Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Investigación de Operaciones y Modelos Matemáticos

DEUSWEB 1.0
Generador de sitios web administrativos basado en la ingeniería de reverso al modelo de datos

Trabajo Especial de Grado
Presentado ante la Ilustre
Universidad Central de Venezuela
Por el Bachiller:
Luis Antonio González Reyes. CI: 14.362.740
Para optar al título de
Licenciado en Computación

Tutor:
Profesor Sergio Rivas

Caracas, Octubre / 2006

ACTA

Quienes suscriben, miembros del Jurado, designado por el Consejo de Escuela de Computación, para examinar el Trabajo Especial de Grado presentado por el **Bachiller Luis Antonio González Reyes, C.I. 14.362.740**, con el título **"DEUSWEB Generador de sitios web administrativos basados en la ingeniería de reverso al modelo de datos"**, a los fines de optar al titulo de Licenciado en Computación, dejan constancia de lo siguiente:

Leído como fue, dicho trabajo por cada uno de los miembros del Jurado, se fijo el día 27 de Octubre de 2006 a las 3:00 p.m., para que su autor lo defendiera en forma pública, se hizo en el Auditorio Manuel Bemporad de la Escuela de Computación, mediante una presentación oral de su contenido, luego de lo cual fueron respondidas las preguntas formuladas. Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobarlo con una nota de _____ puntos, en fe de lo cual se levanta la presente Acta, en Caracas a los 27 días del mes de octubre de año dos mil seis, dejándose también constancia de que actuó como Coordinador del Jurado, el profesor Sergio Rivas.

Prof. Sergio Rivas
Tutor – Jurado Principal

Prof. Joselito de Sousa
Jurado Principal

Prof. Johnny Sepúlveda
Jurado Principal

DEDICATORIA Y AGRADECIMIENTOS

El presente Trabajo Especial de Grado esta dedicado a mi madre Chiquinquirá Reyes, quien me ha entregado su apoyo incondicional a lo largo de todos mis estudios y me ha enseñado a valorar los mismos.

Gracias a mi padre Luis Enrique González por contribuir en la educación que he recibido y a mis hermanos Cesar y Francisco por su apoyo en mis estudios.

Quiero agradecer a mis compañeros de estudio: Jazmín Rincón, Mayerling Sánchez y Viviana Montoya, por ayudarme en los momentos difíciles de la carrera y a los profesores de la Escuela de Computación por la formación académica que he recibido.

Finalmente, quiero agradecer especialmente, a mi padrino Omar Azpúrua, a mi madrina Melita Reyes de Azpúrua y a mis abuelas Flor Reyes e Isabel Fuentes, por brindarme su apoyo en los momentos más difíciles de mi vida y demostrarme que unidos en familia todo se puede lograr.

RESUMEN

El sistema DEUSWEB desarrollado, tiene como objetivo generar de forma automática aplicaciones web que se encarguen de gestionar la información contenida en una base de datos, contribuyendo así a minimizar el tiempo en el desarrollo de este tipo de aplicaciones. La gestión que realiza el sistema sobre la base de datos incluye: listar, buscar, insertar, modificar y eliminar cualquier información contenida en las tablas de la base de datos.

DEUSWEB es una aplicación desarrollada utilizando como lenguaje de programación JAVA, la cual, recibe como entrada una base de datos existente, y apartir de esta, extrae sus metadatos. Con la información obtenida de los metadatos y junto con la información de configuración del sitio suministrada por el usuario, se genera la aplicación web.

La creación de los archivos fuentes del sitio generado, es llevada a cabo por un solo motor implementado en la aplicación: *JSP / Servlets*, el cual es constituido por un conjunto de procesadores. Para este trabajo especial grado solo se encuentra implementado este motor, sin embargo, se pueden agregar y configurar nuevos motores en la aplicación de una forma fácil, en próximas versiones. Los procesadores son los encargados de realizar las tareas para la generación del sitio, tareas como: crear la estructura de directorios, crear los *Servlets*, crear las vistas (JSP), compilar los archivos generados, empaquetar el sitio, etc.

El propósito de este Trabajo Especial de Grado consiste en desarrollar el sistema que permita la creación de sitios web administrativos a partir de una base de datos existente, a través del proceso de desarrollo de software: Proceso Unificado. El alcance de este desarrollo involucra generar código JSP / Servlets a partir de un base de datos MySQL.

Para la creación de este sistema, se estudiaron tres herramientas existentes que generan aplicaciones web administrativas a partir de la ingeniería de reverso al modelo de datos. Del estudio de estas aplicaciones, surgieron las funcionalidades desarrolladas en el sistema. Sin embargo el sistema desarrollado incluye nuevas funcionalidades con respecto a los sistemas evaluados, por ejemplo: el manejo de las claves foráneas y la manipulación de los archivos almacenados en la base de datos.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

INTRODUCCIÓN

En la actualidad, el desarrollo de páginas web esta creciendo día a día, permitiendo a las empresas, e inclusive, a personas particulares divulgar información, a través de Internet, por el mundo entero. Esta información muchas veces esta contenida en un repositorio de información o base de datos, permitiendo así, que la página web posea un contenido dinámico extraído del repositorio de información. Siguiendo este esquema de desarrollo en los sitios web, la empresa o persona particular, solo debe gestionar la información contenida en la base de datos para cambiar la información de su sitio.

Una de las formas existentes para gestionar la información contenida en la base de datos, es a través de un sistema administrativo que permita realizar dicha gestión, pero muchas veces las empresas o personas particulares no desarrollan este tipo de sistemas por los costos de producción involucrados: dedicación horas-hombre, costos operativos, etc. Un sistema administrativo posee ciertas funcionalidades comunes que son independientes de la información contenida en el repositorio de información, tales como: listar, buscar, insertar, modificar y eliminar cualquier dato o información contenido en las tablas de la base de datos.

En el presente Trabajo Especial de Grado, se plantea el desarrollo de un sistema que permita crear sistemas web administrativos, capaces de gestionar la información contenida en una base de datos, realizando las funcionalidades principales descritas anteriormente. Este sistema, deberá recibir como entrada una base de datos, en el caso particular para este trabajo deberá ser MySQL, y a partir de esta, identificará todas las tablas y campos de esas tablas a través de la extracción de los metadatos de la base de datos, para así generar el sistema web administrativo. Es por ello que se concibe la ingeniería de reverso al modelo de datos como parte fundamental en el desarrollo del presente Trabajo Especial de Grado.

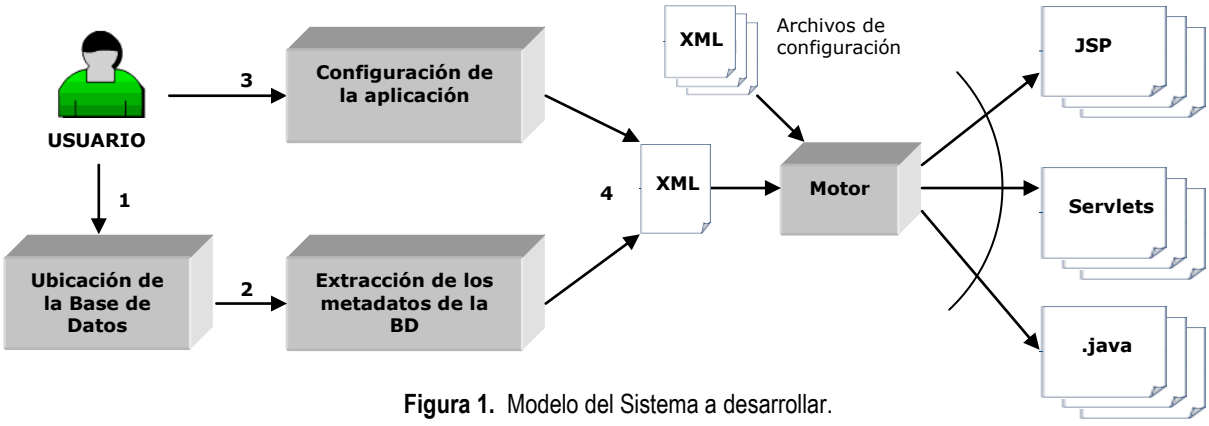


Figura 1. Modelo del Sistema a desarrollar.

A continuación se presentan los objetivos generales y los objetivos específicos que se plantean en el desarrollo de este Trabajo Especial de Grado, así como el alcance que tendrá el mismo. Seguidamente se describirá el contenido de cada capítulo.

Objetivos Generales:

Desarrollar un sistema que permita crear sitios web administrativos a partir de una base de datos existente. La aplicación web generada deberá cumplir con las siguientes funcionalidades sobre las tablas de la base de datos: autenticar al usuario, listar, buscar, agregar, modificar y eliminar.

Objetivos Específicos:

- Desarrollar la aplicación a generar siguiendo un proceso de desarrollo a fin de conocer que archivos deberán ser generados por el sistema generador.
- Diseñar el sistema generador de forma tal que sea mantenible e incrementable.
- Desarrollar la interfaz de usuario del sistema generador.
- Extraer la información de los metadatos de la base de datos.
- Desarrollar los motores que se encarguen de producir el código fuente de la aplicación web a generar.
- Verificar las funcionalidades del sistema generado a través de un caso de estudio.

Alcance:

Del Sistema Generador:

- Será desarrollado utilizando como lenguaje de programación JAVA.
- Corresponderá a un sistema "*standalone*".
- El Sistema Manejador de Base de Datos a utilizar será MySQL.
- La configuración del sistema será implementada a través de documentos XML. Ejemplo: Sistemas Manejadores de Base de Datos existentes, tipo de código fuente a producir y motores existentes.

La aplicación generadora será denominada en lo sucesivo como DEUSWEB.

Del Sistema Generado:

- Será desarrollado utilizando como lenguaje de programación JAVA, implementando las tecnologías JSP y Servlets.
- Deberá cumplir con el patrón arquitectónico MVC.
- La vista de la aplicación dependerá de las hojas de estilo (css).
- Corresponderá a un sistema basado en ambiente web.
- Cumplirá con las funcionalidades básicas: autenticar al usuario, listar, buscar, insertar, modificar, y eliminar cualquier información contenida en la base de datos.

El presente Trabajo Especial de Grado consta de dos partes: el *Marco Teórico*, el cual abarca la teoría e investigación previa para el cumplimiento de los objetivos que se plantean y el *Marco Aplicativo* el cual comprende el proceso de desarrollo utilizado tanto en el sistema generador, como en el sistema generado.

El *Marco Teórico*, se compone de los siguientes capítulos:

Capítulo I: Conceptos de Base de Datos y Conectividad.

En este capitulo, se describen los conceptos fundamentales para el cumplimiento del objetivo general de este Trabajo Especial de Grado. Se incluyen conceptos del Modelo Entidad – Relación y conceptos del modelo relacional, así como también el concepto de metadato, fundamental en el desarrollo de este Trabajo. Este capítulo finaliza, con la descripción del API JDBC, el cual posteriormente es utilizado en la implementación del sistema.

Capitulo II: Herramientas Existentes.

En este capítulo se estudian algunas herramientas existentes en el mercado, que permiten crear aplicaciones web administrativas a partir de la ingeniería de reverso al modelo de datos. Se describe el caso de estudio con el que se evaluó a cada una de las aplicaciones, así como los aspectos que fueron tomados en cuenta para la evaluación.

En el *Marco Aplicativo*, se desarrollan los siguientes capítulos:

Capítulo III: Proceso de Desarrollo a Utilizar.

Se describe el proceso de desarrollo de software utilizado para generar el sistema generador y el sistema generado, señalando cada Flujo de Trabajo y cada Etapa componente de este proceso.

Capítulo IV: Aplicación a Generar.

Se describen las Etapas del Proceso de Desarrollo en la implementación de la aplicación web administrativa. Es necesario desarrollar este sistema, aplicando el proceso de desarrollo, antes de desarrollar la aplicación generadora para conocer que archivos serán necesarios producir por esta última aplicación. En este capitulo se detalla cada uno de los artefactos generados por los flujos de trabajo en cada una de las etapas del proceso de desarrollo.

Capitulo V: Aplicación Generadora.

En este capítulo, se describe el proceso de desarrollo implementado en la aplicación DEUSWEB, la cual genera un sistema web administrativo que gestiona la información contenida en una base de datos. Se describe cada uno de los artefactos generados por el sistema en cada una de las iteraciones del proceso de desarrollo.

Capitulo VI: Ejemplo de Generación (Caso de Estudio).

Este capitulo representa la última etapa del proceso de desarrollo de la aplicación generadora. En él se describe cada uno de los pasos que deben realizarse para la creación de un sitio web administrativo, así como también su posterior implantación en un servidor web. Una vez instalado en el servidor, se verifican cada una de las funcionalidades de la aplicación generada.

PARTE I

Marco Teórico

CAPÍTULO I:

CONCEPTOS DE BASE DE DATOS Y CONECTIVIDAD

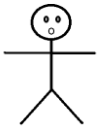
El presente capítulo, reúne los conceptos asociados al modelo lógico basado en objetos y al modelo lógico basado en registros. El primer modelo describe la base de datos a nivel conceptual y de visión, está representado por el Modelo Entidad – Relación. El segundo modelo, describe los datos en los niveles conceptual y físico y está representado por el modelo Relacional. Así mismo en este capítulo se encuentran los conceptos relacionados con la conectividad hacia la base de datos (Metadata y JDBC). Estos conceptos son fundamentales el desarrollo de la parte aplicativa.

1.1. Modelo Entidad – Relación

Fue desarrollado por Peter Chen en 1976 [1] como una metodología gráfica para el diseño de Base de Datos. Puede decirse que es un modelo de representación abstracta del mundo real centrado en las restricciones o propiedades lógicas de una Base de Datos. Este modelo consiste en un conjunto de objetos básicos llamados entidades y de relaciones entre estos objetos. Se emplea para interpretar, especificar y documentar los requerimientos para sistemas de procesamiento de bases de datos.

1.1.1. Entidades

Una *entidad* es un objeto que existe y es distinguible de otros objetos, por ejemplo Juan González, con número de cedula 4.593.752, es una entidad, ya que identifica únicamente a una persona específica. Una entidad puede ser concreta, como un libro, o abstracta, como un concepto. Esta representada por un conjunto de atributos. Cada entidad se describe por medio de un conjunto de pares (atributo, valor).



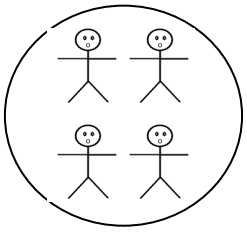
Pedro Pérez.
Cédula de identidad
Nº 14.458.978

Figura 2. Representación de una Entidad.

[1] Ver Chen. *The entity-relationship model-towards a unified view of data.* p.p. 9-36

1.1.2.Conjuntos de Entidades

Un *conjunto de entidades*, también llamado *clase de entidad*, corresponde a una agrupación de entidades del mismo tipo, por ejemplo, en un banco, el conjunto de entidades cliente. Los conjuntos de entidades no necesitan ser disjuntos, es posible definir los conjuntos de entidades de empleados y clientes de un banco, pudiendo ser una persona, ambas o ninguna de las dos anteriores. El concepto de conjunto de entidades corresponde a la noción de definición de tipo en un lenguaje de programación. Una Base de Datos incluye una colección de conjuntos de Entidades cada uno de los cuales contiene un número cualquiera de entidades del mismo tipo. Un Conjunto de Entidades es representado por una tabla en el modelo relacional.



Conjunto de Entidades:

Persona

Figura 3. Representación de un Conjunto de Entidades.

1.1.3.Atributos:

Los *atributos* o *propiedades*, describen las características de una entidad, por ejemplo: nombre, cédula, teléfono. Dentro de los tipos de atributos tenemos:

- ***Atómico:*** Los atributos que no son divisibles, por ejemplo: nombre, edad, sueldo, etc.

- ***Compuestos:*** Cuando un atributo es dividido en pequeñas subpartes. Estos atributos pueden formar una jerarquía de atributos.

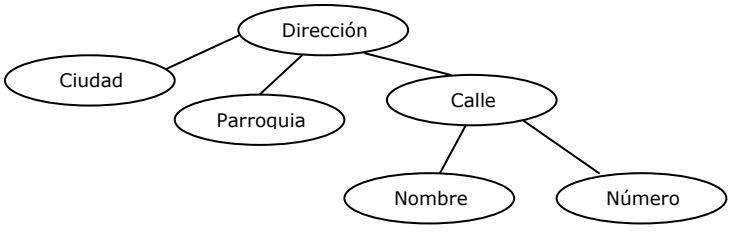


Figura 4. Ejemplo de atributos compuestos.

- ***Derivados:*** Cuando los valores de un atributo son afines, y el valor para este tipo de atributo se puede derivar de los valores de otros atributos. Por ejemplo, la edad y la fecha de

nacimiento de una persona. Si se conoce la fecha de nacimiento, se puede determinar su edad.

- ***Clave:*** Corresponde a un atributo sobre el cual los valores son distintos para cada entidad individual. Este tipo de atributo permite identificar de forma única a una entidad en el conjunto de entidades. Por ejemplo en la entidad Persona, el atributo clave puede ser el número de la cédula.

1.1.4.Relaciones

Una relación R es cualquier asociación, que puede establecerse entre entidades de la misma clase o de clases diferentes.

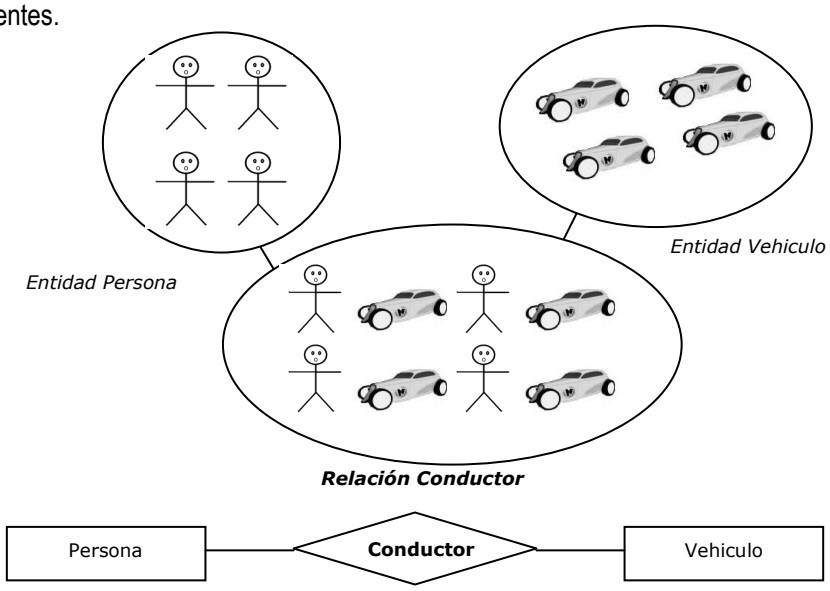


Figura 5. Ejemplo de una Relación entre una Entidad Persona y una Entidad Vehiculo.

Una relación puede incluir muchas entidades. La cantidad de entidades en una relación es el grado de la relación.

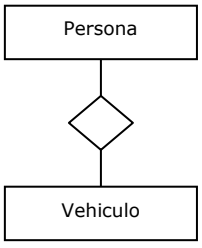


Figura 6. Relación de grado 2

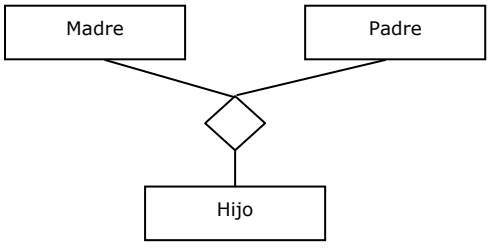


Figura 7. Relación de grado 3

Tipos de Relaciones

- Relaciones Unitarias: Se establecen entre entidades de la misma clase. Por ejemplo Personas son hijos de Personas.
- Relaciones Binarias: Se establecen entre dos entidades. Son relaciones de grado 2. Ejemplo: Profesor trabaja en Facultad.
- Relaciones N-arias: Se establecen entre N clases de entidades, siendo N>2.

Propiedades de las Relaciones

- Roles: Son las funciones que desempeñan cada una de las clases de entidades asociadas. En toda relación existen dos roles diferentes correspondientes a las entidades de cada una de las clases relacionadas. Ejemplo: En la *figura 5*, podría establecerse el siguiente rol para cada clase: *conduce un / es conducido por.*
- Cardinalidad: La cardinalidad de una relación expresa el número de entidades de una clase que pueden asociarse a una entidad de la otra clase. En función de esta propiedad se distinguen tres tipos de relaciones:
  - Relación 1:1, indica que una entidad del conjunto de entidades A solo se puede relacionar con una entidad del conjunto de entidades B. Ejemplo: Una Persona conduce un vehículo.
  - Relación 1:N, indica que una entidad del conjunto de entidades A se puede relacionar con N entidades del conjunto de entidades B. Ejemplo: Una Persona conduce varios vehículos.
  - Relación N:M, indica que N entidades del conjunto de entidades A se puede relacionar con N entidades del conjunto de entidades B. Ejemplo: Varias Personas conducen varios vehículos.

1.2. Modelo Relacional

El modelo relacional, representa los datos y las relaciones entre ellos mediante una colección de tablas, cada una de las cuales tiene un número de columnas con nombres únicos. A continuación, se definen algunos conceptos relacionados con este modelo.

1.2.1. Tabla:

Una tabla representa a un conjunto de entidades o clases de entidades del modelo Entidad – Relación. Igualmente una tabla identifica a una relación del modelo Entidad – Relación.

CÉDULA	NOMBRE	EDAD	SEXO
14.362.740	Luis Antonio González	25	M
14.458.978	Pedro Pérez	24	M
18.789.654	Ana Fernández	18	F
19.123.456	José Reyes	16	F
20.147.589	Isabel Fuentes	19	F

CÉDULA	SALDO
14.362.740	5.000.000
14.458.978	300.000
18.789.654	70.000
19.123.456	1.000.000
20.147.589	900.000

Figura 8. Ejemplo de Tablas en el Modelo Relacional.

1.2.2. Campos

Es la unidad básica de una Tabla. Un campo puede ser, por ejemplo, el nombre de una persona.

CÉDULA	NOMBRE	EDAD	SEXO
14.362.740	Luis Antonio González	25	M
14.458.978	Pedro Pérez	24	M
18.789.654	Ana Fernández	18	F
19.123.456	José Reyes	16	F
20.147.589	Isabel Fuentes	19	F

Campo

Figura 9. Representación de un campo en una tabla.

1.2.3. Tipos de Campos

- Texto: Para introducir cadenas de caracteres hasta un máximo de 255.
- Memo: Para introducir un texto extenso. Hasta 65.535 caracteres
- Numérico: Para introducir números.
- Fecha/Hora: Para introducir datos en formato fecha u hora.
- Moneda: Para introducir datos en formato número y con el signo monetario.
- Autonumérico: En este tipo de campo se numera automáticamente el contenido.

- Si/No: Campo lógico. Este tipo de campo es sólo si queremos un contenido del tipo Sí/No, Verdadero/Falso, etc.
- Objeto OLE: Para introducir una foto, gráfico, hoja de cálculo, sonido, etc.
- Hipervínculo: Podemos definir un enlace a una página Web.
- Asistente para Búsquedas: Crea un campo que permite elegir un valor de otra tabla o de una lista de valores mediante un cuadro de lista o un cuadro combinado.

1.2.4. Dominio

Es el conjunto finito de valores homogéneos (del mismo tipo) y atómicos (indivisibles), caracterizados por un nombre. Ejemplo:

- Dominio de Nacionalidades: venezolana, colombiana, ecuatoriana, etc.
- Dominio Edad de Empleado: posible edad de los empleados de una compañía; un valor entre 18 y 60.

Todo dominio tiene un nombre y un tipo de dato. En el dominio de nacionalidades el tipo de dato es un conjunto de caracteres de longitud máxima 20.

1.2.5. Atributos

Representan las columnas de la tabla. Cada una de las columnas de la tabla posee un nombre único dentro de la misma. Un atributo también se puede definir como el papel que tiene un determinado dominio en una relación.

CÉDULA	NOMBRE	EDAD	SEXO
14.362.740	Luis Antonio González	25	M
14.458.978	Pedro Pérez	24	M
18.789.654	Ana Fernández	18	F
19.123.456	José Reyes	16	F
20.147.589	Isabel Fuentes	19	F

Figura 10. Representación de un atributo en una tabla.

1.2.6. Tupla

Corresponde con una fila de la tabla. Representa una entidad dentro de la tabla.

CÉDULA	NOMBRE	EDAD	SEXO
14.362.740	Luis Antonio González	25	M
14.458.978	Pedro Pérez	24	M
18.789.654	Ana Fernández	18	F
19.123.456	Jose Reyes	16	F
20.147.589	Isabel Fuentes	19	F

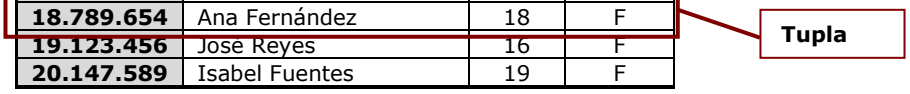


Figura 11. Representación de una tupla en una tabla.

1.2.7. Clave Primaria

La clave primaria, corresponde a un campo o conjunto de campos que identifican unívocamente a una tupla, por ejemplo la cédula de identidad en la tabla Persona. El o los campos de la clave primaria no pueden tomar valores nulos. Una clave primaria, debe cumplir dos condiciones:

- **Unicidad**: No pueden existir dos o más entidades con el mismo valor en todos los atributos que forman la clave.
- **Minimidad**: No existe ningún subconjunto de la clave primaria que cumpla la regla de unicidad.

1.2.8. Clave Foránea

Corresponde a un campo o conjunto de campos de una tabla cuyos valores coinciden con los valores de la clave primaria de otra tabla. Esta clave es el mecanismo fundamental para relacionar tablas entre si. La clave foránea y la correspondiente clave primaria han de estar definidas sobre el mismo dominio.

1.2.9. Claves Compuestas

Se dice que una clave es compuesta cuando dos o más campos componen la clave.

1.2.10. Esquema

Es la descripción de la base de datos. Se realiza mediante un conjunto de definiciones expresadas en un DDL (Lenguaje de definición de datos). El resultado de esta definición es un conjunto de tablas y relaciones que se almacenan en una tabla (o un conjunto de tablas) especial que se identifica como el diccionario de datos o el catálogo de la base de datos. El esquema se especifica durante la fase de diseño, y no es de esperar que se modifique a menudo. Para representar el esquema de una base de datos relacional se debe dar el nombre de sus relaciones, los atributos de éstas, los dominios sobre los que se definen estos atributos, las claves primarias y las claves foráneas. Ejemplo: Esquema de la base de datos de una empresa inmobiliaria:

OFICINA (Onum, Calle, Área, Población, Teléfono, Fax)

EMPLEADO (Enum, Nombre, Apellido, Dirección, Teléfono, Puesto, Fecha_nac, Salario, Onum)
INMUEBLE (Inum, Calle, Área, Tipo, Hab, Alquiler, Pnum, Enum, Onum)
INQUILINO (Qnum, Nombre, Apellido, Dirección, Teléfono, Tipo_pref, Alquiler_max)
PROPIETARIO (Pnum, Nombre, Apellido, Dirección, Teléfono)
VISITA (Qnum, Inum, Fecha, Comentario)

En el esquema, los nombres de las relaciones aparecen seguidos de los nombres de los atributos encerrados entre paréntesis. Las claves primarias son los atributos subrayados. Las claves foráneas se representan mediante los siguientes diagramas referenciales.

EMPLEADO	$\xrightarrow{Onum}$	OFICINA	:	Oficina a la que pertenece el empleado.
INMUEBLE	$\xrightarrow{Pnum}$	PROPIETARIO	:	Propietario del inmueble.
INMUEBLE	$\xrightarrow{Enum}$	EMPLEADO	:	Empleado encargado del inmueble.
INMUEBLE	$\xrightarrow{Onum}$	OFICINA	:	Oficina a la que pertenece el inmueble.
VISITA	$\xrightarrow{Qnum}$	INQUILINO	:	Inquilino que ha visitado el inmueble.
VISITA	$\xrightarrow{Inum}$	INMUEBLE	:	Inmueble que ha sido visitado.

1.2.11. Restricciones

Restricciones Inherentes

- Definición matemática de la relación:
  - No hay dos tuplas iguales.
  - El orden de las tuplas no es significativo.
  - El orden de los atributos no es significativo.
  - Los valores de los atributos son atómicos.

- Regla de Integridad de Entidad: Se refiere a que los campos que conforman la clave primaria no pueden tomar valores nulos, ya que esta clave debe permitir identificar unívocamente cada tupla de la tabla.

Restricciones Semánticas (de usuario)

- Integridad Referencial: Se refiere a que todos los valores no nulos de una clave foránea referencian valores reales de la clave referenciada. Ejemplo: El profesor que imparte una asignatura debe existir en la tabla profesores. La integridad referencial mantiene las conexiones en las bases de datos relacionales.

1.2.12. Borrado en Cascada

El borrado en cascada consiste en que cuando se elimina o modifica una entidad (tupla) del conjunto de entidades (tablas) que contiene la clave primaria referenciada, se deben borrar o modificar todas las tuplas de las tablas que contienen la clave foránea. Por ejemplo: Al modificar el número de un departamento, se debe modificar el número de departamento de todos los empleados asignados a ese departamento. Para el caso de la eliminación, se deben eliminar todos los empleados asignados a ese departamento.

1.3. Metadato

Metadato es la información acerca de la estructura de la base de datos y cada Sistema Manejador de Base de Datos (SMBD) posee diversos mecanismos que posibilitan su acceso.

El modelo relacional sirve de base a los Sistemas Manejadores de Base de Datos Relacionales (SMBDR), que emplean un lenguaje de consulta (*Structure Query Languaje SQL*) para la creación y administración de las bases de datos. Este lenguaje de consulta es un lenguaje estándar que se divide en dos categorías: lenguaje de definición de datos (DDL) y lenguaje de manipulación de datos (DML).

1.4. JDBC

Es el acrónimo de *Java Database Connectivity*, el cual constituye un API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java. Es independiente de la base de datos a la cual se accede utilizando el lenguaje SQL del modelo de base de datos que se utilice.

El API JDBC se presenta como una colección de interfaces Java y métodos de gestión de manejadores de conexión hacia cada modelo específico de base de datos. Un manejador de conexiones (driver) hacia un modelo de base de datos en particular es un conjunto de clases que implementan las interfaces Java y que utilizan los métodos de registro para declarar los tipos de localizadores a base de datos (URL) que pueden manejar. Este driver es el que implementa la funcionalidad de todas las clases de acceso a datos y

proporciona la comunicación entre el API JDBC y la base de datos real. Para utilizar una base de datos particular, el usuario ejecuta su programa junto con la librería de conexión apropiada al modelo de su base de datos, y accede a ella estableciendo una conexión, que se explicará a continuación.

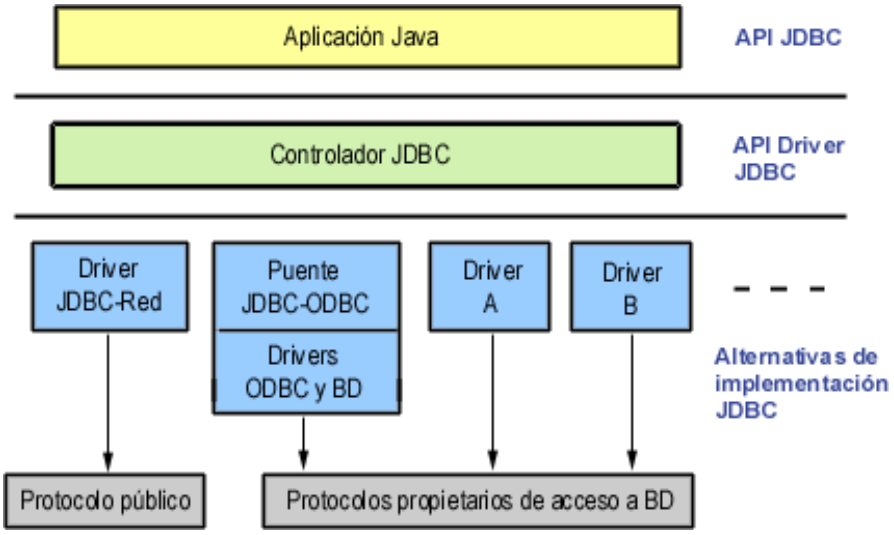


Figura 12. Estructura JDBC. [2]

1.4.1. Como usar el API JDBC

JDBC define ocho interfaces para operaciones con bases de datos, de las que se derivan las clases correspondientes. La siguiente figura, muestra la interrelación entre estas clases según el modelo de objetos de la especificación de JDBC.

[2] Imagen extraida de: *http://www.itapizaco.edu.mx/paginas/JavaTut/froufe/parte21/cap21-3.html*

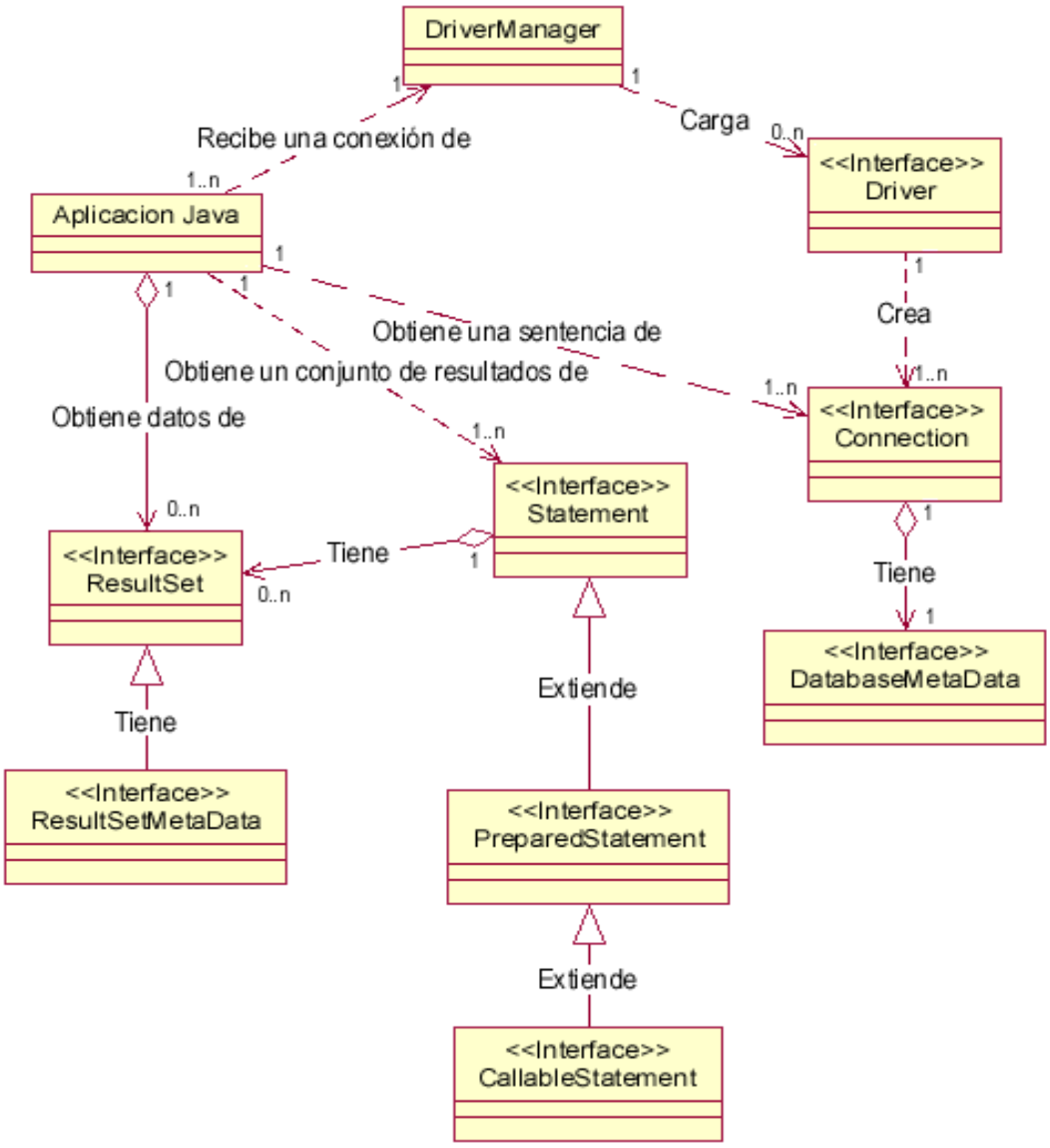


Figura 13. Interrelación de clases en la especificación JDBC. [3]

Lo primero que se debe hacer es cargar el driver que queremos utilizar y para ello se utiliza la siguiente instrucción:

```
Class.forName (String);
```

[3] Imagen extraída de: *http://www.itapizaco.edu.mx/paginas/JavaTut/froufe/parte21/cap21-4.html*

Donde el String pasado por parámetro corresponde al nombre del manejador de conexión. Ejemplo: Si queremos utilizar el puente JDBC-ODBC, se cargaría la siguiente línea de código.

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

Si queremos utilizar el driver de mysql, se cargaría la siguiente línea de código.

```
Class.forName("com.mysql.jdbc.Driver");
```

No se necesita registrar el driver cargado con el *DriverManager* porque la llamada a *Class.forName* lo hace automáticamente. La clase que se encarga de cargar inicialmente todos los drivers JDBC disponibles es entonces *DriverManager*.

Una aplicación puede utilizar *DriverManager* para obtener un objeto de tipo *Connection*, con una base de datos. La siguiente línea de código ilustra la idea general:

```
Connection con;
Con = DriverManager.getConnection(url, Login, Password);
```

Donde:

- **url**, corresponde a un *String* siguiendo una sintaxis basada en la especificación más amplia de los URL, de la forma: *jdbc:subprotocolo//servidor:puerto/base de datos.*
- **Login**, corresponde a un *String* que indica el nombre utilizado para entrar en el controlador de la base de datos.
- **Password**, corresponde a un *String* que indica el password para el controlador de la base de datos.

Ejemplo:

```
Connection conexion;
conexion = DriverManager.getConnection(jdbc:mysql://localhost/BD,
"mi_login", "mi_password");
```

Si uno de los drivers cargados reconoce la URL suministrada por el método *DriverManager.getConnection*, dicho driver establecerá una conexión con el controlador de base de datos especificado en la URL del JDBC.

Una vez que se tiene un objeto de tipo *Connection*, se pueden crear sentencias, *statements*, ejecutables. La siguiente instrucción permite crear un objeto de tipo *Statement*:

```
Statement sentencia = conexion.createStatement();
```

Mediante un objeto de tipo *Statement* se pueden ejecutar sentencias a través de esa conexión. Para ello se dispone de los métodos *execute(String sentencia)* para ejecutar una petición SQL que no devuelve datos o *executeQuery(String sentencia)* para ejecutar una consulta SQL. Este último método devuelve un objeto de tipo *ResultSet*.

A continuación se describe brevemente cada una de las interfaces y clases, correspondientes a la especificación de JDBC:

- *Driver*: Permite conectarse a una base de datos: cada gestor de base de datos requiere un driver distinto.
- *DriverManager*: Permite gestionar todos los drivers instalados en el sistema.
- *DriverPropertyInfo*: Proporciona diversa información acerca de un driver.
- *Connection*: Representa una conexión con una base de datos. Una aplicación puede tener más de una conexión a más de una base de datos.
- *DatabaseMetadata*: Proporciona información acerca de una Base de Datos, como las tablas que contiene, etc.
- *Statement*: Permite ejecutar sentencias SQL sin parámetros.
- *PreparedStatement*: Permite ejecutar sentencias SQL con parámetros de entrada.
- *CallableStatement*: Permite ejecutar sentencias SQL con parámetros de entrada y salida, típicamente procedimientos almacenados.
- *ResultSet*: Contiene las filas o registros obtenidos al ejecutar un SELECT.
- *ResultSetMetadata*: Permite obtener información sobre un ResultSet, como el número de columnas, sus nombres, etc.

1.4.2. ¿ Como obtener información relativa a un campo o columna de la Base de Datos ?

El objeto *ResultSet* devuelto por el método *executeQuery()*, permite recorrer las filas obtenidas, no proporciona información referente a la estructura de cada una de ellas; para ello se utiliza *ResultSetMetaData*, que permite obtener el tipo de cada campo o columna, su nombre, si es del tipo autoincremento, si es sensible a mayúsculas, si se puede escribir en dicha columna, si admite valores nulos, etc. Para obtener un objeto de tipo ResultSetMetaData basta con llamar al método *getMetaData()* del objeto *ResultSet*.

En la siguiente lista aparecen algunos de los métodos más importantes de ResultSetMetaData, que permiten averiguar toda la información necesaria para formatear la información correspondiente a una columna, etc. En la mayoría de los métodos, se recibe por parámetro el número de la columna cuyo valor empieza desde uno (1).

- *getCatalogName(int columna):* Retorna un String, correspondiente al nombre de la columna designado en el catalogo de la base de datos.
- *getColumnCount( ):* Retorna un entero (int) que indica el número de columnas que existen en el ResultSet.
- *getColumnDisplaySize (int columna):* Retorna un entero (int) correspondiente al máximo de caracteres designados en la columna.
- *getColumnLabel (int columna):* Retorna un String, correspondiente al título de la columna que se exhibe cuando se imprime o muestra el nombre de la misma.
- *getColumnName (int columna):* Retorna un String que indica el nombre de la columna designado.
- *getColumnType (int columna):* Retorna un String correspondiente al tipo de la columna (uno de los tipos SQL en java.sql.Types).
- *getColumnTypeName(int columna):* Retorna un String, que indica el nombre del tipo de dato asignado a la columna, usado por la base de datos.
- *getPrecision(int columna):* Retorna un entero (int) que representa el número de dígitos asignados a la columna.
- *getScale(int columna):* Retorna un entero (int) que representa el número de dígitos decimales asignados a la columna.
- *getSchemaName(int columna):* Retorna un String que corresponde al nombre del esquema de la tabla a la cual pertenece la columna.
- *getTableName(int columna):* Retorna un String correspondiente al nombre de la tabla a la que pertenece la columna.
- *isAutoIncrement(int columna):* Retorna un valor lógico (boolean) que indica si la columna es de tipo autoincremento.
- *isCaseSensitive(int columna):* Retorna un valor lógico (boolean) que indica si la columna establece diferencias entre mayúsculas y minúsculas.

- *isCurrency(int columna):* Retorna un valor lógico (boolean) que indica si la columna fue establecida para almacenar valores monetarios.
- *isDefinitivelyWritable(int columna):* Retorna un valor lógico (boolean) que indica si es absolutamente seguro que la columna se puede modificar.
- *isNullable(int columna):* Retorna un entero (int) que indica si la columna puede contener un NULL SQL. Puede devolver los valores columnNoNulls, columnNullable o columnNullableUnknown, miembros finales estáticos de ResultSetMetaData (constantes).
- *isReadOnly(int columna):* Retorna un valor lógico (boolean) que indica si la columna es definitivamente no editable (solo lectura).
- *isSearchable(int columna):* Retorna un valor lógico (boolean) que indica si es posible utilizar la columna para determinar los criterios de búsqueda de un SELECT.
- *isSigned(int columna):* Retorna un valor lógico (boolean) que indica si la columna fue establecida para almacenar valores numéricos con signo.
- *isWritable(int columna):* Retorna un valor lógico (boolean) que indica si la columna puede modificarse, aunque no lo garantiza.

1.4.3. ¿ Cómo obtener información sobre las características de la Base de Datos ?

Como se indicó anteriormente, a través de la interfaz *DatabaseMetaData*, que proporciona JDBC, es posible interrogar sobre las características de la base de datos con la que se está trabajando. Es posible obtener un objeto de tipo *DatabaseMetaData* mediante el método *getMetaData()* de *Connection*.

DatabaseMetaData proporciona diversa información sobre una base de datos, y cuenta con varias docenas de métodos, a través de los cuales es posible obtener gran cantidad de información acerca de una tabla. Entre los métodos tenemos:

- *getCatalogs( ):*Retorna un objeto de tipo *ResulSet*, que contiene un campo de tipo String en cada fila del objeto. Este campo representa el nombre del catalogo disponible en la base de datos. En general el método retorna los nombres de los catálogos disponibles en la base de datos.
- *getColumns(String catalogo, String esquema, String Tabla, String Columna):* Retorna un objeto de tipo *ResulSet* con la descripción de las columnas de la tabla especificada en el catalogo. Cada fila de este objeto esta formada por 18 campos descriptivos, tales como: el nombre de la

columna, el nombre de la tabla que la contiene, el tamaño máximo de la columna, etc. Los datos pasados por parámetro son los siguientes: un nombre del catalogo, un nombre de esquema, un nombre de tabla y un nombre de columna, a fin de establecer un patrón de búsqueda.

El patrón se especifica de la misma forma que en las sentencias SQL, en las cuales se usa *LIKE* para las coincidencias. En particular, el guión bajo (_) en una expresión de búsqueda representa cualquier carácter de la cadena encontrada, y el tanto por ciento (%) representa cualquier número de caracteres consecutivos de la cadena de búsqueda. Por ejemplo, en el patrón de búsqueda ca_a se encontrarían las palabras "casa", "cara", "cala", etc., mientras que la expresión ca%a encontraría cualquier palabra que comenzase por "ca" y terminase en "a", independientemente del número de caracteres que la formasen. Por ejemplo:

```
getColumns (null,null,"ejemplo","%");
```

Los valores de los parámetros correspondientes al catálogo y al esquema son nulos, para indicar que este parámetro debe obtenerse a partir del criterio de búsqueda. Finalmente está el patrón de coincidencia "%" para los nombres de las columnas, que corresponde a: "Dame TODAS las columnas de la tabla ejemplo".

- *getExportedKeys(String catalogo, String esquema, String Tabla):* Retorna mediante un objeto de tipo *ResulSet*, la descripción de cada una de las columnas que son claves foráneas o ajenas y que referencian las columnas que conforman la clave primaria de la tabla. Cada fila de este objeto esta formada por 14 campos descriptivos, tales como: el nombre del catalogo, el nombre del esquema, el nombre de la tabla, el nombre de la columna, etc.

- *getImportedKeys(String catalogo, String esquema, String Tabla):* Retorna mediante un objeto de tipo *ResulSet*, la descripción de cada una de las columnas que son claves primarias en la tabla y claves foráneas o ajenas de otra. Cada fila de este objeto esta formada por 14 campos descriptivos, tales como: el nombre del catalogo, el nombre del esquema, el nombre de la tabla, el nombre de la columna, etc.

- *getPrimaryKeys(String catalogo, String esquema, String Tabla):* Retorna mediante un objeto de tipo *ResulSet*, la descripción de cada una de las columnas que conforman la clave primaria de la tabla especificada. Cada fila de este objeto esta formada por 6 campos descriptivos, tales como: el nombre del catalogo, el nombre del esquema, el nombre de la tabla, etc.

- *getProcedures(String catalogo, String esquema, String Procedimiento):* Retorna mediante un objeto de tipo *ResulSet*, la descripción de cada uno de los procedimientos disponibles en el catalogo. Cada fila de este objeto esta formada por 8 campos descriptivos.

- *getTables(String catalogo, String esquema, String Tabla, String[ ] tipos):* Retorna mediante un objeto de tipo *ResulSet*, la descripción de cada una de las tablas disponibles en el catalogo.

Cada fila de este objeto esta formada por 10 campos descriptivos, tales como: el nombre del catalogo, el nombre del esquema, el nombre de la tabla, tipo de Tabla: "TABLE", "VIEW", "SYSTEM TABLE", "GLOBAL TEMPORARY", "ALIAS", "SYNONYM". etc.

- *getUserName( ):* Retorna un String que contiene el nombre del usuario actual.
- *getURL( ):* Retorna un String que contiene el URL de la base de datos actual.
- *supportsGroupBy( ):* Retorna un valor lógico (boolean) que indica si la base de datos soporta el uso de GROUP BY en un SELECT.

1.4.4. Tipos SQL en Java

Es necesario saber cómo recuperar adecuadamente tipos de datos Java; como int, long, o String, a partir de sus contrapartidas SQL almacenadas en base de datos. Esto puede ser especialmente importante si trabajamos con datos numéricos, que necesiten control decimal con precisión, o con fechas SQL, que tienen un formato muy bien definido.

JAVA	SQL
String	VARCHAR
boolean	BIT
byte	TINYINT
short	SMALLINT
int	INTEGER
long	BIGINT
float	REAL
double	DOUBLE
byte[ ] - byte array: imagenes, sonidos...	VARBINARY (BLOBs)
java.sql.Date	DATE
java.sql.Time	TIME
java.sql.Timestamp	TIMESTAMP
java.math.BigDecimal	NUMERIC

Tabla 1. Conversión de Tipos. De JAVA a SQL.

El tipo de dato *byte[ ]*, es un arreglo de *bytes* de tamaño variable. Esta estructura de datos guarda datos binarios, que en SQL son *VARBINARY* y *LONG-VARBINARY*. Estos tipos se utilizan para almacenar imágenes, ficheros de documentos, y cosas parecidas. Para almacenar y recuperar este tipo de información

de la base de datos, se deben utilizar los métodos para *streams* que proporciona JDBC: *setBinaryStream()* y *getBinaryStream().*

La conversión de tipos en el sentido contrario puede no ser directa, ya que hay tipos SQL cuyo tipo Java correspondiente puede no ser evidente, como VARBINARY, o DECIMAL, etc. La tabla siguiente muestra los tipos Java correspondientes a cada tipo SQL.

SQL	JAVA
CHAR	String
VARCHAR	String
LONGVARCHAR	String
NUMERIC	java.math.BigDecimal
DECIMAL	java.math.BigDecimal
BIT	boolean
TINYINT	byte
SMALLINT	short
INTEGER	int
BIGINT	long
REAL	float
FLOAT	double
DOUBLE	double
BINARY	byte[ ]
VARBINARY	byte[ ]
LONGVARBINARY	byte[ ]
DATE	java.sql.Date
TIME	java.sql.Time
TIMESTAMP	java.sql.Timestamp

Tabla 2. Conversión de Tipos. De SQL a JAVA.

Existe una constante para cada tipo de dato SQL, declarada en *java.sql.Types*; por ejemplo, el tipo TIMESTAMP le corresponde la constante *java.sql.Types.TIMESTAMP*.

Además, JDBC proporciona clases Java nuevas para representar varios tipos de datos SQL: estas son *java.sql.Date*, *java.sql.Time* y *java.sql.Timestamp*.

1.5. Resumen del Capitulo

En este capitulo se han descrito algunos conceptos fundamentales de base de datos. En primer lugar, se definieron los conceptos involucrados en el modelo Entidad – Relación, el cual representa el modelo lógico basado en objetos más utilizado. En este punto se definieron conceptos tales como: entidades, atributos y relaciones.

Seguidamente se definieron los conceptos fundamentales del modelo relacional, el cual representa uno de los modelos mas extendidos en el modelo lógico basado en registros. En este modelo se definieron los siguientes conceptos: tabla, campos, tipos de campos, dominio, atributo, tupla, clave primaria, clave foránea, clave compuesta, esquema, restricciones y borrado en cascada.

Luego se definió metadato, como la información acerca de la estructura de la base de datos y en donde cada Sistema Manejador de Base de Datos (SMBD) posee diversos mecanismos que posibilitan su acceso.

Finalmente se describió el API JDBC, a través del cual un sistema puede comunicarse con el Sistema Manejador de Base de Datos y así ejecutar operaciones sobre una base de datos. En este punto, se detallaron las instrucciones a ejecutar para realizar la conexión a la base datos, se describieron cada una de las clases que componen el API y se hizo énfasis en las clases *ResultSetMetaData* y *DatabaseMetaData*, a través de las cuales es posible obtener los metadatos de un Sistema Manejador de Base de Datos.

Todos los conceptos descritos en este capítulo son la base para el cumplimiento del objetivo general de este Trabajo Especial de Grado, y serán de gran utilidad para realizar el Marco Aplicativo. En el próximo capítulo, se estudian algunas herramientas existentes que permiten crear aplicaciones administrativas que gestionan la información contenida en una base de datos.

CAPÍTULO II:

HERRAMIENTAS EXISTENTES

En el presente capítulo, se estudian tres (3) sistemas existentes: *CodeCharge*, *PHP MySQL Wizard* y *DB2ASP Creator*, a fin de tomar las consideraciones pertinentes en el desarrollo de la aplicación DEUSWEB, a través de las funcionalidades y características propias de cada una de estas herramientas. Cada uno de estos sistemas, permite crear aplicaciones web que gestionan la información contenida en una base de datos. Para la evaluación de estas aplicaciones, se utilizó la misma base datos, cuya estructura se describe en el caso de estudio. Así mismo se definieron los criterios de evaluación a seguir en cada uno de estos sistemas. El caso de estudio esta enmarcado en el contexto de usuarios que compran en una tienda.

2.1. Caso de Estudio para la evaluación

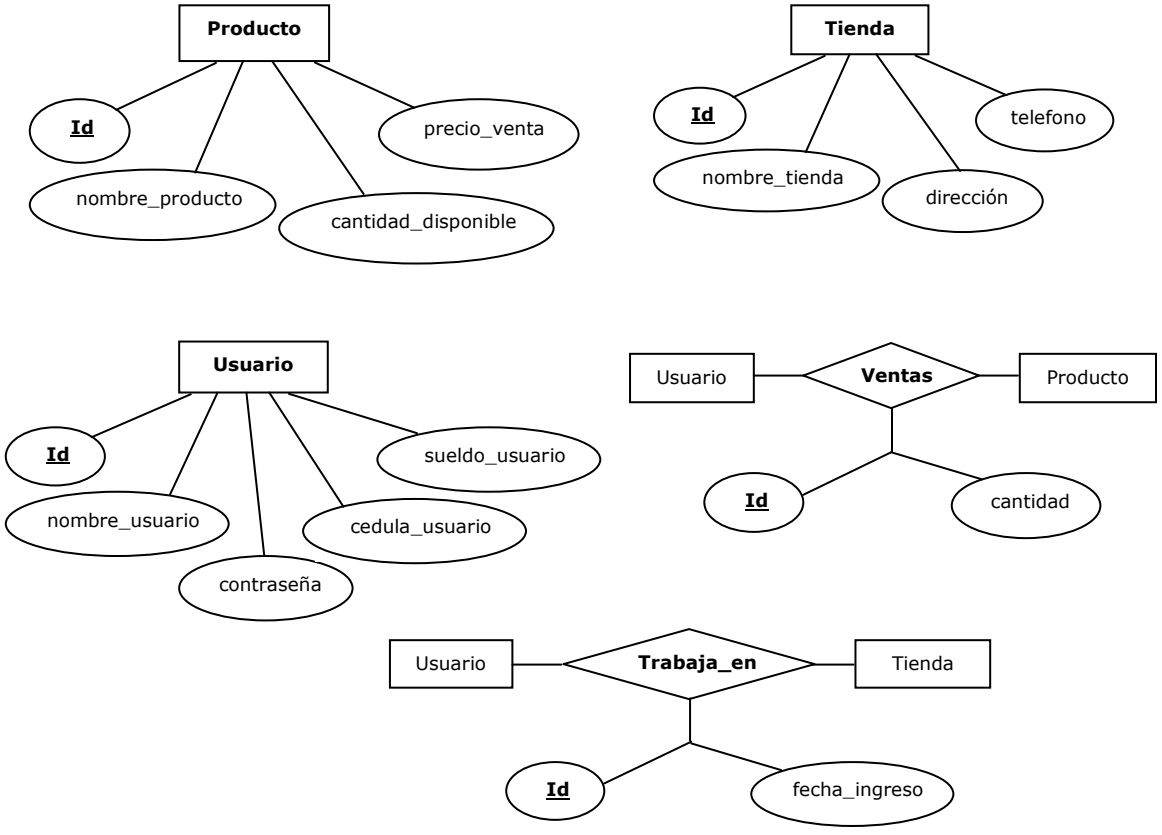


Figura 14. Diagrama E-R. Caso de Estudio, para la evaluación.

2.2. Criterios de Evaluación

2.2.1. Usabilidad: Se refiere a la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso. Principios básicos:

- *Facilidad de Aprendizaje*: se refiere a la facilidad con la que nuevos usuarios pueden tener una interacción efectiva.
- *Flexibilidad*: hace referencia a la variedad de posibilidades con las que el usuario y el sistema pueden intercambiar información.
- *Robustez*: es el nivel de apoyo al usuario que facilita el cumplimiento de sus objetivos.

2.2.2. Legibilidad: Capacidad de poder entender el código generado fácilmente.

2.2.3. Configuración Avanzada: Se refiere a la capacidad que posee la aplicación de poder modificar o alterar el contenido o características de diseño en las páginas generadas.

2.2.4. Aspectos Técnicos:

- ***Lenguajes de Programación que utiliza:*** Se refiere a la capacidad que posee la aplicación para generar código en distintos lenguajes de programación.
- ***Validación de los campos:*** Si la aplicación genera la validación de los campos que componen las páginas generadas.
- ***Paginación:*** Se refiere a la capacidad que posee la aplicación en publicar el contenido de una tabla de forma ordenada, usando enlaces para avanzar y/o retroceder en los registros que se muestran.
- ***Autenticación:*** Si la aplicación permite generar una página que sirva para autenticar al usuario antes de acceder a ciertas páginas también generadas.
- ***Diseño:*** Si la aplicación provee la capacidad de poder escoger el diseño de las páginas generadas.
- ***Publicación:*** Se refiere a la capacidad que tiene la aplicación para subir los archivos generados en el servidor.

Los criterios de evaluación antes descritos tendrán una ponderación en un rango de 0 a 5, siendo 0 la puntuación más baja y 5 la puntuación más alta.

2.3. Aplicación: CodeCharge

Codecharge, una aplicación producida por la empresa YesSoftware, es una herramienta de desarrollo rápido de aplicaciones de Internet, que permite crear aplicaciones para múltiples plataformas adaptadas a bases de datos. Es una aplicación bastante completa que utiliza un asistente para generar los archivos y adicionalmente proporciona las herramientas necesarias para modificar los archivos generados.

A continuación, se explican algunas pantallas que utiliza esta aplicación para generar el sistema web administrativo y posteriormente se muestran algunas pantallas del sitio web generado:

2.3.1. Pantallas para la generación del sitio

Pantalla de propiedades del sitio: En la figura 15, se especifican algunas propiedades generales del sitio a generar. Dentro de estas propiedades se encuentran:

- **Tipo de lenguaje de programación**: Los lenguajes de programación que pueden ser utilizados en los archivos a generar son los siguientes:
    - ASP 3.0 with Templates: genera código ASP 3.0 y separa los templates .html.
    - ASP.Net 1.0 C#: genera archivos .aspx con código C#.
    - ASP.Net 1.0 VB: genera archivos .aspx con código VB.Net.
    - CFML 4.0.1: genera código ColdFusion 4.0.1.
    - CFML 4.0.1 with Templates: genera código ColdFusion 4.0.1 (.cfm) y separa los archivos templates .html.
    - JSP 1.1 JDK 1.3: Genera código JSP 1.1.
    - PHP 4.0 with Templates: genera código PHP 4.0 (.php) y separa los archivos templates .html.
    - Servlets 2.2 JDK 1.3 with Templates: genera código Java que utiliza templates .html
- **Lenguaje del sitio:** Ingles o español.
- **Ruta del Servidor:** Se coloca la ruta completa, donde los archivos generados serán ubicados para ser publicados localmente.
- **URL del Servidor:** Se coloca la dirección web que corresponde a la ruta del servidor. Esta dirección será usada para ver las páginas en modo Live Page.

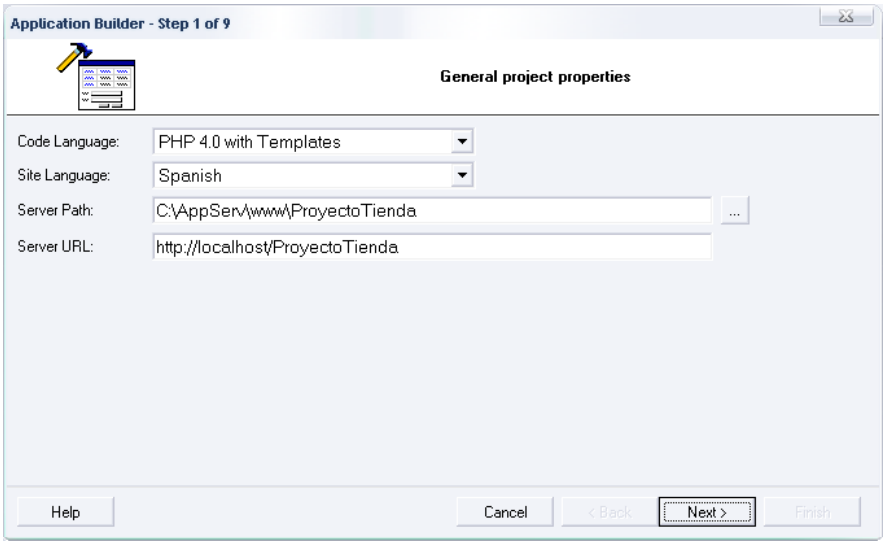


Figura 15. Algunas propiedades generales del sitio a generar. Aplicación *Codecharge*.

Pantalla de opciones de configuración: En otra pantalla de esta aplicación, el usuario establece las opciones de configuración de la aplicación a generar. Estas opciones son las siguientes:

- **Enable smart naming:** Al seleccionar esta opción se convertirán automáticamente los nombres de las tablas en frases dentro del programa generado. Por Ejemplo: la tabla Productos será mostrado en una etiqueta con el título Lista de Productos.
- **Autoincremented Primary Keys:** Al seleccionar esta opción se especifica que las tablas en la base de datos contienen campos claves que son auto incrementadas.
- **Use single keyword field for text and memo fields on search forms:** Al seleccionar esta opción se genera un campo de búsqueda simple que busca contra todos los campos de prueba y memo en las tablas. Si esta opción no esta seleccionada, se crea una sección de búsqueda con múltiples campos de búsqueda.
- **Publish the site after Application Builder completes creating pages:** Al seleccionar esta opción se especifica que se quiere publicar el sitio tan pronto como se crean todas las páginas necesarias.
- **Use Popup Date Picker for date input field:** Al seleccionar esta opción, los campos que sean detectados como campos de fecha serán creados con su correspondiente componente de selección de fecha. Este componente le permite al usuario especificar fácilmente los valores de la fecha.

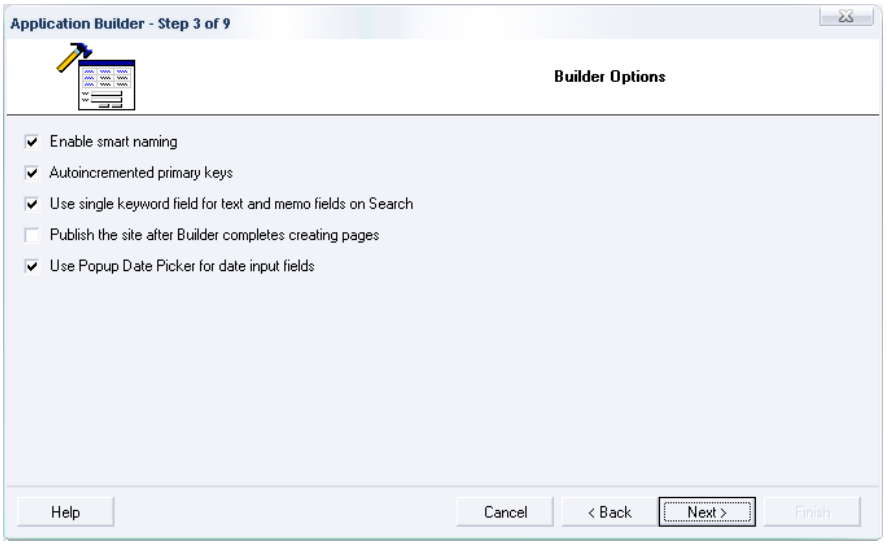


Figura 16. Opciones de configuración del sitio a generar. Aplicación *Codecharge*.

Pantalla de autenticación para el sitio: En este paso se especifica si se quiere usar autenticación y privilegios de usuarios antes de permitir el acceso a ciertas páginas. Si se selecciona *Use authentication*, se creará la pagina de login y se permitirá especificar el nivel de seguridad para cada una de las paginas creadas.

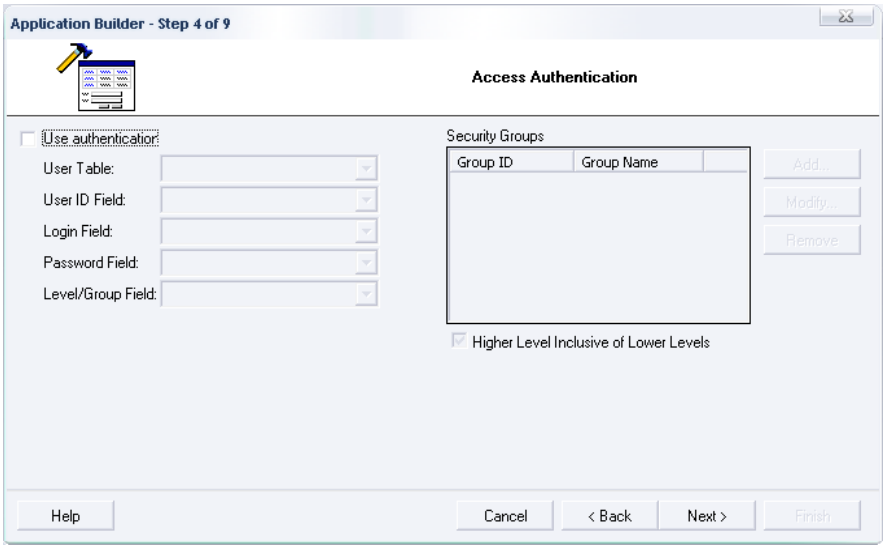


Figura 17. Autenticación del sitio a generar. Aplicación *Codecharge*.

2.3.2. Pantallas del sitio web generado

Vista para buscar o listar un Producto: En ella se aprecia un campo de texto para buscar una o varias filas que posean en alguno de sus campos la palabra clave introducida. Igualmente se aprecia una lista con las filas de la tabla Producto. Este tipo de páginas maneja la paginación para el contenido de la tabla seleccionada, de esta forma no se listan todas las filas de la tabla a la vez, sino que por el contrario se listan las filas de la tabla de 10 en 10.

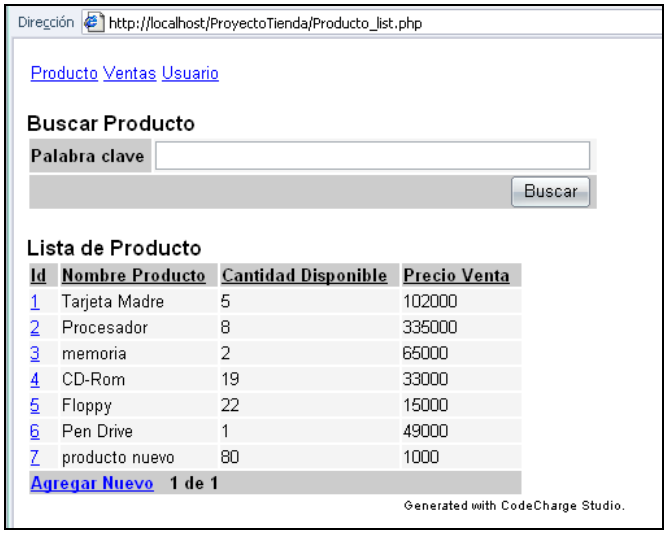


Figura 18. Vista en el Navegador de la página Listar. Aplicación *Codecharge*.

Vista para editar un Producto: En ella se aprecian los campos de textos para cada columna de la base de datos. Es de notar que en la aplicación no se muestran el campo *Id* de producto porque este es un campo de tipo *AUTO INCREMENT.*

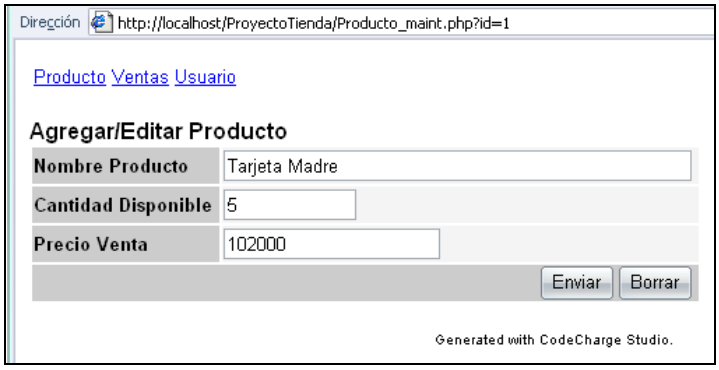


Figura 19. Vista en el Navegador de la página Editar. Aplicación *Codecharge*.

2.3.3. Evaluación de la aplicación

Usabilidad: El software no es fácil de usar, ya que tiene muchas funcionalidades y por esta razón gran cantidad de opciones lo que no permite que el uso específico de generar un sitio web sea comprensible. *(Puntuación: 1 pto).*

Legibilidad del código que produce: Es bastante legible. Por cada tabla se generan dos páginas, una pagina HTML que contiene todo el diseño visual y otra página (.php, .jsp, .asp, etc), que contiene todo el código de la aplicación (encapsulado en clases y métodos) para interactuar con la base de datos. *(Puntuación: 4 ptos).*

Configuración Avanzada: Esta aplicación posee las herramientas necesarias para modificar las páginas generadas fácilmente. Permite crear nuevas tablas html, eliminar filas o columnas, agregar objetos al formulario, etc. *(Puntuación: 5 ptos)*

Usabilidad de la aplicación web producida: Fácil de usar. Las páginas producidas no tienen mayor complicación de uso. *(Puntuación: 5 ptos)*

Aspectos Técnicos:

Lenguajes de Programación que utiliza: Puede generar código en 8 lenguajes de programación distintos *(Puntuación: 5 ptos)*

Validación de los campos: Sí se generan validaciones de los campos en las páginas generadas. *(Puntuación: 5 ptos)*

Paginación: En las páginas generadas que listan las filas de una tabla, se muestran las filas de 10 en 10. *(Puntuación: 4 ptos)*

Autenticación: Si es posible generar una página que realice la autenticación, a fin de restringir el acceso a las páginas que se indiquen. *(Puntuación: 5 ptos)*

Diseño: Se puede escoger el diseño de la página dentro de una gama de diseños preestablecidos, también se permite agregar nuevos diseños. *(Puntuación: 4 ptos)*

Publicación: La aplicación permite configurar el lugar de destino en el servidor local donde se ubicarán las páginas generadas. *(Puntuación: 5 ptos)*

2.4. Aplicación: PHP MySQL Wizard

Esta aplicación producida por SoftGalaxy, es un asistente fácil de usar, utilizado para crear un conjunto completo de páginas web que administran una base de datos de MySQL.

A continuación, se explican algunas pantallas que utiliza esta aplicación para generar el sistema web administrativo y posteriormente se muestran algunas pantallas del sitio web generado:

2.4.1. Pantallas para la generación del sitio

Pantalla de conexión a la base de datos: En la siguiente pantalla el usuario coloca la información requerida para conectarse a una base de datos MySQL. Una vez introducida la información, el usuario presiona *Connect* para que el sistema se conecte al Sistema Manejador de Base de Datos y extraiga el nombre de cada una de las bases de datos a las que se puede acceder con los parámetros especificados. Posteriormente el sistema carga en la lista *DB Select*, el nombre de todas las bases de datos a las que se puede acceder.

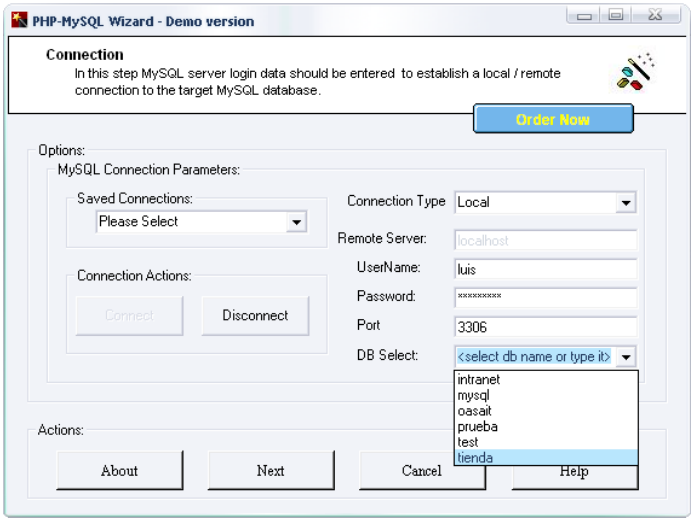


Figura 20. Conexión a la base de datos. Aplicación PHP MySQL Wizard.

Pantalla de selección de tablas: En la siguiente pantalla el sistema muestra dos listas de selección múltiple, la de la izquierda contiene el nombre de todas las tablas contenidas en la base de datos seleccionada y la de la derecha contiene el nombre de las tablas que el usuario ha seleccionado para administrar por el sistema a generar.

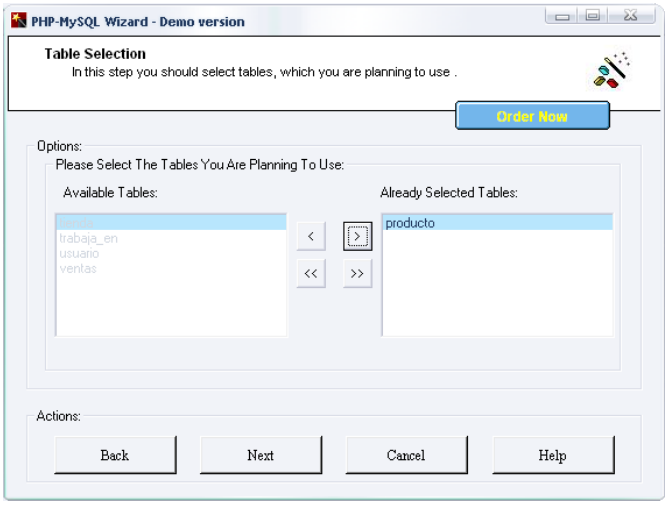


Figura 21. Selección de las tablas de la base de datos. Aplicación PHP MySQL Wizard.

Pantalla de selección de la apariencia para el sitio a generar: La aplicación permite escoger entre tres (3) diseños establecidos por defecto, así como también permite generar diseños nuevos a través de la misma aplicación.

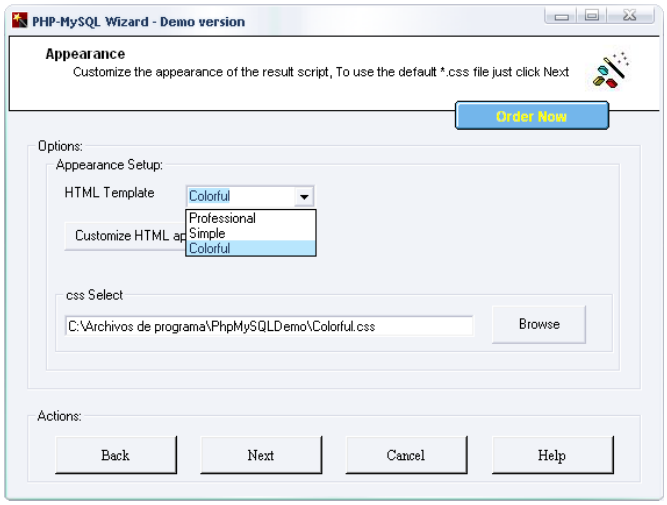


Figura 22. Selección de la apariencia para el sitio a generar. Aplicación PHP MySQL Wizard.

2.4.2. Pantallas del sitio web generado

Página de autenticación: En ella se aprecian dos campos, donde el usuario introduce su nombre de usuario y contraseña para entrar al sitio. El sistema verifica el acceso al sitio con el nombre de usuario y contraseña que tenga acceso a la base de datos.

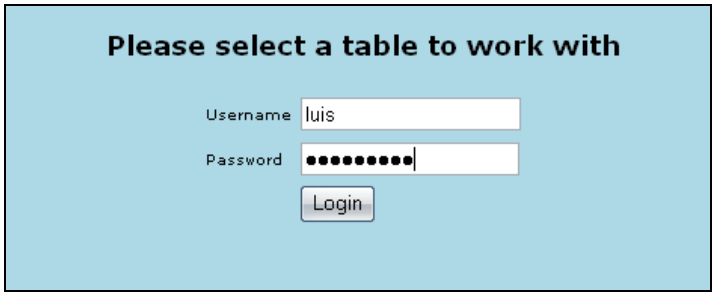


Figura 23. Página de autenticación del sitio generado Aplicación PHP MySQL Wizard.

Vista en el Navegador de la página que permite buscar, listar y agregar información a una tabla en la base de datos: En esta página se aprecian tres partes: La primera, en donde se puede buscar a través de una palabra clave todas aquellas filas de la tabla que contengan dicha palabra; en la segunda se listan todas las filas de la tabla usando paginación; y en la tercera, se puede agregar una nueva fila a la tabla.

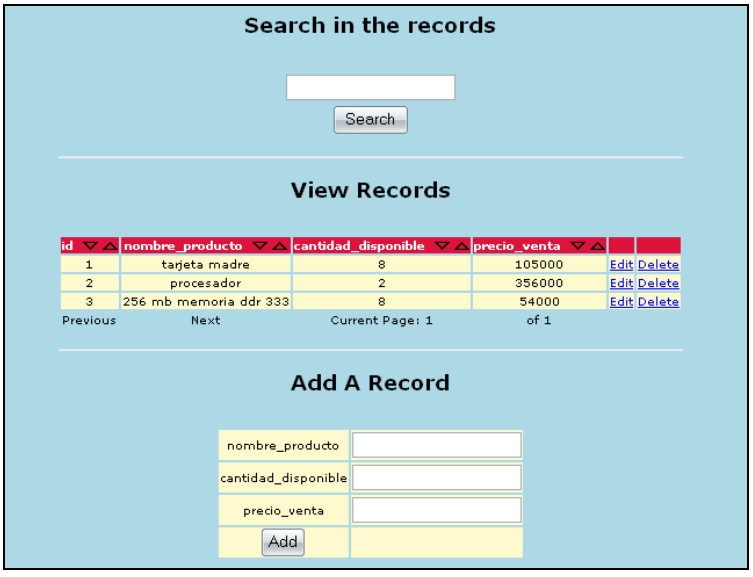


Figura 24. Página principal del sitio generado Aplicación PHP MySQL Wizard.

2.4.3. Evaluación de la aplicación

Usabilidad: El software fácil de usar, las pantallas son explicitas. *(Puntuación: 4 ptos)*

Legibilidad del código que produce: Es bastante legible. Se generan páginas que contiene instrucciones de includes, pero no se genera la aplicación en capas. *(Puntuación: 3 ptos)*

Configuración Avanzada: Esta aplicación no permite modificar las páginas generadas. *(Puntuación: 0 ptos)*

Usabilidad de la aplicación web producida: Fácil de usar. Las páginas producidas no tienen mayor complicación de uso. La aplicación web generada, realiza las siguientes funcionalidades: Buscar, Listar, Agregar, Modificar, Eliminar. *(Puntuación: 5 ptos)*

Aspecto Técnicos:

Lenguajes de Programación que utiliza: Solo genera código en PHP. *(Puntuación: 1 ptos).*

Validación de los campos: La validación de los campos es generada en PHP en el momento de insertar la tupla en la base de datos. No se generan validaciones *javascript* u otro lenguaje similar. *(Puntuación: 2 ptos).*

Paginación: En las páginas generadas que listan las filas de las base de datos, se encuentra contemplada la paginación. Adicionalmente el asistente permite configurar la paginación. *(Puntuación: 5 ptos).*

Autenticación: Se puede generar una página que realice la autenticación, a fin de restringir el acceso a las páginas que se indiquen. *(Puntuación: 5 ptos).*

Diseño: Se permite escoger el diseño de la página dentro de una gama de diseños preestablecidos, e inclusive permite agregar nuevos diseños, configurables desde la propia aplicación. *(Puntuación: 5 ptos).*

Publicación: La aplicación permite configurar el lugar de destino en el servidor local donde se ubicarán las páginas generadas. *(Puntuación: 5 ptos).*

2.5. Aplicación: DB2ASP Creator

Esta aplicación producida por Enterprise Dreams Solutions, es una aplicación web, que genera las páginas de administración de un sitio web utilizando ASP, a partir de una base de datos Access.

A continuación, se explican algunas pantallas que utiliza esta aplicación para generar el sistema web administrativo y posteriormente se muestran algunas pantallas del sitio web generado:

2.5.1. Pantallas para la generación del sitio

Pantalla de conexión a la base de datos: En esta pantalla se coloca la dirección física de la base de datos Access. Seguidamente, Se coloca la contraseña de la base de datos, en caso de que esta posea, y posteriormente se coloca una contraseña de Administración de Proyecto para autenticar al usuario antes de que acceda a las páginas generadas. Luego se coloca el nombre del campo en las tablas y finalmente el número de registros para hacer la paginación.

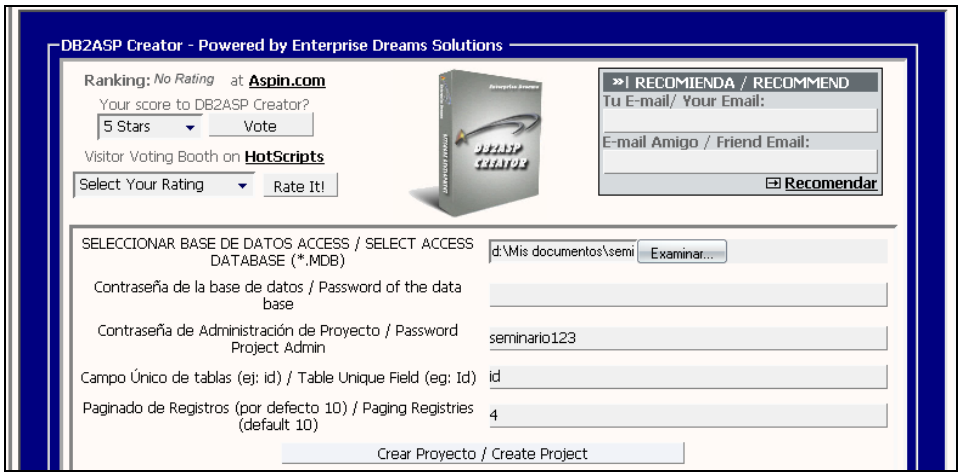


Figura 25. Pantalla de conexión a la base de datos. Aplicación DB2ASP Creator

Pantalla para la visualización del sitio generado: Al generar las páginas, la aplicación informa sobre el éxito de la operación mediante la siguiente pantalla. En esta página, también existe un enlace a la aplicación generada.

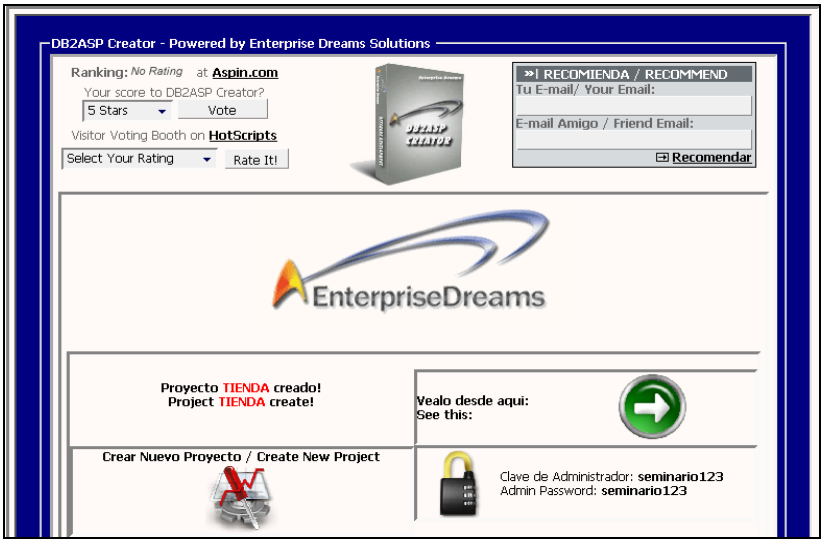


Figura 26. Pantalla para la visualización del sitio generado. Aplicación DB2ASP Creator

2.5.2. Pantallas del sitio web generado

Página principal del sitio: En esta página se observa un área de menú, en donde se permite el acceso a las páginas que listan y agregan las filas, en cada una de las tablas de la base de datos. Así mismo, en el área central, se describe como acceder a cada una de las funcionalidades que posee el sitio.

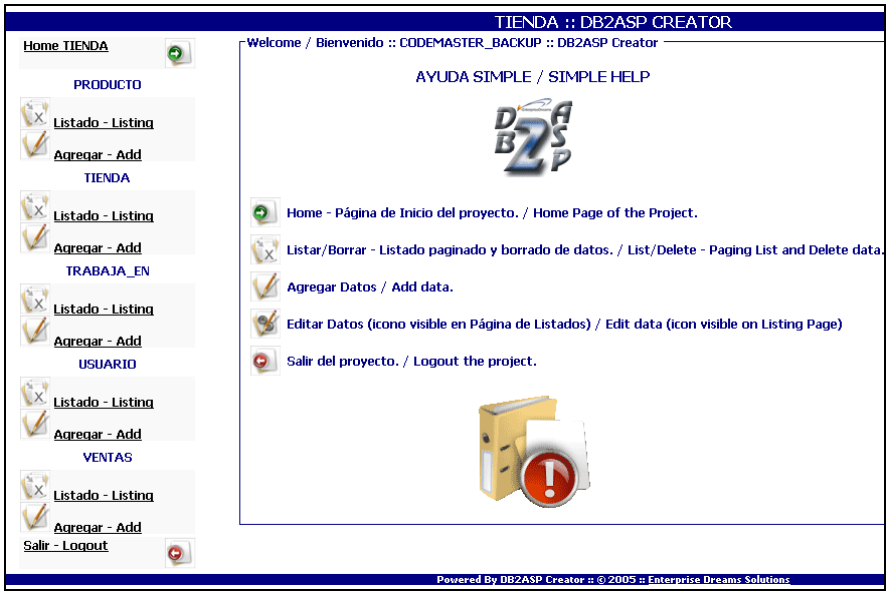


Figura 27. Página principal del sitio generado. Aplicación DB2ASP Creator

Página que lista el contenido de una tabla: En esta página se observa el área de menú descrito anteriormente y un área central en donde se listan las filas de la tabla seleccionada. A través de esta página, también se permite eliminar de la base de datos, las filas seleccionadas.

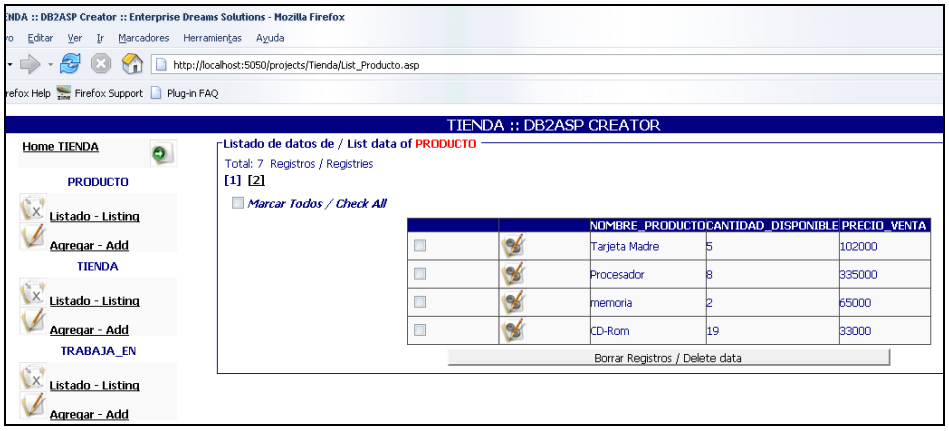


Figura 28. Página listar del sitio generado. Aplicación DB2ASP Creator.

2.5.3. Evaluación de la aplicación

Usabilidad: La aplicación web que produce el código es muy fácil de usar. La interacción del usuario con la aplicación es mínima. *(Puntuación: 5 ptos)*

Legibilidad del código que produce: Es bastante legible. Se generan páginas que contienen instrucciones de includes, pero no se genera la aplicación en capas. *(Puntuación: 3 ptos)*

Configuración Avanzada: Esta aplicación web, no permite modificar las páginas generadas *(Puntuación: 0 ptos)*

Usabilidad de la aplicación web producida: Fácil de usar. Las páginas producidas no tienen mayor complicación de uso. La aplicación web generada, realiza las siguientes funcionalidades: Listar, Agregar, Modificar, Eliminar. *(Puntuación: 4 ptos)*

Aspectos Técnicos:

Lenguajes de Programación que utiliza: Solo genera código en ASP. *(Puntuación: 1 ptos).*

Validación de los campos: Si se generan validaciones de los campos en las páginas generadas. *(Puntuación: 5 ptos).*

Paginación: En las páginas generadas que listan las tuplas de las base de datos, se encuentra contemplada la paginación. Adicionalmente se permite configurar la paginación. *(Puntuación: 5 ptos).*

Autenticación: A través de la aplicación web que genera el sitio web, se genera una página que realiza la autenticación, a fin de restringir el acceso a las páginas que se indiquen. *(Puntuación: 5 ptos).*

Diseño: La aplicación no permite escoger el diseño de la página. *(Puntuación: 0 ptos).*

Publicación: La aplicación no permite configurar el lugar de destino en el servidor local donde se ubicarán las páginas generadas, sin embargo para su ejecución la aplicación web que la genera, provee un enlace. *(Puntuación: 2 ptos).*

2.6. Tabla Comparativa

A continuación se muestra una tabla comparativa con base en el puntaje obtenido en la evaluación realizada a cada una de las tres aplicaciones anteriores:

	CODECHARGE	PHP MYSQL WIZARD	DB2ASP CREATOR
USABILIDAD	1	4	**5**
LEGIBILIDAD DEL CÓDIGO QUE PRODUCE	**4**	3	3
CONFIGURACIÓN AVANZADA	**5**	0	0
USABILIDAD DE LA APLICACIÓN WEB PRODUCIDA	**5**	**5**	4
LENGUAJES DE PROGRAMACIÓN QUE UTILIZA	**5**	1	1
VALIDACIÓN DE LOS CAMPOS	**5**	2	**5**
PAGINACIÓN	4	**5**	**5**
AUTENTICACIÓN	**5**	**5**	**5**
DISEÑO	4	**5**	0
PUBLICACIÓN	**5**	**5**	2
TOTAL	**43**	**35**	**30**

Tabla 3. Comparación entre las aplicaciones evaluadas.

Se puede decir que la primera aplicación: *CodeCharge*, es la mejor aplicación estudiada en este capítulo, sin embargo hay aspectos en los cuales se puede mejorar, tales como: usabilidad, paginación y diseño. Es de importancia hacer notar que la aplicación *codecharge* no solo es una aplicación que genera sitios web administrativos, sino que es una aplicación mas completa que permite modificar los archivos generados en la misma aplicación.

2.7. Resumen del Capitulo

En este capítulo se estudiaron algunas herramientas existentes en el mercado, que permiten crear aplicaciones web administrativas a partir de la ingeniería de reverso al modelo de datos. En primer lugar se describió el caso de estudio con el que se evaluó a cada una de las aplicaciones. Seguidamente se describieron los aspectos que fueron tomados en cuenta para la evaluación y finalmente se estudiaron las aplicaciones. En el estudio de cada aplicación se mostraron algunas pantallas de la misma y de la aplicación generada por cada una de estas. Finalmente se describe en una tabla comparativa cada uno de los aspectos tomados en cuenta en el criterio de evaluación, junto con el puntaje asignado a ese criterio por cada aplicación evaluada.

Se escogió a la aplicación *CodeCharge*, como la mejor aplicación evaluada en este capítulo. Sin embargo el alcance de esta aplicación va más allá de generar un sitio web que gestione el contenido de las tablas de una base de datos, ya que a través de esta aplicación es posible modificar los archivos generados, añadiendo de esta forma una mayor complejidad a la aplicación. A pesar de que *CodeCharge*, traspasa el alcance de este Trabajo Especial de Grado, existen aspectos que se pueden mejorar en esta aplicación con respecto al alcance de este trabajo. Entre ellos esta la usabilidad de la aplicación generadora y del sitio generado, así como la configuración de la paginación del sitio generado.

Del estudio de estas herramientas existentes se obtuvo la base de ¿cómo debía ser un sistema generador de sitios web basado en la ingeniería de reverso al modelo de datos?; y también de ¿cómo debía comportarse el sitio generado?. De la aplicación generadora se puede decir que debe ser fácil de usar, con pocos pasos para que el usuario genere el sitio y de fácil configuración para el diseño y paginación. Así mismo la aplicación generada debe cumplir con las funcionalidades básicas que son: autenticación del usuario, listar, buscar, insertar, modificar y eliminar cualquier información en la base de datos, a la vez que debe ser fácil de usar y con poca profundidad en la navegación.

PARTE 

Marco Aplicativo

CAPÍTULO III:

PROCESO DE DESARROLLO A UTILIZAR

El proceso de desarrollo de software utilizado, fue el Proceso Unificado[4], el cual constituye un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas, en diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyectos. La razón por la cual se escogió al proceso unificado como el proceso de desarrollo, fue por su capacidad de desarrollo en proyectos de tamaño variable y su rápida implementación.

El proceso unificado se puede resumir en tres frases: dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental y utiliza el Lenguaje Unificado de Modelaje *(Unified Modeling Languaje UML)* para preparar todos los artefactos de un sistema de software. El proceso unificado se repite a lo largo de una serie de iteraciones, en donde cada una de ellas constituye una versión del producto.

Este proceso consta de cuatro etapas: inicio o concepción, elaboración, construcción y transición y de cinco flujos de trabajo principales que son: requisitos, análisis, diseño, implementación y pruebas. A continuación se muestra una figura donde se observa la relación entre las etapas, flujos de trabajo e iteraciones.

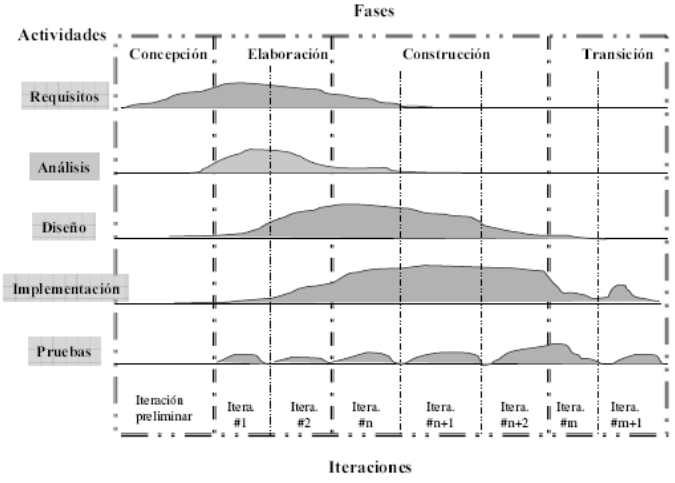


Figura 29. Flujos de Trabajo y Etapas en el Proceso Unificado [5]

[4] Ver Jacobson. "*El Proceso Unificado de Desarrollo de Software*", p.p. 3 -12
[5] Imagen extraida de http://kybele.escet.urjc.es/documentos/IS/Analisis.pdf

A continuación se explican cada uno de los flujos de trabajo que componen el proceso unificado, así como también las etapas del mismo.

3.1. Flujos de Trabajo

3.1.1. Requisitos

La captura de requisitos es el proceso de averiguar lo que se debe construir. El propósito fundamental de este flujo de trabajo es guiar el desarrollo hacia el sistema correcto, que debe y que no debe hacer el sistema, utilizando el lenguaje del cliente para describir los resultados.

3.1.2. Análisis

Se analizan los requisitos descritos en la captura de requisitos, refinándolos y estructurándolos. Se describe utilizando el lenguaje de los desarrolladores y es utilizado para razonar sobre los funcionamientos internos del sistema. Las iteraciones iniciales de la elaboración se centran en el análisis.

3.1.3. Diseño

En este flujo se modela el sistema, incluida la arquitectura para que soporte todos los requisitos. El diseño es el centro al final de la etapa de elaboración y al inicio en la etapa de construcción.

3.1.4. Implementación

Consiste en la implementación del sistema en términos de componentes: archivos de código fuente, scripts, ejecutables, etc. La implementación es el centro durante las iteraciones de construcción.

3.1.5. Prueba

En este flujo de trabajo, se verifica el resultado de la implementación, probando cada construcción, incluyendo tanto como construcciones internas como intermedias, así como las versiones finales del sistema a ser entregadas a terceros.

3.2. Adaptación de las Etapas de UP en DEUSWEB

Como se dijo anteriormente en los objetivos de este Trabajo Especial de Grado, se desarrollará una aplicación que permita crear una aplicación web administrativa a partir de la ingeniería de reverso al modelo de datos. Antes del desarrollo de esta aplicación es necesario desarrollar la aplicación web a generar siguiendo como proceso de desarrollo el Proceso Unificado, a fin de conocer que archivos serán necesarios producir por la aplicación generadora.

Por esta razón este proceso de desarrollo: El Proceso Unificado, es ejecutado en ambas aplicaciones, en la aplicación generadora de sitios web administrativos y en la aplicación generada por los motivos antes descritos.

A continuación se describirán las etapas del proceso unificado en conjunto con las etapas ejecutadas en las aplicación desarrollada.

3.2.1. Etapa de Inicio

En el proceso unificado, durante la *etapa de Inicio*, se desarrolla una descripción del producto final a partir de una buena idea y se presenta el análisis de negocio para el producto. Para ambas aplicaciones desarrolladas, en esta etapa se describieron las funcionalidades del sistema en un lenguaje no técnico para su entendimiento. Básicamente durante la etapa de inicio se hizo una iteración centrado en el flujo de trabajo: Requisitos.

3.2.2. Etapa de Elaboración

Durante la *etapa de elaboración*, se especifican en detalle la mayoría de los casos de uso del producto y se diseña la arquitectura del sistema. En esta etapa se hicieron dos iteraciones. La primera, fue centrada en el flujo de trabajo: análisis, y en esta se desarrollaron los casos de uso.

Un caso de uso es un fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante. Todos los casos de uso juntos, constituyen el modelo de casos de uso, el cual describe la funcionalidad total del sistema. En la descripción de los casos de uso, los cuales se detallan en el capítulo IV y capitulo V, se describen los siguientes puntos:

- **Nombre:** Corresponde al nombre del caso de uso.
- **Descripción:** Breve reseña de lo que hace la funcionalidad.
- **Actores:** Usuario de la funcionalidad.
- **Precondiciones:** Describe que debe ocurrir para poder realizar la funcionalidad.

- **Flujo Normal:** Son los pasos que se efectúan cuando el sistema ejecuta la funcionalidad sin contratiempos.
- **Flujo Alternativo:** Son los pasos que se efectúan en la funcionalidad cuando esta no cumple con el resultado esperado.
- **Poscondiciones:** Corresponde con el resultado esperado por la funcionalidad cuando cumple con el flujo normal.

La segunda iteración en esta etapa fue centrada en el flujo de trabajo: Diseño. En esta iteración se desarrollaron los diagrama de clases y secuencia al igual que se modelo la arquitectura del sistema. Todos los diagramas fueron hechos utilizando el Lenguaje Unificado de Modelaje (UML).

3.2.3. Etapa de Construcción

Durante la *etapa de construcción* se crea el producto, y fue en esta etapa cuando se implementó la aplicación generada y la aplicación generadora, utilizando JAVA como lenguaje de programación para la aplicación generadora y utilizando JAVA en conjunto con las tecnologías: JSP y Servlets para la aplicación generada. En esta etapa, básicamente se hizo una iteración centrada en el flujo de implementación.

3.2.4. Etapa de Transición

En la *etapa de transición*, fundamentalmente se prueba el producto construido, como último paso del proceso de desarrollo. Para la aplicación generadora esta etapa la constituye el caso de estudio, donde se realizan todos los pasos para la creación del sitio web generado y su posterior implantación en un servidor web. Para la aplicación generada esta etapa no fue implementada, ya que como se dijo al principio de este punto, la aplicación generada siguió este proceso de desarrollo a fin de conocer los archivos a producir por la aplicación generadora.

3.3. Resumen del Capítulo

El proceso de desarrollo aplicado a los sistemas: generador y generado, fue el proceso unificado, el cual consta de cuatro fases: inicio, elaboración, construcción y transición y de cinco flujos de trabajo principales que son: requisitos, análisis, diseño, implementación y prueba. Este proceso se basa en casos de uso y utiliza como lenguaje de modelación: UML. Se escogió el proceso unificado como proceso de desarrollo por su capacidad para desarrollar proyectos de tamaño variable y su rápida implementación. En los

próximos capítulos se explica en detalle cada uno de los productos generados por los flujos en las etapas del proceso de desarrollo de cada aplicación.

CAPÍTULO IV:

APLICACIÓN A GENERAR

En el presente capitulo se introduce la aplicación web administrativa generada por el sistema, la cual le permite al usuario gestionar de forma eficaz y eficiente las tablas de una base de datos, pudiendo así realizar operaciones tales como: listar, agregar, modificar, eliminar y buscar cualquier información en la tabla de su preferencia.

Tal como se mencionó en el capítulo anterior, antes de poder desarrollar la aplicación generadora es necesario desarrollar la aplicación web a generar, a fin de conocer que archivos serán necesarios producir por la aplicación generadora. Por esta razón en este capítulo se identifican las etapas del proceso de desarrollo, y en cada una de ellas se detallan los artefactos generados por los flujos de trabajo.

4.1. Etapa de Inicio

4.1.1. Flujo de Trabajo: Requisitos (Iteración 1)

Para el desarrollo de las funcionalidades de este sistema, se tomaron en cuenta los resultados obtenidos luego de evaluar tres sistemas existentes, (ver Capítulo II). Dicha evaluación sirvió para determinar las funcionalidades básicas que debe cumplir este sistema, las cuales son: autenticar al usuario, listar, buscar, insertar, modificar y eliminar cualquier información en la base de datos. También este sistema debe ser fácil de usar y con poca profundidad en la navegación. A continuación se describen las funcionalidades:

En primer lugar, la autenticación del usuario en el sistema. En este punto, el sistema verificará que el usuario tenga acceso, antes de poder gestionar cualquier tabla en la base de datos. Así mismo, luego de verificar que tiene acceso, extraerá los privilegios que posee el usuario sobre las tablas. El sistema, para cualquier usuario con acceso válido, permitirá ver un listado de las filas de cada tabla así como la opción para mostrar sólo las filas que posean cierta información en alguno de sus campos.

Otras funcionalidades del sistema son las operaciones básicas que se pueden realizar sobre una tabla en la base de datos. Estas operaciones son: insertar, modificar y eliminar. Para las funcionalidades que así lo requieran, existirá la validación en la entrada de los datos, la identificación de los campos que almacenan archivos, y la identificación de aquellos campos que sean claves foráneas. En la validación de los datos que

introduce el usuario, el sistema deberá comparar los datos introducidos con la definición de los campos en las tablas correspondientes.

Por último se encuentra la opción de cerrar sesión la cual permitirá desvincular al usuario de la aplicación.

Seguidamente se explica en detalle cada funcionalidad del sistema mediante los diagramas y la descripción de los casos de uso.

4.2. Etapa de Elaboración

4.2.1. Flujo de Trabajo: Análisis (Iteración 2)

Diagramas de Casos de Uso

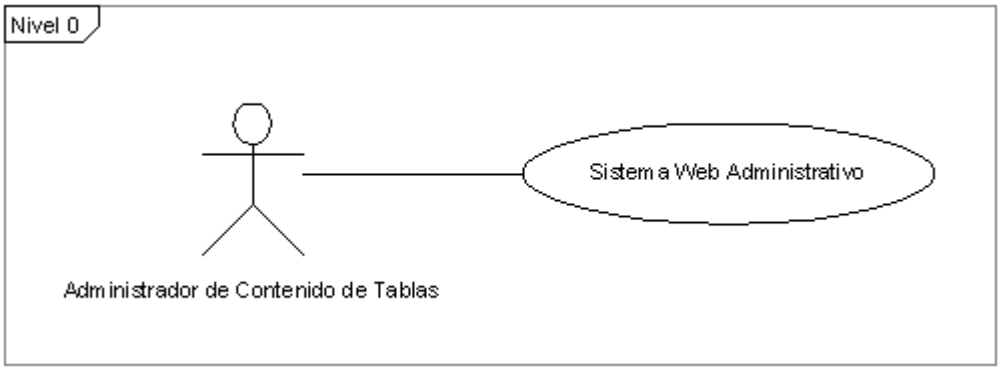


Figura 30: Nivel 0. Diagrama de Casos de Uso. Sitio Web Generado.

Administrador de Contenido de Tablas: Corresponde al usuario del sistema. Es la persona encargada de gestionar las tablas de la base de datos. Debe poseer el conocimiento adecuado para administrar dichas tablas.

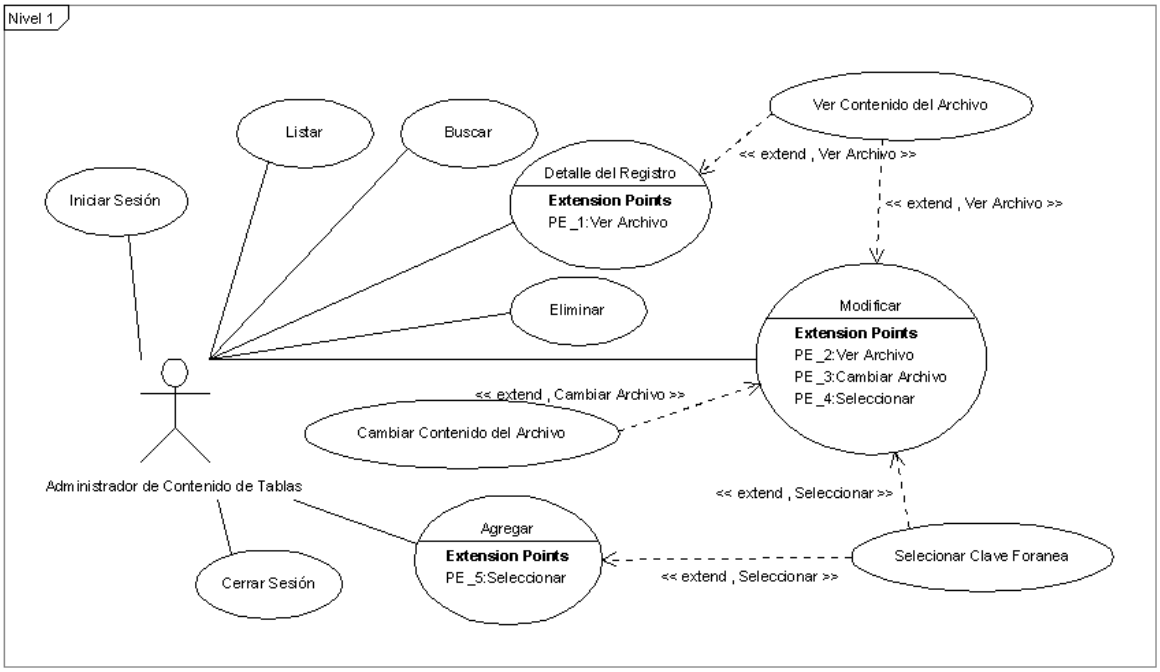


Figura 31: Nivel 1. Diagrama de Casos de Uso. Sitio Web Generado.

Descripción de Casos de Uso

A continuación se describe cada una de las funcionalidades del sistema, explicando cada uno de los casos de uso descritos en el diagrama anterior. En este punto se explica en detalle que debe hacer cada funcionalidad para cumplir con su objetivo particular.

En la descripción de los casos de uso fueron tomados en cuenta los siguientes aspectos: Nombre, Descripción, Actores, Precondiciones, Flujo Normal, Flujo Alternativo, y Poscondiciones. Estos aspectos fueron definidos en el capitulo anterior.

Nombre:	**Iniciar Sesión**
Descripción: Permite al usuario identificarse para entrar en el sistema	
Actores: Usuario del Sitio Web	

Precondiciones: Ninguna
Flujo_Normal: 1. El sistema muestra dos cajas de texto para que el usuario introduzca el nombre y contraseña de Usuario. 2. El actor introduce el nombre y la contraseña. 3. El sistema verifica que el nombre de usuario y contraseña existan en la tabla usuario de la base de datos y extrae los privilegios que posee el usuario sobre el sistema (insertar, modificar, eliminar) y los almacena en variables de sesión.
Flujo_Alternativo: 3. El sistema verifica que el nombre de usuario y contraseña existan en la tabla usuario de la base de datos. Si no existe, se le envía un mensaje de error al usuario.
Poscondiciones: El sistema le otorgó el acceso al actor.

Nombre:	**Listar**
Descripción: Permite mostrar las filas contenidas en la tabla seleccionada. Las filas se muestran usando paginación, con un valor que depende de la selección del usuario en la aplicación generadora. La paginación dependerá del número seleccionado por el usuario en la aplicación generadora y de la cantidad de filas que existan en la tabla.	
Actores: Usuario del Sitio Web	
Precondiciones: Haber seleccionado una tabla de la base de datos	
Flujo_Normal: 1. El sistema verifica que la tabla no este vacía y muestra un subconjunto del total de filas de la tabla. Por cada fila se muestran las primeras cinco (5) columnas siempre que estas no sean de tipo AUTO INCREMENT. En caso de mostrar una columna que es de tipo TEXT o de tipo BLOB, VARBINARY, BINARY o LONGVARBINARY, no se mostrará el contenido sino un texto que dice: Archivo o Texto, dependiendo del caso.	

Flujo_Alternativo: 1. El sistema verifica que la tabla no este vacía. Si lo está, muestra un mensaje que indica que la tabla seleccionada se encuentra vacía.
Poscondiciones: Se mostraron las filas de la tabla.

Nombre:	**Buscar**
Descripción: Permite mostrar todas las filas de la tabla seleccionada que posean en al menos uno de sus campos la palabra clave introducida por el usuario. Las filas se muestran usando paginación, con un valor que depende de la selección del usuario en la aplicación generadora. La paginación dependerá del número seleccionado por el usuario en la aplicación generadora y de la cantidad de filas que existan en la tabla.	
Actores: Usuario del Sitio Web	
Precondiciones: Haber seleccionado una tabla de la base de datos.	
Flujo_Normal: 1. El sistema muestra un campo de texto para introducir la palabra clave a buscar. Esta palabra clave puede ser toda una frase, no posee validación de ningún tipo. 2. El actor introduce la palabra clave. 3. El sistema busca en cada campo de la tabla, dicha palabra clave (sin tomar en cuentas diferencias entre mayúsculas o minúsculas). Cada fila, en la que al menos un campo coincida con la palabra clave, será mostrada a continuación. En el sistema se visualizará la palabra clave con la que se hizo la búsqueda.	
Flujo_Alternativo: 3. El sistema busca en cada campo de la tabla, dicha palabra clave (sin tomar en cuentas diferencias entre mayúsculas o minúsculas). Si no hay ninguna coincidencia, se muestra un mensaje que indica: "*No existen registros en la tabla que posean la palabra clave en alguno de sus campos*".	
Poscondiciones: Se listaron todas las filas de la tabla seleccionada cuyo campo(s) contiene la palabra clave introducida.	

Nombre:	**Detalle del Registro**
Descripción: Permite mostrar todos los valores contenidos en los campos de una fila seleccionada.	
Actores: Usuario del Sitio Web	
Precondiciones: Haber seleccionado una tabla de la base de datos y haber seleccionado una fila en particular.	
Flujo_Normal: 1. El sistema busca la fila seleccionada y extrae todos los campos asociados a la misma mostrándolos en la página en forma de tripletes: N° - Nombre del campo – Campo de Texto, en donde cada uno de estos tripletes corresponde con una columna de la tabla, siempre que esta columna no sea de tipo AUTO INCREMENT. El campo de texto tiene asignado el valor contenido en el campo. En el caso particular de que el campo almacene un archivo, no se mostrará el contenido de éste directamente, sino que aparecerá un enlace para ver o descargar ese archivo.	
Flujo_Alternativo: 1. El sistema busca la fila seleccionada. En el caso de no encontrarla el sistema muestra un mensaje al usuario que indica: "*La fila seleccionada no se encuentra en la tabla*".	
Poscondiciones: Se mostraron todos los valores asignados a los campos de una fila seleccionada.	

Nombre:	**Agregar**
Descripción: Permite insertar una nueva fila en la tabla seleccionada.	
Actores: Usuario del Sitio Web	
Precondiciones: Estar autenticado por el sistema, tener el privilegio para agregar y haber seleccionado una tabla de la BD.	
Flujo_Normal: 1. El actor selecciona la opción "*Agregar Nuevo Registro*". 2. El sistema muestra un formulario compuesto por varios tripletes: N° - Nombre del campo – Campo de	

Texto, en donde cada uno de estos tripletes corresponde con una columna de la tabla, siempre que esta columna no sea de tipo AUTO INCREMENT. En el caso particular de que el campo corresponda a una clave foránea, se muestra al lado derecho del campo de texto un enlace para seleccionar el valor de dicho campo. 3. El sistema valida los datos introducidos por el usuario de acuerdo a los tipos de datos y cantidad de caracteres que posee cada columna en la definición de la tabla en la base de datos y los almacena.
Flujo_Alternativo: 3. El sistema valida los datos introducidos por el usuario de acuerdo a los tipos de datos y cantidad de caracteres que posee cada columna en la definición de la tabla en la base de datos. Si los datos no son correctos, se le muestra un mensaje al actor de ello para permitir que los corrija.
Poscondiciones: Se almacenó una nueva fila en la tabla seleccionada.

Nombre:	**Modificar**
Descripción: Permite modificar los valores de los campos de una fila en la tabla seleccionada.	
Actores: Usuario del Sitio Web	
Precondiciones: Estar autenticado por el sistema, tener el privilegio para modificar y haber seleccionado una tabla de la BD.	
Flujo_Normal: 1. El sistema busca la fila seleccionada y extrae todos los campos asociados para mostrarlos en la página. 2. El actor selecciona la opción: "*Modificar*". 3. El sistema le permite al usuario modificar los valores de los campos de la fila. En el caso particular de que el campo almacene un archivo y éste campo no se encuentre vacío, se muestra un enlace para ver o descargar el archivo contenido en dicho campo, así como también la opción para cambiar el valor de dicho campo introduciendo un nuevo archivo. Si un campo corresponde a una clave foránea, se muestra al lado derecho del campo de texto un enlace para seleccionar el valor de dicho campo. 4. El sistema valida los datos introducidos por el usuario de acuerdo a los tipos de datos y cantidad de caracteres que posee cada columna en la definición de la tabla en la base de datos y los almacena.	

Flujo_Alternativo: 4. El sistema valida los datos introducidos por el usuario de acuerdo a los tipos de datos y cantidad de caracteres que posee cada columna en la definición de la tabla en la base de datos. Si los datos no son correctos, se le muestra un mensaje al actor de ello para permitir que los corrija.
Poscondiciones: Se modificaron los campos de la fila seleccionada de la tabla en la base de datos.

Nombre:	**Eliminar**
Descripción: Permite eliminar una fila en la tabla seleccionada.	
Actores: Usuario del Sitio Web	
Precondiciones: Estar autenticado por el sistema, tener el privilegio para eliminar y haber seleccionado una tabla de la BD.	
Flujo_Normal: 1. El sistema busca la fila seleccionada y extrae todos los campos asociados para mostrarlos en la página. 2. El actor selecciona la opción: "*Eliminar*". 3. El sistema elimina físicamente de la tabla en la base de datos la fila seleccionada.	
Flujo_Alternativo: Ninguno.	
Poscondiciones: Se eliminó la fila seleccionada de la tabla en la base de datos.	

Nombre:	**Ver Contenido del Archivo**
Descripción: Permite ver o descargar el contenido de un campo en la fila, que corresponda a un tipo de dato que represente un archivo: BLOB, VARBINARY, LONGVARBINARY, etc.	
Actores: Usuario del Sitio Web	

Precondiciones: El campo de la fila no debe ser vació y debe almacenar un archivo.
Flujo_Normal: 1. El actor selecciona el enlace "*Ver Archivo".* 2. El sistema busca la fila y extrae el archivo contenido en el campo. 3. El sistema abre el archivo directamente o lo descarga en el computador, según la selección previa del usuario.
Flujo_Alternativo: 2. El sistema busca la fila y si no la consigue, le muestra al actor un mensaje de error como el siguiente: *"Error para obtener el flujo del archivo"*.
Poscondiciones: Se mostró o descargó el archivo almacenado en la tabla de la base de datos.

Nombre:	**Cambiar Contenido del Archivo**
Descripción: Permite colocar un campo de texto para capturar la ruta de un archivo.	
Actores: Usuario del Sitio Web	
Precondiciones: El campo de la fila no debe ser vació y debe almacenar un archivo.	
Flujo_Normal: 1. El actor elige la opción "*Cambiar"*, luego de haber seleccionado la opción "*Modificar*". 2. El sistema extrae de la base de datos, todos los valores de la fila seleccionada excepto el valor contenido en el campo que almacena archivo. En cada uno de los campos de texto, coloca el valor de cada campo. En el campo que almacena el archivo coloca un campo de texto para capturar la ruta de un archivo nuevo.	
Flujo_Alternativo: Ninguno.	

Poscondiciones: Se mostró un campo de texto para capturar la ruta de un archivo.

Nombre:	**Seleccionar Clave Foránea**
Descripción: Permite colocar el campo clave de una fila de una tabla como referencia en una fila de otra tabla.	
Actores: Usuario del Sitio Web	
Precondiciones: El campo de la fila debe ser clave foránea.	
Flujo_Normal: 1. El actor elige la opción "*Seleccionar*", luego de haber seleccionado la opción "*Modificar*" o la opción "*Agregar*" 2. El sistema abre una nueva ventana en donde se listan todas las filas de la tabla a la cual esta relacionada la tabla que contiene la fila que se intenta modificar o agregar. 3. El actor selecciona el número de la fila. 4. El sistema cierra la ventana abierta y coloca el valor de la clave de la fila seleccionada en el campo que corresponde con la clave foránea.	
Flujo_Alternativo: 2. El sistema abre una nueva ventana en donde se listan todas las filas de la tabla a la cual esta relacionada la tabla que contiene la fila que se intenta modificar o agregar. Si no existen filas que listar, se muestra un mensaje que indica que esa tabla no contiene registros que mostrar.	
Poscondiciones: Se seleccionó y se colocó el valor de la clave primaria perteneciente a una fila como clave foránea de otra fila.	

Nombre:	**Cerrar Sesión**
Descripción: Permite desvincular al actor del sistema.	

Actores: Usuario del Sitio Web
Precondiciones: El usuario debe haberse autenticado en el sistema.
Flujo_Normal: 1. El actor selecciona la opción "*Cerrar Sesión"*, en el menú. 2. El sistema elimina las variables de sesión del actor y muestra un mensaje de despedida.
Flujo_Alternativo: Ninguna.
Poscondiciones: Se desvinculó al usuario del sistema.

4.2.2. Flujo de Trabajo: Diseño (Iteración 3)

Antes de explicar los artefactos producidos en esta iteración, es necesario definir el patrón implementado en la arquitectura de este sistema. El patrón implementado fue Modelo – Vista – Controlador, comúnmente llamado por sus siglas MVC.

Patrón de Diseño MVC [6]

Este patrón arquitectónico separa los datos, la interfaz de usuario y la lógica de control a través de:

Modelo: Es la representación específica del dominio de la información sobre la cual funciona la aplicación. El modelo es otra forma de llamar a la capa de dominio. La lógica de dominio añade significado a los datos.

Vista: Este componente presenta el modelo en un formato adecuado para interactuar, representa la interfaz con el usuario u otro sistema. No accede a la base de datos y no contiene lógica de negocio. El poco código no visual que posee la vista se limita a la lógica de presentación, como por ejemplo un ciclo que permita mostrar los elementos de un arreglo.

[6] http://es.wikipedia.org/wiki/Modelo_Vista_Controlador

Controlador: Es lo que manipula al modelo según la entrada del usuario. Este componente responde a eventos, basándose en la vista actual, en el estado del modelo y en las acciones llevadas a cabo por el usuario, el controlador invoca a la API del modelo para actualizar el estado del modelo y seleccionar la siguiente vista.

Para esta implementación estos tres componentes se representan por:

Modelo: Clases Java que encapsulan las operaciones de la aplicación.

Vista: Páginas JSP, HTML, XML, etc.

Controlador: Servlets.

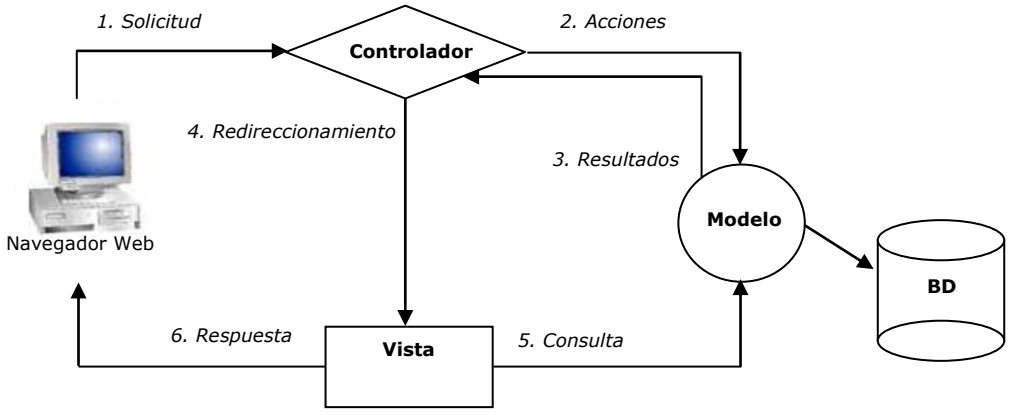


Figura 32: Esquema del patrón MVC.

Arquitectura del Sistema

Para el diseño de esta arquitectura, se definió como lenguaje de programación a utilizar: JAVA, implementando las tecnologías Servlets y JSP.

El usuario, a través del navegador (paso 1), le envía una petición al servidor (paso 2). El encargado de recibir la petición en el lado servidor es el Controlador (Servlets). El Controlador (paso 3), a través del modelo (Clases Java, EJBs) interactúa con el Sistema Manejador de Base de Datos (paso 4) para resolver la consulta. Una vez que se han obtenido los resultados (paso 5), el modelo retorna la respuesta al controlador (paso 6). El controlador genera la respuesta al cliente a través de las páginas JSP (paso 7), que el cliente recibe como páginas HTML (paso 8).

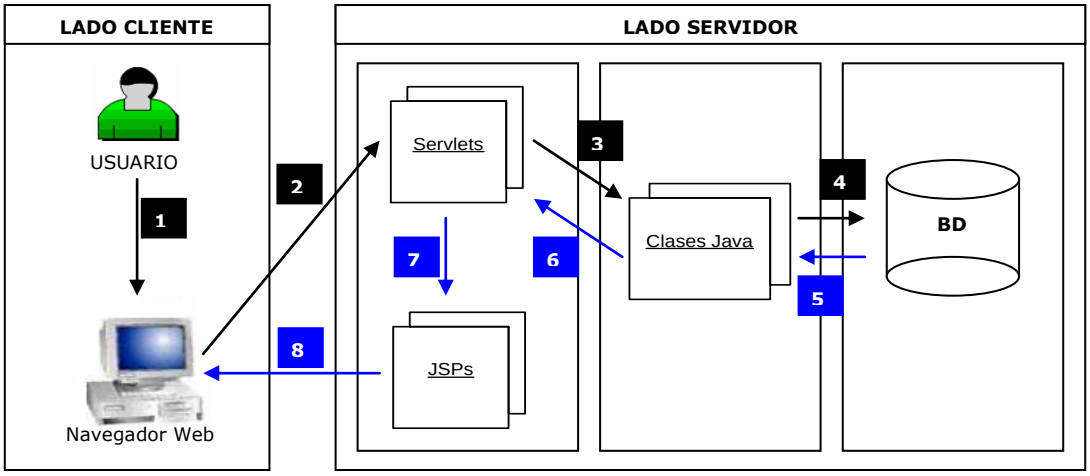


Figura 33: Arquitectura General del Sitio Web Generado.

Diagramas de Clases

Como ya sabemos, la aplicación web generada deberá gestionar diversas tablas de una base de datos y por cada una de estas tablas existirán varias clases para gestionarlas. Para simplificar el diagrama asumiremos que la aplicación web generada posee sólo una tabla que administrar, ya que la interacción con más de una tabla se haría de la misma forma pero con clases propias para cada clase en particular. Otro aspecto no tomado en cuenta para este diagrama, por simplicidad del mismo, fue no mostrar los atributos y los métodos de cada clase, A continuación se muestran los diagramas.

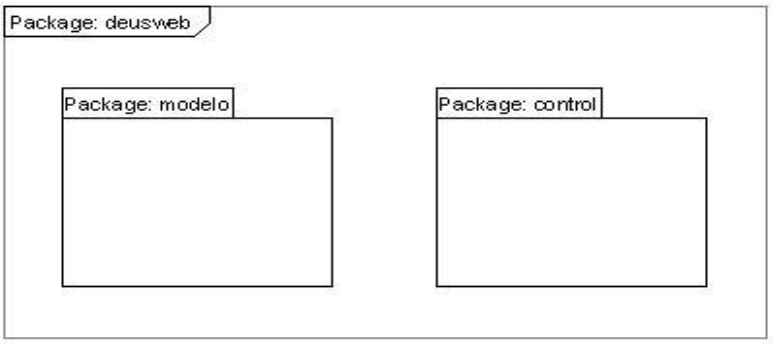


Figura 34: Diagrama de Clases Paquete DEUSWEB. Sitio Web Generado.

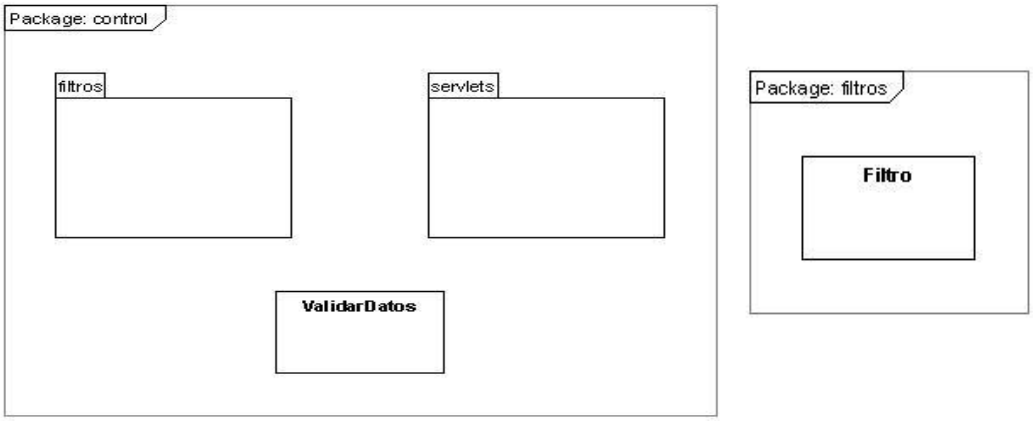


Figura 35: Diagrama de Clases Paquete control y filtros. Sitio Web Generado.

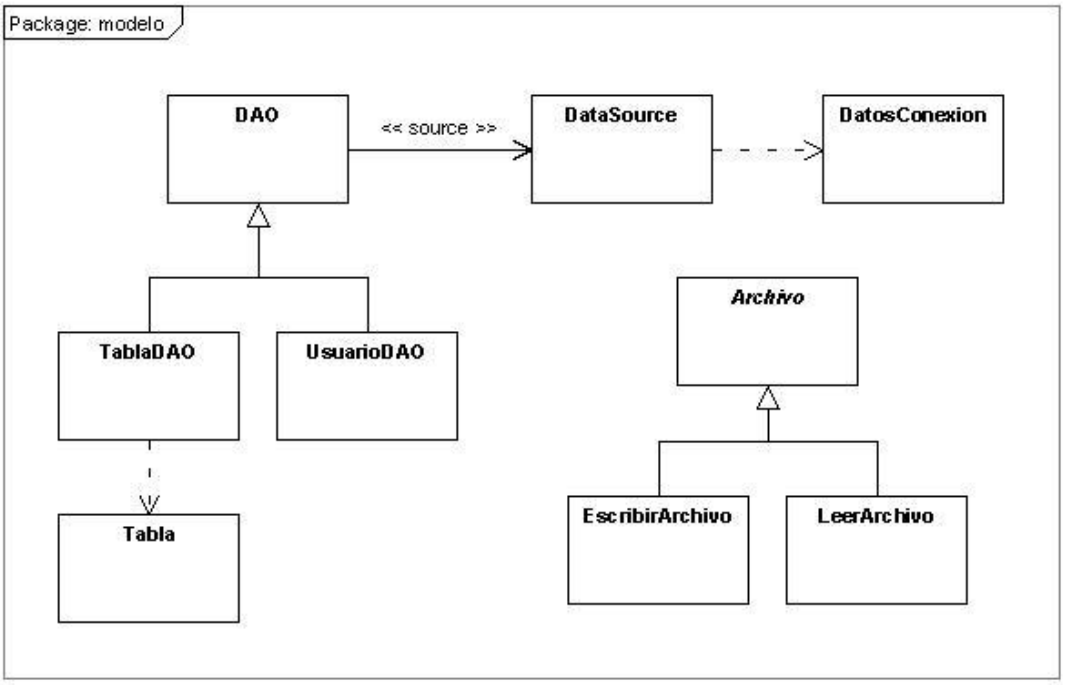


Figura 36: Diagrama de Clases Paquete modelo. Sitio Web Generado.

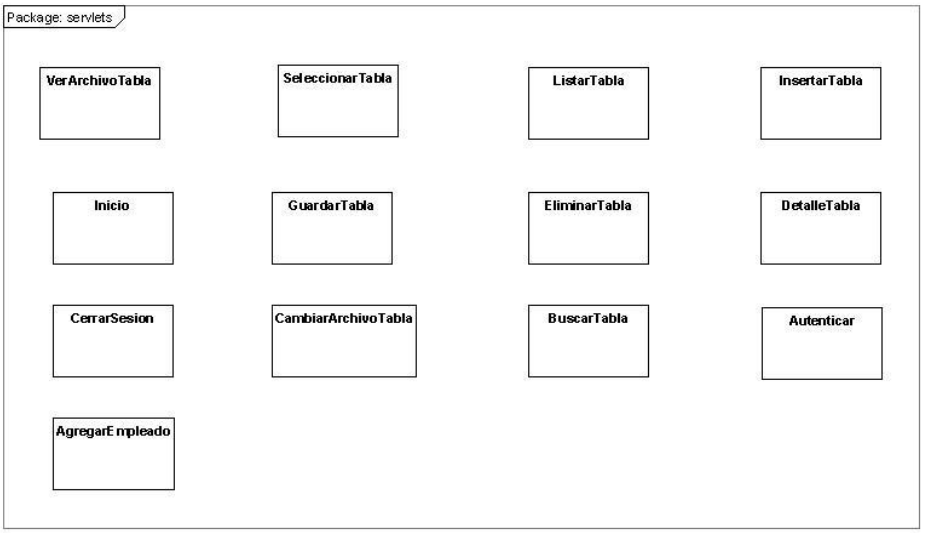


Figura 37: Diagrama de Clases Paquete servlets. Sitio Web Generado.

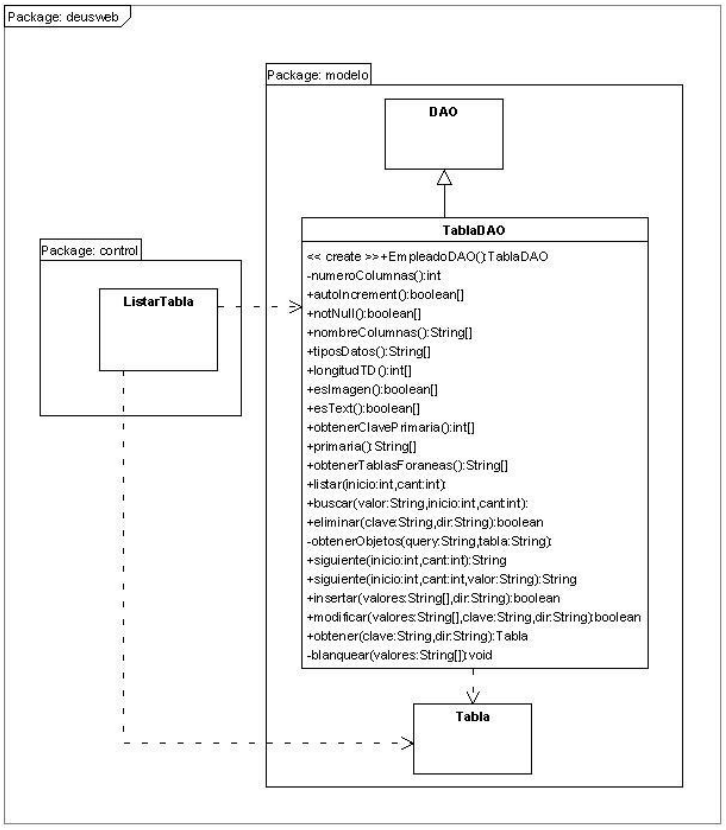


Figura 38: Diagrama de Clases Paquete DEUSWEB.
Ejemplo de las relaciones entre las clases del paquete control y el paquete modelo. Sitio Web Generado.

En este último diagrama se representa el ejemplo de interacción entre las clases de control y las clases del modelo. El usuario invoca a las clases de control para ejecutar una operación. Estas operaciones pueden ser, listar el contenido de una tabla, Buscar dentro de los campos de una tabla, Agregar una fila a la tabla, seleccionar una clave foránea, etc. En fin, el usuario puede invocar a cualquier operación representada por los servlels del paquete control. Estos servlets, para resolver el requerimiento del usuario, interactúan con las clases del modelo (*TablaDAO*), ya que estas, encapsulan las operaciones sobre la base de datos y generan una respuesta a los servlets. Estas respuestas pueden estar a su vez encapsuladas en otros objetos del modelo (*Tabla*) que representan las filas de las tablas en la base de datos. De esta forma el servlet verifica si la consulta al modelo fue exitosa o no, para enviar la respuesta correcta al usuario que hizo la solicitud en primer lugar.

A continuación se detalla, aun más, la interacción entre las clases de la aplicación, por medio del diagrama Web Applications Extensions (WAE).

Diagramas WAE (Web Applications Extensions)

- **Operación Listar:**

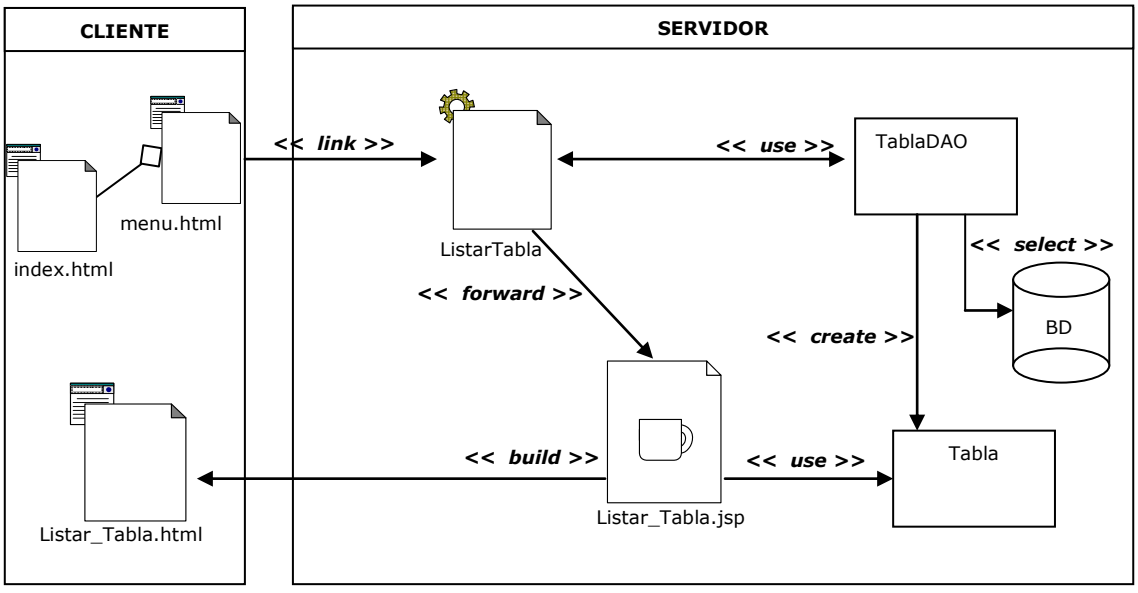


Figura 39: Diagrama WAE. Operación Listar.

- **Operación Agregar:**

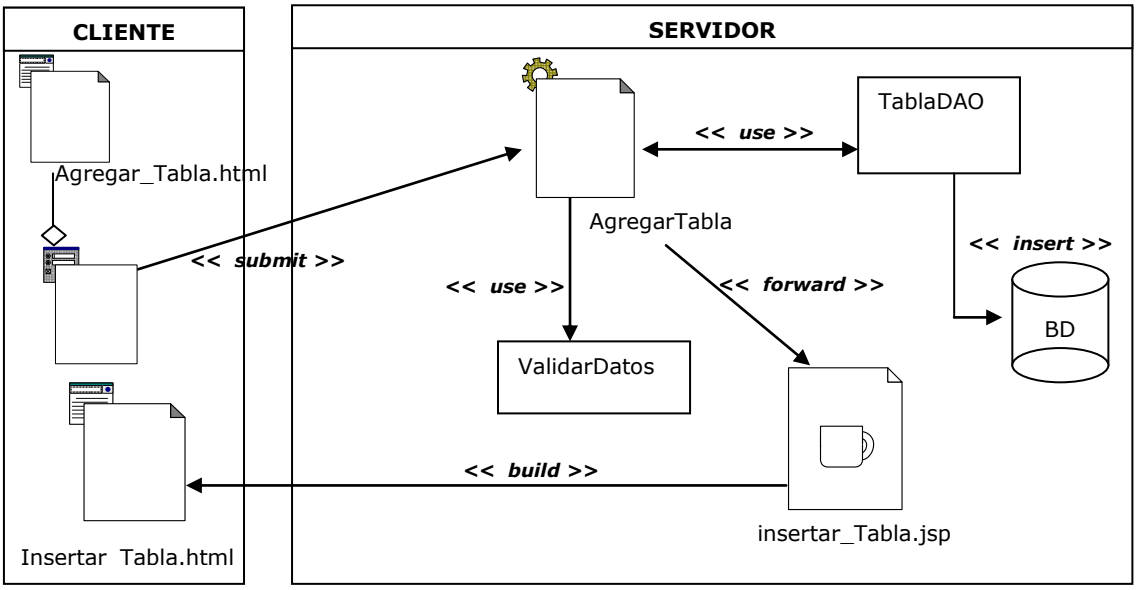


Figura 40: Diagrama WAE. Operación Agregar.

4.3. Etapa de Construcción

4.3.1. Flujo de Trabajo: Implementación (Iteración 4)

Antes de mostrar los artefactos producidos en esta iteración, es conveniente señalar algunas consideraciones que se tomaron en cuenta sobre la aplicación:

Consideraciones

Para el manejo de los archivos almacenados en las tablas de la base de datos, se utilizo un archivo: *control_upload.txt*, el cual ayuda a la gestión de los mismos. Gracias a este archivo es posible ver el contenido y descargar los archivos. Este archivo almacena la clave de la fila que contiene el archivo en la base de datos, el nombre de la tabla y el nombre de la columna, así como también el nombre del archivo que fue cargado en el sistema. Se implementó este archivo ya que era necesario almacenar el nombre del archivo a guardar en la base de datos sin obligar al usuario a definir un campo donde almacenar el nombre y sin obligar al sistema a modificar la base de datos suministrada por el usuario para almacenar este nombre.

La restricción sobre los archivos almacenados en la base de datos es que sólo puede existir un campo por tabla que almacene el contenido de un archivo.

Este sistema, identifica las relaciones entre las tablas a través de las claves foráneas y el sistema permite a través de un enlace, escoger el valor de una clave foránea para un registro particular. Otra consideración tomada en cuenta en esta iteración, corresponde al uso de estas claves foráneas. El sistema solo permite que una tabla se relacione con otra tabla, siempre que la clave foránea este compuesta por un solo campo.

Para el desarrollo de este sistema, también se necesitó identificar si el campo corresponde a un valor autoincrementable, de ser así, el sistema no muestra este campo ya que el propio sistema manejador de base de datos asigna el valor a este campo. Así mismo, se identificó el tipo de dato, la cantidad de caracteres que pueden almacenar cada uno de los campos y si este puede ser vacío, para realizar las validaciones del lado del servidor. De esta forma las validaciones que realiza el sistema depende de la definición de los campos en la base de datos.

A continuación se muestra la estructura de las carpetas creadas y seguidamente algunas pantallas del sistema.

Estructura de Directorios

- **Directorio principal**

Figura 41: Directorio Principal. Aplicación a Generar.

- **Directorio estilo:** En esta carpeta se encuentra el estilo del sitio web generado junto con las imágenes utilizadas por el estilo.

- **Directorio upload:** En esta carpeta se encuentra el archivo control_upload.txt, que almacena la información de las imágenes que se almacenan en la base de datos. Este archivo permite gestionar las imágenes contenidas en las tablas de la base de datos a través de la aplicación, para permitir un mejor control en las operaciones agregar, modificar, eliminar, y ver archivo.

- **Directorio WEB-INF:** En este directorio se almacenan las páginas de sitio así como los archivos de extensión .java que se utilizan. Igualmente se encuentra el descriptor de despliegue o archivo web.xml, en el que se establece el mapeo de los servlets de esta aplicación.

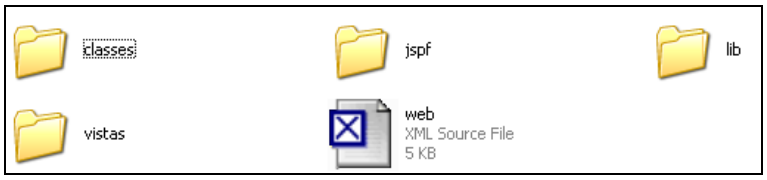


Figura 42: Directorio WEB-INF. Aplicación a Generar.

- **Directorio classes:** Se almacenan todos los archivos de extensión java que se utilizan en el sitio. Incluye los archivos del control y del modelo, que fueron descritos en los diagramas de clases señalados anteriormente.

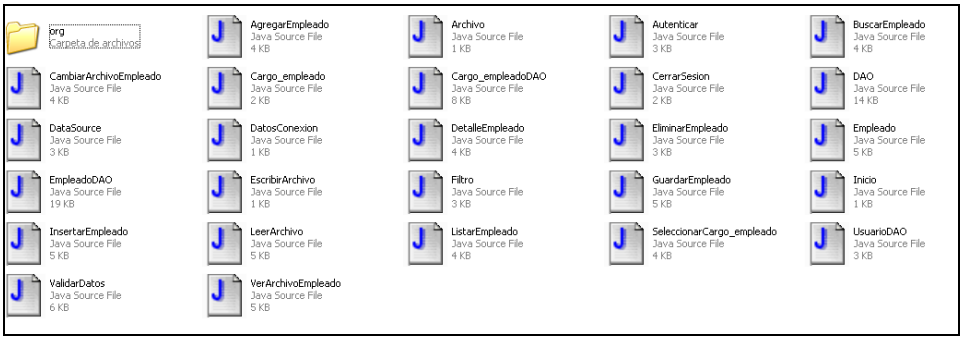


Figura 43: Directorio classes. Aplicación a Generar.

- **Directorio vistas:** Agrupa los archivos de extensión jsp. Corresponde a las páginas con las que interactúa el usuario de la aplicación.

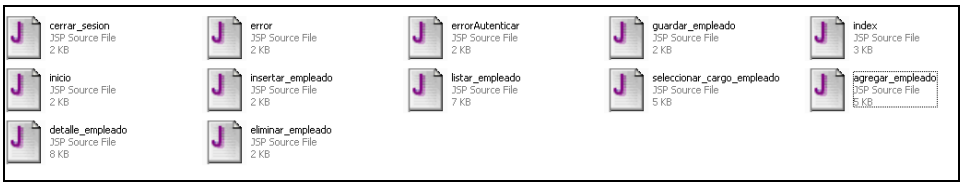


Figura 44: Directorio vistas. Aplicación a Generar.

- **Directorio jspf:** Agrupa los archivos de extensión jsp que son utilizado como *include* en los archivos del directorio vistas.

- **Directorio lib:** Agrupa los archivos .jar que utiliza la aplicación.

Pantallas del sistema

A continuación se muestran algunas pantallas del sistema. Se mostrarán todas las pantallas asociadas a una tabla en particular: Tabla Empleado. Por simplicidad sólo se muestran estas pantallas ya que para las demás tablas del sistema las pantallas son similares.

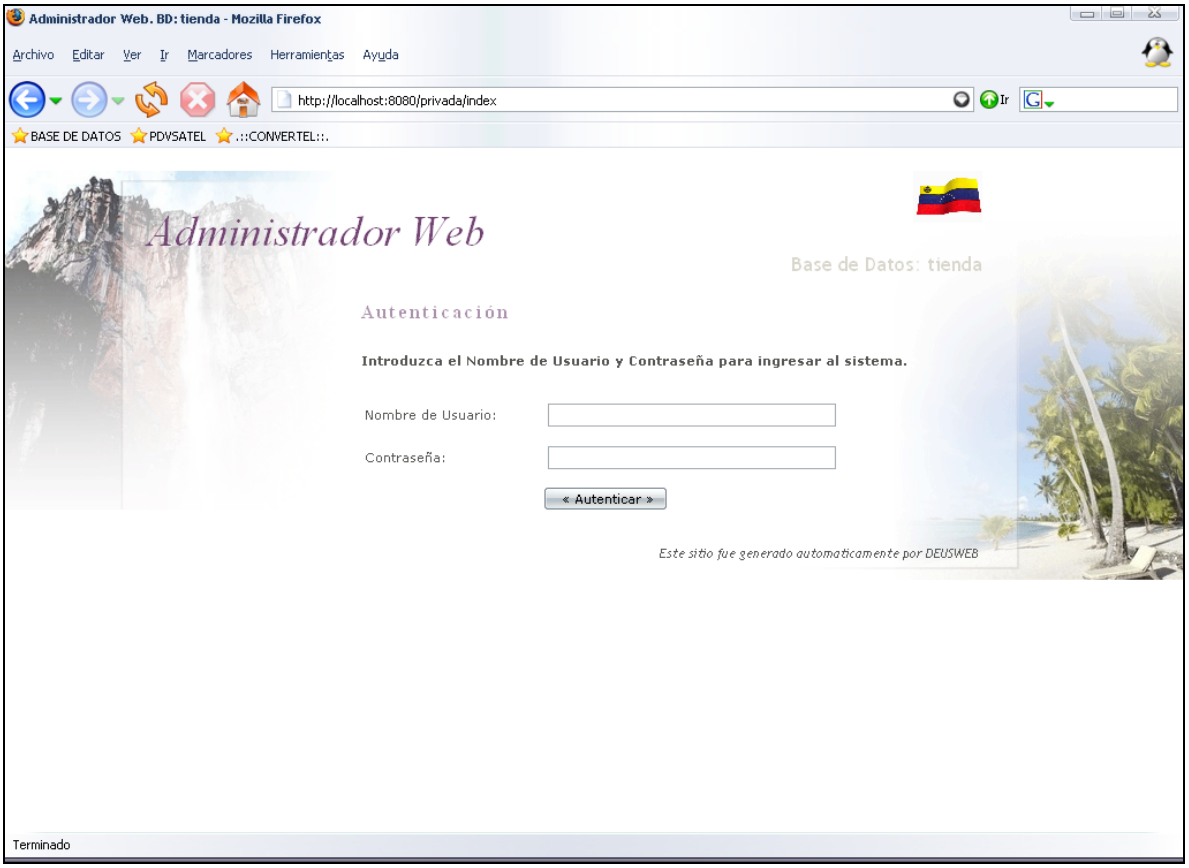


Figura 45: Página de Autenticación. Aplicación a Generar.

Figura 46: Página principal del sistema. Aplicación a Generar.

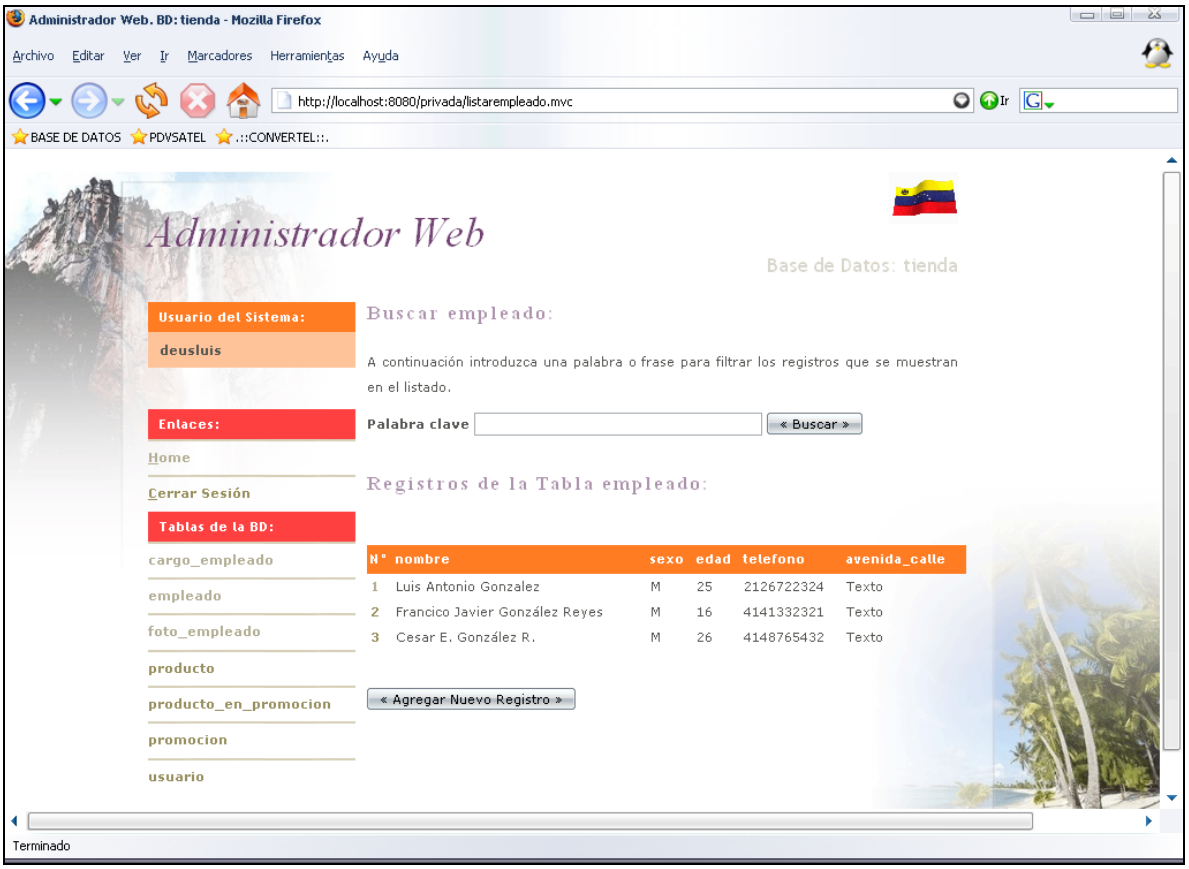


Figura 47: Página donde se listan los elementos de la tabla. Aplicación a Generar.

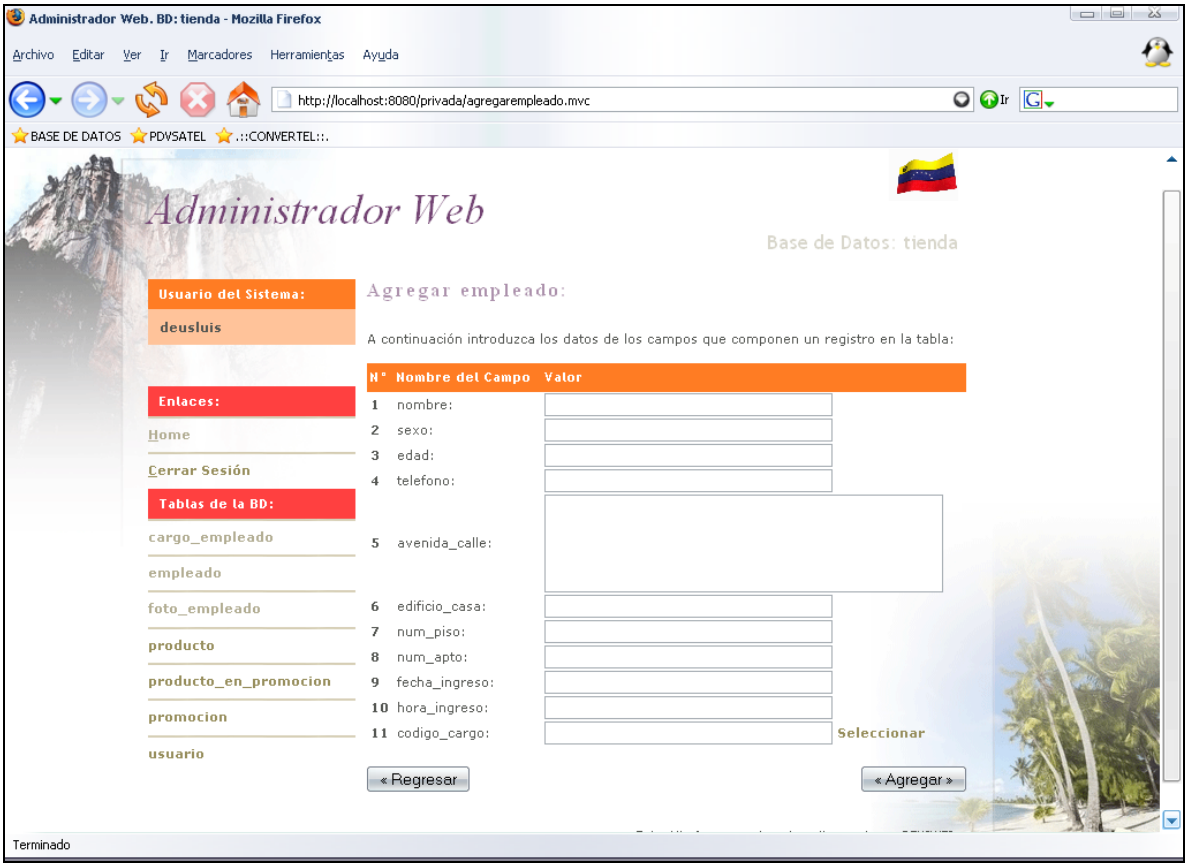


Figura 48: Página para agregar una fila a la tabla. Aplicación a Generar.

4.4. Resumen del Capítulo

En este capítulo, se describió el proceso de desarrollo implementado en la aplicación web administrativa. Esta aplicación será generada por la aplicación descrita en el próximo capitulo.

Fue necesario desarrollar esta aplicación (siguiendo el proceso de desarrollo), antes de desarrollar la aplicación generadora para conocer que archivos serán necesarios producir por esta última aplicación. En este capitulo se detallaron cada uno de los artefactos generados por los flujos de trabajo en cada una de las etapas del proceso de desarrollo.

Esta aplicación le permite al usuario gestionar de forma eficaz y eficiente las tablas de una base de datos, pudiendo así realizar operaciones tales como: autenticar al usuario, listar, agregar, modificar, eliminar y

buscar, cualquier información en la tabla de su preferencia. Así mismo provee otras funcionalidades, las cuales son: la manipulación de los archivos almacenados en la base de datos, a través del archivo *control_upload.txt*, el cual permite manipular los archivos cargados en las tablas de la base de datos sin necesidad de alterar la base de datos de entrada o de obligar al usuario a la definición de un campo que permita almacenar el nombre de los archivos; y la vista en pantalla de aquellos elementos de otras tablas que se relacionan con los datos de la tabla seleccionada, otorgando de esta forma una ayuda visual al usuario para la selección de la clave foránea en la tabla.

Las limitaciones sobre este sistema, se derivan de las funcionalidades extras que posee la aplicación. La restricción sobre los archivos almacenados en la base de datos es que solo puede existir un campo por tabla que almacene el contenido de un archivo y otra limitación corresponde al uso de las claves foráneas, en la que el sistema sólo identifica la relación de una tabla con otra tabla, siempre que la clave foránea este compuesta por un sólo campo.

Para esta aplicación, la etapa de transición no fue implementada, ya que ésta siguió el proceso de desarrollo a fin de conocer los archivos a producir por la aplicación generadora, que se describe en el próximo capítulo.

CAPÍTULO V:

APLICACIÓN GENERADORA (DEUSWEB)

El presente capitulo ahonda en la aplicación generadora (DEUSWEB), como un sistema "*standalone*" que genera aplicaciones web mediante la identificación de una base de datos MySQL. A través de este sistema es posible generar aplicaciones web administrativas que gestionan un conjunto de tablas pertenecientes a una base de datos (tal como se explicó en el capitulo anterior), de una forma rápida y fácil para el usuario.

A continuación se explica en detalle la aplicación generadora, identificando las etapas del proceso de desarrollo. En cada una de estas etapas, se detallan los artefactos generados por los flujos de trabajo.

5.1. Etapa de Inicio

5.1.1. Flujo de Trabajo: Requisitos (Iteración 1)

La aplicación generadora debe basarse en tres aspectos principales, a partir de los cuales se derivan las funcionalidades del sistema. Estos aspectos son los siguientes:

- **Identificación de la base de datos y selección de las tablas**: El sistema obtiene la base de datos y las tablas de esa base de datos que la aplicación generada deberá administrar. Esto se logra con la interacción del usuario y por la manipulación de los metadatos de la base de datos que realiza el sistema.

- **Extracción de la información de cada una de las tablas seleccionadas**: El sistema, extrae los metadatos de la base datos consiguiendo de esta forma toda la información necesaria de cada una de las tablas seleccionadas. Esta información es la siguiente: Nombre de las tablas y columnas de cada tabla, identificación del tipo de dato de cada columna, identificación de AUTO_INCREMENT y NOT_NULL, longitud de la columna, identificación de claves primarias y foráneas por cada tabla.

- **Generación del código fuente**: Con la información extraída en el paso anterior y cierta información suministrada por el usuario del sistema, que tiene que ver con la configuración del sitio, se genera un documento XML. El sistema a través de motores (se relaciona al tipo de código a producir el cual es generado a través de procesadores), genera la aplicación web administrativa. Teniendo como entrada el documento XML mencionado anteriormente, se ejecuta el motor, configurado en la aplicación a través de otro documento XML, para producir el código fuente de la aplicación web generada.

5.2. Etapa de Elaboración

5.2.1. Flujo de Trabajo: Análisis (Iteración 2)

Diagramas de Casos de Uso

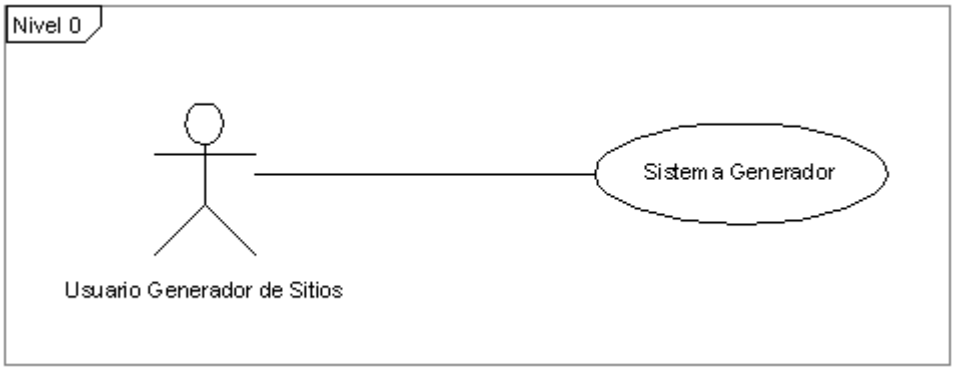


Figura 49: Nivel 0. Diagrama de Casos de Uso. Sistema Generador.

Usuario Generador de Sitios: Corresponde al usuario del sistema. Es la persona que se encarga de interactuar con el sistema para generar una aplicación web administrativa. Debe poseer un conocimiento básico sobre base de datos y páginas web, para así suministrar la información requerida por el sistema.

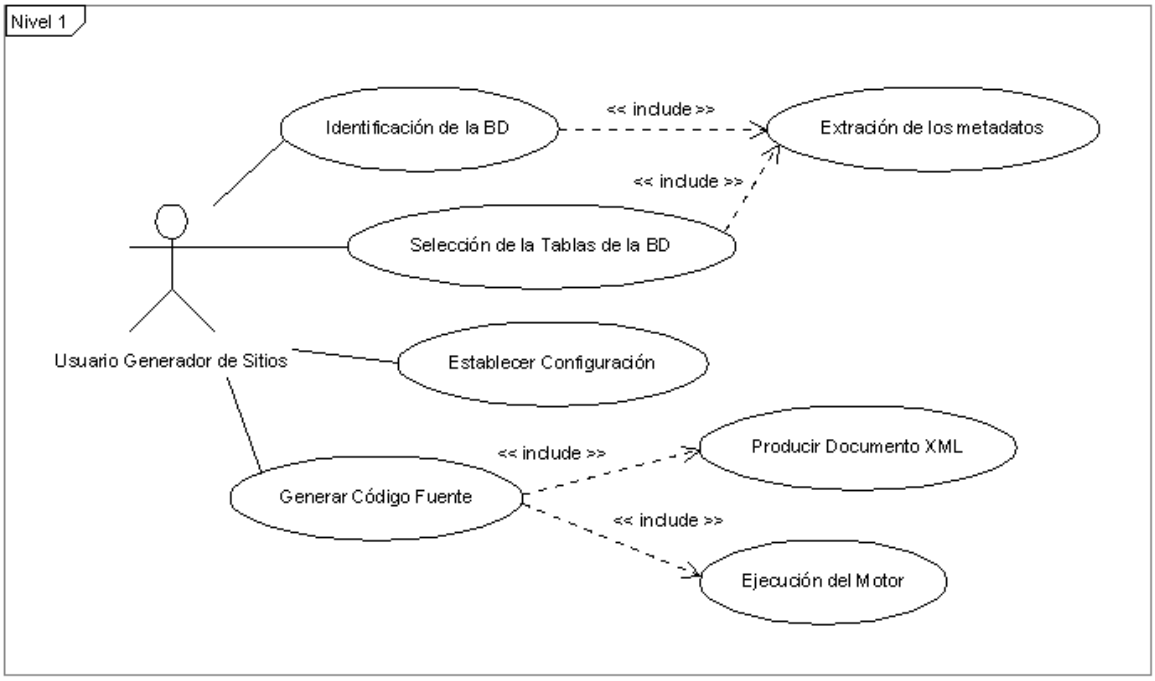


Figura 50: Nivel 1. Diagrama de Casos de Uso. Sistema Generador.

Descripción de Casos de Uso

A continuación se describe cada una de las funcionalidades del sistema, explicando cada uno de los casos de uso descritos en el diagrama anterior. En este punto se explica en detalle que debe hacer cada funcionalidad para cumplir con su objetivo particular.

En la descripción de los casos de uso, fueron tomados en cuenta los siguientes aspectos: Nombre, Descripción, Actores, Precondiciones, Flujo Normal, Flujo Alternativo, y Poscondiciones. Estos aspectos fueron definidos en el capitulo III, correspondiente al Proceso de Desarrollo.

Nombre:	**Identificación de la base de datos**
Descripción: Permite identificar la base de datos.	
Actores: Usuario del Sistema Generador	

Precondiciones: Ninguna
Flujo_Normal: 1. El sistema muestra una lista para que el actor seleccione el sistema manejador de base de datos. Así mismo, muestra los campos para que el actor introduzca los parámetros de conexión a la base de datos: URL, usuario y contraseña. 2. El actor introduce los valores requeridos. 3. El sistema verifica que el usuario haya introducido los campos obligatorios: URL y usuario, e intenta establecer la conexión con el sistema manejador de base de datos. 4. Si la conexión es exitosa, extrae los nombres de las tablas contenidas en la base de datos, si la conexión no fue exitosa muestra un mensaje de error al actor. 5. El sistema desbloquea el botón de Siguiente para que el actor acceda al siguiente paso.
Flujo_Alternativo: 3. El sistema verifica que el actor haya rellenado los campos obligatorios: URL y usuario, en caso de error el sistema le muestra un mensaje de error al usuario.
Poscondiciones: Se estableció la conexión a la base de datos extrayendo los nombres de las tablas contenidas.

Nombre:	**Selección de las Tablas de la Base de Datos**
Descripción: Permite identificar las Tablas	
Actores: Usuario del Sistema Generador	
Precondiciones: Haber identificado la base de datos	
Flujo_Normal: 1. El sistema mostrará dos listas de selección múltiple, la primera mostrará el contenido de todas las tablas existentes en la base de datos y la segunda mostrará las tablas de la base de datos que haya seleccionado el usuario.	

2. El actor seleccionará las tablas que desea administrar a través de la aplicación generada. 3. El sistema desbloquea el botón de Siguiente para que el usuario acceda al siguiente paso.
Flujo_Alternativo: Ninguno
Poscondiciones: Se identificaron las tablas que serán administradas por el sitio web a generar.

Nombre:	**Extracción de los metadatos**
Descripción: Permite extraer los metadatos de la base de datos	
Actores: Identificación de la Base de Datos, Selección de las Tablas de la Base de Datos	
Precondiciones: Ninguna	
Flujo_Normal: 1. El sistema a través de los métodos proporcionados por el API JDBC del Sistema Manejador de Base de Datos, extrae la información concerniente a la estructura de la base de datos: nombre de las tablas, nombre de las columnas, tipo de dato de las columnas, cantidad de caracteres que puede almacenar cada columna, obtener si la columna es AUTO INCREMENT, obtener si la columnas es NOT NULL, obtener los nombres de las columnas que conformen la clave primaria, obtener los nombres de las columnas que conformen la clave foránea y obtener los nombres de las tablas con las cuales se relaciona.	
Flujo_Alternativo: Ninguno	
Poscondiciones: Se extrajeron los metadatos de la base de datos	

Nombre:	**Establecer Configuración**
Descripción: Permite establecer los parámetros de configuración del sitio web a generar.	
Actores: Usuario del Sistema Generador	
Precondiciones: Haber seleccionado las tablas de la base de datos	
Flujo_Normal: 1. El sistema mostrará 4 fichas en la pantalla, cada una de las cuales corresponde a diferentes aspectos de la configuración del sitio. Las fichas serán las siguientes: Autenticación, Paginación, Apariencia, Destino y Lenguaje. 2. En la ficha de Autenticación el actor escoge si quiere o no la pagina de autenticación para el sitio a generar. En caso de escoger: Si, deberá identificar la tabla con la cual el sistema podrá autenticar al usuario. Así mismo deberá identificar las columnas: login, password y de privilegios para las funcionalidades del sitio. 3. En la ficha de Paginación, el actor seleccionará el número de registros que se listaran por cada tabla, a través de la funcionalidad: Listar, del sitio web a generar. 4. En la ficha de Apariencia, el actor seleccionará la hoja de estilo a utilizar por el sitio web a generar, de un pool de hojas de estilo que dispondrá el sistema a través de un documento XML que cargará. 5. En la ficha Destino y Lenguaje, el actor seleccionará la ruta donde se ubicará un archivo comprimido de extensión *war*, con los archivos del sitio web generado. Así mismo seleccionará el motor que será utilizado para generar los archivos fuentes del sitio web. Estos motores serán cargados por el sistema, en base a un documento XML que contendrá dicha información. 6. El sistema desbloquea el botón de Siguiente para acceder al siguiente paso.	
Flujo_Alternativo: Ninguno	
Poscondiciones: Se establecieron los parámetros de configuración del sitio web a generar.	

Nombre:	**Generar Código Fuente**
Descripción: Permite generar los archivos fuentes que corresponden a un sitio web administrativo.	
Actores: Usuario del Sistema Generador	
Precondiciones: Haber establecido los parámetros de configuración del sitio.	
Flujo_Normal: 1. El sistema producirá un documento XML con los metadatos extraídos. 2. De acuerdo al motor seleccionado por el actor en pasos previos, extraerá de un documento XML el nombre del motor que contiene los procesadores que realizarán la tarea de generar el sitio web administrativo. 3. Se mostrará una barra de progreso para indicarle al actor el avance en la producción de los archivos fuentes del sitio. 4. Se muestra un mensaje de éxito en la ejecución de generación del sitio web administrativo. También desbloquea el Botón de Nuevo, que permite iniciar los pasos de generación de sitios nuevamente.	
Flujo_Alternativo: 3. El sistema mostrará una barra de progreso para indicarle al actor el avance en la producción de los archivos fuentes del sitio. En caso de ocurrir algún error, el sistema detiene la ejecución de los motores y muestra un mensaje de error al usuario.	
Poscondiciones: Se generó el sitio web administrativo.	

Nombre:	**Producir Documento XML**
Descripción: Permite generar un documento XML con los metadatos extraídos de la base de datos	
Actores: Generar Código fuente	

Precondiciones: Ninguna
Flujo_Normal: 1. El sistema con los metadatos extraídos en pasos previos, genera un documento XML, que incluye la siguiente información: 1.1. Parámetros de Conexión a la base de datos: Incluye el nombre del Driver a utilizar, la ruta del driver, la URL de conexión a la base de datos, el nombre de usuario y la contraseña. 1.2. Tipos de Datos: Corresponde a la identificación de cada uno de los tipos de datos existentes en *java.sql.Types.* 1.3. Tablas: Corresponde al nombre de las tablas seleccionadas y por cada una de ellas también se coloca la información de los campos. 1.4. Autenticación: Corresponde a la información suministrada para la generación de la página de autenticación del sitio a generar. 1.5. Paginación: Indica el valor asignado a la paginación del sitio a generar. 1.6. Apariencia: Indica el nombre de la hoja de estilos del sitio a generar. 1.7. Destino y Lenguaje: Indica la ruta de destino del archivo de extensión *war* que será generado. Así mismo, se indica el nombre del motor al cual ha de referenciar. Con este nombre, el sistema sabrá que motor será ejecutado.
Flujo_Alternativo: Ninguno
Poscondiciones: Se generó el documento XML con la información suministrada por el usuario junto con la información de los metadatos de la base de datos.

Nombre:	**Ejecución del Motor**
Descripción: Permite ejecutar los procesadores (clases java), que realizan el proceso de generación del sitio.	
Actores: Generar Código fuente	

Precondiciones: Haber generado el documento XML
Flujo_Normal: 1. El sistema, toma el nombre del motor especificado en el documento XML generado en pasos previos, y que contiene la información que seleccionó el usuario. 2. Se extrae el nodo completo del documento XML que posee la especificación de los procesadores a ejecutar, de acuerdo al nombre del motor especificado en el paso anterior. 3. Se ejecutan cada uno de los procesadores. 4. Se produce el sitio web a generar.
Flujo_Alternativo: Ninguno
Poscondiciones: Se generó el sitio web administrativo.

5.2.2. Flujo de Trabajo: Diseño (Iteración 3)

Arquitectura

Para el diseño de esta arquitectura, se definió como lenguaje de programación a utilizar: JAVA, utilizando el paquete *java.swing* para la generación de las interfaces gráficas. Se definió que este sistema será una aplicación "*standalone*" implementado como un asistente o wizard para la generación del sitio web administrativo.

El usuario a través de la interfaz de usuario (paso 1) completa los datos requeridos por la aplicación. Estos datos son enviados a una clase controladora (paso 2), la cual se encarga de hacer las validaciones correspondientes en caso de ser necesario (paso 3). Posteriormente el controlador invoca a las clases que poseen ciertas operaciones que el controlador no se encarga de manejar, (paso 4) algunas de las cuales realizan la conexión a la base de datos, otras manipularán archivos y otras se encargarán de la generación del sitio web. Luego de obtener los resultados, el controlador retorna la respuesta al usuario a través de la interfaz de usuario (paso 5).

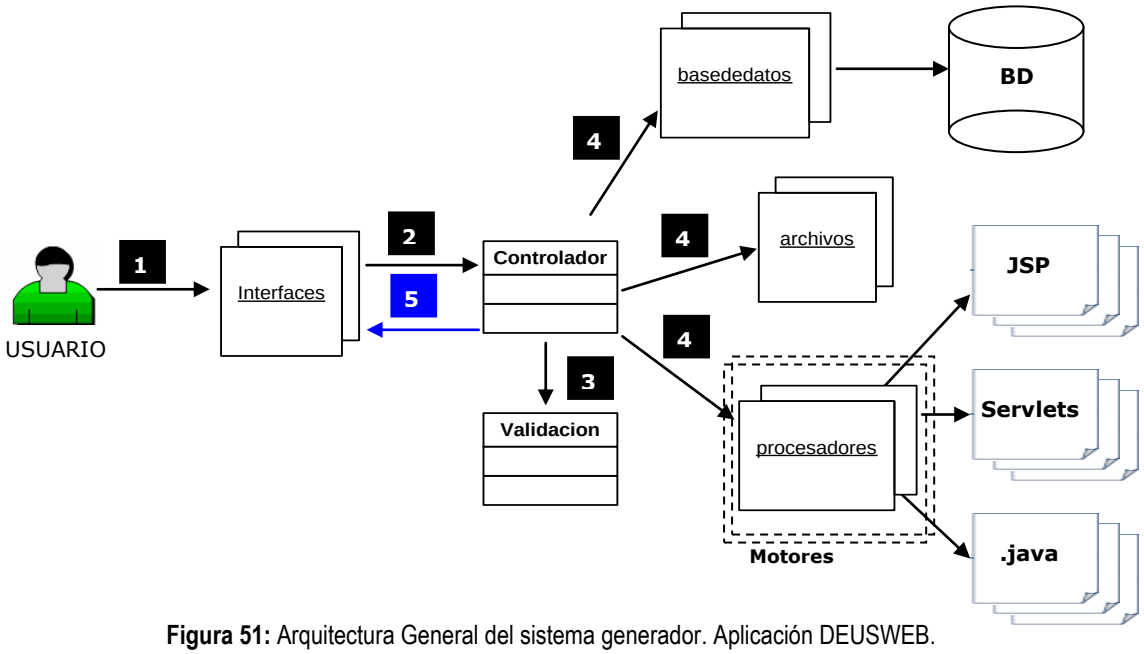


Figura 51: Arquitectura General del sistema generador. Aplicación DEUSWEB.

Diagramas de Clases

Para simplificar el diagrama no se muestran los atributos y los métodos de cada clase, A continuación se muestran los diagramas.

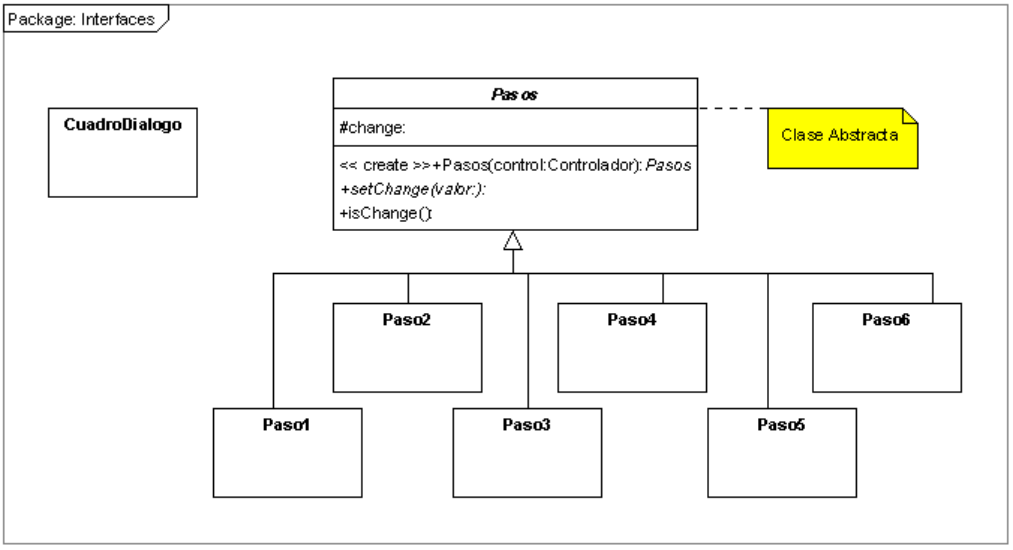


Figura 52: Diagrama de Clases Paquete interfaces. Aplicación DEUSWEB.

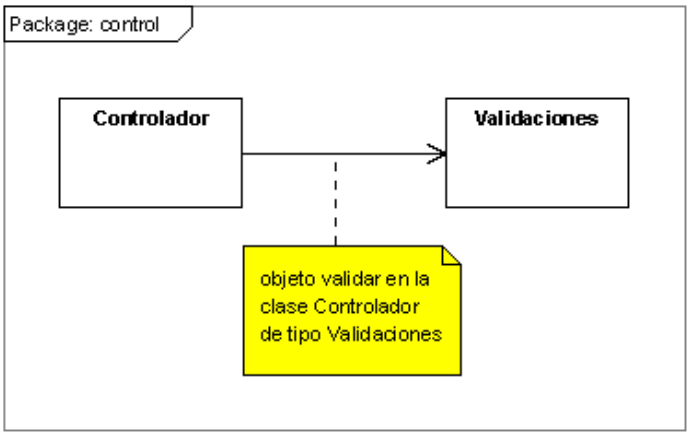


Figura 53: Diagrama de Clases Paquete control. Aplicación DEUSWEB.

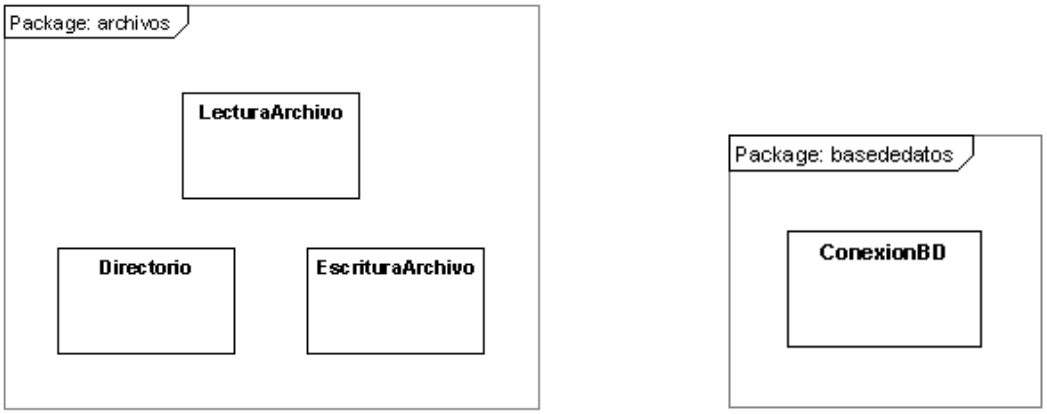


Figura 54: Diagrama de Clases Paquete archivos y basededatos. Aplicación DEUSWEB.

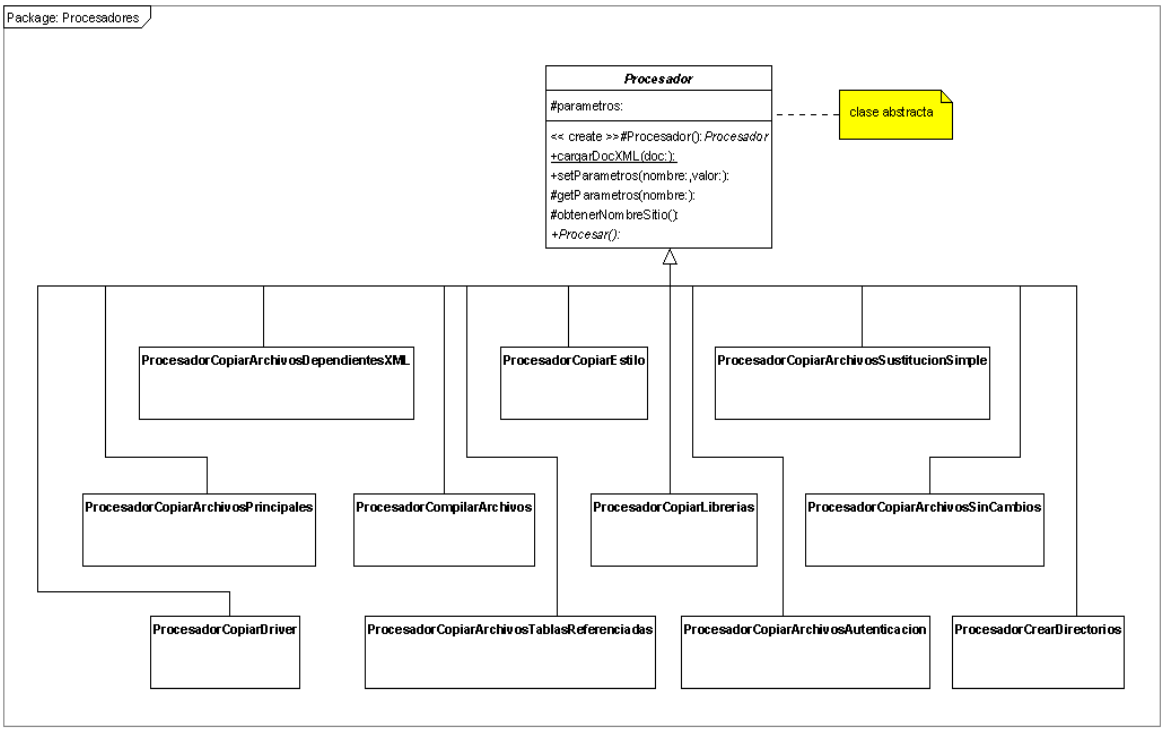


Figura 55: Diagrama de Clases Paquete procesadores. Aplicación DEUSWEB.

Diagramas de Secuencia

A continuación se muestra, en la figura 58, el diagrama de secuencia para la ejecución de la tarea en la que se extrae cada uno de los nombres de las tablas existentes en la base de datos introducida por el usuario.

Seguidamente, en la figura 59, se muestra el diagrama de secuencia en donde se observa la interacción de las clases involucradas en el proceso de generación de los archivos fuentes del sitio web administrativo.

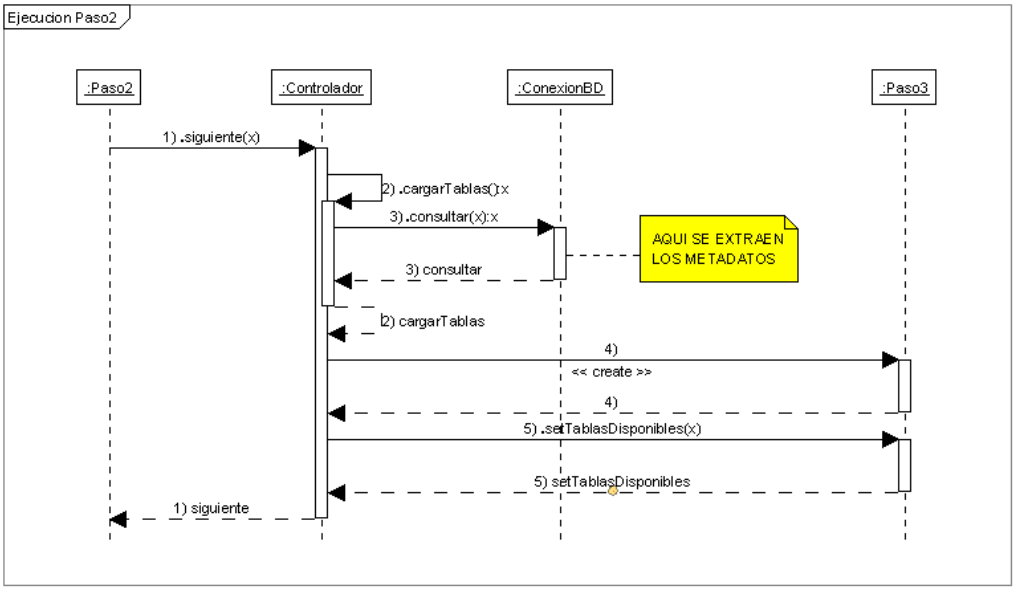


Figura 56: Diagrama de Secuencia. Extracción de los nombres de las Tablas

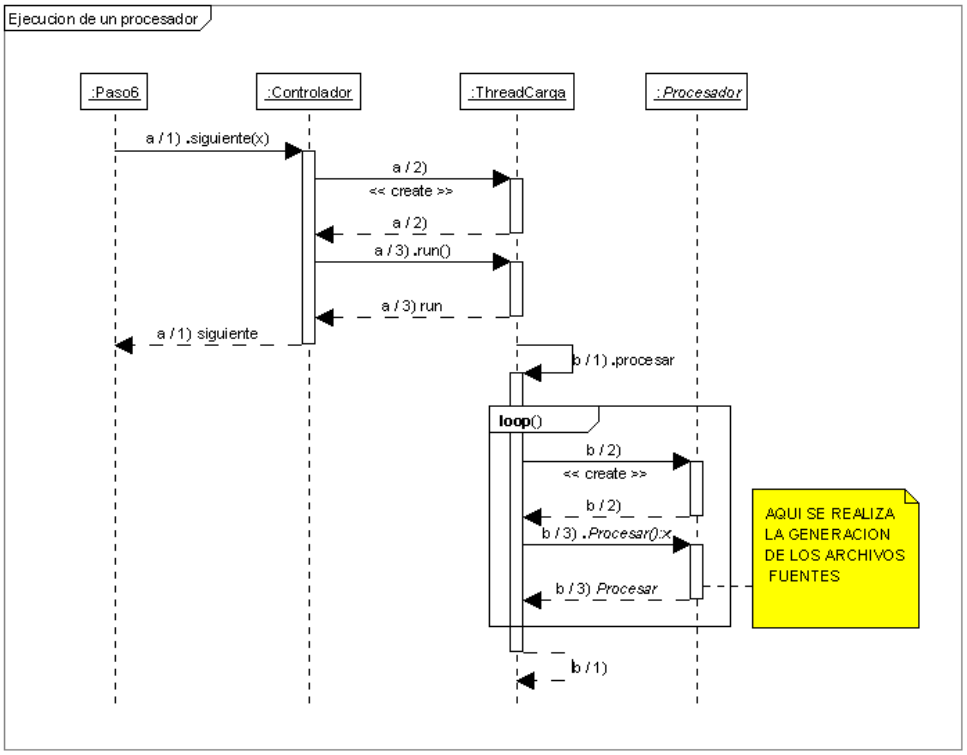


Figura 57: Diagrama de Secuencia. Ejecución de un procesador.

5.3. Etapa de Construcción

5.3.1. Flujo de Trabajo: Implementación (Iteración 4)

En esta iteración se producen las clases correspondientes a las interfaces controlador y archivos para posteriormente en la próxima iteración producir los procesadores.

A continuación se describe la estructura de directorios, así como algunos archivos generados por esta iteración y algunas pantallas del asistente.

Estructura de Directorios

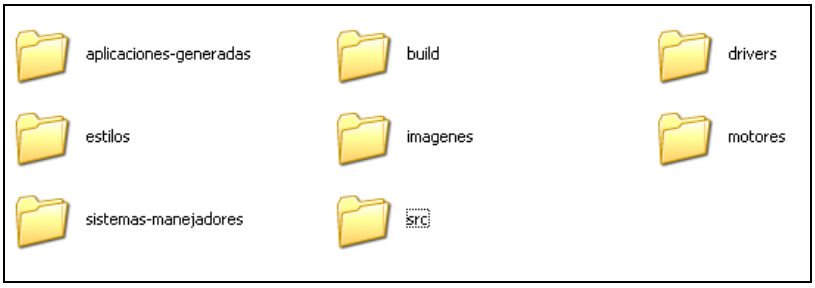


Figura 58: Directorio Principal. Aplicación DEUSWEB.

Directorio aplicaciones-generadas: Esta carpeta, almacena los archivos temporales generados por los procesadores, antes de generar el archivo de extensión *war*.

Directorio build: Contiene los archivos compilados de los archivos fuentes que componen la aplicación. Se organizan en paquetes.

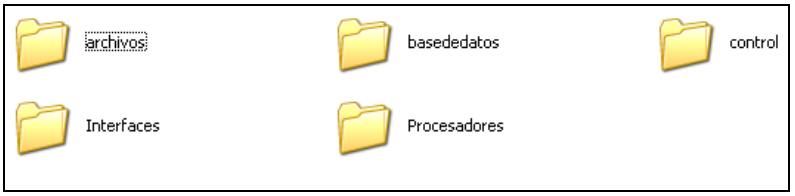


Figura 59: Directorio build. Aplicación DEUSWEB.

Directorio drivers: Contiene los archivos de extensión *jar* que son utilizados por la aplicación. Algunos de estos archivos son copiados a la carpeta lib de la aplicación generada. Este directorio también incluye un archivo *drivers.xml*, el cual contiene la ruta de los drivers de conexión a la base de datos que posee el sistema.

Directorio estilos: Esta carpeta, contiene subdirectorios con los estilos que se pueden aplicar al sitio web generado. La aplicación carga en el paso 4: ficha Apariencia, una lista con los nombres de estos subdirectorios que contienen los estilos.

Directorio imágenes: Almacena todas las imágenes generadas por la aplicación.

Directorio motores: Almacena los archivos necesarios por el motor implementado (conjunto de procesadores) para generar el sitio web. Incluye un archivo *motores.xml*, el cual indica que clases java (procesadores) deben ejecutarse por cada uno de los motores.

Directorio sistemas-manejadores: Almacena el archivo *sistemas-manejadores.xml* el cual describe los sistemas manejadores de base de datos que maneja la aplicación.

Directorio src: Contiene los archivos fuentes de la aplicación. Se organizan en paquetes (*figura 62*).

Archivos utilizados

sistemas-manejadores.xml: es un archivo utilizado por la interfaz, en el paso 2, para cargar los tipos de sistemas manejadores que maneja la aplicación. Este archivo se encuentra en la carpeta *sistemas-manejadores*.

```
<?xml version="1.0" encoding="UTF-8"?>
<sistemas-manejadores>
  <smbd>
    <nombre>MySQL</nombre>
    <driver>com.mysql.jdbc.Driver</driver>
    <url-defecto>jdbc:mysql://localhost:3306/</url-defecto>
    <url-
ayuda>jdbc:mysql://servidor:puerto/base_de_datos</url-ayuda>
  </smbd>
</sistemas-manejadores>
```

Figura 60: Documento XML. Sistemas-manejadores.xml

drivers.xml: es un archivo utilizado por el sistema para conocer la ruta de los drivers de conexión a la base de datos disponibles. Este archivo se encuentra en la carpeta *drivers*.

```
<?xml version="1.0" encoding="UTF-8"?>
<drivers>
     <driver>
        <nombre>com.mysql.jdbc.Driver</nombre>
        <ruta>d:/Mis documentos/AplicacionTesis/drivers/mysql-
connector-java-3.1.12-bin.jar</ruta>
     </driver>
</drivers>
```

Figura 61: Documento XML. Drivers.xml

motores.xml: es un archivo utilizado por el sistema el cual indica que tipo de código puede ser generado por la aplicación para la aplicación generada. Es utilizado por el sistema en el paso 4: Ficha Destino y Lenguaje, para extraer el nombre de los motores existentes.

Este archivo especifica que clases java (procesadores) deben ejecutarse por cada uno de los motores implementados. Cada motor se compone de un nombre y de elementos. Un elemento es una tarea que ejecutar para generar la aplicación web, generalmente esta tarea consiste en generar ciertas páginas, pero también puede consistir en generar la estructura de directorio, en ejecutar la compilación u otra tarea específica. Cada elemento se compone de un nombre, descripción y de un conjunto de procesadores. Estos procesadores no son más que la especificación de una clase Java que ejecuta una operación. El conjunto de procesadores se encarga de cumplir con la generación del elemento al cual pertenecen. Cada procesador especifica una clase y también pueden especificar los parámetros que se necesiten para su ejecución. A continuación se muestra este archivo simplificado, y el uso para la ejecución de los procesadores se analiza en la próxima iteración.

```
<?xml version="1.0" encoding="UTF-8"?>
<motores>
    <motor>
       <nombre>JSP 2.0 / Servlets</nombre>
       <elementos>
             <elemento>
                <nombre>Creacion de Directorios</nombre>
                <descripcion>
                              Se crean todos los directorios que
                              posee el sitio
                </descripcion>
                <procesadores>
                   <procesador>
                     <nombre>Crear Directorios</nombre>
                     <clase>
                       Procesadores.ProcesadorCrearDirectorios
                     </clase>
                     <parametros>
                         <parametro>
                           <nombre>directorioBase</nombre>
                           <valor>
    d:/Mis documentos/AplicacionTesis/aplicaciones-generadas/
                           </valor>
                         </parametro>
                     </parametros>
                  </procesador>
                </procesadores>
             </elemento>
             .
             .
             .
       </elementos>
    </motor>
</motores>
```

Figura 62: Documento xml. Motores.xml

Archivos generados

Documento xml generado a través de la interacción con el usuario y de la extracción de los metadatos de la base de datos. A continuación se describe el documento xml generado en esta iteración el cual constituye el artefacto de entrada para la próxima iteración. Se muestra el documento xml simplificado.

```
<?xml version="1.0" encoding="UTF-8"?>
<DEUSWEB>
  <conexion>
    <driver>com.mysql.jdbc.Driver</driver>
    <ruta>d:/Mis documentos/AplicacionTesis/drivers/mysql-connector-java-
3.1.12-bin.jar</ruta>
    <url>jdbc:mysql://localhost:3306/tienda</url>
    <usuario>root</usuario>
    <contrasena />
  </conexion>
  <tipos-datos>
    <tipo>
      <nombre>ARRAY</nombre>
      <id>2003</id>
    </tipo>
    <tipo>
      <nombre>BIGINT</nombre>
      <id>-5</id>
    </tipo>
  ...
  </tipos-datos>
  <tablas>
    <tabla>
      <nombre-tabla>cargo_empleado</nombre-tabla>
      <clave-primaria>
        <nombre-columna>codigo</nombre-columna>
        <nombre-columna>nombre_cargo</nombre-columna>
      </clave-primaria>
      <clave-foranea />
      <columnas>
        <columna not-null="false" autoincrement="true">
          <nombre>codigo</nombre>
          <tipo-dato>4</tipo-dato>
          <longitud>10</longitud>
        </columna>
        <columna not-null="false" autoincrement="false">
          <nombre>nombre_cargo</nombre>
          <tipo-dato>12</tipo-dato>
          <longitud>50</longitud>
        </columna>
        <columna not-null="false" autoincrement="false">
          <nombre>descripcion</nombre>
          <tipo-dato>-1</tipo-dato>
          <longitud>65535</longitud>
        </columna>
```

```
        <columna not-null="false" autoincrement="false">
          <nombre>sueldo</nombre>
          <tipo-dato>7</tipo-dato>
          <longitud>12</longitud>
        </columna>
      </columnas>
    </tabla>
    <tabla>
      <nombre-tabla>empleado</nombre-tabla>
      <clave-primaria>
        <nombre-columna>codigo</nombre-columna>
      </clave-primaria>
      <clave-foranea>
        <clave>
          <nombre-columna>codigo_cargo</nombre-columna>
          <tabla-referencia>cargo_empleado</tabla-referencia>
          <columna-referencia>codigo</columna-referencia>
        </clave>
      </clave-foranea>
      <columnas>
        <columna not-null="false" autoincrement="true">
          <nombre>codigo</nombre>
          <tipo-dato>4</tipo-dato>
          <longitud>10</longitud>
        </columna>
        <columna not-null="false" autoincrement="false">
          <nombre>nombre</nombre>
          <tipo-dato>12</tipo-dato>
          <longitud>50</longitud>
        </columna>
       ...
      </columnas>
    </tabla>
  </tablas>
  <autenticacion habilitado="true">
    <nombre-tabla>usuario</nombre-tabla>
    <columna-login>login</columna-login>
    <columna-password>pass</columna-password>
    <privilegios habilitado="true">
      <columna-insertar>insertar</columna-insertar>
      <columna-modificar>modificar</columna-modificar>
      <columna-eliminar>eliminar</columna-eliminar>
    </privilegios>
  </autenticacion>
  <paginacion>15</paginacion>
```

```
  <apariencia>venezuela</apariencia>
  <destino-lenguaje>
    <ruta>C:\Documents and Settings\Luis Antonio\Escritorio</ruta>
    <archivo>pruebaf.zip</archivo>
    <motor>JSP 2.0 / Servlets</motor>
  </destino-lenguaje>
</DEUSWEB>
```

Figura 63: Documento XML generado.

Pantallas

La siguiente pantalla (figura 66), corresponde al paso 2 de la aplicación DEUSWEB. En este paso, el usuario debe introducir el tipo de Sistema Manejador de Base de Datos que utilizará, la URL de conexión a la base de datos, el nombre de usuario con privilegio de acceso a la base de datos y la contraseña del mismo. Una vez que se han introducido los datos, se presiona el botón "*Probar Conexión*", para verificar la conexión a la base de datos con los datos suministrados. En caso de ser efectiva, permite la selección del botón siguiente, el cual permite avanzar al siguiente paso. Si la conexión no es efectiva, se mostrará un mensaje de error al usuario indicándole el error ocurrido.

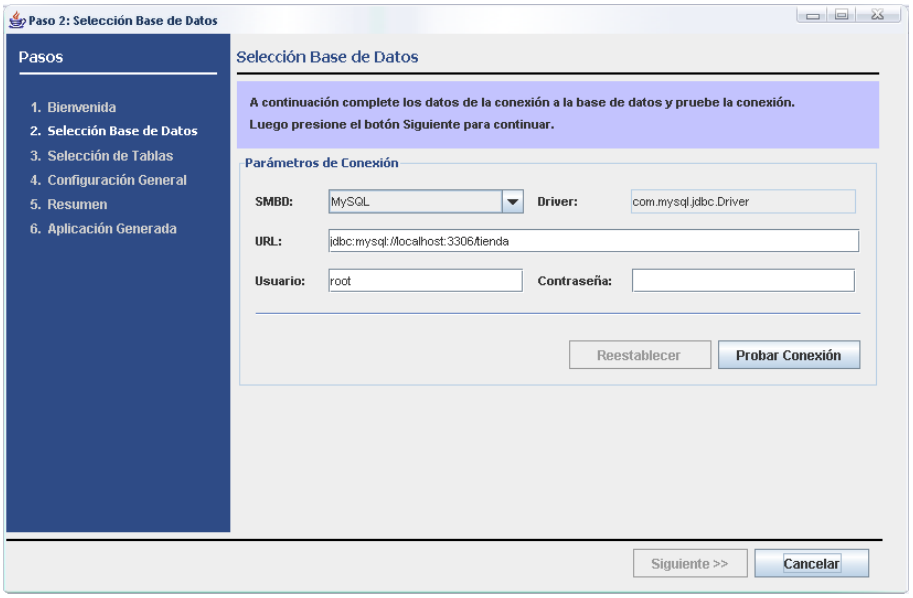


Figura 64: Paso 2.Selección Base de Datos. Aplicación DEUSWEB.

En la siguiente pantalla (figura 67), el usuario debe seleccionar las tablas que desea administrar a través del sistema que será generado.

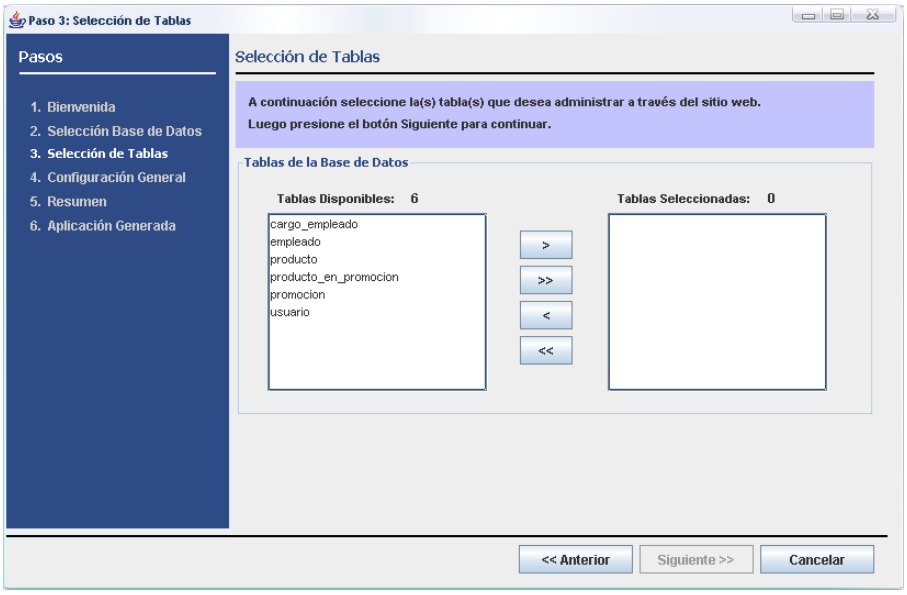


Figura 65: Paso 3. Selección de Tablas. Aplicación DEUSWEB.

En la siguiente pantalla (figura 68), el usuario debe seleccionar los parámetros de configuración del sitio web a generar, tales como: si se va a generar la página de autenticación, cual será la apariencia, cual será el destino, etc.

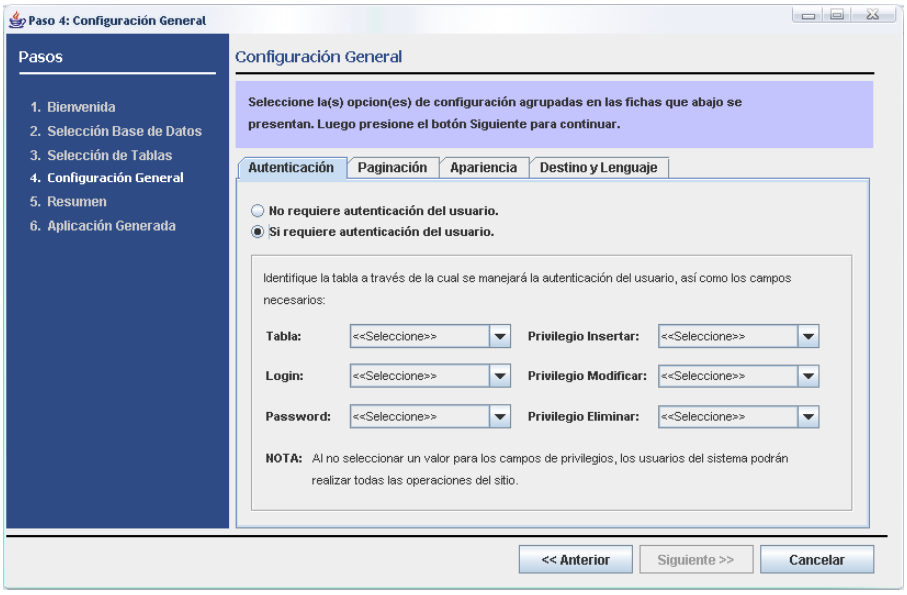


Figura 66: Paso 4. Configuración General del Sitio. Aplicación DEUSWEB.

5.3.2. Flujo de Trabajo: Implementación (Iteración 5)

En esta iteración se producen las clases correspondientes a los procesadores. Antes de mostrar los artefactos producidos en esta iteración, es conveniente señalar algunas consideraciones que fueron tomas en cuenta:

Consideraciones

Para la compilación de las clases generadas para el sitio web administrativo, la compresión de esas mismas clases en formato *jar*, así como la compresión del sitio web completo en formato *war*, se utilizó el *API ant* el cual realiza estas tareas a través de un archivo *build.xml*. A continuación se describe el archivo build.xml utilizado y como el sistema ejecuta cada uno de lo s procesadores del motor.

```
    <project name="general" default="construir">
     <path id="classpath">
           <pathelement location="../../../drivers/jdom.jar"/>
           <pathelement  location="../../../drivers/mysql-connector-java-
3.1.12-bin.jar"/>
           <pathelement location="../../../drivers/cos.jar"/>
           <pathelement location="../../../drivers/servlet-api.jar"/>
     </path>
     <target name="construir">
           <javac srcdir="classes" destdir="classes">
                 <classpath refid="classpath"/>
           </javac>
           <jar destfile="lib/DEUSWEB.jar">
                 <fileset dir="classes" />
           </jar>
           <delete dir="classes" />
           <war destfile="../../<<nombre_sitio>>.war" webxml="web.xml">
                 <fileset dir="../" excludes="WEB-INF/web.xml"/>
           </war>
     </target>
    </project>
```

Figura 67: Documento XML. Archivo build.xml. Aplicación DEUSWEB.

La aplicación DEUSWEB, comprime en un archivo de extensión jar, todos los archivos compilados de la aplicación, así como también los archivos fuentes correspondientes. Con estos archivos fuentes, el usuario puede modificar la aplicación generada.

Descripción de la ejecución de los motores

A través del documento xml generado en la iteración anterior, se extrae el nombre del motor que el usuario seleccionó a través de la interfaz. Este nombre se almacenó en las etiquetas **<destino-lenguaje>** **<motor>** del mencionado documento. El sistema, al extraer el nombre del motor busca los elementos que conforman el motor. Esto lo hace a través del archivo: *motores.xml* también descrito en la iteración anterior.

Ahora el sistema, recorre todos los elementos que constituyen el motor y va ejecutando las clases descritas por cada elemento en la etiqueta **<clase>** junto con los parametros requeridos por la misma en la etiqueta **<parametros>**. La ejecución de estas clases es posible siempre que hereden de la clase **Procesador** (definida por el sistema como una clase abstracta). Esta clase contiene el método: *public abstract boolean Procesar ( )*, a través del cual es posible ejecutar las instrucciones de cada procesador definido. También contiene métodos concretos utilizados como por ejemplo: cargar los parámetros a utilizar por cada procesador.

La ejecución de estos procesadores es llevada a cabo por el sistema a través de una clase interna en la clase controlador llamada: *ThreadCarga* la cual hereda de la clase *Thread*. Fue necesaria la creación de un hilo para la ejecución de esta tarea, para evitar el bloqueo entre la interfaz de usuario y la ejecución de los procesadores, ya que en la interfaz de usuario se muestra una barra de progreso para informar el avance en la generación de los archivos fuentes. La clase *ThreadCarga* contiene el método: *public boolean procesar ( )*, el cual realizar la ejecución de los procesadores. Las instrucciones principales para la ejecución de los procesadores se detallan a continuación:

Por cada procesador:

1. se extrae su nombre y se carga la clase

```
Class C;
   try {
        C = Class.forName(<<nombre_del_procesador>>);
  } catch (ClassNotFoundException ex) {…}
```

2. Se instancia la clase

```
Procesador P;
try {
    P = (Procesador)C.newInstance();
} catch (Exception ex) {… }
```

3. Se cargan los parametros

```
P.setParametros(<<nombre del parametro>>,<<valor del parametro>>);
```

4. Se invoca al método procesar ()

```
P.Procesar():
```

Código fuente de algunos procesadores

Procesador que crea la estructura de directorios:

```
public class ProcesadorCrearDirectorios extends Procesador{
    public ProcesadorCrearDirectorios() {
        super();
    }
    private String[] obtenerDirectorios(){
        String dirs[] = new String [6];
        dirs[0]="/estilo/imagenes";
        dirs[1]="/upload";
        dirs[2]="/WEB-INF/classes";
        dirs[3]="/WEB-INF/jspf";
        dirs[4]="/WEB-INF/lib";
        dirs[5]="/WEB-INF/vistas";
        return dirs;
    }
    public boolean Procesar () {
        // se obtiene el directorio principal
        String   rutaBase   =   (String)   this.getParametros(   (Object)
"directorioBase" );
        String dirPrincipal = this.obtenerNombreSitio();
        File base = new File (rutaBase,dirPrincipal);

        if (rutaBase!=null && rutaBase.length()>0){
            String dirs[] = this.obtenerDirectorios();
            for (int i=0; i<dirs.length; i++){
                File dir = new File (base,dirs[i]);
                if (!dir.mkdirs())
                    return false;
            }
            return true;
        } else
            return false;
    }
}
```

Figura 68: Archivo: ProcesadorCrearDirectorios.java

Procesador que realiza la compilación de los archivos generados, la compresión de las clases en formato jar y la compresión del directorio de desarrollo en formato war:

```
public class ProcesadorCompilarArchivos extends Procesador{
    public ProcesadorCompilarArchivos() {
         super();
    }
    public void compilar (String Destino) throws Exception {
        Launcher.main(new String []{"-buildfile",Destino,"construir"});
    }
    public boolean Procesar () {
        // se obtiene el directorio principal
        String   rutaBase   =   (String)   this.getParametros   ((Object)
"directorioBase");
        String dirPrincipal = this.obtenerNombreSitio();

        // si la ruta existe y no es vacia (no se valida la correctitud)
        if (rutaBase!=null && rutaBase.length()>0){
            String Destino = rutaBase+dirPrincipal+"/WEB-INF/classes";
            try {
                this.compilar(Destino);
                return true;
            } catch (Exception ex) {
                System.out.println("Error Generado: "+ex.getMessage());
                return false;
            }
        } else
            return false;
    }
}
```

Figura 69: Archivo: ProcesadorCompilarArchivos.java

5.4. Resumen del Capítulo

En este capítulo, se describió el proceso de desarrollo implementado en la aplicación DEUSWEB, la cual genera un sistema web administrativo que gestiona la información contenida en una base de datos. Esta aplicación utiliza la ingeniería de reverso al modelo de datos para producir dicho sistema y le permite al

usuario crear una aplicación web administrativa a partir de una base de datos existentes con poca interacción del usuario para producirla (consta de 6 pasos).

En este capítulo se describieron cada uno de los artefactos generados por el sistema en cada una de las iteraciones del proceso de desarrollo.

El siguiente capitulo describe la cuarta etapa del proceso de desarrollo de esta aplicación, que es la etapa de Transición, en la que mayormente domina el flujo de trabajo: prueba. En este próximo capitulo se describe el proceso de generación del sitio web administrativo desde el primer paso de la aplicación generadora (DEUSWEB).

CAPÍTULO VI:

EJEMPLO DE GENERACIÓN CASO DE ESTUDIO

El presente capitulo corresponde a la descripción de la Etapa de Transición de la aplicación generadora. Esta etapa es dominada por el Flujo de Trabajo: Prueba. Este capítulo constituye el ejemplo de generación del sistema DEUSWEB.

6.1. Etapa de Transición de la Aplicación Generadora

6.1.1. Flujo de Trabajo: Prueba (Iteración 6)

El dominio corresponde a una Tienda que vende productos de cualquier tipo, algunos de los cuales puede estar en promoción. Así mismo esta tienda mantiene el control sobre los empleados que laboran: que cargos ocupan, y cuales son sus fotos. Posee una tabla usuarios para la autenticación en el sistema.

Modelo Entidad – Relación

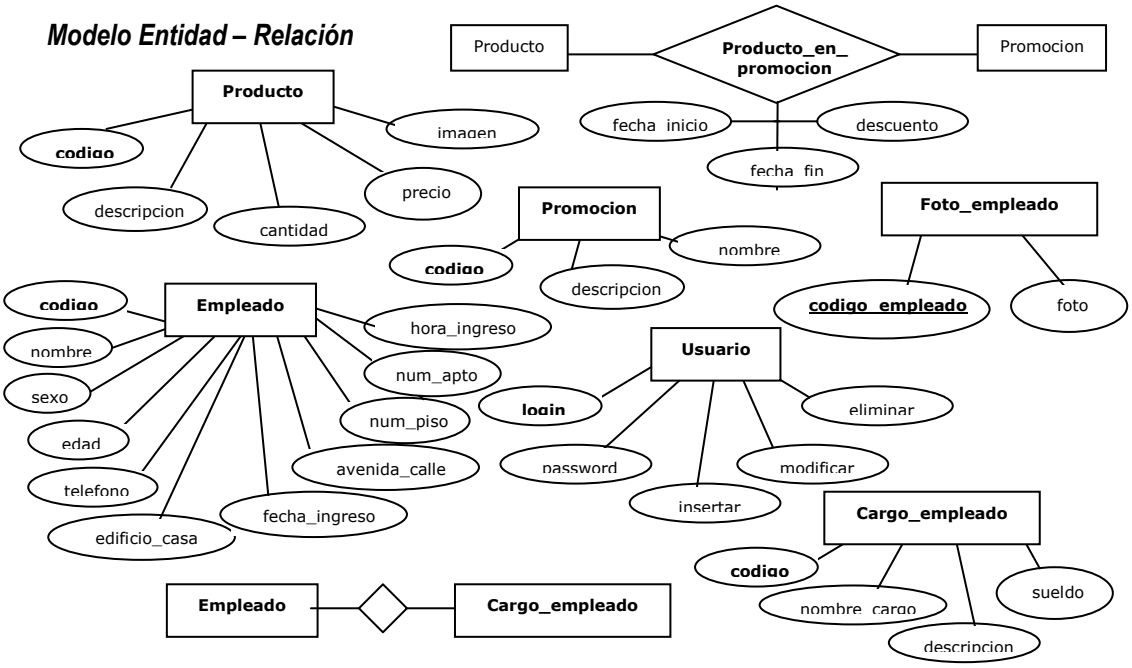


Figura 70. Diagrama E-R. Ejemplo de Generación. Caso de Estudio.

Pantallas (Wizard)

A continuación se muestran dos pantallas de la Aplicación generadora (DEUSWEB). Para la generación del sitio, se escogieron todas las tablas descritas en el Modelo Entidad – Relación, descrito en el punto anterior.

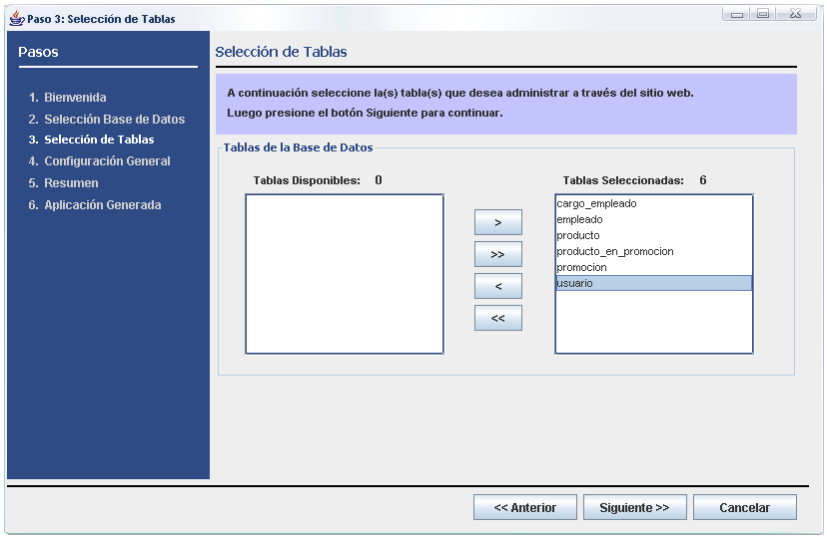


Figura 71. Paso 3. Selección de Tablas. Aplicación Generadora.

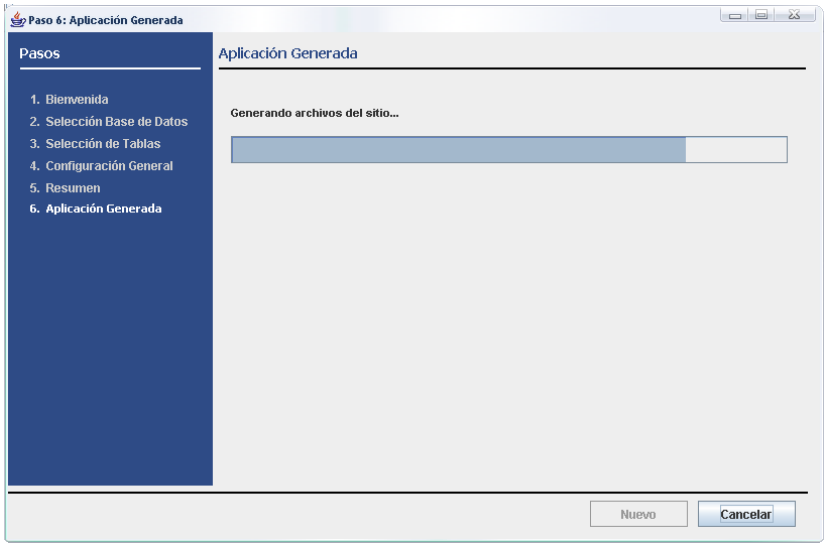


Figura 72. Paso 6. Aplicación Generada. Generando archivos del Sitio.

Carpetas y archivos generados

Luego del paso 6, en la aplicación generadora (ver figura 74), se crea el archivo de extensión *war*, el cual contiene los archivos del sitio generado. Este archivo es colocado en el directorio que el usuario de la aplicación generadora escogió en el paso 4. A continuación se muestran los archivos generados.

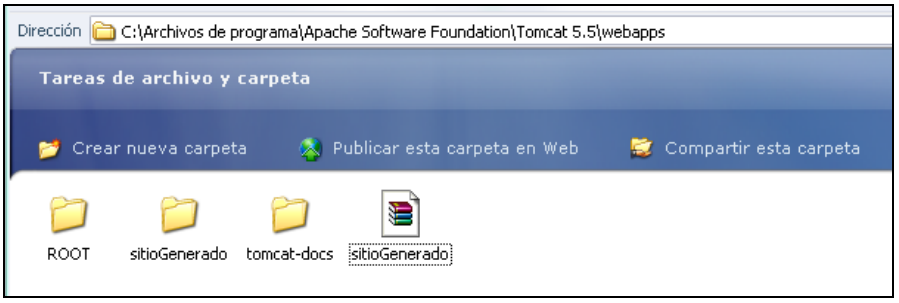


Figura 73. Archivo de extensión *war* generado.

El archivo de extensión war, se descomprime automáticamente, cuando es colocado en el servidor web y la URL de acceso al sitio generado corresponderá a la siguiente estructura:

http://<servidor>/<nombre escogido por el usuario para el sitio>/

La estructura del sitio generado es la siguiente:

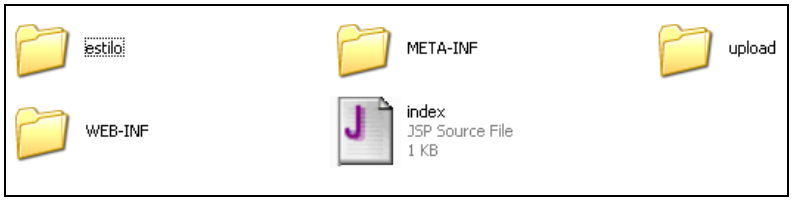


Figura 74. Estructura del sitio generado.

Las páginas del sitio son colocadas en la carpeta WEB-INF, la cual se describe a continuación.

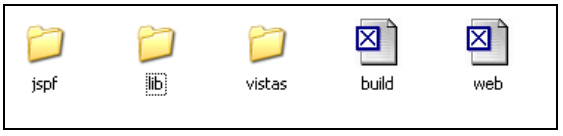


Figura 75. Contenido del Directorio WEB-INF. Aplicación generada.

En el directorio *vistas*, se ubican todos los archivos JSP generados, que corresponden a la interfaz con la cual interactúa el usuario. En el directorio *lib*, se colocan todos los archivos de extensión *jar* utilizados por el sitio. En este directorio se ubicar el archivo *DEUSWEB.jar*, el cual contiene todos los servlets generados para esta aplicación.

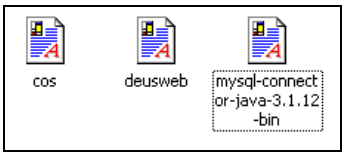


Figura 76. Contenido del Directorio lib. Aplicación Generada.

Puesta en funcionamiento

A continuación se muestran algunas pantallas del sitio web generado.

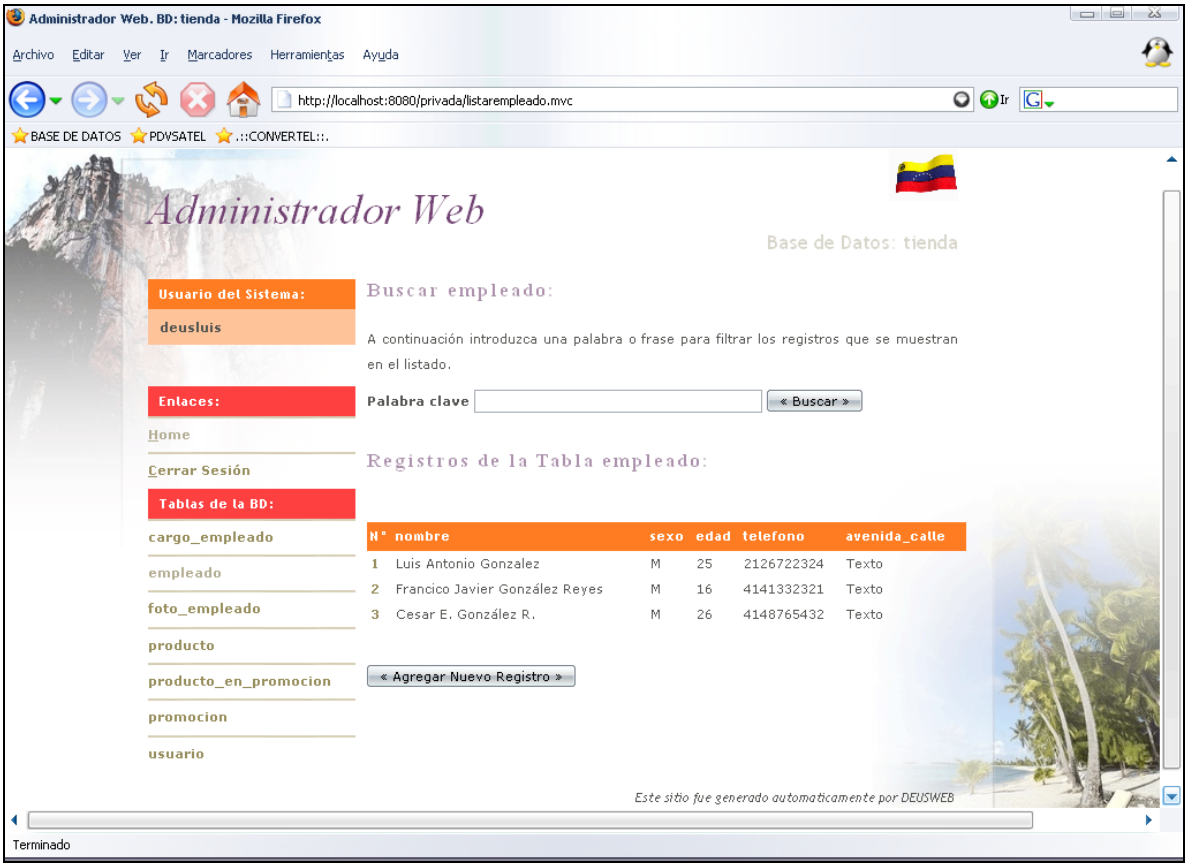


Figura 77. Página que permite Listar el contenido de la Tabla Empleado.

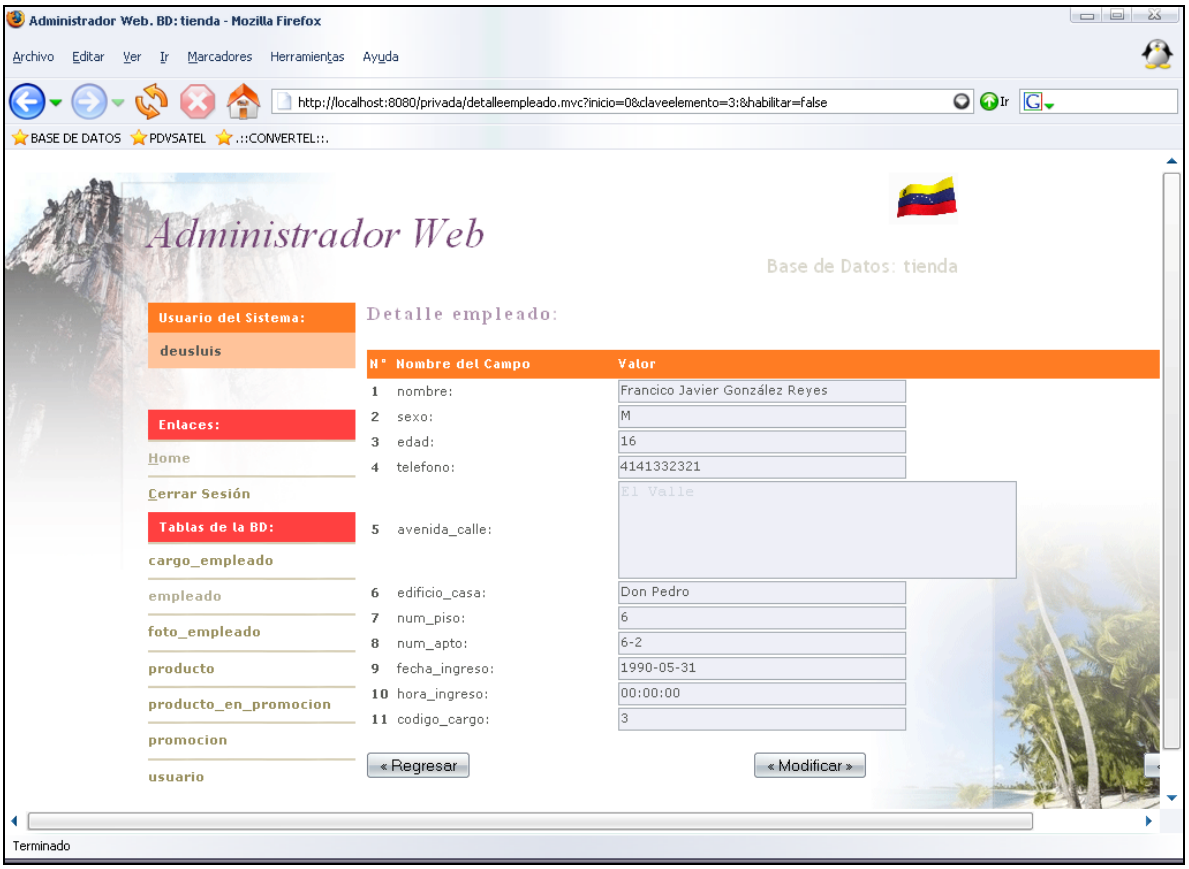


Figura 78. Pagina que muestra el detalle de la Tabla Empleado.

6.2. Resumen del Capítulo

En este capítulo se describió como se genera una aplicación web administrativa a partir del uso de la Aplicación generadora (DEUSWEB). Al inicio del capítulo se describió cual era el caso de estudió que abarcaría la prueba, y posteriormente se mostraron algunas pantallas de la aplicación DEUSWEB para la generación del sitio. Una vez que el sitio fue generado, se describió su estructura, en la que se encuentran las vistas con las que interactúa el usuario y los servlets que controlan la ejecución de las operaciones. Por último se mostraron algunas pantallas de la aplicación web administrativas en ejecución sobre el navegador.

CONCLUSIONES

Dado el auge en el desarrollo de páginas web dinámicas, y dada la importancia de gestionar rápida y eficientemente la información publicada en estos sitios, se ha desarrollado la aplicación DEUSWEB, la cual permite crear aplicaciones web que administran la información contenida en una base de datos.

Hay que tomar en cuenta que la aplicación desarrollada, genera código JSP / Servlets y recibe como entrada una base de datos MySQL. Todas las tablas de la base de datos deben tener definido al menos un campo como clave primaria para el manejo de los archivos que se almacenen en la base de datos, y en caso de utilizar clave foránea, esta debe estar compuesta por un solo campo.

Para la compilación de los archivos generados, se utilizó como herramienta ant, el cual posee un API a través del cual se permite la ejecución de ciertas operaciones tales como: compilar, empaquetar (.jar y .war), eliminar directorios, etc. En principio se intento utilizar el API de ant directamente desde la aplicación DEUSWEB, sin embargo fue muy difícil conseguir los resultados esperados por esta vía. Por esta razón se utilizo el archivo *build.xml* para la ejecución de estas operaciones, invocando al método *main( )* de la clase *Launcher* (propia de Ant), para la ejecución de los comandos configurados a través de este documento xml.

En el desarrollo de la aplicación se encontraron otras funcionalidades que podían ser implementadas para una mejor administración de la base de datos y que no se contemplaban en las tres herramientas existentes evaluadas. Estas funcionalidades fueron desarrolladas y corresponden a: la selección de los valores de una clave foránea y la manipulación de los archivos almacenados en la base de datos.

Para el desarrollo de la aplicación se utilizó el proceso unificado UP, como proceso de desarrollo por su capacidad para desarrollar proyectos de tamaño variable y su rápida implementación.

Los objetivos alcanzados en este trabajo especial de grado fueron los siguientes:

Objetivos logrados:

El desarrollo de un sistema mantenible e incremetable que permite generar aplicaciones web administrativas a partir de la ingeniería de reverso al modelo de datos. El sistema posee la particularidad de poder generar código a través de unos procesadores (clases java) añadidos a la aplicación. Estos procesadores son parte del motor que genera la aplicación web y son configurados desde un archivo *xml* permitiendo así incorporar en próximas versiones nuevas funcionalidades a la aplicación.

Se desarrollo la interfaz gráfica de usuario, que permite la interacción del usuario con la aplicación generadora de forma fácil y amigable. De esta forma, el usuario de la aplicación no tiene por que ser una persona especializada en el desarrollo de un sitio web.

Se estudió la forma de extraer los metadatos de la base de datos, a través de los métodos proporcionado por el API JDBC del Sistema Manejador de Base de Datos. Este proceso fue implementado para desarrollar la aplicación DEUSWEB.

Se verificaron las funcionalidades del sitio web generado, a través de un caso de estudio, el cual permite comprobar el uso del sistema con diferentes tipos de datos en la base de datos. Las funcionalidades básicas del sitio web generado son las siguientes: autenticar al usuario, listar, buscar, insertar modificar y eliminar cualquier información contenida en la base de datos. Así mismo la aplicación web generada posee otras funcionalidades adicionales que son: la selección de los valores correspondientes a la clave foránea de una tabla en particular y la manipulación de los archivos almacenados en la base de datos sin obligar al usuario a definir campos adicionales y sin alterar la estructura de la base de datos suministrada por el usuario.

De los beneficios obtenidos en el desarrollo de esta aplicación tenemos los siguientes:

Beneficios:

- Minimiza el tiempo invertido por las personas o empresas que necesitan desarrollar un sistema web administrativo para la gestión de una base de datos.

- Acelera el desarrollo de aplicaciones web administrativas.

- Puede ser usado por personas que no tengan conocimientos en el desarrollo de aplicaciones web.

- Se pueden incrementar las funcionalidades del sistema, mediante el desarrollo de nuevos motores.

Finalmente se puede decir, que DEUSWEB es un sistema que permite generar aplicaciones web administrativas con las funcionalidades básicas de gestión para el contenido de una base de datos, y con nuevas funcionalidades en comparación con otros sistemas existentes, que le otorga un valor agregado, y que hace posible una mejor administración de la base de datos.

RECOMENDACIONES

Para el sistema DEUSWEB, se proponen las siguientes ideas para trabajos futuros:

- Hacer un instalador de la aplicación, que incluya la instalación del jdk, versión mayor o igual a j2sdk 1.5.0.
- Hacer mayor cantidad de hojas de estilos para enriquecer las posibles vistas de los sistemas web generados.
- Incorporar la vista previa de cómo quedará el sistema web antes de su creación.
- Incorporar el manejo de otros Sistemas Manejadores de Bases de Datos.
- Incorporar más lenguajes de programación a producir por la aplicación, es decir, incorporar la generación de otros tipos de sistemas web, utilizando los motores como el medio para la producción. Pueden generarse sitios web basados en PHP, ASP, etc.
- Optimizar la forma de gestionar los archivos contenidos en la base de datos.
- Optimizar el manejo de las claves foráneas en una tabla.

APORTES A LA INVESTIGACIÓN

El presente trabajo especial de grado puede ser de utilidad para las personas que:

- Desean desarrollar aplicaciones utilizando como base, la ingeniería de reverso al modelo de datos.
- Desean desarrollar / utilizar aplicaciones web que gestionan la información contenida en las tablas de una base de datos.
- Buscan desarrollar aplicaciones utilizando como proceso de desarrollo: UP o Proceso Unificado.
- Buscan generar aplicaciones utilizando código de fácil adaptación a una aplicación.

REFERENCIAS

Aida.info. *Concepto de Usabilidad*.

(http://www.ainda.info/que_es_usabilidad.html)

Beck, K. (1999). *Extreme Programming Explained. Embrace Change*, Pearson Education.

Castro R. *Estructura básica del proceso unificado de desarrollo de software*. Universidad Icesi.

(www.icesi.edu.co/es/publicaciones/publicaciones/contenidos/sistemas_telematica/3/rcastro_estructura-bas-puds.pdf)

Chen, P.P. (1976). *The entity-relationship model-towards a unified view of data*, ACM Transactions on Database Systems, vol. 1, no. 1, pp. 9-36.

Cisterna, M. (2002). *Métodos De Optimización De Consultas Para El Lenguaje SQL.* Santiago de Chile. Capitulo III.

(http://macine.epublish.cl/tesis/index-3_1_.html)

CodeCharge.

(http://www.yessoftware.com/download/download_form.php?product_id=1)

DB2ASPCreator.

(http://www.archivospc.com/programas/archivos/DB2ASPCreator.php)

Programación Extrema.

(http://www.extremeprogramming.org/)

Fernández, F. Muñoz, Y. (2001) *JDBC*. Universidad de Chile.

(www.dcc.uchile.cl/~lmateu/CC60H/Trabajos/jfernand/)

Informática para Ingenieros. *Tema 4: Software Básico*.

(http://webpages.ull.es/users/mbmelian/BD.pdf)

Jacobson, I. Booch, G. Rumbaugh, J. *El Proceso Unificado de Desarrollo de Software.* Madrid: Addison Wesley, 2000.

Letelier P, Penadés M. *"Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP)".* Universidad Politécnica de Valencia.

(www.willydev.net/descargas/masyxp.pdf)

Marqués M. "Apuntes de Ficheros y Base de Datos". Universitat Jaume I.

(www3.uji.es/~mmarques/f47/apun/)

Martinez, A. Martinez, R. *Guía a Rational Unified Process*. Escuela Politécnica Superior de Albacete – Universidad de Castilla la Mancha.

(www.info-ab.uclm.es/asignaturas/42551/trabajosAnteriores/Presentacion-Guia RUP.pdf)

Merchan O. *Cuaderno Docente, Fundamentos de Base de Datos*. Universidad del Azuay.

(www.uazuay.edu.ec/estudios/sistemas/basededatosI.pdf)

Modelo Vista Controlador

(http://es.wikipedia.org/wiki/Modelo_Vista_Controlador)

Molpeceres A. (2002). *Procesos de Desarrollo: RUP, XP y FDD*.

(www.willydev.net/descargas/%20articulos/general/cualxpfddrup.pdf)

Palacio J. *Gestión y procesos en empresas de software*.

(http://www.navegapolis.net/files/articulos/gestion_y_procesos.pdf)

PHP MySQL Wizard.

(http://www.hotlib.com/products1.php?id=25164)

Proceso Unificado de Desarrollo

(http://kybele.escet.urjc.es/documentos/IS/Analisis.pdf)

Sun Microsystems. *JavaTM 2 Platform Standard Edition 5.0 API Specification*.

(http://java.sun.com/j2se/1.5.0/docs/api/index.html)

Tutorial de Acceso a Base de Datos.

(www.programacion.net/bbdd/tutorial/jdbc/)

Tutorial de Java. Capitulo 21.

(www.itapizaco.edu.mx/paginas/JavaTut/froufe/parte21/cap21-4.html)

Web official de RUP.

(http://www.rational.com/products/rup/)

ANEXOS

1. Script de la base de datos (Caso de Estudio)

```
CREATE TABLE producto
(
   codigo INT (10) NOT NULL AUTO_INCREMENT,
   descripcion VARCHAR (50) NOT NULL,
   cantidad INT (6) NOT NULL,
   precio DOUBLE NOT NULL,
   imagen BLOB,
   PRIMARY KEY (codigo)
) ENGINE = INNODB;

CREATE TABLE promocion
(
   codigo INT (10) NOT NULL AUTO_INCREMENT,
   nombre VARCHAR (50) NOT NULL,
   descripcion TEXT,
   PRIMARY KEY (codigo)
) ENGINE = INNODB;

CREATE TABLE producto_en_promocion
(
   codigo_producto INT (10) NOT NULL,
   codigo_promocion INT (10) NOT NULL,
   fecha_inicio DATETIME NOT NULL,
   fecha_fin DATETIME NOT NULL,
   descuento DECIMAL,
   PRIMARY KEY (codigo_producto,codigo_promocion),
   INDEX (codigo_producto),
   INDEX (codigo_promocion),
   FOREIGN KEY (codigo_producto) REFERENCES producto (codigo)
   ON UPDATE CASCADE ON DELETE CASCADE,
   FOREIGN KEY (codigo_promocion) REFERENCES promocion (codigo)
   ON UPDATE CASCADE ON DELETE CASCADE
) ENGINE = INNODB;

CREATE TABLE cargo_empleado
(
   codigo INT (10) NOT NULL AUTO_INCREMENT,
   nombre_cargo VARCHAR (50) NOT NULL,
   descripcion TEXT NOT NULL,
   sueldo FLOAT NOT NULL,
   PRIMARY KEY (codigo),
```

```
   INDEX (codigo)
) ENGINE = INNODB;

CREATE TABLE empleado
(
   codigo INT (10) NOT NULL AUTO_INCREMENT,
   nombre VARCHAR (50) NOT NULL,
   sexo CHAR NOT NULL,
   edad TINYINT NOT NULL,
   telefono BIGINT,
   avenida_calle TEXT,
   edificio_casa VARCHAR (50),
   num_piso VARCHAR (6),
   num_apto VARCHAR (6),
   fecha_ingreso DATE,
   hora_ingreso TIME,
   codigo_cargo INT (10) NOT NULL,
   PRIMARY KEY (codigo),
   INDEX (codigo_cargo),
   FOREIGN KEY (codigo_cargo) REFERENCES cargo_empleado (codigo)
   ON UPDATE CASCADE ON DELETE CASCADE

) ENGINE = INNODB;

CREATE TABLE foto_empleado
(
   codigo_empleado INT (10) NOT NULL,
   foto MEDIUMBLOB,
   PRIMARY KEY (codigo_empleado),
   INDEX (codigo_empleado),
   FOREIGN KEY (codigo_empleado) REFERENCES empleado (codigo)
   ON UPDATE CASCADE ON DELETE CASCADE
) ENGINE = INNODB;

CREATE TABLE usuario
(
   login VARCHAR (20) NOT NULL,
   password VARCHAR (20) NOT NULL,
   insertar BOOL NOT NULL,
   modificar BOOL NOT NULL,
   eliminar BOOL NOT NULL,
   PRIMARY KEY (login)
) ENGINE = INNODB;
```

CONTACTOS

Nombre y Apellido	Dirección de correo electrónico
Luis A. Gonzalez Reyes	luisgreyes@gmail.com
Sergio Rivas	sergiorivas@gmail.com
Johnny Sepúlveda	jsepulve@ciens.ucv.ve
Joselito de Sousa	josedes@gmail.com