objetivo principal de estos catálogos es proporcionar acceso universal a la información, adoptando modelos basados en infraestructuras tecnológicas avanzadas que permitan al usuario final acceder a la información de manera transparente sin importar que forma adopte, ni donde se encuentre.

Los catálogos digitales tienen diversas ventajas, entre ellas, es la difusión de la información de forma masiva, la cual puede ser presentada de forma dinámica y atractiva, además de que se tiene la facilidad de corregir errores de escritura o cambiar el diseño para llamar la atención. Son mucho menos costosos que los catálogos impresos y son publicados por medio de enlaces haciendo referencia a una página Web.

Otra de las ventajas de los catálogos digitales es que al contar con la información digitalizada, ésta se puede exportar y migrar o adaptar según la evolución del sistema del catálogo digital.

I.1 Objetivos

Objetivo general:

Desarrollo de una aplicación: un catálogo digital de libros con una interfaz amigable para el usuario, publicado en un sitio Web.

Objetivos particulares:

- Elaboración de una base de datos para guardar la información que se presentará en el catálogo.
- Elaboración de un sitio Web para la presentación de la información de la base de datos.
- Construir un catálogo digital de libros de la bibliografía recomendada para las materias del bloque de Ciencias de la Ingeniería de la Facultad de Ingeniería de la UNAM, con una interfaz amigable para el usuario en el cual pueda consultar la carátula del libro, parte del contenido (índice) o en su caso obtener el libro, la ficha bibliográfica y un resumen de éste.

I.2 Metodología

Para el desarrollo de la aplicación planteada anteriormente, se propone el uso de un enfoque por fases para el análisis y el diseño, el cual consiste en un ciclo específico de actividades del analista y el usuario. (SDLC, *Systems Development Life Cycle[8]).*

Las fases que se siguieron para el desarrollo del catálogo digital son:

1. Identificación de problemas, oportunidades y objetivos.

2. Determinación de los requerimientos de información.

3. Análisis de las necesidades del sistema.

4. Diseño del sistema recomendado.

5. Desarrollo y documentación del software.

6. Pruebas y mantenimiento del sistema.

7. Implementación y evaluación del sistema.

I.3 Requerimientos del problema

- Se requiere desarrollar un sistema, en el cual se pueda guardar la información, sobre la bibliografía recomendada para las materias del bloque de Ciencias de la Ingeniería de la Facultad de Ingeniería de la UNAM, para poder después ser consultada por medio de un catálogo digital.

- La aplicación a desarrollar debe tener la facilidad de poder ser consultada a través de una página Web.

- Debe ser amigable para los usuarios que consultan el catálogo digital a través de dicha página Web.

II Marco Teórico

La aplicación Web hará uso de la estructura de la arquitectura de software llamada "Modelo 2". Esta arquitectura de software se utiliza para realizar páginas JSP y es realmente el Modelo Vista Controlador (MVC) enfocado para el diseño de aplicaciones Web. Es por eso que los dos términos tanto "modelo 2" y "MVC" se puedan utilizar indistintamente en el mundo Web. La arquitectura [19] modelo 2 y sus derivados son la piedra angular para todas las aplicaciones Web dado que dan una estructura modular [23] con la cual da fuerza al diseño Web.

El esquema general del modelo 2 puede ser definido en el siguiente diagrama.

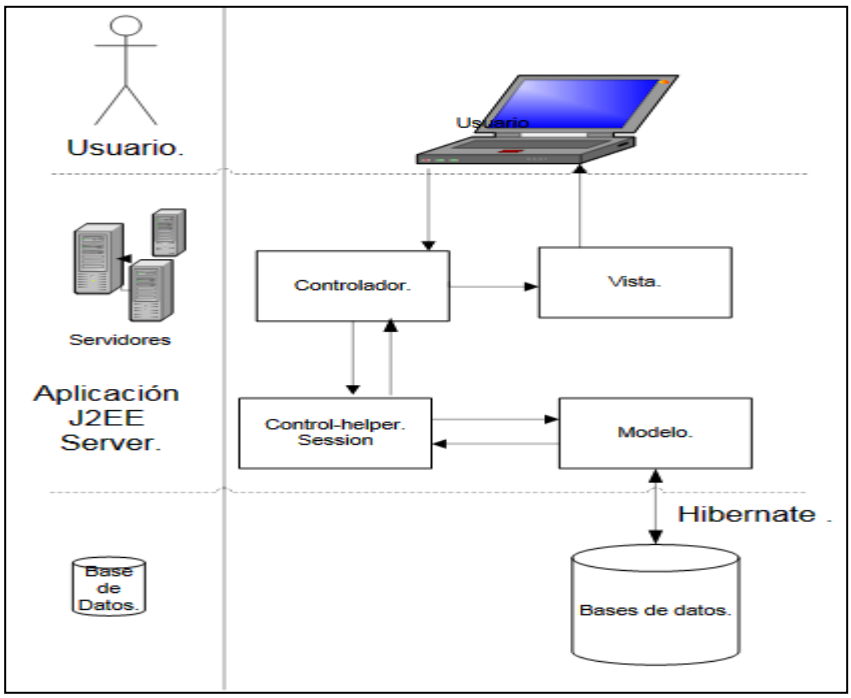

Figura 1. Esquema general del modelo 2 Utilizado para el estándar JSP.

Como se muestra en el esquema general de la Figura 1.

1. El Controlador recibe las peticiones del usuario.

2. La clase helper se encarga de realizar las consultas a la base de datos dentro del modelo (no directamente en la base de datos dado que el único elemento que tiene contacto con la base de datos es el modelo).

3. El controlador toma los datos obtenidos y los envía a la Vista, con lo que el usuario obtiene la respuesta.

II.1 Modelo Vista Controlador (MVC)

Model-View-Controller [20] (MVC) es un patrón de diseño, usado en aplicaciones Web que necesitan tener la capacidad de mostrar múltiples vistas de los mismos datos (la base de datos). El MVC está definido por tres partes: la primera (el modelo) se encarga de los datos, la segunda (las vistas web) que tienen como función mostrar una parte o la totalidad de los datos y por último el controlador encargado de los eventos que afectan a los dos anteriores.

Modelo.

El modelo define los datos que se utilizarán en la aplicación. Es la única parte del sistema que habla con la base de datos (por medio de Hibernate).

Vista.

La vista muestra los datos al usuario. La vista no hace ninguna transformación de los datos, sólo presenta los datos. Usualmente hay múltiples vistas.

Controlador.

El controlador es el programa que une las vistas y el modelo. Toma la entrada del usuario y se da cuenta de lo que significa para el modelo. Indica al modelo que se actualice a sí mismo, y hace que el nuevo estado del modelo esté disponible para la vista [4].

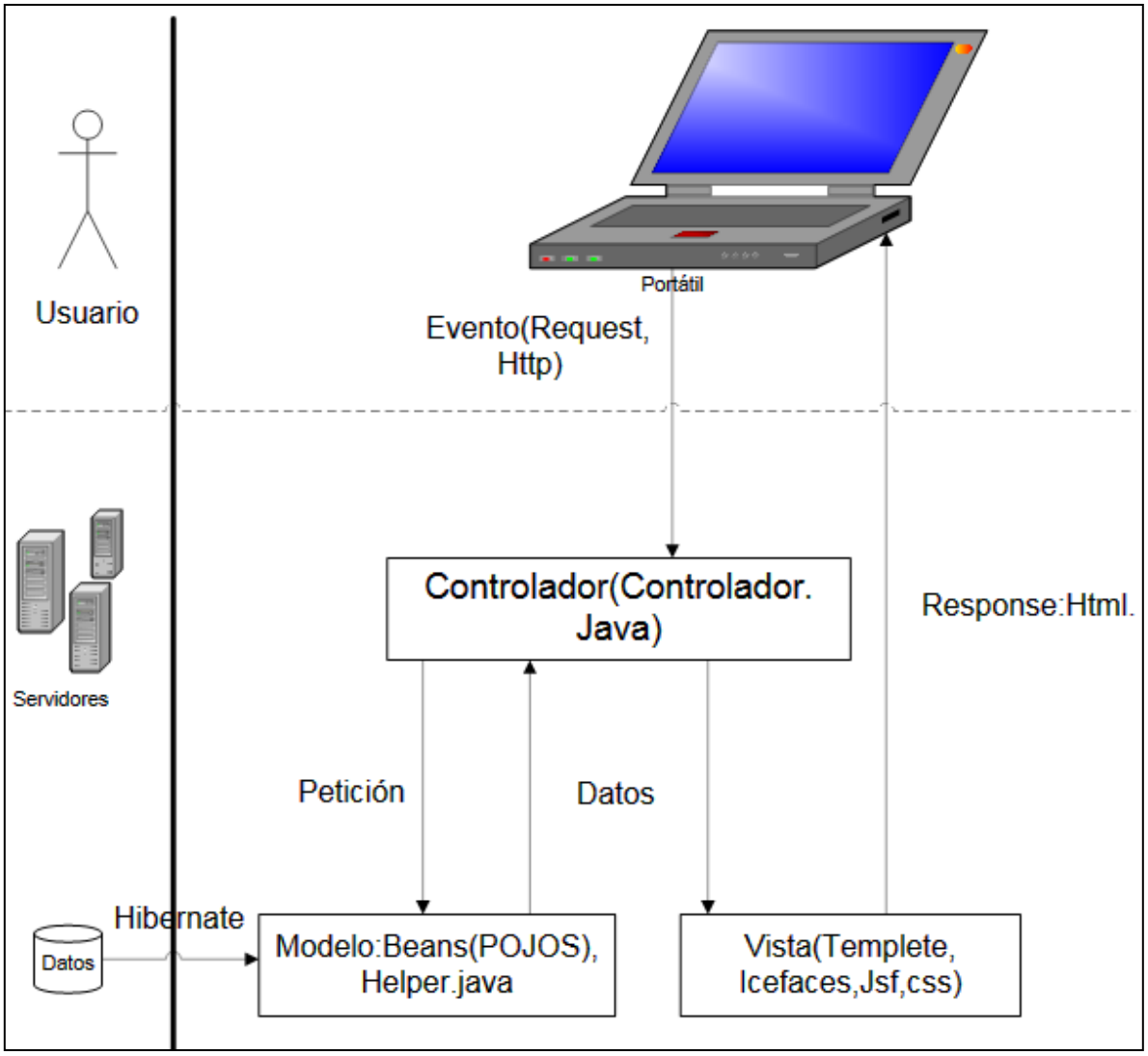

Figura 2. Modelo Vista Controlador para la aplicación Web que se implementara.

II.2 Modelo

Como se puede observar en el esquema del Modelo Vista Controlador se utilizará una base de datos relacional para formar la parte del Modelo.

II.2.1 Bases de datos

"Las bases de datos son parte fundamental de los sistemas de cómputo modernos. Dado que permiten un manejo dinámico de la información, proporcionan seguridad, precisión y control; además de proveer elementos relevantes para la toma de decisiones operativas y tácticas relacionadas con el manejo de grandes volúmenes de información" [12].

"Una base de datos es, un conjunto estructurado de datos guardados sobre soportes accesibles por la computadora, para satisfacer simultáneamente varios usuarios de manera selectiva y en un tiempo razonable" [2].

Las bases de datos tienen objetivos muy específicos (integridad de los datos, confidencialidad, concurrencia de acceso, seguridad de funcionamiento y todas estas funciones son controladas por un "Sistema de Gestión de Bases de Datos", SGBD, que es un software encargado específicamente de que la base de datos funcione correctamente.

Una Base de Datos puede ser definida como una colección estructurada y no redundante de datos y relaciones que los asocian, almacenada sobre soportes accesibles por computadora y destinada a realizar numerosas aplicaciones.

Una base de datos está realizada para almacenar hechos del mundo real o simplemente datos y es capaz de generar nuevos conocimientos.

Una base de datos se dice que es inteligente si tiene ciertas propiedades de inteligencia humana, es decir, que tenga una interfaz inteligente, donde el usuario pueda interactuar con la computadora de manera casi natural.

II.2.2 Modelo Relacional

El modelo relacional fue propuesto E.F.Codd en los laboratorios de IBM en California, es el modelo más empleado, dado su versatilidad y potencia.

Un modelo relacional es un modelo de datos basado en el concepto matemático de relación. Una relación es una parte del producto cartesiano de una lista de atributos, en donde cada atributo representa un valor de un dominio.

Un dominio en la teoría de relaciones es un conjunto de valores posibles que puede tomar un atributo en nuestra base de datos, por ejemplo, la edad de un estudiante podría estar comprendida estrictamente en el intervalo (19, 26). Los dominios de una relación no son necesariamente distintos. Si se designa por D_1, D_2,…, D_n. una secuencia ordenada de n dominios, la relación será una parte del producto cartesiano D_1 x D_2 x …x D_n, los elementos de la relación de grado n son tuplas con notación $(d_1 , d_2, …, d_n)$ en donde cada d_i toma valor en su dominio correspondiente.

En el modelo relacional:

- Las bases de datos relacionales almacenan datos y resultados en tablas.
- Las tablas están compuestas de filas y columnas.
- Cada tabla tiene una llave primaria, que es un identificador único de la tabla; la llave primaria puede estar compuesta por una o varias columnas.
- Para establecer relaciones entre las tablas es necesario incluir la llave primaria de una tabla A en una tabla B, en la tabla B la columna tomará el nombre de llave foránea.

II.2.3 Lenguajes de interrogación

Fundamentalmente, han existido dos enfoques teóricos de estos lenguajes, que se les clasifica en dos grandes grupos:

II.2.3.1 Álgebra Relacional

El Álgebra Relacional es una notación algebraica, en la cual las consultas se expresan aplicando operadores especializados a las relaciones.

El *Álgebra Relacional* fue introducida por E. F. Codd en 1972. Consiste en un conjunto de operadores con las relaciones.

- SELECT (σ): extrae *tuplas* a partir de una relación que satisfagan una restricción dada. Sea *R* una tabla que contiene un atributo *A*. $\sigma_{A=a}(R) = \{t \in R \mid t(A) = a\}$ donde t denota una tupla de *R* y t(A) denota el valor del atributo *A* de la tupla t (la proyección de R sobre el atributo x).
- PROJECT (Producto cartesiano) (π): extrae *atributos* (columnas) específicos de una relación. Sea R una relación que contiene un atributo X. $\pi_X(R) = \{t(X) \mid t \in R\}$, donde t(X) denota el valor del atributo X de la tupla t.
- PRODUCT (×): construye el producto cartesiano de dos relaciones. Sea R una tabla de grado (arity) k_1 y sea S una tabla con grado (arity) k_2. R × S es el conjunto de las $k_1 + k_2$-tuplas cuyos primeros k_1 componentes forman una tupla en R y cuyos últimos k_2 componentes forman una tupla en S.
- UNION (∪): supone la unión de la teoría de conjuntos de dos tablas. Dadas las tablas R y S (y ambas deben ser del mismo grado), la unión R ∪ S es el conjunto de las tuplas que están en R S o en las dos.
- INTERSECT (∩): construye la intersección de la teoría de conjuntos de dos tablas. Dadas las tablas R y S, R ∪ S es el conjunto de las tuplas que están en R y en S. De nuevo requiere que R y S tengan el mismo rango.
- DIFFERENCE (− or \): supone el conjunto diferencia de dos tablas. Sean R y S de nuevo dos tablas con el mismo rango. R - S Es el conjunto de las tuplas que están en R pero no en S.
- JOIN (∏): conecta dos tablas por sus atributos comunes. Sea R una tabla con los atributos A, B y C y sea S una tabla con los atributos C, D y E. Hay un atributo común para ambas relaciones, el atributo C. $R \prod S = \pi_{R.A,R.B,R.C,S.D,S.E}(\sigma_{R.C=S.C}(R \times S))$. ¿Qué estamos haciendo aquí? Primero calculamos el producto cartesiano R × S. Entonces seleccionamos las tuplas cuyos valores para el atributo común C sea igual ($\sigma_{R.C = S.C}$). Ahora tenemos una tabla que contiene el atributo C dos veces y lo corregimos eliminando la columna duplicada.

II.2.3.2 El Cálculo Relacional

El cálculo relacional expresa las consultas a la base de datos utilizando un lenguaje formal. Es el lenguaje de cálculo de predicados donde se trata de saber si el predicado es verdadero o falso.

II.3 Vista: Diseño de la página Web

Con respecto al MVC la parte de la vista está formada por un templete Web implementado con XHTML, CSS. ICEfaces se utiliza para mostrar el contenido obtenido de la base de datos en conjunto con el templete.

II.3.1 Templete Web e ICEfaces

Un templete Web es una herramienta utilizada para separar el contenido (la información obtenida de la base de datos) de la presentación. El templete Web ofrece una estructura en la cual se puede mostrar los datos obtenidos de la base de datos [39].

Al mantener el contenido separado de la estructura, se puede elegir en el futuro para cambiar el templete Web sin perder su contenido.

ICEfaces se encarga de mostrar el contenido obtenido de la base de datos. El usuario hace una petición en el navegador, ahí se realiza un proceso (en la parte de la implementación explicaré la implementación este proceso) que regresa un valor que es tomado por ICEfaces, éste lo muestra en el navegador en conjunto con el templete.

II.4 Interfaz

Para establecer la comunicación entre la base de datos y la página Web, se configurara con Hibernate.

Hibernate utiliza dos tipos configuración, la primera la conexión con la base de datos, la segunda ayuda a crear POJOS y archivos de mapeos (en la parte de la implementación se explicara ampliamente).

Con la configuración de Hibernate se ha creado la comunicación entre la base de datos y el modelo.

II.4.1 Controlador

Ya que se tiene la comunicación con la base de datos y el modelo, una clase java llamada controlador se encarga de unir las vistas y el modelo. El controlador ve todos los eventos solicitados por el usuario, hace consultas con ayuda de una clase llamada helper y regresa una respuesta la cual será formada en el navegador por medio de ICEfaces.

III Herramientas de Software para la implementación de la aplicación

Para el desarrollo del sitio Web se utilizaran las siguientes herramientas de software, las cuales están divididas según el Modelo Vista Controlador en la siguiente tabla.

Tabla. Herramientas de software enmarcadas en el Modelo Vista Controlador.

<table>
<tr><th>Modelo Vista Controlador</th><th colspan="3">Herramienta de software</th></tr>
<tr><td>Modelo</td><td rowspan="4">La aplicación general será administrada desde IDE NETBEANS.</td><td>Base de Datos la cual será administrada con Mysql desde phpMyAdmin.
La comunicación entre base de datos y la aplicación será administrada mediante Hibernate formando el modelo.</td><td>Servidor Mysql.

Librerías Hibernate.</td></tr>
<tr><td>Vista</td><td>Está formado por el templete el cual utiliza la tecnología XHTML y CSS.</td><td>Interpretado por el Navegador.</td></tr>
<tr><td rowspan="2"></td><td>Para mostrar la información dentro de la página se utiliza la tecnología ICEfaces.</td><td rowspan="2">Servidor GlassFish.</td></tr>
<tr><td>Controlador</td><td>El controlador Utiliza la tecnología Java.</td></tr>
</table>

III.1 Sistema manejador de bases de datos (MySQL)

Las bases de datos relacionales almacenan y muestran resultados en tablas. Para añadir, acceder, y procesar los datos almacenados en una base de datos, necesitamos un sistema manejador de base de datos o gestor de base de datos.

MySQL es un sistema de gestión de base de datos relacional, multihilo y multiusuario, desde enero de 2008 una subsidiaria de Sun Microsystems y ésta a su vez de Oracle Corporation desde abril de 2009.

MySQL es un sistema de administración de bases de datos (Database Management System, DBMS) para bases de datos relacionales. MySQL no es más que una aplicación que permite gestionar archivos llamados de bases de datos. MySQL, como base de datos relacional, utiliza múltiples tablas para almacenar y organizar la información.

MySQL fue escrito en C y C++ y destaca por su gran adaptación a diferentes entornos de desarrollo, permitiendo su interactuación con los lenguajes de programación más utilizados como PHP, Perl y Java y su integración en distintos sistemas operativos [22].

III.2 phpMyAdmin

phpMyAdmin es una herramienta de software libre, escrita en PHP con la cual se administra MySQL sobre Red (World Wide Web). phpMyAdmin soporta un amplio rango de operaciones de MySQL. La mayoría de las operaciones de uso frecuente son compatibles con la interfaz de usuario, tablas, campos, relaciones, índices, usuarios, permisos etc.

- Interfaz Web intuitiva.

Figura 3. Interfaz phpMyAdmin.

- Se puede ejecutar desde el administrador cualquier sentencia SQL. Soporta las características MySQL:

  - navegar y borrar bases de datos, tablas, vistas, campos e índices.
  - crear, copiar, eliminar y alterar cambiar el nombre de bases de datos, tablas, campos e índices.
  - mantenimiento del servidor, bases de datos y tablas, con propuestas sobre la configuración del servidor.
  - ejecutar, editar y marcar cualquier instrucción SQL.
  - administrar usuarios y privilegios de MySQL.
  - gestión de procedimientos almacenados y disparadores.

III.3 NetBeans IDE

El IDE NetBeans es un IDE - una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java - pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el IDE NetBeans. El IDE NetBeans es un producto libre y gratuito sin restricciones de uso.

El NetBeans IDE es un IDE de código abierto, escrito completamente en Java usando la plataforma NetBeans. El NetBeans IDE soporta el desarrollo de todos los tipos de aplicación Java (J2SE, Web, EJB y aplicaciones móviles). Entre sus características se encuentra un sistema de proyectos basado en Ant, control de versiones y refactoring.

NetBeans IDE 6.5, la cual fue lanzada el 19 de noviembre de 2008, extiende las características existentes del Java EE (incluyendo Soporte a Persistencia, EJB 3 y JAX-WS). Adicionalmente, el NetBeans Enterprise Pack soporta el desarrollo de Aplicaciones empresariales con Java EE 5, incluyendo herramientas de desarrollo visuales de SOA, herramientas de esquemas XML, orientación a Web servicies (for BPEL), y modelado UML. El NetBeans C/C++ Pack soporta proyectos de C/C++, mientras el PHP Pack, soporta PHP 5.

Modularidad. Todas las funciones del IDE son provistas por módulos. Cada módulo provee una función bien definida, tales como el soporte de Java, edición, o soporte para el sistema de control de versiones. NetBeans contiene todos los módulos necesarios para el desarrollo de aplicaciones Java en una sola descarga, permitiéndole al usuario comenzar a trabajar inmediatamente.

Sun Studio, Sun Java Studio Enterprise, y Sun Java Studio Creator de Sun Microsystems han sido todos basados en el IDE NetBeans [38].

Las siguientes herramientas de software se administraron mediante Netbeans.

III.3.1 Servidor GlassFish

GlassFish [27] es un servidor de aplicaciones desarrollado por Sun Microsystems que implementa las tecnologías definidas en la plataforma Java EE y permite ejecutar aplicaciones que siguen está especificación. La versión comercial es denominada Sun GlassFish Enterprise Server. Es gratuito y de código libre, se distribuye bajo un licenciamiento dual a través de la licencia CDDL y la GNU GPL.

GlassFish está basado en el código fuente donado por Sun y Oracle Corporation, éste último proporcionó el módulo de persistencia TopLink. GlassFish tiene como base al servidor Sun Java System Application Server de Sun Microsystems, un derivado de Apache Tomcat y que usa un componente adicional llamado Grizzly que usa Java NIO para escalabilidad y velocidad.

III.3.2 Java

El lenguaje de programación Java está diseñado para satisfacer los desafíos de desarrollo de aplicaciones en un amplio entorno, desde aplicaciones de escritorio hasta lo que son entornos de red de amplia distribución. Teniendo como meta que estos consuman un mínimo de recursos del sistema, pueden ejecutarse en cualquier plataforma (hardware y software) y se puede ampliar de forma dinámica.

III.3.2.1 Principales Características de Java

- **Lenguaje gratuito**

Creado por SUN Microsystems absorbido por Oracle, que distribuye gratuitamente el producto base, denominado JDK (Java Development Toolkit) o actualmente J2SE (Java 2 Standard Edition). API distribuida con el J2SE muy amplia.

- **Simple**

El crecimiento masivo de Internet nos lleva a una forma completamente nueva de ver el desarrollo y la distribución de software. La tecnología Java permite el desarrollo seguro, robusto y aplicaciones multi-plataformas de forma heterogénea y de redes distributivas.

El sistema surgió para satisfacer estas necesidades, es sencillo, por lo que se puede programar fácilmente por la mayoría de los desarrolladores, de una manera familiar de modo que los desarrolladores actuales pueden aprender fácilmente el lenguaje de programación Java.

- **Orientado a Objetos**

Java fue diseñado como un lenguaje orientado a objetos [18]. Los objetos agrupan en estructuras encapsuladas tanto sus datos como los métodos (o funciones) que manipulan esos datos. La tendencia del futuro, a la que Java se suma, apunta hacia la programación orientada a objetos, especialmente en entornos cada vez más complejos y basados en red.

- **Distribuido**

Se dice que Java es distribuido [16] ya que tiene un amplio api de rutinas para manejo fácil de los protocolos TCP/IP como HTTP y FTP. Con estas rutinas los programadores pueden acceder a los objetos mediante una URL con tal facilidad como si fuera un archivo local de una PC.

- **Dinámico**

El lenguaje Java y su sistema de ejecución en tiempo real son dinámicos en la fase de enlazado. Las clases sólo se enlazan a medida que son necesitadas. Se pueden enlazar nuevos módulos de código bajo demanda, procedente de fuentes muy variadas, incluso desde Internet.

- **Arquitectura neutral y portable**

En el lenguaje de programación Java, todo el código fuente es primero escrito en un archivo que tiene la extensión ".java", los archivos se compilan y entonces se genera un archivo ".class".

Figura 4. Esquema de compilación de un archivo .java y generación del .class[25].

El archivo ".class" no contiene código que es nativo de su procesador, sino que contiene bytecodes, el lenguaje de la máquina virtual de Java (Java Virtual Machine por sus iniciales JVM), reconoce Java, corre la aplicación con una instancia de máquina virtual de Java, de aquí surge el concepto de la portabilidad, es decir, la máquina virtual puede trabajar en varios sistemas operativos. Lo único que cambia de sistema operativo a otro es la máquina virtual para cada caso. Entonces el archivo .class puede correr sobre Microsoft Windows, en un Solaris Operarating (Solaris OS), Linux, o en Mac, a través de la máquina virtual con lo que es posible correr la misma aplicación en múltiples plataformas.

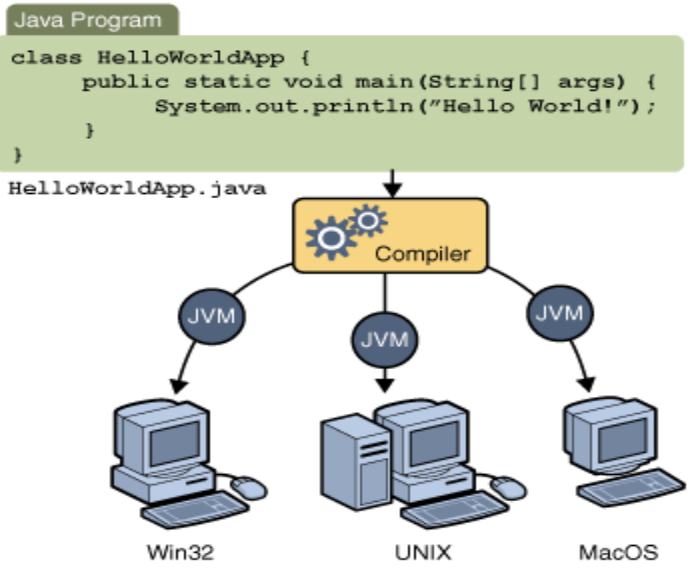

Figura 5. Esquema de portabilidad entre sistemas operativos [25].

La arquitectura neutral es parte de la portabilidad. La tecnología Java es portable al ser estricta en su definición del lenguaje base. La tecnología Java pone un patrón y especifica

los tamaños de sus tipos de datos básicos y el comportamiento de sus operadores aritméticos. Sus programas son los mismos en todas las plataformas, no hay incompatibilidades de tipo de datos a través de arquitecturas de hardware y software.

- **Robusto**

Java está diseñado para crear un software de alta fiabilidad, ofrece un primer nivel de comprobación en tiempo de compilación, seguido de un segundo nivel de comprobación en tiempo de ejecución. Con esto podemos desarrollar software confiable. Sus características de memoria liberan a los programadores de una familia entera de errores (la aritmética de punteros), ya que se ha prescindido por completo los punteros y la recolección de basura elimina la necesidad de liberación explícita de memoria.

- **Multihilo**

Java soporta sincronización de múltiples hilos de ejecución (multithreading) a nivel de lenguaje, especialmente útiles en la creación de aplicaciones de red distribuidas. Así, mientras un hilo se encarga de la comunicación, otro puede interactuar con el usuario mientras otro presenta una animación en pantalla y otro realiza cálculos.

- **Seguro**

La tecnología Java está diseñada para funcionar en entornos distribuidos, lo que significa que la seguridad es de vital importancia. Java permite construir aplicaciones las cuales no pueden ser invadidas desde afuera. En el entorno de red, las aplicaciones escritas en lenguaje Java son seguras, evitando intrusos.

- **Herramienta Java Bean**

Los JavaBeans son un modelo de componentes creado por Sun Microsystems para la construcción de aplicaciones en Java [5].

Se usan para encapsular varios objetos en un único objeto (la vaina ó Bean en inglés), para hacer uso de un sólo objeto en lugar de varios más simples.

La especificación de JavaBeans de Sun Microsystems los define como "componentes de software reutilizables que se puedan manipular visualmente en una herramienta de construcción".

A pesar de haber muchas semejanzas, los JavaBeans no deben confundirse con los Enterprise JavaBeans (EJB), una tecnología de componentes del lado servidor que es parte de Java EE.

Para la implementación de todo el sitio Web una de las herramientas más utilizadas son los beans para la elaboración de parte del Modelo (cuando hablo de Modelo me refiero a la parte del Modelo Vista Controlador) en la parte de los POJOS (en la implementación se explican ampliamente los POJOS y el Modelo Vista Controlador). También se utiliza beans para el Controlador (El Controlador me refiero al controlador del Modelo vista Controlador) este también guarda la estructura de un beans.

Para funcionar como una clase JavaBean, una clase debe obedecer ciertas convenciones sobre nomenclatura de métodos, construcción y comportamiento. Estas convenciones permiten tener herramientas que puedan utilizar, reutilizar, sustituir y conectar JavaBeans.

Las convenciones requeridas son:

- Debe tener un constructor sin argumentos.
- Sus propiedades deben ser accesibles mediante métodos get y set que siguen una convención de nomenclatura estándar.
- Debe ser serializado.

III.3.2.2 ICEfaces

ICEfaces [35] es la herramienta mediante la cual se mostraran los datos en la página Web. ICEfaces toma las ventajas del estándar JSF combinándolos con AJAX para construir las vistas.

ICEfaces es un framework Ajax de código abierto, que permite a desarrolladores de aplicaciones Java EE crear y desplegar aplicaciones de Internet enriquecidas (RIA) para servidores usando el lenguaje de programación Java.

ICEfaces aprovecha todos los estándares basados en los ecosistemas de herramientas y ambientes de ejecución Java. Las características de un trabajo rico en aplicaciones, son desarrolladas en Java puro y en un modelo puro de cliente ligero. No se requieren Applets o plug-ins propietarios para los navegadores. ICEfaces son aplicaciones JavaServer Faces (JSF) por lo tanto las habilidades en desarrollo de aplicaciones Java EE se aplican directamente y los desarrolladores Java evitan hacer cualquier desarrollo de Java Script.

III.3.2.2.1 Arquitectura

La primera meta de la arquitectura de ICEfaces es proveer una aplicación a los desarrolladores, con un modelo de trabajo de desarrollo familiar Java y cubren completamente las complejidades del desarrollo en JavaScript de bajo nivel Ajax. La clave de la arquitectura ICEfaces es un modelo de aplicación servidor-céntrico, donde todas las aplicaciones lógicas son desarrolladas en Java puro y se ejecuta en una aplicación de servidor ambiente de ejecución Java (JRE). Esto significa que la infraestructura Java EE, provoca mejores ambientes de desarrollo. La manera en que las aplicaciones Java EE son desarrolladas actualmente no cambia de manera significante y no se necesitará reestructurar completamente las aplicaciones existentes a las características enriquecidas basadas en Ajax.

Las características enriquecidas de la presentación de ICEfaces, están basadas en el estándar Java Server Faces. El desarrollo de aplicaciones ICEfaces es esencialmente desarrollo JSF el cual promueve una arquitectura basada en componentes, usando una

declarativa basada en etiquetas de definiciones de interfaz de usuario y datos dinámicos obligatorios, dentro de un modelo de datos de una aplicación que reside en un servidor. Utilizando un conjunto de componentes de ICEfaces habilitados para Ajax, los cuales proveen todos los componentes de los estándares JSF, al igual que un conjunto completo de componentes extendidos, los desarrolladores pueden crear una aplicación estándar JSF que se adhiere al estándar de servidor céntrico con aplicación de ciclo de vida JSF, pero se beneficia automáticamente de características basadas en Ajax de ICEfaces. Debido a que ICEfaces soporta aplicaciones estilo Ajax Push, se extiende el framework JSF para proveer un trigger-based, iniciado por el servidor de procesamiento de API, el cual hace simple mejorar las aplicaciones con las actualizaciones de presentación instantáneas, basadas en los cambios de estado de la aplicación del servidor residente. Usando ICEfaces y técnicas de programación Java/JSF los desarrolladores de aplicaciones serán capaces de desarrollar aplicaciones sin tener que escribir una sola línea de JavaScript.

Mientras que el mecanismo de presentaciones fundamental basado en Ajax en la implementación ICEfaces es completamente transparente al desarrollador de aplicaciones, es útil entender qué está pasando detrás de una aplicación de ICEfaces. Hay tres elementos esenciales en la arquitectura ICEfaces.

III.3.2.2.2 ICEfaces Framework

El Framework es una extensión del framework JSF estándar, con la principal diferencia que en ICEfaces está relacionado a la fase de formación (en la fase donde se refresca el navegador). En el estándar JSF, en la fase de formación se produce un nuevo marcador para el estado de la aplicación actual, y se entrega al navegador, donde ocurre una actualización de toda la página. Con el framework de ICEfaces, al refrescar el navegador ocurre dentro de un lado del servidor (DOM) y únicamente agrega los cambios al DOM y se entregan al navegador y se reensamblan con un puente de Ajax ligero (Ajax Bridge).

Esto resulta en una actualización transparente y ligera de la página del navegador con únicamente los elementos necesarios de la presentación siendo formados. El framework de ICEfaces también prevé un tiempo de administración completo de Ajax Push usando el formado de APIs en el servidor inicializado, e integra los mecanismos sin problemas con el ciclo de vida JSF.

III.3.2.2.3 Puente Ajax (Ajax Bridge)

El puente Ajax tiene como elementos un servidor residente y un cliente residente que coordinan la comunicación basada en Ajax entre el cliente navegador y el servidor en donde se encuentra la aplicación. El puente es responsable de entregar los cambios que se aumentan en la presentación desde la fase de formado al cliente del navegador, y reensamblar estos cambios en el navegador DOM para afectar los cambios de presentación. El puente también es responsable de detectar la interacción del usuario con la presentación y entregar al usuario eventos de vuelta a la aplicación para el procesamiento a través del ciclo de vida JSF estándar. Un mecanismo llamado "entrega parcial" es construir dentro de ICEfaces generación de eventos automáticos a través del puente, entonces el desarrollador de la aplicación no es expuesto al mecanismo de evento de bajo nivel. El Puente Ajax es estabilizado automáticamente a la primera página cargada de la aplicación y coordina las actualizaciones de la presentación y la transmisión de eventos de usuario por el tiempo de vida entero de la aplicación.

III.3.2.2.4 Suite de componentes ICEfaces

La suite de componentes provee todos los bloques de creación para la aplicación de interfaz de usuario. Esto incluye tanto los componentes del estándar JSF y un arreglo amplio de componentes avanzados que permite al desarrollador ensamblar interfaces de aplicaciones sofisticadas eficientemente y usan su atributo de presentación parcial para facilitar la generación de eventos automatizada sobre el puente de Ajax basado en la interacción de usuarios con la presentación de componentes. Opcionalmente los componentes ICEfaces pueden ser habilitados con una variedad de efectos script.aculo.us como el "arrastra y suelta". Nuevamente, los componentes ICEfaces tienen atributos que permiten diversos efectos, por lo que el desarrollador nunca es expuesto a la programación de bajo nivel de JavaScript para obtener las características dinámicas para un componente.

III.3.2.3 Hibernate

"Hibernate es una herramienta de mapeo objeto/relacional de código abierto para ambientes Java. El término mapeo se refiere "mapeo objeto/relacional" (OMR por sus siglas en inglés), se refiere a la técnica de "mapear" la representación de los datos desde un

modelo relacional hacia un modelo de objetos, con un esquema de base de datos en SQL" [24].

III.3.2.3.1 Persistencia

"La persistencia es un proceso por el que un objeto cualquiera se puede convertir en una secuencia de bytes con la que más tarde se podrá reconstruir el valor de sus variables. Esto permite guardar un objeto en un archivo o mandarlo por lo red" [6].

Se dice que Hibernate es una máquina de persistencia ya que abre un canal de comunicación entre el modelo relacional y el modelo orientado a objetos.

Hibernate se encarga de la configuración de la conexión a la base de datos, además de crear las clases de persistencia (POJOS) y los archivos de mapeo, esto se explicará en el capítulo IV.

III.3.2.3.2 Lenguaje de consulta de Hibernate (HQL)

Las consultas a la base de datos, no serán hechas directamente en la base de datos, debido que se mostrarán en un ambiente de objetos, después de hacer un proceso (El proceso se verá paso a paso en el diseño de la interfaz.) para que la base de datos que está en el mundo relacional pueda ser trabajada en el mundo de los objetos. Este proceso será realizado mediante Hibernate. Al pasar por el proceso veremos a la base de datos como un objeto, entonces la tecnología Hibernate tiene su propio generador de consultas que trabaja con objetos.

Hibernate usa un lenguaje de consulta de gran alcance (HQL) que es similar en apariencia a SQL. En comparación con éste, sin embargo, HQL es completamente orientado a objetos y comprende nociones de la programación orientada a objetos (Herencia, Polimorfismo).

En el sistema, en la parte de la interfaz de la clase Helper es donde se ejecutan las consultas, en el capítulo IV se puede ver una tabla de ejemplos de las consultas HQL.

III.3.3 XHTML

HTML [34], siglas de HyperText Markup Language (*Lenguaje de Marcado de Hipertexto*), es el lenguaje de mercado predominante para la elaboración de páginas Web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML se escribe en forma de "etiquetas", rodeadas por corchetes angulares (<,>). HTML también puede describir, hasta un cierto punto, la apariencia de un documento, y puede incluir un *script* (por ejemplo Javascript), el cual puede afectar el comportamiento de navegadores Web y otros procesadores de HTML.

HTML también es usado para referirse al contenido del tipo de MIME text/html o, todavía más, ampliamente como un término genérico para el HTML; ya sea en forma descendida del XML (como XHTML 1.0 y posteriores) o en forma descendida directamente de SGML (como HTML 4.01 y anteriores).

III.3.4 CSS

Las hojas de estilo en cascada [33] (en inglés *Cascading Style Sheets*), CSS es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). El W3C (World Wide Web Consortium) es el encargado de formular la especificación de las hojas de estilo, que servirán de estándar para los agentes de usuario o navegadores.

La idea que se encuentra detrás del desarrollo de CSS es separar la *estructura* de un documento de su *presentación*.

Por ejemplo, el elemento de HTML <h1> indica que un bloque de texto es un encabezamiento y que es más importante que un bloque etiquetado como <H2>. Versiones más antiguas de HTML permitían atributos extra dentro de la etiqueta abierta para darle

formato (como el color o el tamaño de fuente). No obstante, cada etiqueta <H1> debía disponer de la información si se deseaba un diseño consistente para una página y además, una persona que lea esa página con un navegador pierde totalmente el control sobre la visualización del texto.

Cuando se utiliza CSS, la etiqueta <H1> no debería proporcionar información sobre cómo va a ser visualizado, solamente marca la estructura del documento. La información de estilo separada en una hoja de estilo, específica cómo se ha de mostrar <H1>: color, fuente, alineación del texto, tamaño y otras características no visuales como definir el volumen de un sintetizador de voz por ejemplo.

La información de estilo puede ser adjuntada tanto como un documento separado o en el mismo documento XHTML. En este último caso podrían definirse estilos generales en la cabecera del documento o en cada etiqueta particular mediante el atributo "style".

Para la aplicación Web configuré como externa la página CSS.

Una hoja de estilo externa, es una hoja de estilo que está almacenada en un archivo diferente al archivo donde se almacena el código XHTML de la página Web. Esta es la manera de programar más potente, porque separa completamente las reglas de formateo para la página XHTML de la estructura básica de la página.

IV Implementación

Una de las ventajas de implementar los sistemas de computo mediante el uso del Modelo Vista Controlador, es que podemos ir probando cada segmento de código por separado, también se pueden hacer cambios de manera sencilla, por ejemplo, si el diseño de la página Web requiere de cambios, estos se realizan de manera fácil y sin mucho esfuerzo y sin pérdida de información.

La implementación cuenta con las siguientes etapas, las cuales están en marcadas en el Modelo Vista Controlar.

Tabla. Etapas de implementación.

Modelo	Diseño de la base de datos.
Vista	Diseño de la página Web.
Controlador	Une vistas, modelo y maneja las peticiones de los usuarios, también se encarga de hacer las consultas a la base de datos a través de la clase helper.

IV.1 Diseño de la base de datos

Al elaborar el análisis para la realización de la base de datos utilizamos el modelo entidad relación, para visualizar entidades y sus relaciones, estas son el reflejo del mundo real de la información, que el usuario busca sobre los libros que deseamos mostrar a través de la Web.

El diseño de la base de datos es el proceso de análisis del problema que necesitamos resolver y se puede dividir de la siguiente manera:

- Diseño conceptual.
- Diseño lógico.
- Diseño físico.

Diseño conceptual

En esta etapa se debe construir un esquema de la información que el usuario necesita, independientemente de cualquier consideración física. A este esquema se le denomina esquema conceptual. Al construir el esquema, se descubren la semántica (significado) de los datos que el usuario necesita: encuentran entidades, atributos y relaciones. El objetivo es comprender:

- La perspectiva que cada usuario tiene de los datos.
- La naturaleza de los datos, independientemente de su representación física.
- El uso de los datos a través de las áreas de aplicación.

El esquema conceptual se construye utilizando la información que se encuentra en la especificación de los requisitos de usuario. Durante todo el proceso de desarrollo del esquema conceptual éste se prueba y se valida con los requisitos de los usuarios. El esquema conceptual es una fuente de información para el diseño lógico de la base de datos.

El esquema conceptual se puede utilizar para que el diseñador transmita al usuario lo que ha entendido sobre la información que ésta maneja. Para ello, se utiliza el modelo entidad-relación [29].

Diseño lógico

El diseño lógico es el proceso de construir un esquema de la información que utiliza el usuario, basándose en un modelo de base de datos específico, independiente del BDMS concreto que se vaya a utilizar y de cualquier otra consideración física.

En esta etapa, se transforma el esquema conceptual en un esquema lógico que utilizará las estructuras de datos del modelo de base de datos en el que se basa el BDMS que se vaya a utilizar, como puede ser el modelo relacional, el modelo de red, el modelo jerárquico o el modelo orientado a objetos. Conforme se va desarrollando el esquema lógico, éste se va probando y validando con los requisitos de usuario.

La normalización es una técnica que se utiliza para comprobar la validez de los esquemas lógicos basados en el modelo relacional, ya que asegura que las relaciones (tablas) obtenidas no tienen datos redundantes.

El esquema lógico es una fuente de información para el diseño físico. Además, juega un papel importante durante la etapa de mantenimiento del sistema, ya que permite que los futuros cambios que se realicen sobre los programas de aplicación o sobre los datos, se representen correctamente en la base de datos.

Tanto el diseño conceptual, como el diseño lógico, son procesos iterativos, tienen un punto de inicio y se van refinando continuamente. Ambos se deben ver como un proceso de aprendizaje, en el que el diseñador va comprendiendo el funcionamiento de la base de datos y el significado de los datos que maneja. El diseño conceptual y el diseño lógico son etapas clave para conseguir un sistema que funcione correctamente. Si el esquema no es una representación fiel de la de la base de datos, será difícil, si no imposible, definir todas las vistas de usuario (esquemas externos), o mantener la integridad de la base de datos. También puede ser difícil definir la implementación física o el mantener unas prestaciones

aceptables del sistema. Además, hay que tener en cuenta que la capacidad de ajustarse a futuros cambios, es un sello que identifica a los buenos diseños de bases de datos. Por todo esto, es fundamental dedicar el tiempo y las energías necesarias para producir el mejor esquema que sea posible.

Diseño físico

En general, el propósito del diseño físico es describir cómo se va a implementar físicamente el esquema lógico obtenido en la fase anterior. Concretamente, en el modelo relacional, esto consiste en:

- Obtener un conjunto de relaciones (tablas) y las restricciones que se deben cumplir sobre ellas.
- Determinar las estructuras de almacenamiento y los métodos de acceso que se van a utilizar para conseguir unas prestaciones óptimas.

IV.1.1 Modelo Conceptual: Modelo entidad-relación

El Modelo entidad relación es una representación de los requerimientos del usuario y los muestra en una serie de objetos llamados entidades, asociaciones y atributos. Las asociaciones pueden ser de tres tipos 1:1, 1:N, N:M.

Una entidad son los objetos del mundo.

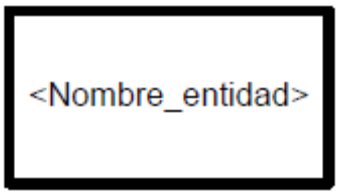

Figura 6. Representación entidad.

Atributo son las características de la entidad.

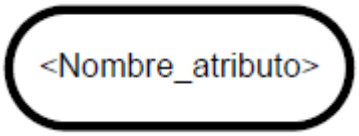

Figura 7. Representación atributo.

Línea. Vínculo para unir entidades y asociaciones.

Figura 8. Representación línea.

Asociación. Tipo de asociación entre entidades. Denota el tipo de elementos que puede tener una relación asociada a otra. El modelo relacional indica el tipo de asociación en base a la asociación existente entre las tablas de la base datos. El tipo de asociación se representa mediante una etiqueta en el exterior de la relación, respectivamente: "1:1", "1:N" y "N:M.

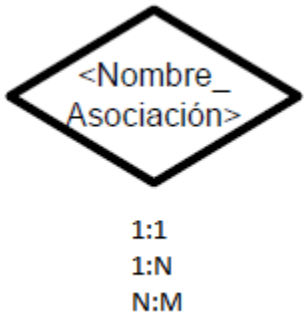

Figura 9. Representación asociación.

Ejemplos de las relaciones.

- Cada esposo (entidad) está casado (relación) con una única esposa (entidad) y viceversa. Es una relación 1:1.

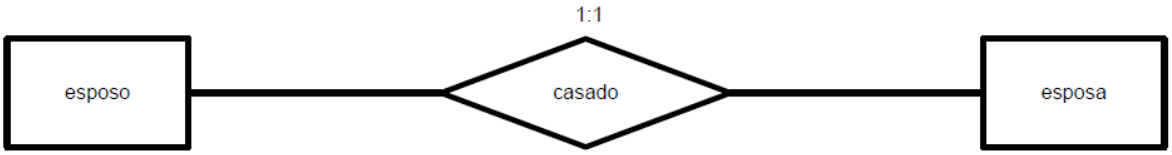

Figura 10. Relación 1:1.

- Una factura (entidad) se emite (relación) a una persona (entidad) y sólo una, pero una persona puede tener varias facturas emitidas a su nombre. Todas las facturas se emiten a nombre de alguien. Es una relación 1:N.

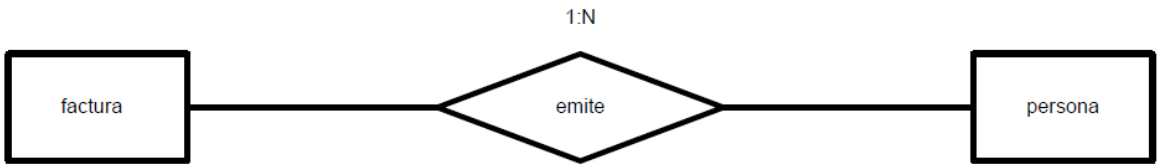

Figura 11. Relación 1:N.

- Un cliente (entidad) puede comprar (relación) varios artículos (entidad) y un artículo puede ser comprado por varios clientes distintos. Es una relación N:M.

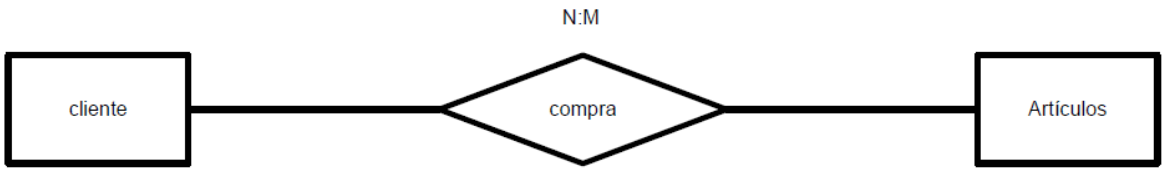

Figura 12. Relación N:M.

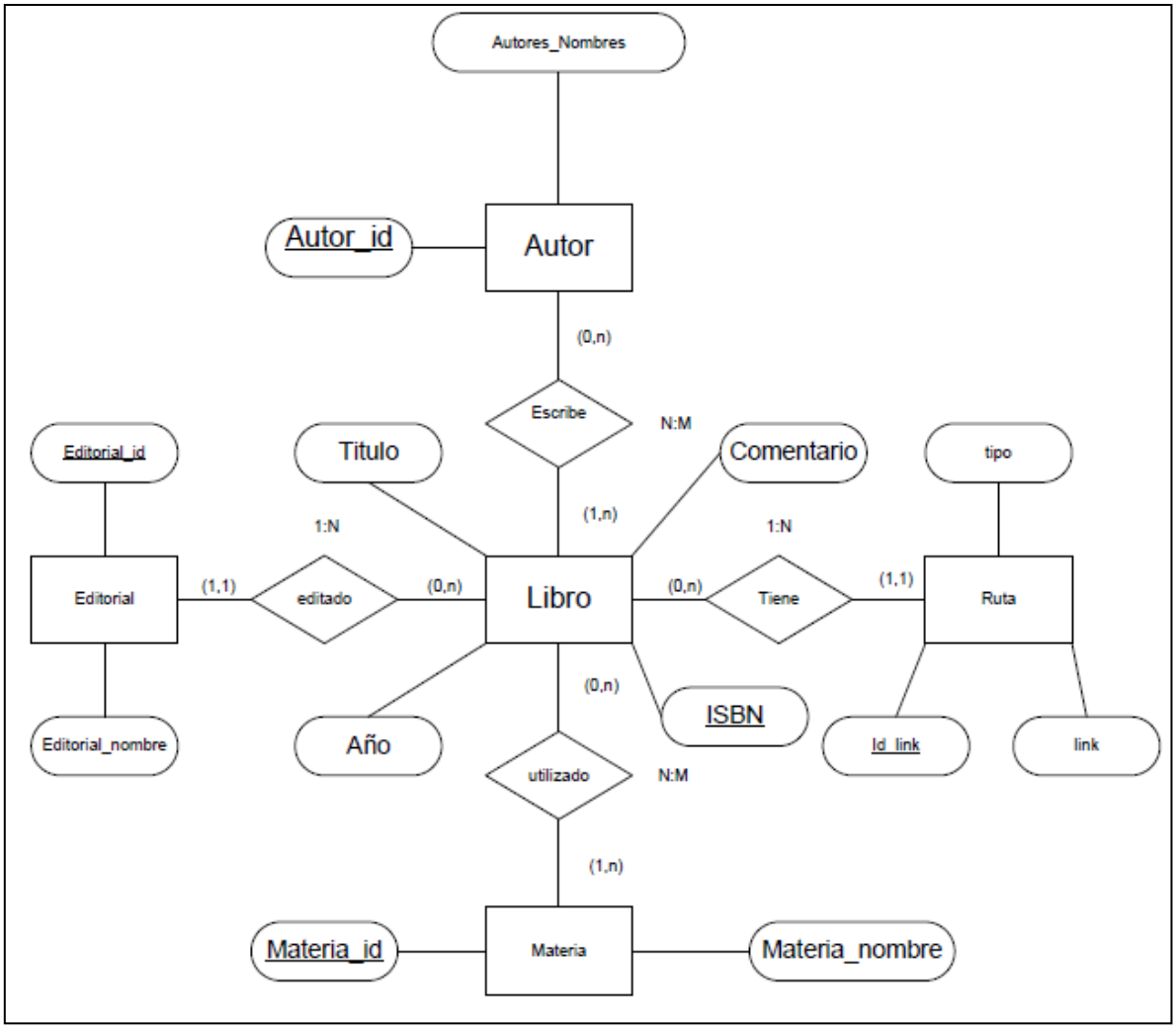

Figura 13. Modelo entidad relación.

IV.1.1.1 Diccionario de datos

El diccionario de datos [28] proporciona información adicional sobre la base de datos. En este se describe las tablas y sus atributos.

Tabla	Columna	Tipo	Precisión	Pk	Comentario
Autor	Autor_id	Int	11	Sí	Autor ID
Autor	Autor_Nombre	Varchar	150		Nombre ó Nombres de los Autores.

Tabla	Columna	Tipo	Precisión	Pk	Comentario
Libro	ISBN	Varchar	20	Sí	ISBN[36](International Standard Book Number (en español, 'número estándar internacional de libro')) es el ID
Libro	Titulo	Varchar	150		Nombre del libro
Libro	Año	Int	4		Año de publicación
Libro	Comentario	Varchar	200		Comentario sobre el libro

Tabla	Columna	Tipo	Precisión	Pk	Comentario
Ruta	Id_link	Int	20	Sí	Ruta ID
Ruta	Link	Varchar	150		Contiene una URL hacia una imagen ó PDF asociado con el libro por ejemplo(http://localhost:37895/ice/imagenes/ imagen.jpg)
Ruta	Tipo	Varchar	150		Se especifica de qué tipo de documento se trata.

Tabla	Columna	Tipo	Precisión	Pk	Comentario
Materia	Materia_id	Int	11	Sí	Materia ID
Materia	Materia_Nombre	varchar	50		Nombre de la materia.

Tabla	Columna	Tipo	Precisión	Pk	Comentario
Editorial	Editorial_id	Int	11	Sí	Editorial ID
Editorial	Editorial_Nombre	varchar	50		Nombre de la editorial

Tabla	Columna	Tipo	Precisión	Pk	Comentario
Escribe	#Autor_id	Int	11		fk
Escribe	#ISBN	Varchar	20		fk

Tabla	Columna	Tipo	Precisión	Pk	Comentario
Editado	#Editorial_id	Int	11		fk
Editado	#ISBN	Varchar	20		fk

Tabla	Columna	Tipo	Precisión	Pk	Comentario
Tiene	#id_link	Int	11		fk
Tiene	#ISBN	Varchar	20		fk

Tabla	Columna	Tipo	Precisión	Pk	Comentario
Utilizado	#Materia_id	Int	11		fk
Utilizado	#ISBN	Varchar	20		fk

IV.1.1.2 Diseño lógico

En el modelo lógico convertimos al modelo entidad relación en tablas con un formato que se adecue al BDMS.

- Autor(#Autor_id, Autores_nombres)
- Libro(#ISBN,Titulo, Comentario, Año)
- Editorial(#Editorial_id, Editorial_nombre)
- ruta(#id_link, tipo,link)
- Materia(#Materia_id, Materia_nombre)
- Escribe(#Autor_id, #ISBN)
- Editado(#Editorial_id, #ISBN)
- Tiene(#id_link, #ISBN,)
- Utilizado(#ISBN, #Materia_id)

Los atributos subrayados indican que se trata de la llave primaria de la relación.

Estas relaciones ya están en tercera forma normal.

Columna	ISBN	AÑO	TITULO	COMENTARIOS	EDITORIAL	RUTA
TIPO LLAVE	PK				FK	FK
NN/U	NN/U	NN	NN	NN	NN	NN

IV.1.1.3 Creación de la base de datos y captura de datos

Desde phpMyAdmin(en el Anexo D se describe como trabajar con phpMyAdmin) se crea la base de datos, después desde el administrador se va creando cada tabla con cada uno de sus atributos.

Se crea la base de datos.

Figura 14. Creación de la base de datos.

Se crean las tablas.

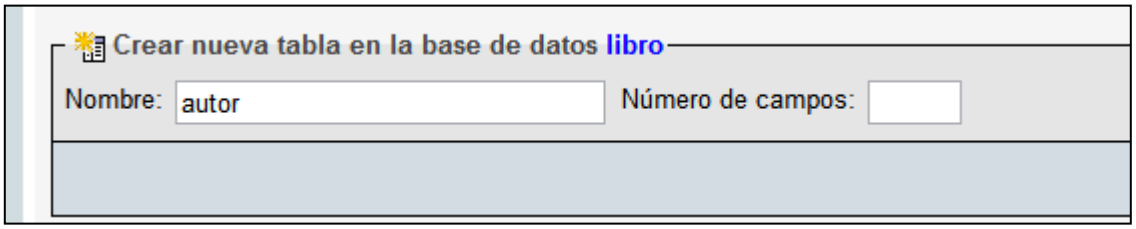

Figura 15. Creación de la tabla.

Se definen los atributos.

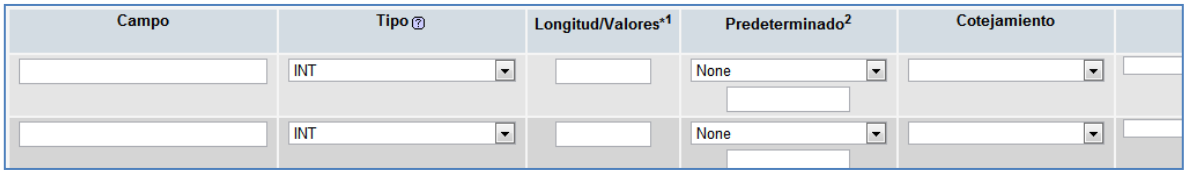

Figura 16. Configuración de campos.

Una vez creada la base de datos, la captura de datos se realiza por el manejador:

Seleccionamos la pestaña "Insertar", en esta opción dependiendo de la tabla se mostrara un formato para introducir los datos a la base de datos. Ver figura 17.

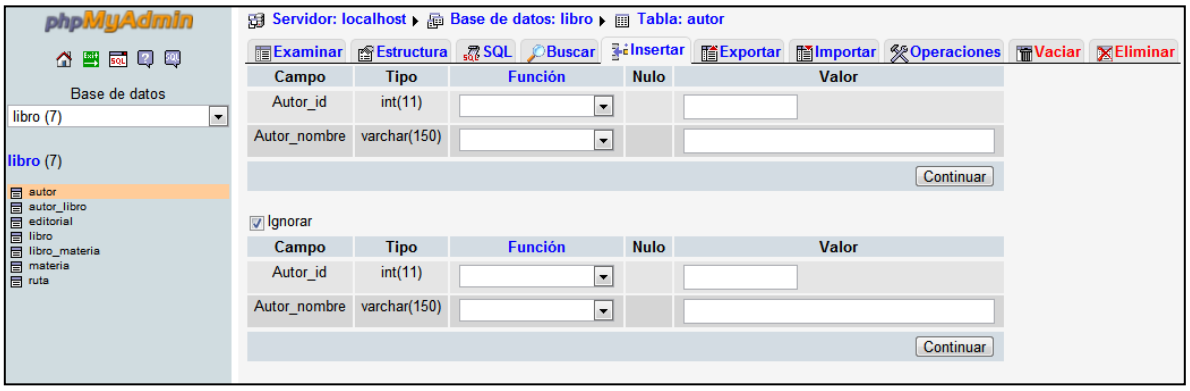

Figura 17. Inserción de valores dentro de la base de datos.

Los valores agregados a la base de datos pueden ser revisados desde phpMyAdmin, en la figura 18 podemos observar los datos capturados en la base de datos en la página Web.

Figura 18. Datos vistos desde el phpMyAdmin.

IV.2 Diseño de la página Web

Para comenzar con el diseño del templete Web, se hace un esquema general de cómo se verá, como el que se observa en la figura 19, el esquema se irá formando con las etiquetas DIV de XHTML después en base a CSS le daremos la forma que sea necesaria.

El diseño del templete Web se dividirá en 5 partes:

- La división Header estará en la parte de arriba, está contendrá solo el logo de la página.
- La división Menú como su nombre lo dice contendrá los enlaces hacia todas las vistas posibles en el sitio Web.

- Las “sub_ventana_uno” contendrán la información general de la vista correspondiente.

- La “sub_ventana_dos” mostrará casos particulares seleccionados en la división uno.

- El footer no contendrá nada solo delimitara la página.

Figura 19. Esquema de cómo será la vista de la página Web.

El templete lo podemos en parte generar mediante Netbeans, pero como el templete que mostrará la información tiene la estructura descrita en la figura 19, esta estructura no es como ninguna de las que genera Netbeans debido a esto el templete se tendrá que elaborar como se muestra a continuación.

IV.2.1 Procedimiento para generar un Templete Web desde Netbeans

1. Creamos un archivo JavaServer Faces>Facelets Templete.

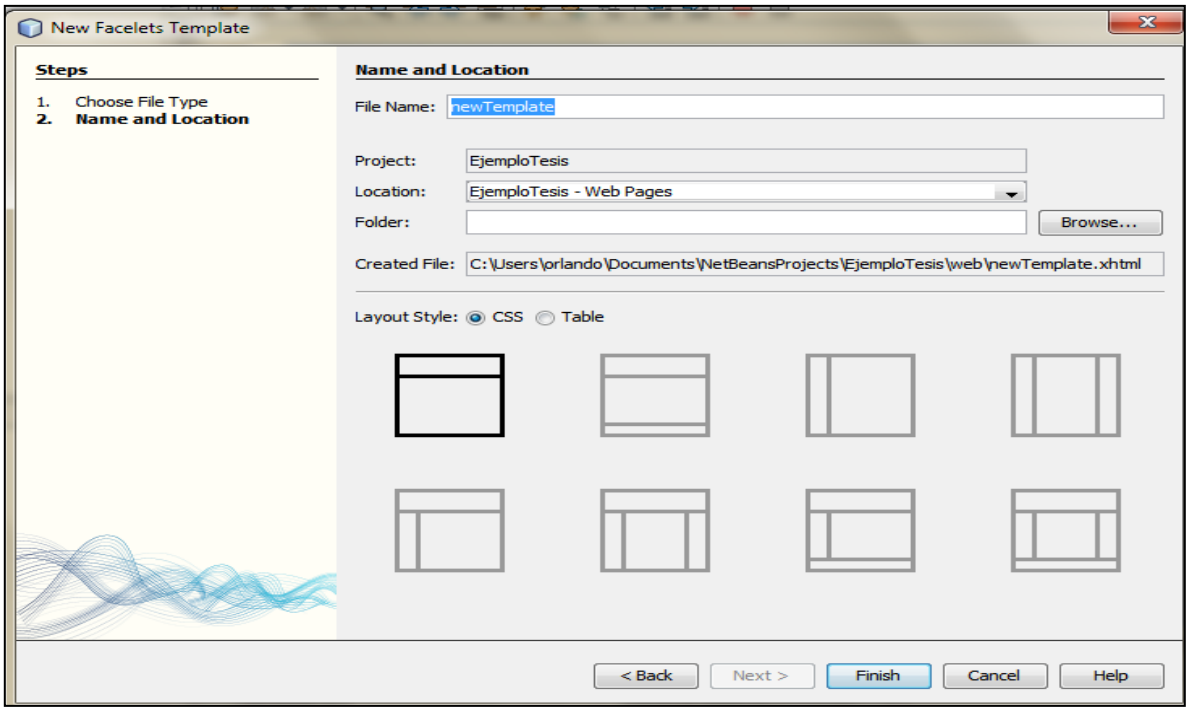

Figura 20. Creación de templete.

Tendremos que agregar al templete generado las divisiones ya mencionadas.

```
<div id="header"></div>

<div id="menu"></div>

<div class="sub_ventana_uno"></div>

<div class="sub_ventana_dos"></div>

<div id="footer"></div>
```

Estas divisiones a su vez tienen unas etiquetas que son las referencias (<ui:insert name=" ">Content</ui:insert>) para que se pueda insertar el contenido de la base de datos de las diferentes vistas mediante ICEfaces.

```
<div id="menu"><ui:insert name="menu">Content</ui:insert></div>

<div class="sub_ventana_uno"><ui:insert name="body">Content</ui:insert></div>

<div class="sub_ventana_dos"><ui:insert name="acc">Content</ui:insert></div>
```

Con lo cual obtenemos.

Templete.xhtml.

```
<?xml version='1.0' encoding='UTF-8' ?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"

xmlns:ui="http://java.sun.com/jsf/facelets"

xmlns:f="http://java.sun.com/jsf/core"

xmlns:h="http://java.sun.com/jsf/html"

xmlns:ice="http://www.icesoft.com/icefaces/component">

   <h:head>

      <title>library/bookcase</title>

       <h:outputScript name="jsf.js" library="javax.faces"/>


       <link href="default.css" rel="stylesheet" type="text/css" />
```

```
   </h:head>
  <h:body>
<div id="header"></div>
<div id="menu"><ui:insert name="menu">Content</ui:insert></div>
<div class="sub_ventana_uno"><ui:insert name="body">Content</ui:insert></div>
<div class="sub_ventana_dos"><ui:insert name="acc">Content</ui:insert></div>
<div id="footer"></div>
</h:body>
</html>
```

Ya que tenemos el archivo XHTML con las divisiones correspondientes cada una de las divisiones tomará forma mediante el uso del archivo CSS default.css.

El default.css es un archivo externo que da forma a las divisiones y al menú, en seguida pongo algunos fragmentos del código con su respectiva explicación.

El margin y padding hacen que no quede un pequeño margen, el cual por defecto se encuentra entre la parte de arriba del navegador y el contenido de la página.

El #content ima hace que una imagen tome las características descritas por ejemplo forma un margen 2px esto es un marco negro.

El sub_ventana_uno y sub_ventana_dos ponen las divisiones en izquierda y derecha respectivamente.

```
* {
        margin: 0;


        padding: 0;


}
#content img {
   text-align: center;
   vertical-align: middle;
   border-right-style: groove;
   border-left-style: groove;
   border-bottom-style: groove;
   border-top-style: groove;
   margin-left: 2px;
   margin-bottom: 2px;
   margin-top: 2px;
}
.sub_ventana_uno {
        float: left;
}
.sub_ventana_dos {
        float: right;
}
 .sub_ventana_uno,  .sub_ventana_dos {
```

```
        width: 250px;
}
/* Header */

#header {
        width: 960px;

        height: 92px;
        margin: 0 auto;
}
/* footer */
#footer {
        height: 100px;
        padding: 20px;
        background: #DDDDDD;
        border-top: 1px solid #999999;
}
#footer p {
        margin: 0;
        text-align: center;      }
```

IV.2.2 Configuración del menú de navegación

El menú tiene dos aspectos el primero el visual que es configurado también XHTML y CSS dando la forma que deseamos.

Figura 21. Menú dentro de la vista.

El segundo aspecto es la función de hacer link a las diversas vistas de la aplicación, lo cual se configura mediante un archivo llamado faces-config.xml. Éste cuenta con un modo gráfico donde iremos haciendo enlaces a las diversas vistas, cada enlace toma un nombre el cual se asocia con el link en la vista correspondiente.

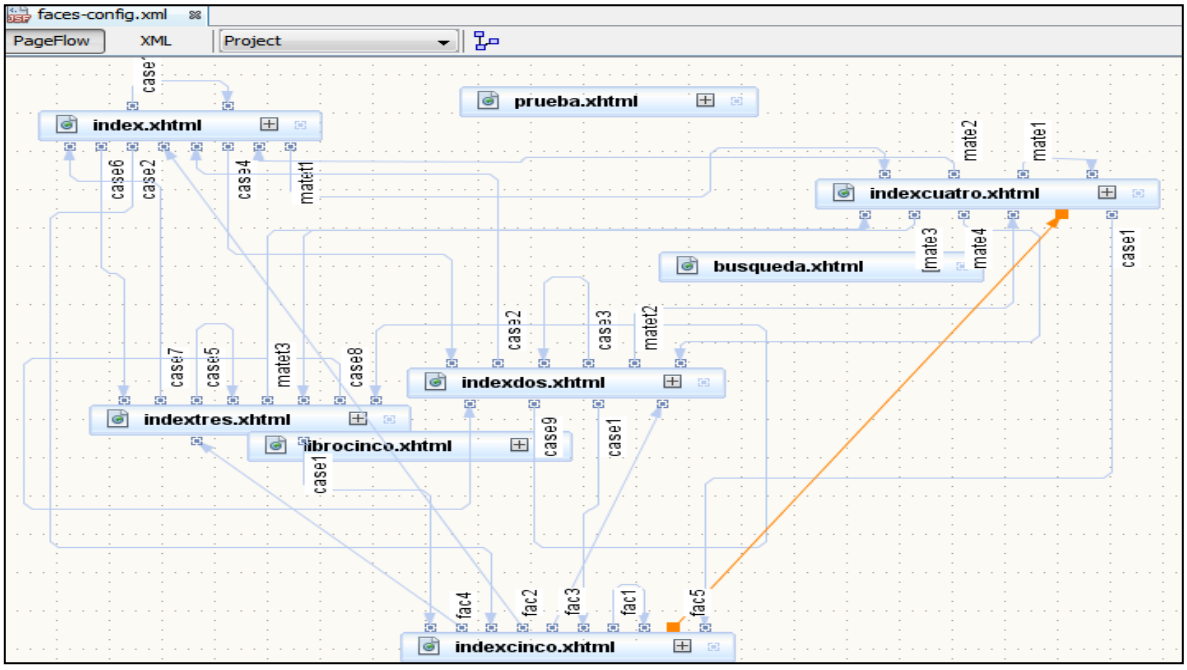

Figura 22. Modo gráfico de la configuración de links faces-config.xml.

Al configurar un link en el modo gráfico éste se refleja en el archivo .xml, después en cada vista se asocia el nombre correspondiente en el archivo .xml.

Fragmento del archivo “faces-config.xml”.

```
<faces-config version="2.0"
   xmlns="http://java.sun.com/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-facesconfig_2_0.xsd">


<navigation-rule>
      <from-view-id>/index.xhtml</from-view-id>
      <navigation-case>
         <from-outcome>case1</from-outcome>
         <to-view-id>/index.xhtml</to-view-id>
      </navigation-case>
```

Cada una de las vistas (index.xhtml, indexdos.xhtml, indextres.xhtml, indexcuatro.xhtml, indexcinco.xhtml, búsqueda.xhtml) se une con las otras, con esto se forman los links como se observa en la figura 22, cada uno de los link toma un nombre, ese nombre en la respectiva vista es configurado de la siguiente manera:

El siguiente código es un fragmento de código que se encuentra dentro de la vista index.xhtml, en este código solo se ejemplificara uno de los casos para esta vista que es case1 como se puede ver faces-config.xml solo regresa sobre el mismo.

```
<ui:define name="menu">
        <ul>
           <li class="active">
              <f:view>
                 <ice:form id="iniciouno" >
                    <ice:commandLink
                       id="submit"
                       action="case1"
                       >
                       <b><ice:outputText value="Inicio"/></b>

                    </ice:commandLink>
                 </ice:form>
              </f:view>
           </li>
```

IV.3 Interfaz entre la base de datos y la aplicación Web

Para poder mostrar los datos dentro de la página Web, necesitamos un proceso el cual hace que la base de datos formada en el modelo relacional sea vista como objetos, con la finalidad de ser mostrada mediante ICEfaces (ICEfaces es un tecnología Java orientada a objetos), es decir, haciendo una analogía se puede decir que el modelo relacional y el orientado a objetos son dos idiomas distintos, los cuales no se entienden entre ellos, con el proceso descrito a continuación, la base de datos será traducida del mundo relacional al mundo de los objetos. Con esto se podrán recuperar los datos de la base de datos en forma de objetos los cuales podrán ser desplegados en la página Web.

IV.3.1 Configuración de la base de datos con la página Web

Para la comunicación entre la base de datos y la aplicación Web se configuró mediante el uso de Hibernate.

Hibernate abre un canal de comunicación entre la base de datos y la aplicación Web.

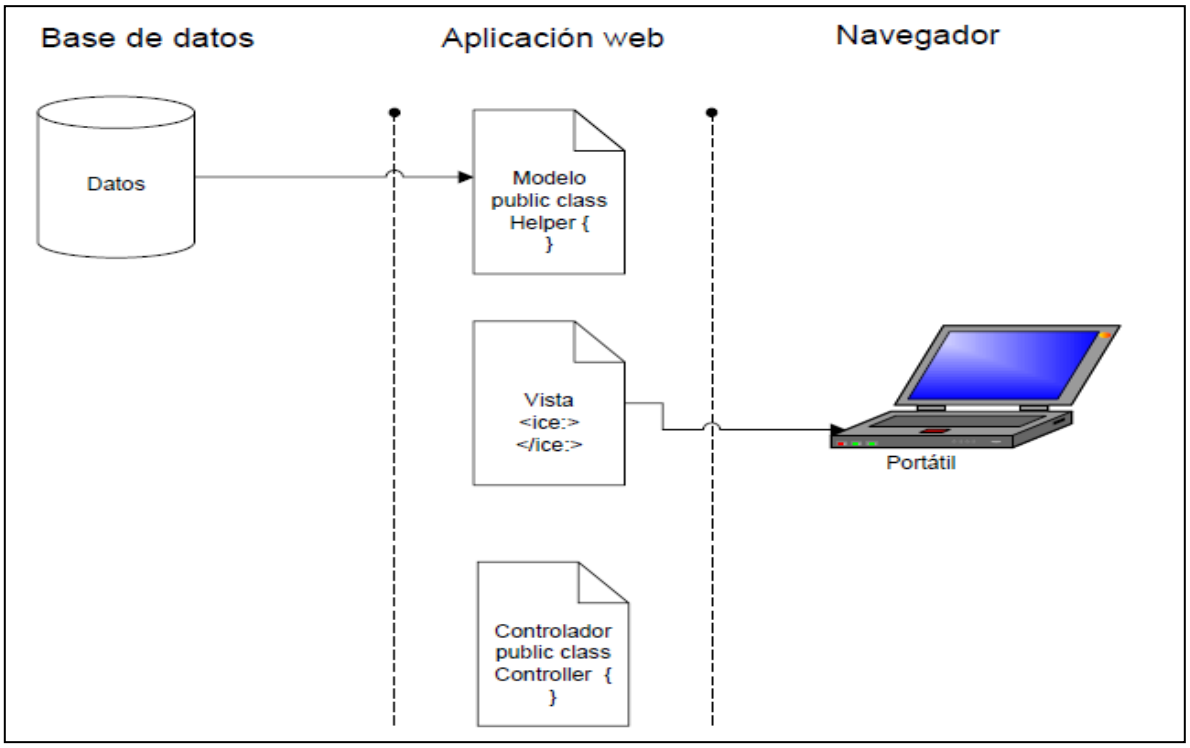

Figura 23. Modelo Vista Controlador.

Conexión entre la base de datos y la aplicación.

IV.3.1.1 Hibernate usa dos tipos de configuraciones de archivos

Mediante el uso de Netbeans se configura de manera sencilla la conexión con la base de datos siendo administrada con las herramientas del framework Hibernate. A continuación se describe paso a paso como configurar mediante el uso de Netbeans la conexión con la

base de datos mediante los POJOS y archivos de Mapeo desde los cuales se obtendrán las consultas.

IV.3.1.2 Procedimiento para generar un nuevo proyecto en Netbeans

1. Se crea un nuevo proyecto Java “Web>Web Application”.

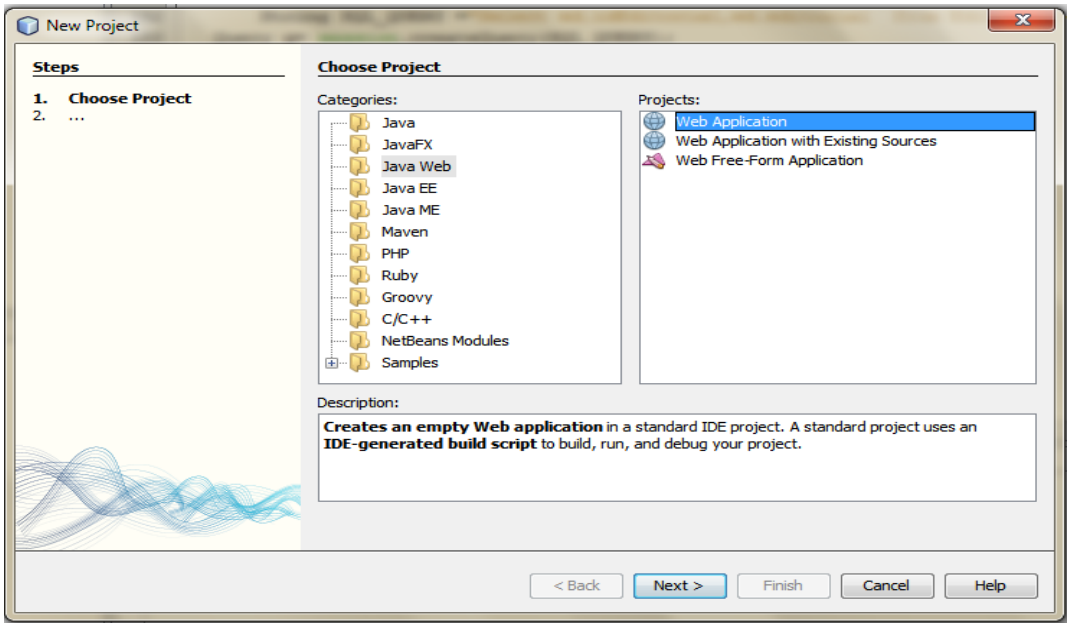

Figura 24. Creación de un nuevo proyecto.

2. Presionar next y escribir el nombre al nuevo proyecto, en este caso “EjemploTesis”.

3. El siguiente paso es seleccionar el servidor y la versión de Java que vamos a utilizar que son GlassFish V3 Domian y Java EE 6 Web.

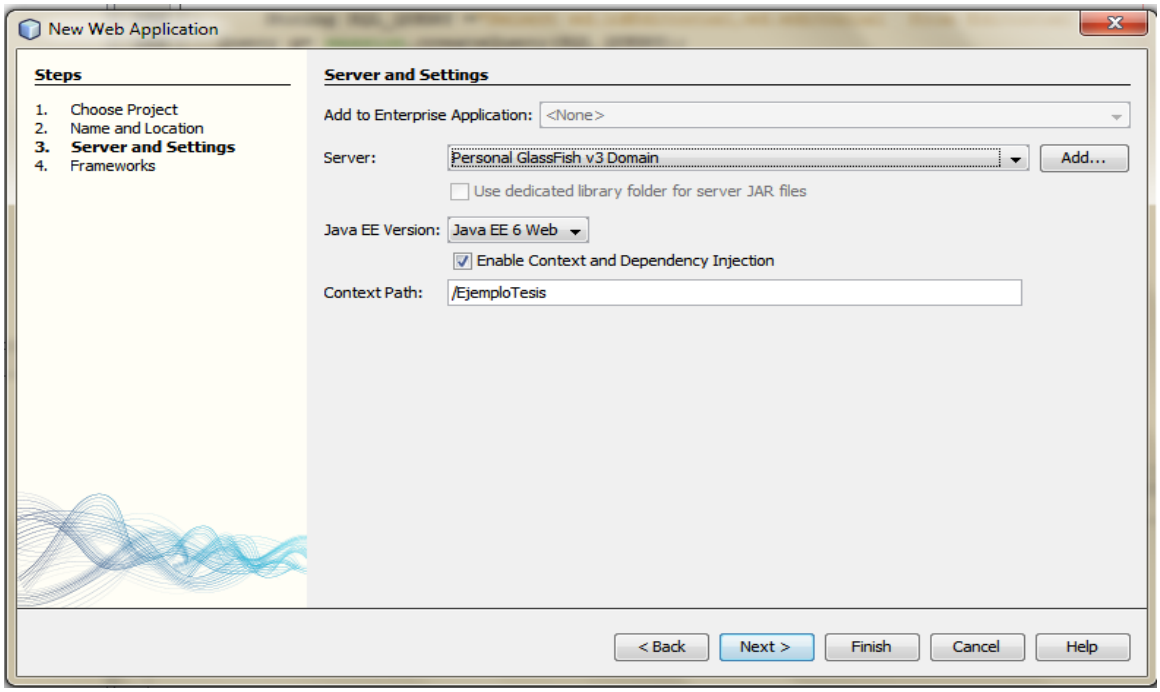

Figura 25. Selección del servidor y la versión de Java a utilizar.

4. Seleccionar ICEfaces que tiene las librerías para mostrar los datos dentro del templete, también se selecciona Hibernate 3.2.5 y selecciona la base de datos que se va a utilizar y presionar Finish.

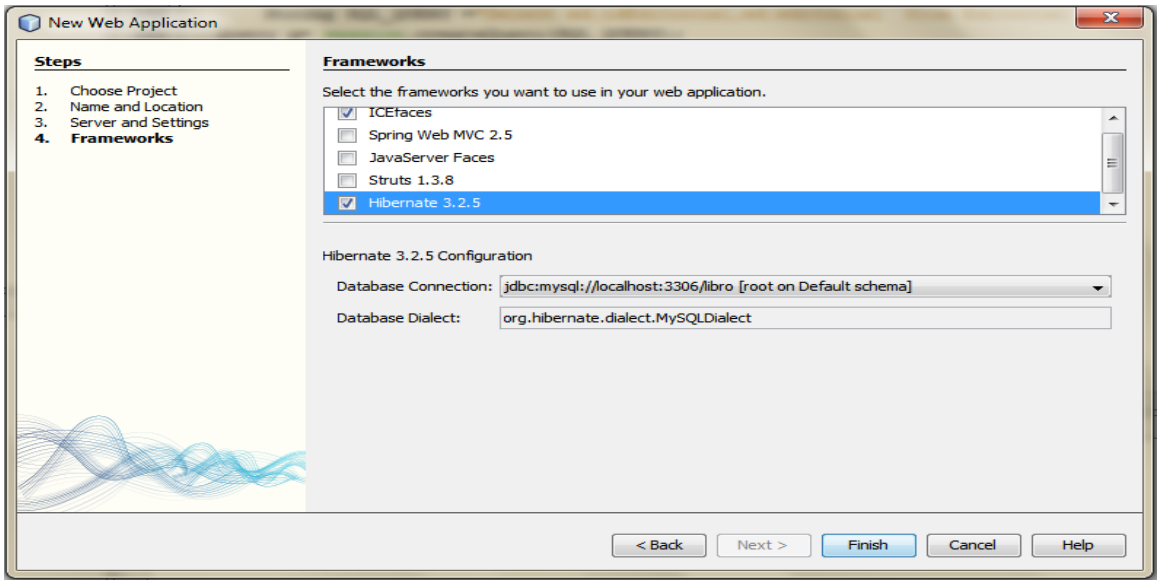

Figura 26. Selección de los frameworks a utilizar.

Con el procedimiento descrito anteriormente se genera el nuevo proyecto y por defecto es creado el archivo global de configuración (hibernate.cfg.xml) de la base de datos.

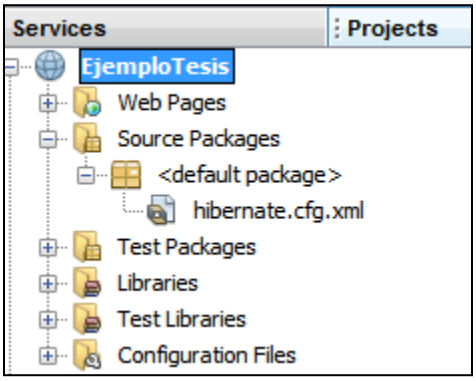

Figura 27. Estructura creada con Netbeans.

Al dar click en la pestaña de "Libraries" se puede ver el conector Java para la conexión a la base de datos, las librerías de Icefaces, Hibernate, la versión del JDK de Java a utilizar además de el Servidor seleccionado GlassFish v3 Domian.

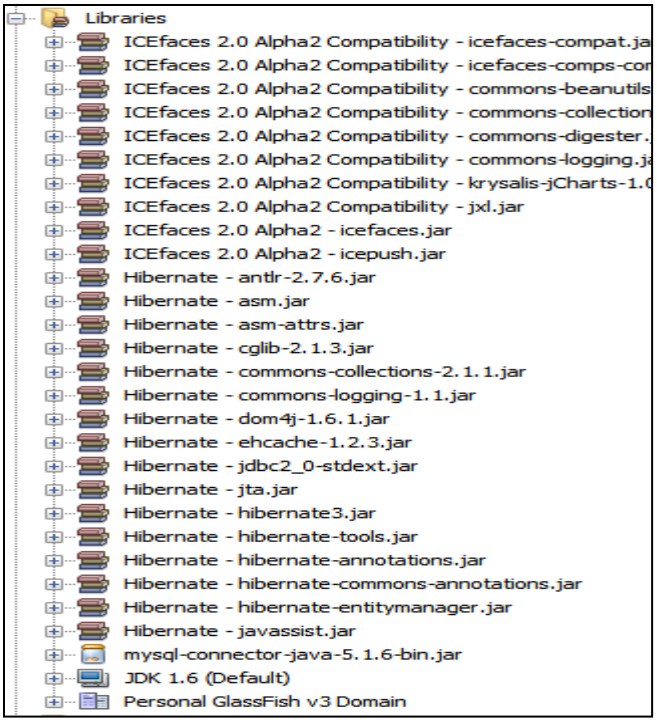

Figura 28. Librerías.

Ya con el proyecto con sus correspondientes librerías se puede comenzar a desarrollar.

IV.3.1.3 Primera parte de la configuración de Hibernate (Archivo de Configuración global)

El archivo de configuración global (hibernate.cfg.xml) contiene declaraciones para la configuración de la base de datos y su comportamiento.

"hibernate.cfg.xml" en este archivo se configuran las siguientes propiedades:

- hibernate.dialect especifica el tipo de manejador de base de datos que se está utilizando, en este caso Mysql y le permita a Hibernate generar SQL optimizado para una BD relacional en particular (en el anexo B se puede ver las diferentes bases de datos con las que puede trabajar Hibernate).
- Donde se encuentra el driver de la base de datos.
- Donde se encuentra la base de datos.
- El usuario de la base de datos.
- La contraseña de la base de datos.
- hibernate.show_sql pone en alto el registro de depuración de las instrucciones SQL.
- hibernate.current_session_context_class establecer el valor del thread para permitir la gestión automática de Hibernate en el contexto de sesión.
- Las tablas mapeadas.

hibernate.cfg.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration
DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

  <session-factory>

    <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>

    <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
```

```
 <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/libro</property>
   <property name="hibernate.connection.username">root</property>
   <property name="hibernate.connection.password">*********</property>
   <property name="hibernate.show_sql">true</property>
   <property name="hibernate.current_session_context_class">thread</property>
   <mapping resource="tesis/Editorial.hbm.xml"/>
   <mapping resource="tesis/Libro.hbm.xml"/>
   <mapping resource="tesis/Materia.hbm.xml"/>
   <mapping resource="tesis/Ruta.hbm.xml"/>
   <mapping resource="tesis/Autor.hbm.xml"/>
  </session-factory>
</hibernate-configuration>
```

IV.3.1.3.1 Manejo de sesión Hibernate

Para realizar una consulta se necesita primero que Hibernate arranque [26].

Dicho arranque incluye construir un objeto SessionFactory global, y almacenarlo en algún lugar de fácil acceso para el código de la aplicación (para eso se crea HibernateUtil).

Una SessionFactory puede abrir nuevos objetos Session. Cada objeto Session representa una "unidad de trabajo" de un solo Thread. El código de SessionFactory es thread-safe, y es instanciado sólo una vez.

Para esto creamos una clase que nos ayude a inicializar y acceder a sesión Factory de Hibernate para obtener un objeto “session”.

La clase llama configurate() y carga el archivo de configuración global hibernate.cfg.xml y después construye el “sessionFactory” para obtener el objeto “session”.

Para crear la clase de manejo de sesión Hibernate y Netbeans tienen como predeterminada la creación de la clase HibernateUtil.

IV.3.1.3.2 Procedimiento para generar la clase de manejo de sesión "HibertateUtil.java"

1. En el proyecto se creó un nuevo archivo, vamos a la carpeta de Hibernate y seleccionamos crear una clase "HibertateUtil.java".

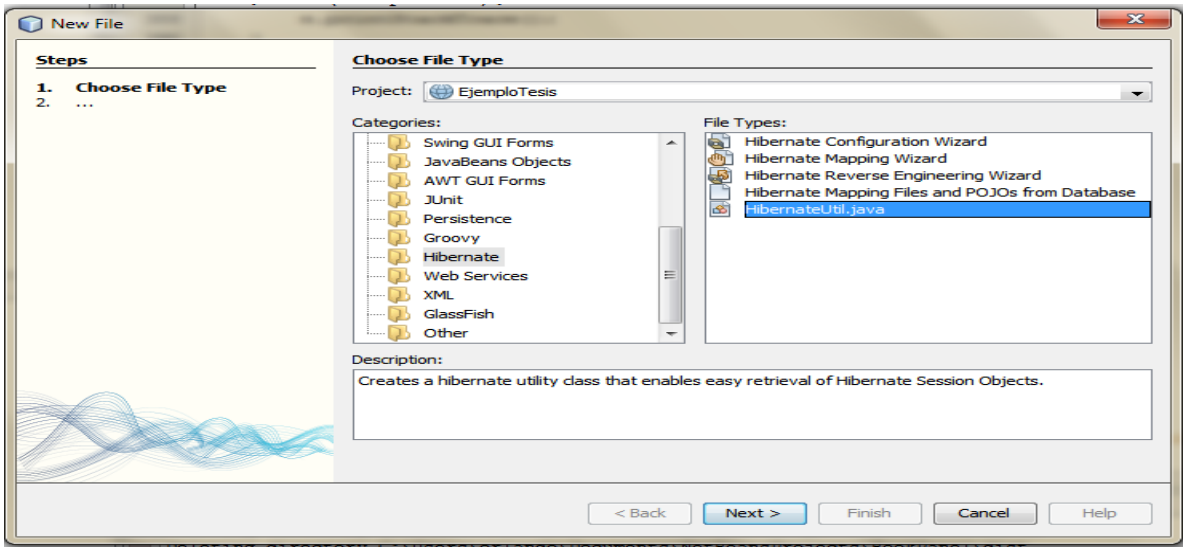

Figura 29. Creación del archivo de sesión.

2. Presionamos finish y se genera la clase HibertateUtil.java.

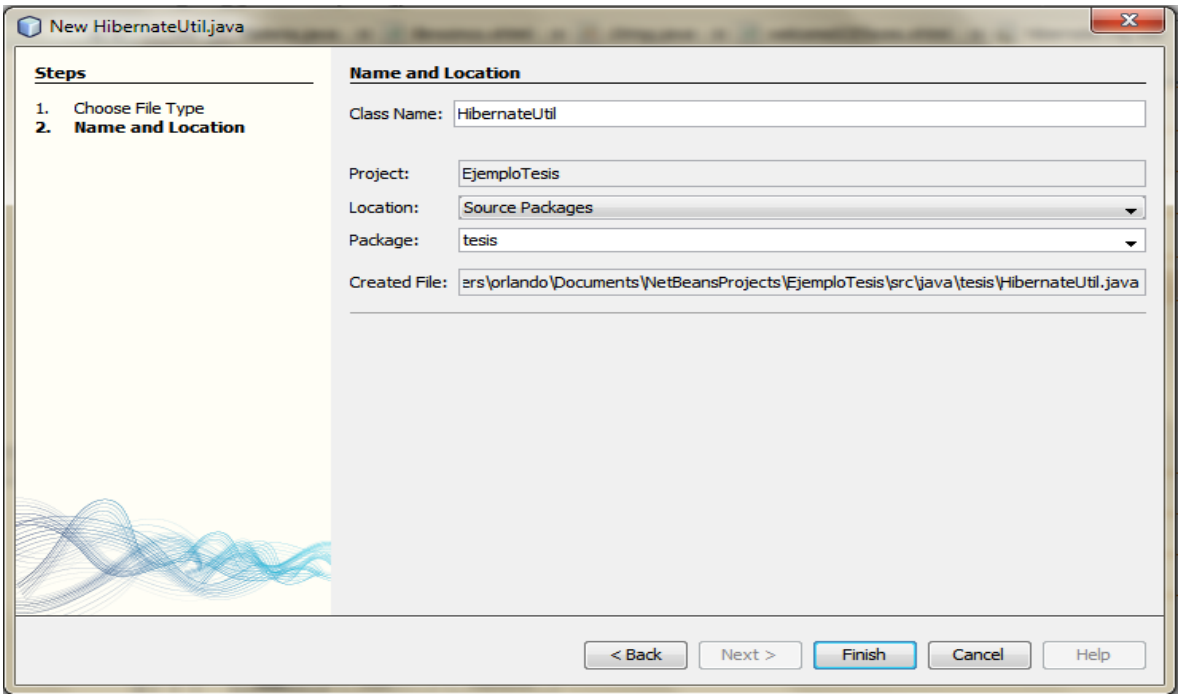

Figura 30. Creación del archivo de sesión.

En la figura 31 podemos observar la clase generada.

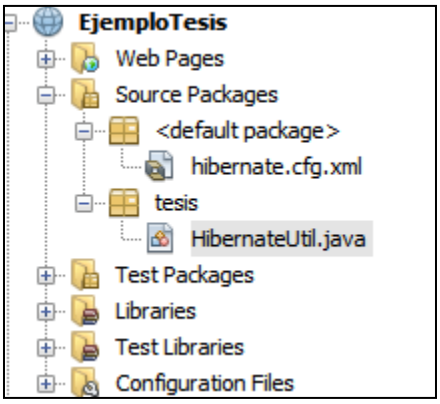

Figura 31. Estructura creada con Netbeans.

La clase “HibertateUtil.java”.

```
package tesis;
import org.hibernate.cfg.AnnotationConfiguration;
import org.hibernate.SessionFactory;


/**
 * Hibernate Utility class with a convenient method to get Session Factory object.
 *
 * @author orlando
 */
public class HibernateUtil {
    private static final SessionFactory sessionFactory;



 static {
        try {
            // Create the SessionFactory from standard (hibernate.cfg.xml)
            // config file.
            sessionFactory = new AnnotationConfiguration().configure().buildSessionFactory();
        } catch (Throwable ex) {
            // Log the exception.
            System.err.println("Initial SessionFactory creation failed." + ex);
            throw new ExceptionInInitializerError(ex);
```

```
        }
    }
    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }
}
```

IV.3.1.4 La segunda configuración que utiliza Hibernate es la de crear las clases POJOS y los archivos de mapeo (.xml)

Hibernate realiza un procedimiento para ver la base de datos relacional como objetos, esto con el fin de facilitar que los datos puedan ser mostrada en la página Web, dado que los datos desplegados ahí necesariamente son objetos. El procedimiento que hace Hibernate es la creación de POJOS (plain old Java object) y archivos de Mapeo.

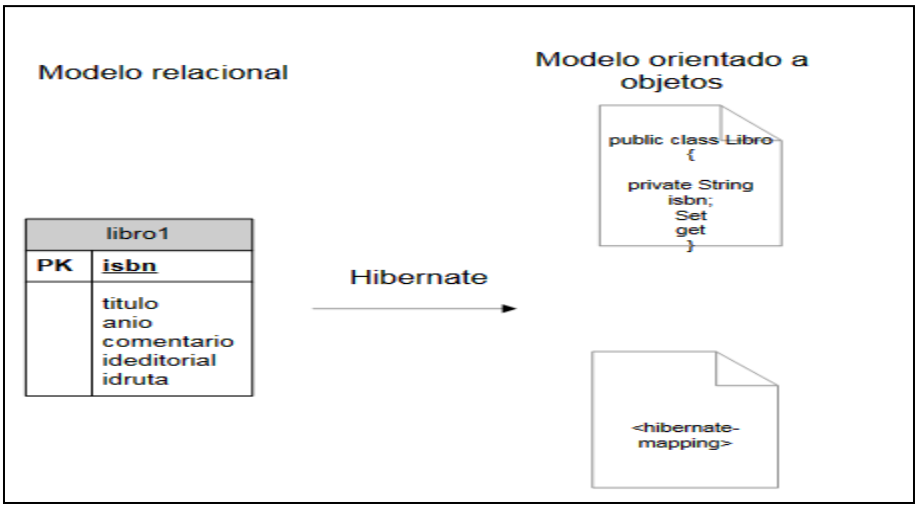

Figura 32. Esquema General trasformación del modelo relacional al modelo orientado a objetos.

Un POJO [17] es la representación de los datos en una clase Java la cual está estructurada como un bean, es decir, cada una de las tablas de la base de datos es representada dentro de una clase Java, las columnas en las tablas serán representadas en la clase Java como un bean, es decir, una variable con sus setters y getters simples, los cuales sirven para recuperar (visualizar los datos en un página de Internet) y escribir (dentro de la base de datos) los datos dentro de la vista.

Los POJOS son objetos de un solo thread y corta vida, que contienen "estado" persistente. La única característica notable que tienen, es que están asociados con una sesión. En cuanto la sesión se cierra, serán desprendidos y quedarán listos para ser usados en cualquier capa de la aplicación.

Hibernate necesita saber cómo cargar y almacenar objetos de la clase persistente. Aquí es donde entra en juego el archivo de mapeo. Éste le dice a Hibernate a qué tabla y en qué base de datos tiene que acceder, además de qué columnas de dicha tabla tiene que utilizar.

El término mapeo se refiere a "mapeo objeto/relacional" (OMR por sus siglas en inglés), se refiere a la técnica de "mapear" la representación de los datos desde un modelo relacional hacia un modelo de objetos.

IV.3.1.4.1 Creación de POJOS y archivos de Mapeo

Los POJOS y archivos de Mapeo son generados con Netbeans mediante el uso de Hibernate de manera rápida.

Antes de explicar el procedimiento para elaborar los POJOS y Archivos de Mapeo se tiene que crear el archivo Reverse Engineering el cual indicara las tablas a mapear.

IV.3.1.4.2 Ingeniería inversa

Para generar las clases .java y los archivos de mapeo .hbm.xml Hibernate usa una estrategia llamada reverse engineering () en la cual se asignan nombres de las tablas, columnas y foreignkeys a las clases, propiedades y asociaciones. Se utiliza también para las asignaciones de tipos SQL a tipos de Hibernate.

IV.3.1.4.2.1 Procedimiento para generar el archivo Reverse Engineering mediante Netbeans

Se crea un nuevo archivo en la carpeta de "Hibernate >Hibernate Reverse Engineering hibernate.reveng.xml".

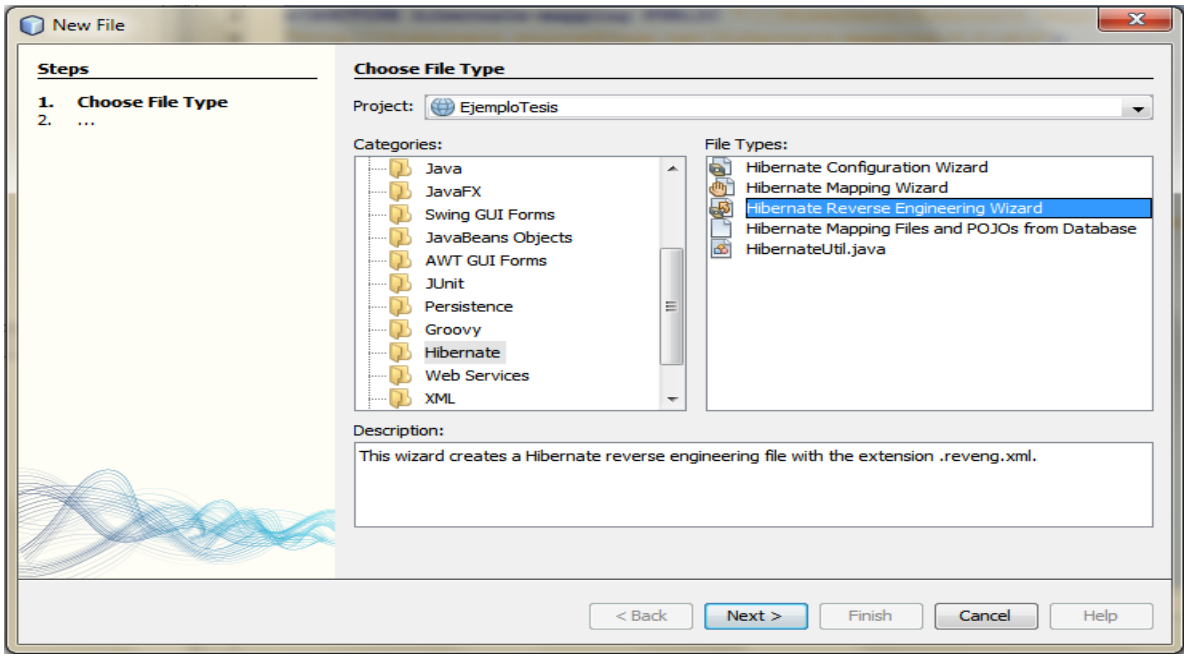

Figura 33. Creación archivo Reverse Engineering.

2. Se escribe el nombre por defecto "hibernate.reveng" y la dirección donde estará dentro del proyecto.

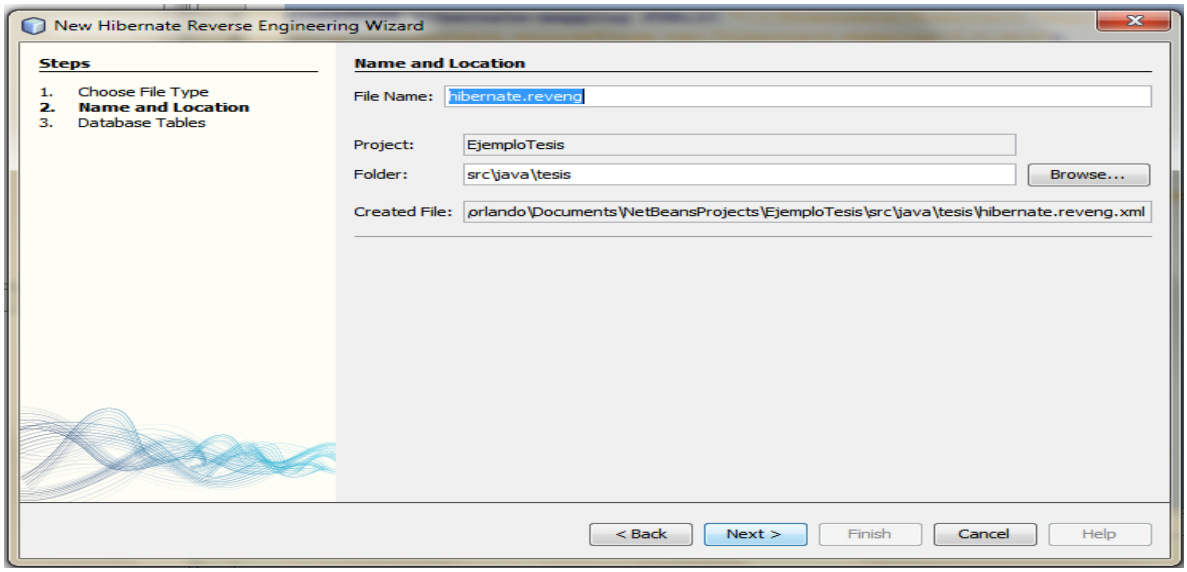

Figura 34. Localización archivo Reverse Engineering.

3. Se seleccionan las tablas de las cuales queremos obtener los POJOS (.java y .hbm.xml).

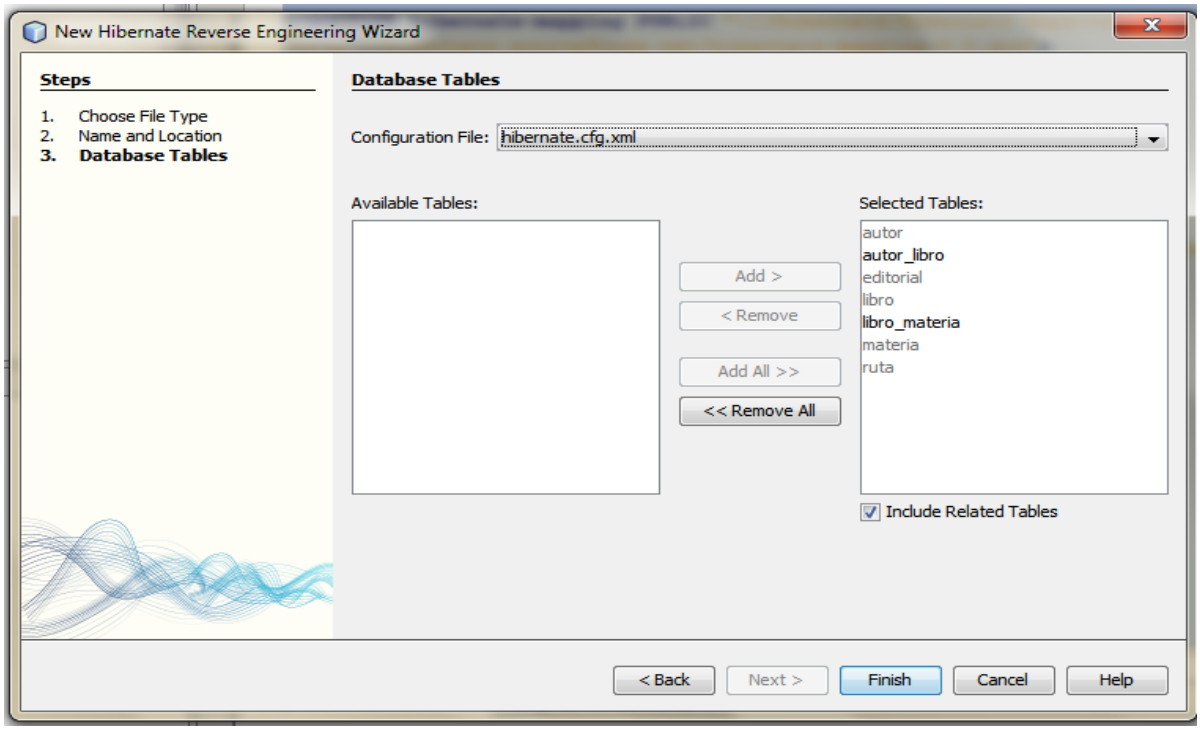

Figura 35. Selección de las tablas de la base de datos.

Con el procedimiento descrito anterior obtenemos el Hibernate Reverse Engineering que contienen referencia las tablas a mapear.

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE hibernate-reverse-engineering PUBLIC "-//Hibernate/Hibernate Reverse Engineering DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-reverse-engineering-3.0.dtd"><hibernate-reverse-engineering>

  <schema-selection match-catalog="libro"/>

  <table-filter match-name="libro"/>

  <table-filter match-name="editorial"/>

  <table-filter match-name="autor"/>

  <table-filter match-name="autor_libro"/>



  <table-filter match-name="libro_materia"/>

  <table-filter match-name="materia"/>

  <table-filter match-name="ruta"/>

</hibernate-reverse-engineering>
```

Ya que tenemos “Reverse Engineering” ya se puede construir los archivos de Mapeo y los POJOS.

IV.3.1.4.3 Procedimiento para elaborar los POJOS y archivos de Mapeo mediante Netbeans

1. El siguiente paso es crear un nuevo archivo “Hibernate>Hibernate Mapping Files and Pojos from database”.

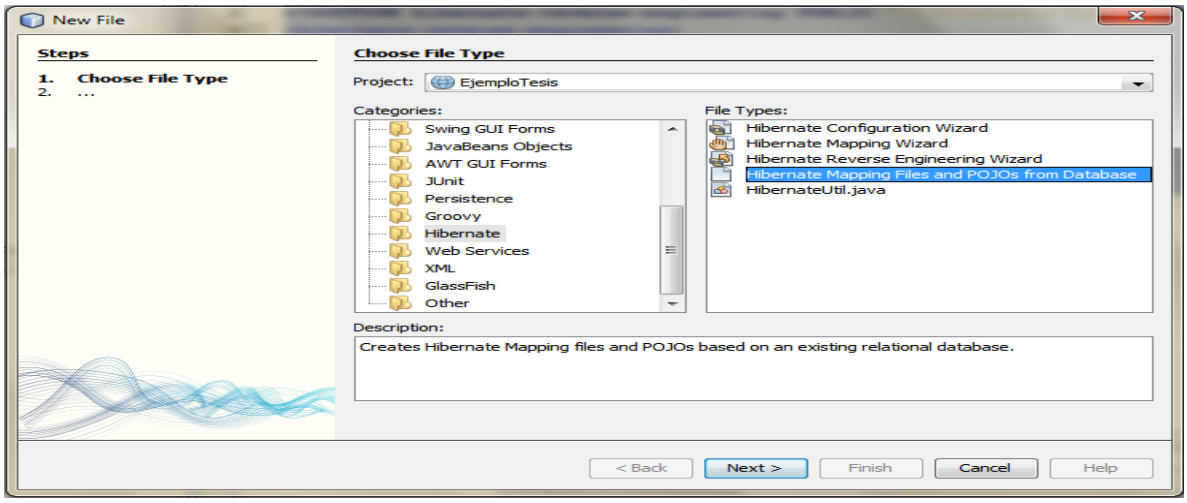

Figura 36. Nuevo archivo de POJOS.

2. Por defecto aparece el archivo de configuración global "hibernate.cfg.xml" y el "hibernate.reveng.xml", se selecciona el paquete donde estarán los .java y ".hbm.xml".

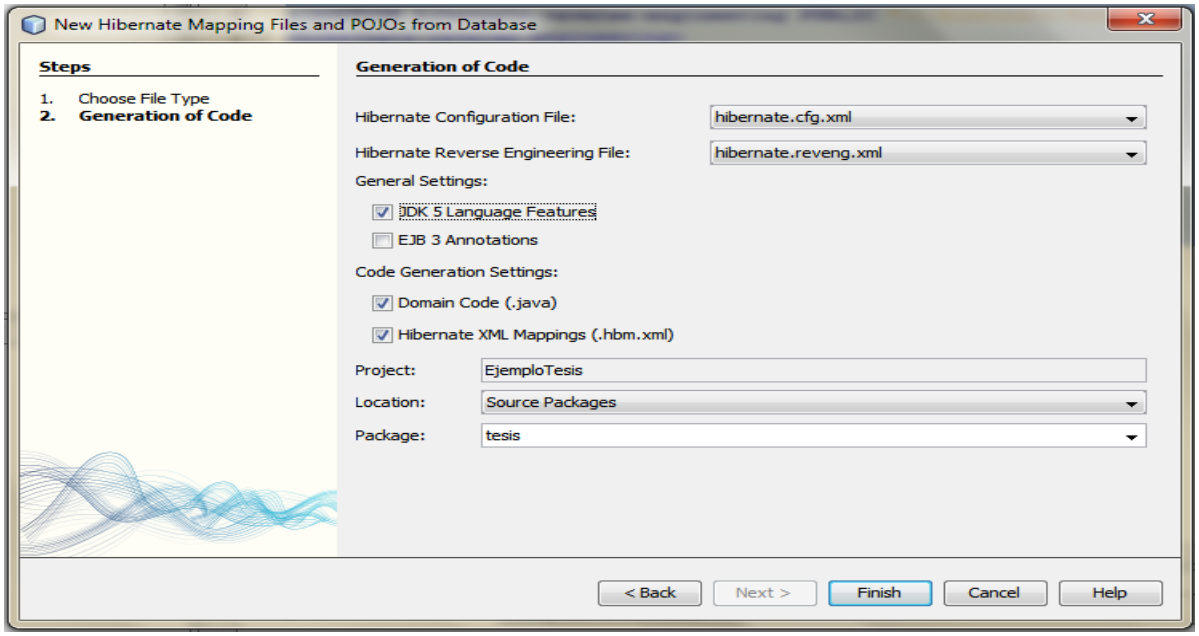

Figura 37. Generación de POJOS.

Con el procedimiento descrito anteriormente se generan los Pojos .java y los archivos de mapeo .hbm.xml de cada una de las tablas de la base de datos.

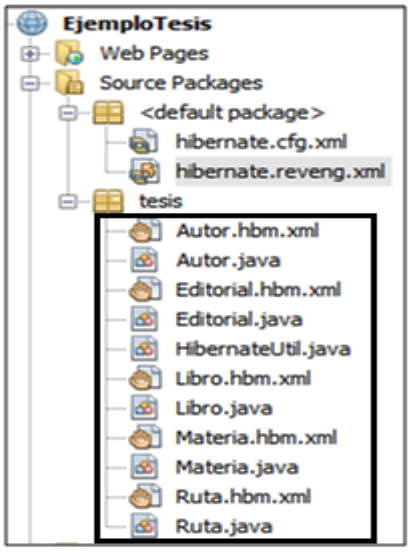

Figura 38. Estructura con los POJOS.

IV.3.1.4.3.1 Ejemplo de POJOS y archivos de Mapeo generados

Hibernate utiliza a los POJOS para establecer persistencia con la base de datos.

Ejemplo del pojo generado para la tabla libro “Libro.java”.

```
package tesis;

import java.util.HashSet;

import java.util.Set;

/**

 * Libro generated by hbm2java

 */

public class Libro  implements java.io.Serializable {


    private String isbn;
```

```
    private Editorial editorial;
    private Ruta ruta;
    private String titulo;
    private int anio;
    private String comentario;
    private Set<Autor> autors = new HashSet<Autor>(0);
    private Set<Materia> materias = new HashSet<Materia>(0);

   public Libro() {
   }

   public Libro(String isbn, Editorial editorial, String titulo, int anio, String comentario) {
       this.isbn = isbn;
       this.editorial = editorial;
       this.titulo = titulo;
       this.anio = anio;
       this.comentario = comentario;
   }
   public Libro(String isbn, Editorial editorial, Ruta ruta, String titulo, int anio, String
comentario, Set<Autor> autors, Set<Materia> materias) {
      this.isbn = isbn;
      this.editorial = editorial;
      this.ruta = ruta;
```

```
    this.titulo = titulo;
    this.anio = anio;
    this.comentario = comentario;
    this.autors = autors;
    this.materias = materias;
  }
    public String getIsbn() {
     return this.isbn;
  }

  public void setIsbn(String isbn) {

     this.isbn = isbn;
  }
  public Editorial getEditorial() {
     return this.editorial;
  }

  public void setEditorial(Editorial editorial) {
     this.editorial = editorial;
  }
  public Ruta getRuta() {
     return this.ruta;
  }
```

```
 public void setRuta(Ruta ruta) {
    this.ruta = ruta;
}
public String getTitulo() {
    return this.titulo;
}

public void setTitulo(String titulo) {
    this.titulo = titulo;
}

public int getAnio() {
    return this.anio;
}
    public void setAnio(int anio) {
    this.anio = anio;
}
public String getComentario() {
    return this.comentario;
}
    public void setComentario(String comentario) {
    this.comentario = comentario;
}
public Set<Autor> getAutors() {
```

```
        return this.autors;


    }
        public void setAutors(Set<Autor> autors) {
        this.autors = autors;
    }
    public Set<Materia> getMaterias() {
        return this.materias;
    }
        public void setMaterias(Set<Materia> materias) {
        this.materias = materias;
    }
}
```

Archivo de Mapeo.

Los archivos de Mapeo tienen el sufijo ".hbm.xml". En cada uno de estos archivos de Mapeo se declaran las propiedades de los POJOS correspondiente, el nombre de las columnas de la base de datos, tipo de dato, referencia y el tipo de relación.

"Libro.hbm.xml".

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<!-- Generated 30/04/2010 03:50:01 PM by Hibernate Tools 3.2.1.GA -->
```

```
<hibernate-mapping>
   <class name="tesis.Libro" table="libro" catalog="libro">


      <id name="isbn" type="string">
         <column name="isbn" length="20" />
         <generator class="assigned" />
      </id>
      <many-to-one name="editorial" class="tesis.Editorial" fetch="select">
         <column name="ideditorial" not-null="true" />
      </many-to-one>
      <many-to-one name="ruta" class="tesis.Ruta" fetch="select">
         <column name="idruta" />
      </many-to-one>
      <property name="titulo" type="string">
         <column name="titulo" length="150" not-null="true" />
      </property>
      <property name="anio" type="int">
         <column name="anio" not-null="true" />
      </property>
      <property name="comentario" type="string">
         <column name="comentario" length="200" not-null="true" />
      </property>
      <set name="autors" inverse="false" table="autor_libro">
         <key>
```

```
            <column name="idlibro" length="20" not-null="true" />
        </key>
        <many-to-many entity-name="tesis.Autor">

            <column name="idautor" not-null="true" />
        </many-to-many>
    </set>
    <set name="materias" inverse="false" table="libro_materia">
        <key>
            <column name="idlibro" length="20" not-null="true" />
        </key>
        <many-to-many entity-name="tesis.Materia">
            <column name="idmateria" not-null="true" />
        </many-to-many>
    </set>
  </class>
</hibernate-mapping>
```

Con el procedimiento descrito anteriormente obtenemos la representación en el mundo de los objetos de la base de datos.

IV.3.1.5 Clase Helper

Una vez obtenida la representación de la base de datos en el mundo de los objetos, se crea una Clase helper, en esta clase obtendremos acceso a la sesión de Hibernate llamando al getSession de la clase HibernateUtil.java y se crean los métodos que sean necesarios para obtener la información de la base de datos. En esta parte se ejecutan las consultas con

el lenguaje HQL que es el lenguaje de consulta de Hibernate, estas consultas no son directamente a la base de datos si no a los objetos previamente definidos por los Pojos y archivos de mapeo.

SessionFactory.getCurrentSession() es llamado en la clase “Libroshelper.java”, una vez que obtenemos una SessionFactory (fácilmente, gracias a la HibernateUtil). El código getCurrentSession() siempre devuelve la unidad "actual" de trabajo. En el hibernate.cfg.xml se configuro thread debido a esto, la unidad actual de trabajo está ligada al thread de Java que se esté ejecutando en ese momento en la aplicación.

Una sesión comienza cuando se le necesita por primera vez, cuando se hace la primera llamada a getCurrentSession(), entonces, es ligada al thread actual por Hibernate. Cuando la transacción termina, Hibernate desliga la sesión del thread, y la cierra. Si se llama getCurrentSession() de nuevo, obtiene una nueva sesión y comienza una nueva unidad de trabajo.

La clase Libroshelper.java solo abre una sesión para ejecutar las diferentes operaciones de la base de datos, esto es lo recomendable para evitar anti-pattern (Un anti patrón de diseño es un patrón de diseño que invariablemente conduce a una mala solución de un problema).

LibrosHelper.java

```
import java.util.ArrayList;

import java.util.Iterator;

import java.util.List;
```

```
import org.hibernate.Query;

import org.hibernate.Session;

/**

*
```

```
 * @author Orlando
 */
public class LibrosHelper {
    Session session = null;
    public LibrosHelper() {
        this.session = HibernateUtil.getSessionFactory().getCurrentSession();
    }
public List getLb() {
            List<Libro> reLibro = null;
  try{
    // En este paso se lee el archivo hibernate.cfg.xml

    org.hibernate.Transaction tx = session.beginTransaction();
     Libro lb=new Libro();

    //Se crea la clausula Select  HQL
     String SQL_QUERY ="Select
lb.isbn,lb.editorial.idEditorial,lb.ruta.idLink,lb.titulo,lb.comentario,lb.anio from Libro lb ";
 Query q= session.createQuery(SQL_QUERY);
reLibro= (List<Libro>) q.list();
  }catch(Exception e){
      e.printStackTrace();
  }
        return reLibro;
    }
```

```
}
```

El código anterior muestra la estructura que tiene la clase “LibrosHelper.java” por cada consulta se debe agregar un método similar a getLb() cambiando las consultas.

Las siguientes son algunas de las consultas que se realizaron a la base de datos.

Tabla de ejemplo de consultas en el lenguaje de consultas de HQL.

```
String SQL_QUERY ="Select au.autorId,au.autorNombre from Autor au where au.autorId between '"+startID+"' and '"+endID+"'";

String SQL_QUERY ="Select mt.idMateria,mt.materia  from Materia mt";

String SQL_QUERY ="Select ed.idEditorial,ed.editorial  from Editorial ed";

String SQL_QUERY ="Select lib.isbn,lib.titulo,t.autorNombre  from Libro lib inner join lib.autors t where t.autorId="+a;


String SQL_QUERY ="Select lib.isbn,lib.titulo  from Libro lib where lib.editorial="+a;

String SQL_QUERY ="Select lib.isbn,lib.titulo,t.materia from Libro lib inner join lib.materias t where t.idMateria="+a;

String SQL_consultatres ="Select au.autorId,au.autorNombre  from Autor au inner join au.libros t where t.isbn="+c;
```

Con la creación de la clase Helper la estructura del proyectó queda como en la figura 39.

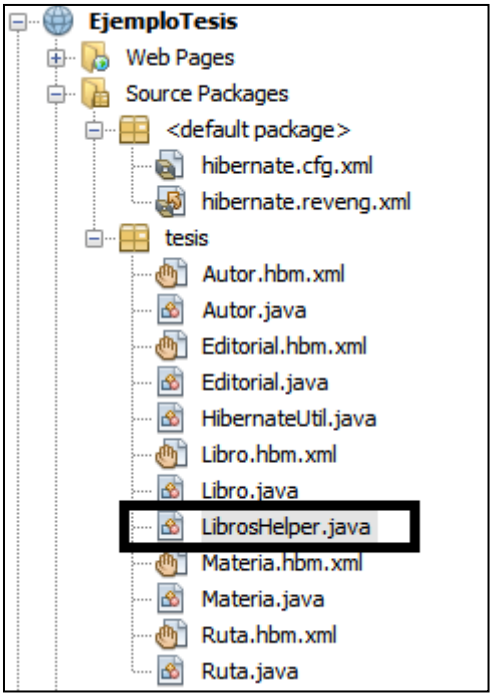

Figura 39. Estructura proyecto clase Helper.

IV.3.1.6 Controlador

El controlador se encarga de recibir las peticiones que el usuario realiza, en un método llama a otro método de la clase Helper con lo cual se realiza la consulta y después prepara (retorna) el valor que recibirá ICEfaces y será mostrado en la página.

También en el controlador se configura a través beans las acciones de los botones. ICEfaces maneja mediante el uso de un bean las acciones que tendrán los botones, por ejemplo, cuando hacemos Click en un botón, éste estará configurado en el controlador con la estructura de un bean, es decir, una variable con set y get.

LibrosController.java

```
package tesis;

import javax.faces.bean.ManagedBean;
```

```
import javax.faces.bean.SessionScoped;
import javax.faces.model.DataModel;
import javax.faces.model.ListDataModel;
import java.util.Arrays;
import java.util.List;
import javax.faces.model.SelectItem;
/**
 *
 * @author Orlando
 */
@ManagedBean(name="LibrosController")
@SessionScoped
public class LibrosController  {
        private DataModel reLibro;
    public DataModel getReLibro() {
   if (reLibro == null) {
       reLibro = new ListDataModel(helper.getLb());
   }
   return reLibro;
}
   /**
    * @param reLibro the reLibro to set
    */
   public void setReLibro(DataModel reLibro) {
       this.reLibro = reLibro;
```

```
    }

}
```

El controlador ejecuta la petición del usuario, el método instancia la consulta y regresa un valor que será mostrado en la vista correspondiente.

Con la creación de la clase LibrosController.java la estructura del proyecto queda como se muestra en la figura 40.

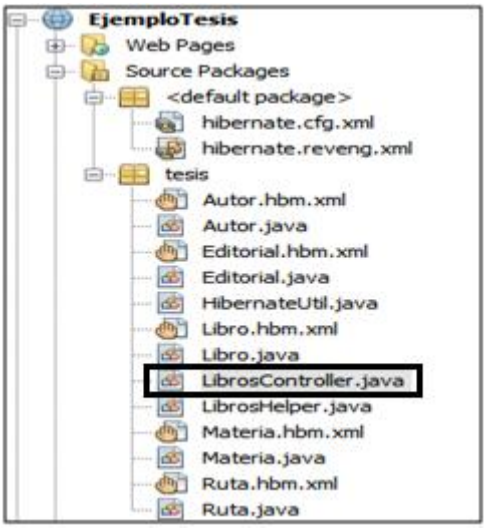

Figura 40. Estructura proyecto controlador.

IV.3.1.7 Diagrama de clases

Mediante el diagrama de clases se pueden ver los atributos y métodos de las clases generadas, así como del resto de las clases que se utilizaran.

Diagrama de clases.

Autor
private int autorId; private String autorNombre;

private Set<Libro> libros = new HashSet<Libro>(0);
public Autor() public Autor(int autorId, String autorNombre) public Autor(int autorId, String autorNombre, Set<Libro> libros) public int getAutorId() public void setAutorId(int autorId) public String getAutorNombre() public void setAutorNombre(String autorNombre) Set<Libro> getLibros() setLibros(Set<Libro> libros)

Editorial
private int idEditorial; private String editorial; private Set<Libro> libros = new HashSet<Libro>(0);
Editorial() public Editorial(int idEditorial, String editorial) public Editorial(int idEditorial, String editorial, Set<Libro> libros) public int getIdEditorial() public void setIdEditorial(int idEditorial) public String getEditorial() public void setEditorial(String editorial) public Set<Libro> getLibros() public void setLibros(Set<Libro> libros)

Clase Libro

private String isbn;
private Editorial editorial;
private Ruta ruta;
private String titulo;
private int anio;
private String comentario;
private Set<Autor> autors = new HashSet<Autor>(0);
private Set<Materia> materias = new HashSet<Materia>(0);

Libro() Libro(String isbn, Editorial editorial, String titulo, int anio, String comentario) public Libro(String isbn, Editorial editorial, Ruta ruta, String titulo, int anio, String comentario, Set<Autor> autors, Set<Materia> materias) getIsbn() setIsbn(String isbn)

getEditorial() setEditorial(Editorial editorial) getRuta() setRuta(Ruta ruta) getTitulo() setTitulo(String titulo) getAnio() setAnio(int anio) getComentario() setComentario(String comentario) getAutors() setAutors(Set<Autor> autors) getMaterias() setMaterias(Set<Materia> materias)

Materia
private int idMateria; private String materia; private Set<Libro> libros = new HashSet<Libro>(0);
Materia() Materia(int idMateria, String materia) Materia(int idMateria, String materia, Set<Libro> libros) getIdMateria() setIdMateria(int idMateria) getMateria() setMateria(String materia) getLibros() setLibros(Set<Libro> libros)

Ruta
private int idLink; private String link; private String tipo; private Set<Libro> libros = new HashSet<Libro>(0);
Ruta() Ruta(int idLink, String link, String tipo) Ruta(int idLink, String link, String tipo, Set<Libro> libros) getIdLink()

setIdLink(int idLink) getLink() setLink(String link) getTipo() setTipo(String tipo) getLibros() setLibros(Set<Libro> libros)

HibernateUtil
Static getSessionFactory()

librosController
LibrosHelper helper; DataModel reMateria; private DataModel reLibro; private DataModel redosLibro; private DataModel reEditorial; private DataModel reName; private DataModel reNombre; private DataModel multiLibro; private DataModel aproLib;
getReLibro() setReLibro(DataModel reLibro) getRedosLibro() setRedosLibro(DataModel redosLibro) getReEditorial() actionReName(String a) getReName() setReName(DataModel reName) actionReNombre(String a) getReNombre() setReNombre(DataModel reNombre) actionMultiLibro(String a,String b) getMultiLibro() setMultiLibro(DataModel multiLibro) getReMateria() actionAproLib(int a)

getAproLib() setAproLib(DataModel aproLib)

LibrosHelper
LibrosHelper() getMateria() getLb() pruebados(a) getEditorial() pruebapruebados(a) pruebaprueba(a) actionMultiLibro(String a,String b) prueba(a)

IV.3.1.8 Integración entre la información obtenida de la base de datos y página Web (Vista)

Para integrar los datos obtenidos de la base de datos con el diseño Web se utiliza ICEfaces. Este toma el valor que obtuvo el controlador tras la petición del usuario y muestra en el navegador los resultados de la consulta con las ventajas de un JSF y AJAX que brinda ICEfaces.

Como mención en la parte del diseño Web el templete Web tiene divisiones las cuales contienen etiqueta insert por ejemplo la del menú <ui:insert name="menu"> esta etiqueta insertara todo lo que se encuentre en la etiqueta <ui:define <ui:define name="menu">; esta etiqueta está definida dentro de una de las vistas.

```
<div id="menu"><ui:insert name="menu">Content</ui:insert></div>

<div class="sub_ventana_uno"><ui:insert name="body">Content</ui:insert></div>

<div class="sub_ventana_dos"><ui:insert name="acc">Content</ui:insert></div>
```

Para dar un ejemplo de la integración entre el contenido obtenido de la base de datos y templete Web, explicaré la integración del templete con los datos obtenidos sobre la tabla Materia.

Para crear las vistas desde Netbeans se crea un templete cliente, éste da una estructura general de lo que será la vista.

El siguiente código contiene los principales elementos del templete.

Para que el templete pueda contener etiquetas ICEfaces se tiene que integrar la URL que indica la localización de las librerías de ICEfaces la cual es el siguiente: xmlns:ice="http://www.icesoft.com/icefaces/component.

Index.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
<!--
   Document   : welcomeICEfaces
   Created on : 11/05/2010, 09:44:24 PM
   Author     : Orlando
-->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:ice="http://www.icesoft.com/icefaces/component"
```

La siguiente etiqueta <ui:composition indica al templete que va a ser utilizado dentro de ésta, cada una de las etiquetas <ui:define será insertada dentro del templete en su correspondiente etiqueta <ui:insert.

```
<ui:composition template="./templete.xhtml">
```

La siguiente etiqueta <ui:define name="menu"> inserta dentro del templete sustituyendo en el templete la etiqueta <ui:insert name="menu">.

```
<ui:define name="menu">
        <ul>
          <li>
            <f:view>
              <ice:form id="mateuno" >
                <ice:commandLink
                  id="submit"
                  action="mate2"
                 >
                  <b> <ice:outputText value="Inicio"/></b>
                </ice:commandLink>
              </ice:form>
            </f:view>
          </li>
          <li>
            <f:view>
              <ice:form id="matedos" >
```

```
                <ice:commandLink
                   id="submit"
                   action="mate4"
                   >
                   <b><ice:outputText value="Autores"/></b>
                </ice:commandLink>
             </ice:form>
          </f:view>
       </li>
       <li>
          <f:view>
             <ice:form id="matetres" >
                <ice:commandLink
                   id="submit"
                   action="mate3"
    >
                   <b><ice:outputText value="Libros"/></b>
                </ice:commandLink>
             </ice:form>
          </f:view>
       </li>
       <li class="active">
          <f:view>
             <ice:form id="matecuatro" >
                <ice:commandLink
```

```
                    id="submit"
                    action="mate1"
                    >
                    <b><ice:outputText value="Materias"/></b>
                  </ice:commandLink>
                </ice:form>
              </f:view>
            </li>
            <li>
              <f:view>
  <ice:form id="matecinco" >
                  <ice:commandLink
                    id="submit"
                    action="case1"
                    >
                    <b><ice:outputText value="Editorial"/></b>
                  </ice:commandLink>
                </ice:form>
              </f:view>
            </li>
          </ul>
      </ui:define>
```

La etiqueta <ui:define name=”body”> corresponde a la div sub_ventana_uno en el templete dicha etiqueta se insertara <ui:insert name=”body”>.

La etiqueta “ice:dataTable” toma el valor obtenido del controlador “librosController.reMateria” formando la tabla en base a su estructura ya definida en las librerías ICEfaces en esa misma tabla se agrega un link llamado “ver”.

```
<ice:commandLink
action="#{librosController.actionRetresLibro(item[0])}"  value="ver"   />
</h:column>
```

Este link despliega los libros de la material correspondiente.

```
<ui:define name="body" >
  <ice:form id="iceForm">
    <ice:outputText value="Tabla Autor"/>
    <ice:dataTable
      rows="5"
      id="inventoryList"
      value="#{librosController.reMateria}"
      var="item"
      rendered="#{librosController.reMateria.rowCount > 0}">
      <!-- Stock number -->
      <ice:column>
        <f:facet name="header">
          <ice:outputText value="Materia"/>
        </f:facet>
        <ice:outputText value="#{item[1]}"/>
      </ice:column>
      <!-- Model number -->
```

```
                <h:column>
                    <f:facet name="header">
                   <h:outputText value="Libros"/>
                    </f:facet>
                    <ice:commandLink
action="#{librosController.actionRetresLibro(item[0])}"  value="ver"   />
                </h:column>
            </ice:dataTable>
            <!-- Paginator with page controls -->
            <ice:data tor id="dataScroll_3"
                        for="inventoryList"
                        paginator="true"
                        fastStep="3"
                        paginatorMaxPages="4">
                <f:facet name="first">
                    <ice:graphicImage
                        url="./xmlhttp/css/rime/css-images/arrow-first.gif"
                        style="border:none;"
                        title="First Page"/>
                </f:facet>
                <f:facet name="last">
                    <ice:graphicImage
                        url="./xmlhttp/css/rime/css-images/arrow-last.gif"
                        style="border:none;"
                        title="Last Page"/>
```

```
            </f:facet>
            <f:facet name="previous">
               <ice:graphicImage
                  url="./xmlhttp/css/rime/css-images/arrow-previous.gif"
                  style="border:none;"
                  title="Previous Page"/>
            </f:facet>
          <f:facet name="next">
               <ice:graphicImage
                  url="./xmlhttp/css/rime/css-images/arrow-next.gif"
                  style="border:none;"
                  title="Next Page"/>
            </f:facet>
            <f:facet name="fastforward">
               <ice:graphicImage url="./xmlhttp/css/rime/css-images/arrow-ff.gif"
                           style="border:none;"
                           title="Fast Forward"/>
            </f:facet>
            <f:facet name="fastrewind">
               <ice:graphicImage url="./xmlhttp/css/rime/css-images/arrow-fr.gif"
                           style="border:none;"
                           title="Fast Backwards"/>
            </f:facet>
         </ice:dataPaginator>
         <!-- Display counts about the table and the currently displayed page -->
```

```
            <ice:dataPaginator id="dataScroll_2" for="inventoryList"
                               rowsCountVar="rowsCount"
                               displayedRowsCountVar="displayedRowsCount"
                               firstRowIndexVar="firstRowIndex"
                               lastRowIndexVar="lastRowIndex"
                               pageCountVar="pageCount"
                               pageIndexVar="pageIndex">
                <ice:outputFormat
                    value="Se encontraron {0} Materias, Mostrar {1} Materias, del {2} al {3}.
Páginas {4} / {5}."
                    styleClass="standard">
                    <f:param value="#{rowsCount}"/>
                    <f:param value="#{displayedRowsCount}"/>
                    <f:param value="#{firstRowIndex}"/>
                    <f:param value="#{lastRowIndex}"/>

                    <f:param value="#{pageIndex}"/>
                    <f:param value="#{pageCount}"/>
                </ice:outputFormat>
            </ice:dataPaginator>
        </ice:form>
</ui:define>
```

La etiqueta <ui:define name="acc" > insertará los libros de la material a la cual se dio click en el botón “ver”, se actualizará en “acc” la cual en el templete es la sub_ventana_dos.

```
 <ui:define name="acc"  >

<ice:outputText value="La Materia tiene los siguientes libros:"/>

        <ice:panelSeries style="border:1px solid silver;" var="items"
value="#{librosController.retresLibro}" >

            <div class="box-blue">

               <br/>

                <ice:outputText value="Nombre de la Materia:"/>

            <br/>

              <ice:outputText value="#{items[2]}"/>

            <br/>

            <ice:outputText value="Nombre del Libro:"/>

            <br/>

            <ice:outputText value="#{items[1]}"/>

            <br/>

            <ice:outputText value="ISBN:"/>

            <br/>

            <ice:outputText value="#{items[0]}"/>

            <br/>

            </div>

         </ice:panelSeries>

      </ui:define>
```

Con la integración de los datos y la pagina web la estructura del proyecto queda como se muestra el figura 41.

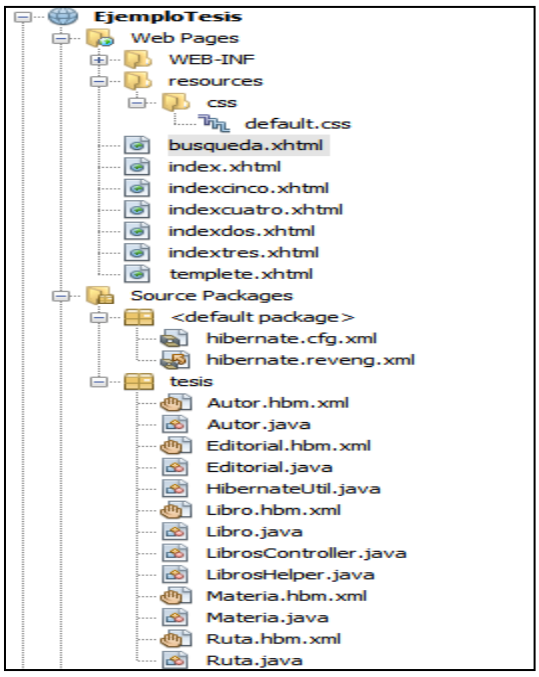

Figura 41. Estructura Final del proyecto.

Para ejecutar el proyecto damos click derecho y presionamos “run” con esto el sitio web estará disponible.

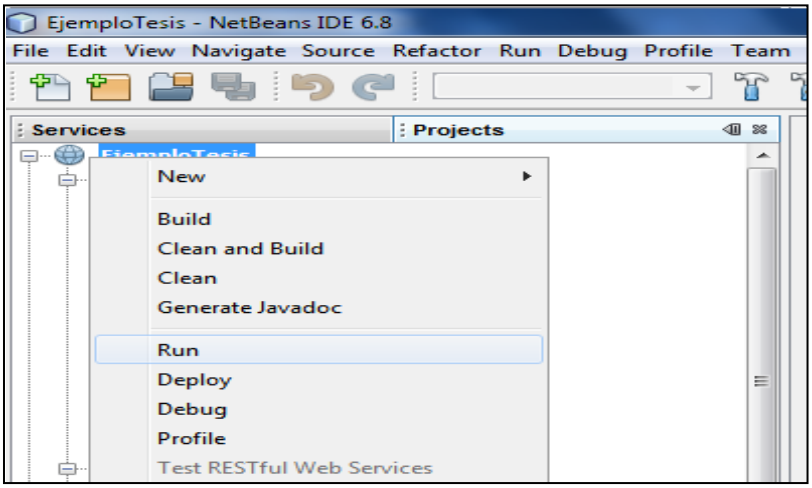

Figura 42. Ejecución del proyecto.

Hasta el momento solo se ha estado del lado del desarrollador a continuación se explicará del lado del usuario.

IV.3.1.9 Manual de Usuario de la aplicación Web

La integración entre el templete Web y los datos obtenidos de la base de datos se muestran a continuación en las siguientes figuras, que son explicadas como un “Manual de Usuario de la aplicación Web”.

Para entrar en la aplicación es necesario teclear en el Navegador la dirección URL siguiente:

http://localhost:37895/ice/index.jsf

Al ingresar la URL se desplegará la pantalla de “Inicio”.

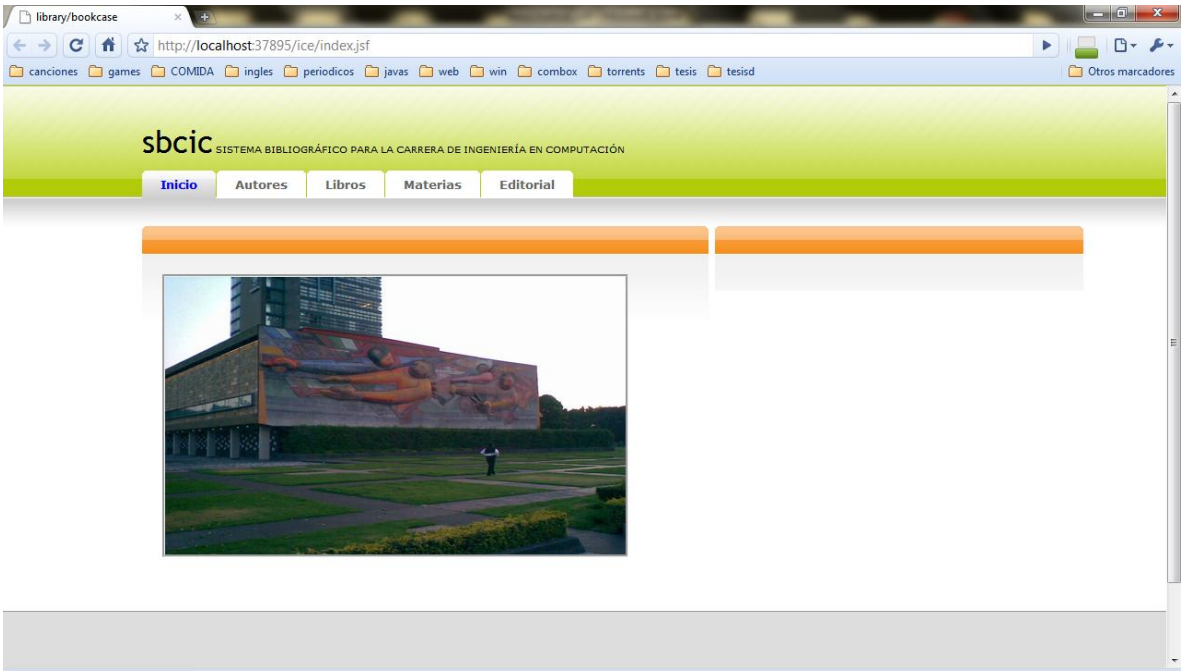

Figura 43. Vista de inicio.

Para consultar una lista de autores dar Click en la pestaña “Autores”.

Figura 44. Menú en la vista de inicio.

Al dar Click se ha seleccionado Autores.

Figura 45.Menú en la vista Autores.

La vista Autores muestra en general a los Autores que están en la bibliografía de la carrera de Ingeniería en Computación.

Figura 46. Vista de Autores.

Para consultar los libros publicados por algún Autor dar Click en “Ver”.

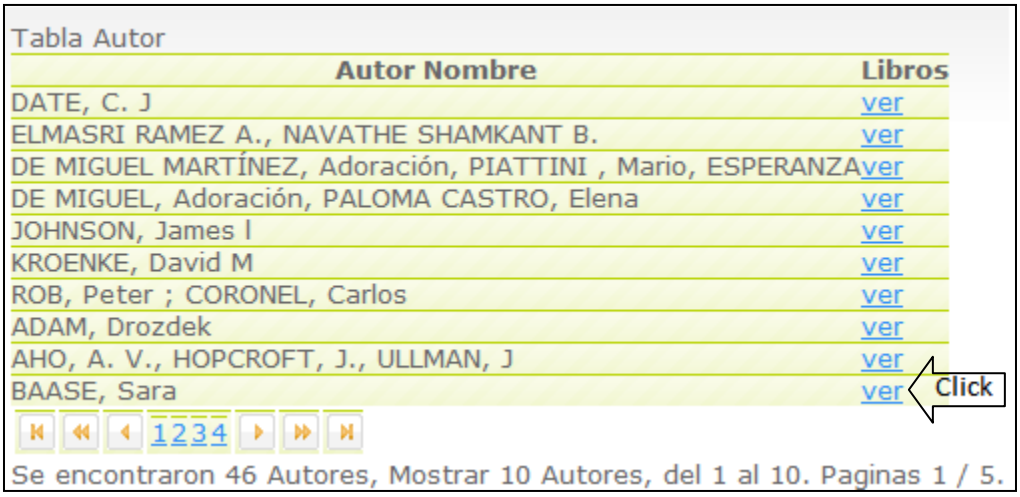

Figura 47. Tabla Autor en la vista autor.

El resultado se desplegará en la parte derecha de la pantalla, por ejemplo si damos Click en la autora "BAASE, Sara" el resultado sería el siguiente:

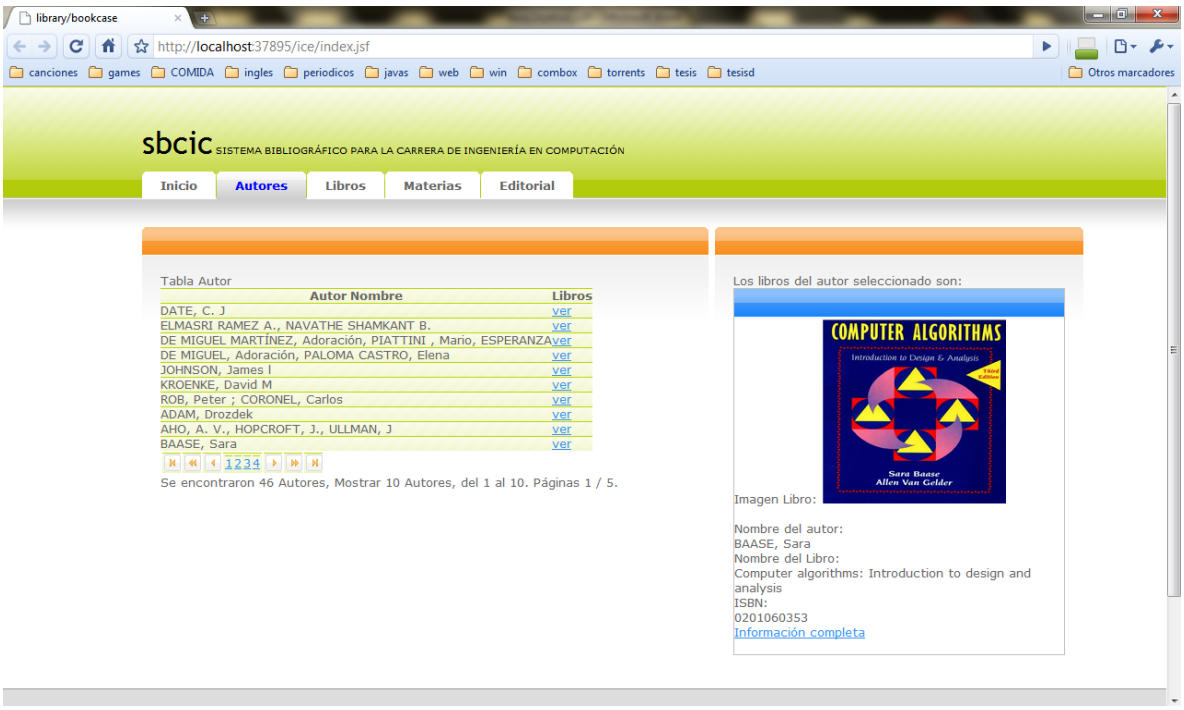

Figura 48. Vista autores después de dar click en alguno de los autores se despliega sus libros

En la parte inferior de la tabla de autores se muestra:

"Se encontraron 46 Autores, Mostrar 10 Autores, del 1 al 10.Páginas 1/5".

En la parte inferior de la tabla aparece un menú de navegación.

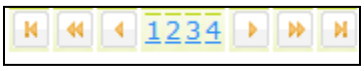

Figura 49. Menú de navegación de la tabla Autor.

Este menú nos muestra las siguientes páginas, en la cual podemos consultar más autores de la lista.

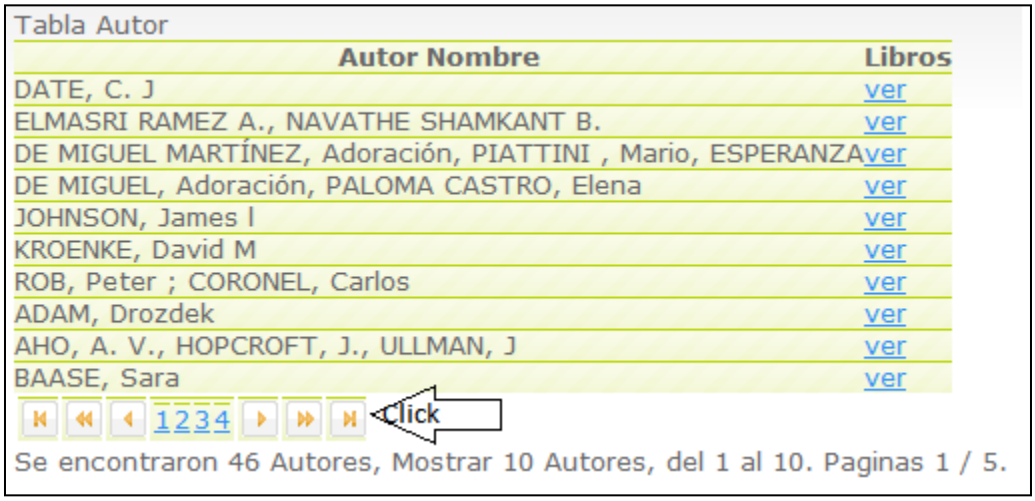

Figura 50. Tabla Autores da click al menú de navegación de la tabla.

Al dar Click en el "4" obtenemos el siguiente resultado:

"Se encontraron 46 Autores, Mostrar 10 Autores, del 31 al 40."

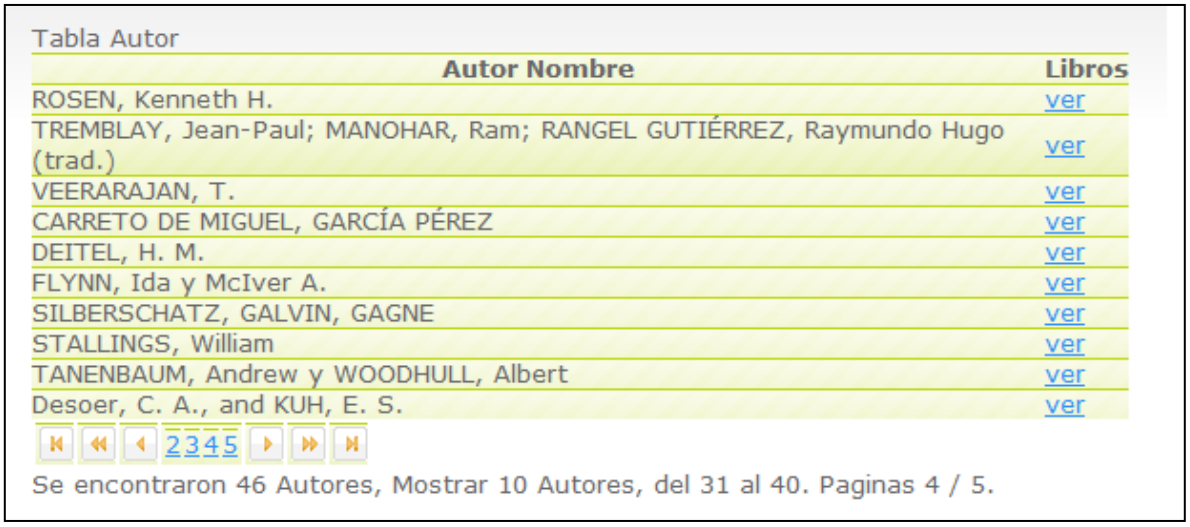
Tabla Autor

Autor Nombre	Libros
ROSEN, Kenneth H.	ver
TREMBLAY, Jean-Paul; MANOHAR, Ram; RANGEL GUTIÉRREZ, Raymundo Hugo (trad.)	ver
VEERARAJAN, T.	ver
CARRETO DE MIGUEL, GARCÍA PÉREZ	ver
DEITEL, H. M.	ver
FLYNN, Ida y McIver A.	ver
SILBERSCHATZ, GALVIN, GAGNE	ver
STALLINGS, William	ver
TANENBAUM, Andrew y WOODHULL, Albert	ver
Desoer, C. A., and KUH, E. S.	ver

2345

Se encontraron 46 Autores, Mostrar 10 Autores, del 31 al 40. Paginas 4 / 5.

Figura 51. Tabla actualizada después de dar Click en 4.

Desde inicio o en cualquiera de las otras vistas podremos acceder a la vista “Materia”, ésta muestra de inicio las Materias de la carrera de Ingeniería en Computación.

Al dar Click en “Materias”.

Figura 52. Menú en la tabla materia.

Obtenemos:

Figura 53.Vista materia.

Figura 54. Tabla en vista materia muestra las materias de la carrera de Ingeniería en Computación.

Al dar Click en "Ver" en alguna de las Materias, por ejemplo en "Ingeniería de Software", se despliegan todos los libros de la Materia seleccionada.

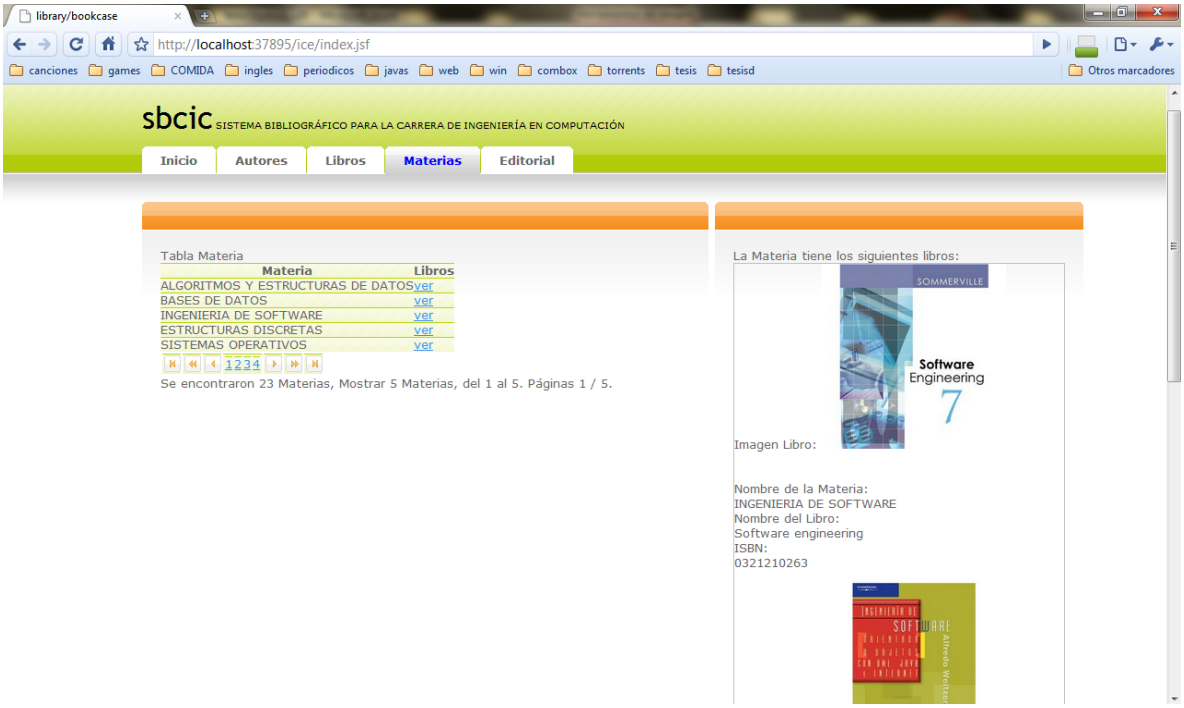

Figura 55. Vista materia después de dar click a "ver" en la columna bases de datos, se despliegan las materias de bases de datos.

Al dar Click en "Ver" en otra de las Materias, se mostrarán los libros correspondientes a dicha materia, únicamente se actualizará el lado derecho de la página.

Si queremos ver los libros que tiene cierta editorial en particular, Damos Click en "Editorial".

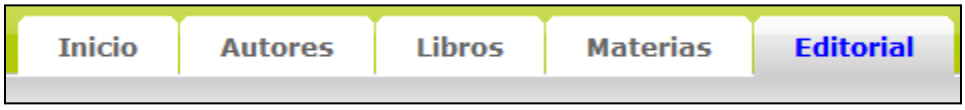

Figura 56. Menú en la vista editorial.

Se despliega el nombre de las editoriales que tengan libros en el plan de estudios de la carrera de Ingeniería en Computación.

Figura 57.Vista editorial.

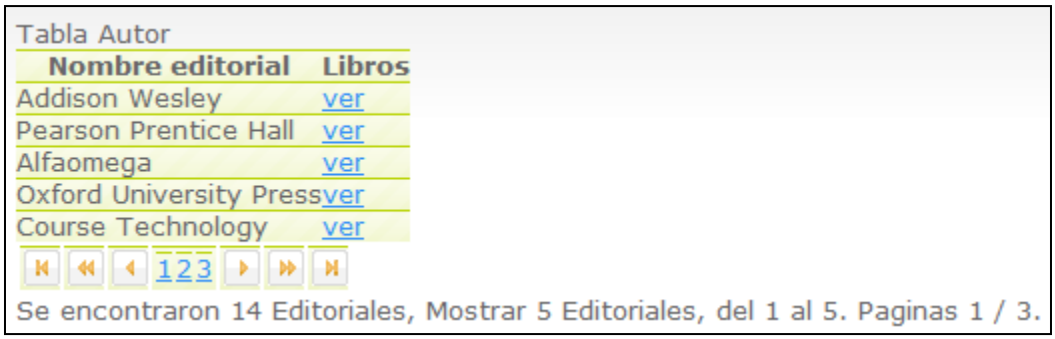

Figura 58. Tabla que muestra las editoriales.

Al dar Click en “Ver” en alguna de las editoriales se despliegan los libros correspondientes a dicha editorial.

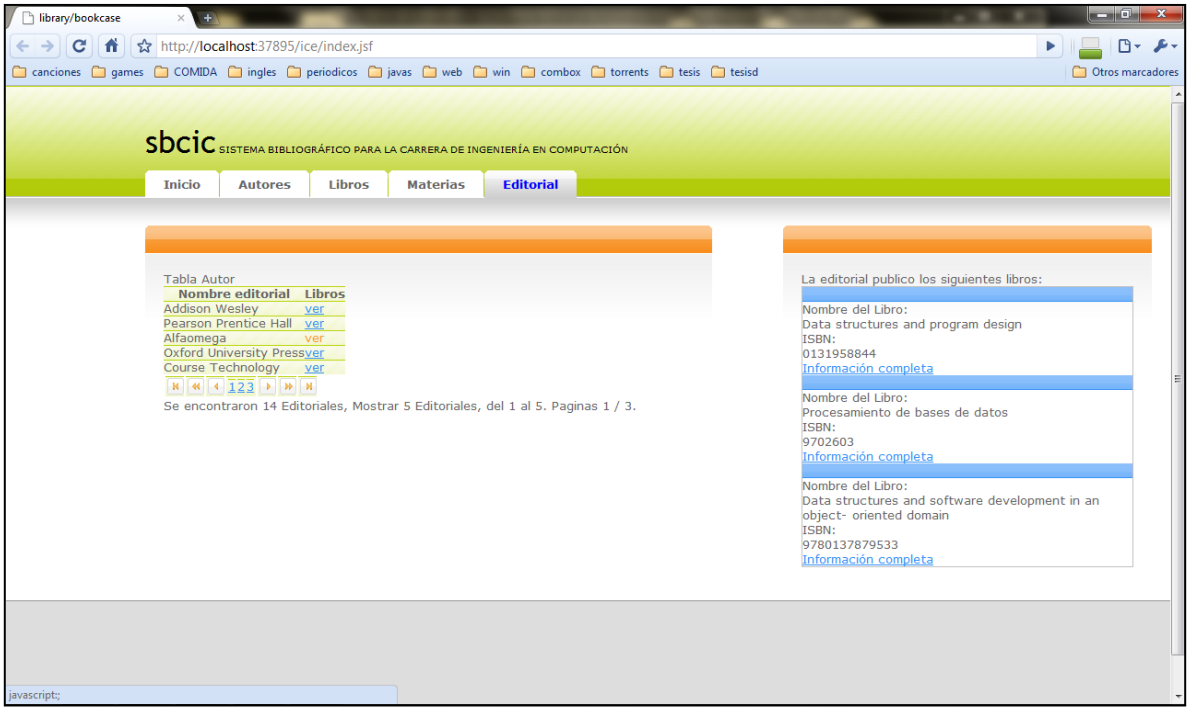

Figura 59. Vista editorial que despliega los libros de la editorial seleccionada.

V Conclusiones

Con el desarrollo descrito a lo largo de la tesis, se logró cumplir con los objetivos propuestos en el inicio de la tesis.

1) Se desarrolló una base de datos relacional en MySQL, con ella se logró organizar la información, proporcionando la estructura para que uno o varios usuarios puedan recuperar la información desde la aplicación web.

2) Se elaboró un templete Web, para presentar los datos. El templete Web tiene una estructura la cual hace que los usuarios puedan consultar los datos de manera sencilla y amigable; del lado del desarrollador hace fácil el mantenimiento.

3) Se desarrolló un sistema Web, con una interfaz amigable para el usuario, publicado en un sitio web.

Para cumplir con todos los objetivos, el sistema Web se estructuró usando el modelo vista controlador (MVC). La separación en módulos proporcionada por este modelo hace que el desarrollo sea más eficiente.

Para elaborar el sistema Web, fue necesario hacer uso de diversas herramientas de software tal es el caso de Java para el manejo de la parte de control del sistema, el manejo de la base de datos fue realizado con MySQL , para abrir el canal de comunicación entre la base de datos y la aplicación se utilizó Hibernate, la vista del usuario se implementó mediante el uso XHTML y CSS, para presentar los datos en la vista se utilizó ICEfaces. Cada una de estas herramientas realiza su función específica haciendo posible el desarrollo del sistema.

Anexo A

Libros digitales.

"Un libro digital (ebook) es una edición digitalizada la cual fue elaborada para ser vendida por Internet, con una estructura y diseño adaptados para su visualización en computadoras y nuevos dispositivos diseñados especialmente para este fin."

Los libros digitales (ebook) tienen hoy un crecimiento a destacar ya sea desde una computadora de escritorio, una laptop, o el seguimiento de dispositivos portátiles por ejemplo iPad de Apple y muchos que están adaptando para hacer una lectura digital hace que los libros digitales se puedan presentar en estos días como una realidad que se extenderá en el futuro podremos ver como los libros digitales toman fuerza en base a sus ventajas.

Ventajas del libro digital tenemos.

- Difusión más rápida de los temas podríamos estar al día con las nuevas Publicaciones.
- Podríamos tener versiones donde se corrijan las erratas.
- Se podría buscar de manera rápida tanto el libro como la información que contiene y con la ventaja que desde cualquier PC con Internet.
- Portabilidad podríamos revisar muchos libros.
- Podría ser leído mucha gente forma simultanea.
- Podríamos poner información links sobre el tema y contenido multimedia.

Anexo B

La propiedad Dialect le dice a Hibernate la base de datos que estamos utilizando en específico. Hibernete puede trabajar con las siguientes bases de datos.

Lista de Hibernate SQL Dialects

RDBMS	Dialect
DB2	org.hibernate.dialect.DB2Dialect
DB2 AS/400	org.hibernate.dialect.DB2400Dialect
DB2 OS390	org.hibernate.dialect.DB2390Dialect
PostgreSQL	org.hibernate.dialect.PostgreSQLDialect
MySQL	org.hibernate.dialect.MySQLDialect
MySQL with InnoDB	org.hibernate.dialect.MySQLInnoDBDialect
MySQL with MyISAM	org.hibernate.dialect.MySQLMyISAMDialect
Oracle (any version)	org.hibernate.dialect.OracleDialect
Oracle 9i/10g	org.hibernate.dialect.Oracle9Dialect
Sybase	org.hibernate.dialect.SybaseDialect
Sybase Anywhere	org.hibernate.dialect.SybaseAnywhereDialect
Microsoft SQL Server	org.hibernate.dialect.SQLServerDialect
SAP DB	org.hibernate.dialect.SAPDBDialect
Informix	org.hibernate.dialect.InformixDialect

RDBMS	Dialect
HypersonicSQL	org.hibernate.dialect.HSQLDialect
Ingres	org.hibernate.dialect.IngresDialect
Progress	org.hibernate.dialect.ProgressDialect
Mckoi SQL	org.hibernate.dialect.MckoiDialect
Interbase	org.hibernate.dialect.InterbaseDialect
Pointbase	org.hibernate.dialect.PointbaseDialect
FrontBase	org.hibernate.dialect.FrontbaseDialect
Firebird	org.hibernate.dialect.FirebirdDialect

Anexo C

Datos utilizados.

Los datos utilizados en la base de datos fueron tomados de los planes de estudio de la carrera de Ingeniería en Computación.

Los siguientes datos muestran una parte de la información que fue capturada en la base de datos.

ALGORITMOS Y ESTRUCTURAS DE

Bibliografía básica: Temas para los que se recomienda.

ALGORITMOS Y ESTRUCTURAS DE DATOS .

ADAM, Drozdek
Data structures and algorithms in C++
3rd. edition
U.S.A.
Thomson, 2005

AHO, A. V., HOPCROFT, J., ULLMAN, J
Estructuras de datos y algoritmos
México
Addison-Wesley Iberoamericana, 1998

BAASE, Sara
Computer algorithms: Introduction to design and analysis
3rd. edition
Massachusetts
Addison-Wesley, 2000

BERGIN, Joseph,
Data abstraction: The object oriented approach using C++
New York
McGraw-Hill,1994

EUAN, J., CORDERO, L.
Estructuras de datos
México
Limusa, 1989

HERNÁNDEZ, Roberto; et. al.
Estructuras de datos y algoritmos
Madrid
Pearson Educación, 2000

JOYANES, AGUILAR, Luis y ZAHONERO MARTÍNEZ, Ignacio
Algoritmos y estructuras de datos. Una perspectiva en C
España
McGraw-Hill, 2004

KENNETH A. BERMAN, JEROME L. Paul
Algorithms: Sequential, Parallel, and Distributed
U.S.A.
Thomson, 2005

KRUSE, Robert Leroy
Data structures and program design
3rd edition
New Jersey
Prentice Hall, 1994

TREMBLAY, J., SORENSON,
An introduction to data structures with applications
2nd edition
U.S.A.
McGraw-Hill, 1984

TREMBLAY, J. y CHESTON, G.A.
Data structures and software development in an object-oriented domain
New Jersey
Pearson Education, 2003

Bibliografía complementaria:

BRASSARD, G, BRATLEY,
Fundamentos de algoritmia
Madrid
Prentice-Hall, 1997

DE GIUSTI, Armando E.
Algoritmos, datos y programas
México
Pearson Educación, 2001

KINGSTON, J.
Algorithms and Data Structures: Design Correctness and Analysis
2nd edition

GB
Addison-Wesley, 2001

KNUTH, Donald E.
The art of computer programming
Vol. I. Fundamental algorithms
3rd. Ed
[s.l.i.] USA
Addison Wesley, 1998

KNUTH, Donald E.
The art of computer programming
Vol. 3. Sorting and searching
3rd. Ed
[s.l.i.] USA
Addison Wesley, 1998

KOZEN, Dexter
The design and analysis of algorithms
New York
Springer, 1992

RICHARDSON, David R.
The Book on Data Structures: Volume I
[s.l.i.] USA
Iuniverse Inc, 2002

BASES DE DATOS

Bibliografía básica: Temas para los que se recomienda:

DATE, C. J,
An Introduction to Database Systems
8a. Edición
Reading, Massachussets, U.S.A
Addison Wesley, 2003

ELMASRI RAMEZ A., NAVATHE SHAMKANT B.,
Fundamentos de Sistemas de Bases de datos,
Pearson Prentice Hall,
ISBN: 8478290516, 2003

DE MIGUEL MARTÍNEZ, Adoración, PIATTINI , Mario, ESPERANZA, Marcos
Diseño de bases de datos relacionales
México
Alfaomega, 2000

DE MIGUEL, Adoración, PALOMA CASTRO, Elena
Diseño de bases de datos (Problemas Resueltos)
México
Alfaomega, 2001

JOHNSON, James l,
Bases de datos, modelos, lenguajes, diseño
México
Oxford, 2000

KROENKE, David M.,
Procesamiento de bases de datos
8a. Edición
México
Pearson / Prentice Hall, 2003

ARELLANO M., Lucila P. y Hernandez Hdez. Luciralia
Manual de prácticas de la asignatura de Bases de Datos
UNAM, Fac. De Ingeniería., DIE

Bibliografía complementaria:

ROB, Peter ; CORONEL, Carlos
Database systems (Design, Implementation and Management)
6th. Edition
[s.l.i.] U.S.A.
Course Technology, 2004

LONEY, Kevin
Oracle Database 106: The Complete Reference
[s.l.i] U.S.A.
Mc Graw Hill – Osborne Media, 2004

Anexo D

phpMyAdmin.

Xampp es un paquete integrado por diferente software de desarrollo como Apache, MySQL, phpMyAdmin, PHP, Perl, entre otros.

La herramienta phpMyAdmin es parte Xampp y desde ella se administra la base de datos.

Para iniciar phpMyAdmin vamos a la carpeta donde se instalo Xampp damos click en "xampp_start".

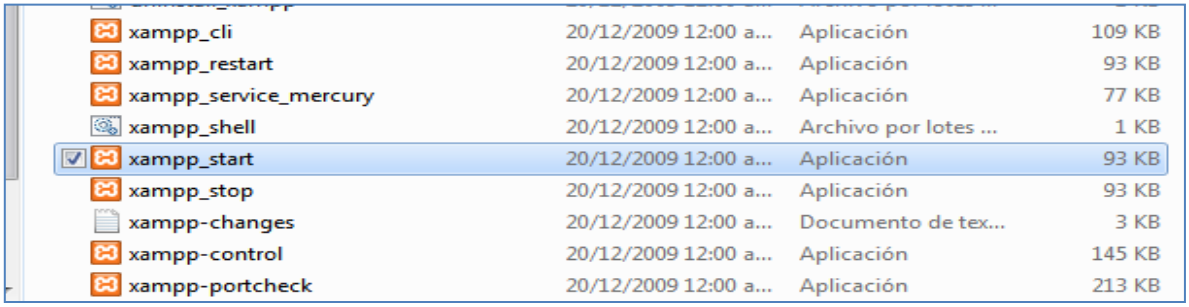

Figura 60. Inicio de xampp.

Con lo cual se inicializa el Xampp.

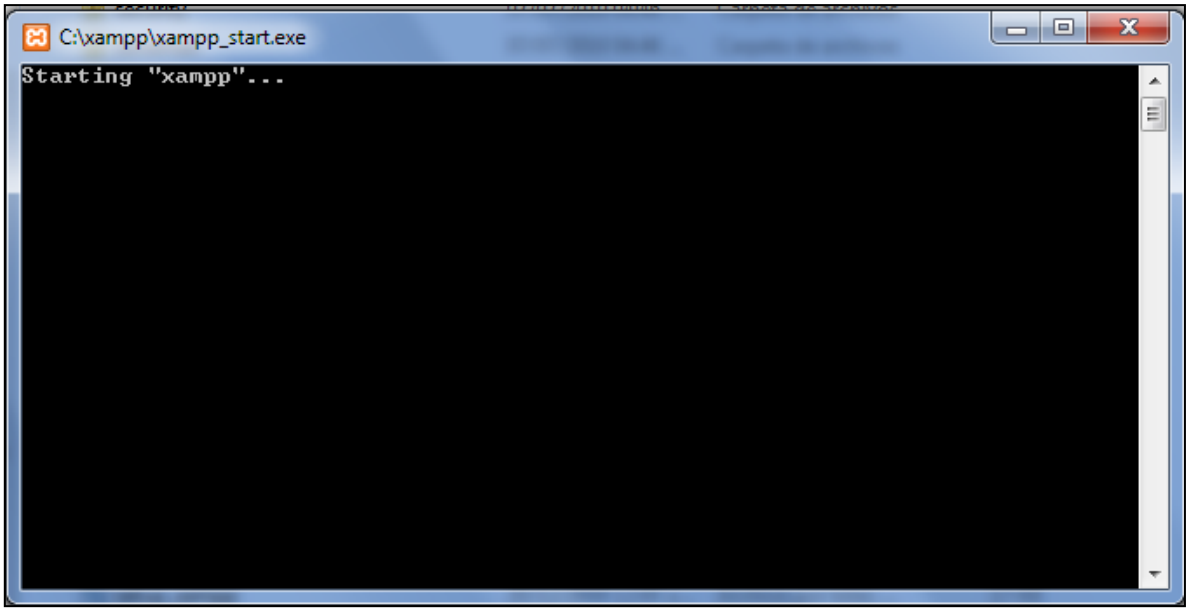

Figura 61. Ejecución de xampp.

Ya inicializado Xampp desde el navegador se teclea la siguiente URL

http://localhost/xampp/

aparecerá la interfaz.

Damos click en phpMyAdmin.

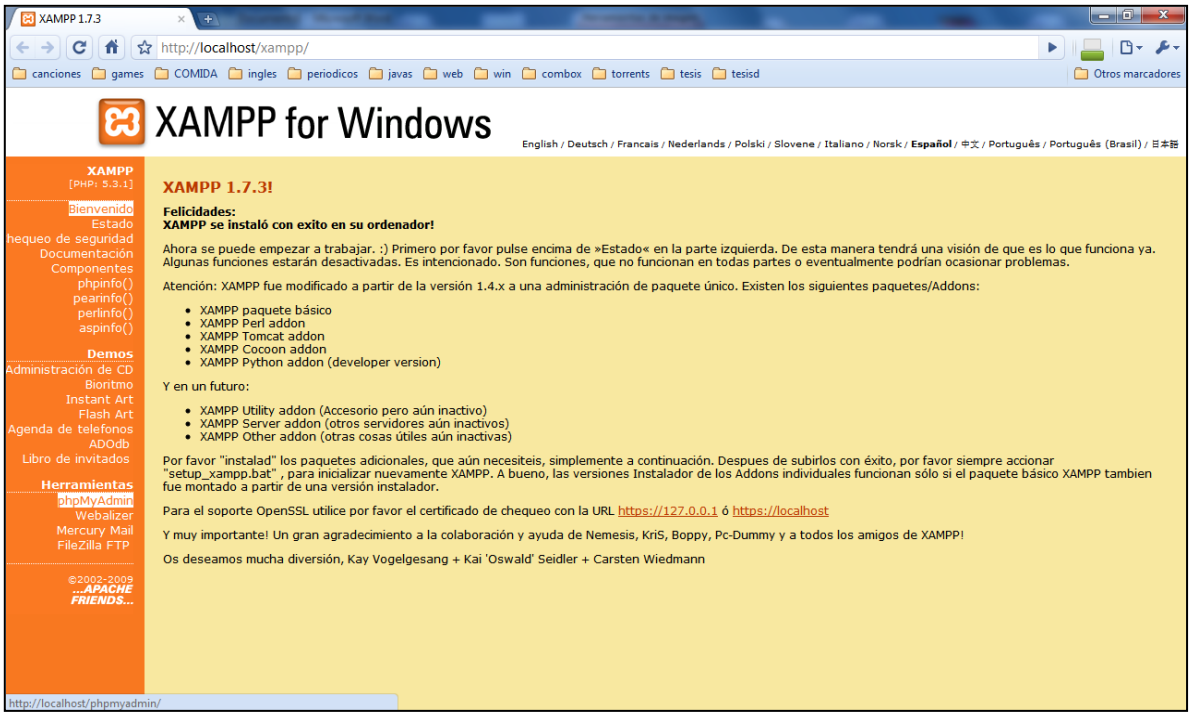

Figura 62. Interfaz xampp.

phpMyAdmin pide el usuario y la contraseña para poder acceder.

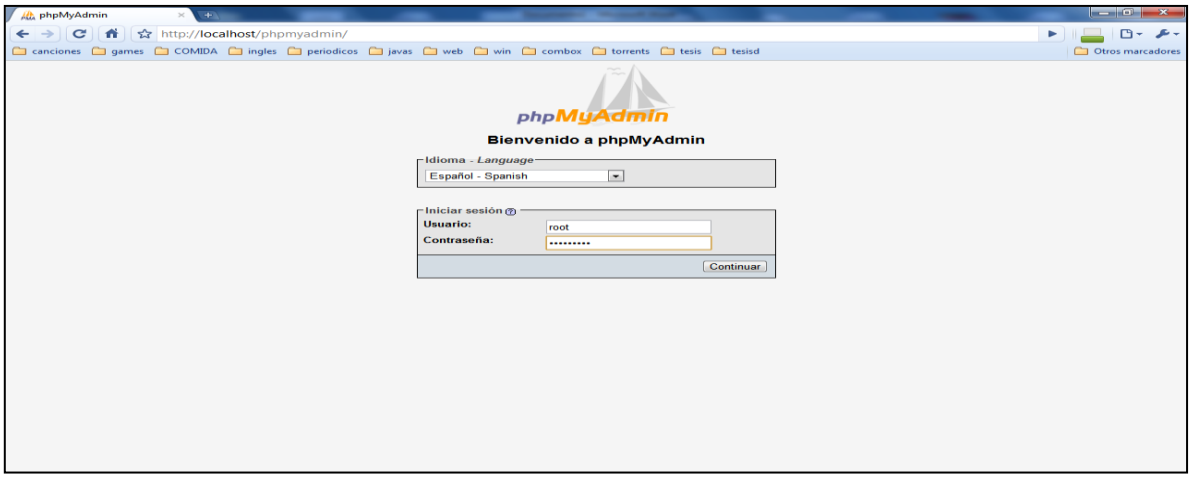

Figura 63. Entrada a phpMyAdmin.

Con esto se accede a la interfaz phpMyAdmin donde se puede crear, modificar o agregar nuevos datos a la base de datos.

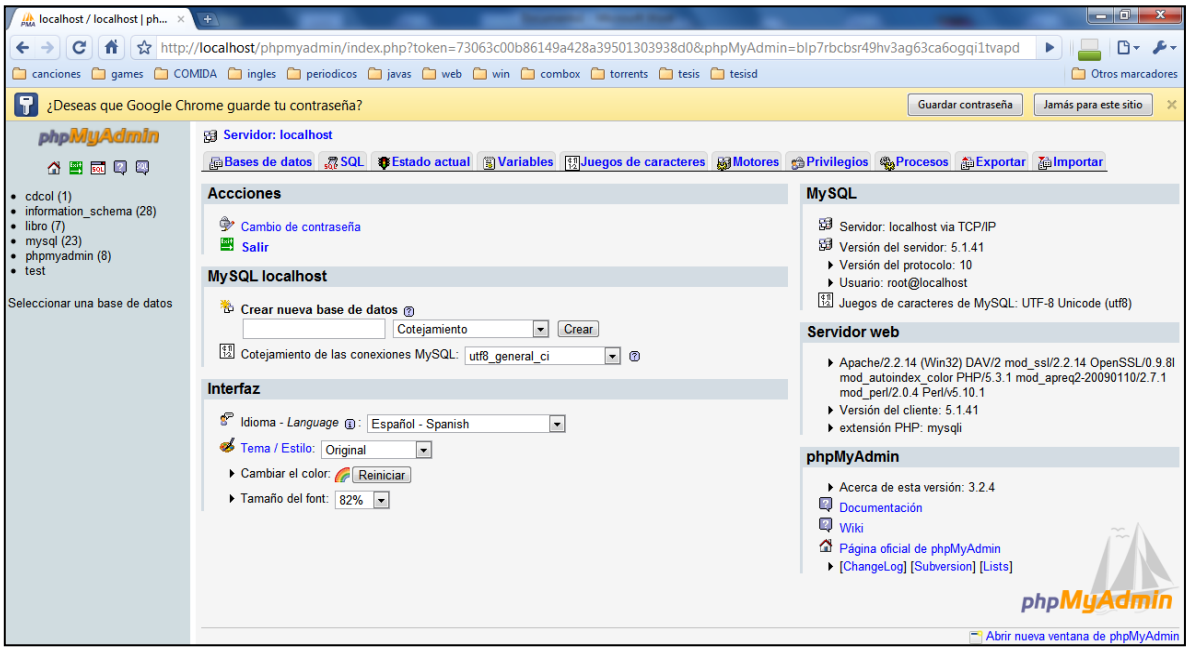

Figura 64. Interfaz phpMyAdmin.

Desde phpMyAdmin se pueden administrar a los usuarios del sistema se pueden crear usuarios desarrolladores con todos los privilegios, por ejemplo, usuarios para capturar la información solo con privilegios de insertar valores.

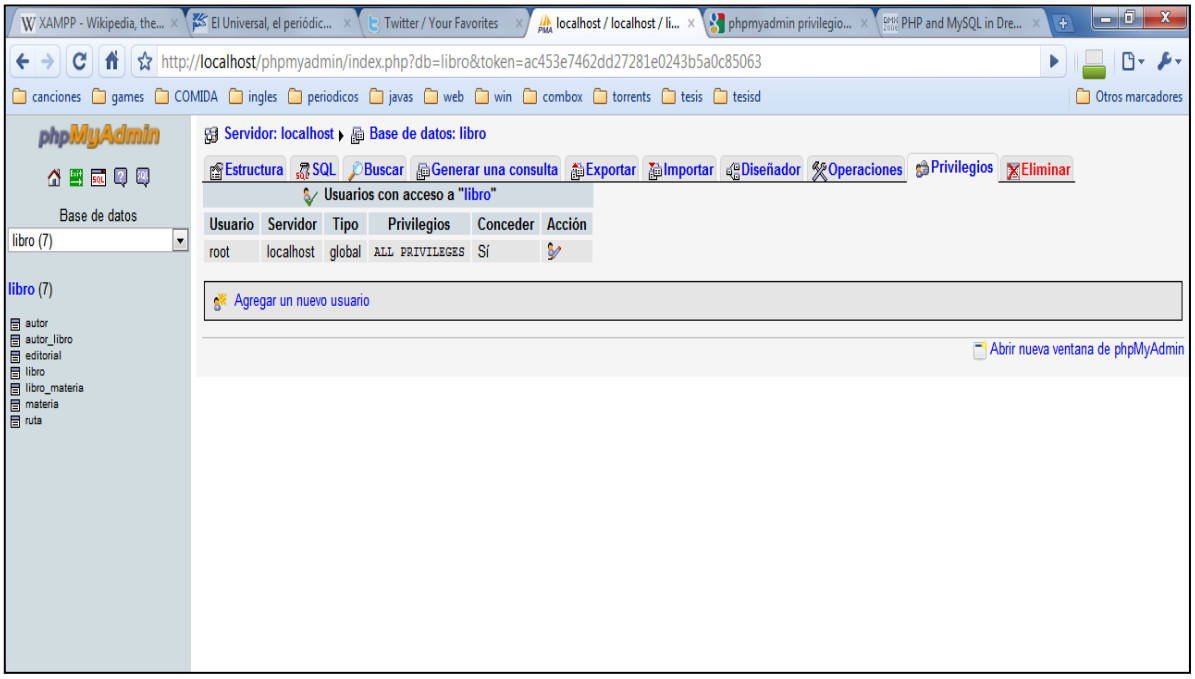

Figura 65. Creación de usuarios.

Glosario

AJAX[30]. acrónimo de ***A****synchronous* ***J****avaScript* ***A****nd* ***X****ML* (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

Ajax es una tecnología asíncrona, en el sentido de que los datos adicionales se requieren al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. JavaScript es el lenguaje interpretado (scripting language) en el que normalmente se efectúan las funciones de llamada de Ajax mientras que el acceso a los datos se realiza mediante *XMLHttpRequest*, objeto disponible en los navegadores actuales. En cualquier caso, no es necesario que el contenido asíncrono esté formateado en XML.

Ajax es una técnica válida para múltiples plataformas y utilizable en muchos sistemas operativos y navegadores dado que está basado en estándares abiertos como JavaScript y Document Object Model (DOM).

JavaBean. Los **JavaBeans** son un modelo de componentes creado por Sun Microsystems para la construcción de aplicaciones en Java. Se usan para encapsular varios objetos en un único objeto (la vaina o Bean en inglés), para hacer uso de un sólo objeto en lugar de varios más simples.

La especificación de JavaBeans de Sun Microsystems los define como "componentes de software reutilizables que se puedan manipular visualmente en una herramienta de construcción".

Dentro de un JavaBean podemos distinguir tres partes:

- Propiedades: Los atributos que contiene.
- Métodos: Se establecen los métodos get y set para acceder y modificar los atributos.

- Eventos: Permiten comunicarnos con otros JavaBeans.

Footer. En el diseño Web se utiliza el término para señalar la parte Pie de de Página (la parte más baja de cualquier página Web).

Framework[14]. Desde el punto de vista del desarrollo de software, un framework es una estructura de soporte definida, en la cual otro proyecto de software puede ser organizado y desarrollado.

Los frameworks suelen incluir:
- Soporte de programas.
-Bibliotecas.
- Lenguaje de scripting.
- Software para desarrollar y unir diferentes componentes de un proyecto de desarrollo de programas.

Los frameworks permiten:
- Facilitar el desarrollo de software.
- Evitar los detalles de bajo nivel, permitiendo concentrar más esfuerzo y tiempo en identificar los requerimientos de software.

GlassFish. es un servidor de aplicaciones que implementa la plataforma **JavaEE5,** por lo que soporta las últimas versiones de tecnologías como:Icefaces, JSP, JSF, Servlets, EJBs, Java API para Servicios Web (JAX-WS), Arquitectura Java para Enlaces XML (JAXB), Metadatos de Servicios Web para la Plataforma Java 1.0, y muchas otras tecnologías.

Header. En el diseño Web se utiliza para definir al encabezado (la parte superior de cualquier página Web).

Hibernate es una herramienta de Mapeo objeto-relacional para la plataforma Java (y disponible también para .Net con el nombre de NHibernate) que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) que permiten establecer estas relaciones.

Hibernate es software libre, distribuido bajo los términos de la licencia GNU LGPL.

HQL. Hibernate ofrece también un lenguaje de consulta de datos llamado **HQL** (*Hibernate Query Language*) similar a SQL.

ICEfaces es un framework de desarrollo web creado sobre la especificación JSF y con capacidad de procesamiento de solicitudes AJAX, que nos permiten a los desarrolladores web construir aplicaciones con contenido enriquecido, programando únicamente en java y sin tener que agregar un applet u objetos que dependan de complementos propios del navegador.

IDE. (Integrated Development Environment - Entorno integrado de desarrollo). Aplicación compuesta por un conjunto de herramientas útiles para un programador.

Un entorno IDE puede ser exclusivo para un lenguaje de programación o bien, poder utilizarse para varios. Suele consistir de un editor de código, un compilador, un debugger y un constructor de interfaz gráfica GUI.un ejemplo de IDE es Netbeans.

Instanciado. El término instancia se refiere siempre a un objeto creado por el objeto-clase en tiempo de ejecución y por supuesto, pueden tener propiedades y métodos. En la aplicación, por ejemplo, la clase helper se encarga de ejecutar los querys entonces si necesitamos en el controlador hacer un query cuando este se crea se dice que se instancio un objeto de la clase helper.

Interfaz[14]. En software, parte de un programa que permite el flujo de información entre un usuario y la aplicación, o entre la aplicación y otros programas o periféricos. Esa parte de un programa está constituida por un conjunto de comandos y métodos que permiten estas intercomunicaciones.

ISBN. International Standard Book Number (en español, ‘número estándar internacional de libro’).

Mapeo. Es un archivo XML que describe las características de los POJOS.

Modelo MVC (Modelo Vista Controlador). Es un patrón de diseño que separa la estructura de una aplicación web en tres partes el Modelo se encarga de los datos, el controlador de la comunicación entre todos los elementos y las vista despliega los resultados para aplicaciones Java Web (también es conocido como modelo 2).

Chrome. Es un Navegador de Internet distribuido por Google.

Persistencia. Se llama "persistencia" de los objetos a su capacidad para guardarse y recuperarse desde un medio de almacenamiento.

POJOS (Plain Old Java Object). Es una implementación de un bean el cual da persistencia a los datos obtenidos desde la base de datos.

RIA. Rich Internet Applications (Aplicaciones de Internet Enriquecidas) Son aplicaciones web que tienen la mayoría de las características de las aplicaciones tradicionales, estas aplicaciones utilizan un "navegador web" estandarizado para ejecutarse y por medio de "plugin" o independientemente una "virtual machine", se agregan las características adicionales.

Esta surge como una combinación de las ventajas que ofrecen las aplicaciones Web y las aplicaciones tradicionales. Buscan mejorar la experiencia del usuario.

Normalmente en las aplicaciones Web, hay una recarga continua de páginas cada vez que el usuario pulsa sobre un enlace. De esta forma se produce un tráfico muy alto entre el cliente y el servidor, llegando muchas veces, a recargar la misma página con un mínimo cambio.

En los entornos RIA, en cambio, no se producen recargas de página, ya que desde el principio se carga toda la aplicación, y sólo se produce comunicación con el servidor cuando se necesitan datos externos como datos de una Base de Datos o de otros ficheros externos.

Tupla. En matemáticas, es una secuencia ordenada de objetos, esto es, una lista con un número limitado de objetos (una secuencia infinita se denomina en matemática como unafamilia). Las tuplas se emplean para describir objetos matemáticos que tienen estructura, es decir que son capaces de ser descompuestos en un cierto número de componentes. Por ejemplo, un Grafo dirigido se puede definir como una tupla de (V, E) donde V es el conjunto de nodos y E es el subconjunto de V × V que denota los vértices del grafo.

Bibliografía

[1] Basham, K Sierra y B. Bates. *Head First Servlets & JSP. Passing the Sun Certified Web Component Developer Exam*. Sebastopol: O'Reilly Media, Inc, 2008.

[2] Delobel C, Adiba M. Bases de données et systèmes relationels. Paris : Dunod, 1982.

[3] Degoulet P, Fieschi, M. Systèmes de gestion de bases de données. Paris, 1995.

[4] T. Downey. *Web Development with Java: Using Hibernate, JSPs and Servlets*. Miami: Springer, 2007.

[5] B. Eckel. *Piensa en Java*. 4a ed. Madrid: Prentice-Hall, 2007.

[6] J. García, J. I. Rodríguez. *Aprenda Java como si estuviera en primero*. Escuela Superior de Ingenieros Industriales de la Universidad de Navarra, 1999.

[7] J. Gosling, H. McGilton. *The Java language environment: a white paper*, Sun Microsystems Computer Co., 1995.

[8] K. E. Kendall y J. E. Kendall. *Análisis y diseño de sistemas*. 6a ed. México: Pearson Educación, 2005.

[9] Proteco. "Introducción al diseño de bases de datos" Notas para el curso Introducción al diseño de bases de datos. Dirección General de Servicios de Cómputo Académico. UNAM. Septiembre 2007.

[10] E. Robson, E. T. Freeman. *Head First HTML with CSS & XHTML*. Sebastopol: O'Reilly Media, Inc, 2005.

[11] A.Stellman y J. Greene. *Head First PMP: A Brain-Friendly Guide to Passing the Project Management Professional Exam*. Sebastopol: O'Reilly Media, Inc, 2007.

[12] E. Suárez. "Diseño e Implementación de Bases De Datos Relacionales" Notas para el Curso Bases de datos. Programa de tecnología en cómputo. Facultad de Ingeniería, UNAM. Junio 2009.

[13] A. Vázquez. "Bases de Datos" Notas para el curso de Bases de datos. Facultad de Ingeniería, UNAM.

[13b] M. Adoración and M. Piattini. *Fundamentos y modelos de Bases de Datos.* México: Alfaomega, Ra-Ma, 1999.

[14] Alegsa, "Definición de Framework". [online]. Disponible: http://www.alegsa.com.ar/Dic/framework.php

[15] Alegsa, "Diccionario de informatica". [online]. Disponible: http://www.alegsa.com.ar

[16] Cafe au Lait Java News and Resources, "Java is Distributed," Marzo 1997. [Online]. Disponible: http://www.ibiblio.org/java/slides/hope/09.html

[17] Crespo, "Introducción a Hibernate," 2003-2007. [Online]. Disponible: http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=hibernate

[18] Departamento de Tratamiento de la Información y Codificación del Instituto de Física Aplicada del Consejo Superior de Investigaciones Científicas de España, "¿Qué es Java?," 1997-1999. [Online]. Disponible: http://www.iec.csic.es/criptonomicon/java/quesjava.html

[19] Desarrollo Fácil. Programación, diseño, usabilidad, SEO…, "¿Qué es el paradigma MVC (Modelo, Vista, Controlador)?," Abril 2010. [Online]. Disponible: http://www.desarrollofacil.com/que-es-el-paradigma-mvc-modelo-vista-controlador/

[20] Escuela Universitaria de Ingeniería de Vitoria. Universidad del País Vasco. "Laboratorio 6. Patrón MVC," Mayo 2007. [Online]. Disponible: http://lsi.vc.ehu.es/mikell/docencia/Programacion-II/Praktika/Integracion_MVC.pdf

[21] Equipo de Desarrollo PostgreSQL "Operaciones en el Modelo de Datos Relacional," 2010. [Online]. Disponible: http://www.ibiblio.org/pub/linux/docs/LuCaS/Postgresql-es/web/navegable/todopostgresql/x24684.html

[22] Esepé Studio. Especialistas Web, "Qué es MySQL," 2003-2007. [Online]. Disponible: http://www.esepestudio.com/articulo/desarrollo-web/bases-de-datos- The Database mysql/Que-es-MySQL.htm

[23] Federico Cargnelutti/Wiki, "Un enfoque modular para el Desarrollo Web," Junio 2008. [Online]. Disponible: http://translate.google.com/translate?hl=en&sl=en&tl=es&u=http://articles.fedecarg.com/wiki/A_Modular_Approach_to_Web_Development

[24] Hibernate. Relational Persistence for Java and .NET, "HIBERNATE - Relational Persistence for Idiomatic Java," Junio 2009. [Online]. Disponible: http://docs.jboss.org/hibernate/core/3.3/reference/en/html/tutorial.html

[25] Oracle, "The Java Technology Phenomenon," 1995, 2010. [Online]. Disponible: http://java.sun.com/docs/books/tutorial/getStarted/intro/definition.html

[26] Programación en Castellano, "Persistencia de Objetos Java utilizando Hibernate," Febrero 2005. [Online]. Disponible: http://www.programacion.com/articulo/persistencia_de_objetos_java_utilizando_hibernate_306

[27] Sun,"Glassfish". [online]. Disponible: http://blogs.sun.com/AlanVargas/entry/qu%C3%A9_es_glassfish

[28] The Database Programmer, "Using a Data Dictionary," Junio 2008. [Online]. Disponible: http://database-programmer.blogspot.com/2008/06/using-data- dictionary.html

[29] Universidad Jaume I, "Diseño de bases de datos," Febrero 2001. [Online]. Disponible: http://www3.uji.es/~mmarques/f47/apun/node69.html

[30] Wikipedia, "Ajax," Julio 2010. [Online]. Disponible: http://es.wikipedia.org/wiki/AJAX

[31] Wikipedia, "CSS", Julio 2010 [Online]. Disponible: http://en.wikipedia.org/wiki/Cascading_Style_Sheets

[32] Wikipedia, “GlassFish,” Junio 2010. [Online]. Disponible: http://es.wikipedia.org/wiki/GlassFish

[33] Wikipedia, “Hojas de estilo en cascada,” Julio 2010. [Online]. Disponible: http://es.wikipedia.org/wiki/Hojas_de_estilo_en_cascada

[34] Wikipedia, “HTML,” Julio 2010. [Online]. Disponible: http://es.wikipedia.org/wiki/HTML

[35] Wikipedia, “ICEfaces,” Junio 2010. [Online]. Disponible: http://en.wikipedia.org/wiki/ICEfaces

[36] Wikipedia, “ISBN,” Julio 2010. [Online]. Disponible: http://es.wikipedia.org/wiki/ISBN

[37] Wikipedia, “Modelo entidad-relación,” Julio 2010. [Online]. Disponible: http://es.wikipedia.org/wiki/Modelo_entidad-relacion

[38] Wikipedia, “NetBeans,” Junio 2010. [Online]. Disponible: http://es.wikipedia.org/wiki/NetBeans

[39] Wikipedia, “Web template,” Junio 2010. [Online]. Disponible: http://en.wikipedia.org/wiki/Web_template