ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE INGENIERÍA

GUÍA PARA EL DESARROLLO DE APLICACIONES WEB UTILIZANDO LA TECNOLOGÍA OBJETO RELACIONAL DE ORACLE

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

CABALLERO MORÁN JORGE MARCELO
LEÓN PÁLIZ PABLO CHRISTIAN

DIRECTOR: Ing. Patricio Moreno

Quito, Septiembre 2006

DECLARACIÓN

Nosotros, Caballero Morán Jorge Marcelo y León Páliz Pablo Christian, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Jorge Marcelo Caballero Morán

León Páliz Pablo Christian

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Caballero Morán Jorge Marcelo y León PállZ Pablo Christian, bajo mi supervisión.

_____

Ing. Patricio Moreno

DIRECTOR DE PROYECTO

DEDICATORIA

Este proyecto de titulación está dedicado a las personas que me han ayuda a hacer realidad este sueño, ser profesional y sobre todo persona.

Se lo dedico de manera muy especial a Dios, a mis padres, a mi familia que con su esfuerzo y perseverancia me enseñaron a salir adelante.

Jorge Caballero

DEDICATORIA

Esta tesis está dedicada para mis padres Margarita Páliz y Alejandro León cuyo esfuerzo y sacrificio me han permitido ser la persona quien soy y han sido ejemplo de amor y trabajo, siendo esto el mayor tesoro que me han dado.

Pablo León

AGRADECIMIENTO

Agradezco a Dios por darme la fortaleza necesaria para seguir adelante en los momentos más difíciles de mi vida y saber enfrentar cada uno de los obstáculos que se me presentaron y mostrarme que la vida es bella.

Agradezco a mis padres y hermano por su confianza, apoyo y enseñanza por inculcarme que la perseverancia es la base de todos los éxitos personales.

Jorge Caballero

AGRADECIMIENTO

Agradezco a Dios ya que me supo sujetar en los momentos de debilidad y temor, y me ha permitido alcanzar metas que nunca soñé lograr siendo mi respaldo cada momento de mi8 vida.

Agradezco a mis padres por su apoyo, ya que sin su ayuda no hubiese logrado este difícil reto.

Agradezco a mis hermanos Lucía y David a su cariño y respaldo.

Agradezco a mis hermanos del "Alfa y Omega" y amigos por sus oraciones, por esa frase de aliento cuando más la necesitaba y su amistad.

Pablo León

CONTENIDO

CAPITULO 1 1

FUNDAMENTOS TEÓRICOS.....1

1.1 ESTUDIO DEL MODELO OBJETO-RELACIONAL DE ORACLE.....1

1.1.1 ANTECEDENTES..... 1

1.1.2 DEFINICIONES..... 2

1.1.3 ORACLE Y EL ESTANDAR SQL3 3

1.1.3.1 Tipos Definidos por el Usuario (UDT).....3

1.1.3.2 Reference Type (REF) 7

1.1.3.3 Collection Types 7

1.1.3.4 Herencia 9

1.1.3.5 Constraints para Object Tables.....9

1.1.3.6 Indexación para Object Tables y Tablas Anidadas.....10

1.1.3.7 Triggers para Object Tables 11

1.2 FUNDAMENTOS PARA EL DESARROLLO DE APLICACIONES WEB 11

1.2.1 HISTORIA..... 11

1.2.2 INGENIERÍA WEB..... 13

1.2.3 PROCESO DE LA INGENIERÍA WEB..... 13

1.2.4 PORQUE DESARROLLAR EN WEB..... 15

1.2.5 VENTAJAS WEB 15

1.2.6 DESVENTAJAS WEB..... 15

1.3 ESTUDIO DE TECNOLOGÍAS DE DESARROLLO DE APLICACIONES WEB EN ORACLE.....17

1.3.1 DESIGNER..... 17

1.3.2 ORACLE DEVELOPER SUITE..... 19

1.3.2.1 Oracle Forms.....19

1.3.2.2 Oracle Reports.....21

1.3.3 HTML DB..... 22

1.4 ESTUDIO DE METODOLOGÍAS 23

1.4.1 ANÁLISIS ORIENTADO A OBJETOS 24

1.4.2 DISEÑO ORIENTADO A OBJETOS..... 25

1.4.3 IMPLEMENTACIÓN ORIENTADA A OBJETOS..... 25

CAPITULO 2 26

DEFINICIÓN DE LA GUÍA 26

2.1 DESCRIPCIÓN DE LA METODOLOGÍA A UTILIZAR EN LA GUÍA……29

2.2 ALCANCE DE LA METODOLOGÍA A UTILIZAR EN LA GUÍA……...30

2.3 ETAPAS DE LA METODOLOGÍA A UTILIZAR EN LA GUÍA31

2.3.1 ETAPA 1: FASE DE REQUERIMIENTOS Y ANÁLISIS..... 31

2.3.1.1 Actividades 31
2.3.1.2 Estrategias 31
2.3.2 ETAPA 2: FASE DE DISEÑO 33
2.3.2.1 Actividades 33
2.3.2.2 Estrategias 33
2.3.3 ETAPA 3: FASE DE IMPLEMENTACIÓN Y PRUEBAS 42
2.3.3.1 Actividades 42
2.3.3.2 Estrategias 42

CAPITULO 3 57

APLICACIÓN DE LA GUÍA EN LA CONSTRUCCIÓN DE UNA APLICACIÓN PARA LA ADMINISTRACIÓN DE BIBLIOTECAS 57

3.1 DESCRIPCIÓN DE LOS REQUERIMIENTOS DE LA APLICACIÓN 57
3.2 ARQUITECTURA DE LA APLICACIÓN 59
3.3 DESARROLLO DE LA APLICACIÓN 63
3.3.1 ETAPA 1: FASE DE REQUERIMIENTO Y ANÁLISIS 63
3.3.3.1 Identificación de Metas Globales 64
3.3.3.2 Perfiles de Usuario del Sistema 64
3.3.3.3 Herramientas de Análisis y Diseño de la Aplicación 65
3.3.3.4 Modelo de Casos de Uso 65
3.3.3.5 Diagrama de Clases de Análisis 90
3.3.3.6 Diagrama de Secuencias 92
3.3.2 ETAPA 2: FASE DE DISEÑO 101
3.3.4.1 Diagrama de Clases de Diseño 101
3.3.4.2 Diseño de Navegación del Sistema 103
3.3.4.3 Diseño General de Interfaces de Usuario 108
3.3.4.4 Diseño Arquitectónico de la Aplicación 121
3.3.3 ETAPA 3: FASE DE IMPLEMENTACIÓN Y PRUEBAS 123
3.3.5.1 Caracterización de las Herramientas de Desarrollo 123
3.3.5.2 Implementación del Modelo Objeto Relacional 125
3.3.5.3 Diagrama de Componentes 131
3.3.5.4 Diagrama de Despliegue 131
3.3.5.5 Estándares de Programación 131
3.3.5.6 Pruebas de la Aplicación 134

CAPITULO 4 135

CONCLUSIONES Y RECOMENDACIONES 135

CONCLUSIONES 135
RECOMENDACIONES 136

BIBLIOGRAFÍA 138

ANEXOS.....A

ANEXO A. ESTUDIO DE METODOLOGÍAS ..... A
ANEXO B. PLANTILLA PARA LAS ESPECIFICACIONES DE CASOS DE USO.. Y
ANEXO C. PATRONES DE DISEÑO Y USABILIDAD.....CC
ANEXO D. PLANTILLA PARA LOS CASOS DE PRUEBA ..... NN
ANEXO E. DESCRIPCIÓN DE LAS CLASES, ATRIBUTOS Y MÉTODOS DE LA APLICACIÓN SIABI.....RR
ANEXO F. CÓDIGO FUENTE DEL MODELO OBJETO RELACIONAL DEL SISTEMA SIABI XX
ANEXO G. PRUEBAS DEL SISTEMA.....ZZZ

INDICE DE FIGURAS

FIGURA 1-1: Arquitectura de Oracle Reports Server 22
FIGURA 2-1: Secuencial o Lineal 35
FIGURA 2-2: Reticular 36
FIGURA 2-3: Jerárquica 37
FIGURA 2-4: Web 37
FIGURA 2-5: Mixta 38
FIGURA 2-6: Wap con un Gateway 41
FIGURA 2-7: Transformación de Clases 45
FIGURA 2-8: Transformación de Atributos Multivaluados 46
FIGURA 2-9: Transformación de Atributos Compuestos 47
FIGURA 2-10: Transformación de Atributos Derivados 48
FIGURA 2-11: Generalización 49
FIGURA 2-12: Asociación Uno a Uno 50
FIGURA 2-13: Asociación Uno a Muchos 51
FIGURA 2-14: Asociación Muchos a Muchos 52
FIGURA 2-15: Agregaciones Simples 53
FIGURA 2-16: Composición 54
FIGURA 3-1: Proceso de una Solicitud al Forms Services 60
FIGURA 3-2: Objetos definidos en la Base de Datos 61
FIGURA 3-3: Diagrama de Componentes 62
FIGURA 3-4: Diagrama de Despliegue 63
FIGURA 3-5: Administrar Usuario 67
FIGURA 3-6: Administrar Documento y/o Ejemplar 68
FIGURA 3-7: Administrar Prestamo 69
FIGURA 3-8: Reserva y Visualización de Contenido de Documento 70
FIGURA 3-9: Administrar Idioma 70
FIGURA 3-10: Administrar Autor 71
FIGURA 3-11 Administrar Categoría 71
FIGURA 3-12: Reportes 72
FIGURA 3-13: Diagrama de Clases de Analisis 91
FIGURA 3-14: Diagrama de Clases de Diseño 102
FIGURA 3-15: Diseño de Navegación para el Usuario 104
FIGURA 3-16: Diseño de Navegación para el Bibliotecario Parte I 105
FIGURA 3-17: Diseño de Navegación para el Bibliotecario Parte II 106
FIGURA 3-18: Diseño de Navegación para el Administrador 107
FIGURA 3-19: Interfaz de Consulta de SIABI 108
FIGURA 3-20: Interfaz de Reservación de SIABI 109
FIGURA 3-21: Interfaz de Prestamo de SIABI 110
FIGURA 3-22: Interfaz de Prestamo de SIABI 111
FIGURA 3-23: Interfaz de Pagar Multa de SIABI 112
FIGURA 3-24: Interfaz de Prestamo de SIABI 113
FIGURA 3-25: Interfaz de Reportes de SIABI 114
FIGURA 3-26: Interfaz de Documento de SIABI 115
FIGURA 3-27: Interfaz de Ejemplar de SIABI 116
FIGURA 3-28: Interfaz de Categoria de SIABI 117
FIGURA 3-29: Interfaz de Idioma de SIABI 118
FIGURA 3-30: Interfaz de Autor de SIABI 119
FIGURA 3-31: Interfaz de Usuario de SIABI 120

FIGURA 3-32: Diseño Arquitectonico para el Usuario..... 121
FIGURA 3-33: Diseño Arquitectonico para el Administrador..... 121
FIGURA 3-34: Diseño Arquitectonico para el Bibliotecario..... 122

INDICE DE TABLAS

TABLA 1-1: Ejemplo de Implementación de un Object Table 5
TABLA 1-2: Ejemplo de Implementación de un Object View 5
TABLA 1-3: Ejemplo de Implementación de Distinct Type 6
TABLA 1-4: Ejemplo de Implementación de Cast 6
TABLA 1-5: Ejemplo de Implementación de Row Type 6
TABLA 1-6: Ejemplo de Implementación de REF 7
TABLA 1-7: Ejemplo de Implementación de Varray 8
TABLA 1-8: Ejemplo de Implementación de Tabla Anidada 8
TABLA 1-9: Ejemplo de Implementación de Herencia 9
TABLA 1-10: Ejemplo de Implementación de Constraints dara Object Tables . 10
TABLA 1-11: Ejemplo de Implementación de Indexación 10
TABLA 1-12: Ejemplo de Implementación de Trigger para Object Tables 11
TABLA 2-1: Elementos Básicos de una Guía Técnica 29
TABLA 2-2: Guía para el Diseño del Modelo Objeto Relacional 44
TABLA 3-1: Privilegios Definidos para los Usuarios 64
TABLA 3-2: Herramientas de Análisis y Deseño de la Aplicación 65
TABLA 3-3: Actores del Sistema 65
TABLA 3-4: Notación del diseño de Navegación 103
TABLA 3-5: Requerimientos mínimos de Hardware para cada Herramienta .. 125
TABLA 3-6: Implementación del Object Type Prestamo 125
TABLA 3-7: Estandar de Codificación de Objetos de la Aplicación 134

RESUMEN

La Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto Relacional de Oracle muestra una propuesta para el desarrollo de aplicaciones Web utilizando modelos Objeto Relacional y una alternativa para los desarrolladores que tienen un amplio conocimiento de PL/SQL.

La Guía pone a consideración actividades y estrategias para las fases de requerimiento y análisis, diseño e implementación y pruebas orientados a objetos, así como los diseños de: interfaz de usuario y de navegación (navegación y contenido) siendo este ultimo una particularidad de las aplicaciones Web.

El objetivo principal de este proyecto es desarrollar una guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto Relacional de Oracle que oriente a los desarrolladores en la implementación de objetos en la base de datos así como su manipulación, además se proporcionan lineamientos para gestionar el desarrollo de dichas aplicaciones.

El presente proyecto consta de: un estudio del modelo Objeto Relacional de Oracle, Fundamentos para el desarrollo de aplicaciones Web, un estudio de las tecnologías para el desarrollo de aplicaciones Web en Oracle, la estructura de una guía técnica para el desarrollo de aplicaciones Web, la elaboración de una guia para el desarrollo de aplicaciones Web utilizando la tecnología Objeto Relacional de Oracle, un caso de estudio en donde se muestra como se aplica la guía y las conclusiones y recomendaciones de este proyecto.

PRESENTACIÓN

En esta era de la conectividad global, de Internet, la masificación de los sistemas de información necesita tener formas más eficientes de representar la información en la automatización de los procesos productivos de una empresa por lo que, los sistemas que utilizan los modelos objeto relacionales están en auge.

El presente proyecto de titulación tiene como objetivo principal mostrar el desarrollo de aplicaciones Web utilizando el modelo objeto relacional, para lo cual ésta propone varias actividades y estrategias de desarrollo dentro de cada fase.

La presente guía está dirigida a personas que tienen conocimiento en el desarrollo de aplicaciones Web, el lenguaje de programación PL/SQL y versiones de Bases de Datos Oracle 9i o versiones posteriores y que desean utilizar el modelo Objeto Relacional en sus aplicaciones.

CAPITULO 1
FUNDAMENTOS TEÓRICOS

1.1 ESTUDIO DEL MODELO OBJETO-RELACIONAL DE ORACLE

1.1.1 ANTECEDENTES

Las estructuras del modelo relacional que utilizan los DBMS habían sido suficientes para manejar aplicaciones tradicionales (transaccionales) como las comerciales o de negocios, estas aplicaciones se caracterizan por manejar tipos de datos simples en grandes volúmenes, por lo general alfanuméricos, los cuales son representados con facilidad y precisión en un computador. Actualmente las aplicaciones han evolucionado en lo referente a la manipulación de datos, lo que da lugar a modelar datos complejos que requieren otros paradigmas de los aspectos metodológicos del modelaje conceptual.

En el mercado existen sistemas como los de información geográfica (CAD/CAM), las bases de datos multimediales, entre otras que tiene la necesidad de manipular información como imágenes, video, voz, etc.; tipos de datos que el modelo relacional no lo permite ya que no tiene la estructura y las operaciones adecuadas para manejar estos datos.

En principio se propuso extender el modelo relacional para poder manipular tipos de datos complejos, lo que dio lugar a las base de datos extendidas que proponían mejoras al modelo relacional, mientras que los investigadores de la orientación a objetos definían las bases de datos orientadas a objetos (OODBMS) que por naturaleza son extensibles, con lo cual se puede representar tipos de datos complejos e incluir sus operaciones encapsuladas.

En las bases de datos orientadas a objetos se crean objetos que pueden ser persistentes y que se pueden compartir entre varios programas, es decir que almacenan objetos persistentes en almacenamiento secundario y soportan

compartir objetos entre varias aplicaciones. Los OODBMS juntan a las bases de datos con el paradigma de la orientación a objetos para lo cual se agrega mecanismos manejadores de base de datos como indexación de datos, control de concurrencia y manejo de transacciones.

Por otro lado las bases de datos extensibles evolucionaron a lo que se conoce hoy en día como Bases de Datos Objeto Relacionales (ORDBMS) en la cual se combinan las nociones de extensibilidad, orientación a objetos y en algunos casos hasta tablas anidadas lo que es una alternativa practica y valida al uso de un OODBMS.

A principio de los 90s los enfoques de ORDBMS y OODBMS entraron el conflicto, por una parte los OODBMS puros proponían extensiones a los lenguajes de programación mientras que los ORDBMS proponen un enfoque híbrido que a partir de las bases de datos relacionales incorporan la orientación a objetos, es decir que deben poder manejar objetos, reglas y ser compatible con los RDBMS. En 1996 se logró un consenso en cuanto a las particularidades del modelo objeto relacional, lo cual se plasmo en el estándar SQL-3.

1.1.2 DEFINICIONES [5],[12],[29],[10]

Las Bases de Datos Objeto Relacionales son una evolución de los RDBMS ya que integran los objetos al modelo de datos relacional e integran características del paradigma de la orientación a objetos.

Los ORDBMS utilizan los siguientes tipos de datos:

- Tipos de datos abstractos (TADs) definidos por el usuario.

---

[5] GIETZ William, Oracle9*i* Application Developer's Guide Object-Relational Features Release 2 (9.2), Oracle Corporation, C.A., March 2002.
[12] http://www-etsi2.ugr.es/depar/ccia/mabd/material/teoria/avan4.pdf
[29] http://www.lania.mx/~jalba/bdd/oracle.doc
[10] http://www.bd.cesma.usb.ve/ci5311/sd04/apuntesOR04.pdf

- Tipos de datos complejos (estructurados o agregados) definidos en base a los tipos atómicos con la utilización de constructores para crear: conjuntos, tuplas, arreglos, entre otros.
- Herencia, definición de tipos de datos como subtipo de otros.
- Tipos Referencia, apuntadores a objetos de otro tipo.
- BLOB y CLOB Objetos grandes de tipo binario o carácter.

Para estos tipos de datos se necesitan las siguientes funciones de manipulación:

- Métodos definidos por el usuario, son métodos asociados a los TADs. Los métodos son definidos en el type pero separado del cuerpo del type y se los invoca utilizan la notación punto.
- Operadores, para los tipos de datos complejos (estructurados o agregados) como por ejemplo el constructor conjunto que tiene los métodos: pertenece a, subconjunto de, igualdad de conjuntos, intersección, unión y diferencia de conjuntos entre otros operadores.
- Operadores para los tipos referencia.

1.1.3 ORACLE Y EL ESTANDAR SQL3 [30],[15],[14]

SQL3 se basa en SQL92, con la incorporación de la orientación a objetos. Este estándar provee muchas mejoras sobre SQL92 que no se limitan a la inclusión de los conceptos orientados a objetos sino también de nuevos tipos de datos y predicados. Entre las nuevas estructuras de datos tenemos:

1.1.3.1 Tipos Definidos por el Usuario (UDT)

En esta categoría se encuentran:

- ***Tipo de dato abstracto (TAD).*** Es un tipo abstracto de datos definidos por el usuario, su definición incluye la estructura del type, sus operaciones

[30] http://sales.esicom.com/sales/oracle/appdev.816/a76976/adobjint.htm
[15] http://aixdoc.urz.uni-heidelberg.de/doc_link/en_US/local/oracle.8.1.6/appdev.816/a76976/adobjxmp.htm
[14] http://www.objs.com/x3h7/sql3.htm

como igualdad, ordenamiento y las que definen el comportamiento del TAD. Las operaciones se implementan con procedimientos llamados rutinas.
Para almacenar de manera persistente a un TAD en una base de datos se declara el tipo (type) como columnas de la tabla (table).
Los atributos y las operaciones de un TAD pueden ser públicos o privados, los públicos pueden ser accedidos fuera del TAD.
Las instancias de un TAD se crean con las funciones constructoras del sistema. Se puede utilizar la notación del punto para acceder a las partes de un TAD. Además podemos definir junto con la estructura las funciones apropiadas para manipular las instancias del type definido.

- ***Object Types:*** Un object type es un tipo de dato al cual se lo puede utilizar de varias maneras por ejemplo: se puede especificar un object type como un tipo de dato en una columna, en una tabla relacional o se pueden declarar variables de tipo object para almacenar un valor de ese object type. Un valor de un object type es una instancia de ese tipo.

  Los object type tienen varias diferencias a los demás tipos de datos que son propios de la base de datos. Los object type están compuestos por atributos y métodos:

  - Los Atributos son las características de interés del type object. Por ejemplo un objeto persona puede tener varios atributos tales como nombre, sexo, edad, estado civil, etc.

  - Los Métodos son procedimientos o funciones que permiten al usuario interactuar con los atributos del object.

- ***Object Tables:*** Es una tipo especial de tabla en la cual cada fila representa un objeto. Por ejemplo, la siguiente sentencia crea un objeto idioma y define un object table para cada objeto idioma.

TABLA 1-1: EJEMPLO DE IMPLEMENTACIÓN DE UN OBJECT TABLE

```
create or replace type idioma as object
(idi_id number(10),
idi_nombre varchar2(50),
static function crearidioma(vidiomanom varchar2) return ref
idioma,
member procedure actualizaridioma(vidiomanv varchar2,
vidiomaoriginal varchar2),
static function buscaridioma(vidiomaref in out ref idioma,
vididioma in out number) return varchar2);
/
create table idioma_t of idioma;
```

ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006.

- ***Object View:*** Un object view es una manera de accesar a los datos usando características objeto relacional. Por ejemplo en el siguiente object view cada fila es un objeto de tipo idioma.

TABLA 1-2: EJEMPLO DE IMPLEMENTACIÓN DE UN OBJECT VIEW

```
create table idioma_t (
idi_id number(10),
idi_nombre varchar2(50));

create type idioma as object (
idino number(10),
idi_nombre varchar2(50));

create view idi_view of idioma
with object identifier (idino) as
select e.idi_id, e.idi_nombre
from idioma_t e
where e.idi_nombre = 'español';
```

ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006.

Los object views son útiles para el prototipado o transición a aplicaciones orientadas a objetos porque los datos en la vista pueden ser tomados desde las tablas relacionales y accesar a ellas como si las tablas estuvieran definidas como un object table.

- ***Distinct Types***: Declara tipos de datos diferentes basados en un tipo definido, es decir, que a pesar de que el tipo base sea el mismo, los nuevos tipos declarados son diferentes y las operaciones compatibles o que los

combinan producen un error. Por ejemplo el caso de un número real que tiene parte real e imaginaria:

TABLA 1-3: EJEMPLO DE IMPLEMENTACIÓN DE DISTINCT TYPE

```
create distinct type c_real as number (4);
create distinct type c_imaginario number (4);
```

ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006.

Los dos tipos definidos son iguales pero no se pueden sumar, restar, etc, pues se produce un error, para este caso se puede definir funciones para hacer las conversiones de tipo, con la opción cast.

TABLA 1-4: EJEMPLO DE IMPLEMENTACIÓN DE CAST

```
declare X c_real;
declare Y c_imaginario;
cast (X as number(4)) + cast(Y as number(4));
```

ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006.

- ***Row Type:*** Es utilizado para definir un registro que contiene campos de varios tipos, es decir un tipo compuesto por otros tipos. Por ejemplo:

TABLA 1-5: EJEMPLO DE IMPLEMENTACIÓN DE ROW TYPE

```
create row type direccion
(numero number(3),
 calle varchar2(20),
 ciudad  varchar2(15));

create table direccion_t of type direccion;
```

ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006.

Una vez definido un row type este puede ser utilizado para definir una tabla con filas del tipo row type, de hecho de esta manera se da persistencia al tipo, además esta tabla puede tener más columnas.

Adicionalmente se puede definir funciones que devuelvan valores de tipo row type. En los row type se puede utilizar la notación punto para acceder a elementos del registro compuesto.

1.1.3.2 Reference Type (REF)

Es un tipo especial de datos que se lo utiliza para definir identificadores de objetos, es decir que es un apuntador a un tipo (*type*). Para todo tipo complejo y estructurado se puede definir un REF para referenciar a instancias de ese tipo.

Se puede cambiar un REF para que apunte a un objeto diferente del mismo object type o le asigna un valor nulo.

TABLA 1-6: EJEMPLO DE IMPLEMENTACIÓN DE REF

```
create or replace type multa as object
(mul_id number,
mul_monto number(5, 2),
mul_motivo varchar2(200),
mul_estado varchar2(1),
pre_multa ref prestamo);
/
```

ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006.

1.1.3.3 Collection Types

Oracle soporta dos tipos de colecciones: varrays y tablas anidadas. Las Colecciones pueden ser usadas en otros tipos de datos.

- Un varray es una colección ordenada de elementos donde la posición es identificada por un índice, el cual se lo usa para acceder a dicho elemento. Al definir un varray se debe definir el número máximo de elementos que puede contener. Por ejemplo la siguiente sentencia declara un varray el cual puede definir un número máximo de 10 elementos.

TABLA 1-7: EJEMPLO DE IMPLEMENTACIÓN DE VARRAY

```
create type email as varray(5) of varchar2(30);
```

ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006.

- Una tabla anidada puede tener varios elementos pero en su definición no se especifica el número máximo de elementos. Es posible realizar las mismas operaciones que en las tablas simples. Los elementos de una tabla anidada se almacenan en tablas separadas que contienen una columna que hace referencia a la fila de la tabla padre, por ejemplo la siguiente sentencia define un object type usuarioconsulta en el cual se implementa una tabla anidada la cual contiene referencias del object type *reservacion*.

TABLA 1-8: EJEMPLO DE IMPLEMENTACIÓN DE TABLA ANIDADA

```
--Tipo Colección de referencias al object type
reservacion
create type list_reservacion as table of ref
reservacion;
/

create or replace type reservacion as object
(reser_id number,
reser_fecha date);
/

create or replace type usuarioconsulta as object
(usu_id varchar2(10),
usu_nom varchar2(100),
usu_dir varchar2(200),
usu_telefono varchar2(15),
usu_estado varchar2(1),
res_usuario list_reservacion);
/

create table usuarioconsulta_t of
usuarioconsulta(primary key(usu_id))
nested table res_usuario store as
usuario_reservacion_t;
```

ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006.

El tipo res_usuario se especifica para almacenar las referencias a las reservaciones realizadas por el usuario.

1.1.3.4 Herencia

La herencia es la especialización de un object type mediante la creación de subtipos, se pueden crear subtipos que heredan todas las características de su supertipo más la especialización que se defina en el subtipo; entre las características que se heredan tenemos: atributos, restricciones, disparadores, métodos.

TABLA 1-9: EJEMPLO DE IMPLEMENTACIÓN DE HERENCIA

Ejemplo de Supertipo.

```
create or replace type documento as object
(
doc_id number(6),
doc_titulo varchar2(30),
doc_fecha_ingreso  date,
doc_contenido varchar2 (4000),
doc_año number,
idi_documento ref Idioma,
aut_documento ref Autor,
static function crearDocumento(vdoc_titulo varchar2, vdoc_fecha_ingreso
date,vdoc_contenido varchar2,vdoc_año number,vidi_documento ref
Idioma,vaut_documento ref Autor) return  documento,
member function actualizarDocumento(vdoc documento) return documento) not final;
/
```

Ahora podemos crear el subtipo.

```
create or replace type revista under documento
(
numero_publicacion number(4)
);
```

ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006.

El subtipo creado tendrá los mismos atributos que el tipo documento más el atributo numero_publicacion que se sitúa al final de los anteriores. La sentencia NOT FINAL permite que los atributos y métodos del object type documento puedan ser heredados por el object type revista.

1.1.3.5 Constraints para Object Tables

Se pueden definir constraints en object table así como en otras tablas, a nivel de los atributos de un objeto. Por ejemplo se define un constraint de clave primaria para la columna per_ssno del object table persona_t . Y en la tabla

departamento se definen constraints de restricción que verifiquen integridad de los atributos de localización

TABLA 1-10:EJEMPLO DE IMPLEMENTACIÓN DE CONSTRAINTS PARA OBJECT TABLES[5]

```
create type localizacion (
loc_edif_num number,
loc_ciudad varchar2(40) );

create type persona (
per_ssno number,
per_nombre varchar2(100),
per_direccion varchar2(100),
per_oficina localizacion );

create table persona_t of persona (
per_ssno primary key );
create table departmento (
dep_num char(5) primary key,
dep_nombre char(20),
dep_loc localizacion,
constraint dept_loc
unique(dep_loc.loc_edif_num,
dep_loc.loc_ciudad),
constraint dept_loc1
check (dep_loc.loc_ciudad is not null) );
```

Fuente: GIETZ William, Oracle9*i* Application Developer's Guide Object-Relational Features Release 2 (9.2), Oracle Corporation, C.A., March 2002.

1.1.3.6 Indexación para Object Tables y Tablas Anidadas

Se puede definir índices en Object table o en las tablas de almacenamiento de las columnas de las tablas anidadas o atributos así como en otras tablas. A continuación mostramos un ejemplo de indexación sobre el atributo ciudad:

TABLA 1-11: EJEMPLO DE IMPLEMENTACIÓN DE INDEXACIÓN

```
create type multa as object (
mul_valor number (4,2),
mul_motivo varchar2 (20));

create table usuarioconsulta_t (
usu_id char(5) primary key,
usu_nombre char(20),
usu_multa multa);

create index usu_mul
on usuarioconsulta_t (usu_multa.mul_motivo);
```

ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006.

1.1.3.7 Triggers para Object Tables

Se puede definir triggers en object tables, pero no se los puede definir para las columnas o atributos de las tablas anidadas. A continuación presentamos un trigger de actualización cuando hay un cambio de dirección de una oficina.

TABLA 1-12: EJEMPLO DE IMPLEMENTACIÓN DE TRIGGER PARA OBJECT TABLES

```
create table mudanza (
ssno number,
old_oficina localizacion,
new_oficina localizacion);

create trigger trig_mudanza
before update
of oficina
on person_t
for each row
when new.oficina.loc_ciudad = 'redwood shores'
begin
if :new.oficina.loc_edif_num = 600 then
insert into mudanza (ssno, old_office, new_office)
values (:old.ssno, :old.oficina, :new.oficina);
end if;
end;
```

Fuente: GIETZ William, Oracle9*i* Application Developer's Guide Object-Relational Features Release 2 (9.2), Oracle Corporation, C.A., March 2002.

1.2 FUNDAMENTOS PARA EL DESARROLLO DE APLICACIONES WEB

1.2.1 HISTORIA

El advenimiento del Web y de la Internet ha posibilitado el acceso a la información desde casi cualquier sitio, por lo que las aplicaciones Web han evolucionado a gran escala en lo que a tecnologías se refiere.

A continuación se describe una breve reseña de la evolución de las tecnologías desde sus comienzos hasta la actualidad.

***HTML (hypertext mark-up language)*:** HTML es un lenguaje basado en etiquetas que nos permite representar de una manera sencilla una gran variedad de contenidos así como también referenciar a otros recursos

(imágenes, sonido, etc.), enlaces a otros documentos, pudiendo mostrar formularios para posteriormente procesarlos, etc.

CGI (Common Gateway Interface): Con la llegada de CGI se posibilitó que el contenido de las páginas Web se generen de manera dinámica. CGI es un mecanismo de comunicación entre el servidor WEB y una aplicación externa, esta aplicación puede estar desarrollada en casi cualquier lenguaje.

Una de las desventajas de los CGI es el rendimiento, ya que al momento de realizar una petición al servidor se crea un nuevo proceso, lo que implica un costo muy alto en lo que a recursos del sistema se refiere.

Fast- CGI: Es similar al CGI mencionado anteriormente pero posee una ventaja la cual es la creación de un solo proceso persistente por cada programa Fast-CGI en lugar de por cada solicitud del cliente, teniendo como inconveniente la proliferación de procesos en el caso de peticiones concurrentes.

Páginas Dinámicas en el Servidor: Este avance tecnológico propone un nuevo enfoque al desarrollo de aplicaciones Web, permitiendo insertar fragmentos de lógica de programación en la estructura HTML de la página Web, al contrario de lo que se hacia en los CGIs, que el lenguaje de programación utilizaba sentencias de impresión para generar salidas HTML.

En la actualidad el desarrollo Web ha tenido un gran crecimiento lo que ha dado como resultado una gran diversidad de alternativas tales como jsp, aspx, php, etc.

Servicios Web**:** La arquitectura de servicios WEB plantea algo más que una técnica para el desarrollo de aplicaciones Web, representa un modelo de computación distribuida para Internet. Bajo este concepto ya no solo se trata la comunicación usuario-aplicación, sino que de manera adicional se maneja la interacción aplicación-aplicación.

Los servicios Web son componentes software que presentan las siguientes características distintivas para el programador:

- Son accesibles a través del protocolo SOAP (*Simple Object Access Protocol*) el cual es un protocolo de comunicaciones por paso de mensajes XML que es la base sobre la que sustentan los servicios Web.

- Su interfaz se describe con un documento WSDL (*Web Services Description Language*) que es un estándar de descripción de servicios Web que utiliza para ello un documento XML. Dicho documento proporcionará a las aplicaciones toda la información necesaria para acceder a un servicio.

1.2.2 INGENIERÍA WEB [8]

La Ingeniería Web está relacionada con el establecimiento y utilización de principios científicos, de ingeniería y gestión, y con enfoques sistemáticos y disciplinados del éxito del desarrollo, empleo y mantenimiento de sistemas y aplicaciones basados en Web de alta calidad.

En la actualidad la Ingeniería Web ha adquirido gran importancia y su aplicación esta creciendo en la medida en que las aplicaciones Web se integran con otros sistemas y se requiere construir sistemas fiables, utilizables y adaptables, siendo esta la principal razón por la cual es necesario un enfoque disciplinado para el desarrollo de aplicaciones Web.

El proceso de ingeniería Web comienza con la formulación del problema hasta resolver el problema con la aplicación Web. Se realiza la planificación del proyecto y se analizan los requisitos que implican el desarrollo de la aplicación propuesta, entonces se llevan a cabo el diseño de las interfaces de usuario y la navegación de la aplicación. El sistema se implementa utilizando lenguajes y herramientas especializadas asociadas con la Web, para luego comenzar con las pruebas. Un punto importante que se debe tener en cuenta es el establecimiento de los mecanismos para el control de configuraciones, garantía de calidad y soporte, ya que las aplicaciones Web están en constante evolución.

1.2.3 PROCESO DE LA INGENIERÍA WEB

Este proceso posee características como inmediatez, evolución y crecimiento continuos, los cuales requieren un proceso incremental y evolutivo, que permite

[8] PRESMAN ROGER; Ingeniería de Software, un enfoque práctico; quinta edición; Mc. Graw Hill; 2002

que el usuario se involucre activamente, facilitando el desarrollo de productos que se ajustan a lo que el usuario busca y necesita.

Las actividades que forman parte del proceso son: formulación, planificación, análisis, modelación, generación de páginas, test y evaluación del cliente.

La Formulación identifica las metas, objetivos y establece el alcance de la primera entrega.

La Planificación genera la estimación del coste general del proyecto, la evaluación de riesgos asociados con el esfuerzo y el calendario del desarrollo y fechas de entrega.

El Análisis especifica los requerimientos e identifica el contenido. También se definen los requisitos del diseño gráfico.

La Modelación se compone de dos tareas. Una consiste en el diseño y producción del contenido que forma parte de la aplicación. La otra, en el diseño de la arquitectura, navegación e interfaz de usuario. Es importante tener en cuenta que el diseño de la interfaz es independientemente del valor del contenido y servicios prestados.

En la Generación de páginas se integra contenido, arquitectura, navegación e interfaz para crear de manera estática o dinámicamente el aspecto más visible de la aplicación que son las páginas.

El Test busca errores a todos lo niveles: contenido, funcional, navegacional, rendimiento, etc. El hecho de que las aplicaciones residan en la red, y que interoperen en plataformas muy distintas, hace que el proceso de test sea difícil. Finalmente, el resultado es sometido a la evaluación del cliente.

Las tareas que se proponen para el cumplimiento del proceso serían aplicables a cualquier aplicación W*eb,* independientemente del tamaño y complejidad de la misma.

1.2.4 PORQUE DESARROLLAR EN WEB

En la actualidad existe una gran tendencia a desarrollar aplicaciones en Web, por lo que es importante poder analizar si efectivamente esto es lo más adecuado, pues no todas las aplicaciones son candidatas ideales para ser desarrolladas en Web. Existen factores que determinan cuando una aplicación es más idónea para ser desarrollada en la Web. Por lo cual se analizará las ventajas y desventajas de ésta.

1.2.5 VENTAJAS WEB

El desarrollo de aplicaciones orientadas a la Web nos brinda las siguientes ventajas:

- Usar elementos estándar de interfaz HTML.
- La mayoría de usuarios potenciales tienen familiaridad en navegar por Internet y están acostumbrados con la apariencia de las páginas Web.
- Beneficios de una instalación centralizada, por ejemplo: las actualizaciones sin tener que distribuir un programa ó run-time, es decir, en vez de tener que repartir la aplicación (run-time) a cientos ó miles de máquinas, únicamente colocamos el nuevo software en un sitio y de inmediato todos los usuarios acceden a éste.
- Si en la aplicación a desarrollar hay usuarios remotos, es más fácil usar firewalls corporativos sobre HTTP, que sobre cualquier otro protocolo.
- Algunas aplicaciones requieren bastante poder de procesamiento; en lugar de tener que actualizar individualmente cada estación de trabajo, es mejor aumentar el poder de procesamiento en el servidor, esta será una solución mejor en costos que actualizar varias estaciones de trabajo.

1.2.6 DESVENTAJAS WEB

El desarrollo de aplicaciones orientadas a la Web presenta las siguientes desventajas:

- Los mismos estándares HTML tienen también algunas limitaciones.

- Dependiendo del navegador que se use pueden aparecer diferentes para varios usuarios, esto puede causar alguna confusión si los usuarios se mueven de una estación de trabajo a otra diferente.
- Los controles HTML por que ellos no tienen una forma estándar de proporcionar una máscara de ingreso. Por ejemplo, un número telefónico puede ser ingresado de muchas maneras. En una aplicación Windows, es posible especificar una máscara de ingreso muy fácilmente; en Web es mucho más difícil, hay que usar lenguaje script en el navegador del cliente.
- En la Web se tiene un control limitado sobre el lugar exacto donde aparecerán los elementos en la pantalla. El HTML estándar no permite el posicionamiento absoluto de los elementos. Esto podría hacer el diseño menos elegante que en una aplicación Windows.
- El desempeño puede ser más bajo en una aplicación Web debido a que se están enviando los datos y el diseño de la pantalla cada vez que se pide un documento HTML. En una aplicación Windows, la interfaz es armada localmente solo una vez y únicamente los datos son traídos a través de una conexión de red rápida. De esta manera, no sería muy conveniente tener un aplicativo de atención en ventanilla de gran volumen de transacciones bajo Web.
- Con HTML no es muy fácil realizar la integración con otros productos. Utilizando Microsoft Internet Explorer, es más sencillo integrar que con otros navegadores, pero aun así requiere demasiada programación, Hacerlo de esta manera es casi como crear una aplicación Windows.

1.3 ESTUDIO DE TECNOLOGÍAS DE DESARROLLO DE APLICACIONES WEB EN ORACLE [16][17][26][27][31][32][2]

Oracle posee una variedad de herramientas que son utilizadas para el desarrollo de aplicaciones Web, las mismas que se detallan a continuación.

1.3.1 DESIGNER

Designer es una herramienta de Oracle que posee un módulo llamado el Oracle Designer Web Server Generator con el cual es posible desarrollar aplicaciones Web completamente funcionales para el usuario. Las aplicaciones generadas se basan en la especificación del diseño de la base de datos y los módulos que se almacenan en el Oracle Designer Repository.

Uno de los componentes principales para el proceso de generación es una especificación del diseño de módulos, los cuales son diseñados con Design Editor. Éste registra las tablas y columnas usadas por el módulo, los enlaces entre ellas e información detallada de cómo utiliza los datos el módulo.
Otras entradas del proceso de generación incluyen los enlaces de los módulos y preferencias. Los enlaces de los módulos definen la navegación entre módulos si existen varios módulos en la aplicación que se está desarrollando y se crean usando el Designer Editor. Las preferencias determinan el aspecto y comportamiento general de la aplicación generada; las preferencias pueden ser personalizadas para adaptarse a requerimientos particulares.

Durante la generación el Web Server Generator crea un conjunto de paquetes PL/SQL que son compilados en la base de datos para permitir que la aplicación se ejecute vía Oracle Aplication Server (OAS). Se puede usar cualquier explorador HTML para ejecutar las aplicaciones generadas.

---

[16] http://databasejournal.com/icom_includes/feeds/dbjournal/xml_front-10.xml
[17] http://www.databasejournal.com/RealMedia/ads/click_lx.ads/intm/it
[26] http://www.topxml.com/Biztalk-2006/rn-231519_Connecting-to-Oracle-Forms-Server-9i.aspx
[27] http://sheikyerbouti.developpez.com/webutil-docs/Webutil_store_edit_docs.htm
[31] http://ics02.leidenuniv.nl/902sol_rn/relnotes.902/a92187/toc.htm
[32] http://www.oracle.com/technology/products/ias/daily/nov02.html
[2] BROWN BRADLEY, Oracle 8i, Desarrollo de Soluciones Web, Madrid Mac Graw Hill; 2001

Para la manipulación de datos, las aplicaciones OAS generadas utilizan paquetes API PL/SQL, que ofrecen procedimientos de inserción, actualización, eliminación y bloqueo para cada tabla de la aplicación: Estos paquetes pueden ser generados e instalados una vez finalizado el diseño de las tablas, y antes de generar la aplicación Web.

El Web Server Generator hace un uso completo del Web Developer PL/SQL Toolkit. Web Server Generator ofrece varias ventajas sobre el uso del propio Web Developer Toolkit. La primera ventaja consiste en que Web Server Generator genera una cantidad importante de código automáticamente, esto minimiza el volumen de codificación manual requerido para cada módulo. Web Server Generator genera todos los elementos GUI, tales como tablas, botones, checks y listas emergentes. Otras ventajas incluyen la centralización del código, un repositorio de objetos de base de datos, un aspecto y comportamiento común de la aplicación. Por otra parte, el Web Developer PL/SQL Toolkit ofrece la ventaja de un control completo sobre la apariencia y comportamiento de la aplicación. Cuando se desarrolla de forma manual todo su código usando PL/SQL Toolkit, el desarrollador tiene más flexibilidad en métodos usados para la recuperación y manipulación de datos. PL/SQL Toolkit pierde cierta flexibilidad al generar aplicaciones con otra herramienta, además editar el código generado tampoco resulta una buena opción porque el código generado es complejo y difícil de manipular. La otra desventaja de Web Server Generator es el aprendizaje de esta herramienta.

Las aplicaciones OAS que usan Web Developer PL/SQL Toolkit están constituidas por paquetes de base de datos que producen de forma dinámica páginas Web HTML y JavaScript a partir de peticiones de usuario realizadas desde OAS. Conocer los procedimientos, funciones, estructuras de registros y parámetros usados en estos paquetes permite al desarrollador mejorar las aplicaciones usando varios métodos.

Un tema interesante en cuanto al desarrollo mediante el Web Server Generator es que permite la navegación entre módulos con cookies y PL/SQL, el Web

Server Generator contiene hiperenlaces para navegación basados en enlaces específicos, clave y los módulos. En la mayoría de los casos, estos hiperenlaces ofrecerán una navegación suficiente; sin embargo es posible tener hiperenlaces flexibles y dinámicos utilizando HTML, PL/SQL y cookies. Estas técnicas permiten un control sobre la navegación, y cierto control sobre la posición del hiperenlace en la pantalla.

1.3.2 ORACLE DEVELOPER SUITE

Una de las características más importantes incorporadas en Oracle Developer es su capacidad para ser trasladado a varias plataformas distintas. Actualmente, la plataforma de desarrollo más popular es Internet. Oracle Developer Suite tiene herramientas de desarrollo con lenguaje de 4 generación que son Oracle Forms y Oracle Reports.

1.3.2.1 Oracle Forms

El proceso de ejecutar Oracle Forms en la Web es más complejo que el entorno cliente/servidor. Sin embargo, los resultados de este tipo de implementación son justificados, ya que se puede tomar un formulario para que sea utilizado por los usuarios que acceden a una aplicación cliente/servidor, para luego tomar este mismo formulario y ponerlo a disposición de los usuarios en una intranet, extranet y hasta en el Internet y lo notorio de esto es que no se requieren hacer modificaciones en el código.

Oracle Forms es una herramienta usada para desarrollar pantallas GUI interactivas con robustas prestaciones de validaciones de datos. Con Oracle Forms, se puede construir en poco tiempo un formulario que gestione automáticamente la inserción, actualización y eliminación de los datos mediante SQL.

Para incorporar Oracle Forms a la Web una alternativa consiste en usar HTML, Forms, un lenguaje script (como JavaScript) para la validación a nivel de cliente, y un CGI para manipular los datos de forma adecuada. Otra alternativa sería desarrollar un applet Java que ofrezca esta capacidad, desarrollándolo

con una herramienta como JDeveloper. No obstante, ninguno de estos métodos es tan fácil como crear un formulario con Oracle Forms.

En la implantación de Oracle Forms el cliente solicita un URL, que apunta a una ruta virtual que identifica el Servidor Oracle HTTP, y en este punto, se descarga un applet en el cliente. Este applet Java es capaz de pintar el fondo del formulario, realizar la validación de los datos y comunicarse con el Forms Server. Forms Server es un programa ejecutado en el servidor que lee y comprende un archivo FMX, el Servidor Oracle HTTP determina cómo debe mostrarse el formulario, y envía esta información al applet Java del cliente. A su vez el explorador del cliente genera la página dándole el aspecto que tendría si ejecutara esta aplicación en una arquitectura cliente/servidor. El applet Java del cliente y el Servidor Oracle HTTP mantiene un dialogo abierto mediante TCP/IP, similar a una sesión de usuario cliente/servidor comunicándose con SQL*Net.

Por tanto, podría decir que el Application Server se convierte en lo que tradicionalmente se ha conocido como el cliente en el mundo cliente/servidor.

Para tomar una decisión entre utilizar o no Oracle Forms o HTML Forms, hay algunos aspectos. Oracle Forms trabaja de manera similar a una sesión cliente/servidor, es decir que mantiene un estado (una conexión activa durante la vida de la sesión) en la Web. Una conexión a la base de datos sólo está abierta en cada petición desde el momento en el que se inicia el procedimiento solicitado hasta que termina. No obstante, debe mantener internamente este estado lo cual conlleva a una diferencia significativa en el número de usuario concurrentes de Oracle Forms, así como en el consecuente aumento del número de licencias

FORMS SERVICES: Oracle Forms Services es un servidor de aplicaciones y servicios asociados que se encuentran optimizados para el despliegue de aplicaciones Web desarrolladas con Forms como cualquier otra aplicación que usa el Oracle HTTP Server. Forms Services brinda funcionalidad adicional y

servicios nativos para asegurar que las aplicaciones Forms sean escalables y funcionen sobre cualquier tipo de red.

Form Services está integrado con el Oracle Enterprise Manager (OEM) lo cual permite la administración y monitoreo de Forms Listeners, Balance de Carga para los clientes y servidores

1.3.2.2 Oracle Reports

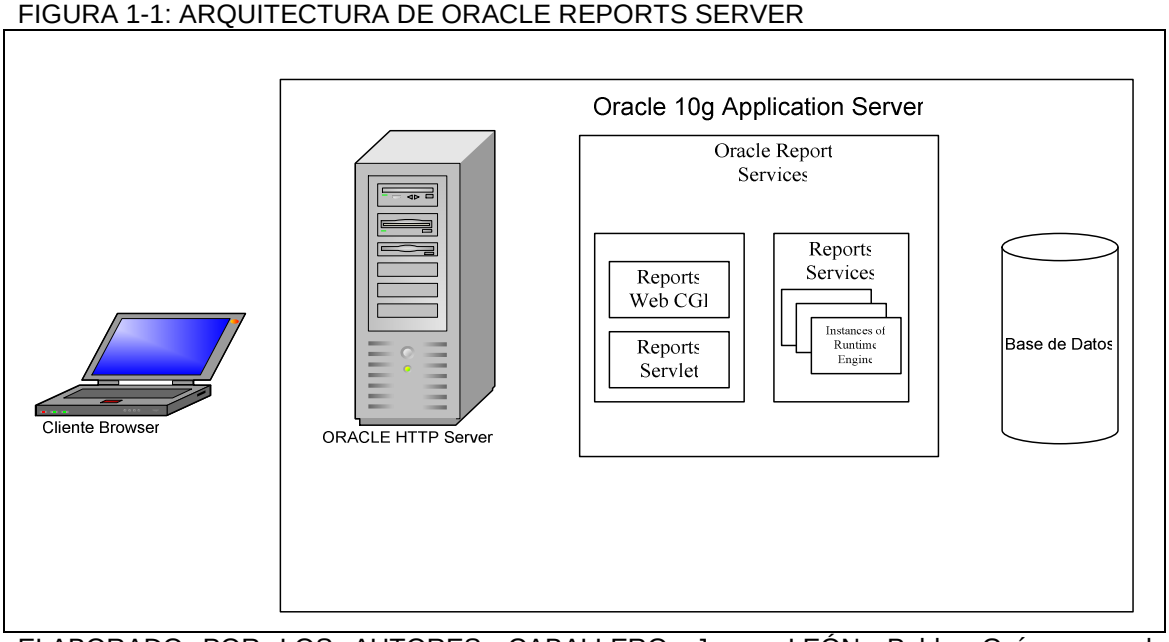
Dadas las características principales de Oracle Forms podría no estar totalmente preparado para iniciarse en el desarrollo para el Internet, mientras que Oracle Reports sí lo está. Oracle Reports permite la generación de archivos para su presentación en la Web en los siguientes formatos:

- HTML para ser mostrado en un explorador Web.
- HTML con hojas de estilo en cascada (HTMLCSS), que puede mostrarse en un explorador Web compatible con HTML.
- PDF que puede mostrarse con un visor PDF individual o un conector, como Adobe Acrobat Reader.

Oracle Reports hace posible generar salidas HTML o PDF sin modificar la definición del informe, a menudo el informe resulta más útil en la Web si se le añade funcionalidad Web (como por ejemplo marcadores y enlaces hipertexto).

El Web Wizard de Reports Builder es la forma más rápida de implantar un informe estático en la Web, además permite agregar funcionalidad Web básica y generar de forma inmediata salida HTML o PDF.
ORACLE REPORTS SERVICES: Permite a los clientes Web ejecutar reportes Oracle desde un simple navegador Web. Oracle Reports Services soporta la publicación en formatos HTML y PDF. En la FIGURA 1-1 se muestra la Arquitectura de Oracle Reports Server

FIGURA 1-1: ARQUITECTURA DE ORACLE REPORTS SERVER



ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006.

1.3.3 HTML DB

Oracle HTML DB es un entorno para desarrollo y despliegue de aplicaciones Web centradas en la base de datos con un servicio compartido mediante la habilitación de múltiples grupos de trabajo para la construcción y acceso a aplicaciones como si estas estuvieran corriendo en bases de datos separadas, gracias a las utilidades que brinda, tales como: temas de diseño, controles para navegación, manejadores de formas y reportes flexibles. Oracle HTML BD acelera el proceso del desarrollo de aplicaciones.

La plataforma de desarrollo Oracle HTML DB consiste de los siguientes componentes

- Application Builder
- SQL Workshop
- Administration

***Application Builder*:** El Application Builder se utiliza para ensamblar una interfaz HTML (aplicación) con objetos de la base de datos tales como tablas y procedimientos.

Una vez que se crea una aplicación, el motor HTML DB muestra la aplicación usando las plantillas y elementos de la interfaz de usuario que se haya especificado.

Una página es el bloque básico de construcción de una aplicación. Cada página puede tener botones, campos y adicionalmente puede incluir lógica de la aplicación.

SQL Workshop: Este componente permite ver y manejar los objetos de la base de datos desde un Web browser.

HTML DB Administration: En el ambiente de desarrollo Oracle HTML DB, los desarrolladores se registran dentro de un área compartida denominada workspace. Los usuarios son divididos en dos roles: Desarrollador y Administrador del Workspace.

Los Desarrolladores pueden crear y editar aplicaciones así como reportes.

El Administrador del Workspace usa el Workspace de HTML DB para crear y editar cuentas de usuarios, manejar grupos, y manejar servicios de desarrollo.

1.4 ESTUDIO DE METODOLOGÍAS [6],[19],[7],[35]

En las dos últimas décadas las notaciones de modelado y posteriormente las herramientas pretendieron ser la solución para el desarrollo de software, sin embargo, las expectativas no fueron satisfechas. Esto se debe en gran parte a que otro importante elemento, la metodología de desarrollo, no había sido tomado en cuenta, pues de nada sirven buenas notaciones y herramientas si no se proveen lineamientos para su aplicación.

---

[6] Larman Craig, UML y Patrones, Prentice Hall, 2ª Edición, 2004
[19] http://kybele.escet.urjc.es/documentos/BD/T4-DisenyoBDOR_Alumnos.pdf
[7] Marcos E., Vela B., Cavero J. M. and Caceres P., Aggregation and composition in object-relational database design. Advanced Databases and Information Systems, September 2001
[19] http://www.willydev.net/descargas/articulos/general/umlTotal.pdf, 2004

Existen varios tipos de metodologías entre las cuales podemos destacar las metodologías orientadas a objetos y las metodologías ágiles. En el Anexo A se incluye un estudio donde se analiza los aspectos relevantes de algunas de las metodologías más importantes tanto orientadas a objetos como ágiles. Entre las cuales tenemos:

Metodologías Orientadas a Objetos

- RUP (Rational Unified Process)
- OOHDM (Método de Diseño Hipermedia Orientado a Objetos)
- Metodología de Análisis y Diseño Orientado a Objetos
- Metodología basada en Agregación y Composición

Metodologías Ágiles

- MSF (Microsoft Solution Framework)
- XP (Extreme Programming)

Adicionalmente, es necesario una metodología para el diseño del modelo Objeto-Relacional y al realizar el estudio respectivo se encontró como una propuesta consistente la metodología basada en agregación y composición.

Por tanto, para la elaboración de la guía desarrollada en este trabajo se utiliza como base la metodología de análisis y diseño orientado a objetos y la metodología para el diseño de base de datos Objeto Relacional.

Las siguientes subsecciones contienen una breve descripción de los elementos de la metodología orientada a objetos que deberán considerarse en la guía.

1.4.1 ANÁLISIS ORIENTADO A OBJETOS

El análisis orientado a objetos pone énfasis en la investigación del problema y los requerimientos, es decir, presta especial atención en encontrar y descubrir los objetos o conceptos en el dominio del problema. El análisis orientado a objetos identifica las clases (atributos y comportamiento) así como las relaciones entre éstas y su posterior implementación en un leguaje de programación.

1.4.2 DISEÑO ORIENTADO A OBJETOS

El diseño orientado a objetos pone énfasis en una solución conceptual que satisface los requerimientos de la aplicación, es decir, la definición de los objetos de software y como ellos colaboran para satisfaces los requerimientos de la aplicación.

En el diseño orientado a objetos existen varias etapas a saber que son:

- Entendimiento del Problema
- Identificar una o varias posibles soluciones
- Describir abstracciones de la solución de manera que el diseño este expresado en forma sencilla.

Entre las características principales del diseño orientado a objetos tenemos:

- Los objetos son abstracciones del mundo real o entidades del sistema.
- Los objetos son independientes y encapsulan el estado y la representación de la información
- Los objetos se comunican mediante el paso de parámetros.
- La funcionalidad del sistema utiliza los métodos de los objetos.

Entre los componentes del diseño orientado a objeto tenemos:

- Objetos, sus atributos y métodos
- La organización de los objetos dentro de una jerarquía.
- La especificación de interfaces de objetos

1.4.3 IMPLEMENTACIÓN ORIENTADA A OBJETOS

En la implementación orientada a objetos los objetos de diseño son implementados utilizando un lenguaje de programación.

CAPITULO 2

DEFINICIÓN DE LA GUÍA

INTRODUCCIÓN

Para elaborar una guía técnica existe una diversidad de metodologías, las mismas que pueden adaptarse dependiendo del tema a tratar y la profundidad del mismo, ya que una guía técnica depende del uso y las condiciones para las que fue diseñada, sin embargo sus características y funciones básicas deben permanecer invariables.

CRITERIOS PARA LA ELABORACIÓN DE UNA GUÍA TÉCNICA [9]

Para elaborar una guía se debe tener en cuenta los siguientes criterios:

- El material debe estar dirigido para las personas interesadas (ejecutores) y debe contener temas relacionados a los objetivos de la guía.
- La guía debe plantear objetivos y alcances generales, para que el o los ejecutores puedan evaluar los posibles resultados erróneos en la ejecución de los mismos.
- Si la guía posee actividades técnicas, estas deben seguir los pasos del método científico-tecnológico: Tareas por resolver, buscar y analizar maneras de resolverlas. Y finalmente, se opta por una respuesta y se dejan, en claro, por escrito los avances obtenidos
- La guía debe contener un esquema de actualización y verificación de los contenidos de la misma.
- La guía debe contener tareas que impliquen una puesta a prueba de los conocimientos por medio de acciones, ya que no hay materiales puramente cognitivos o puramente activos o valorativos

[9] Fuente: LOYOLA, Jaime. VALLEJO, Carlos, Vulnerabilidades en las redes de información y técnicas para enfrentarlas, Quito, 2005.

- Asegurar que la guía integre procedimientos, de modo que permita diferentes formas de realización, flexibles con distintos antecedentes y experiencias para que puedan hacer uso de acuerdo a la situación.
- La guía debe poseer información actualizada y relacionada con sus antecedentes para facilitar su entendimiento.
- Cerciorarse que la guía posea un número de acciones y de respuestas adecuada para la correcta implantación de la misma.
- Un aspecto importante que se debe tomar en cuenta al momento de elaborar una guía, es que esta deberá ser lo suficientemente flexible para ser aplicada a los diversos ambientes de ejecución y a los posibles ejecutores, conservando la independencia del entorno sobre el que va a utilizarse
- Proveer actividades para aquellos usuarios que encuentran dificultades para entender y comprender los aspectos técnicos de la guía.
- Facilitar la bibliografía para acceder a otras fuentes de conocimiento: textos, Internet, y revistas.

CARACTERÍSTICAS Y FUNCIONES DE UNA GUÍA TÉCNICA

Una guía técnica debe tener las siguientes características:

- Proporcionar información sobre el tema para la cual fue elaborada la guía.
- Proporcionar orientación del tema.
- Establecer objetivos y actividades.

Una guía técnica debe tener las siguientes funciones:

- Dilucidar en el desarrollo las dudas que puedan obstaculizar el normal progreso de la investigación.
- Establecer recomendaciones para orientar el trabajo de los lectores.
- Establecer actividades integradas de aprendizaje en el lector, para así asegurar o garantizar la aplicación de la guía.

- Proponer prácticas para que el lector evalúe su progreso y lo motive a compensar sus deficiencias mediante investigaciones posteriores.

DISEÑO DE UNA GUÍA TÉCNICA

Las principales directrices a tener en cuenta al diseñar una guía técnica son el fondo y la forma.

El fondo está relacionado con:

- El conocimiento o la experiencia que se desea proponer a los lectores.
- La transformación didáctica que ese conocimiento o experiencia requieren para que se aumenten las probabilidades de éxito de quienes la usen.

La forma se refiere:

- La estructura o secuencia que se utilizará para expresar esa estrategia de enseñanza.
- Los aspectos de presentación como lenguaje, tipografía, diagramación, etc.

ESTRUCTURA DE UNA GUÍA TÉCNICA

En la Tabla 2-1 se muestran los elementos básicos que deben constar en una guía técnica

TABLA 2-1: ELEMENTOS BÁSICOS DE UNA GUÍA TÉCNICA [9]

ELEMENTOS	DESCRIPCIÓN
Titulo	Describe el propósito de la guía en forma sintética o metafórica
Contenido	Muestra en forma ordenada el contenido del trabajo
Presentación	Permite al autor expresar el propósito general de la guía, orientar la lectura y hacer consideraciones previas para una mejor comprensión de los contenidos
Aplicación	Identificar los ámbitos en los que puede ser aplicada la guía
Objetivos	Le dice al lector que se espera que aprenda o haga como resultado luego de realizar las actividades que se proponen en la guía.
Alcance	Se definen los puntos que abarca la guía
Definiciones básicas	Resume los conceptos que introduce la guía
Resumen de Contenidos	Presenta en forma concisa todos los puntos fundamentales de los que consta la guía, para así facilitar su comprensión.
Desarrollo de Contenidos	Presenta de manera general la temática de estudio, especificando su campo de acción al que pertenece, destacando el valor y la utilidad que ofrece.
Temática de estudio	Presenta en forma clara, ordenada y exacta los temas y subtemas correspondientes al contenido que se presenta.
Evaluación	La guía propone formas de evaluar el progreso y el logro del o de los objetivos para los que fue diseñada. Mantiene informado al lector de su progreso.
Bibliografía	En la medida de lo posible la guía tiene la información necesaria para cada actividad o muestra cómo encontrarla. En caso que se requiera información mas completa o se desee hacer lecturas complementarias esas se pueden agregar al final.

2.1 DESCRIPCIÓN DE LA METODOLOGÍA A UTILIZAR EN LA GUÍA

La presente guía tiene como objetivo proporcionar lineamientos para el desarrollo de aplicaciones Web que utilice un modelo objeto relacional implementado en la base de datos Oracle 10g.

[9] Fuente: LOYOLA, Jaime. VALLEJO, Carlos, Vulnerabilidades en las redes de información y técnicas para enfrentarlas, Quito, 2005.

De acuerdo con el estudio de metodologías realizado en la sección 1.4 y el **Anexo A**, la presente sección muestra la estructura a utilizar en la guía, que consiste en:

- Actividades recomendadas para el desarrollo de aplicaciones Web que implementen objetos en la base de datos Oracle 10g.

- Estrategias asociadas a las actividades recomendadas que detallan los pasos necesarios para obtener cada una de las actividades.

Los fundamentos metodologícos utilizados en la guía se detallan en la descripción de cada estrategia. Entre los más importantes podemos mencionar:

- Lineamientos para implementar aplicaciones con el modelo objeto relacional los cuales son aplicables en plataformas Oracle10g.

- Estándares UML para el Análisis, Diseño e Implementación Orientado a Objetos incluyendo conceptos de Ingeniería Web.

- Las actividades seguirán los pasos del método científico tecnológico: hay una tarea por resolver, se buscan y analizan maneras de enfrentarlas. Finalmente, se opta por una respuesta y se dejan, en claro, por escrito los avances alcanzados.

2.2 ALCANCE DE LA METODOLOGÍA A UTILIZAR EN LA GUÍA

La presente guía muestra como gestionar el análisis, diseño e implementación de aplicaciones Web utilizando las ventajas del desarrollo orientado a objetos.

Se proporcionarán actividades, estrategias que detallen las principales etapas que permitan describir cada fase del desarrollo de software propuesta en la guía, ente las cuales tenemos:

- Etapa 1: FASE DE REQUERIMIENTOS Y ANÁLISIS.
- Etapa 2: FASE DE DISEÑO.
- Etapa 3: FASE DE IMPLEMENTACIÓN Y PRUEBAS.

La siguiente sección detalla las etapas de la guía. El documento completo que contiene la guía se encuentra en el **Anexo H**.

2.3 ETAPAS DE LA METODOLOGÍA A UTILIZAR EN LA GUÍA

2.3.1 ETAPA 1: FASE DE REQUERIMIENTOS Y ANÁLISIS

2.3.1.1 Actividades

La fase de análisis orientada a objetos (AOO) es la primera actividad técnica que se desarrolla como parte de la ingeniería de software Orientado a Objetos.

El propósito del AOO es definir todas las clases que son relevantes al problema que se va a resolver, las operaciones y atributos asociados, las relaciones y comportamientos asociadas con ellas. Para cumplirlo se deben ejecutar las siguientes tareas:

- Definir los requerimientos básicos de la aplicación los cuales deben comunicarse entre el cliente y el ingeniero de software mediante el modelo de Casos de Uso.

- Identificar las clases y asociaciones entre clases mediante el diagrama de clases de análisis.

- Modelar el comportamiento de la Aplicación.

2.3.1.2 Estrategias

Para la definición de los requerimientos del usuario se recomienda seguir los siguientes pasos:

- Para la toma de requerimientos se realizan entrevistas, cuestionarios, lluvia de ideas y prototipos.

- Al desarrollar una aplicación Web se recomienda identificar las metas globales para lo cual se sugiere responder las siguientes preguntas

    - ¿Cuál es la motivación principal para la Aplicación Web?
    - ¿Por qué es necesaria la Aplicación Web?
    - ¿Quién va a utilizar la Aplicación Web?

  Adicionalmente se pueden identificar metas específicas para el sitio Web, de las cuales se identifican dos categorías:

    - Metas Informativas: Proporcionan el contenido y/o información específicos para el usuario final.
    - Metas Aplicables: Indica las tareas que la Aplicación Web realiza.

- Se recomienda desarrollar el perfil del usuario en el cual se recoge las características relevantes de los usuarios potenciales incluyendo antecedentes, conocimientos, preferencias.

- Definir las herramientas que se van a utilizar para el análisis y diseño de la Aplicación.

- Se realizan los diagramas de casos de uso, los cuales son una descripción de escenarios de como interactúan los actores (gente, máquinas u otros sistemas) con el sistema a construir, los mismos que permiten modelar el sistema desde el punto de vista del usuario.

- Se detallan los requisitos funcionales los cuales describen como funcionará el sistema mediante las Especificaciones de Casos de Uso. Debe escribirse una especificación por cada caso de uso. Una posible plantilla para las especificaciones funcionales se incluye en el **Anexo B**.

Se identifican las clases y relaciones entre las clases para lo cual:

- Se realiza el modelo de clases de análisis en el cual se incluyen las relaciones entre las clases, el cual puede ser un solo diagrama grande o varios pequeños por cada caso de uso.

Modelar el comportamiento de la aplicación mediante:

- Diagramas de Secuencias (muestra los eventos del sistema para un escenario de un caso de uso) para los cuales debe crearse mínimo uno por cada caso de uso.

2.3.2 ETAPA 2: FASE DE DISEÑO

2.3.2.1 Actividades

En la fase de Diseño se crea una solución a nivel lógico para satisfacer los requisitos, basándose en la información reunida en la fase de Análisis para lo cual se recomiendan las siguientes actividades.

- La identificación de clases, sus atributos y operaciones.

- La especificación de interfaces

- Realizar el diseño arquitectónico de la aplicación.

2.3.2.2 Estrategias

Para realizar la identificación de objetos, sus atributos y operaciones se propone realizar:

- Diagramas de Clases de Diseño en el cual se debe incluir atributos y operaciones de todas las clases, e incluir clases de implementación.

  Si fuera el caso de la aplicación se podría hacer uso de patrones al realizar el diseño de la aplicación para lo cual se adjunta en el **ANEXO C**

tipos de patrones que pueden ser de ayuda para el desarrollo de una Aplicación Web utilizando el diseño orientado a objetos.

Para realizar el diseño de las interfaces se propone las siguientes estrategias

- Realizar el Diseño de Navegación del Sistema. En el cual se muestra los links.

- Diseño General de Interfaces de Usuario en el cual se debe incluir los bosquejos de las principales interfaces.

  En el diseño de las interfaces se deben realizar las siguientes consideraciones:

  - Los errores de la aplicación por pequeños que sean pueden causar que el usuario salga de la Aplicación Web y busque un sitio alternativo
  - La velocidad de lectura en el monitor es aproximadamente 25% más lento que en documentos impresos.
  - Evitar utilizar páginas sin funcionalidad (“Páginas en Construcción“).
  - Los usuarios prefieren no tener que utilizar el scroll para leer el contenido de una página
  - Deben diseñarse menús de navegación y títulos de manera consistente y estos deben estar disponibles en todas las páginas que el usuario pueda visitar.
  - No se debe confiar en las funciones del navegador para ayudar en la navegación para lo cual se recomienda realizar las pruebas correspondientes en los diferentes navegadores disponibles en el mercado.
  - La estética nunca debe estar encima de la funcionalidad de la aplicación.
  - La navegación debe ser obvia a los usuarios casuales.

El Diseño Arquitectónico de la Aplicación requiere hacer ciertas consideraciones para lo cual se detalla los puntos principales a tomar en cuenta.

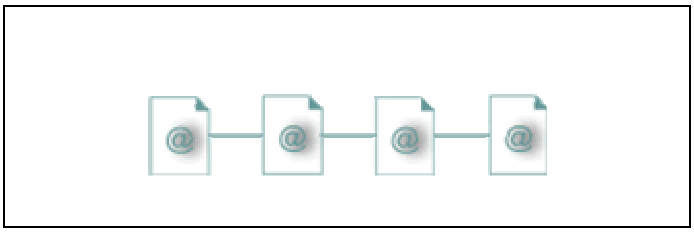
En una aplicación Web el diseño arquitectónico de la aplicación se centra en la estructura global y en la aplicación de patrones de diseño, la cual se realiza paralelamente con el diseño del contenido.

Una buena estructura permitirá al lector visualizar todos los contenidos de una manera fácil y clara, mientras que un conjunto de páginas Web con una mala estructura producirá en el usuario una sensación de no saber donde se encuentra, no podrá encontrar rápidamente lo que busca por lo cual es necesario planificar la estructura de la Aplicación Web. Incluso para los sitios más pequeños, es conveniente hacer algún esquema sencillo, para la mayoría de los casos una hoja de papel y un lapicero bastará, pero si el nivel de complejidad es mayor, es recomendable utilizar programas que permita manejar estructuras de tipo grafo.

Tipos de Estructuras

1. Secuencial o lineal: Es la forma más simple de organizar la información. La información sigue un flujo natural como una narrativa lineal o con un orden lógico

FIGURA 2-1: SECUENCIAL O LINEAL



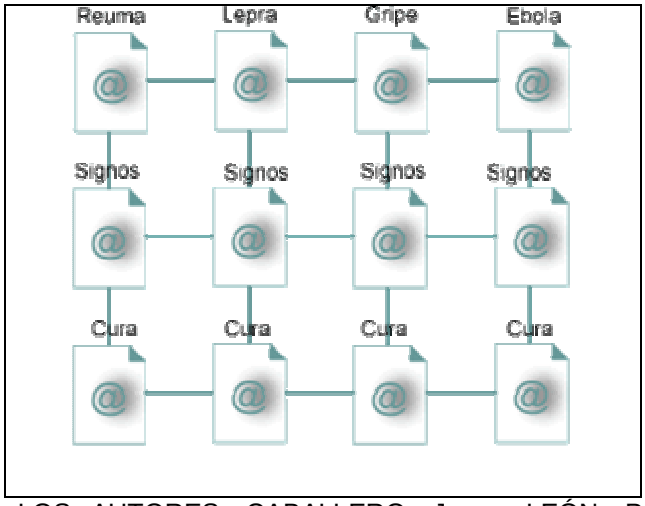
ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006.

La estructura secuencial puede ser: cronológica, una serie de temas ordenados lógicamente de lo más general a lo más específico o incluso ordenado alfabéticamente.

Este tipo de organización sólo suele funcionar para sitios pequeños; a medida que la narrativa se torna más compleja, necesita una mayor estructuración para seguir siendo compresible.

2. Reticular: Muchos manuales de procedimientos, listados de cursos universitarios, descripciones de casos médicos se adaptan a este tipo de estructura.

FIGURA 2-2: RETICULAR



ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006.

Las estructuras reticulares son una buena forma de asociar conceptos, en una serie de categorías estándar como "eventos", "tecnología", "cultura",etc.

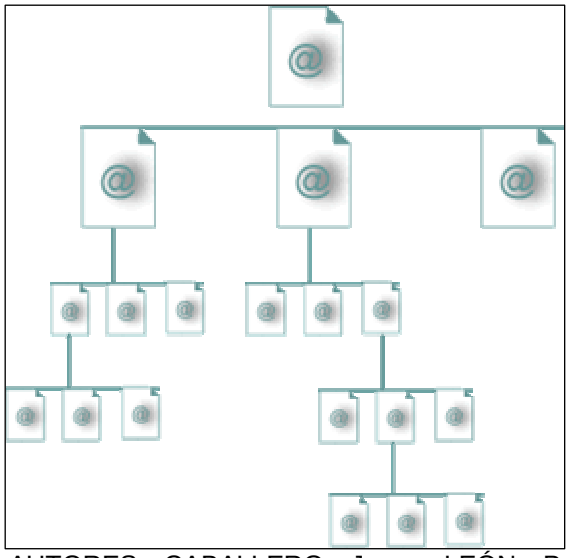
Para tener éxito con este tipo de estructura es necesario que las unidades individuales de una estructura reticular, compartan una estructura de temas y subtemas uniforme.

Este tipo de estructura puede ser difícil de comprender si el usuario no consigue establecer la relación entre las diferentes categorías de información.

En estas estructuras, resultan muy útiles los mapas gráficos del "Site".

3. Estructura Jerárquica: es quizá la mejor manera de organizar información compleja. Este tipo de estructura está especialmente bien adaptada para los Web "Sites", que siempre parten de una única página de inicio.

FIGURA 2-3: JERÁRQUICA

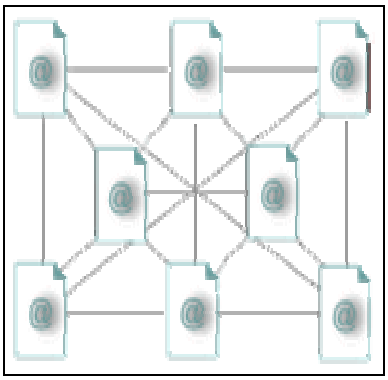


ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006.

La organización jerárquica impone disciplina a la hora de analizar el contenido, ya que las jerarquías sólo funcionan bien cuando el material está perfectamente actualizado.
Dado que las estructuras jerárquicas son habituales en las oficinas e instituciones resulta fácil para los usuarios construir modelos mentales.

4. Web: Estas estructuras imponen pocas restricciones en el patrón de la información.

FIGURA 2-4: WEB



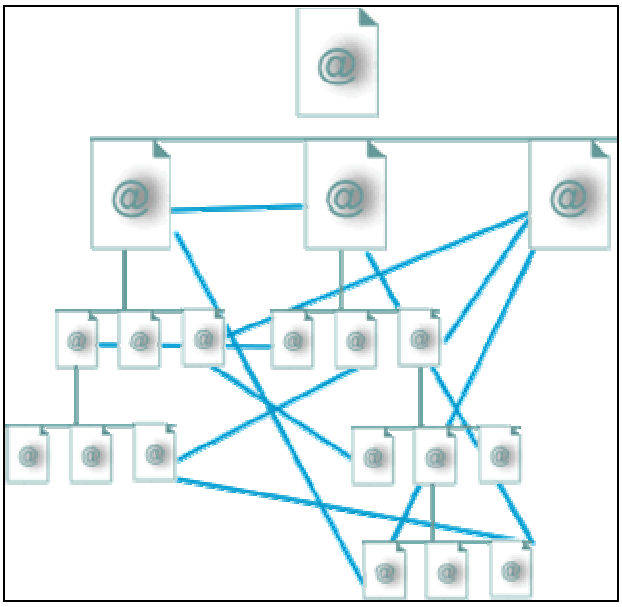
ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006.

En este caso el objetivo es realizar asociación de la información y la libre circulación de ideas; los usuarios siguen sus intereses en una secuencia única para cada usuario.

La meta es explotar al máximo el poder del hipertexto en la Web, pero este tipo de estructuras usualmente confunden al usuario, por lo cual funcionan mejor en Aplicaciones Web pequeñas, en los que predominan los listados de enlaces y que están dirigidas a usuarios muy instruidos o a usuarios experimentados.

5. Estructura Mixta: Los estudios de "funcionalidad" indican que los usuarios recuerdan mejor la información con este tipo de estructura (jerárquica con enlaces cruzados del tipo estructura Web), ya que la estructura jerárquica resulta demasiado restrictiva y la no lineal ofrece demasiada información y lleva a la confusión.

FIGURA 2-5: MIXTA



ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006.

Adicionalmente se deben realizar las siguientes consideraciones sobre la estructura de un sitio Web.

1.- Tener una estructura y conseguir que refleje el contenido de sitio y sus relaciones.

2.- La estructura no debe reflejar en ningún caso la organización de la compañía. La estructura debe determinarse por las tareas, incluso si eso significa que varios departamentos tienen que colaborar para realizar un sólo documento.

Cuando se desarrollan aplicaciones Web se recomienda considerar que estas deberían también poder ser ejecutadas desde dispositivos móviles tales como PDA's y celulares, en este caso se deben realizar las siguientes consideraciones ya que los dispositivos móviles presentan ciertas particularidades:

- Dificultades en la comunicación con el humano: pantallas muy pequeñas, dispositivos de entrada lentos.
- Dificultades tecnológicas: fuente de energía limitada y menor velocidad de procesamiento.

- Dificultades en la conexión de red: tiempos de latencia muy prolongados, y ancho de banda altamente variable, por infinidad de factores como el cambio de celda, condiciones de tráfico, competencia con la señal de voz, etc.
- Menos CPU/Energía/Memoria en el Terminal.
- La heterogeneidad de los dispositivos hace que las características físicas afecten negativamente a la usabilidad de una aplicación en un dispositivo móvil.
- Los mecanismos de entrada no son coherentes entre ellos. Por ejemplo PDA ofrece punteros y clic en pantallas sensibles mientras que la mayoría de los teléfonos móviles incorporan una serie de botones.

Para solucionar los problemas antes mencionados se deben tomar en cuenta los siguientes puntos:

- Escribir en este tipo de dispositivos es difícil para lo cual es preferible realizar la selección de opciones, aunque lleve más pasos; paro lo cual se hacen las siguientes sugerencias:

    - Usar los números para la entrada tanto como sea posible.
    - Usar abreviaturas comunes como los códigos de país.
    - Ofrecer opciones (radio – buttons, listboxes, números) y valores por defecto cuando sean aplicables.
    - Dar etiquetas a los botones del hardware cuando sea posible.
    - Usar los convenios estándar de los botones.
    - Las pantallas de los móviles son en general bastantes más pequeñas y en general están dotadas de baja resolución y profundidad de color. Así que leer textos largos es más difícil mientras que trabajar con fragmentos de información es mejor.

Una cuestión básica es reducir la necesidad de entradas de usuario tanto como sea posible.

- Sólo hay que dar información relevante. Por ejemplo, en cuanto a las fechas no es necesario dar información de minutos y segundos de no ser necesario. Es preferible evitar utilizar contenidos multimedia, gráficos y efectos visualmente vistosos, prevaleciendo en este caso la calidad del contenido y el servicio que se brinda.

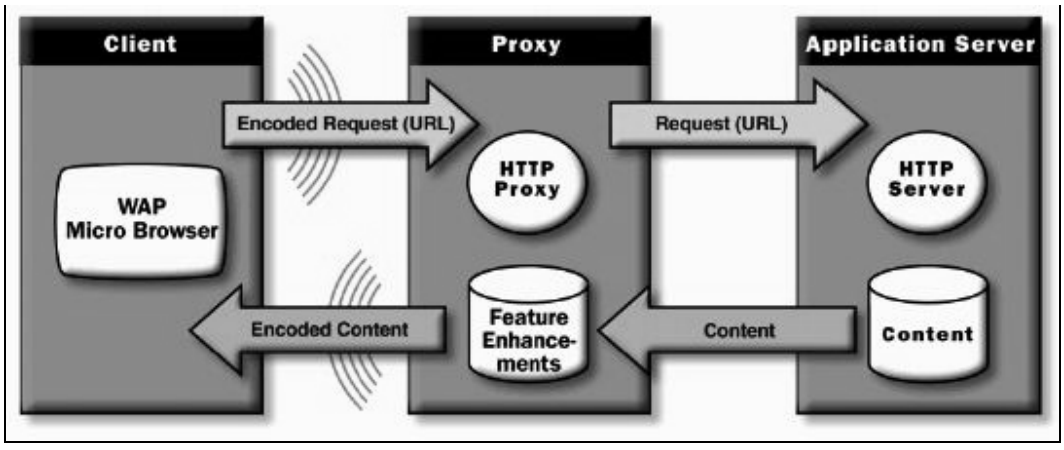
- Adicionalmente hay que tener cuidado con alterar el camino principal de acción del usuario con publicidad, contenidos poco relevantes o pasos innecesarios

- En cuanto a la estructura de la aplicación esta debe ser sencilla en la que es preferible tener muchas categorías con poca profundidad.

La gestión de la navegación hacia atrás es importante así como conservar la información introducida anteriormente para no obligar al usuario a teclearla de nuevo.

Siendo la tecnología WAP para dispositivos móviles la más utilizada se detallan los componentes de la misma

FIGURA 2-6: WAP CON UN GATEWAY [22]



FUENTE: http://www.tejedoresdelweb.com/307/article-1873.html

[22] http://www.tejedoresdelweb.com/307/article-1873.html

WAP funciona con gateway o proxies que conectan internet con los servicios móviles. Los gateways son operados por los proveedores de las redes de telefonía inalámbrica.

Los gateways son la pieza fundamental, puesto que además de conectar dos mundos (la Web tradicional y la Web inalámbrica), realizarán procesos de codificación de datos (por ejemplo, reducción de resolución en imágenes, convertir formatos de sonido, etc.)

2.3.3 ETAPA 3: FASE DE IMPLEMENTACIÓN Y PRUEBAS

2.3.3.1 Actividades

En la fase de Implementación se propone realizar las siguientes actividades:

- Caracterización de las herramientas que se van a utilizar en el desarrollo de la aplicación.
- Implementación del modelo objeto relacional en la base de datos.
- Realizar el Modelo de Implementación
- Pruebas de la Aplicación.

2.3.3.2 Estrategias

Realizar la caracterización de las herramientas de implementación en donde se describen las características, beneficios del software que se va a utilizar para:

- Front-End.
- Lenguaje de programación.
- Back-End.
- Sistema Operativo.
- Requerimientos de Hardware

Para la implementación del modelo objeto relacional en la base de datos se la realiza mediante la transformación del modelo resultante de realizar el diagrama de clases UML a un modelo objeto relacional.

Esta guía parte del diseño de asociaciones, agregaciones, composiciones y generalizaciones que se obtuvo como resultado en el diagrama de clases en la fase de diseño para luego aplicarlos en el modelo objeto relacional.

Una de las principales diferencias entre el modelo relacional y objeto relacional es la Primera Forma Normal (1NF), que es una de las reglas básicas en el modelo relacional. En el modelo Objeto Relacional esta regla fue eliminada para que una columna de una tabla (object table) pueda contener una colección de datos, tal es el caso de las tablas anidadas o varrays.

A continuación se describe en la Tabla 2.2 una guía que será utilizada posteriormente para el diseño de la base de datos objeto relacional.

TABLA 2-2: GUÍA PARA EL DISEÑO DEL MODELO OBJETO RELACIONAL

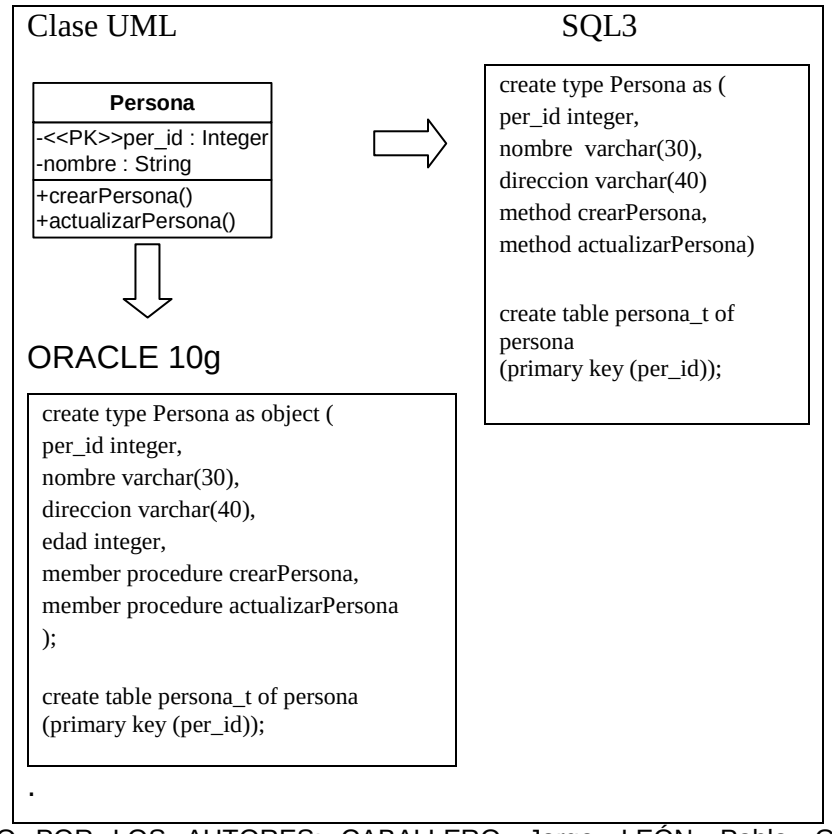
UML	**SQL3**	**Oracle**
Clase	Tipo Estructurado	Object Type
Atributo o Primary Key <<PK>> o UNIQUE <<AK>> o Multivaluado <<MA>> o Compuesto o Calculado	Atributo o PK en la tabla o UNIQUE en la tabla o ARRAY o ROW o Trigger/Method	Atributo o PK en la tabla o UNIQUE en la tabla o VARRAY o Object Type o Trigger/Member
Asociación o Uno a Uno o Uno a Muchos o Muchos a Muchos	 o REF/REF o REF/ARRAY o ARRAY/ARRAY	 o REF/REF o REF/Nested Table o Nested Table/ Nested Table
Composición	ARRAY	Nested table embebida en la persistencia del objeto TODO(ver definición de composición)
Agregación	ARRAY	Nested Table
Generalización	Types/Typed Tables	UNDER (Oracle soporta la herencia pero solo de tipos.No soporta herencia de tablas.)

ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006.

A continuación se detalla como se realiza la transformación del modelo de clases al modelo SQL3 primeramente para posteriormente adaptarlo a Oracle

Clases.- Únicamente las clases persistentes deben ser implementadas en el modelo objeto relacional de la base de datos. Para transformar una clase persistente del diagrama de clases es necesario definir el tipo de objeto así como la extensión. Un object type en SQL3 está definido como un tipo estructura, y su extensión y persistencia está definida como una tabla del object type, igualmente que para Oracle con la diferencia de sintaxis en la cual se especifica el tipo “AS OBJECT “, ver Figura 2-7.

FIGURA 2-7: TRANSFORMACIÓN DE CLASES



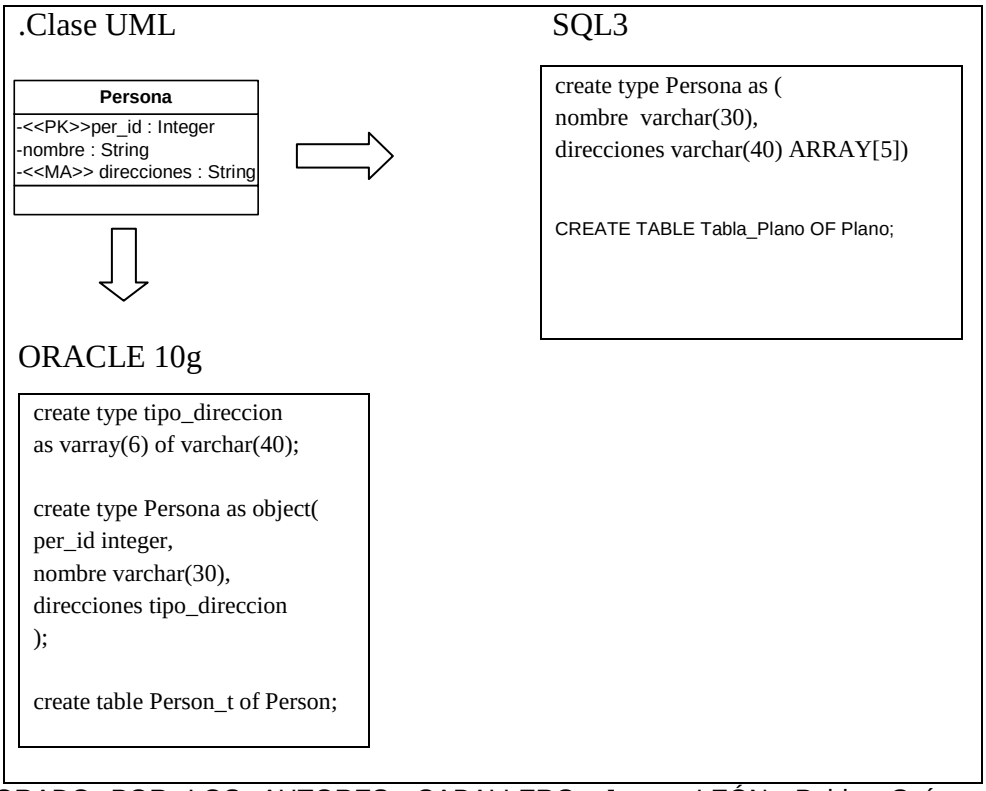
ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006.

Atributos y Métodos.- Cada atributo de la clase UML es transformado a un atributo del object type (tanto para Oracle como para SQL3). Los niveles de visibilidad no son soportados por tanto estos no son tomados en cuenta en la

fase de implementación, los niveles de visibilidad pueden ser implementados mediante vistas o privilegios.

- Atributos Multivaluados: Son representados en SQL3 y en Oracle mediante el tipo colección. En SQL3 los tipos colección se implementan mediante el tipo VARRAY por que es el único tipo de colección soportado por el estándar, mientras que en Oracle se puede escoger entre VARRAY y las tablas anidadas (nested table), ver Figura 2-8.

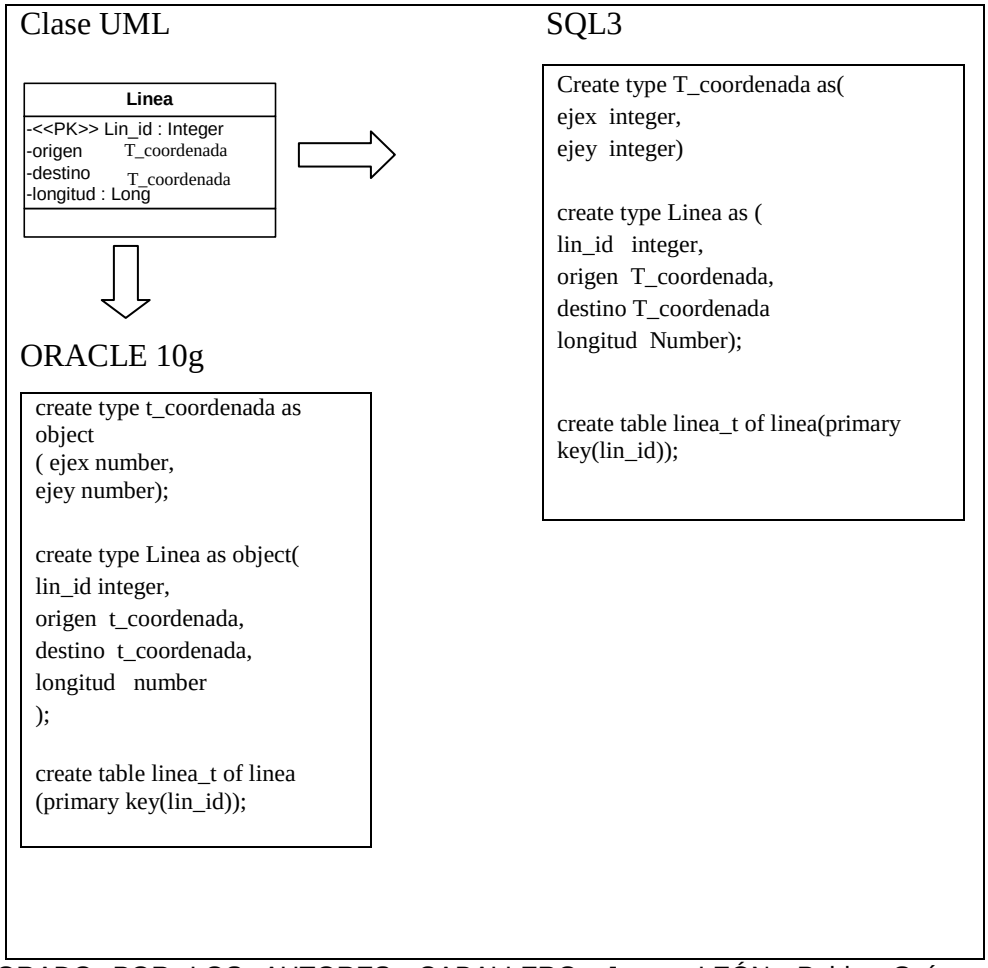
FIGURA 2-8: TRANSFORMACIÓN DE ATRIBUTOS MULTIVALUADOS



ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006.

- Atributos Compuestos: Pueden ser representados en el modelo objeto relacional en SQL3 mediante ROW type mientras que en Oracle se implementan mediante object type sin extensión (definiendo el object type sin especificar una tabla asociada) ver Figura 2-9.

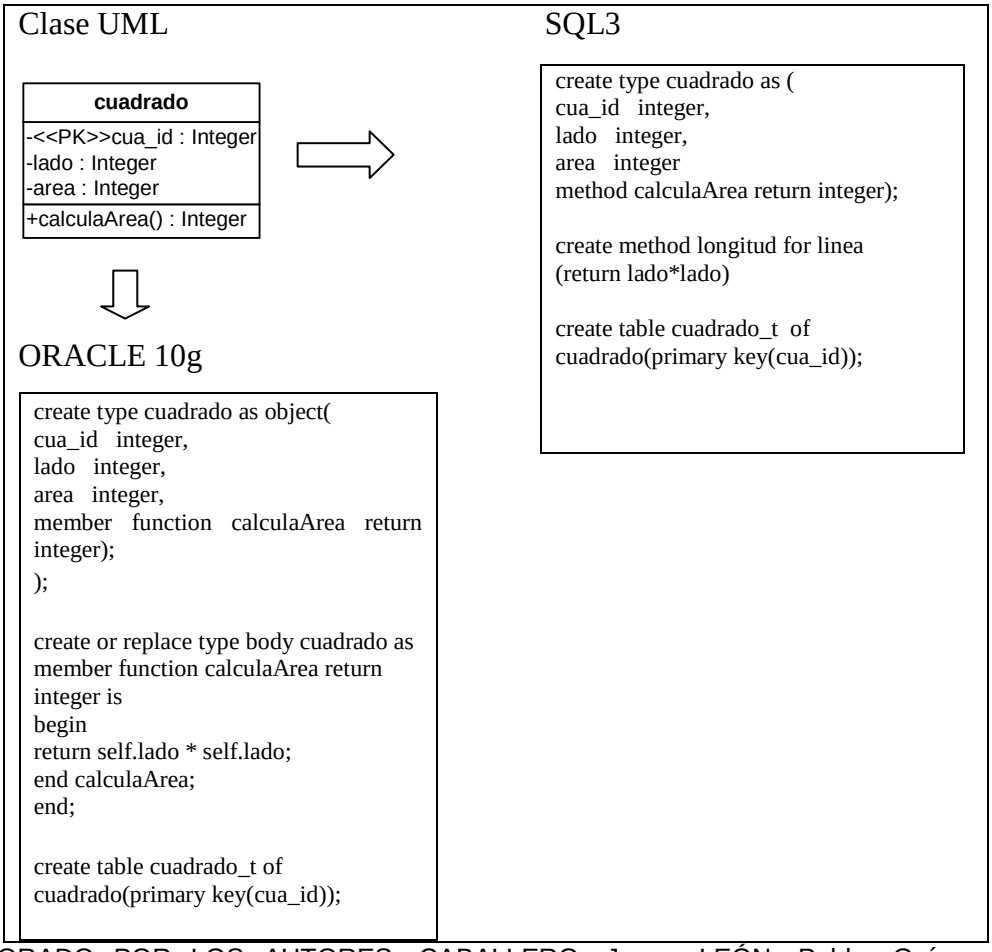
FIGURA 2-9: TRANSFORMACIÓN DE ATRIBUTOS COMPUESTOS



ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006.

- Atributos Derivados: Pueden ser implementados mediante triggers, métodos o incluso pueden utilizarse las dos alternativas ver Figura 2-10.

FIGURA 2-10: TRANSFORMACIÓN DE ATRIBUTOS DERIVADOS

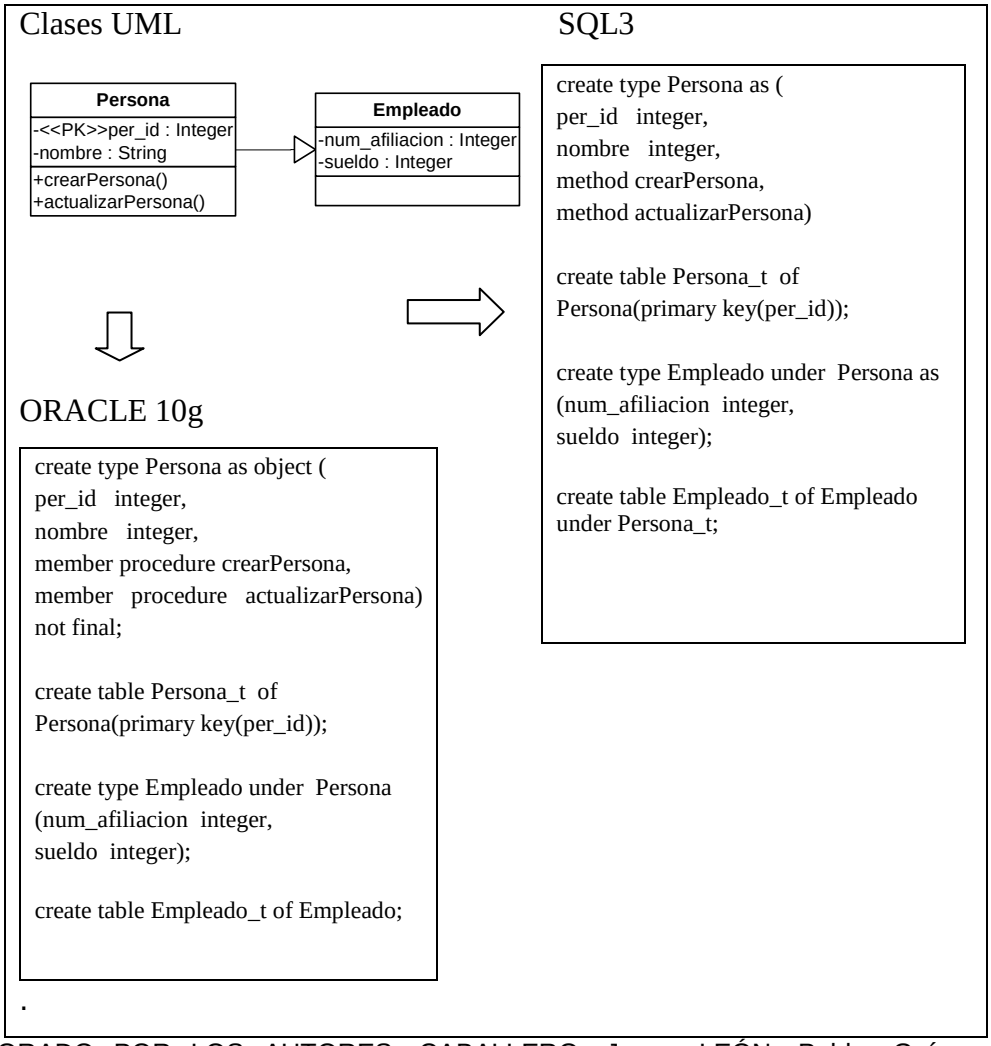


ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006.

- Métodos: Son transformados a SQL3 y Oracle especificando el nombre del método en la definición del object type ver Figura 2-10.

Generalización.- La definición se la realiza incluyendo la cláusula UNDER en el object type. Ver Figura 2-11

FIGURA 2-11: GENERALIZACIÓN



.

ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006.

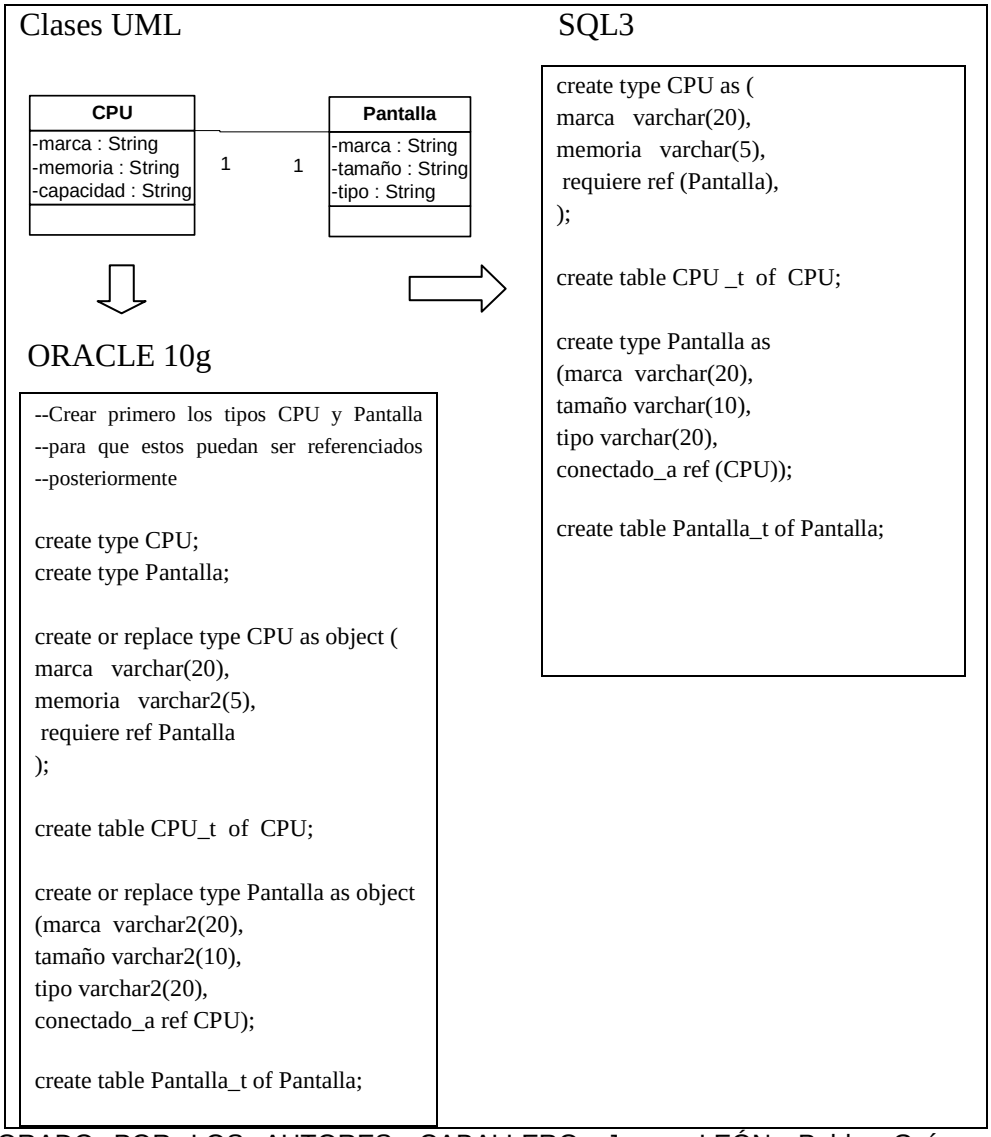
Transformación de las Asociaciones.- Las asociaciones UML pueden ser representadas en un esquema objeto relacional como relaciones unidireccionales o bidireccionales. En el caso de que se conozca que las consultas requieran información de ambas direcciones de la asociación entonces se recomienda que se implemente como una relación bidireccional mejorando de esta manera los tiempos de respuesta. Sin embargo se debería tener en cuenta que las relaciones bidireccionales no son mantenidas por el sistema, es por esto que la consistencia debe ser garantizada por medio de triggers o métodos.

Por lo tanto las relaciones bidireccionales a pesar de mejorar en algunos casos los tiempos de respuesta tienen un alto costo de mantenimiento.

Dependiendo del máximo de multiplicidad de las dos clases envueltas en una asociación, se propone las siguientes reglas de transformación.

Uno a Uno.- Se implementa a través de un tipo REF en cada object type que participa en la asociación. Si la mínima multiplicidad es uno, sería necesario poner la restricción de NOT NULL al atributo REF al que corresponde el tipo al que se hace referencia en la tabla; ya que las restricciones deben ser definidas en la tabla en lugar de realizarlas en el object type, ver Figura 2-12.

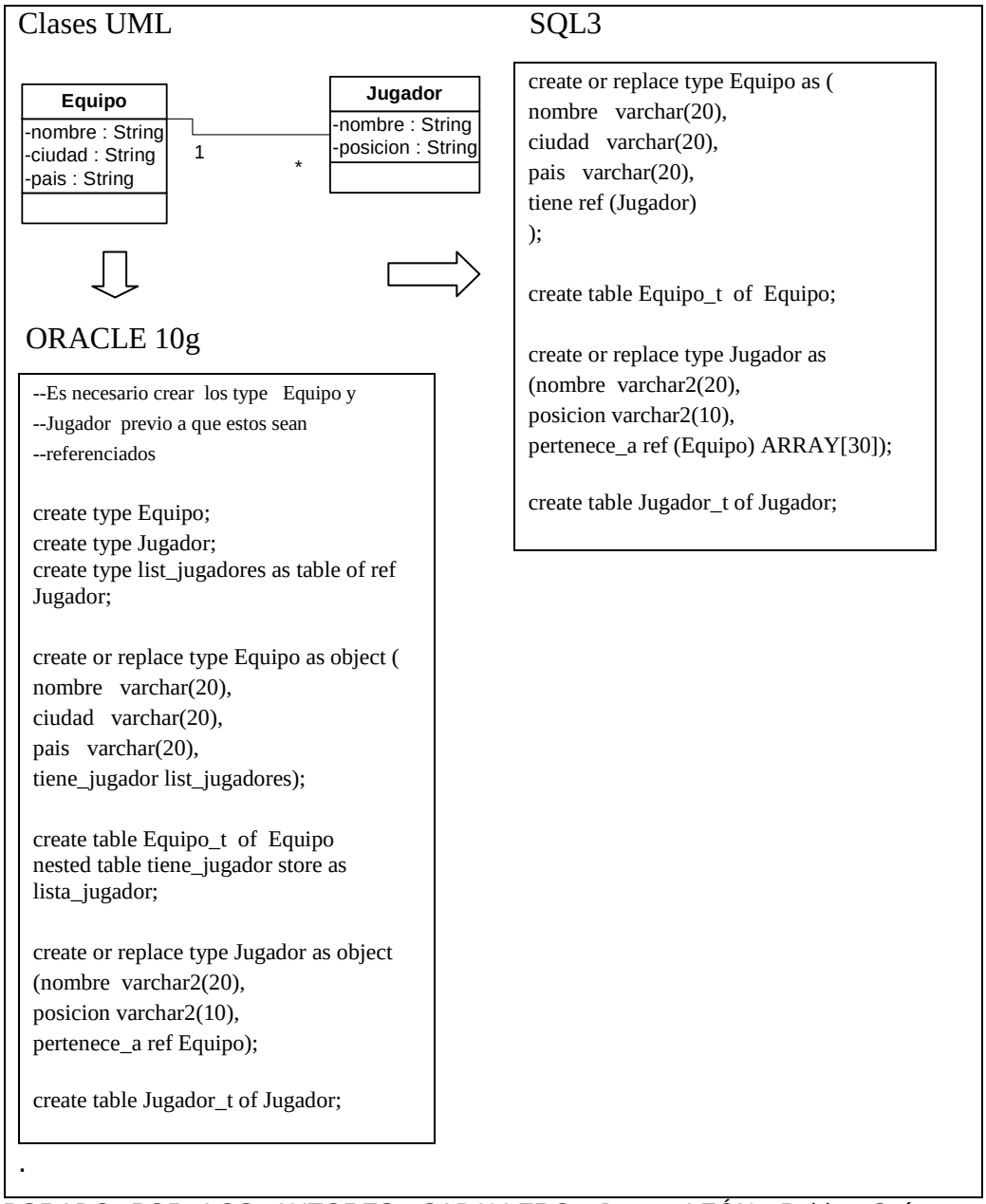
FIGURA 2-12: ASOCIACIÓN UNO A UNO



ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006.

Uno a Muchos.- Esta transformación se la realiza incluyendo un atributo tipo REF en el Object type que participa en la asociación con multiplicidad N e incluyendo un atributo de tipo colección en el object type que participa con multiplicidad uno. Los tipos colección contienen referencias (REF type) a otros object type que participan en la relación, ver Figura 2-13.

FIGURA 2-13: ASOCIACIÓN UNO A MUCHOS

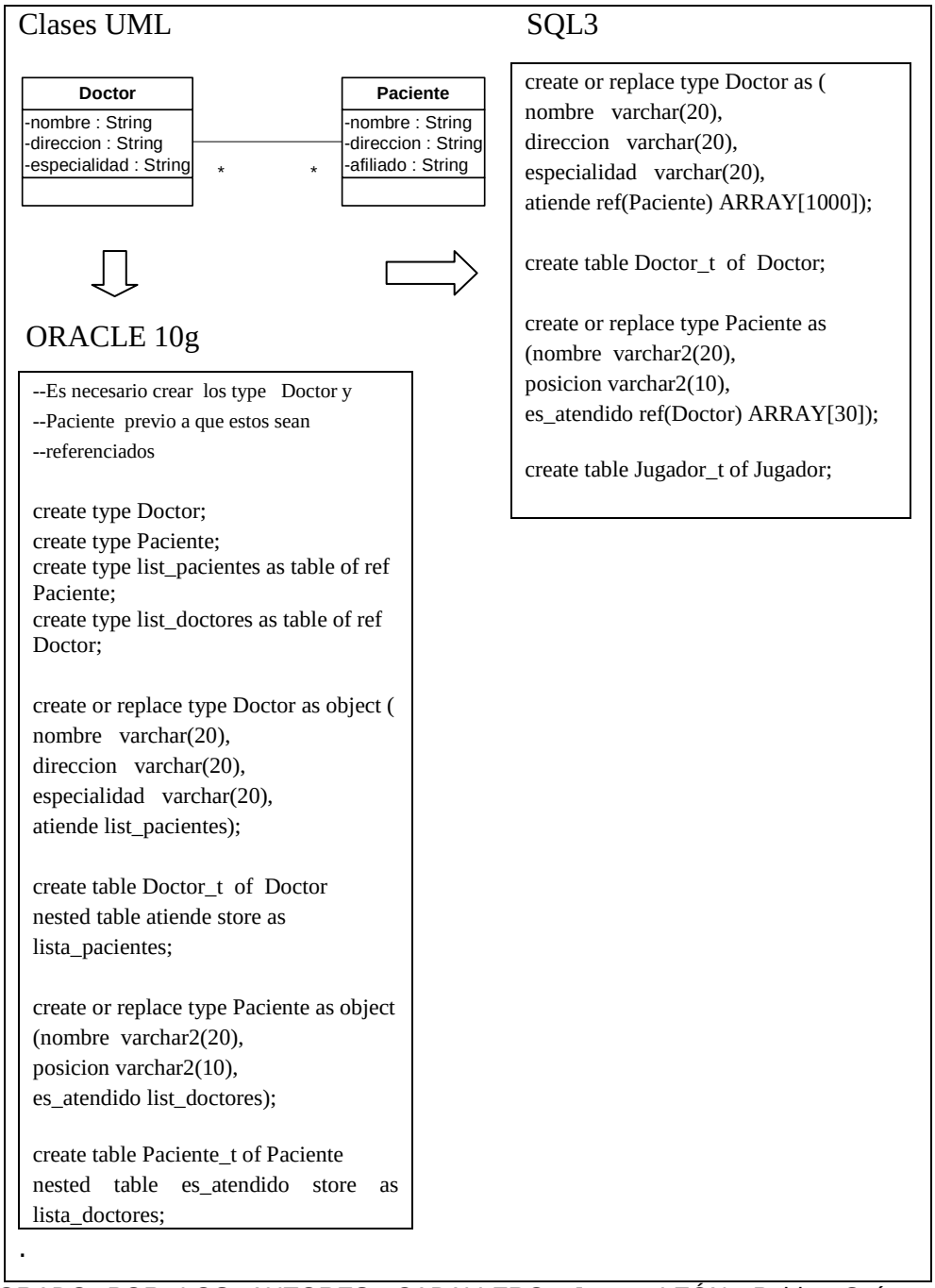


ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006.

Muchos a Muchos.- Se hace el mismo razonamiento que en el caso anterior, en este caso se implementa definiendo tablas anidadas en cada object type

que participa en la relación, ver Figura 2-14, aunque de ser posible se recomienda transformar a relaciones de uno a muchos.

FIGURA 2-14: ASOCIACIÓN MUCHOS A MUCHOS

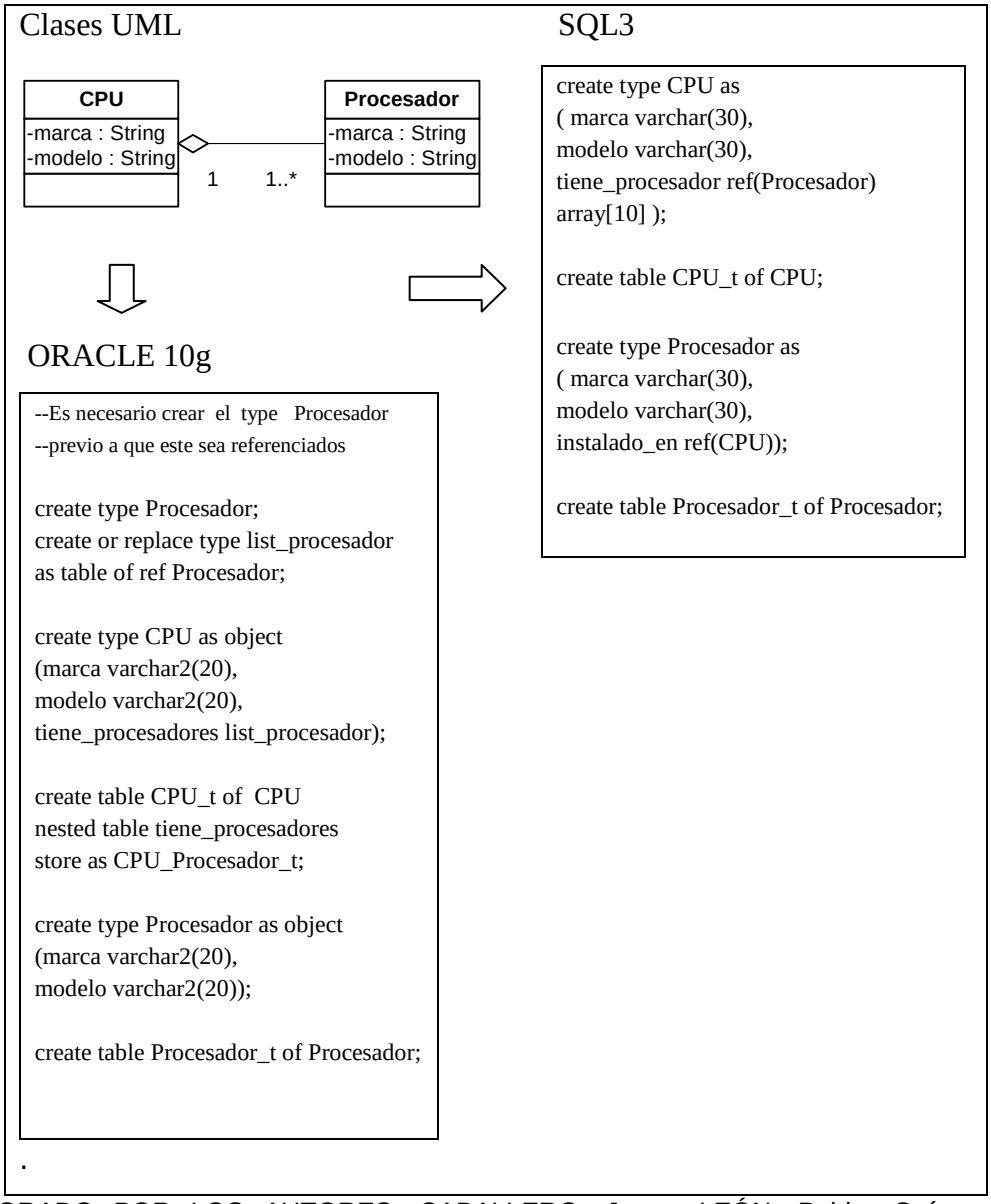


ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006.

Agregaciones simples[19] .- Para representar este tipo de agregación en un modelo objeto relacional se incluye en la definición del tipo TODO un atributo de tipo colección. Esta colección contiene referencias a sus PARTES

En SQL3 la colección es definida por medio de ARRAY mientras que en Oracle la colección debería ser una tabla anidada, ver Figura 2-15.

FIGURA 2-15: AGREGACIONES SIMPLES



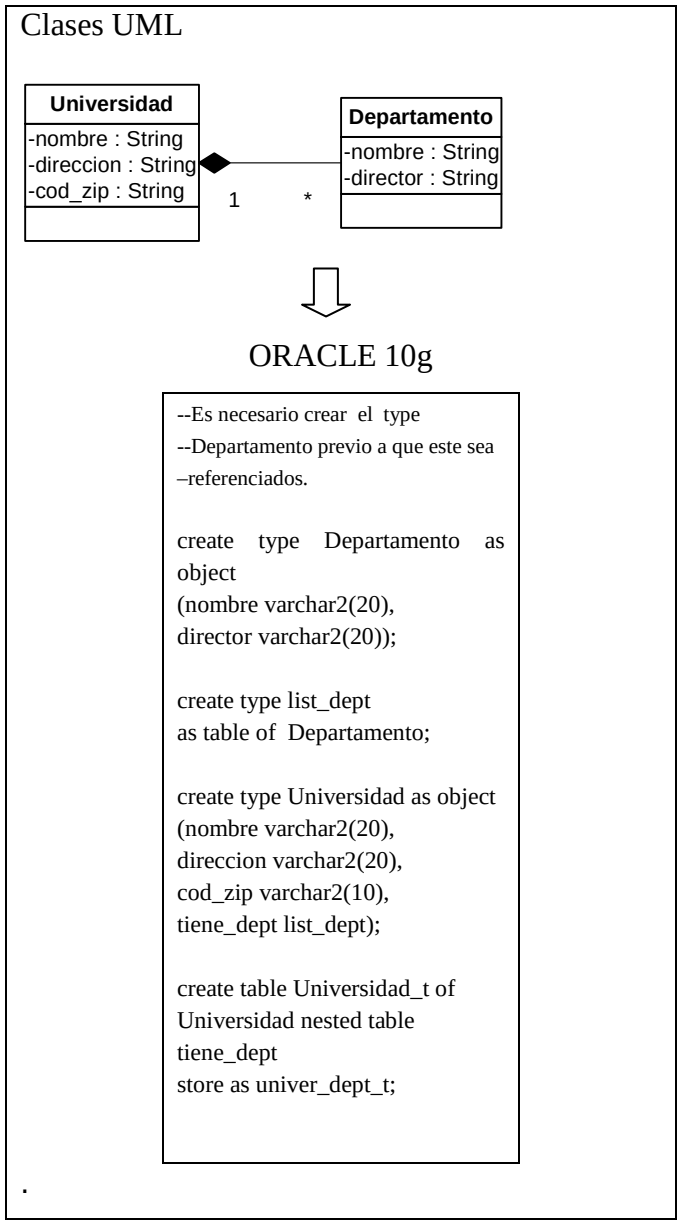
.

ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006.

[19] http://kybele.escet.urjc.es/documentos/BD/T4-DisenyoBDOR_Alumnos.pdf.

Composiciones[19] .- Para representar una agregación o composición en SQL3 se debe introducir un atributo en la definición del TODO, en el caso de SQL3 puede ser únicamente un ARRAY y las restricciones deben ser implementadas mediante el uso de checks o triggers. En Oracle la composición es implementada mediante la utilización de tablas anidadas; donde la persistencia de la PARTE esta incluida en la tabla anidada del TODO. En la Figura 2-16 no se incluye la implementación en SQL3 ya que es la misma que la agregación.

FIGURA 2-16: COMPOSICIÓN



[19] http://kybele.escet.urjc.es/documentos/BD/T4-DisenyoBDOR_Alumnos.pdf

Para realizar el Modelo de Implementación se tienen las siguientes estrategias:

- Realizar el diagrama de Componentes y de Despliegue, en el diagrama de componentes se modelan los aspectos físicos de un sistema y en el de despliegue se identifica la topología de procesadores y dispositivos sobre los que se ejecuta el software.

  - El Diagrama de Componentes representa módulos físicos de código (componentes) conectados por relaciones de dependencia. Los diagramas de componentes muestran tanto los componentes de software (código fuente, binario y ejecutable) como las relaciones lógicas entre ellos en un sistema. Estos componentes pueden ser:

    - Componentes: librerías dinámicas (dll), ejecutables, páginas web.
    - interfaces.
    - Relaciones de dependencia, generalización, asociación y realización.

  - El Diagrama de despliegue muestra las relaciones entre los componentes de hardware y software del sistema, este diagrama se utiliza para representar las particiones físicas (nodos) del sistema de información y los interfaces existentes entre ellas. Muestra la configuración (relaciones físicas) de los nodos que participan en la ejecución y de los componentes hardware y software que residen en ellos. Un diagrama de despliegue es un diagrama que muestra la configuración de los nodos que participan en la ejecución y de los componentes que residen en ellos. Contiene: Nodos, Servidores o Procesadores, Dispositivos, Relaciones de Dependencia y asociación.

- Se detallan los estándares de programación utilizados tanto para nombres de: la base de datos, formas, controles, variables, funciones, métodos, objetos, atributos, persistencias, tablas anidadas y colecciones. Los estandares utilizados para el desarrollo de la aplicación se los detalla en el capítulo 3 en la sección 3.3.2.3.5 Estándares de programación.

Para realizar las pruebas se adjunta una plantilla en el **ANEXO D**. La plantilla recomienda realizar un caso de prueba por cada caso de uso.

Adicionalmente para realizar las pruebas de una Aplicación Web se recomienda hacer las siguientes consideraciones.

- Revisar el modelo del contenido para descubrir errores tipográficos, errores gramaticales, errores en la consistencia del contenido, errores en representaciones gráficas y de referencias.

- Probar la navegación de la Aplicación en la cual se revisa los enlaces de navegación para asegurar su correspondencia con los específicos para cada usuario.

- Se prueba la funcionalidad global de la Aplicación Web, y se realizan pruebas de validación basándose éstas en casos prácticos, tomando en cuenta la arquitectura que se haya seleccionado (lineal, jerárquica, etc.)

- Probar que la Aplicación Web sea compatible con una variedad de configuraciones haciendo las pruebas con los Sistemas Operativos y plataformas de hardware para navegadores probables en los cuales vaya a ser utilizada la aplicación.

- Se selecciona un grupo de usuarios que abarque todos los roles posibles de usuarios y se evalúan los resultados de estas pruebas.

CAPITULO 3

APLICACIÓN DE LA GUÍA EN LA CONSTRUCCIÓN DE UNA APLICACIÓN PARA LA ADMINISTRACIÓN DE BIBLIOTECAS

3.1 DESCRIPCIÓN DE LOS REQUERIMIENTOS DE LA APLICACIÓN

La Aplicación a desarrollar debe permitir a los usuarios conectarse a la biblioteca mediante la Web. El usuario accede a la página principal consulta de documentos, ingresa la palabra o frase mediante la cual el sistema va a buscar dependiendo del tipo de documento que el usuario necesita pudiendo ser estos:

- Libro
- Tesis
- Revista

En el caso de que el sistema encuentre la información desplegará el/los título/s de los documentos encontrados en la búsqueda realizada por la palabra o frase tanto en el título, autor del documento como en el contenido del mismo.

El usuario una vez que encuentra el documento que requiere tiene la posibilidad de: visualizar su contenido, así como la opción de reservar el ejemplar, no pudiendo ser reservado si ya esta prestado, perdido o reservado. La reservación solo puede efectuarse para el mismo día, la misma que una vez realizada no puede ser cancelada. El número máximo de ejemplares que se puede reservar es de tres.

En la Administración de Biblioteca tendrá los siguientes módulos:

- Préstamo
- Reportes
- Documento

- Ejemplares
- Categoría
- Autor
- Idioma
- Usuario del Sistema
- Usuario Consulta

Préstamo.- Este módulo permite realizar el préstamo, devolución y renovación del ejemplar. El préstamo de un ejemplar puede ser realizado cuando el usuario haya reservado el ejemplar o en los casos de: no estar reservado por otro usuario, prestado o perdido, un usuario puede hacer máximo 3 préstamos. La devolución de un ejemplar se la puede realizar cuando el ejemplar no este multado. La renovación del ejemplar se la realiza por el mismo número de días para el cual fue prestado y no puede ser renovado un ejemplar cuando se encuentra multado. Este módulo es utilizado por el bibliotecario.

El pago de la multa puede ser realizado tanto en la devolución y renovación del préstamo. La multa puede ser por concepto de atraso u otros. El valor de la multa depende de la categoría del ejemplar en el caso de atraso y en otros casos es criterio del Bibliotecario.

En el caso de que el usuario haya hecho una reservación del ejemplar, con lo cual se asegura que se asegura su préstamo, entonces, el usuario que realizó la reservación se acerca a la biblioteca entrega los documentos que sean necesarios para realizar el préstamo y llena la ficha del préstamo para posteriormente retirar el documento.

Reportes.- Este módulo permite realizar los reportes de: documentos no devueltos a tiempo, usuarios multados en un determinado rango de tiempo.
Este módulo es utilizado por el bibliotecario y el administrador

Documento.- Este módulo permite crear y actualizar un documento, en el caso de crear un documento se crea un ejemplar y despliega su código, además se

puede actualizar la información del documento. Los datos ingresados dependen del tipo del documento. Este módulo es utilizado por el Bibliotecario.

Ejemplar.- Este módulo permite crear más ejemplares de un determinado documento. Adicionalmente permite actualizar la información del ejemplar. Este módulo es utilizado por el Bibliotecario.

Categoría.- Este módulo permite actualizar la información de las categorías existentes que son original y copia que se las utiliza para cualificar a un ejemplar. Este módulo es utilizado por el Bibliotecario.

Autor.- Este módulo permite crear y actualizar la información del autor**.** Para crear el autor se necesita .Este módulo es utilizado por el Bibliotecario.

Idioma.- Este módulo permite crear y actualizar la información del idioma**.** Este módulo es utilizado por el Bibliotecario.

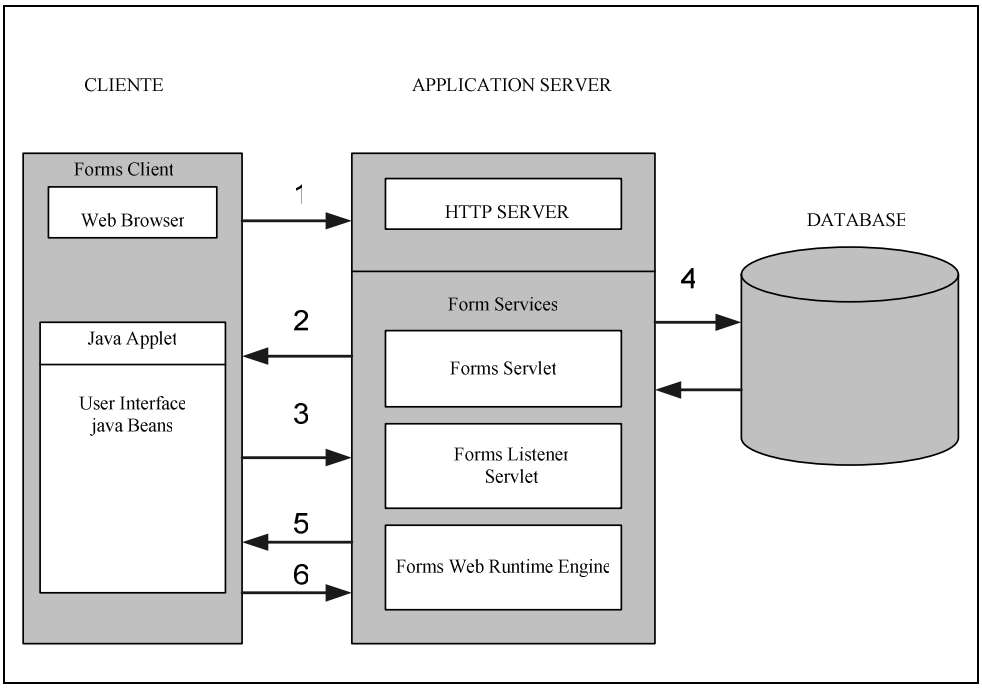
Usuario del Sistema.- Este módulo permite crear y actualizar la información del usuario del sistema, el mismo que puede ser Administrador o Bibliotecario. Este módulo es utilizado por el Administrador.

Usuario Consulta.- Este módulo permite crear y actualizar la información del usuario al se que puede reservar o prestar un ejemplar. Este módulo es utilizado por el Bibliotecario.

3.2 ARQUITECTURA DE LA APLICACIÓN

En la Figura 3-1 se muestra como interactúan el cliente Web, el Forms Server y la base de datos (Oracle 10g).

FIGURA 3-1: PROCESO DE UNA SOLICITUD AL FORMS SERVICES



ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006

1. EL URL ingresado en el browser referencia al Oracle HTTP Server (OHS) accesando al Forms Servlet.

2. El Forms servlet es una parte de Forms Services, dinámicamente genera una página HTML para el applet Forms el cual tiene todos los parámetros requeridos en la aplicación Forms. Posteriormente es descargado al browser del usuario, esta página HTML inicializa la descarga del cliente Forms Java, generado por un applet genérico. El cliente Forms Java se ejecuta en un entorno de Java, el cual puede ser el Java Plug-in de Sun, Oracle Jinitiator, o Microsoft Internet Explorer native Virtual Machine. Después de la descarga inicial desde el servidor, las clases Java usadas en el cliente son almacenadas en cache del browser del usuario, eliminando la necesidad de descargar nuevamente los mismos archivos para acceder nuevamente a la aplicación.

3. El cliente Forms Java accede al Forms Listener Servlet en el OAS para iniciar un nuevo proceso Web en ejecución.

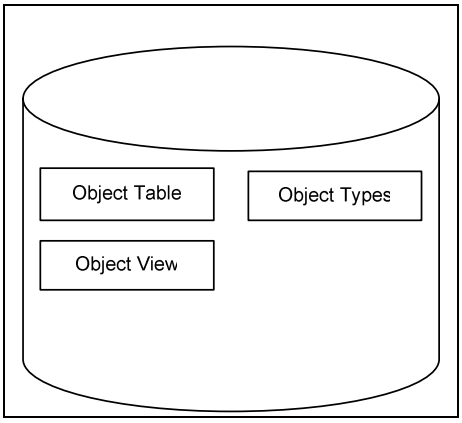
4. Un proceso Web es ejecutado al iniciar el Forms Server mediante el Forms Listener Servlet para cada sesión de usuario. Este es el proceso que se ejecuta para conectarse a la base de datos.

5. El mecanismo utilizado por el cliente Forms Java permite reducir la carga en la red al mínimo; en lugar de enviar el texto completo de cada mensaje, el Forms Web engine compara el contenido de cada mensaje con el mensaje enviado previamente, determinando la diferencia entre ambos. Si es diferente entonces se envía el mensaje.

6. La interfaz de usuario del Forms es totalmente generada como un applet Java usando un conjunto de clases genéricas. Este mismo conjunto de clases Java es usado para generar cualquier número y tamaño de aplicaciones Forms, sin necesidad de descargas adicionales. La Interfaz de usuario de Forms Java provee la misma calidad, presentación e interacción para el usuario que las aplicaciones cliente/servidor.

En la Figura 3-2 se muestran los objetos definidos por los desarrolladores en la base de datos.

FIGURA 3-2: OBJETOS DEFINIDOS EN LA BASE DE DATOS

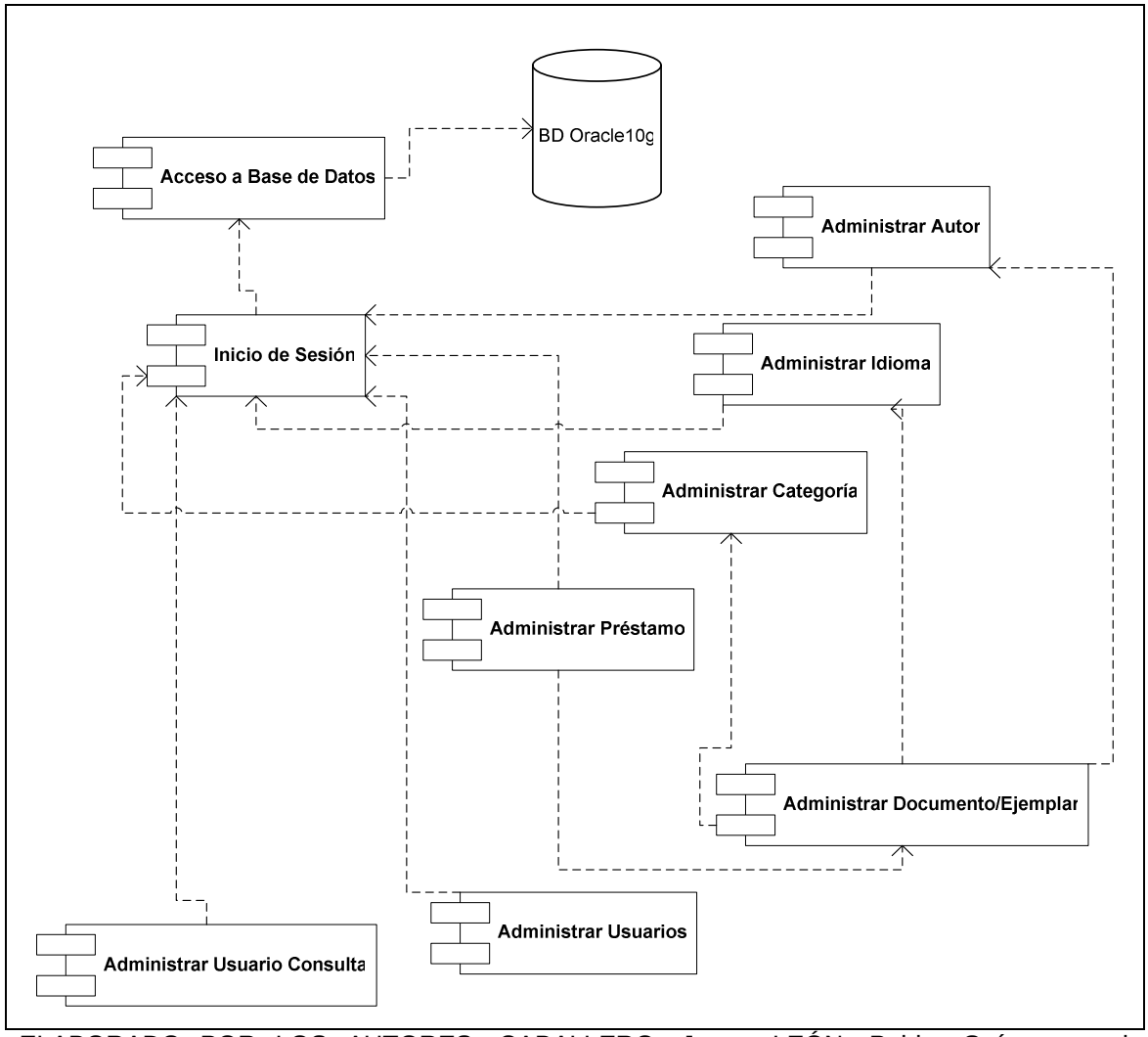


ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006.

A continuación presentamos los diagramas de componentes y de despliegue de la aplicación SIABI.

El Diagrama de Componentes muestra los modulos físicos de código (componentes) y las relaciones lógicas entre ellos en un sistema. La Figura 3-3 se muestra el diagrama de componentes de la aplicación SIABI.

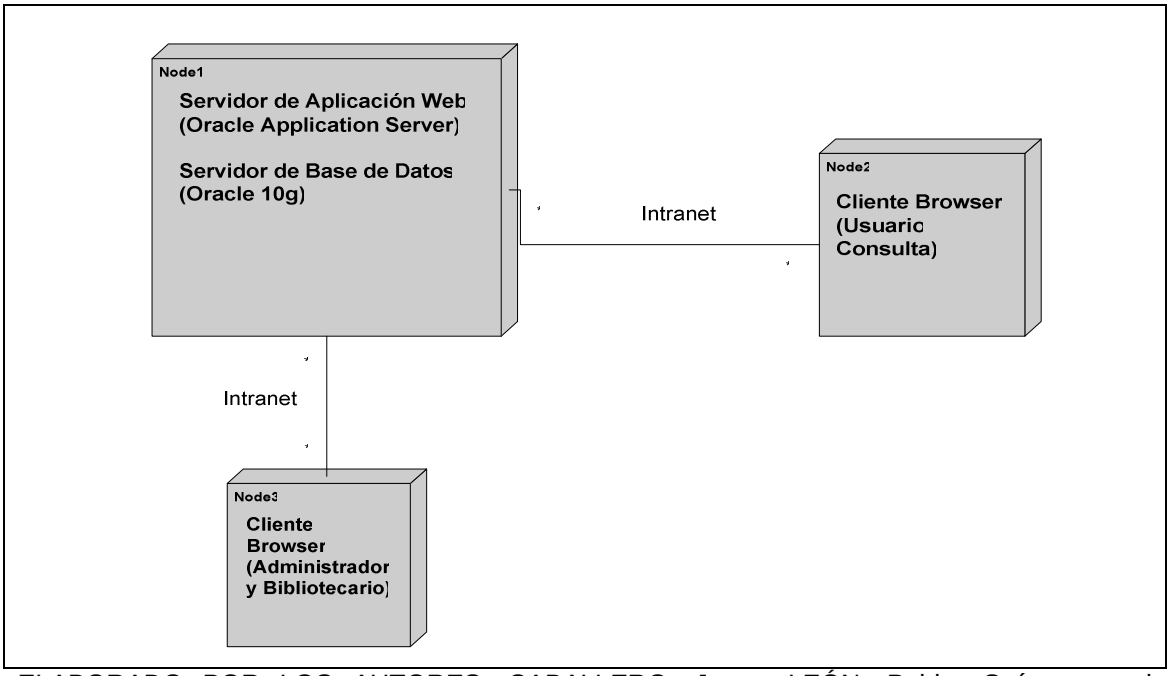
FIGURA 3-3: DIAGRAMA DE COMPONENTES



ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006

El diagrama de despliegue identifica la topología de procesadores y dispositivos sobre los que se ejecuta el sistema SIABI. La Figura 3-4 muestra el diagrama de despliegue de la aplicación SIABI

FIGURA 3-4: DIAGRAMA DE DESPLIEGUE



ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006

Los diagramas de componente y de despliegue se obtuvieron en la Etapa 3 (Fase de Implementación y Pruebas), siendo incluidos en esta sección a fines de anticipar la arquitectura de la aplicación para mostrárselas al lector.

3.3 DESARROLLO DE LA APLICACIÓN

La aplicación de caso de estudio Administración de Biblioteca se ha desarrollado con el fin de validar la guía presentada en el capitulo 2.

Para la validación de la guía se tomaron en cuenta las actividades y estrategias que la guía plantea para desarrollar aplicaciones Web utilizando modelos Objeto-Relacional a fin de demostrar la funcionalidad de la guía.

3.3.1 ETAPA 1: FASE DE REQUERIMIENTO Y ANÁLISIS

La recolección de los requerimientos del sistema de administración de bibliotecas (SIABI) se la realizó a partir de los requerimientos encontrados en proyectos de titulación y en la observación del funcionamiento de aplicaciones para la administración de bibliotecas. Una vez determinado el funcionamiento de una biblioteca se continúo con el análisis de la información recolectada para

iniciar el desarrollo de una solución para describir la implementación del modelo Objeto Relacional en una Aplicación Web.

3.3.3.1 Identificación de Metas Globales

Las metas globales identificadas para el Sistema de Administración de Bibliotecas (SIABI) son:

- Permitir a los usuarios de la biblioteca consultar el material bibliográfico disponible.
- Automatizar el proceso de consulta del material bibliográfico disponible en la biblioteca, con lo cual, se mejora substancialmente la atención en la biblioteca.
- Los usuarios potenciales de la aplicación son los usuarios de la biblioteca.
- Permitir a los usuarios realizar consultas y reservación del material bibliográfico disponible en la biblioteca.

3.3.3.2 Perfiles de Usuario del Sistema

En SIABI identificamos los perfiles de Administrador, Bibliotecario y Usuarios de la biblioteca. Independientemente de ésta distinción, todos los usuarios dispondrán de un sistema de privilegios que determinarán sus acciones en el sistema.

En la Tabla 3-1 se muestran los privilegios que tienen cada perfil en los distintos módulos del sistema SIABI.

TABLA 3-1: PRIVILEGIOS DEFINIDOS PARA LOS USUARIOS

PERFILES	PERMISOS						
	Modulo Préstamo	Modulo Reporte Consulta	Modulo Documento Ejemplar	Modulo Categoría	Modulo Administrar Usuario	Modulo Administrar Bibliotecario	Modulo Consulta Reservación
Administrador		✓				✓	
Bibliotecario	✓	✓	✓	✓	✓		✓
Usuario							✓

ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006

3.3.3.3 Herramientas de Análisis y Diseño de la Aplicación

En la Tabla 3-2 se muestran las herramientas utilizada en el desarrollo del sistema SIABI. Es importante mencionar que las herramientas que son utilizadas para fines del desarrollo del caso de estudio nos permiten modelar la aplicación

TABLA 3-2: HERRAMIENTAS DE ANÁLISIS Y DESEÑO DE LA APLICACIÓN

TIPO DE HERRAMIENTAS	CARACTERÍSTICAS UTILIZADAS	HERRAMIENTAS SELECCIONADAS
Herramientas Case	• Generar: modelo de datos y diagramas UML.	• Herramienta de Diseño y Modelamiento para UML. IBM Rational XDE Modeler.
Herramientas para el modelado de interfaces	• Permita modelar interfaces centradas en el usuario.	• Herramienta para el Modelado de Interfaces: Microsoft Visio 2004.

ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006.

3.3.3.4 Modelo de Casos de Uso

Para gestionar los requerimientos de la aplicación utilizamos como estrategia la identificación de los actores, la elaboración de: diagramas de casos de uso y especificación de casos de uso

ACTORES

Al establecer los requisitos de la aplicación se determina que intervienen los actores Usuario de la Biblioteca, Bibliotecario y Administrador que se describen en la Tabla 3-3

TABLA 3-3: ACTORES DEL SISTEMA

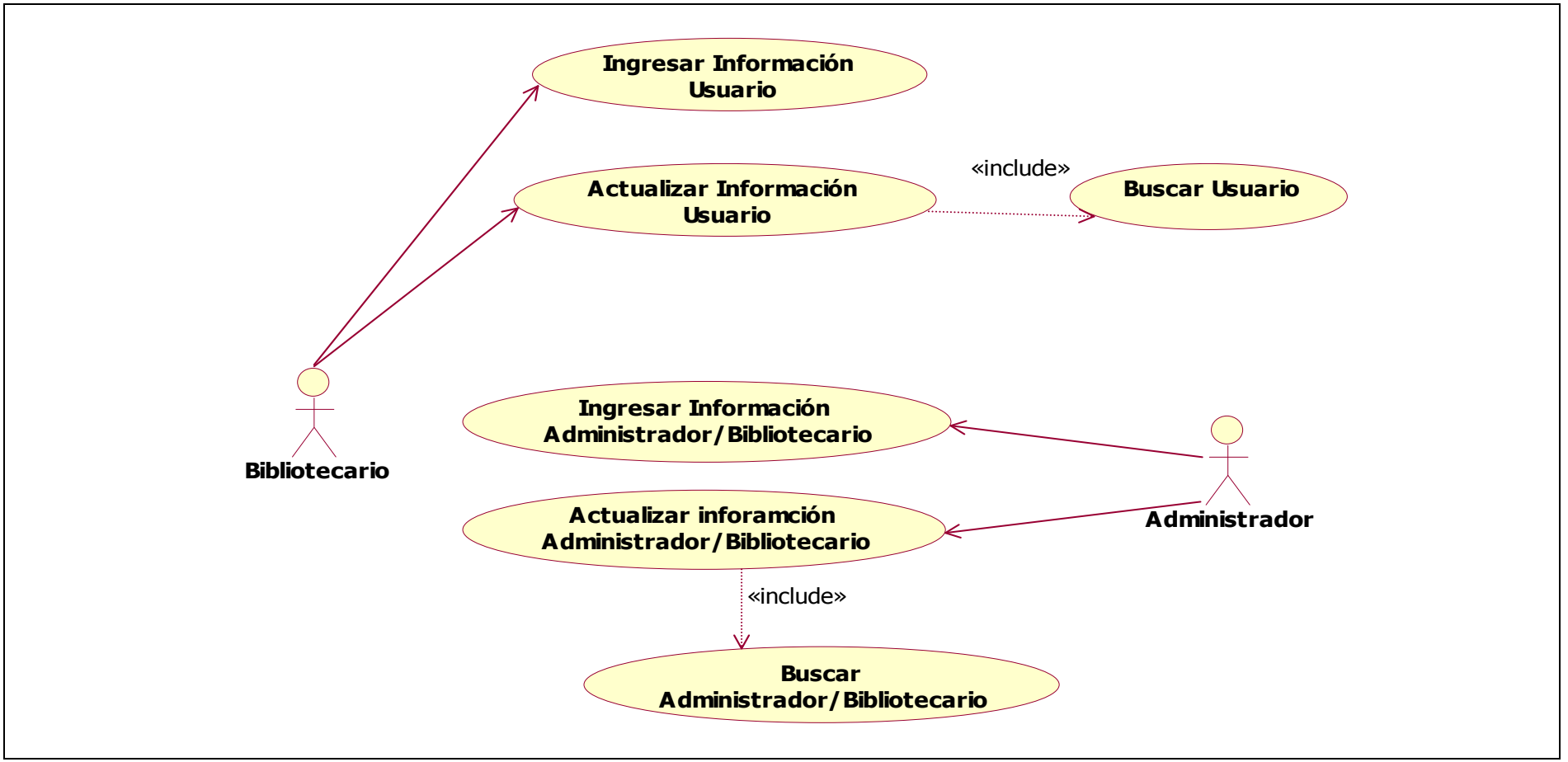
ACTOR	DESCRIPCIÓN
Usuario	Es la persona que solicita el préstamo de un documento de la biblioteca.
Bibliotecario	Es la persona que administra los documentos y los prestamos de los documentos así como el reporte del estado de los documentos de la Biblioteca.
Administrador	Es la persona encargada de administrar la información del bibliotecario así como los reportes de la aplicación

ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006

DIAGRAMAS DE CASOS DE USO

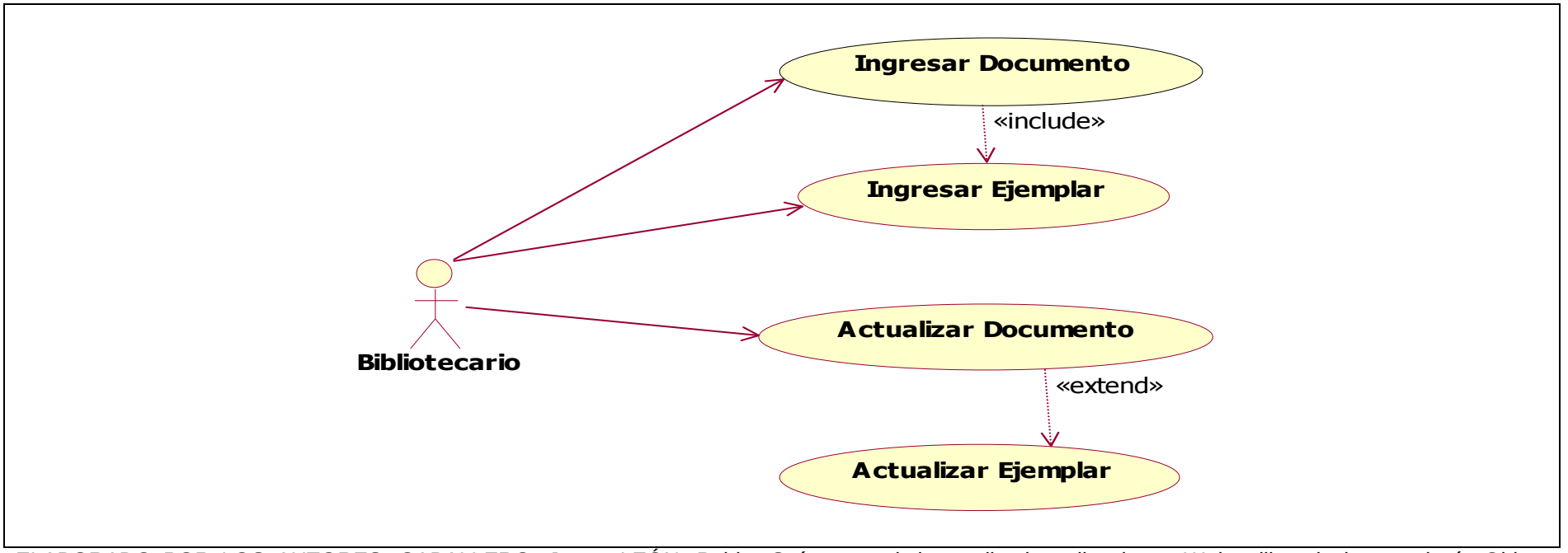
Al establecer los requerimientos de la aplicación se determina el modelo del negocio que lo ilustramos en las figuras siguientes:

FIGURA 3-5: ADMINISTRAR USUARIO



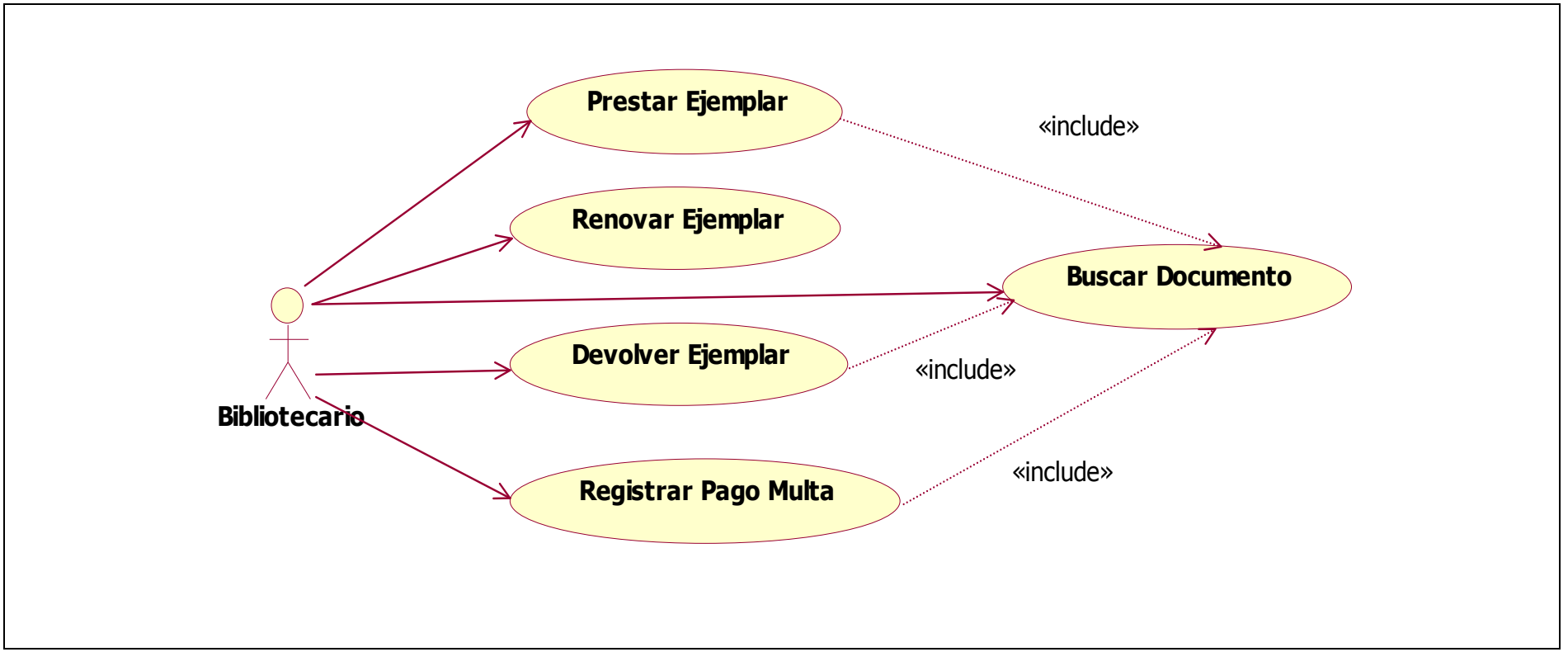
ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006

FIGURA 3-6: ADMINISTRAR DOCUMENTO Y/O EJEMPLAR



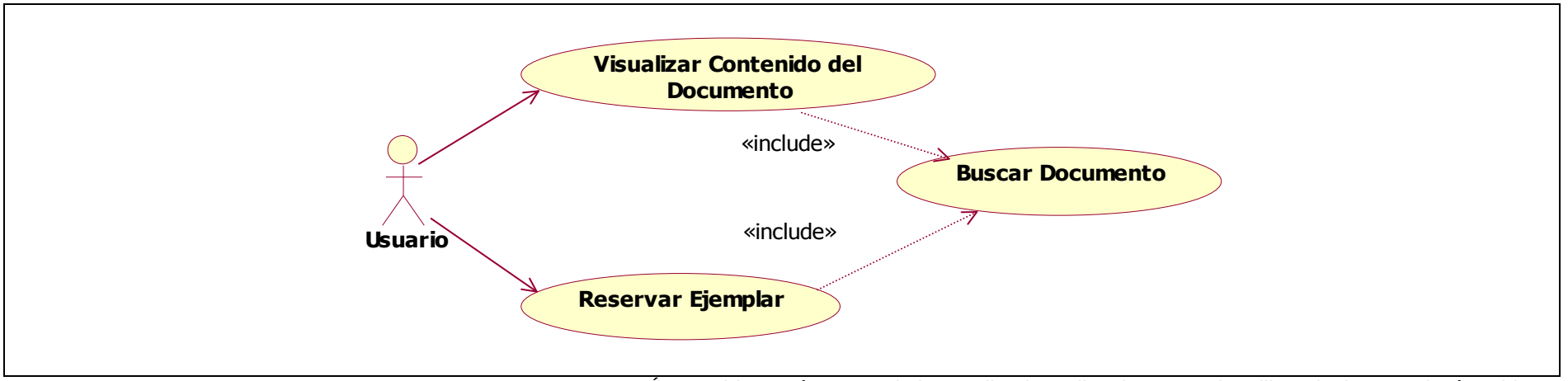
ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006

FIGURA 3-7: ADMINISTRAR PRESTAMO



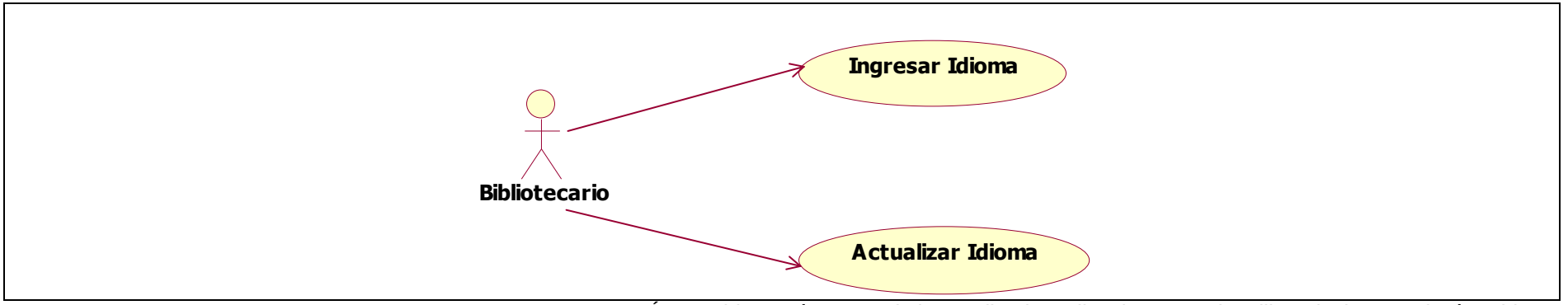
ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006

FIGURA 3-8: RESERVA Y VISUALIZACIÓN DE CONTENIDO DE DOCUMENTO



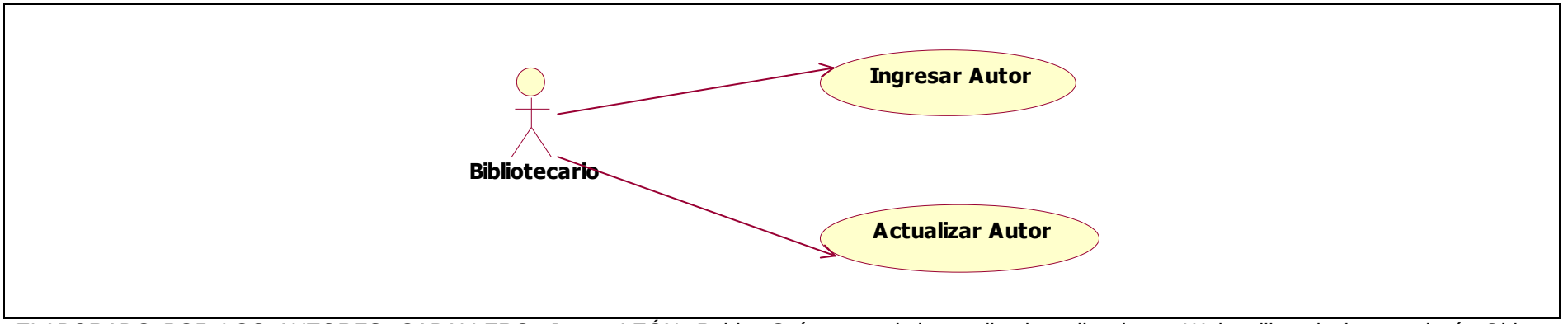
ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006

FIGURA 3-9: ADMINISTRAR IDIOMA



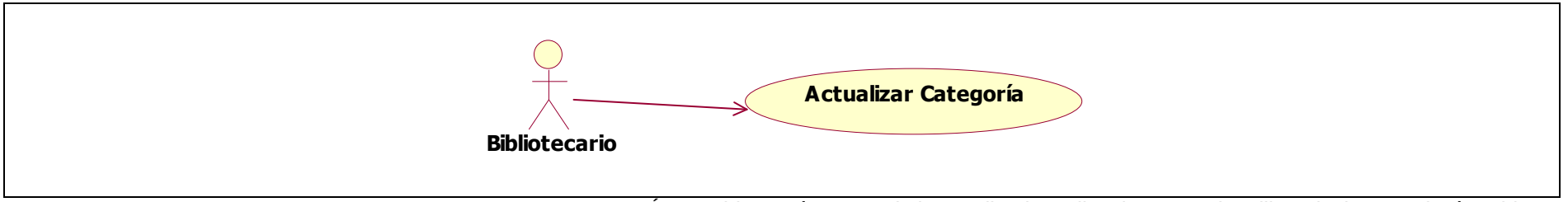
ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006

FIGURA 3-10: ADMINISTRAR AUTOR



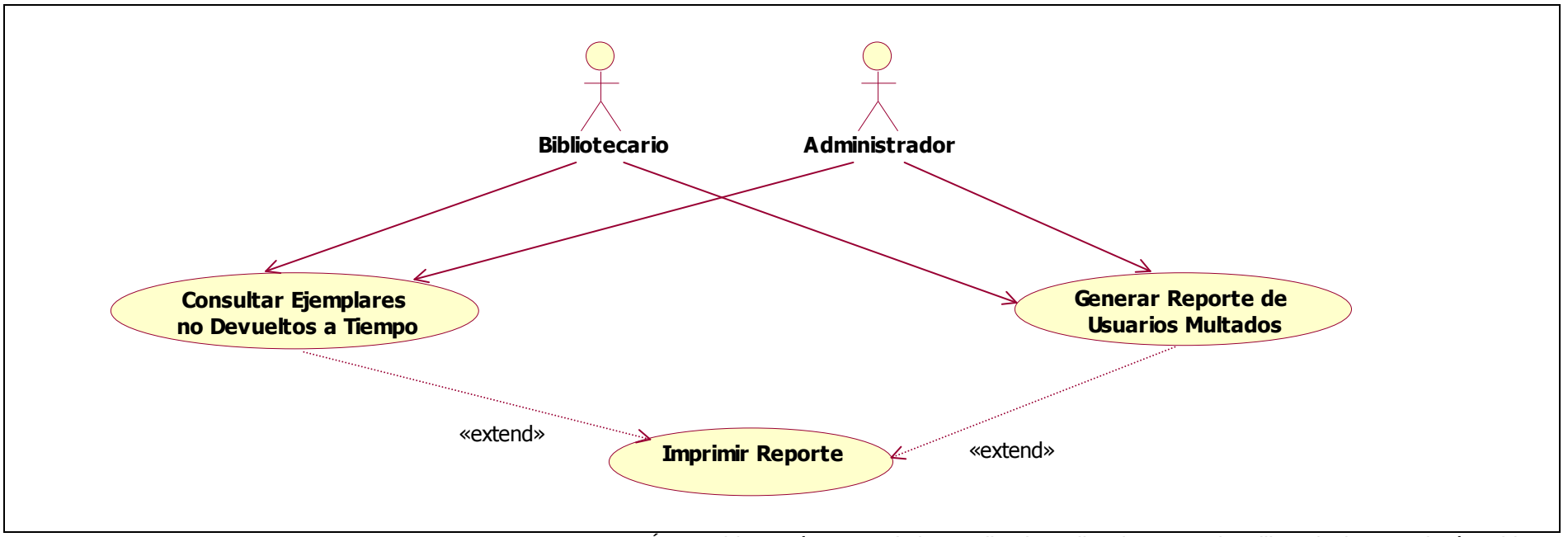
ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006

FIGURA 3-11 ADMINISTRAR CATEGORÍA



ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006

FIGURA 3-12: REPORTES



ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006

ESPECIFICACIONES DE CASOS DE USO

Las especificaciones de casos de uso permiten establecer los requerimientos funcionales de la aplicación:

ESPECIFICACION DEL CASO DE USO: INGRESAR USUARIO
Descripción general: Caso de uso que tiene por objetivo permitir que el actor Bibliotecario registre el ingreso de información del usuario en el Sistema.
Actores: Bibliotecario
Pre-Condiciones: Un bibliotecario activo en el sistema
Pos-Condiciones: El ingreso de la información del Usuario se haya realizado con éxito

Flujo Base o Principal

Paso	**Actor(es)**	**Sistema**
FB10	El bibliotecario ingresa la información del usuario en el sistema	
FB20		El sistema valida la información del usuario
FB30		El sistema crea un usuario
FB40		El sistema despliega un mensaje de transacción realizada satisfactoriamente

Flujos Alternativos

Paso	**Actor(es)**	**Sistema**
FA1: Actualizar información del usuario		
FA1-10	Si el bibliotecario desea actualizar la información del usuario en el sistema	
FA1-20		El sistema invoca al caso de uso actualizar información del usuario

Flujos de Error

Paso	**Actor(es)**	**Sistema**
FE1: El usuario a ingresar ya existe en el sistema		
FE1-10	El bibliotecario ingresa información de un usuario que ya existe en el sistema	
FE1-20		El sistema despliega un mensaje de: El usuario ya existe en el sistema.
FE2: La información ingresada no es válida		
FE2-10	El bibliotecario ingresa información incorrecta de un usuario en el sistema	
FE2-20		El sistema despliega un mensaje de: La información ingresada no es válida.

ESPECIFICACION DEL CASO DE USO: ACTUALIZAR USUARIO
Descripción general: Caso de uso que tiene por objetivo permitir que el actor Bibliotecario registe la actualización de la información del usuario en el Sistema.
Actores: Bibliotecario
Pre-Condiciones: Un bibliotecario activo y que el usuario haya sido creado en el sistema

Pos-Condiciones: La información del usuario haya sido actualizada exitosamente

Flujo Base o Principal

Paso	Actor(es)	Sistema
FB10	El bibliotecario ingresa los parámetros de búsqueda del usuario que desea actualizar	
FB20		El sistema busca la información del usuario
FB30		El sistema despliega la información del usuario
FB40	El bibliotecario ingresa la información del usuario a modificar	
FB50		El sistema valida la información del usuario a ser modificado.
FB60		El sistema registra la actualización de información del usuario
FB70		El sistema despliega un mensaje de transacción realizada satisfactoriamente

Flujos de Error

Paso	Actor(es)	Sistema
FE1: No se encontró el usuario a ser modificado		
FE1-10	Si el bibliotecario quiere modificar la información de un usuario que no existe en el sistema.	
FE1-20		El sistema despliega un mensaje de: El usuario no esta registrado en el sistema
FE2: La información ingresada no es válida		
FE2-10	El bibliotecario ingresa información incorrecta del usuario en el sistema	
FE2-20		El sistema despliega un mensaje de: La información ingresada no es válida.

ESPECIFICACION DEL CASO DE USO: INGRESAR ADMINISTRADOR-BIBLIOTECARIO
Descripción general: Caso de uso que tiene por objetivo permitir que el actor Administrador registre el ingreso de la información del Administrador/Bibliotecario en el Sistema.
Actores: Administrador
Pre-Condiciones: Un Administrador activo en el sistema
Pos-Condiciones: El ingreso de la información del Administrador/Bibliotecario se haya realizado con éxito

Flujo Base o Principal

Paso	Actor(es)	Sistema
FB10	El Administrador ingresa la información del administrador/bibliotecario en el sistema	
FB20		El sistema valida la información del administrador/bibliotecario

Paso	Actor(es)	Sistema
FB30		El sistema crea un administrador/bibliotecario
FB40		El sistema despliega un mensaje de transacción realizada satisfactoriamente

Flujos Alternativos

Paso	Actor(es)	Sistema
FA1: Actualizar información del administrador/bibliotecario		
FA1-10	Si el Administrador desea actualizar información del administrador/bibliotecario en el sistema	
FA1-20		El sistema invoca al caso de uso actualizar información del administrador/bibliotecario

Flujos de Error

Paso	Actor(es)	Sistema
FE1: El administrador/bibliotecario a ingresar ya existe en el sistema		
FE1-10	Si el Administrador ingresa información de un administrador/bibliotecario que ya existe en el sistema	
FE1-20		El sistema despliega un mensaje de: El administrador/bibliotecario ya existe en el sistema.
FE2: La información ingresada no es válida		
FE2-10	El Administrador ingresa información incorrecta del administrador/bibliotecario en el sistema	
FE2-20		El sistema despliega un mensaje de: La información ingresada no es válida.

ESPECIFICACION DEL CASO DE USO: ACTUALIZAR ADMINISTRADOR-BIBLIOTECARIO
Descripción general: Caso de uso que tiene por objetivo permitir que el actor Administrador registe la actualización de la información del administrador/bibliotecario en el Sistema.
Actores: Administrador
Pre-Condiciones: Un Administrador activo y que el administrador/bibliotecario haya sido ingresado en el sistema
Pos-Condiciones: La información del administrador/bibliotecario haya sido actualizada exitosamente

Flujo Base o Principal

Paso	Actor(es)	Sistema
FB10	El Administrador ingresa los parámetros de búsqueda del administrador/bibliotecario que desea actualizar	
FB20		El sistema busca el administrador/bibliotecario

Paso	Actor(es)	Sistema
FB30		El sistema despliega la información del administrador/bibliotecario
FB40	El administrador ingresa la información del administrador/bibliotecario a modificar	
FB50		El sistema valida la información del administrador/bibliotecario a ser modificado
FB60		El sistema registra la actualización de información del administrador/bibliotecario
FB70		El sistema despliega un mensaje de transacción realizada satisfactoriamente

Flujos de Error

Paso	Actor(es)	Sistema
FE1: No se encontró el administrador/bibliotecario a ser modificado		
FE1-10	Si el Administrador quiere modificar la información de un administrador/bibliotecario que no existe en el sistema.	
FE1-20		El sistema despliega un mensaje de: El administrador/bibliotecario no esta registrado en el sistema.
FE2: La información ingresada no es válida		
FE2-10	El Administrador ingresa información incorrecta del administrador/bibliotecario en el sistema	
FE2-20		El sistema despliega un mensaje de: La información ingresada no es válida.

ESPECIFICACION DEL CASO DE USO: INGRESAR DOCUMENTO Y/O EJEMPLAR
Descripción general: Caso de uso que tiene por objetivo permitir que el actor Bibliotecario ingrese un nuevo material bibliográfico (Documento - Ejemplar). En el caso de que se ingresa un nuevo documento se crea al mismo tiempo el ejemplar, adicionalmente la aplicación permite ingresar más ejemplares relacionados con el mismo Documento
Actores: Bibliotecario
Pre-Condiciones: Que el módulo de Administración de Documentos del Sistema esté disponible para el actor Bibliotecario y debe estar activo en el sistema.
Pos-Condiciones: La información del Documento-Ejemplar sea ingresada exitosamente

Flujo Base o Principal

Paso	Actor(es)	Sistema
FB10	El Bibliotecario ingresa la información del Documento y/o Ejemplar	
FB20		El sistema valida la información del Documento y/o Ejemplar

Paso	Actor(es)	Sistema
FB30		El sistema crea el documento y/o ejemplar; y genera código para identificación del Documento y/o Ejemplar.
FB40		El sistema despliega la información del Documento y/o Ejemplar creado.
FB50		El sistema despliega mensaje de Transacción realizada satisfactoriamente.

Flujos Alternativos

Paso	Actor(es)	Sistema
FA1: Bibliotecario desea actualizar Documento y/o Ejemplar		
FA1-10	Si el bibliotecario desea actualizar la información del Documento y/o Ejemplar.	
FA1-20		El sistema invoca al caso de uso Actualizar Información del Documento y/o Ejemplar.
FA2: Bibliotecario desea generar Reporte de Estado del Ejemplar		
FA2-10	Si el Bibliotecario desea obtener un reporte que describa el estado del Ejemplar.	
FA2-20		El sistema invoca al caso de uso Generar Reporte de Estado del Ejemplar.

Flujos de Error

Paso	Actor(es)	Sistema
FE1: El Bibliotecario ingresa un Documento que ya existente		
FE1-10	Si el Bibliotecario ingresa El Título de un Documento que ya existe.	
FE1-20		El Sistema despliega un mensaje que indica que el Documento ya ha sido ingresado.
FE2: El Bibliotecario ingresa información que no es válida		
FE2-10	El bibliotecario ingresa información incorrecta del Documento y/o Ejemplar en el sistema	
FE2-20		El sistema despliega un mensaje de: La información ingresada no es válida.

ESPECIFICACION DEL CASO DE USO: ACTUALIZAR DOCUMENTO Y/O EJEMPLAR
Descripción general: Caso de uso que tiene por objetivo permitir que el actor Bibliotecario actualice el Documento y/o Ejemplar; dependiendo de si se desea actualizar el Documento y/o Ejemplar
Actores: Bibliotecario
Pre-Condiciones: El Módulo de Administración de Documentos del Sistema esté

disponible para el actor Bibliotecario y debe estar activo en el sistema
Pos-Condiciones: La información del documento y/o ejemplar sea actualizada exitosamente

Flujo Base o Principal

Paso	Actor(es)	Sistema
FB10	El Bibliotecario ingresa los parámetros de búsqueda del Documento y/o que desea actualizar	
FB20		El sistema busca la información del documento y de los ejemplares disponibles
FB30		El sistema despliega la información del documento y de los ejemplares disponibles.
FB40	El Bibliotecario ingresa la información del Documento y/o Ejemplar a modificar	
FB50		El sistema valida la información del Documento y/o Ejemplar a ser modificado
FB60		El Sistema registrar la actualización de la información del Documento y/o Ejemplar.
FB70		Despliega mensaje de Transacción realizada satisfactoriamente.

Flujos Alternativos

Paso	Actor(es)	Sistema
FA1: Bibliotecario desea ingresar Documento y/o Ejemplar		
FA1-10	Si el bibliotecario desea ingresar un nuevo Documento y/o Ejemplar.	
FA1-20		El sistema invoca al caso de uso Ingresar Documento y/o Ejemplar.
FA2: Bibliotecario desea generar Reporte de Estado del Ejemplar		
FA2-10	Si el Bibliotecario desea obtener un reporte que describa el estado del Ejemplar.	
FA2-20		El sistema invoca al caso de uso Generar Reporte de Estado del Ejemplar.

Flujos de Error

Paso	Actor(es)	Sistema
FE1: No se encuentra el Documento que se va a Actualizar		
FE1-10	Si el documento que se desea actualizar no se encuentra registrado en el sistema.	
FE1-20		El sistema presenta un mensaje de que no se pudo encontrar el documento.
FE2: La información ingresada no es válida		

FE2-10	El bibliotecario ingresa información incorrecta del Documento y/o Ejemplar en el sistema	
FE2-20		El sistema despliega un mensaje de: La información ingresada no es válida.

ESPECIFICACION DEL CASO DE USO: PRESTAMO DE EJEMPLAR
Descripción general: Caso de uso que tiene por objetivo permitir al actor Bibliotecario registrar el préstamo de un ejemplar en el Sistema.
Actores: Bibliotecario
Pre-Condiciones: El ejemplar no este prestado ni reservado en el sistema
Pos-Condiciones: El bibliotecario realizó el préstamo de un ejemplar al usuario exitosamente

Flujo Base o Principal

Paso	Actor(es)	Sistema
FB10	El bibliotecario ingresa los parámetros de búsqueda del Documento-Ejemplar a ser prestado	
FB20		El sistema busca la información del ejemplar
FB30		El sistema verifica el estado del préstamo del ejemplar
FB40		El sistema verifica si existe reservación del ejemplar
FB50		El sistema despliega disponibilidad del ejemplar
FB60	El bibliotecario ingresa la información del préstamo	
FB70		El sistema valida la información del préstamo.
FB80		El sistema crea el préstamo del ejemplar
FB90		El sistema despliega un mensaje de transacción realizada satisfactoriamente

Flujos Alternativos

Paso	Actor(es)	Sistema
FA1: Renovar préstamo		
FA1-10	Si el bibliotecario desea renovar el préstamo de un ejemplar	
FA1-20		El sistema invoca al caso de uso renovar préstamo
FA2: Devolver Ejemplar		
FA2-10	Si el bibliotecario desea devolver el ejemplar prestado	
FA2-20		El sistema invoca al caso de uso devolver ejemplar.

Flujos de Error

Paso	Actor(es)	Sistema
FE1: Cuando el ejemplar esta prestado		
FE1-10	Si el bibliotecario va a realizar el préstamo de un ejemplar que ya esta prestado	
FE1-20		El sistema despliega un mensaje de: El ejemplar se encuentra prestado
FE2: Cuando el ejemplar esta reservado		
FE2-10	Si el bibliotecario va a realizar el préstamo de un ejemplar que esta reservado por otro usuario	
FE2-20		El sistema despliega un mensaje de: El ejemplar se encuentra reservado
FE3: La información ingresada no es válida		
FE3-10	El bibliotecario ingresa información incorrecta del préstamo en el sistema	
FE3-20		El sistema despliega un mensaje de: La información ingresada no es válida.

ESPECIFICACION DEL CASO DE USO: RENOVAR PRÉSTAMO
Descripción general: Caso de uso que tiene por objetivo permitir que el actor Bibliotecario registre la renovación del Préstamo de un ejemplar
Actores: Bibliotecario
Pre-Condiciones: El Ejemplar debe estar prestado al mismo usuario
Pos-Condiciones: El bibliotecario realice la renovación del préstamo exitosamente.

Flujo Base o Principal

Paso	Actor(es)	Sistema
FB10	El Bibliotecario ingresa los parámetros de búsqueda del Ejemplar prestado	
FB20		El sistema busca la información del Préstamo
FB30		El sistema despliega la información del Ejemplar Prestado
FB40	El Bibliotecario ingresa la información de la renovación del Préstamo.	
FB50		El sistema valida la información de la renovación del Préstamo
FB60		El Sistema registra la renovación del Ejemplar.
FB70		El sistema despliega un mensaje de Transacción realizada satisfactoriamente.

Flujos Alternativos

Paso	Actor(es)	Sistema
FA1: Pago de Multa		
FA1-10	Si el bibliotecario encuentra que el préstamo del Ejemplar tiene multas pendientes	
FA1-20		El sistema invoca al caso de uso Registrar Pago de Multa

Flujos de Error

Paso	Actor(es)	Sistema
FE1: El Bibliotecario realiza la renovación de Ejemplar Multado		
FE1-10	Si el Bibliotecario ingresa la Devolución de un Ejemplar que tiene multa pendiente.	
FE1-20		El sistema despliega un mensaje que indica que debe realizar primero el pago de multas para realizar la devolución de un Ejemplar.
FE2: La información ingresada no es válida		
FE2-10	El bibliotecario ingresa información incorrecta de la renovación del préstamo en el sistema.	
FE2-20		El sistema despliega un mensaje de: La información ingresada no es válida.

ESPECIFICACION DEL CASO DE USO: DEVOLVER EJEMPLAR
Descripción general: Caso de uso que tiene por objetivo permitir que el actor Bibliotecario registre la devolución del ejemplar.
Actores: Bibliotecario
Pre-Condiciones: El Documento haya sido prestado
Pos-Condiciones: el actor Bibliotecario registra exitosamente la devolución del Documento

Flujo Base o Principal

Paso	Actor(es)	Sistema
FB10	El Bibliotecario ingresa parámetros de búsqueda del Ejemplar a ser devuelto.	
FB20		El sistema verifica si el ejemplar tiene multas.
FB30		Despliega el ejemplar prestado al Usuario
FB40	Ingresa la información de la devolución del Ejemplar	
FB50		El sistema valida la información de la devolución del Ejemplar
FB60		El Sistema registra la devolución del Ejemplar
FB70		Despliega mensaje de Transacción realizada satisfactoriamente.

Flujos Alternativos

Paso	Actor(es)	Sistema
FA1: Pagar Multa		
FA1-10	Si el Bibliotecario encuentra que el Préstamo del Ejemplar tiene multas pendientes	
FA1-20		El sistema invoca al caso de uso Registrar Pago de Multa
FA2: Prestar Ejemplar		
FA2-10	Si el Bibliotecario desea realizar el Préstamo de un Ejemplar	
FA2-20		El sistema invoca al caso de uso Préstamo de Ejemplar

Flujos de Error

Paso	Actor(es)	Sistema
FE1: El Bibliotecario realiza la devolución de un Préstame Multado		
FE1-10	Si el Bibliotecario ingresa la información de la Devolución de un Ejemplar que tiene multa pendiente.	
FE1-20		El sistema despliega un mensaje que indica que se debe ingresar primero el pago de multa para realizar la devolución del Ejemplar.
FE2: La información ingresada no es válida		
FE2-10	El bibliotecario ingresa información incorrecta de la devolución del préstamo de un ejemplar en el sistema.	
FE2-20		El sistema despliega un mensaje de: La información ingresada no es válida.

ESPECIFICACION DEL CASO DE USO: REGISTRAR PAGO DE MULTA
Descripción general: Caso de uso que tiene por objetivo permitir que el actor Bibliotecario ingrese en el Sistema los Pagos de Multas
Actores: Bibliotecario
Pre-Condiciones: El Usuario va a devolver o renovar y tiene multas pendientes del ejemplar
Pos-Condiciones: El actor Bibliotecario ingresa el pago de la multa exitosamente

Flujo Base o Principal

Paso	Actor(es)	Sistema
FB10		El sistema calcula el valor de la multa
FB20		Despliega el valor de la multa
FB30	El Bibliotecario ingresa el motivo de la multa	
FB40		El sistema registra el pago de la Multa

Paso	Actor(es)	Sistema
FB50		El sistema despliega mensaje de Transacción realizada satisfactoriamente.

ESPECIFICACION DEL CASO DE USO: VISUALIZAR CONTENIDO DE UN DOCUMENTO
Descripción general: Caso de uso que tiene por objetivo permitir a los actores Usuario y Bibliotecario visualizar el contenido de un documento en el Sistema.
Actores: Usuario
Pre-Condiciones: Debe existir el documento en el Sistema
Pos-Condiciones: El sistema haya desplegado exitosamente el contenido de un documento.

Flujo Base o Principal

Paso	Actor(es)	Sistema
FB10	El Usuario ingresa los parámetros de búsqueda del documento a ser visualizado	
FB20		El sistema busca la información del documento
FB30		El sistema despliega el documento encontrado
FB40	El Usuario desea visualizar el contenido e información del documento	
FB50		El sistema despliega el contenido e información del documento

Flujos de Error

Paso	Actor(es)	Sistema
FE1: El sistema no puede desplegar el contenido e información del documento		
FE1-10	Si el usuario desea visualizar el contenido e información del documento y no lo puede desplegar	
FE1-20		El sistema despliega un mensaje que indica que el sistema no pudo desplegar la información del documento

ESPECIFICACION DEL CASO DE USO: RESERVACIÓN DE EJEMPLAR
Descripción general: Caso de uso que tiene por objetivo permitir que el actor Usuario registre la reservación de un Ejemplar en el Sistema.
Actores: Usuario
Pre-Condiciones: El Ejemplar debe existir y debe estar prestado.
Pos-Condiciones: El usuario ha realizado exitosamente la reservación de un ejemplar en el sistema

Flujo Base o Principal

Paso	Actor(es)	Sistema
FB10	El usuario ingresa los parámetros de búsqueda del Documento-Ejemplar a ser reservado	
FB20		El sistema busca la información del Ejemplar a ser reservado
FB30		El sistema despliega la información de los ejemplares encontrados.
FB40	El usuario ingresa la información de la reservación del ejemplar.	
FB50		El sistema valida la información de la reservación del ejemplar
FB60		El sistema verifica el estado del préstamo del ejemplar.
FB70		El sistema registra la reservación del ejemplar.
FB80		El sistema despliega información de transacción realizada satisfactoriamente.

Flujos Alternativos

Paso	Actor(es)	Sistema
FA1: Usuario desea visualizar el contenido e información del documento		
FA1-10	Si el usuario desea desplegar el contenido e información del documento	
FA1-20		El sistema invoca al caso de uso visualizar contenido e información del documento
FA2: Préstamo de Ejemplar		
FA2-10	Si el bibliotecario realiza el préstamo de un ejemplar	
FA2-20		El sistema invoca al caso de uso Préstamo de Ejemplar

Flujos de Error

Paso	Actor(es)	Sistema
FE1: El Ejemplar esta reservado para la misma fecha		
FE1-10	Si el Usuario desea reservar un ejemplar para la misma fecha en la que ya ha sido reservado	
FE1-20		El sistema despliega un mensaje de ejemplar ya ha sido reservado
FE2: El usuario desea Reservar un documento para una fecha anterior a la actual		
FE2-10	Si el Usuario desea reservar un ejemplar para una fecha anterior a la actual	
FE2-20		El sistema despliega un mensaje de el ejemplar no puede ser reservado en esta fecha

FE3: La información ingresada no es válida		
FE3-10	El usuario ingresa información incorrecta de la reservación del ejemplar en el sistema.	
FE3-20		El sistema despliega un mensaje de: La información ingresada no es válida.

ESPECIFICACION DEL CASO DE USO: INGRESAR IDIOMA
Descripción general: Caso de uso que tiene por objetivo permitir que el actor Bibliotecario ingrese la descripción de Idiomas que serán utilizados para el registro de Documentos.
Actores: Bibliotecario
Pre-Condiciones: El Bibliotecario debe estar activo en el Sistema
Pos-Condiciones: el actor Bibliotecario ingresa exitosamente el Idioma

Flujo Base o Principal

Paso	Actor(es)	Sistema
FB10	El Bibliotecario ingresa el Idioma.	
FB20		El sistema crea el Idioma
FB30		El sistema despliega un mensaje de Transacción realizada satisfactoriamente.

Flujos Alternativos

Paso	Actor(es)	Sistema
FA1: Bibliotecario desea actualizar Idioma		
FA1-10	Si Bibliotecario desea actualizar el idioma	
FA1-20		El sistema invoca al caso de uso Actualizar Idioma

ESPECIFICACION DEL CASO DE USO: ACTUALIZAR IDIOMA
Descripción general: Caso de uso que tiene por objetivo permitir que el actor Bibliotecario registre la actualización de la descripción del Idioma en el sistema.
Actores: Bibliotecario
Pre-Condiciones: El Bibliotecario debe estar activo en el Sistema
Pos-Condiciones: El actor Bibliotecario actualiza exitosamente la descripción del Idioma

Flujo Base o Principal

Paso	Actor(es)	Sistema
FB10	El Bibliotecario ingresa el idioma a ser actualizado	
FB20		El Sistema busca el Idioma
FB30		Despliega la información del Idioma
FB40	El Bibliotecario ingresa la actualización de la información del Idioma	
FB50		El sistema registra la actualización de la información del Idioma

Paso	Actor(es)	Sistema
FB60		El sistema despliega un mensaje de Transacción realizada satisfactoriamente.

Flujos Alternativos

Paso	Actor(es)	Sistema
FA1: Bibliotecario desea ingresar Idioma		
FA1-10	Si Bibliotecario desea ingresar un idioma	
FA1-20		El sistema hace la llamada al caso de Uso Ingresar Idioma

Flujos de Error

Paso	Actor(es)	Sistema
FE2: El Idioma no pudo ser actualizado		
FE2-10	El Bibliotecario ingresa el idioma a ser actualizado y el sistema no puedo encontrarlo.	
FE2-20		El sistema despliega un mensaje de: El Idioma no pudo ser encontrado.

ESPECIFICACION DEL CASO DE USO: INGRESAR AUTOR
Descripción general: Caso de uso que tiene por objetivo permitir que el actor Bibliotecario registre el ingreso de la información del Autor en el Sistema.
Actores: Bibliotecario
Pre-Condiciones: El bibliotecario este activo en el sistema
Pos-Condiciones: La información del autor se haya ingresado exitosamente.

Flujo Base o Principal

Paso	Actor(es)	Sistema
FB10	El bibliotecario ingresa la información del autor	
FB20		El sistema valida la información del autor
FB30		El sistema crea el registro del autor en el sistema
FB40		El sistema despliega un mensaje de transacción realizada exitosamente

Flujos Alternativos

Paso	Actor(es)	Sistema
FA1: Actualizar información del autor		
FA1-10	Si el bibliotecario desea actualizar la información del autor en el sistema	
FA1-20		El sistema invoca al caso de uso Actualizar información del Autor

Flujos de Error

Paso	Actor(es)	Sistema
FE1: La información del autor a ingresar ya existe en el sistema		

FE1-10	Si el bibliotecario ingresa información de un autor que ya existe en el sistema	
FE1-20		El sistema despliega un mensaje de: El autor ya existe en el sistema.
FE2: La información ingresada no es válida		
FE2-10	El Bibliotecario ingresa información incorrecta del autor en el sistema.	
FE2-20		El sistema despliega un mensaje de: La información ingresada no es válida.

ESPECIFICACION DEL CASO DE USO: ACTUALIZAR AUTOR
Descripción general: Caso de uso que tiene por objetivo permitir que el actor Bibliotecario registre la actualización de la información del Autor en el Sistema.
Actores: Bibliotecario
Pre-Condiciones: Que exista el autor en el sistema
Pos-Condiciones: La información del autor se haya modificado exitosamente.

Flujo Base o Principal

Paso	Actor(es)	Sistema
FB10	El bibliotecario ingresa los parámetros de búsqueda del autor a ser actualizado	
FB20		El sistema busca la información del autor a ser actualizado
FB30		El sistema despliega la información del autor
FB40	El bibliotecario ingresa la información del autor a ser actualizada	
FB50		El sistema valida la información del autor a ser modificada
FB60		Registra la actualización de la información del autor
FB70		El sistema despliega un mensaje de transacción realizada satisfactoriamente

Flujos Alternativos

Paso	Actor(es)	Sistema
FA1: Ingresar información del autor		
FA1-10	Si el bibliotecario desea ingresar información del autor en el sistema	
FA1-20		El sistema invoca al caso de uso Ingresar información del autor

Flujos de Error

Paso	Actor(es)	Sistema
FE1: La información ingresada no es válida		

FE1-10	El Bibliotecario ingresa información incorrecta en la actualización del autor en el sistema.	
FE1-20		El sistema despliega un mensaje de: La información ingresada no es válida.
FE2: El autor no pudo ser encontrado		
FE2-10	El Bibliotecario ingresa los parámetros de búsqueda del autor y el sistema no lo pudo encontrar.	
FE2-20		El sistema despliega un mensaje de error: El autor no pudo ser encontrado

ESPECIFICACION DEL CASO DE USO: ACTUALIZAR CATEGORÍA
Descripción general: Caso de uso que tiene por objetivo permitir que el actor Bibliotecario registre la actualización de la información de la categoría en el sistema.
Actores: Bibliotecario
Pre-Condiciones: El Módulo de Administración de Documentos del Sistema esta disponible para el actor Bibliotecario y este debe estar activo en el sistema
Pos-Condiciones: El Bibliotecario actualiza la Categoría

Flujo Base o Principal

Paso	Actor(es)	Sistema
FB10	El Bibliotecario ingresa los parámetros de búsqueda de la Categoría	
FB20		El Sistema busca la Categoría a ser actualizada.
FB30		El sistema despliega la información de la Categoría
FB40	Bibliotecario ingresa la descripción de la Categoría a ser actualizada	
FB50		El sistema valida la información de la categoría a ser modificada
FB60		El sistema registra la actualización de la Categoría
FB70		El sistema despliega un mensaje de Transacción realizada satisfactoriamente.

Flujos Alternativos

Paso	Actor(es)	Sistema
FA1: Bibliotecario desea ingresar Categoría		
FA1-10	Si el bibliotecario desea ingresar una nueva Categoría.	
FA1-20		El sistema invoca al caso de uso ingresar Categoría.

Flujos de Error

Paso	Actor(es)	Sistema
FE1: La información ingresada no es válida		

FE1-10	El Bibliotecario ingresa información incorrecta en la actualización de la categoría en el sistema.	
FE1-20		El sistema despliega un mensaje de: La información ingresada no es válida.
FE2: La categoría no pudo ser encontrada		
FE2-10	El Bibliotecario ingresa los parámetros de búsqueda de la categoría a ser actualizada y el sistema no la pudo encontrar	
FE2-20		El sistema despliega un mensaje de error: La categoría no pudo ser ingresada.

ESPECIFICACION DEL CASO DE USO: REPORTE DE USUARIOS MULTADOS.
Descripción general: Caso de uso que tiene por objetivo permitir que los actores Administrador y Bibliotecario disponga del reporte de Usuarios Multados.
Actores: Administrador y Bibliotecario.
Pre-Condiciones: Deben existir Usuarios que tengan multas pendientes.
Pos-Condiciones: El Sistema imprime exitosamente el reporte de Usuarios Multados.

Flujo Base o Principal

Paso	Actor(es)	Sistema
FB10	El Administrador/Bibliotecario ingresa parámetros necesarios para generar el reporte	
FB20		El Sistema genera el reporte de Usuarios Multados
FB30		Imprime Reporte

Flujos de Error

Paso	Actor(es)	Sistema
FE1: Los parámetros ingresados en el reporte no son válidos		
FE1-10	El Administrador/Bibliotecario ingresa información incorrecta en los parámetros del reporte	
FE1-20		El sistema despliega un mensaje de: Los parámetros ingresados no son válidos.

ESPECIFICACION DEL CASO DE USO: EJEMPLARES NO DEVUELTOS A TIEMPO
Descripción general: Caso de uso que tiene por objetivo permitir a los actores Administrador y Bibliotecario consultar los Ejemplares no devueltos a tiempo.
Actores: Administrador y Bibliotecario
Pre-Condiciones: El Ejemplar no sea devuelto a tiempo por el Usuario
Pos-Condiciones: La consulta de un Ejemplar no Devuelto a Tiempo se realice exitosamente

Flujo Base o Principal

Paso	Actor(es)	Sistema

Paso	Actor(es)	Sistema
FB10	El Administrador/Bibliotecario ingresa los parámetros de búsqueda de Ejemplares no devueltos a tiempo	
FB20		El sistema verifica la fecha de caducidad del préstamo
FB30		El sistema despliega los Ejemplares no devueltos a tiempo

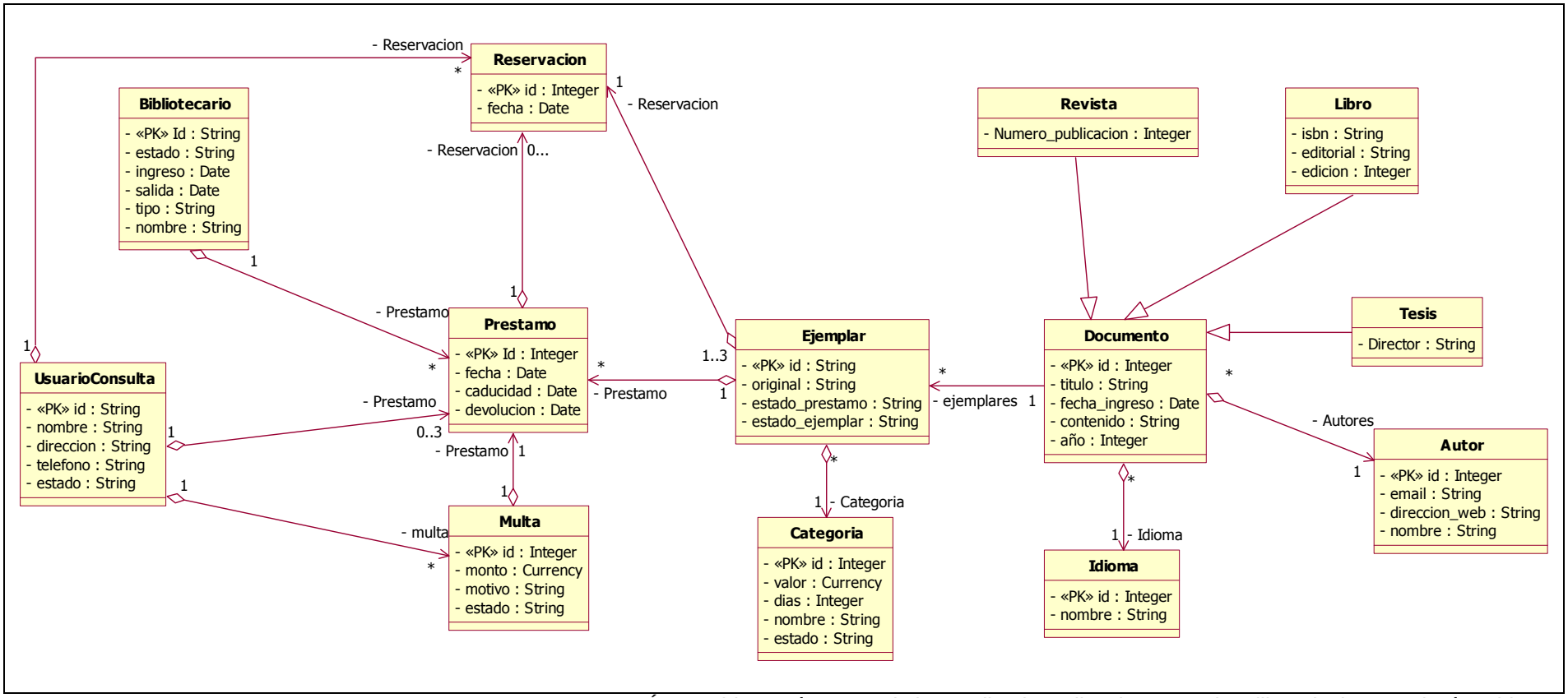
Flujos de Error

Paso	Actor(es)	Sistema
FE1: Los parámetros ingresados en la consulta no son válidos		
FE1-10	El Administrador/Bibliotecario ingresa información incorrecta en los parámetros de la consulta	
FE1-20		El sistema despliega un mensaje de: Los parámetros ingresados no son válidos.

3.3.3.5 Diagrama de Clases de Análisis

El diagrama de clases de análisis permite identificar: clases, relaciones entre clases. En la Figura 3-13 se muestra el diagrama de clases de análisis de SIABI

FIGURA 3-13: DIAGRAMA DE CLASES DE ANALISIS

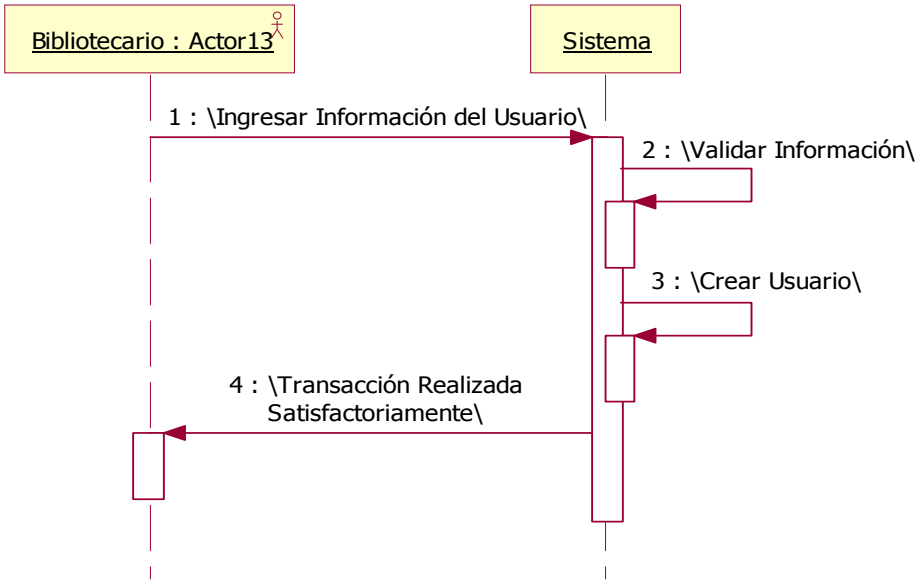


ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006

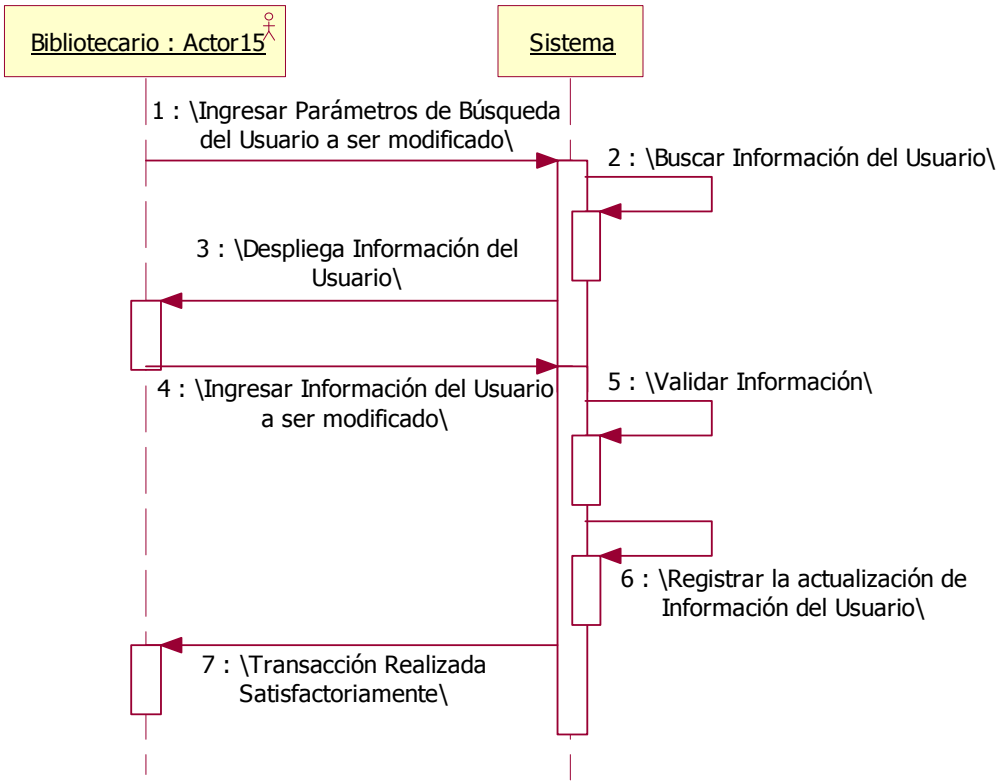
3.3.3.6 Diagrama de Secuencias

Para modelar la interacción entre los actores del sistema y la aplicación utilizamos como estrategia los diagramas de secuencia.

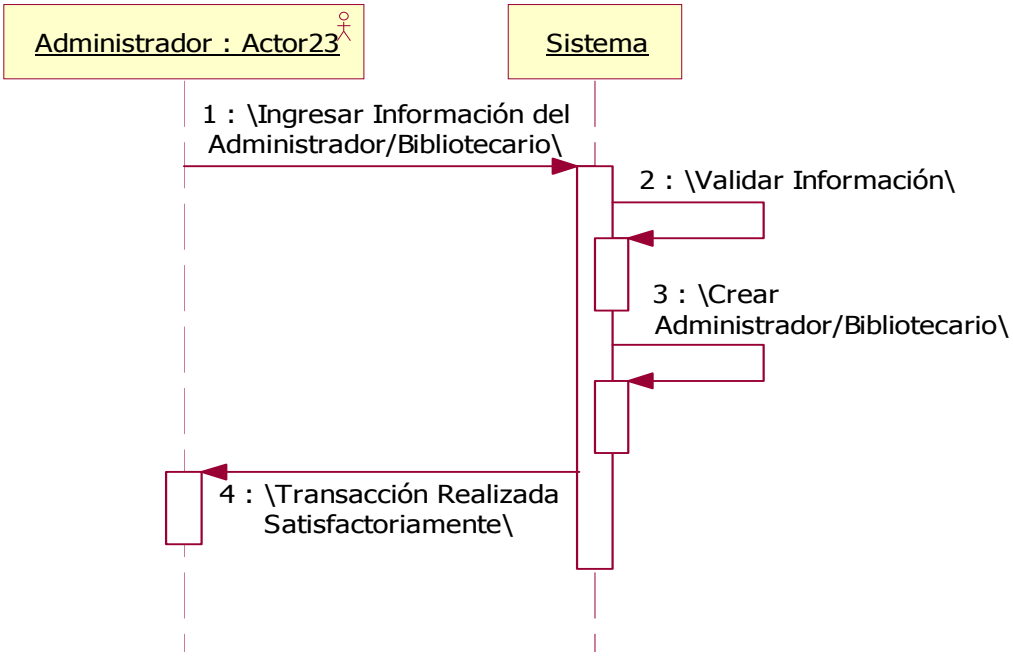
INGRESAR INFORMACIÓN DE USUARIO



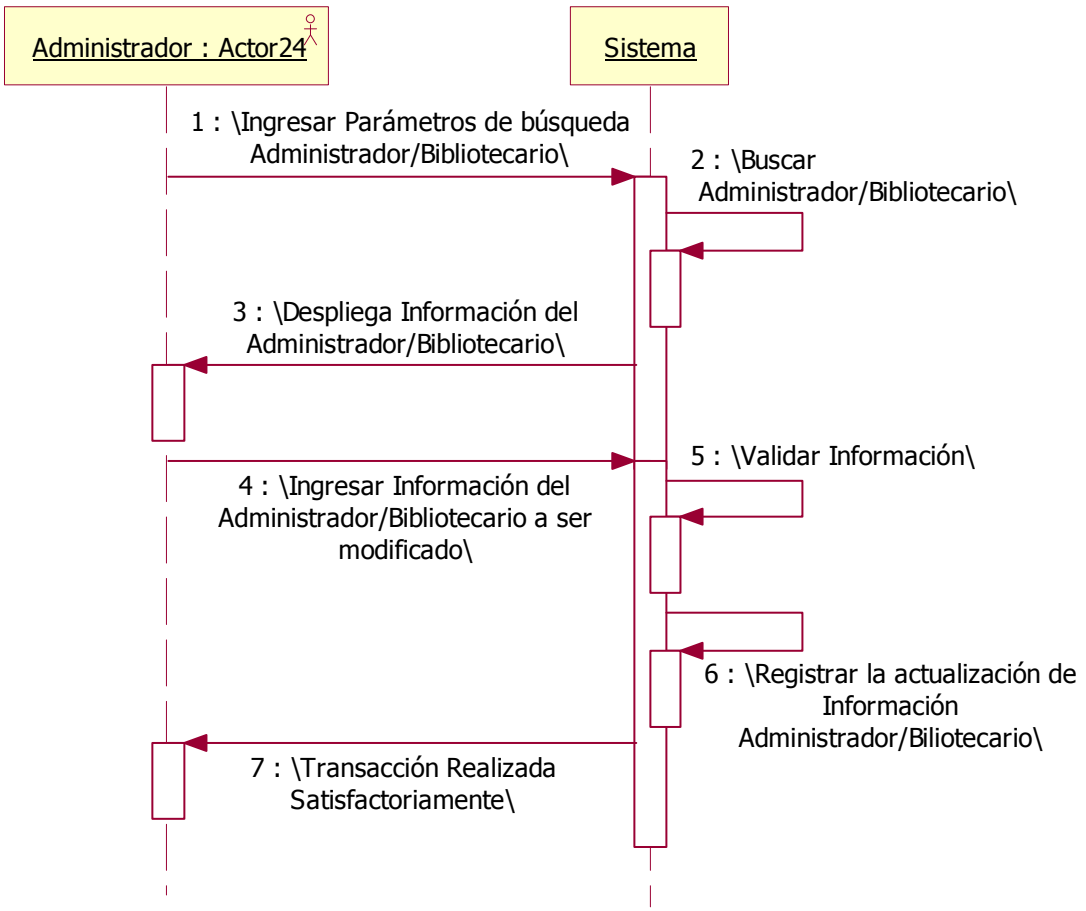
ACTUALIZAR INFORMACIÓN DE USUARIO



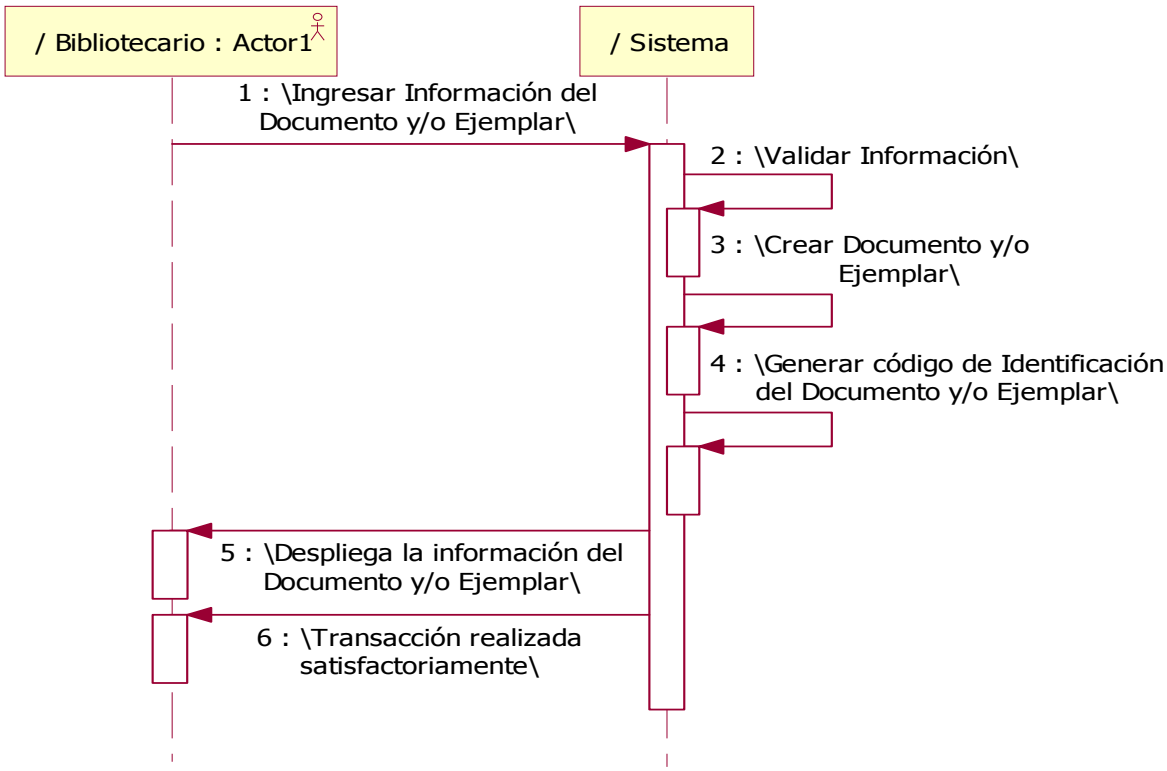
INGRESAR INFORMACIÓN DE ADMINISTRADOR/BIBLIOTECARIO



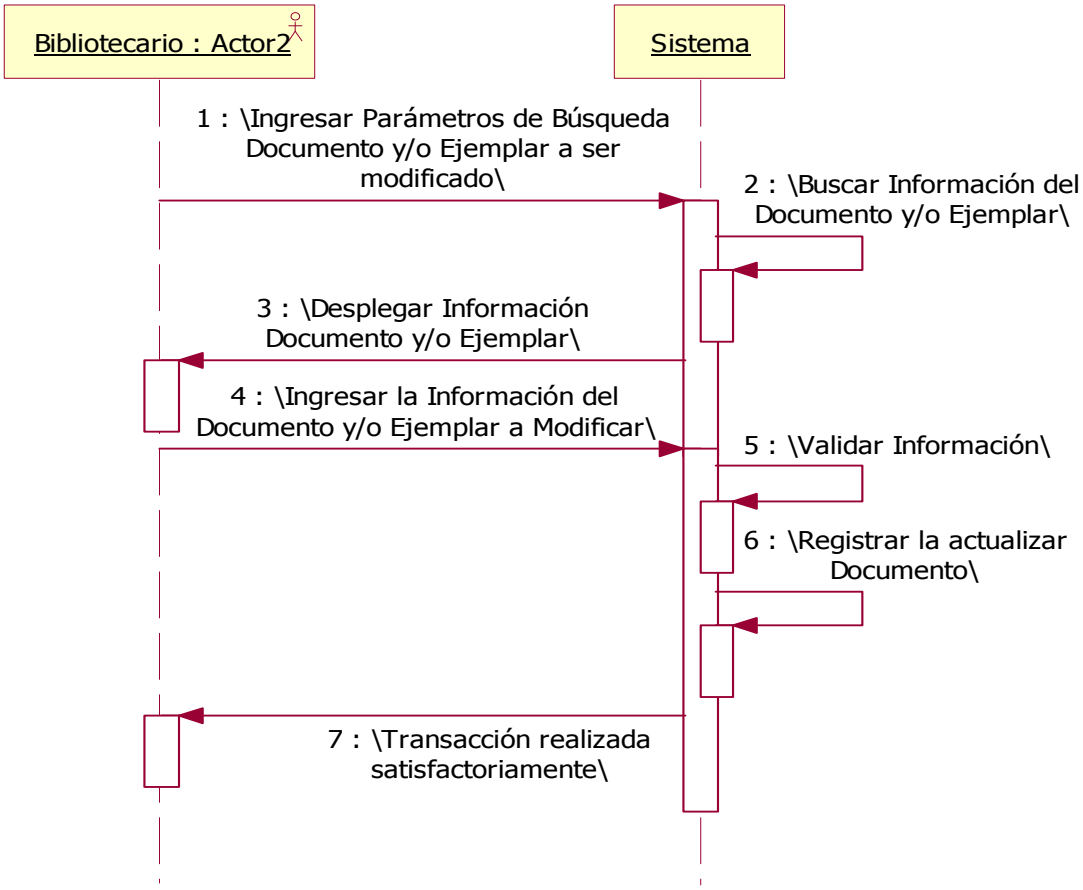
ACTUALIZAR INFORMACIÓN DEL ADMINISTRADOR/BIBLIOTECARIO



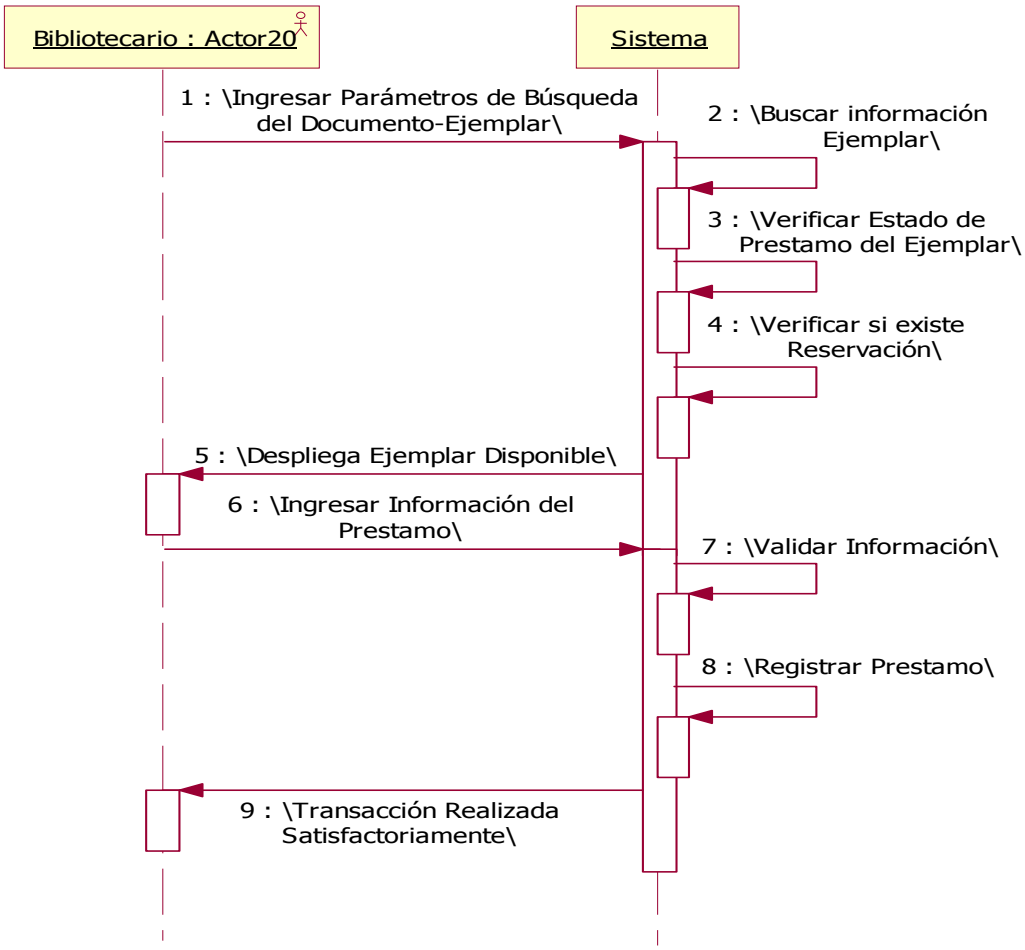
INGRESAR DOCUMENTO Y/O EJEMPLAR



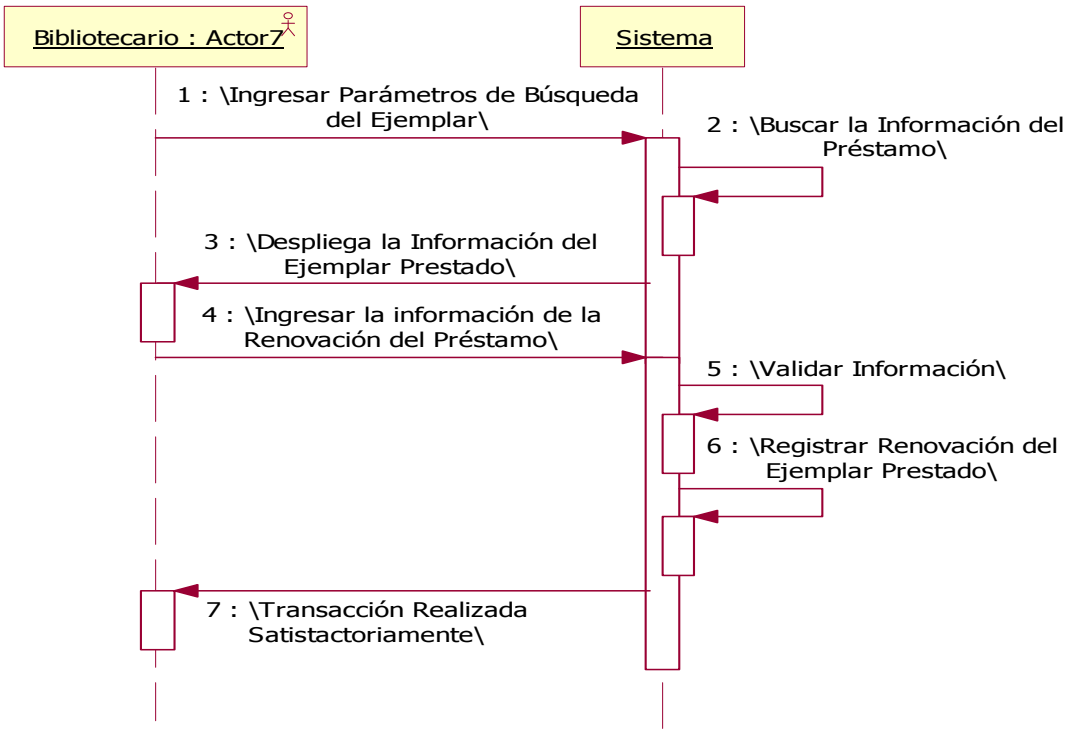
ACTUALIZAR DOCUMENTO Y/O EJEMPLAR



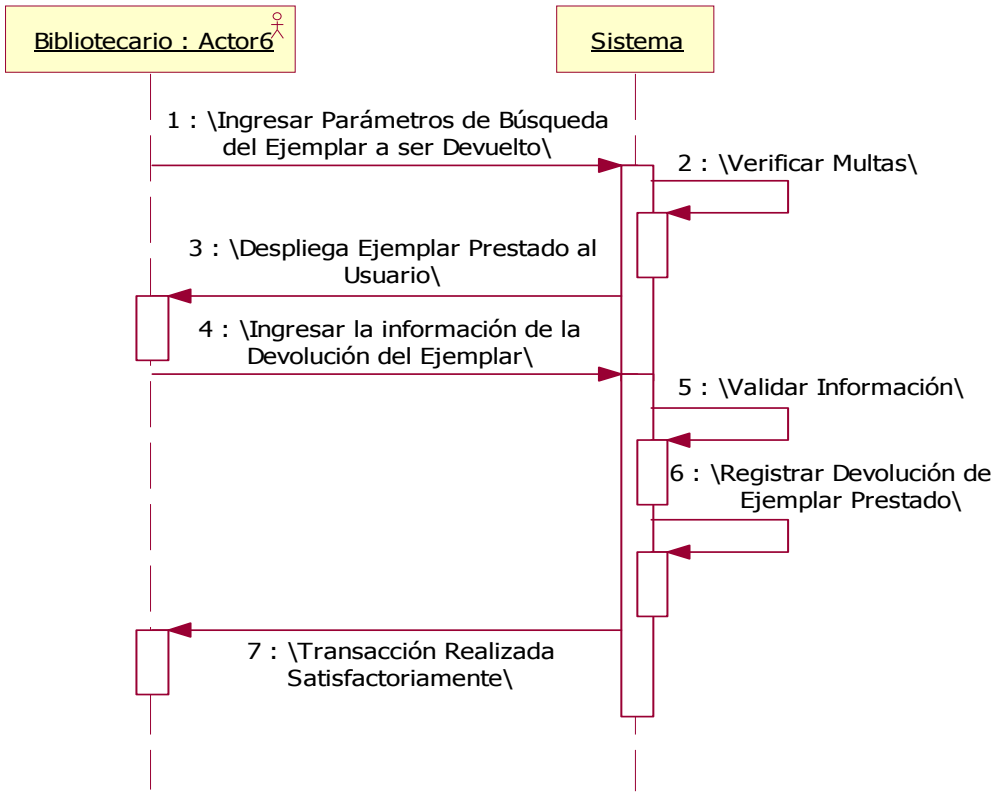
PRESTAMO DE EJEMPLARES



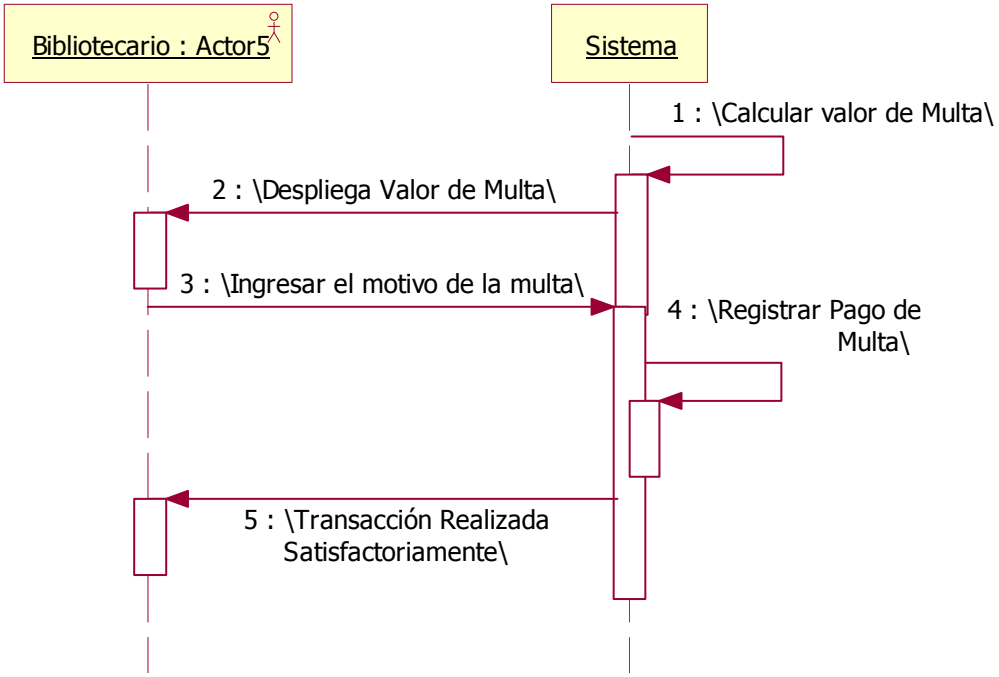
RENOVAR PRESTAMO



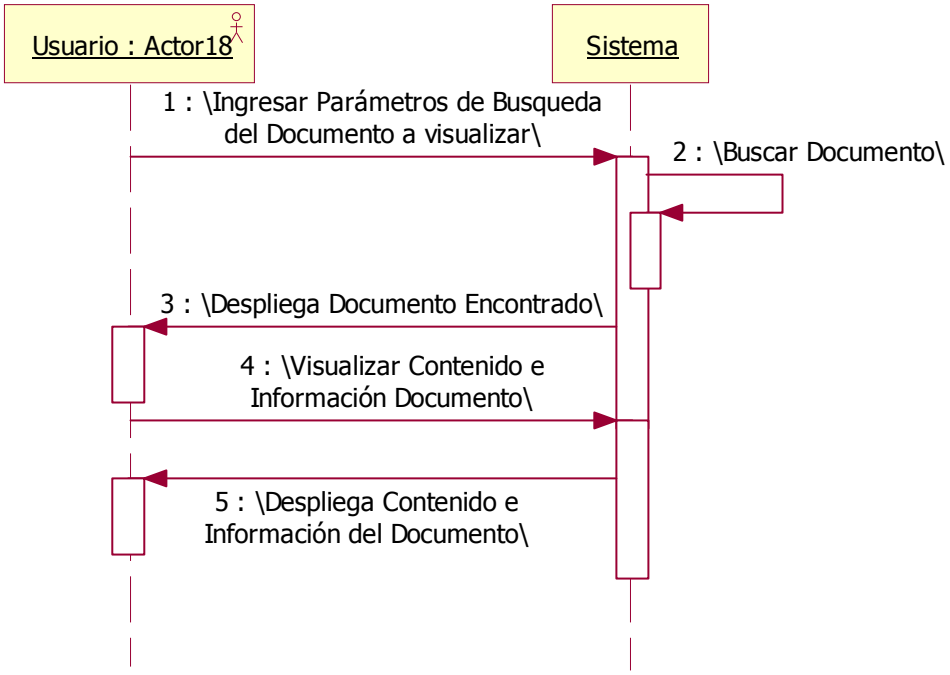
DEVOLVER DOCUMENTO



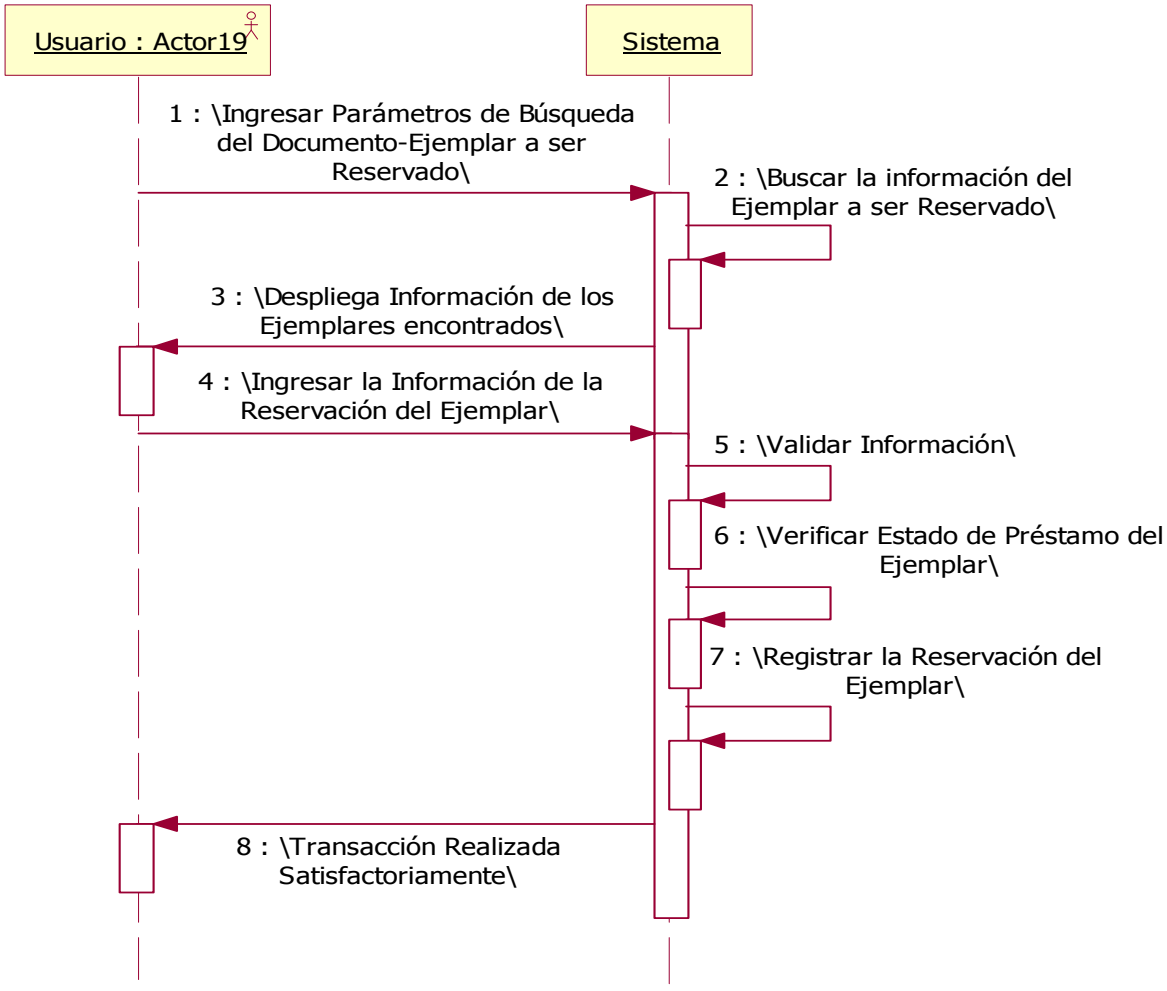
REGISTRAR PAGO DE MULTA



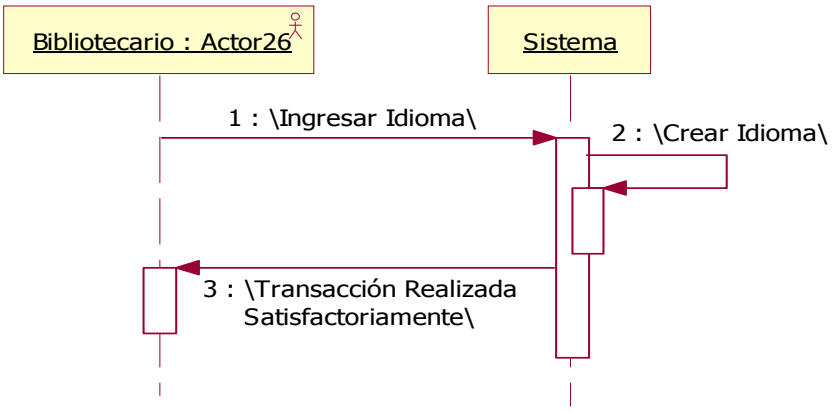
VISUALIZAR CONTENIDO DEL DOCUMENTO



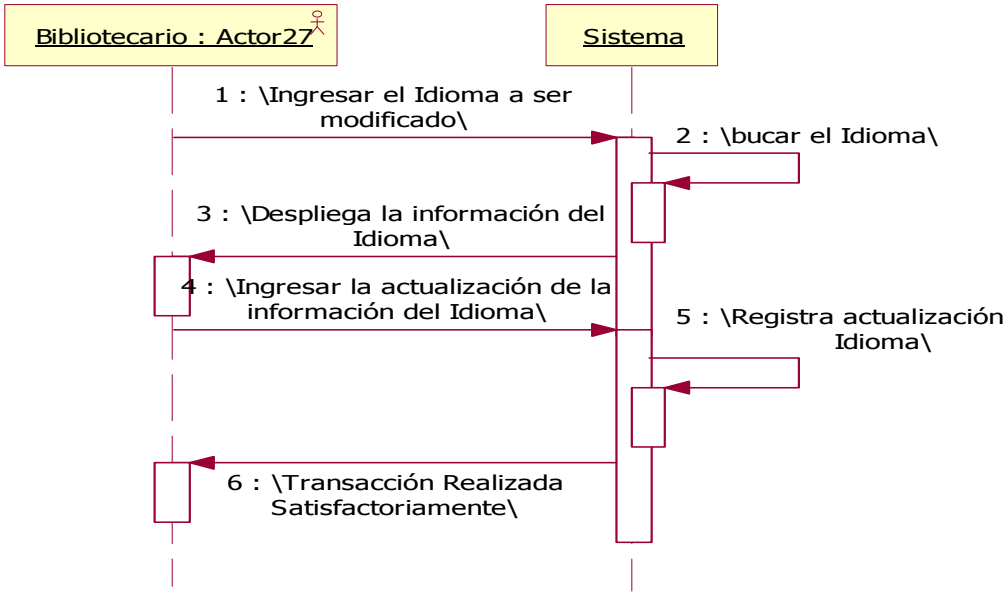
RESERVACIÓN DEL DOCUMENTO



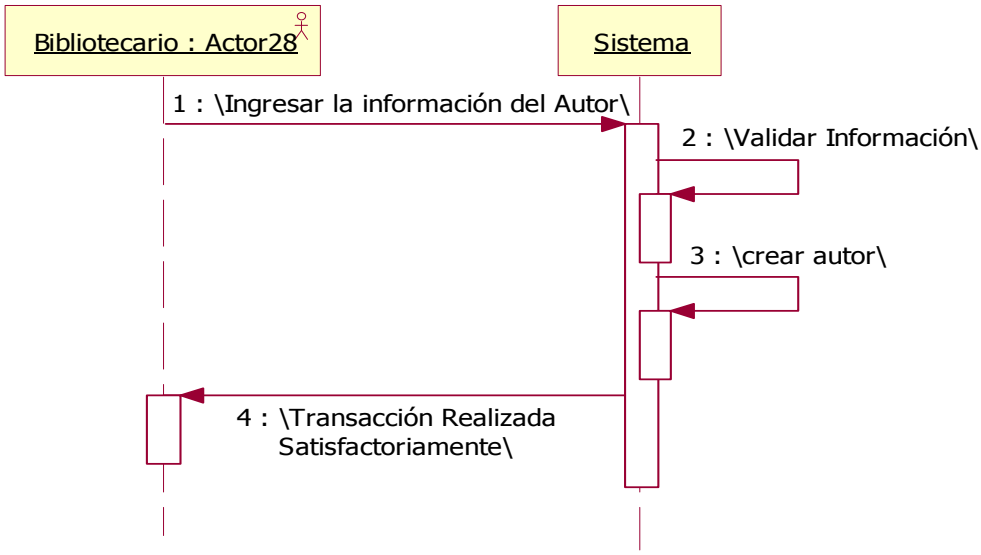
INGRESAR IDIOMA



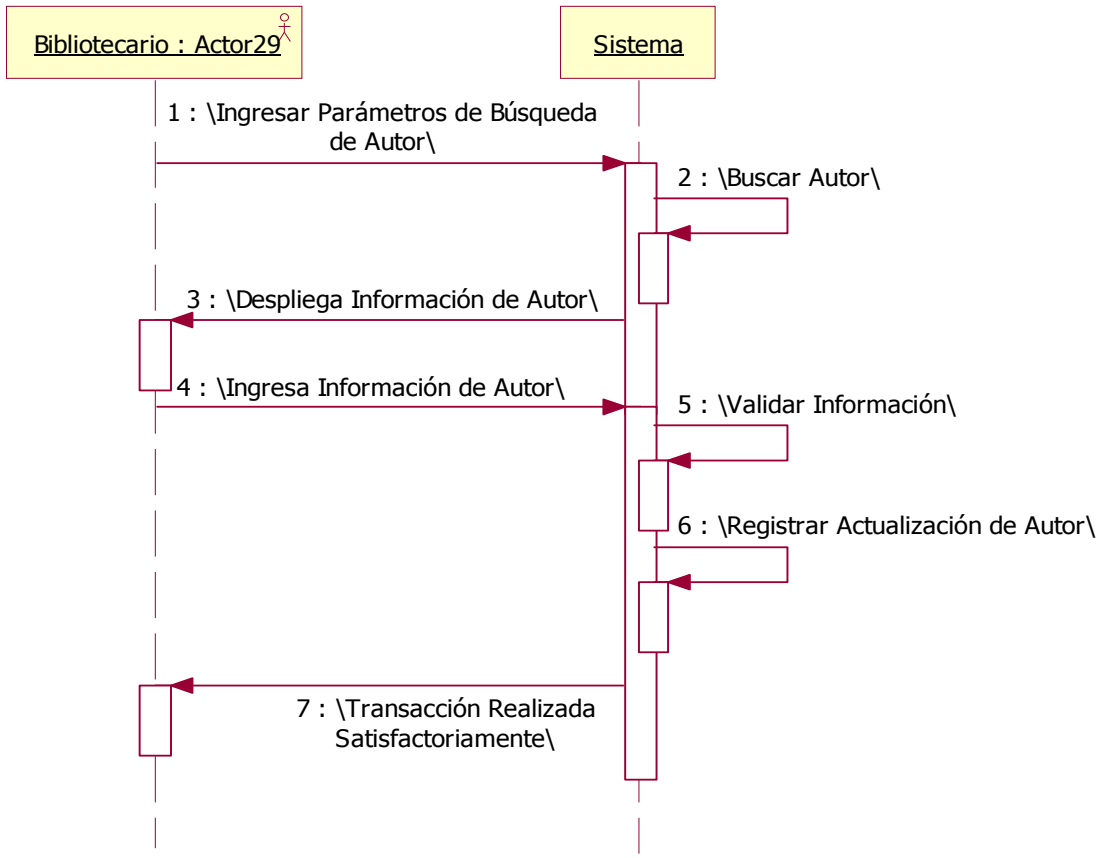
ACTUALIZAR IDIOMA



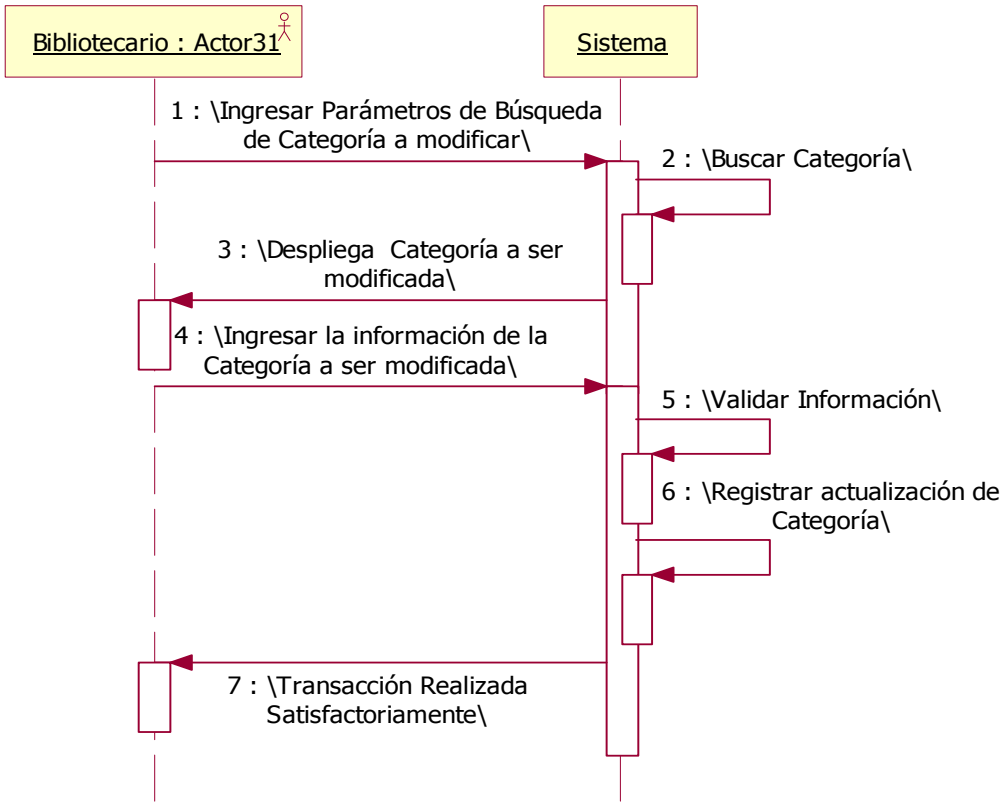
INGRESAR AUTOR



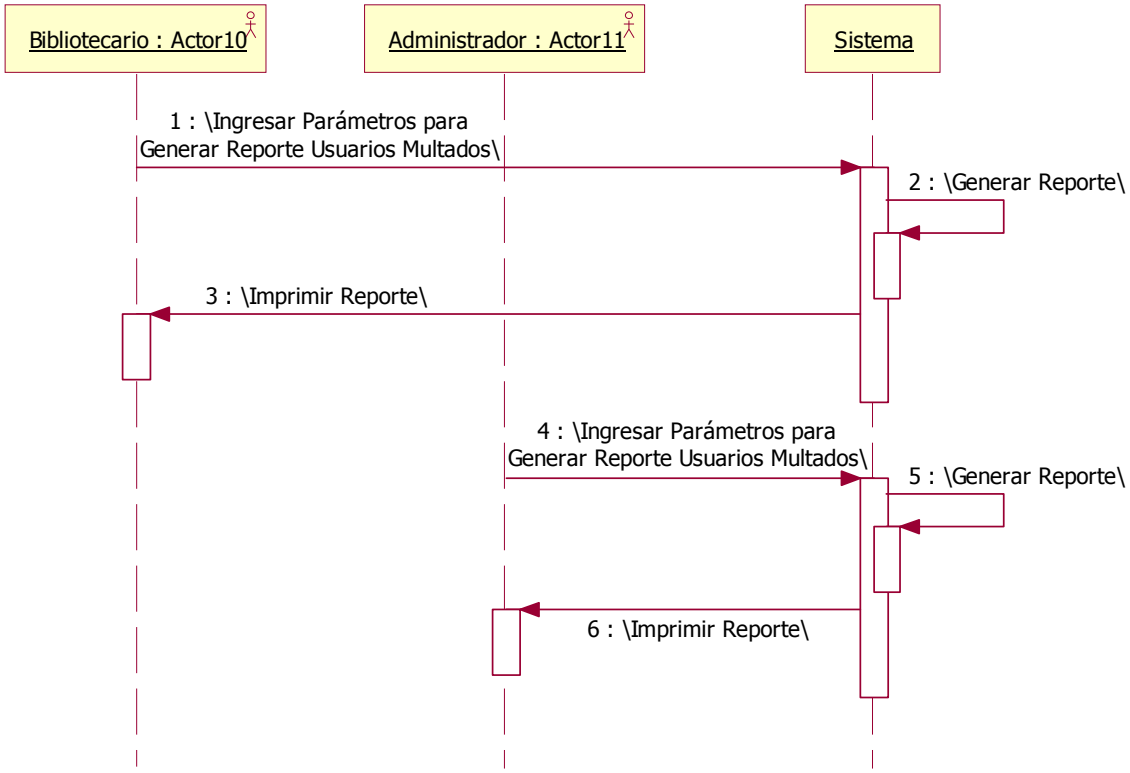
ACTUALIZAR AUTOR



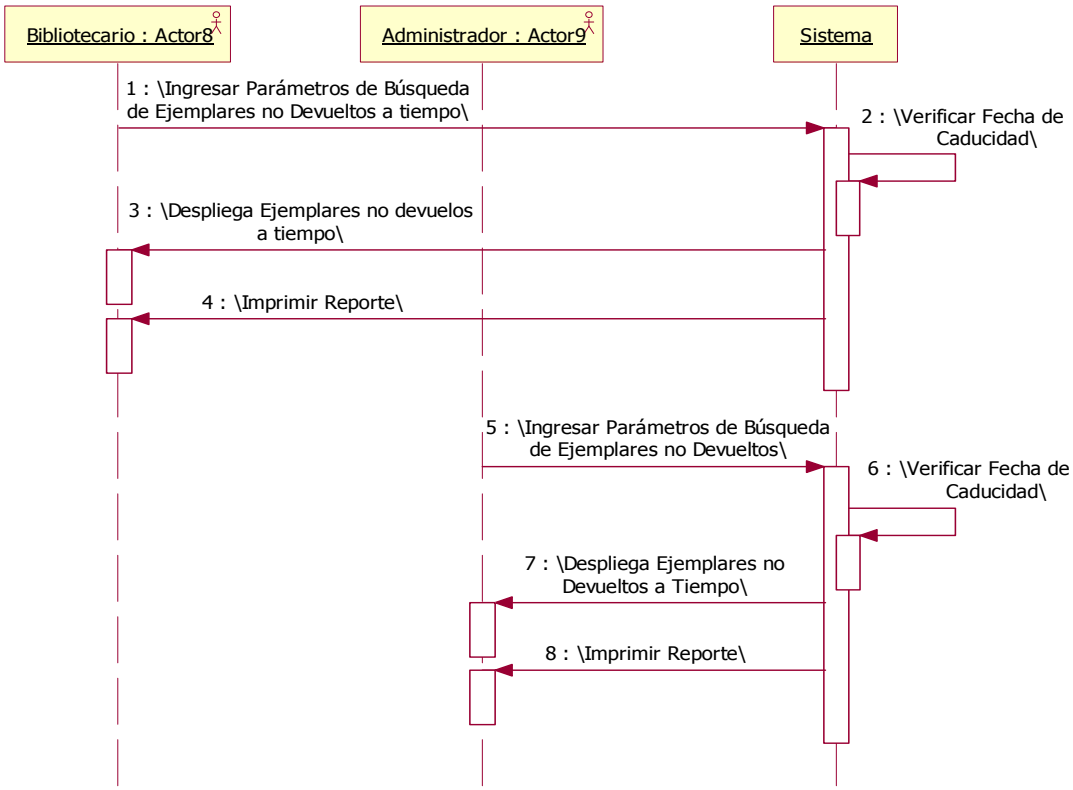
ACTUALIZAR CATEGORÍA



REPORTE USUARIOS MULTADOS



EJEMPLARES NO DEVUELTOS A TIEMPO



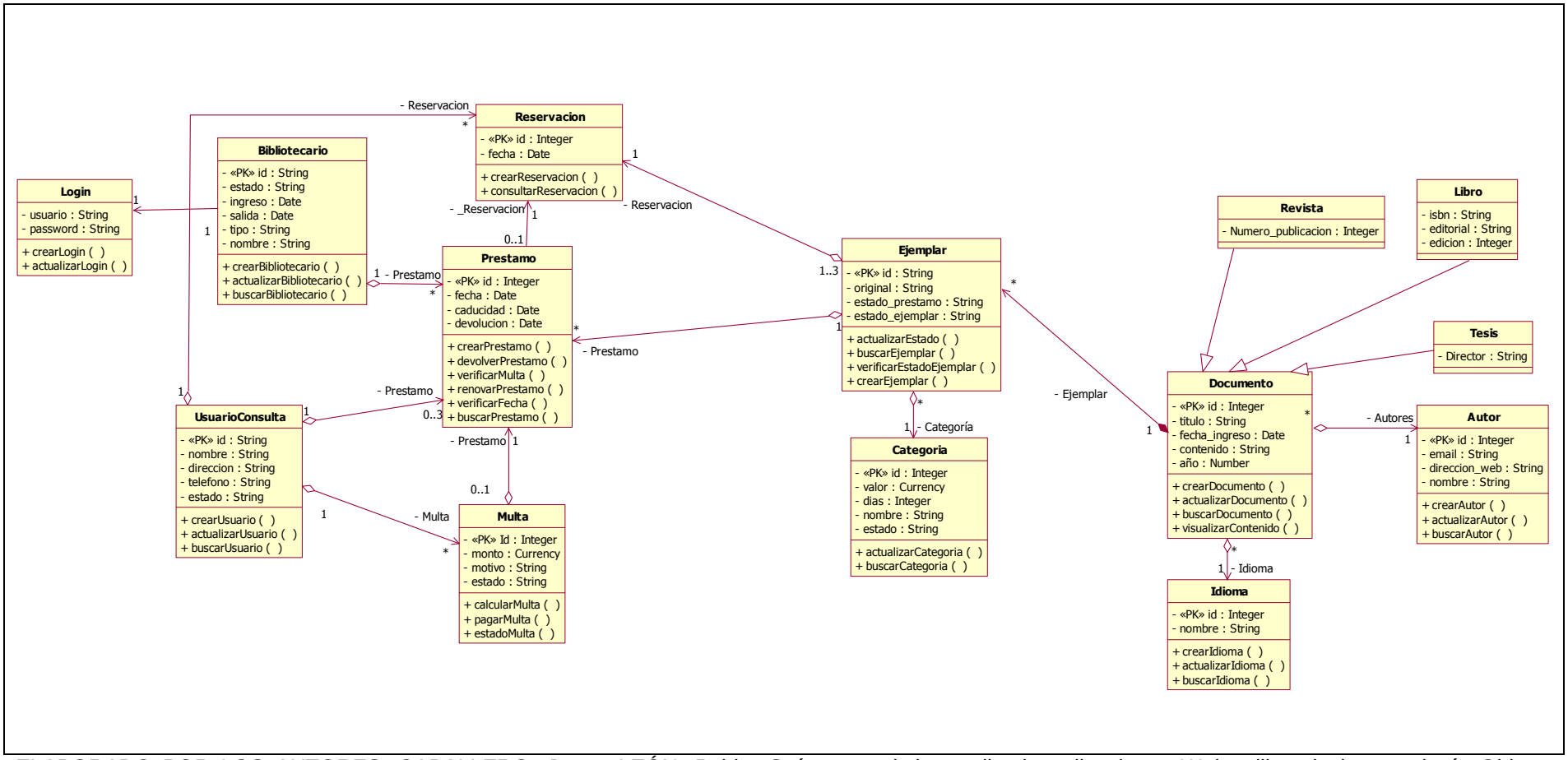
3.3.2 ETAPA 2: FASE DE DISEÑO

3.3.4.1 Diagrama de Clases de Diseño

El diagrama de clases de diseño nos permite identificar: las operaciones de las clases y las clases de Implementación.

En la Figura 3-14 se muestra el diagrama de clases de diseño de SIABI.

FIGURA 3-14: DIAGRAMA DE CLASES DE DISEÑO



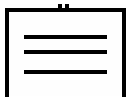
ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006.

3.3.4.2 Diseño de Navegación del Sistema

El diseño de navegación muestra los enlaces que proporciona la aplicación para cada uno de los usuarios

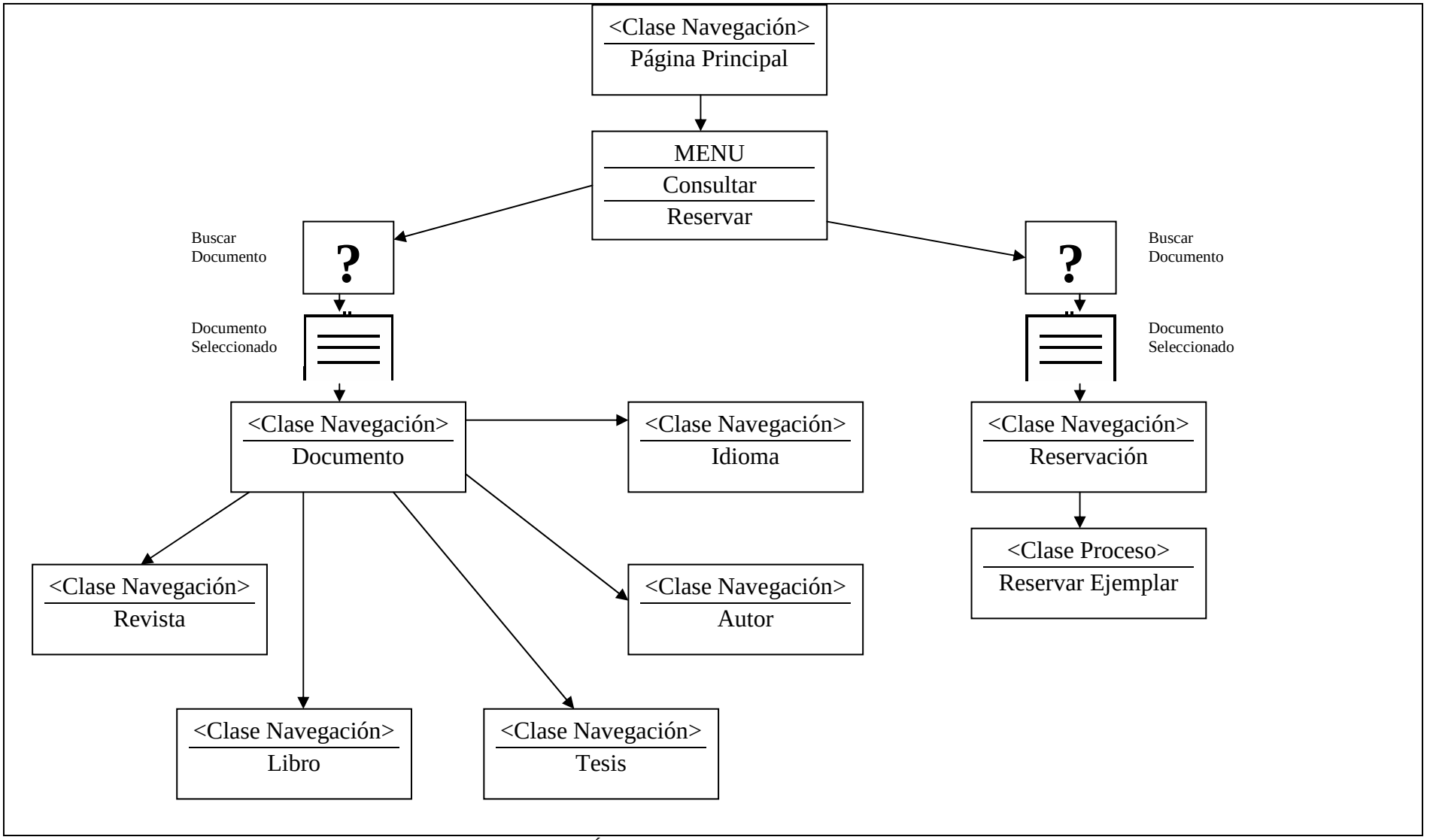
En la TABLA 3-4 mostramos una descripción de la notación que utilizamos en el diseño de navegación del sistema.

TABLA 3-4: NOTACIÓN DEL DISEÑO DE NAVEGACIÓN

NOTACIÓN	DESCRIPCIÓN
<Clase Navegación> Página Principal	Esta notación indica como se relacionan las clases cuando se accede a una determinada página del sistema
<Clase Proceso> Reservar Ejemplar	Esta notación indica que se va a realizar un proceso sobre las clases de navegación
	Esta notación es utilizada para indicar que se va a realizar una búsqueda sobre las clases de navegación
	Esta notación es utilizada para mostrar el resultado de una búsqueda o el acceso a una determinada página del sistema

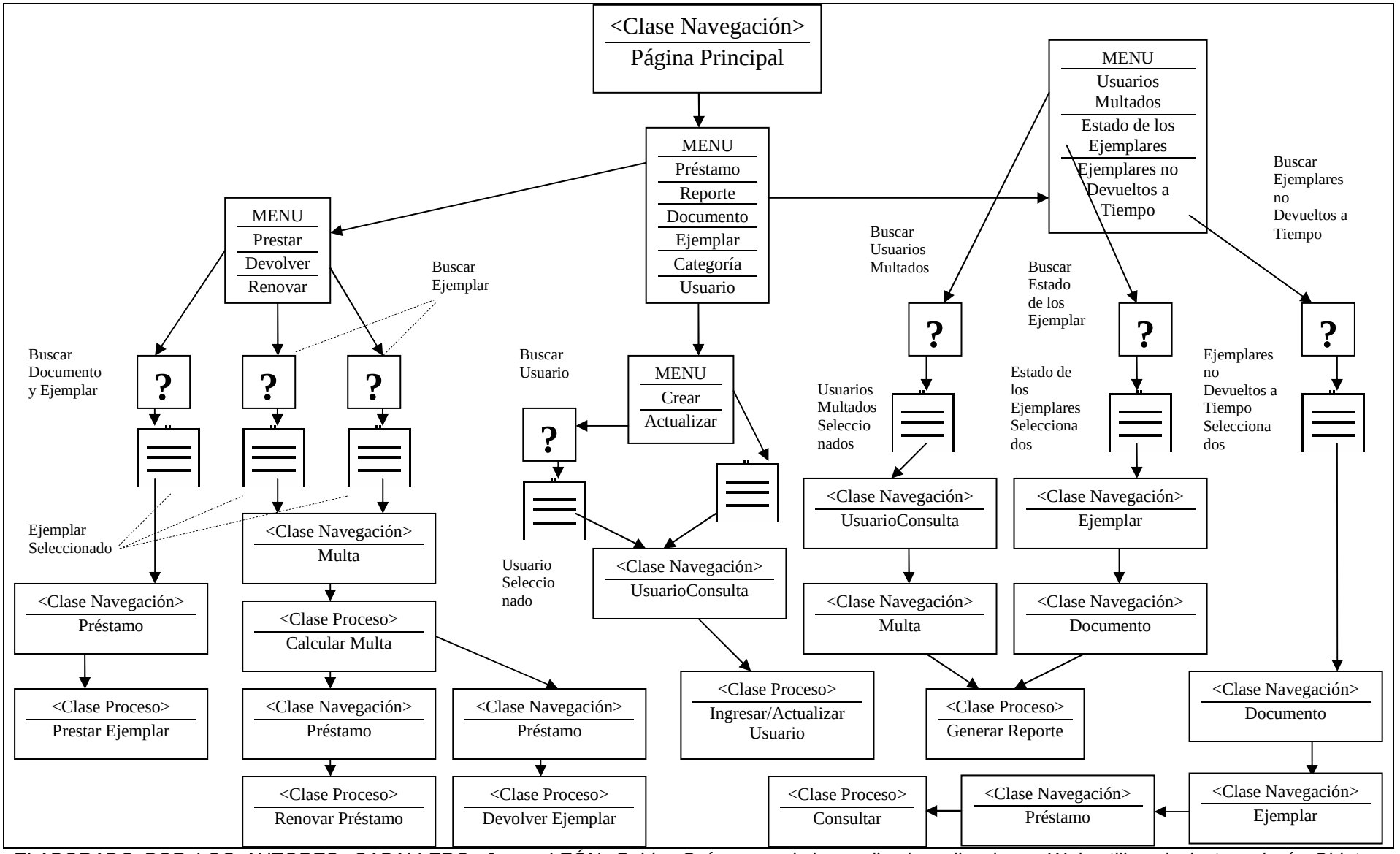
ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006

FIGURA 3-15: DISEÑO DE NAVEGACIÓN PARA EL USUARIO



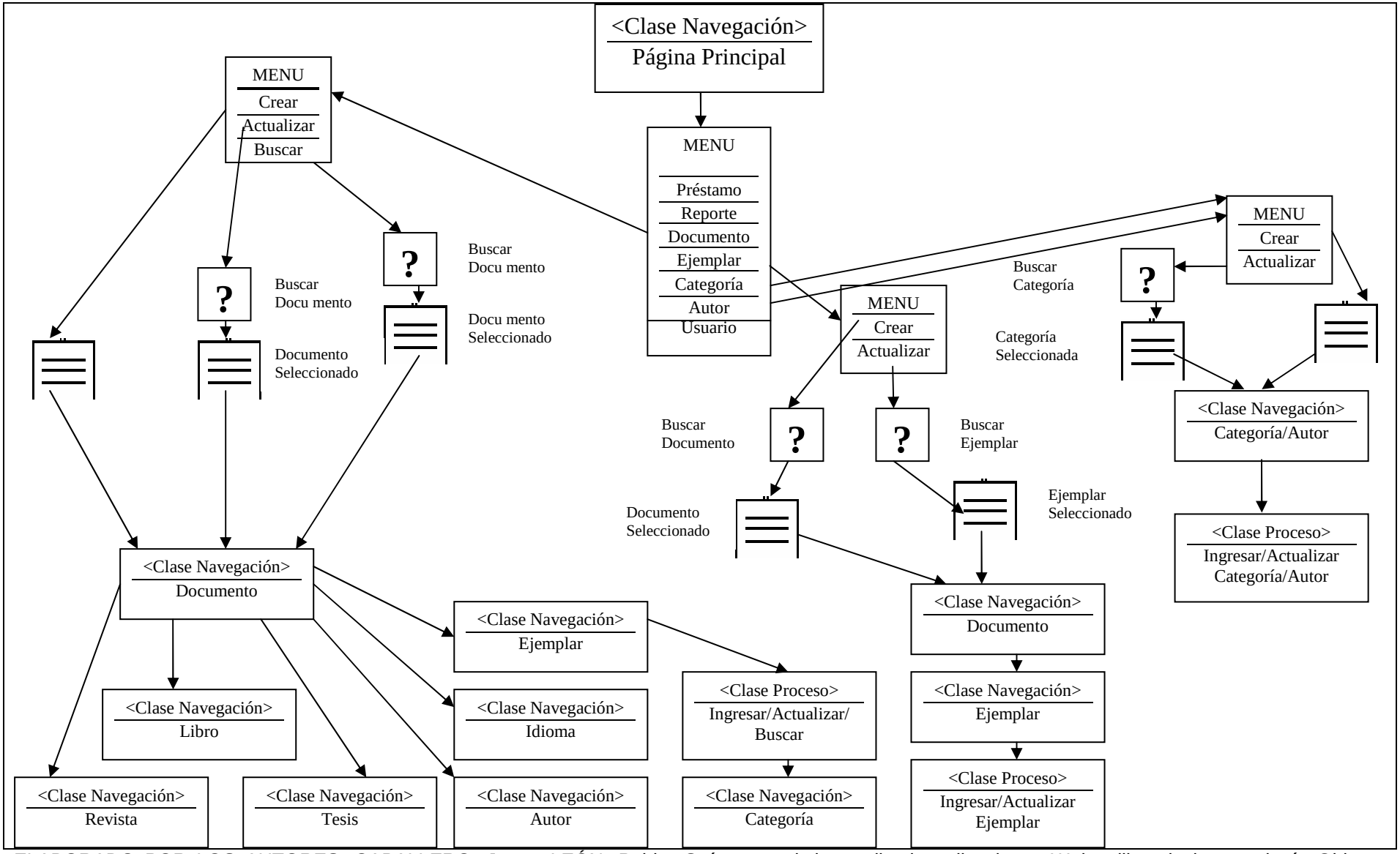
ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006

FIGURA 3-16: DISEÑO DE NAVEGACIÓN PARA EL BIBLIOTECARIO PARTE I



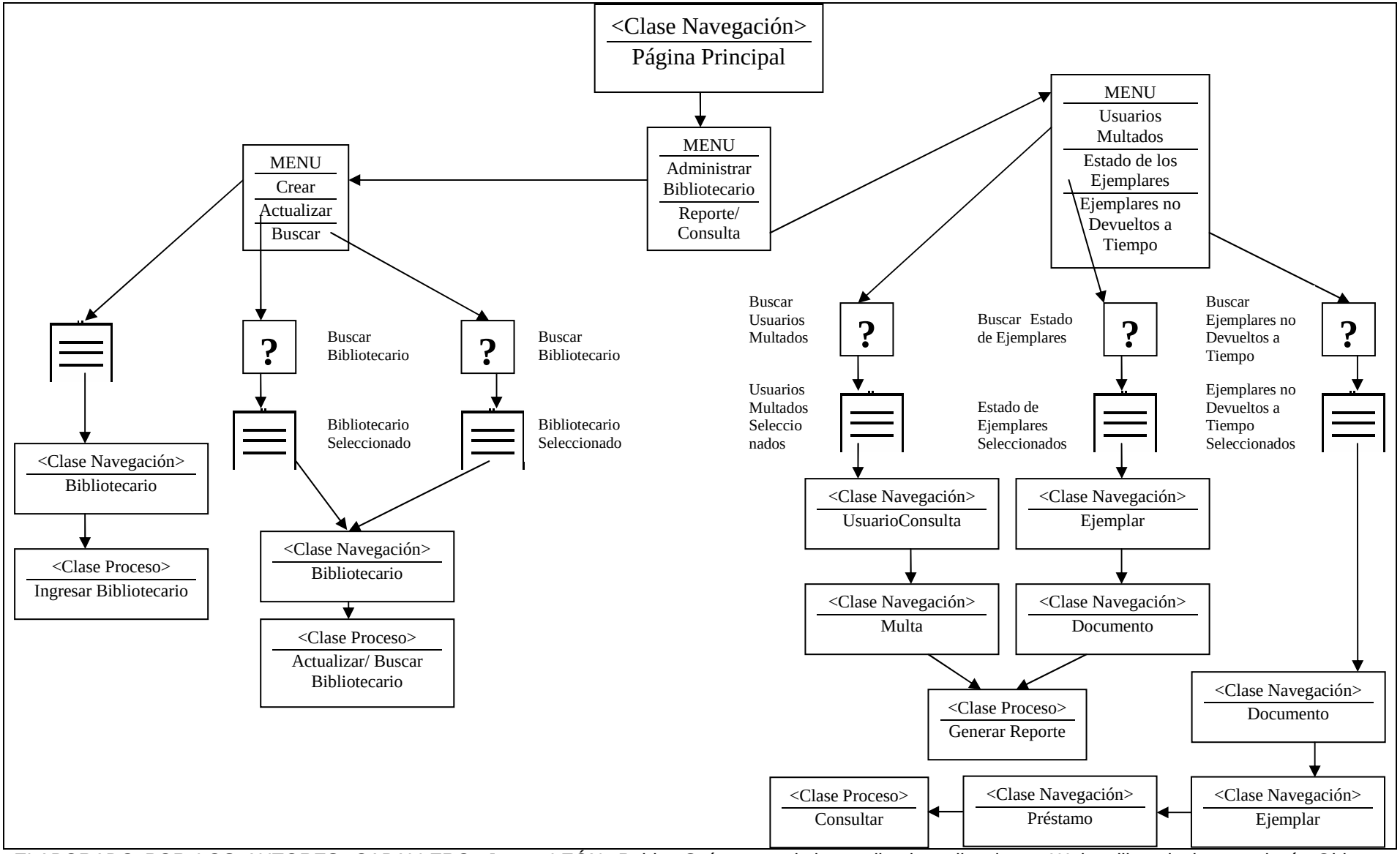
ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006

FIGURA 3-17: DISEÑO DE NAVEGACIÓN PARA EL BIBLIOTECARIO PARTE II



ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006

FIGURA 3-18: DISEÑO DE NAVEGACIÓN PARA EL ADMINISTRADOR



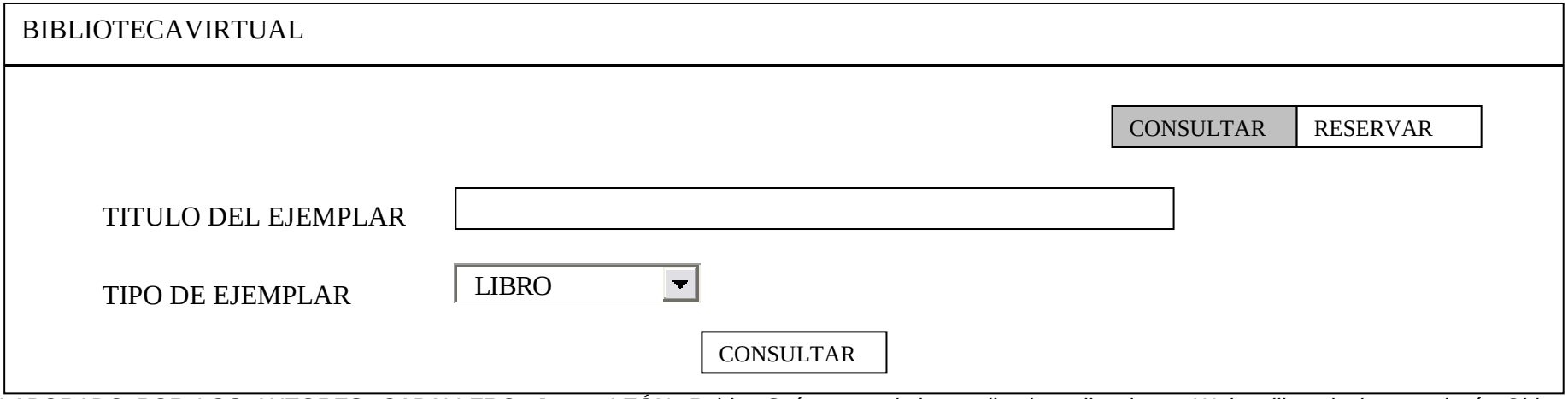
ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006

3.3.4.3 Diseño General de Interfaces de Usuario

Este diseño muestra un bosquejo de las principales interfaces del sistema.

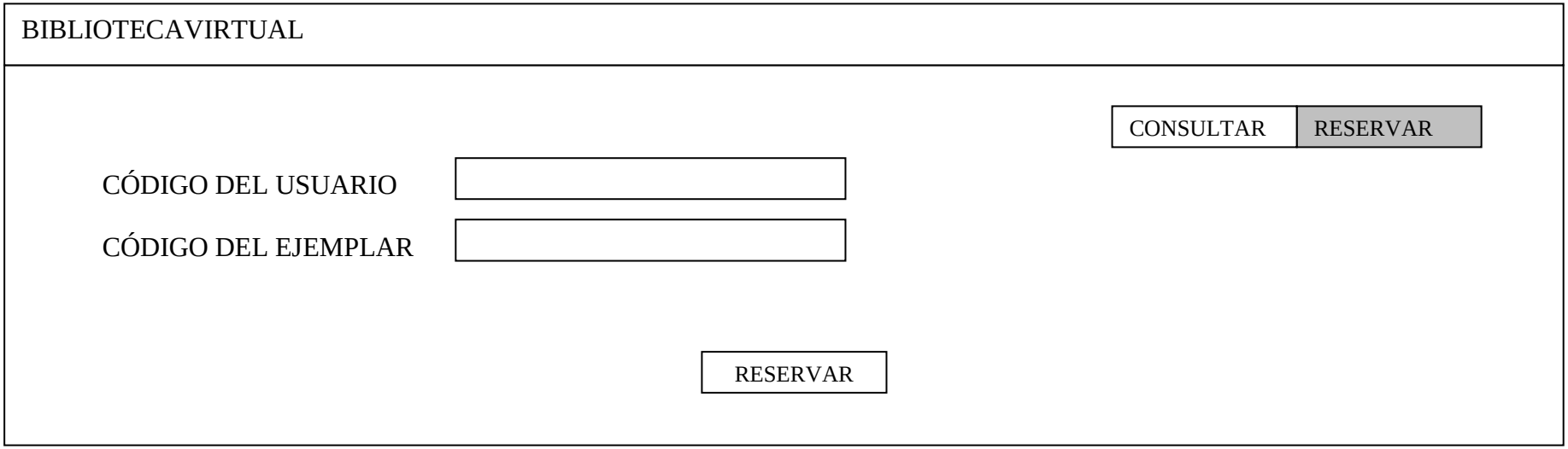
INTERFACES DE USUARIO

FIGURA 3-19: INTERFAZ DE CONSULTA DE SIABI



ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006

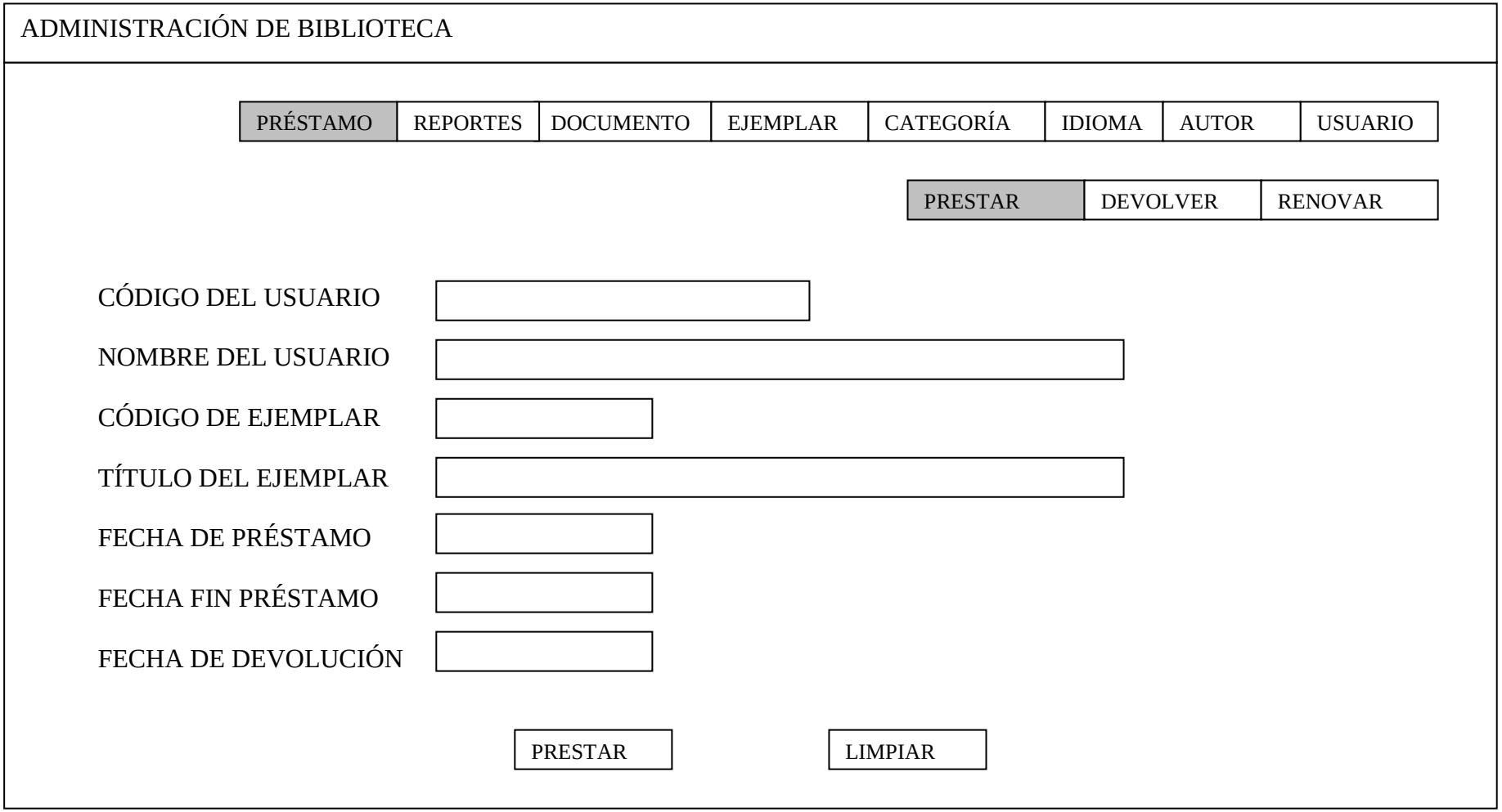
FIGURA 3-20: INTERFAZ DE RESERVACIÓN DE SIABI



ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006

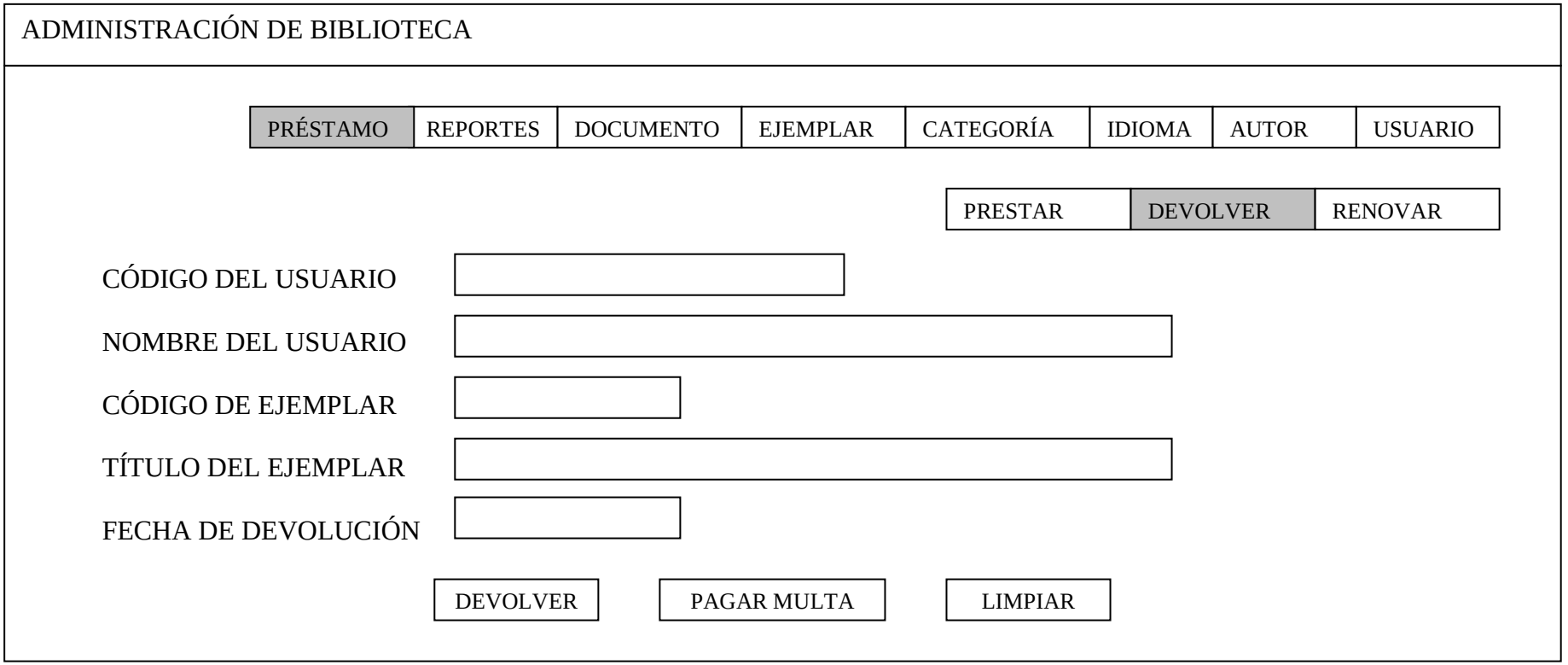
INTERFACES DEL ADMINISTRADOR Y BIBLIOTECARIO

FIGURA 3-21: INTERFAZ DE PRESTAMO DE SIABI



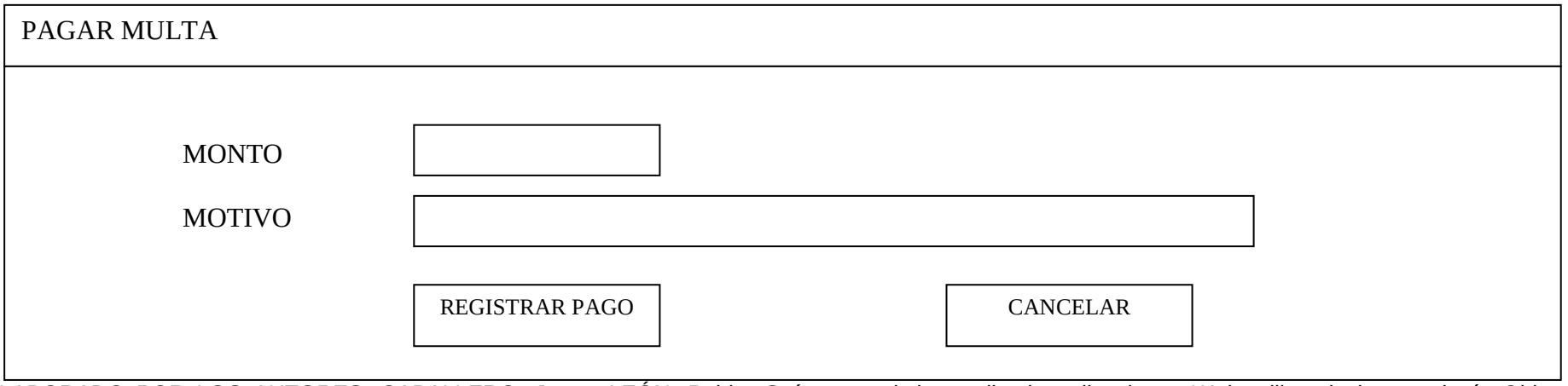
ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006

FIGURA 3-22: INTERFAZ DE PRESTAMO DE SIABI



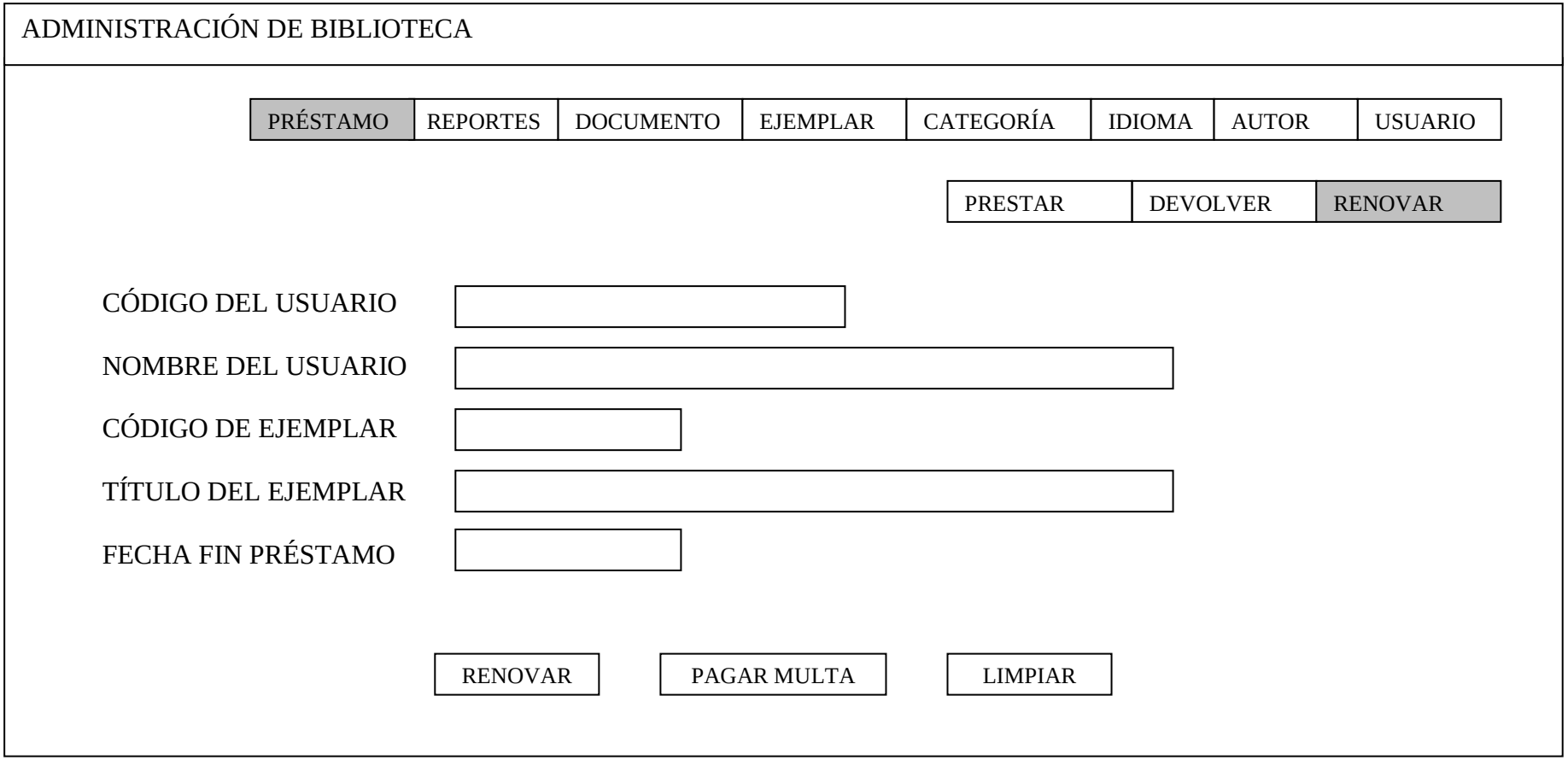
ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006

FIGURA 3-23: INTERFAZ DE PAGAR MULTA DE SIABI



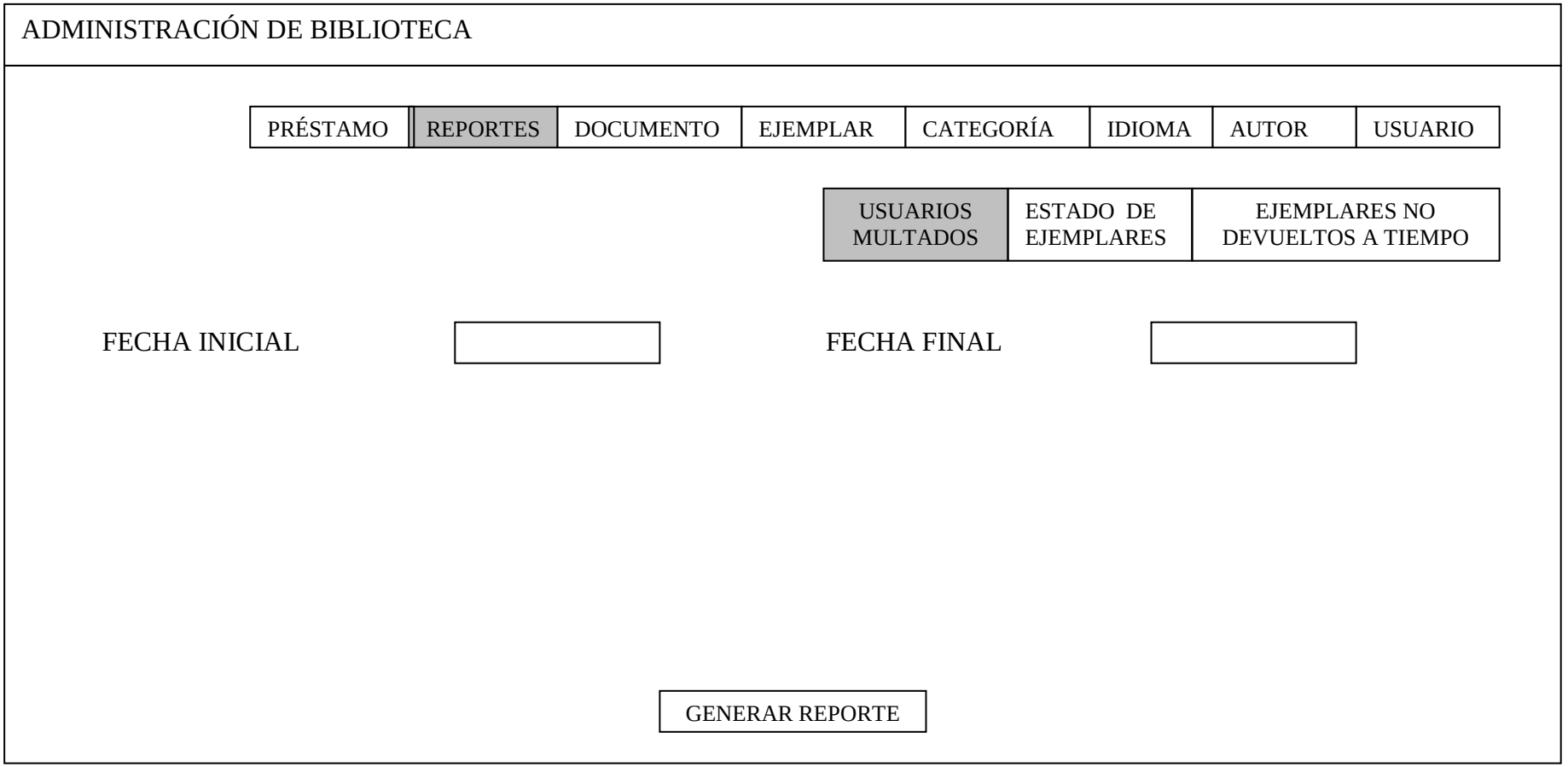
ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006

FIGURA 3-24: INTERFAZ DE PRESTAMO DE SIABI



ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006

FIGURA 3-25: INTERFAZ DE REPORTES DE SIABI

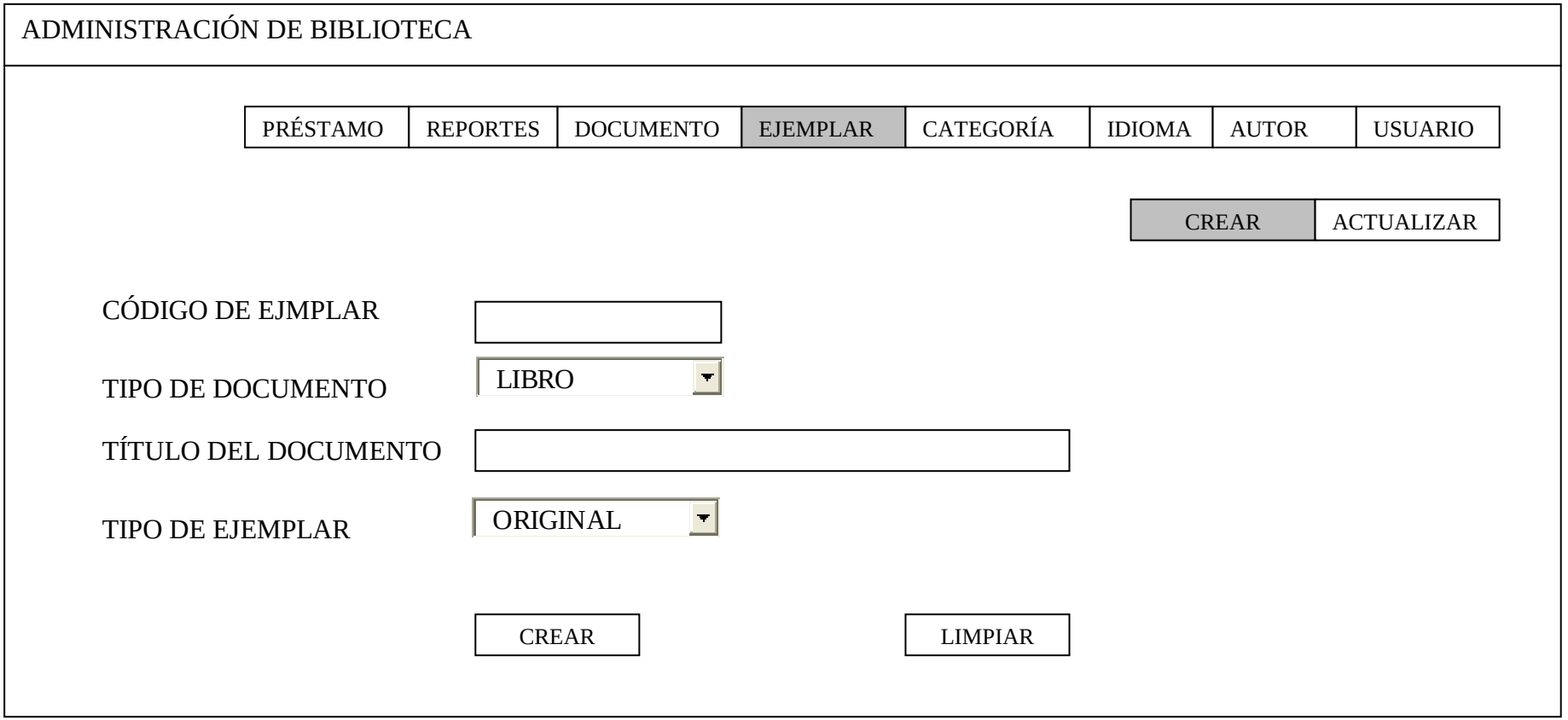


ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006

FIGURA 3-26: INTERFAZ DE DOCUMENTO DE SIABI

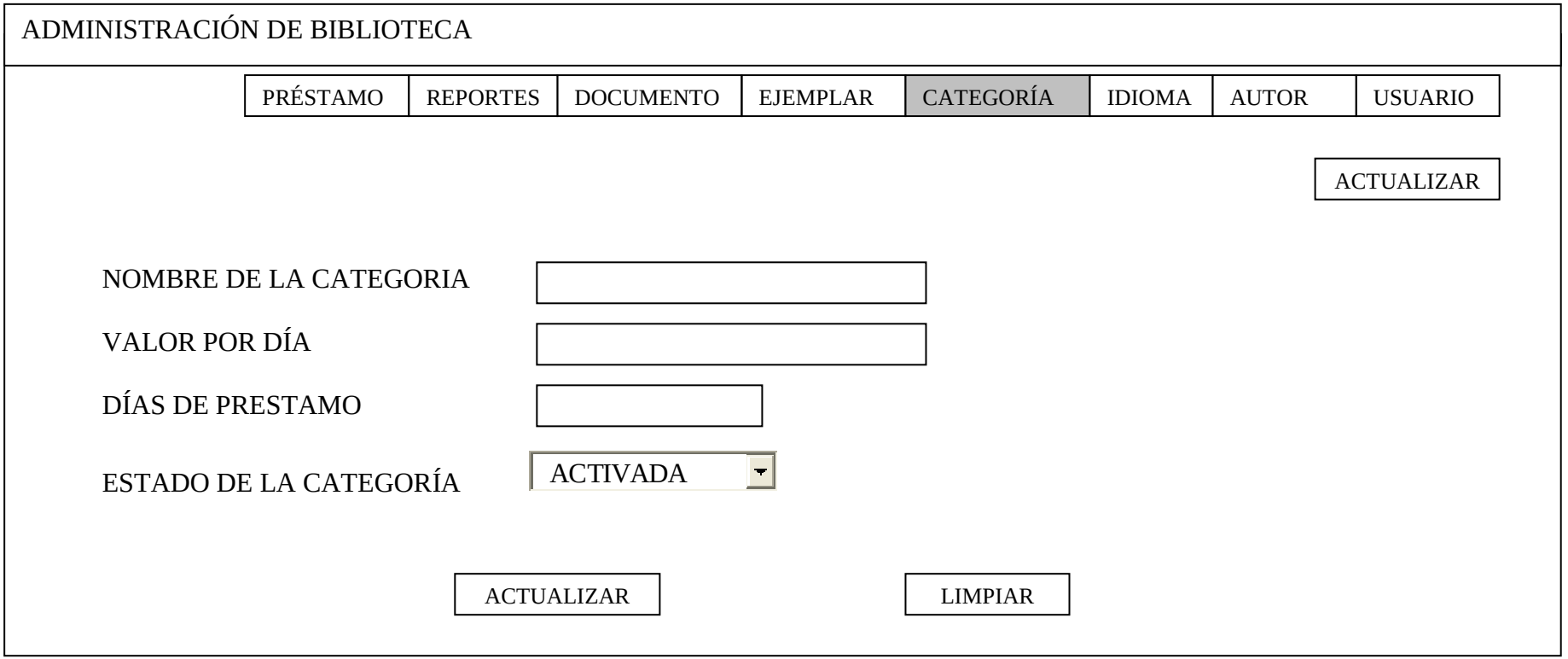
ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006

FIGURA 3-27: INTERFAZ DE EJEMPLAR DE SIABI



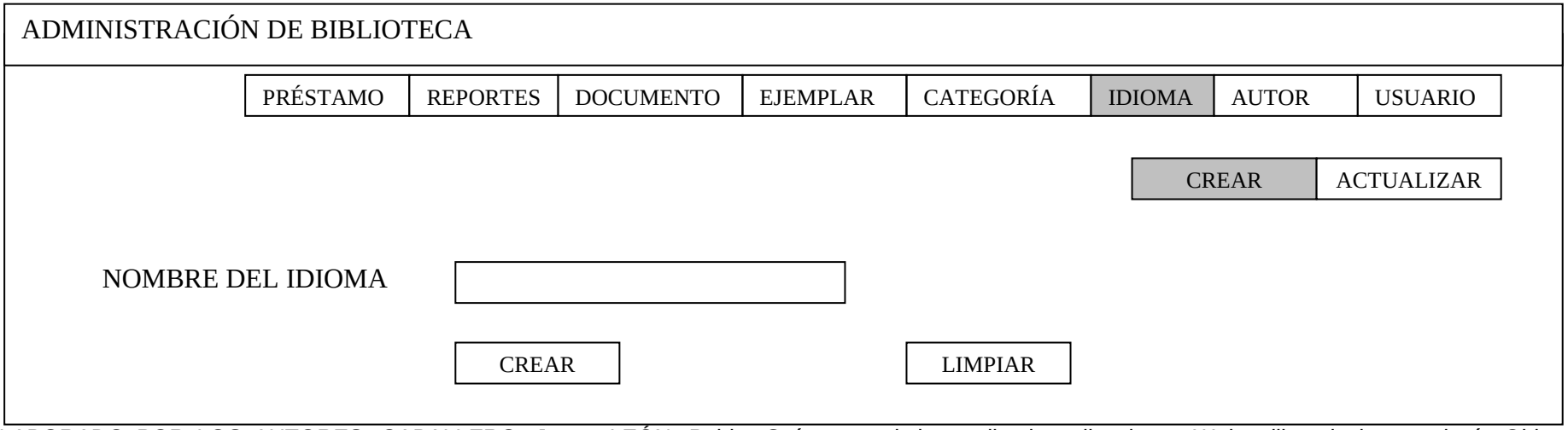
ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006

FIGURA 3-28: INTERFAZ DE CATEGORIA DE SIABI



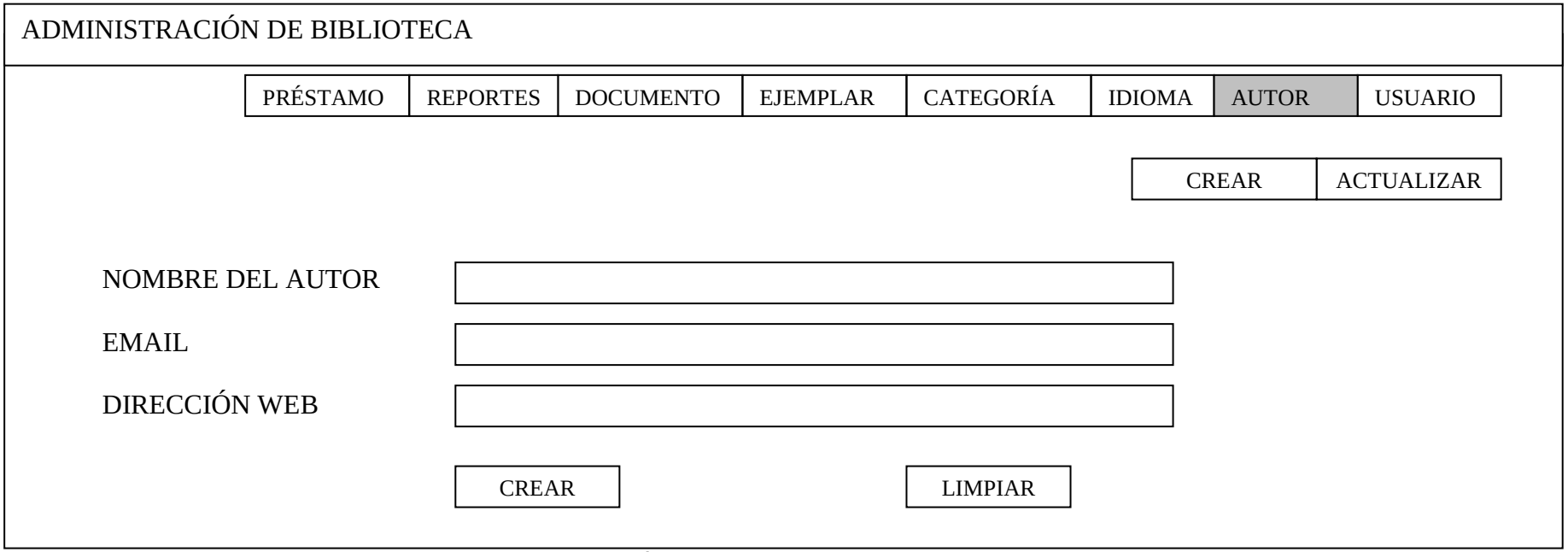
ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006

FIGURA 3-29: INTERFAZ DE IDIOMA DE SIABI



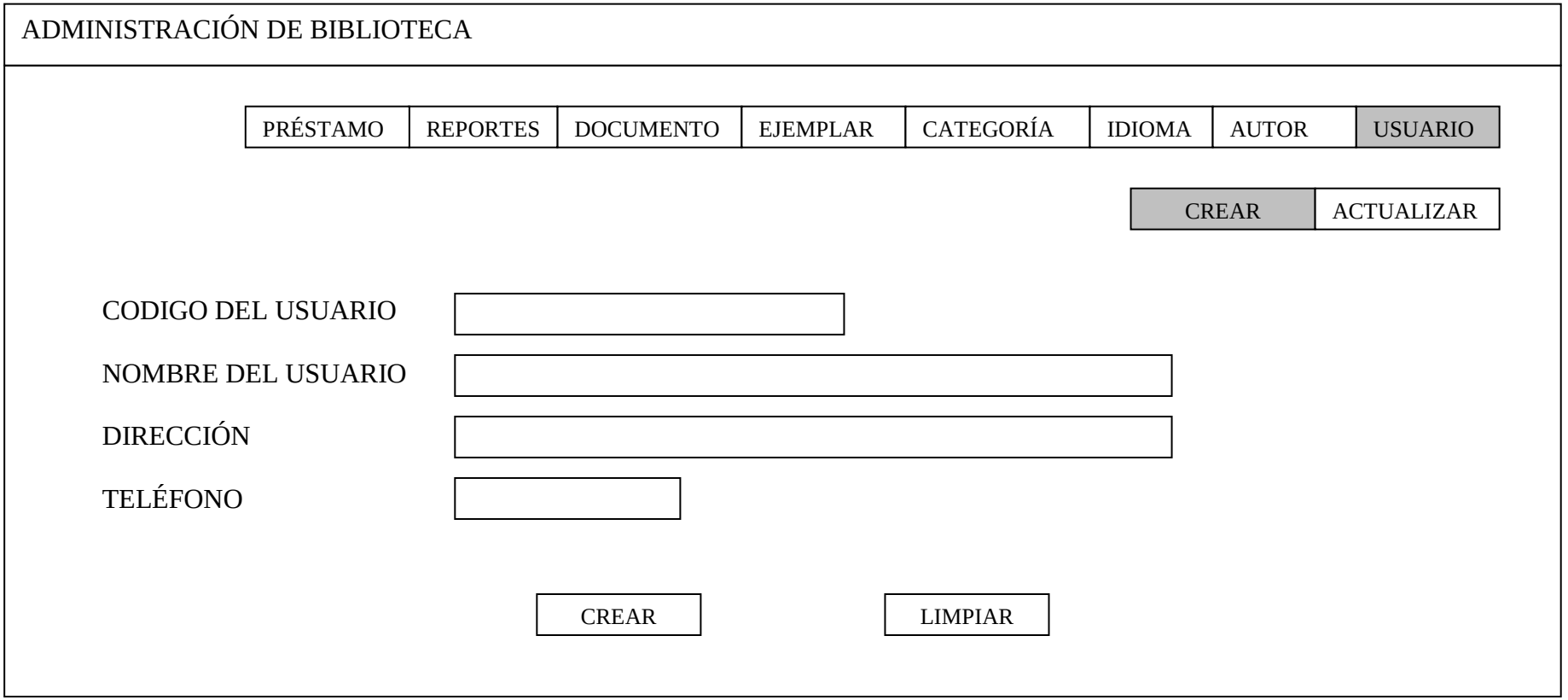
ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006

FIGURA 3-30: INTERFAZ DE AUTOR DE SIABI



ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006

FIGURA 3-31: INTERFAZ DE USUARIO DE SIABI



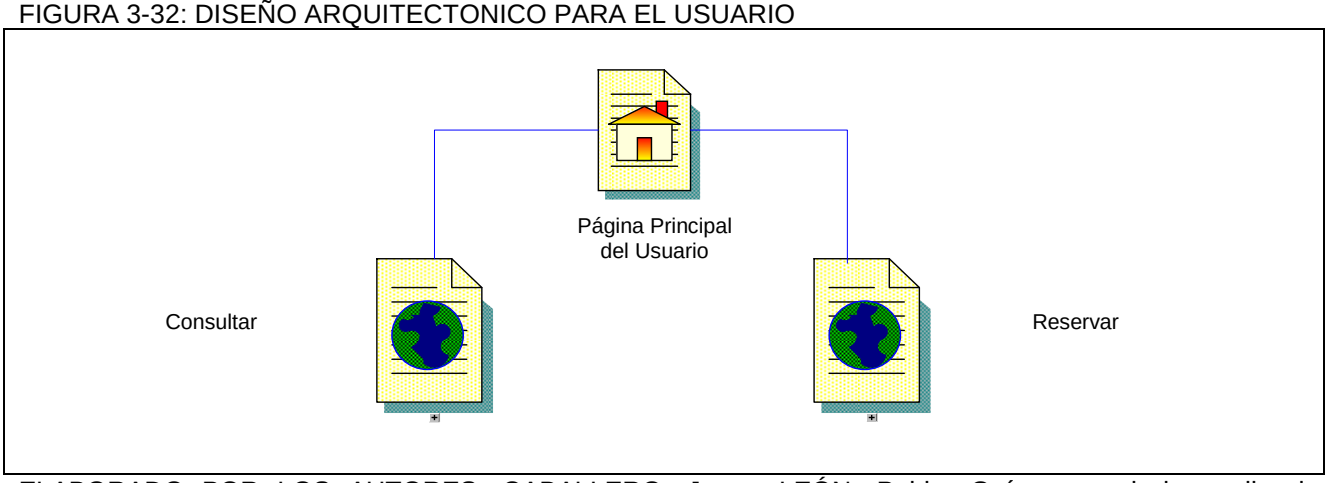
ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006

3.3.4.4 Diseño Arquitectónico de la Aplicación

El diseño arquitectónico muestra la estructura global de la aplicación y da una idea de cómo se dispone el contenido de la aplicación. La presente aplicación tiene una estructura jerárquica.

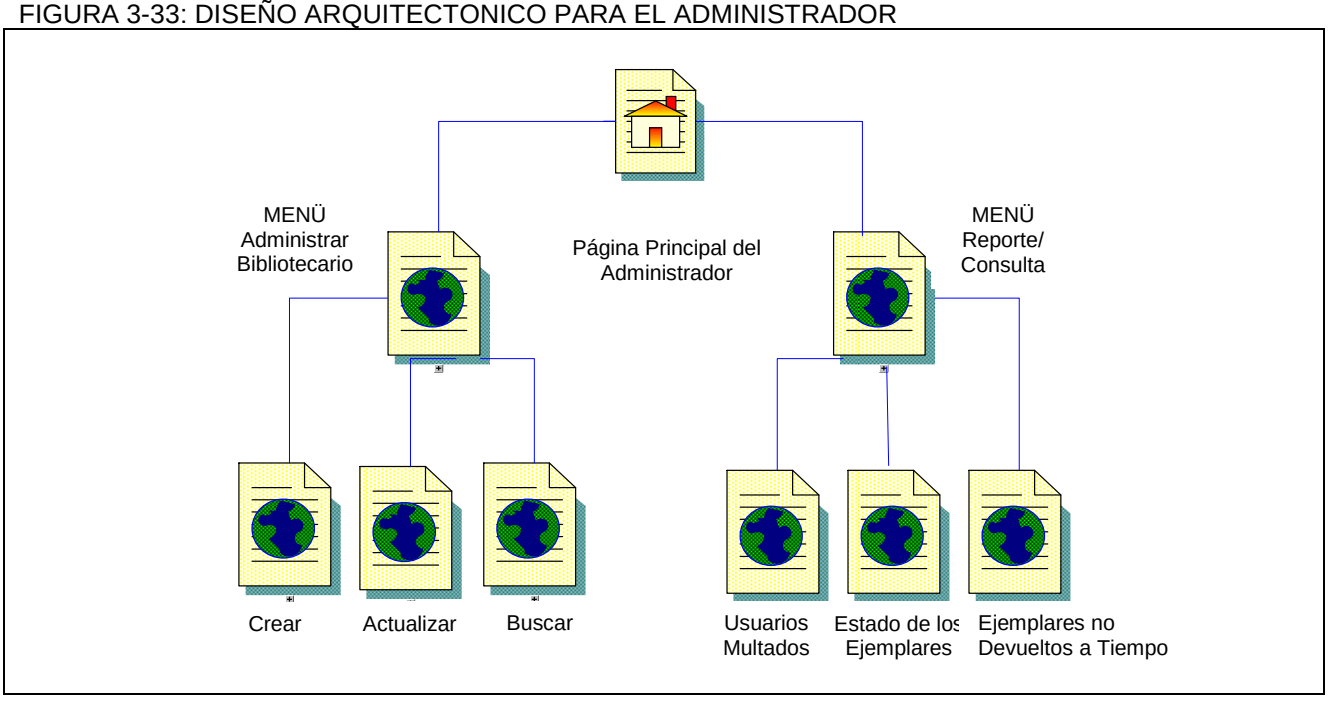
A continuación presentamos el diseño arquitectónico para los perfiles Administrador, Bibliotecario y Usuario de la aplicación.

FIGURA 3-32: DISEÑO ARQUITECTONICO PARA EL USUARIO



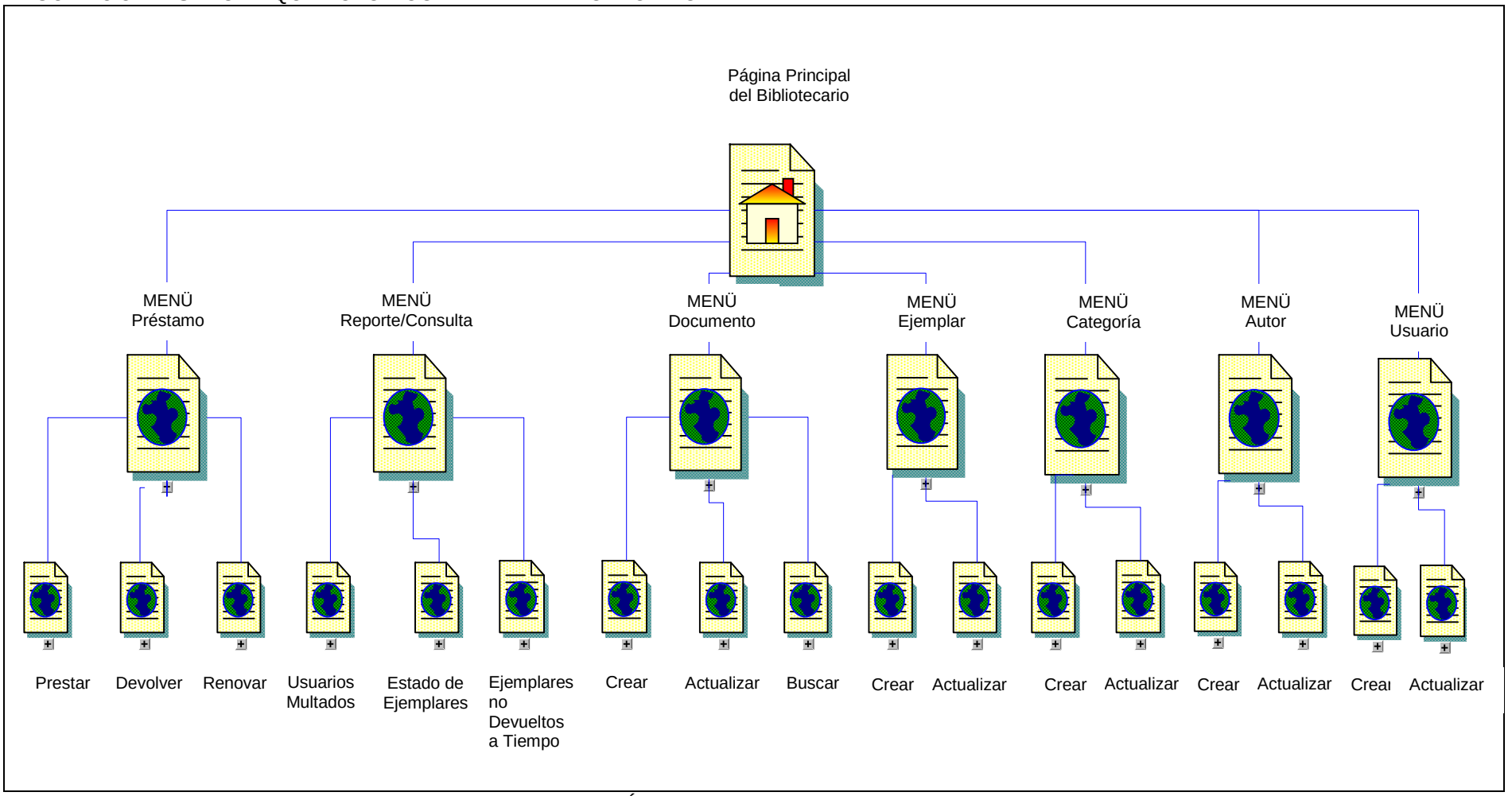
ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006

FIGURA 3-33: DISEÑO ARQUITECTONICO PARA EL ADMINISTRADOR



ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006

FIGURA 3-34: DISEÑO ARQUITECTONICO PARA EL BIBLIOTECARIO



ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006

3.3.3 ETAPA 3: FASE DE IMPLEMENTACIÓN Y PRUEBAS

3.3.5.1 Caracterización de las Herramientas de Desarrollo

Front-End.- Para el desarrollo de las interfaces de la aplicación (SIABI) se utilizará la herramienta Oracle Forms la cual permite desplegar aplicaciones en la Web. Esta herramienta posee: el Forms Developer (entorno de desarrollo) y Form Service (entorno de despliegue). Oracle Forms Services es el nombre de la colección de componentes usados para desplegar una aplicación Forms en Web. Adicionalmente para la generación de los reportes necesarios se utilizará Oracle Reports la cual permite el rápido y fácil desarrollo de reportes Web.

Oracle Forms provee una herramienta de desarrollo declarativa que utiliza PL/SQL como lenguaje de programación, las aplicaciones desarrolladas con Forms permiten construir agradables Java UI (User Interfase) sin escribir una sola línea de código Java. La interfaz de una aplicación construida en Forms es completamente un applet Java. Oracle Forms permite de manera rápida, fácil y eficiente la construcción de aplicaciones Web escalables que soportan una gran cantidad de usuarios.

Al elegir a Oracle como herramienta de desarrollo se tiene como objetivo brindar una alternativa a los desarrolladores acostumbrados a utilizar PL/SQL, para que también puedan desarrollar aplicaciones Web sin necesidad de aprender otro lenguaje orientado a la Web tal como java, php, etc.

Las aplicaciones realizadas con Oracle Developer pueden ser ejecutadas en una intranet, extranet e incluso el Internet.

Lenguaje de Programación.- El lenguaje seleccionado para desarrollar la aplicación (SIABI) es PL/SQL; el mismo que es un lenguaje procedural que amplía las capacidades del estándar SQL y permite definir secuencias de control de flujo y toma de decisiones.

El lenguaje PL/SQL está incorporado en:

- Servidor de la base de datos Oracle.
- Herramientas de Oracle tales como Forms, Reports

PL/SQL soporta todas las consultas y manipulación de datos que se usan en SQL, e incluye nuevas características:

- El manejo de variables.
- Estructuras modulares.
- Estructuras de control.
- Control de excepciones.

Los paquetes PL/SQL se pueden almacenar en la base de datos como otro objeto, y todos los usuarios que tengan los permisos necesario pueden acceder a estos paquetes. Los programas se ejecutan en el servidor permitiendo de esta manera el ahorro de recursos a los clientes.

La herramienta utilizada para realizar los scripts para el diseño de la base de datos objeto relacional será SQL plus, herramienta propia de la base de datos. Se sugiere utilizar PL/SQL Developer que permite agilitar el proceso de programación y corrección de errores.

Back-End.- La base de datos que se va a utilizar es Oracle 10g herramienta que se basa en la tecnología cliente/servidor la misma que posee un lenguaje de 5ta generación que es PL/SQL antes mencionado.
Oracle Database 10*g* brinda tiempo de respuesta aceptables, alta disponibilidad, escalabilidad, soporte de transacciones, estabilidad y es multiplataforma.

Sistemas Operativos para el desarrollo del sistema SIABI.- XP es el Sistema Operativo que se va a utilizar para el desarrollo de la aplicación, este sistema operativo usa el núcleo de Windows NT. Incorpora una nueva interfaz y tiene capacidades multimedia, dispone de soporte para redes inalámbricas y asistencia remota. Otra de las consideraciones que se debe notar es que XP es el sistema operativo más usado por los usuarios de Internet en la actualidad.

Sistemas Operativos para los clientes Web.- El sistema operativo para los clientes Web puede ser cualquier sistema operativo que tenga un Internet Explorer 6 o superior, Netscape 7.0 o superior.

Requerimientos de Hardware.- Para el desarrollo de aplicaciones Web con Oracle Database 10g y Oracle Developer Suite 10g se recomienda utilizar los siguientes requerimientos de Hardware que se detallan en la Tabla 3-5:

TABLA 3-5: REQUERIMIENTOS MÍNIMOS DE HARDWARE PARA CADA HERRAMIENTA [25],[26]

	Procesador	**Memoria**	**Disco Duro**
DATABASE	Intel Pentium 1 GHz	1024 MB	2.5 GB
	Procesador	**Memoria**	**Disco Duro**
DEVELOPER SUITE	Intel Pentium 500 MHz	128 MB	943 MB

3.3.5.2 Implementación del Modelo Objeto Relacional

En la Tabla 3-6 se muestra una parte del código fuente del sistema SIABI. La implementación del object type prestamo y sus funciones. Toda la implementación del sistema fue realizada en código pl/sql

TABLA 3-6: IMPLEMENTACIÓN DEL OBJECT TYPE PRESTAMO

[25] Guía de Conceptos de Oracle Universal Installer 10*g* Release 1 (10.1), Oracle Corporation,
[26] Installation and Configuration Documentation. Oracle Developer Suite 10*g*, Oracle Corporation

- CREACION DEL OBJECT TYPE PRESTAMO

```
create or replace type prestamo as object (
pre_id number,
pre_fecha date,
pre_devolucion date,
pre_caducidad date,
mul_prestamo ref multa,
res_prestamo ref reservacion,
static function crearPrestamo(vpre_fecha date,vres_prestamo ref reservacion , vfecha_fin date) return ref
prestamo,
static procedure  devolverPrestamo(vidprestamo number),
member function verificarMulta(vidprestamo number) return boolean,
static procedure renovarPrestamo(vidprestamo number),
member function verificarFecha(vidprestamo number) return number,
static function  buscarPrestamo(vidprestamo number) return prestamo);
/
```

- PERSISTENCIA DEL PRÉSTAMO

```
create table prestamo_t of prestamo(primary key(pre_id));
```

- PAQUETE DE PRÉSTAMO (utilizado para la inserción y actualización del objeto préstamo)

El paquete prestamo_p está implementado para que en este se realicen las inserciones y actualiciones en la tabla del objeto, el mismo es invocado por los métodos del objeto para con esto encapsular este tipo de operaciones.

```
create or replace package prestamo_p as
function crearPrestamo_p(vprestamo prestamo) return ref prestamo;
procedure devolver_p(vidprestamo number);
procedure renovar_p(vidprestamo) ;
end prestamo_p;
/
create or replace package body prestamo_p as
function crearPrestamo_p (vprestamo prestamo) return ref  prestamo is
vrefprestamo ref prestamo;
begin
insert into prestamo_t  (pre_id, pre_fecha,pre_caducidad,res_prestamo) values (vprestamo.pre_id,
vprestamo.pre_fecha,vprestamo.pre_caducidad,vprestamo.res_prestamo);
begin
select ref(vpre)
into vrefprestamo
from prestamo_t vpre
where vpre.pre_id=vprestamo.pre_id;
return vrefprestamo;
exception
when others then
raise_application_error(-20999,'Error en crear préstamo :'||vprestamo.pre_id||'-'||sqlerrm);
end;
end crearPrestamo_p;
procedure devolver_p(vidprestamo number) is
begin
update prestamo_t vpre
set vpre.pre_devolucion= sysdate
where vpre.pre_id=vidprestamo;
exception
when  others then
raise_application_error(-20999,'Error en devolución del libro '||sqlerrm);
end devolver_p;
procedure renovar_p(vidprestamo) is begin
```

```
update  prestamo_t vpre
set vpre.pre_fecha = trunc(sysdate),
vpre.pre_caducidad = trunc(sysdate) + vcategoria
where vpre.pre_id =vidprestamo;
end renovar_p;
end prestamo_p;
```

- CUERPO DEL OBJECT TYPE PRÉSTAMO

En el cuerpo del object type está la implementación de los métodos definidos en la creación del object type, como por ejemplo:

En el método crearPrestamo se hace la inserción en la tabla objeto y retorna la referencia el objeto creado para poder hacer referencia a este objeto posteriormente.

```
create or replace type body prestamo as
static function crearPrestamo(vpre_fecha date,vres_prestamo ref reservacion , vfecha_fin date) return ref
prestamo is
vprestamo prestamo:= new
prestamo(retorna_sec('secpre'),vpre_fecha,null,vfecha_fin,null,vres_prestamo);
vrefpre ref prestamo;
begin
vrefpre := prestamo_p.crearprestamo_p(vprestamo);
return vrefpre;
exception
when others then
raise_application_error(-20999,'Error Prestano.crearprestamo '||sqlerrm);
end;

static procedure devolverPrestamo(vidprestamo number)  is
begin
prestamo_p.devolver_p(vidprestamo);
end;

member function verificarMulta(vidprestamo number) return boolean is vmulta multa;
begin
if mul_prestamo is not null then
begin
select deref(vpre.mul_prestamo) into vmulta
from prestamo_t vpre
where vpre.pre_id =vidprestamo;
if vmulta.estadoMulta = 'P' then
return true;
else
return false;
end if;
exception
when others then
raise_application_error(-20999,'Error en verificar multa '||sqlerrm);
end;
elsif trunc(pre_caducidad) >= trunc(sysdate) then
return true;
else
   return false;
end if;
end;

static procedure renovarPrestamo(vidprestamo number) is vcategoria number;
begin
prestamo_p.renovar(vidprestamo number);
exception
```

```
when others then
raise_application_error(-20999,'Error en renovar prestamo '||sqlerrm);
end;

member function verificarfecha(vidprestamo number) return number is
vdias number;
begin
select to_date(vpre.pre_caducidad,   'yyyy/mm/dd') - to_date(vpre.pre_fecha, 'yyyy/mm/dd')
into vdias
from prestamo_t vpre
where vpre.pre_id=vidprestamo;
return vdias;
exception
when others then
raise_application_error(-20999,'Error en verificar fecha  '||sqlerrm);
end;

member function  buscarPrestamo(vidprestamo number) return prestamo is
vprestamo prestamo;
begin
select value(vpre)
into vprestamo
from prestamo_t  vpre
where vpre.pre_id=vidprestamo;
return vprestamo;
exception
when no_data_found then
return null;
when others then
raise_application_error(-20999,'Error en buscar prestamo '||sqlerrm);
end;
end;
```

- CREAR PRESTAMO (utilizado para instanciar y manipular el objeto préstamo)

En el procedimiento que se presenta a continuación se realiza todo el proceso de creación del préstamo y la actualización de las referencias en los objetos usuarioconsulta y bibliotecario, inicialmente verifica si el ejemplar se encuentra reservado o prestado para dependiendo de esto realizar el préstamo. El procedimiento recibe como parámetro el tipo de documento (vtipodoc) para de está manera identificar si se realiza el préstamo de una revista, libro o tesis.

```
procedure crearPrestamo_a(videjemplar varchar2,vidusuario  number,vtipodoc varchar2,vidbiblio
varchar2,vfechafin out date) is
vejemplar ejemplar;
vrefreserva ref reservacion;
vidusuario number;
vexcreserv exception;
vexcpres exception;
vexcperd exception;
vnumdias number;
vrefprestamo ref prestamo;
vdocid number:=Administra.retornadocid_a(videjemplar);
vusuario usuarioconsulta;
vrefusuario ref usuarioconsulta;
vbiblio bibliotecario;
vrefbiblio ref bibliotecario;
vidbibliot varchar2(10):=vidbiblio;
begin
--verifica si el ejemplar se encuentra reservado por el usuario que solicita el préstamo
vidusuario:= verificaReservacion(videjemplar,vrefreserva , vidusuario , vejemplar ,vtipodoc);

if vidusuario is null then
```

```
  raise vexcreserv;
else
--identica si el ejemplar se encuentra prestado o si está en buen estado para poder ser prestado.
 if vejemplar.eje_estado_prestamo='P' then
     raise vexcpres;
   elsif vejemplar.eje_estado_ejemplar = 'P' then
     raise vexcperd;
   end if;
end if;
begin
--identifica el número de días para los que puede ser prestado el ejemplar dependiendo de la categoría a la --
--que pertenece
  select cat.CAT_DIAS
  into vnumdias
  from categoria_t cat
  where substr(cat.CAT_NOMBRE,1,1)=vejemplar.EJE_ORIGINAL;
  vfechafin:=sysdate+vnumdias;
exception
when no_data_found then
raise_application_error(-20088,'Error Administra_Pre.crearPrestamo_a No parametrizada la categoria');
end;
--llamada al método crearprestamo del object type prestamo para crear el objeto crear prestamo.
vrefprestamo:=prestamo.crearprestamo(sysdate,  vrefreserva , sysdate+vnumdias);
begin
--se inserta la referencia del préstamo en el ejemplar
 vejemplar:=ejemplar_p.crearprestamoejemplar(vdocid,  vtipodoc, videjemplar, vrefprestamo);
  vejemplar.eje_estado_prestamo:='p';
  --actualizar documento
 if  vtipodoc='R' then
    revista.actualizarRevistaEjm(vdocid, videjemplar , vejemplar);
  elsif vtipodoc='T' then
    tesis.actualizarTesisEjm(vdocid, videjemplar , vejemplar);
  elsif vtipodoc='L' then
    libro.actualizarLibroEjm(vdocid, videjemplar , vejemplar);
  end if;
exception
when others then
raise_application_error(-20099,'Error Administra_Pre.crearPrestamo_a ejemplar '||sqlerrm);
end;
-- actualizar bibliotecario

vbiblio:=bibliotecario.buscarbibliotecario(vidbibliot,vrefbiblio);
vbiblio.actualizarbibliotecario(null,vidbiblio, vrefprestamo,null);

---actualizar usuario

vusuario:=usuarioconsulta.buscarusuario(vidusuarior,  vrefusuario);
vusuario.actualizarusuario(vidusuarior,null,null, null,   null, VREFPRESTAMO,  null, null);

---eliminar reservación
finReserva_a(videjemplar);

exception
when vexcreserv then
raise_application_error(-20014,'Ejemplar está reservado');
when vexcpres then
raise_application_error(-20015,'Ejemplar ya está prestado');
when vexcperd then
raise_application_error(-20015,'Ejemplar está perdido');
when others then
raise_application_error(-20099,'Error Administra_Pre.crearPrestamo_a '||sqlerrm);
end crearPrestamo_a;
```

- CREAR REVISTA (se utliza para la inserción en el objeto revista y en la tabla anidada del objeto)

Mediante la siguiente función se muestra como se realiza la inserción en una tabla anidada, siendo el caso de la inserción de un ejemplar en una revista.

```
function crearRevista_p(vdoc documento, vnumpublica number,vejemplar ejemplar) return ref revista is
vrevista ref revista;
vcoleje  list_ejemp_doc;
begin
--inserción en el object type revista.
  insert into revista_t (doc_id, doc_titulo,doc_fecha_ingreso, doc_contenido, doc_año,
idi_documento,aut_documento,eje_documento,numero_publicacion)
  values(vdoc.doc_id, vdoc.doc_titulo, vdoc.doc_fecha_ingreso,vdoc.doc_contenido, vdoc.doc_año,
vdoc.idi_documento,vdoc.aut_documento,null,vnumpublica);
  begin
--se selecciona la referencia del objeto creado y el collection type el cual es posteriormente actualizado en la
--tabla anidada
    select ref(vrev),vrev.eje_documento
    into vrevista, vcoleje
    from revista_t vrev
    where vrev.doc_id = vdoc.doc_id;
  exception when others then
    raise_application_error(-20999,'Error en crear revista '||vdoc.doc_id||'- '||sqlerrm);
  end;
  begin
      if vcoleje is null then
        vcoleje:=list_ejemp_doc();
      end if;
--se reserva memoria en el collection type
      vcoleje.extend;
--se inserta el ejemplar  en el collection type
      vcoleje(vcoleje.last):=vejemplar;
---actualización del object type revista para insertar el ejemplar en la tabla anidada.
      update revista_t r
      set   r.eje_documento = vcoleje
      where r.doc_id =vdoc.doc_id ;
  end;
  return vrevista;
end crearRevista_p;
```

- CONSULTAS

La siguiente consulta presenta el listado de los documento que no han sido devueltos, en esta consulta se puede identificar el uso de *deref* sentencia que permite seleccionar los atributos del object type préstamo desde el ejemplar el cual posee la referencia del préstamo. Adicionalmente se muestra la manera de accesar a las tablas anidadas mediante la sentencia *table.*

```
select r.doc_titulo, decode(e.eje_original,'o','original','c','copia') , deref(p.column_value).pre_caducidad
from revista_t r, table(r.eje_documento) e, table(e.pre_ejemplar) p
where deref(p.column_value).pre_devolucion is null
and deref(p.column_value).pre_caducidad between :p_fechaini and :p_fechafin
```

ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006.

En el Anexo F se muestra la implementación del modelo Objeto-Relacional del sistema SIABI (Sistema para la Administración de Bibliotecas) en Oracle 10*g*.

3.3.5.3 Diagrama de Componentes

El Diagrama de Componentes muestra los modulos físicos de código (componentes) y las relaciones lógicas entre ellos en un sistema. El diagrama de componentes de la aplicación SIABI se muestra en la sección 3.2 (Arquitectura de la Aplicación).

3.3.5.4 Diagrama de Despliegue

El diagrama de despliegue identifica la topología de procesadores y dispositivos sobre los que se ejecuta el sistema SIABI. El diagrama de despliegue de la aplicación SIABI se muestra en la sección 3.2 (Arquitectura de la Aplicación).

3.3.5.5 Estándares de Programación

Para desarrollar la presente aplicación se emplearon los siguientes estándares de programación:

Estándares de Codificación

- Estándares para nombrar a las variables

  Los nombres empleados en las variables identificaran exactamente la información que se necesita tener en la aplicación.

  Los nombres de las variables serán escritas con letras minúsculas. Los nombres de las variables incluirá al inicio la letra (v). Por Ejemplo: vnombre.

BACK-END

- Estándares para nombrar a la base de datos

  El nombre de la instancia de base de datos consta de un número máximo de 8 caracteres escritos en letras mayúsculas. Para la presente aplicación la instancia de la base de datos se llamará BIBLIO.

- Estándares para nombrar a los objetos (type) en la base de datos

  Para nombrar a los objetos que formarán parte del modelo de la instancia de base de datos, utilizaremos sustantivos o palabras que las describan completamente, escritas con la primera letra en mayúsculas.

- Estándares para nombrar a los atributos en la base de datos
  Los atributos serán nombrados de la siguiente manera:

  - El nombre del atributo identificará exactamente que tipo de dato o dato abstracto se necesita representar en la instancia de base de datos.
  - El nombre del atributo se escribirá en letras minúsculas.
  - El nombre del atributo incluirá al inicio las tres primeras letras del nombre del objeto a la que pertenece el atributo. En caso de que existiera alguna coincidencia con las iniciales de otro objeto se añadirá o quitará una letra a fin de que diferencie un objeto de otra.
  - Las iniciales del objeto se separan del nombre del atributo con el símbolo (_). Por Ejemplo: pre_codigo.

  Las colecciones serán nombradas de la siguiente manera:

  - El nombre de la colección identificará exactamente a que objeto pertenece.
  - El nombre de la colección se escribirá en letras minúsculas.

- El nombre de la colección incluirá al inicio list_, seguido del nombre del objeto al que pertenece la lista. Por ejemplo: list_ejemplar.

- Estándares para nombrar a las persistencias en la base de datos.

  Los nombres empleados en las persistencias tendrán el mismo nombre del objeto incluyendo al final el símbolo (_) seguido de la letra t, escrita en minúsculas. Por Ejemplo: Bibliotecario_t.

  En el caso de las tablas anidadas se la nombrará de la siguiente manera:

  - El nombre de la tabla anidada se escribirá en minúsculas.
  - El nombre de la tabla anidada comenzará con el nombre del objeto donde se aloja la tabla anidada seguida del nombre del objeto de la que es definida y termina con la letra t separados por el símbolo (_). Por Ejemplo: bibliotecario_prestamo_t

- Estándares para nombrar a los métodos y funciones en la base de datos

  Los métodos y funciones serán nombrados de la siguiente manera:

  - La letra inicial de la primera palabra con minúscula, seguida de letras minúsculas.
  - La letra inicial de la segunda palabra con mayúscula, seguida de letras minúsculas.
  - Sin separaciones entre las palabras. Por Ejemplo: crearUsuario.

- Estándar para nombrar a los paquetes en la base de datos
  Para nombrar a los paquetes utilizamos el/los nombres del type o los type del que procede. Por Ejemplo: prestamo_p.

FRONT-END

- Estándares para nombrar a las páginas

  Los nombres de las páginas Web serán escritos con letras mayúsculas.

- Estándares para nombrar a los objetos de los formularios

  Los objetos que creen dentro de un formulario deberán seguir el estándar definido en la Tabla 3-7.

TABLA 3-7: ESTANDAR DE CODIFICACIÓN DE OBJETOS DE LA APLICACIÓN

NOMBRE	DENOMINACIÓN
Cajas de Texto	txt*nombre_del_campo*
Combos	cmb*nombre_del_campo*
Botones de Comando	btn*nombre_del_comando*

ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006

3.3.5.6 Pruebas de la Aplicación

Para realizar las pruebas del sistema SIABI vamos a utilizar las plantillas del Anexo D. La ejecución de las pruebas del sistema está en el Anexo G.

CAPITULO 4
CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

La guía incorpora criterios para el desarrollo de aplicaciones Web que utilizan el modelo objeto relacional, basados en los lineamientos de las metodologías utilizadas, los mismos que posibilitaron gestionar la naturaleza de la aplicación.

La tecnología Objeto Relacional que utilizan los ORDBMS nos permiten modelar situaciones de la vida real de una mejor manera debido a su mayor flexibilidad en el almacenamiento de datos complejos, manipulación y validación.

Los ORDBMS son compatibles con los RDBMS, debido a su enfoque híbrido, es decir, que los ORDBMS son una extensión de los RDBMS, por lo cual pueden manipular tipos de datos complejos como: objetos, object table, object view, métodos, funciones, colecciones, nested table.

En las Bases de Datos Objeto Relacional, los objetos que tengan como atributo un objeto de tipo colección están exentos de la primera forma normal 1N, datos atómicos, que es una característica del modelo relacional, lo que le permite definir tablas con atributos cuyos valores pudieran ser tablas complejas.

Los conceptos de la Ingeniería Web proporcionan a los desarrolladores lineamientos para gestionar las aplicaciones Web desde la formulación de la aplicación hasta el desarrollo de la misma, tomando en cuenta aspectos importantes tales como fiabilidad, usabilidad, disponibilidad y escalabilidad de la aplicación.

Al desarrollar aplicaciones que utilicen el modelo Objeto Relacional nos podemos encontrar con el problema de dangling reference (referencias colgantes) que

ocurre cuando se elimina un objeto que esta siendo referenciado por otros objetos, por tanto es necesario el uso de constraints de integridad o triggers de control

Como resultado de este proyecto de titulación obtenemos una guía técnica que plantea objetivos, alcance, proporciona actividades y estrategias y flexibilidad para el desarrollo de aplicaciones Web utilizando modelos Objeto Relacionales.

La guía técnica desarrollada esta enfocada en herramientas Oracle, sin embargo en el capítulo 2 se presenta la implementación para el estándar SQL3 permitiendo con esto, que la guía pueda ser aplicada con otras herramientas de desarrollo que utilicen el modelo Objeto Relacional.

Uno de los problemas que se presentaron en el desarrollo de la aplicación utilizando tecnologías Oracle fue la ausencia de una herramienta que permita integrar todas las fases de desarrollo, principalmente en el diseño lógico y generación del modelo objeto relacional a implementar en la base de datos.

RECOMENDACIONES

Para el desarrollo de aplicaciones Web con tecnología Objeto Relacional, se recomienda invertir un mayor esfuerzo en la fase de diseño de la aplicación, debido a la complejidad para abstraer y representar los objetos en la base de datos y para diseñar el sitio Web

En una guía técnica, se recomienda que sea sucinta y que contenga los temas necesarios para el desarrollo de las actividades y estrategias que se plantean en la misma.

Para la elaboración del modelo Objeto Relacional en Oracle 10g, se recomienda realizar una investigación minuciosa de los tipos de estructuras que permite implementar así como sus respectivas restricciones dependiendo de cada

necesidad, puesto que la guía muestra las más importantes y necesarias para nuestro caso de estudio.

Para controlar las referencias colgantes, se recomienda utilizar los constraints de integridad en cuanto sea posible ya que la utilización de triggers de control es mucho más compleja.

En la implementación del modelo Objeto Relacional en Oracle 10g, se recomienda realizar estudios complementarios de aspectos como: optimización, almacenamiento y criterios de implementación como la utilización de VARRAYs, tablas anidadas, asociaciones, índices, constraints, object view y REF.

Se recomienda realizar la evaluación de la guía como un estudio complementario con la finalidad de incluir nuevos criterios y lineamientos a la presente guía

Entre los puntos importantes que deben ser tomados en cuenta en el desarrollo de aplicaciones Web es el perfil del usuario/s que utilizará la aplicación, así como el diseño arquitectónico elegido el cual es resultado de análisis de la aplicación que se va a desarrollar, ya que de estos depende que la aplicación brinde la utilidad necesario para los usuarios que la requieren .

BIBLIOGRAFÍA

LIBROS

1. ARLOW J; NEUSTADT L; UML and the Unified Process Practical Object – Oriented Analisys & Design; Primera edición; Addison Wesley; 2002

2. BROWN BRADLEY, Oracle 8i, Desarrollo de Soluciones Web, Madrid Mac Graw Hill; 2001

3. CUEVA LOVELLE Juan Manuel, Introducción a UML, Lenguaje para modelar objetos, España-Oviedo, 2000

4. Dr. Il-Yeol Song, Object Oriented Analysis and Desing with UML Lecture Notes, College of Information Science and Technology, Drexel University USA, 2005.

5. GIETZ William, Oracle9i Application Developer's Guide Object-Relational Features Release 2 (9.2), Oracle Corporation, C.A., March 2002

6. LARMAN CRAIG, UML y Patrones, Prentice Hall, 2ª Edición, 2004

7. MARCOS E., VELA B., CAVERO J. M. AND CACERES P., Aggregation and composition in object-relational database design. Advanced Databases and Information Systems, September 2001

8. PRESMAN ROGER; Ingeniería de Software, un enfoque práctico; quinta edición; Mc. Graw Hill; 2002

TESIS

9. LOYOLA, Jaime. VALLEJO, Carlos, Vulnerabilidades en las redes de información y técnicas para enfrentarlas, Quito, 2005

DIRECCIONES ELECTRÓNICAS

10. ABAD Soraya, Tecnología Objeto Relacional, 20 Octubre 2006.
http://www.bd.cesma.usb.ve/ci5311/sd04/apuntesOR04.pdf

11. ANÖNIMO, Metodología de desarrollo de software Orientado por Objetos
http://www.cs.ualberta.ca/~pfiguero/soo/metod/

12. ANÖNIMO, Modelo Objeto Relacional, 2002
http://www-etsi2.ugr.es/depar/ccia/mabd/material/teoria/avan4.pdf

13. ANÖNIMO, Ingeniería de Software
http://www.angelfire.com/scifi/jzavalar/apuntes/IngSoftware.html

14. ANÖNIMO, SQL3, 2004
http://www.objs.com/x3h7/sql3.htm

15. ANÖNIMO, Oracle 9i O-R SQL: Multiple nested tables - Topic Powered by eve community, 2004
http://aixdoc.urz.uni-heidelberg.de/doc_link/en_US/local/oracle.8.1.6/appdev.816/a76976/adobjxmp.htm

16. CALLAN Steve, Deploying Forms on the Web, 2005
http://databasejournal.com/icom_includes/feeds/dbjournal/xml_front-10.xml

17. CALLAN Steve, Reports on the Web, 2005
http://www.databasejournal.com/RealMedia/ads/click_lx.ads/intm/it

18. MARTINEZ Javier, Ingeniería Web, Junio 2006
http://javimartinez.iespana.es/trabajoingenieriaweb.pdf

19. ESPERANZA Marcos, Diseño de Base de Datos Objeto-Relacionales, Junio 2006
http://kybele.escet.urjc.es/documentos/BD/T4-DisenyoBDOR_Alumnos.pdf

20. Oracle, Oracle Objects Example, Julio 2006
http://www.unix.org.ua/orelly/oracle/prog2/ch18_02.htm

21. Oracle, Basic Components of Oracle Objects, Julio 2006
http://www.stanford.edu/dept/itss/docs/oracle/10g/appdev.101/b10799/adobjbas.htm

22. Oracle, Object-Relational Model, Julio 2006
www-db.stanford.edu/~ullman/fcdb/oracle/or-objects.html

23. Oracle, Object Views, Julio 2006
http://www.unix.org.ua/orelly/oracle/prog2/ch20_01.htm

24. Oracle, Support for Collection Datatypes, Julio 2006
http://www.stanford.edu/dept/itss/docs/oracle/10g/appdev.101/b10799/adobjcol.htm

25. Oracle, Applying an Object Model to Relational Data, Julio 2006
http://www.stanford.edu/dept/itss/docs/oracle/10g/appdev.101/b10799/adobjvew.htm

26. Oracle, Connecting to Oracle Forms Server 9i, 2005
http://www.topxml.com/Biztalk-2006/rn-231519_Connecting-to-Oracle-Forms-Server-9i.aspx

27. Oracle, Webutil, 2005
http://sheikyerbouti.developpez.com/webutil-docs/Webutil_store_edit_docs.htm

28. Oracle, Design Considerations for Oracle Objects, Julio 2006
http://www.stanford.edu/dept/itss/docs/oracle/10g/appdev.101/b10799/adobjdes.htm.

29. Oracle, Modelo Objeto Relacional de Oracle, 2004
http://www.lania.mx/~jalba/BDD/ORACLE.doc.

30. Oracle, PL/SQL and Object Type, 2005
http://sales.esicom.com/sales/oracle/appdev.816/a76976/adobjint.htm

31. Oracle, Designer, 2005
http://ics02.leidenuniv.nl/902sol_rn/relnotes.902/a92187/toc.htm

32. Oracle, Internet Application Server, Noviembre 2002
http://www.oracle.com/technology/products/ias/daily/nov02.html

33. PERE Martra, UML, Junio 2006
http://www.programacion.com/tutorial/uml/7/

34. WIKIPEDIA, Estructura de Sitios Web, Junio 2006
http://www.wikilearning.com/como_se_puede_estructurar_un_sitio_web-wkccp-10112-6.htm.

35. Xavier Ferré Grau, Maria Isabel Sánchez Segura, Desarrollo Orientado a Objetos con UML, Junio 2004
http://www.willydev.net/descargas/articulos/general/umlTotal.pdf

ANEXOS

ANEXO A. Estudio de Metodologías

METODOLOGÍAS ORIENTADAS A OBJETOS

RUP (RATIONAL UNIFIED PROCESS)

Rational Unified Process (RUP) es una metodología con estructura flexible y ampliable que reúne las mejores prácticas de desarrollo para dar respaldo a proyectos de cualquier tamaño o alcance. Todos los tipos de proyectos pueden obtener buenos resultados con su incorporación.

RUP es una metodología basada en UML (Unified Modeling Language) que pretende implementar las mejores prácticas en la ingeniería de software, tales como:

- Gestión de Requerimientos
- Desarrollo de Software Iterativos
- Arquitectura basada en Componentes
- Desarrollo Visual de Software (Modela con UML)
- Verificación Continua de la Calidad del Software
- Gestión de Cambios

Gestión de Requerimientos

En la gestión de requerimientos RUP propone:

- Obtener los requerimientos y organizarlos
- Documentar los requerimientos de funcionalidad y restricciones
- Rastrear y documentar las decisiones
- Captar y comunicar requerimientos del negocio

Dentro de este proceso los casos de uso han probado ser una buena forma de captar requerimientos y guiar el diseño, la implementación y las pruebas.

Desarrollo de Software Iterativo

El proceso de desarrollo de software iterativo de RUP se estructura en fases (no como un modelo lineal de desarrollo de software), y cada fase se concluye con puntos de revisión, que son sumamente importantes ya que en estos puntos se revisan los requerimientos establecidos para cada fase, basados en los controles de calidad. De esta manera, si un producto o proceso no pasa el punto de revisión de calidad, se rediseña o se cancela, evitando así, los problemas de coste extra, y productos de mala calidad, que no satisfacen los requerimientos establecidos. Los puntos de revisión están basados en cuestionarios elaborados a partir de métricas establecidas o producto de la experiencia y de la investigación.

El proceso iterativo permite una compresión creciente de los requerimientos y aborda las tareas más riesgosas primero, con lo cual se logra reducir los riesgos y tener un subsistema ejecutable tempranamente.

Arquitectura basada en Componentes

Este proceso se basa en diseñar una arquitectura funcional lo más rápido posible. Esta arquitectura debe tener las siguientes características:

- Flexible
- Fácil de Usar
- Intuitivamente Comprensible
- Promueve la reutilización de componentes

Además apoya el desarrollo basado en componentes, tanto nuevos como preexistentes.

Desarrollo Visual de Software

En este proceso se hace un modelado visual de la estructura y comportamiento de la arquitectura y los componentes. Para el modelado visual RUP se basa en UML

El modelado visual tiene:

- Bloques de construcción:
  - Ocultan detalles
  - Permiten la comunicación en el equipo de desarrollo
  - Permiten analizar la consistencia:
    - Entre las componentes
    - Entre diseño e implementación

Verificación Continua de la Calidad del Software

Este proceso no solo verifica la funcionalidad en esencia sino también el rendimiento y la confiabilidad. Además ayuda a planificar, diseñar, implementar, ejecutar y evaluar las pruebas que aseguran la calidad como parte del desarrollo de software

Gestión de Cambios

- Los cambios son inevitables, pero es necesario evaluar si éstos son necesarios y rastrear su impacto.

- RUP indica como controlar, rastrear y monitorear los cambios dentro del proceso iterativo de desarrollo.

La metodología RUP tiene un ciclo de vida del proyecto que se divide en las siguientes fases:

- Fase de Concepción
- Fase de Elaboración
- Fase de Construcción
- Fase de Transición

Fase de Concepción

Está fase principalmente dirigida al entendimiento de los requerimientos y determinar el alcance del esfuerzo de desarrollo. En esta fase se define:

El alcance del sistema en donde se captura el contexto y los requerimientos más importantes así como las restricciones hasta el nivel en el cual se puedan derivar criterios de aceptación para el producto final, la planificación y estimación del coste para el desarrollo del proyecto, lo que incluye la elaboración de un plan de gestión y de un plan de trabajos que contempla el uso de los recursos, sintetizar una arquitectura candidata es que decir que el plan de trabajos se basa en una arquitectura candidata, que es la mejor solución en base a las restricciones de coste, plazo de desarrollo y tecnología disponibles. En algunos casos se puede requerir la construcción de un prototipo inicial, pero solo para demostrar la viabilidad de la solución y la selección de la organización del proyecto que incluye la metodología y herramientas a utilizar para el control y ejecución del mismo.

Fase de Elaboración

Esta fase consiste en planificar las actividades necesarias y los recursos requeridos, especificando las características y el diseño de la arquitectura del software. En esta fase se define y valida una arquitectura estable, que será la base para el resto del desarrollo, se hace un refinamiento de la visión del sistema, basándose en la nueva información que se obtenga durante esta fase, se establece una sólida comprensión de los casos de uso más críticos que influyen en las decisiones arquitectónicas y de planificación, se crean los planes de desarrollo detallados para las iteraciones de la fase de construcción, y se refina la arquitectura y se seleccionan los componentes, es decir que se evalúan y seleccionan los componentes más apropiados para su integración en la arquitectura seleccionada. Esta fase culmina con la arquitectura del ciclo de vida

Fase de Construcción

Esta fase consiste en el desarrollo del producto, hasta que este en primera versión y listo para ser enviado a los usuarios. En esta fase se define: la gestión de los recursos para optimizar y controlar los procesos de construcción del software, además se realiza y completa el desarrollo de los componentes y/o subsistemas,

los mismos que son puesto a prueba por un conjunto de criterios definidos al inicio del proyecto.

Fase de Transición

En esta fase se realizar la transición del producto a los usuarios, lo cual incluye: manufactura, envío, entrenamiento, soporte y mantenimiento del producto hasta que el cliente esté satisfecho. Esta fase termina con la versión de producto.

OOHDM (MÉTODO DE DISEÑO HIPERMEDIA ORIENTADO A OBJETOS)

El tamaño, la complejidad y el número de aplicaciones crecen en forma acelerada, por lo cual una metodología de diseño sistemática es necesaria para disminuir la complejidad y admitir la evolución y reusabilidad.

OOHDM es una metodología de diseño de aplicaciones hipermedia, orientado al desarrollo de aplicaciones Web. Esta metodología es utilizada para producir aplicaciones (galerías interactivas, presentaciones multimedia y aplicaciones Web) en las cuales el usuario pueda aprovechar el potencial del paradigma de la navegación de sitios Web, mientras se ejecuta transacciones sobre bases de información.

En primer lugar, la navegación posee algunos problemas. Una estructura de navegación robusta es una de las claves del éxito en las aplicaciones hipermedia. Si el usuario entiende dónde puede ir y cómo llegar al lugar deseado, es una buena señal de que la aplicación ha sido bien diseñada.

Construir la interfaz de una aplicación Web es una tarea compleja; no sólo se necesita especificar cuáles son los objetos de la interfaz que deberían ser implementados, sino también la manera en la cual estos objetos interactuarán con el resto de la aplicación.

En hipermedia existen requerimientos que deben ser satisfechos en un entorno de desarrollo unificado. Por un lado, la navegación y el comportamiento funcional de la aplicación deberían ser integrados. Por otro lado, durante el proceso de diseño se debería poder separar las decisiones de la estructura navegacional de la aplicación, de aquellas relacionadas con el modelo del dominio.

OOHDM propone el desarrollo de aplicaciones hipermedia a través de un proceso compuesto por cuatro etapas: diseño conceptual, diseño navegacional, diseño de interfaces abstractas e implementación.

Diseño Conceptual

Durante esta etapa se construye un esquema conceptual representado por los objetos del dominio (clases), las relaciones y colaboraciones existentes establecidas entre ellos (subsistemas). Las clases son descritas como en los modelos orientados a objetos tradicionales. Sin embargo, los atributos pueden ser de múltiples tipos para representar perspectivas diferentes de las mismas entidades del mundo real.

Se usa notación similar a UML (Lenguaje de Modelado Unificado) y tarjetas de clases y relaciones similares a las tarjetas CRC (Clase Responsabilidad Colaboración). El esquema de las clases consiste en un conjunto de clases conectadas por relaciones. Los objetos son instancias de las clases. Las clases son usadas durante el diseño navegacional para derivar nodos, y las relaciones que son usadas para construir enlaces.

Diseño Navegacional

La primera generación de aplicaciones Web fue pensada para realizar navegación a través del espacio de información, utilizando un simple modelo de datos de hipermedia.

En OOHDM, la navegación es considerada un paso crítico en el diseño de aplicaciones. Un modelo navegacional es construido como una vista sobre un diseño conceptual, admitiendo la construcción de modelos diferentes de acuerdo con los diferentes perfiles de usuarios. Cada modelo navegacional provee una vista subjetiva del diseño conceptual.

El diseño de navegación es expresado en dos esquemas: el esquema de clases navegacionales y el esquema de contextos navegacionales. En OOHDM existe un conjunto de tipos predefinidos de clases navegacionales que son: nodos, enlaces y estructuras de acceso. La semántica de los nodos y los enlaces son tradicionales de las aplicaciones hipermedia y las estructuras de acceso, tales como índices o recorridos guiados, representan los posibles caminos de acceso a los nodos.

La principal estructura del espacio navegacional es la noción de contexto navegacional. Un contexto navegacional es un conjunto de nodos, enlaces, clases de contextos, y otros contextos navegacionales (contextos anidados), los que pueden ser definidos por comprensión o extensión, o por enumeración de sus miembros.

Los contextos navegacionales juegan un rol similar a las colecciones y fueron inspirados sobre el concepto de contextos anidado y organizan el espacio navegacional en conjuntos convenientes que pueden ser recorridos en un orden particular y que deberían ser definidos como caminos para ayudar al usuario a lograr la tarea deseada.

Los nodos son enriquecidos con un conjunto de clases especiales que permiten de un nodo observar y presentar atributos, así como métodos cuando se navega en un contexto particular.

Diseño de Interfaz Abstracta

Una vez que las estructuras navegacionales son definidas, se debe especificar los aspectos de interfaz. Esto significa definir la forma en la cual los objetos navegacionales pueden aparecer, cómo los objetos de interfaz activarán la navegación y el resto de la funcionalidad de la aplicación, qué transformaciones de la interfaz son pertinentes y cuándo es necesario realizarlas.

Una clara separación entre diseño navegacional y diseño de interfaz abstracta es que permite construir diferentes interfaces para el mismo modelo navegacional, dejando un alto grado de independencia de la tecnología de interfaz de usuario.

El aspecto de la interfaz de usuario de aplicaciones interactivas (en particular las aplicaciones Web) es un punto crítico en el desarrollo que las modernas metodologías tienden a descuidar. En OOHDM se utiliza el diseño de interfaz abstracta para describir la interfaz del usuario de la aplicación de hipermedia.
El modelo de interfaz ADVs (Vista de Datos Abstracta) específica la organización y comportamiento de la interfaz, pero la apariencia física real, los atributos, y la disposición de las propiedades de las ADVs en la pantalla real son hechas en la fase de implementación.

Implementación

En esta fase se debe implementar el diseño. Hasta ahora, todos los modelos fueron construidos en forma independiente de la plataforma de implementación; en esta fase es tenido en cuenta el entorno particular en el cual se va a correr la aplicación.

El primer paso que se debe realizar dentro de esta fase es definir los ítems de información que son parte del dominio del problema. Debe identificar también, cómo son organizados los ítems de acuerdo con el perfil del usuario y su tarea; decidir qué interfaz debería ver y cómo debería comportarse. A fin de implementar todo en un entorno Web.

METODOLOGÍA DE ANÁLISIS Y DISEÑO ORIENTADO A OBJETOS

Análisis Orientado a Objetos

El análisis orientado a objetos pone énfasis en la investigación del problema y los requerimientos, es decir, presta especial atención en encontrar y descubrir los objetos o conceptos en el dominio del problema. El análisis orientado a objetos identifica las clases (atributos y comportamiento) así como las relaciones entre éstas y su posterior implementación en un leguaje de programación.

El análisis orientado a objetos propone la elaboración de los siguientes artefactos:

- Especificación de Requerimientos
- Diagrama de Casos de Uso
- Especificación de los Casos de Uso
- Modelo del Dominio
- Modelo de Clases de Análisis
- Diagrama de Colaboración
- Diagrama de Secuencia
- Diagramas de Estado

Diseño Orientado a Objetos

El diseño orientado a objetos pone énfasis en una solución conceptual que satisface los requerimientos de la aplicación, es decir, la definición de los objetos de software y como ellos colaboran para satisfaces los requerimientos de la aplicación.

En el diseño orientado a objetos existen varias etapas a saber que son:

- Entendimiento del Problema
- Identificar una o varias posibles soluciones
- Describir abstracciones de la solución de manera que el diseño este expresado en forma sencilla.

Entre las características principales del diseño orientado a objetos tenemos:

- Los objetos son abstracciones del mundo real o entidades del sistema.
- Los objetos son independientes y encapsulan el estado y la representación de la información
- Los objetos se comunican mediante el paso de parámetros.
- La funcionalidad del sistema utiliza los métodos de los objetos.

Entre los componentes del diseño orientado a objeto tenemos:

- Objetos, sus atributos y métodos
- La organización de los objetos dentro de una jerarquía.
- La especificación de interfaces de objetos

El diseño orientado a objetos propone elaborar los siguientes artefactos:

- Diagrama de Clases de Diseño
- Diseño de Navegación del Sistema
- Interfaces de usuario

Implementación Orientada a Objetos

En la implementación orientada a objetos los objetos de diseño son implementados utilizando un lenguaje de programación.

La implementación orientada a objetos propone la elaboración de los siguientes artefactos:

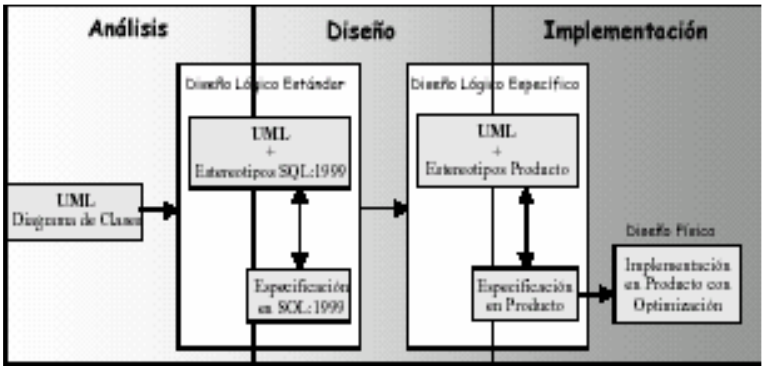
- Caracterización de las herramientas de desarrollo
- Implementación del Modelo Objeto Relacional
- Diagrama de Componentes
- Diagrama de Despliegue
- Estándares de Programación

- Pruebas del Sistema

METODOLOGÍA BASADA EN AGREGACIÓN Y COMPOSICIÓN

La metodología para el diseño de bases de datos objeto relacionales esta basado en la propuesta de Bertino, Cáceres y Marcos para el diseño de bases de datos orientadas a objetos. A continuación se muestran los principales pasos de la metodología.

FIGURA A-1: METODOLOGÍA PARA EL DISEÑO DE BASES DE DATOS OBJETO RELACIONALES



En la ***fase de análisis*** propone usar el diagrama de clases de UML para diseñar el esquema conceptual en lugar del modelo extendido Entidad-Relación debido a que UML es un lenguaje estándar para el diseño de sistemas orientados a objetos. A diferencia del modelo Entidad-Relación, UML tiene la ventaja de que permite el diseño de todo el sistema haciendo de esta manera más fácil la integración entre las diferentes vistas del sistema.

La ***fase de diseño*** esta dividida en dos pasos:

Diseño Estándar, es un diseño lógico independiente del producto de software que se vaya a utilizar para la implementación.

Diseño Específico, el cual es el diseño basado en un producto específico tal como (Oracle 10*g*, Informix, etc.) sin considerar afinamiento de la base de datos o tareas de optimización.

El diseño estándar es importante en el diseño de una base de datos objeto-relacional debido a que cada producto implementa un modelo objeto relacional diferente. Esta fase propone 2 alternativas: definir el esquema en SQL99, ya que este no depende de un producto específico de software y/o una notación grafica que describa un estándar para el modelo objeto relacional, el cual puede ser UML extendido con los estereotipos requeridos para SQL99 (estereotipos requeridos para el modelo objeto relacional).

La ***fase de implementación*** incluye las tareas de diseño físico, tareas de afinamiento de la base de datos y optimización de tiempos de respuesta y espacio de almacenamiento.

En esta fase la metodología propone utilizar las siguientes reglas para implementar los objetos del modelo objeto relacional:

TABLA A-1: GUÍA PARA EL DISEÑO DEL MODELO OBJETO RELACIONAL

UML	**SQL3**
Clase	Tipo Estructurado
Atributo o Primary Key <<PK>> o UNIQUE <<AK>> o Multivaluado <<MA>> o Compuesto o Calculado	Atributo o PK en la tabla o UNIQUE en la tabla o ARRAY o ROW o Trigger/Method
Asociación o Uno a Uno o Uno a Muchos o Muchos a Muchos	 o REF/REF o REF/ARRAY o ARRAY/ARRAY
Composición	ARRAY
Agregación	ARRAY
Generalización	Types/Typed Tables

ELABORADO POR LOS AUTORES: CABALLERO, Jorge. LEÓN, Pablo, Guía para el desarrollo de aplicaciones Web utilizando la tecnología Objeto-Relacional de Oracle, Quito 2006.

METODOLOGÍAS ÁGILES

Las metodologías ágiles de desarrollo de software tienen como objeto delinear los valores y principios que permitan desarrollar software rápidamente y respondiendo a los cambios que pueden surgir durante el proyecto, es decir, todo en el software cambia sus requisitos, el diseño, la tecnología, el equipo cambia, etc., el problema no es el cambio en sí mismo, puesto que sabemos que el cambio va a suceder, más bien el problema es la incapacidad de adaptarnos a dicho cambio cuando éste ocurre.

Las metodologías ágiles son muy efectivas aplicadas especialmente a proyectos pequeños donde el entorno del sistema es muy cambiante, requisitos volátiles y/o basados en nuevas tecnologías, donde se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad.

MSF (MICROSOFT SOLUTION FRAMEWORK)

Microsoft Solution Framework más que una metodología es un marco de trabajo que provee una estructura orientada a facilitar el análisis, diseño e implementación de soluciones tecnológicas. MSF permite exponer, revelar y manejar riesgos críticos, determinar los criterios de planificación, y establecer las interdependencias necesarias para una ejecución exitosa de los proyectos. MSF no es una metodología en el sentido estricto, con estructuras de trabajo, tareas y productos predeterminados más bien MSF provee mecanismos flexibles que mantiene una estrecha relación con los objetivos del negocio para aplicar soluciones adecuadas a los problemas tecnológicos.

MSF tiene las siguientes características:

- Punto de visión, para proveer la guía requerida para tomar decisiones técnicas.
- Puntos de referencia, para realizar un seguimiento efectivo del desarrollo de los procesos o proyectos, con énfasis en el manejo de los riesgos durante todo el ciclo de vida.

- Capacidad de reutilización, para tomar ventaja del conocimiento previo en forma estructurada y consistente en un ambiente tecnológico flexible.
- Adaptable a cualquier parte del proyecto
- Es Escalable ya que puede organizar equipos desde tres personas o más
- Es Flexible ya que utilizada en el ambiente de desarrollo de cualquier cliente.
- Utiliza Tecnología Agnóstica para desarrollar soluciones basadas sobre cualquier tecnología.

Modelos MSF

MSF está compuesto por 5 modelos. Los modelos son:

Modelo de Arquitectura del Proyecto: Está diseñado para acortar la planificación del ciclo de vida, este modelo define las pautas para construir proyectos empresariales a través del lanzamiento de versiones.

Modelo de Equipo: Está diseñado para mejorar el rendimiento del equipo de desarrollo y hacer frente a los nuevos cambios, es decir, proporciona una estructura flexible para organizar a los equipos de un proyecto. Puede ser escalado dependiendo del tamaño del proyecto y del equipo de personas disponibles. La idea del modelo es organizar a las personas y a las actividades que desempeñan para lograr el éxito del proyecto.

Este modelo se define como un equipo de pares no jerarquizados (“team of peers”) que trabajan en roles interdependientes y cooperativos. Cada miembro del equipo tiene un rol definido y una misión específica. Los líderes de cada equipo son responsables por la administración, guía y coordinación; mientras que los miembros realizan las tareas y objetivos delineados en el plan de trabajo.

Las metas asociadas a los roles son:

Administración de Programas: Controla la entrega de las restricciones del proyecto.

Desarrollo: Ejecución de las especificaciones del proyecto.

O

Pruebas: Aprobación de la versión solo si las deficiencias han sido corregidas o catalogadas.

Planeación y puesta en producción

Administración de versiones: Administración, planeación y distribución de las versiones puestas en producción.

Experiencia del Usuario: Mejorar el performance del usuario.

Administración del Producto: Satisfacción o enfoque al Cliente.

El modelo de equipo de MSF combina los roles para afrontar proyectos para lo cual define 2 equipos alternos: el funcional y el de desarrollo. El equipo funcional es unidisciplinario que se encarga de las áreas funcionales del proyecto como arquitectura, procesos y servicios administrativos, etc, El equipo de desarrollo es multidisciplinario y se enfoca en la construcción de soluciones.

Modelo de Proceso: Está diseñado para mejorar el control del proyecto, minimizando el riesgo, y aumentar la calidad acortando el tiempo de entrega. Proporciona una estructura de pautas a seguir en el ciclo de vida del proyecto, describiendo las fases, las actividades, la liberación de versiones y explicando su relación con el Modelo de equipo. Este modelo nos proporciona una descripción de alto nivel de las secuencias de actividades necesarias para el desarrollo de un software en vez de una serie de patrones inmutables, este modelo ofrece una gran flexibilidad en los procesos, por lo que puede ser aplicado en una amplia gama de proyectos de TI.

A continuación presentamos el ciclo de vida del modelo de procesos:

Visión: La visión identifica la situación deseada a futura y, en consecuencia, el camino a recorrer desde la situación actual hasta el estado deseado, es decir, la estrategia. .

Una vez que se establece la visión de una solución, se ha analizado la problemática del negocio y no de la tecnología, se identifican los beneficios y valores de una solución a un problema y los posibles factores de riesgo.

Planificación: La estrategia se materializa en el plan, que determina las actividades para desarrollar una versión de la solución, considera las conexiones con el desarrollo de fases subsiguientes, logrando así la coherencia y alineación con la visión definida. Para que el plan sea consistente, la fase contempla la especificación funcional de la solución que ha de ser construida y planificada, donde se detallan metas de la solución, requerimientos, características, arquitectura conceptual, recursos y otros que completen las especificaciones.

Desarrollo: Obtenido el plan, comienza el desarrollo de la solución que contempla la construcción paralela y secuenciada de los componentes del producto y el inicio de las pruebas sobre esos componentes, para garantizar dos aspectos: calidad técnica y satisfacción de los requerimientos del negocio.

Estabilización: La estabilización comienza con las pruebas beta del producto. La corrección de errores menores y el afinamiento de la solución para manejar los volúmenes reales son parte de este proceso.

Implementación: Se implementa la solución tecnológica y sus componentes, se transfiere el proyecto a producción y se obtiene la aprobación final del cliente sobre el proyecto o versión (en el caso de versión para la generación de nuevas versiones). Además se especifican sus características críticas y normales de operación así como los lineamientos de mantenimiento.

Modelo de Diseño del Proceso: Está diseñado para distinguir entre los objetivos empresariales y las necesidades del usuario. Proporciona un modelo centrado en el usuario para obtener un diseño eficiente y flexible a través de un enfoque iterativo. Las fases de diseño conceptual, lógico y físico proveen tres perspectivas diferentes para los tres tipos de roles: los usuarios, el equipo y los desarrolladores.

Modelo de Aplicación: Está diseñado para mejorar el desarrollo, el mantenimiento y el soporte de aplicaciones, proporciona tres tipos de servicios (servicios de usuarios, negocios y datos) para diseñar y desarrollar aplicaciones

software. Los servicios utilizados en este modelo son escalables, y pueden ser usados en un solo ordenador o incluso en varios servidores.

Disciplinas MSF

MSF incorpora tres disciplinas: Administración de proyectos, Gestión de Riesgos y Administración de la Disponibilidad que son importantes para el óptimo funcionamiento de los modelos de equipo y de proceso.

Administración de Proyectos: Describe la gestión del proyecto dentro del modelo de equipo de MSF, y permite obtener mayor escalabilidad, desde proyectos pequeños a proyectos largos y complejos. Esto implica la gestión de parámetros de calidad, costos, plazos y límites previamente acordados

Gestión del Riesgo: Está diseñado para ayudar al equipo a identificar las prioridades, tomar las decisiones estratégicas correctas y controlar las emergencias que puedan surgir. Este modelo proporciona un entorno estructurado para la toma de decisiones y acciones valorando los riesgos que puedan provocar. Además se reconoce que en el mismo ciclo de vida de desarrollo de software nos encontramos con situaciones inciertas y nos ayuda a lidiar con esa incertidumbre, evaluar el riesgo continuamente y constituir una base sólida para la toma de decisiones dentro de la organización del proyecto.

Administración de la Disponibilidad: A nivel organizacional, la disponibilidad se refiere al estado actual de las medidas colectivas de capacidades individuales. Esta información se usa en la planificación estratégica y evalúa la capacidad para lograr la adopción exitosa y realización de una inversión de tecnología.

La Disciplina de la disponibilidad refleja los principios de comunicación abierta. Esta disciplina reconoce esto y proyecta el cambio inherentemente al ambiente en que se desarrolla, así como al ambiente en que se entregara el proyecto

R

XP (EXTREME PROGRAMMING)

Es una de las metodologías de desarrollo de software más exitosas en la actualidad utilizada para proyectos de corto plazo y equipos pequeños. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto.

En XP se siguen un conjunto de técnicas y principios de sentido común a niveles extremos, entre las que podemos destacar:

- El código será revisado continuamente, mediante la programación en parejas (dos personas por máquina).
- Las pruebas se las hacen todo el tiempo, no sólo de cada nueva clase (pruebas unitarias) sino que también los clientes comprobarán que el proyecto cumple los requerimientos (pruebas funcionales).
- Las pruebas de integración se efectuarán siempre, antes de añadir cualquier nueva clase al proyecto, o después de modificar cualquiera existente (integración continua), para lo que se utilizan los frameworks de testing, como el xUnit.
- Se (re)diseñará todo el tiempo (refactoring), dejando el código siempre en el estado más simple posible.
- Las iteraciones serán radicalmente más cortas de lo que es usual en otros métodos, de manera que la retroalimentación sea tan a menudo como sea posible.

Además es necesario mencionar las historias de usuario que son la técnica utilizada en XP para especificar los requisitos del software. Para realizarlas se utilizan tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales.

El tratamiento de las historias de usuario es muy dinámico y flexible, se debe tener en cuenta que estas pueden ser modificadas, remplazadas y eliminadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en el tiempo definido.

El ciclo de vida ideal de XP consiste de 6 fases: Exploración, Planificación de la Entrega o Versiones, Iteraciones, Producción, Mantenimiento y Muerte del Proyecto.

Fases de XP

Fase de Exploración

En esta fase se definen las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto.

Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración implica un periodo corto de tiempo, todo esto dependiendo de la familiaridad que tengan los programadores con la tecnología y del alcance del proyecto.

Planificación de la Entrega

Se establecen prioridades de cada historia de usuario y se hacen las estimaciones para determinar el esfuerzo necesario de cada una de ellas. Se establece el alcance de la primera entrega y se determina un cronograma en conjunto con el cliente. Esta fase dura un período corto de tiempo.

Dentro de esta fase las estimaciones de esfuerzo son realizadas por los programadores utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos. Adicionalmente se lleva un registro de la “velocidad” de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos

correspondientes a las historias de usuario que fueron determinadas en la última iteración. La planificación se puede realizar basándose en el tiempo o el alcance.

Iteraciones

Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que permitan la creación de esta arquitectura, teniendo en cuenta que el cliente es quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción.

Producción

Se realizan pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase.

Las ideas que han sido propuestas y las sugerencias son documentadas para su posterior implementación.

Mantenimiento

Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente. De esta forma, la velocidad de desarrollo puede bajar después de la puesta del sistema en producción.

Muerte del Proyecto

El proyecto termina cuando se llega al punto en que el cliente no tiene más historias para ser incluidas en el sistema. Esto requiere que se satisfagan las

necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura. El proyecto también puede finalizar cuando se determina la no viabilidad del proyecto por motivos tales como incremento del costo o no se obtenga el beneficio esperado.

SELECCIÓN DE LA METODOLOGÍA

Para realizar la selección de la metodología se van a considerar algunos criterios para determinar cual es la más adecuada para nuestro estudio; entre los criterios que se van a analizar tenemos:

- Facilidad de Aplicación
- Flexibilidad
- Conocimiento de la metodología

Para la selección de la metodología se va a utilizar el PJA (Proceso Jerárquico Analítico). Que es un proceso que permite la solución de problemas complejos, estructurando una jerarquía de criterios y resultados extrayendo juicios para el desarrollo de prioridades, con lo que induce a la predicción de resultados probables.

Proceso Jerárquico Analítico

El PJA consta de los siguientes pasos:

Paso 1: Objetivo

Selección de una metodología para el desarrollo de aplicaciones Web que utilizan modelos Objeto Relacionales

Paso 2: Criterios

- Facilidad de Aplicación
- Flexibilidad
- Conocimiento de la metodología

Paso 3: Alternativas

- RUP
- OOHDM
- Análisis y Diseño Orientado a Objetos
- MSF
- XP

Paso 4: Matriz Binaria de Prioridades para los Criterios

	Facilidad de Aplicación	Flexibilidad	Conocimiento de la metodología
Facilidad de Aplicación	1/1	9/7	9/4
Flexibilidad	7/9	1/1	7/4
Conocimiento de la metodología	4/9	4/7	1/1

Paso 5: Cálculo del Vector de Pesos o Ranking

Vector de pesos =

$$\begin{pmatrix} 0.450 \\ 0.350 \\ 0.200 \end{pmatrix}$$

Paso 6 y Paso 7:

Por Criterio: Facilidad de Aplicación

Matriz Binaria de Prioridades para los criterios

	Análisis y Diseño Orientado a Objetos	RUP	OOHDM	MSF	**XP**
Análisis y Diseño Orientado a Objetos	1/1	9/7	9/2	9/4	9/5
RUP	7/9	1/1	7/2	7/4	7/5
OOHDM	2/9	2/7	1/1	2/4	2/5
MSF	4/9	4/7	4/2	1/1	4/5
XP	5/9	5/7	5/2	5/4	1/1

Cálculo del Vector de Pesos o Ranking

Vector Pesos =

X criterio

$$\begin{pmatrix} 0.333 \\ 0.259 \\ 0.074 \\ 0.148 \\ 0.185 \end{pmatrix}$$

Por Criterio: Flexibilidad

Matriz Binaria de Prioridades para los criterios

	Análisis y Diseño Orientado a Objetos	RUP	OOHDM	MSF	**XP**
Análisis y Diseño Orientado a Objetos	1/1	9/8	9/2	9/5	9/5
RUP	8/9	1/1	8/2	8/5	8/5
OOHDM	2/9	2/8	1/1	2/5	2/5
MSF	5/9	5/8	5/2	1/1	5/5
XP	5/9	5/8	5/2	5/5	1/1

Cálculo del Vector de Pesos o Ranking

Vector Pesos =

X criterio

0.310
0.276
0.069
0.172
0.172

Por Criterio: Conocimiento de la Metodología

Matriz Binaria de Prioridades para los criterios

	Análisis y Diseño Orientado a Objetos	RUP	OOHDM	MSF	**XP**
Análisis y Diseño Orientado a Objetos	1/1	1/1	9/2	9/4	9/4
RUP	1/1	1/1	9/2	9/4	9/4
OOHDM	2/9	2/9	1/1	2/4	2/4
MSF	4/9	4/9	4/2	1/1	4/4
XP	4/9	4/9	4/2	4/4	1/1

Cálculo del Vector de Pesos o Ranking

Vector Pesos =

X criterio

0.321
0.321
0.071
0.143
0.143

Paso 8: Formar la Matriz Criterios por Alternativas

	Facilidad de Aplicación	Flexibilidad	Conocimiento de la metodología
Análisis y Diseño Orientado a Objetos	0.333	0.310	0.321
RUP	0.259	0.276	0.321
OOHDM	0.074	0.069	0.071
MSF	0.148	0.172	0.143
XP	0.185	0.172	0.143

Paso 9: Multiplicar la matriz criterios alternativas por Vector de Pesos

$$\begin{pmatrix} 0.333 & 0.310 & 0.321 \\ 0.259 & 0.276 & 0.321 \\ 0.074 & 0.069 & 0.071 \\ 0.148 & 0.172 & 0.143 \\ 0.185 & 0.172 & 0.143 \end{pmatrix} X \begin{pmatrix} 0.450 \\ 0.350 \\ 0.200 \end{pmatrix} = \begin{pmatrix} 0.323 \\ 0.277 \\ 0.072 \\ 0.155 \\ 0.172 \end{pmatrix}$$

Paso 10: Determinación de la mejor Alternativa

Alternativas	**Valores**
Análisis y Diseño Orientado a Objetos	***0.323***
RUP	0.277
OOHDM	0.072
MSF	0.155
XP	0.172

La *METODOLOGÍA DE ANÁLISIS Y DISEÑO ORIENTADO A OBJETOS* es la seleccionada, por haber obtenido el mayor puntaje 0.323.

ANEXO B. Plantilla para las Especificaciones de Casos de Uso

MODO DE USO

A continuación daremos una explicación breve de cómo ingresar la información en la plantilla y el significado de cada ítem.

La plantilla consta de cuatro secciones: cabecera, flujo principal del caso de uso, flujos alternativos o relacionados al caso de uso y de un flujo de error que se da cuando el flujo principal del caso de uso ha fallado.

Para mayor compresión utilizaremos de ejemplo el caso de uso Login que es muy fácil de comprender.

CABECERA.- La cabecera de la plantilla proporciona información del caso de uso. A continuación detallaremos cada ítem de esta sección:

ESPECIFICACIÓN DEL CASO DE USO.- Ingresar el nombre del caso de uso.
Descripción General.- Ingresar el objetivo del caso de caso, es decir, que acción se desea describir con el caso de uso.
Actores.- Ingresar los actores (pueden ser usuarios de la aplicación u otros sistemas que se relacionan con la aplicación a desarrollar) que interviene en el caso de uso.
Pre-Condiciones.- Ingresar las condiciones que necesita el caso de uso para ejecutarse con éxito.
Pos-Condiciones.- Ingresar lo que se espera que realice el caso de uso después de su ejecución.

TABLA B-1: CABECERA DE LA PLANTILLA

ESPECIFICACION DEL CASO DE USO: LOGIN
Descripción general: Este caso de uso tiene por objetivo permitir que el actor Usuario se autentifique en el Sistema.
Actores: Usuario
Pre-Condiciones: La página principal del Sistema esta disponible para el actor Usuario

Pos-Condiciones: El actor Usuario se ha autentificado correctamente en el Sistema o la página de Login muestra el mensaje de error y se despliega nuevamente.

Fuente: Dr. Il-Yeol Song, Object Oriented Analysis and Desing with UML Lecture Notes, College of Information Science and Technology, Drexel University USA, 2005

Flujo Base o Principal.- Describe la interacción del actor con el sistema, es decir, las acciones que el actor realiza sobre el sistema. A continuación detallaremos cada parte esta sección:

Paso.- Es la numeración que se el asigna a cada una de las acciones. La numeración cuenta de dos partes: una de caracteres (FB que significa Flujo Base) y otra numérica (en salto de 10).

Actores.- Ingresar la acción que desea realizar el actor con el sistema. Los actores que intervienen en el caso de uso están definidos en la cabecera de la plantilla.

Sistema.- Ingresar la acción que el sistema realiza o ejecuta ante la petición del actor.

TABLA B-2: FLUJO BASE O PRINCIPAL

Paso	Actor(es)	Sistema
FB10	El Usuario hace click en el botón login de la pagina principal	
FB20		El sistema despliega la página Login
FB30	El usuario ingresa su Id y password y hace click en el botón Login	
FB40		El sistema valida la información de la cuenta del usuario y retorna la página principal

Fuente: Dr. Il-Yeol Song, Object Oriented Analysis and Desing with UML Lecture Notes, College of Information Science and Technology, Drexel University USA, 2005

Flujos Alternativos.- Son los casos de uso con los que se relaciona un caso de uso en particular. A continuación detallaremos cada parte esta sección

Paso.- Es la numeración que se el asigna a cada una de las acciones. La numeración cuenta de dos partes: una alfanumérica (FA que significa Flujo Alternativo seguido de un número que indica el número de flujos alternativos) y otra numérica (en salto de 10).

Actores.- Ingresar la acción que desea realizar el actor con el sistema. Los actores que intervienen en el caso de uso están definidos en la cabecera de la plantilla.

Sistema.- Ingresar la acción que el sistema realiza o ejecuta ante la petición del actor.

TABLA B-3: FLUJOS ALTERNATIVOS

Paso	Actor(es)	Sistema
FA1: Cliente desea abrir una cuenta		
FA1-10	Si el usuario hace click en el botón nueva cuenta en la página de Login	
FA1-20		El sistema invoca al caso de uso abrir cuenta
FA2: Cliente desea solicitar su palabra recordatoria		
FA2-10	Si el Usuario hace click en el botón de su palabra recordatoria en la página de Login.	
FA2-20		El sistema despliega la palabra recordatoria guardada para ese Usuario, en un cuadro de dialogo separado
FA2-30	Cuando el Usuario hace click en el botón OK	
		El sistema retorna al Usuario a la página de Login

Fuente: Dr. Il-Yeol Song, Object Oriented Analysis and Desing with UML Lecture Notes, College of Information Science and Technology, Drexel University USA, 2005

Flujos de Error.- Son los flujos de error que se consideran cuando el flujo principal de un caso de uso no se puede realizar con éxito. A continuación detallaremos cada parte de esta sección:

Paso.- Es la numeración que se el asigna a cada una de las acciones. La numeración cuenta de dos partes: una alfanumérica (FE que significa Flujos de Error seguido de un número que indica el número de Flujos de Error) y otra numérica (en salto de 10).

Actores.- Ingresar la descripción de la acción errada del actor. Los actores que intervienen en el caso de uso están definidos en la cabecera de la plantilla.

Sistema.- Ingresar la acción que el sistema realiza o ejecuta ante la petición del actor.

TABLA B-4: FLUJOS DE ERROR

Paso	Actor(es)	Sistema
FE1: User ID incorrecto		
FE1-10	Si el Usuario ingresa un Usuario Id que el sistema no reconoce	
FE1-20		El sistema despliega un mensaje al Usuario que ingrese un Id diferente o que haga click en el botón nueva cuenta
FE2: Password incorrecto: primero y segundo intento fallido		
FE2-10	Si el Usuario ingresa un password incorrecto	
FE2-20		El sistema despliega un mensaje al Usuario que ingrese nuevamente su password
FE3: Password incorrecto: tercer intento fallido		
FE3-10	Si el Usuario ingresa incorrectamente 3 veces el password	
FE3-20		El sistema despliega una página diciendo al Usuario que su servicio va a ser congelado.

Fuente: Dr. Il-Yeol Song, Object Oriented Analysis and Desing with UML Lecture Notes, College of Information Science and Technology, Drexel University USA, 2005

ANEXO C. Patrones de Diseño y Usabilidad

PATRONES DE USABILIDAD[27]

El proceso de diseño comienza con la construcción de un modelo de la arquitectura de software a partir de un conjunto de requisitos funcionales. Aunque los ingenieros de software no diseñan este modelo preliminar este se encamina para que sea poco fiable o tenga bajo rendimiento, la mayoría de requisitos no funcionales usualmente son revisados más tarde. Así, el diseño preliminar obtenido es evaluado con respecto a ciertos atributos de calidad, concretamente la usabilidad en este caso. Si el valor de usabilidad obtenido es adecuado con respecto a lo solicitado en los requisitos el diseño finalizaría y se pasaría a las etapas de desarrollo posteriores. Si la usabilidad necesita mejorarse, se entraría en un ciclo de mejora en la arquitectura software y nuevas evaluaciones, hasta que el nivel de usabilidad sea el adecuado. Este proceso no implica que no se evalúe la usabilidad del sistema una vez terminado, simplemente se pretende adelantar el ciclo de evaluación/mejora de la usabilidad con el fin de ahorrar esfuerzos en el proceso de desarrollo.

ATRIBUTOS, PROPIEDADES Y PATRONES DE USABILIDAD

La usabilidad de los sistemas software se suele evaluar sobre el sistema finalizado, intentando asignar valores a los atributos de usabilidad clásicos: aprendizaje, eficiencia, fiabilidad y satisfacción.

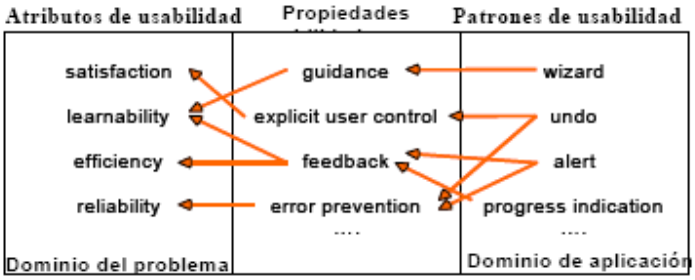
Sin embargo, el nivel de abstracción de estos atributos de usabilidad es demasiado alto como para poder estudiar los mecanismos que deben aplicarse a la arquitectura del software para mejorarlos.

Por tanto, la filosofía que se propone ha sido descomponer estos atributos en dos niveles intermedios de conceptos que se acercan más a la solución software: las propiedades de usabilidad y los patrones de usabilidad.

[27] www.willydev.net/descargas/prev/PatronesUsa.pdf

El primer nivel consiste en relacionar los atributos de usabilidad antes mencionados con unas propiedades de usabilidad específicas que determinan las características de usabilidad que han de ser mejoradas en el sistema. El segundo nivel está planteado para identificar unos mecanismos concretos que podrían incorporarse a la arquitectura de software para mejorar la usabilidad del sistema final. Estos mecanismos se han denominado patrones de usabilidad y abordan alguna necesidad indicada por una propiedad de usabilidad. El procedimiento que se ha seguido para identificar la relación entre los atributos, las propiedades y los patrones de usabilidad se describe en detalle en la Tabla C-1, se presenta un subconjunto de dicha relación. Esta tabla muestra cómo las propiedades de usabilidad relacionan los patrones con los atributos de usabilidad en un sentido cualitativo (una flecha indica que una propiedad afecta positivamente a un atributo, es decir, mejora este atributo).

TABLA C-1: RELACIONES ENTRE ATRIBUTOS, PROPIEDADES Y PATRONES



En la segunda columna de la Tabla C-2 se enumeran los patrones de usabilidad propuestos. La primera columna de la tabla indica las propiedades de usabilidad que se relacionan con cada patrón.

TABLA C-2: LISTA DE PATRONES DE USABILIDAD

Propiedad de Usabilidad	Patrón de Usabilidad
NATURAL MAPPING	
CONSISTENCY (functional, interface, evolutionary)	
ACCESSIBILITY (internationalisation)	Different languages
CONSISTENCY, ACCSESIBILITY (multichannel, disabilities)	Different access methods
FEEDBACK	Alert
ERROR MANAGEMENT, FEEDBACK	Status indication
EXPLICIT USER CONTROL, ADAPTABILITY (user expertise)	Shortcuts (key and tasks)
ERROR MANAGEMENT (error prevention)	Form/field validation
ERROR MANAGEMENT (error correction),	Undo
GUIDANCE, ERROR MANAGEMENT	Context-sensitive help
GUIDANCE, ERROR MANAGEMENT	Wizard
GUIDANCE, ERROR MANAGEMENT	Standard help
GUIDANCE, ERROR MANAGEMENT	Tour
MINIMISE COGNITIVE LOAD, ADAPTABILITY, ERROR MANAGEMENT (error prevention)	Workflow model
ERROR MANAGEMENT (error correction)	History l ogging
GUIDANCE, ERROR MANAGEMENT (error prevention)	Provision of views
ADAPTABILITY (user preferences)	User profile
ERROR MANAGEMENT, EXPLICIT USER CONTROL	Cancel
EXPLICIT USER CONTROL	Multi-tasking
MINIMISE COGNITIVE LOAD ERROR MANAGEMENT (error prevention)	Commands aggregation
EXPLICIT USER CONTROL	Action f or multiple objects
MINIMISE COGNITIVE LOAD, ERROR MANAGEMENT (error prevention)	Reuse information

PATRONES ARQUITECTÓNICOS DE USABILIDAD

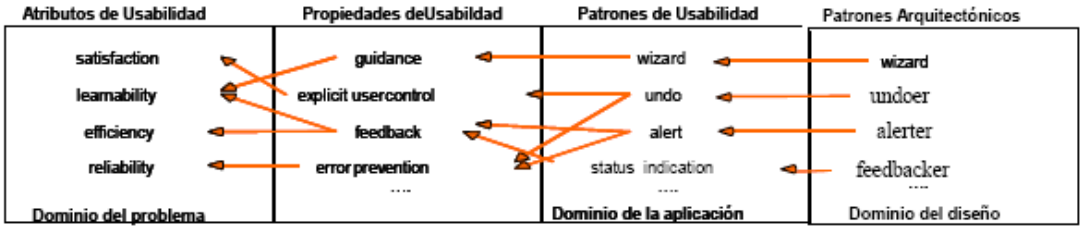
El concepto de patrón más ampliamente utilizado en el desarrollo del software es el patrón de diseño, que se utiliza particularmente en el paradigma orientado a objetos. En este contexto, un patrón de diseño es una descripción de las clases y de los objetos que trabajan conjuntamente para resolver un problema concreto. Estos patrones muestran una solución a un problema, que se ha obtenido a partir de su uso en otras aplicaciones diferentes.

En esta propuesta además de la idea del patrón de usabilidad, también se usa el concepto de patrón arquitectónico. Dado que el patrón de usabilidad se ha definido como un mecanismo a ser incorporado en el diseño de una arquitectura software a fin de abordar una propiedad de usabilidad concreta, un patrón arquitectónico determinará cómo se incorporará este patrón de usabilidad en una arquitectura de software; es decir, qué efecto tendrá la inclusión del patrón de usabilidad en los

componentes de la arquitectura del sistema. Al igual que los patrones de diseño, los patrones arquitectónicos reflejarán una posible solución a un problema: la incorporación de un patrón de usabilidad concreto en un diseño de software.

Por lo tanto, el patrón arquitectónico es el último eslabón en la cadena atributo, propiedad y patrón de usabilidad, y conecta la usabilidad del sistema software con la arquitectura de éste. Así, se puede añadir otra columna a la Tabla C-1, tal como se muestra en la Tabla C-3.

TABLA C-3. RELACIONES ENTRE LOS ATRIBUTOS, PROPIEDADES Y PATRONES DE USABILIDAD Y PATRONES ARQUITECTÓNICOS



PROCEDIMIENTO DE OBTENCIÓN DE LOS PATRONES ARQUITECTÓNICOS DE USABILIDAD

A continuación, se describe el procedimiento que se ha seguido para identificar los patrones arquitectónicos que representan soluciones de diseño para los patrones de usabilidad propuestos. Este procedimiento se compone de dos partes:

1. La aplicación de un proceso de inducción para abstraer los patrones arquitectónicos a partir de unos diseños particulares para varios proyectos desarrollados tanto por investigadores como por profesionales. Para ello, se adoptaron los siguientes pasos:

    1.1. Se construyeron modelos de diseño para varios sistemas sin incluir los patrones de usabilidad.

1.2. Para cada patrón de usabilidad, se modificaro los diseños anteriores para incluir la funcionalidad correspondiente a los patrones considerados.

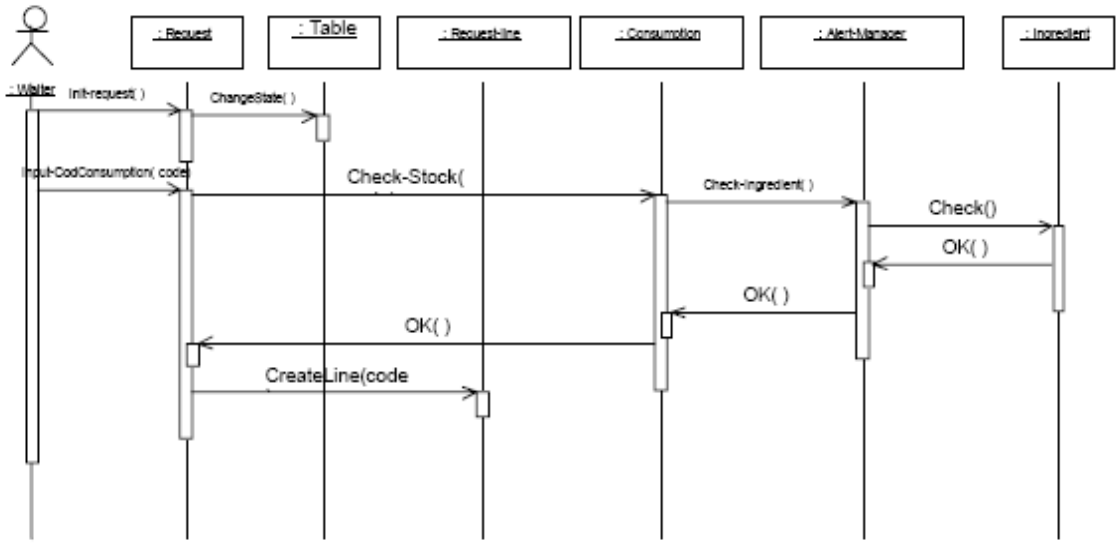
1.3. Para cada patrón de usabilidad, mediante un proceso de abstracción, se obtuvo el patrón arquitectónico correspondiente a partir de las modificaciones introducidas por los desarrolladores en el diseño.

2. La aplicación de los patrones arquitectónicos resultantes del paso anterior a varios desarrollos para validar su viabilidad.

Para ilustrar este proceso, se muestra a continuación la inducción del patrón arquitectónico relacionado con el patrón de usabilidad indication a partir de una aplicación de gestión de pedidos y mesas de un restaurante.

El diagrama de secuencia representado en la Figura C-1 muestra parte del diseño de esta aplicación, concretamente la parte relacionada con la introducción del menú pedido por el cliente del restaurante.

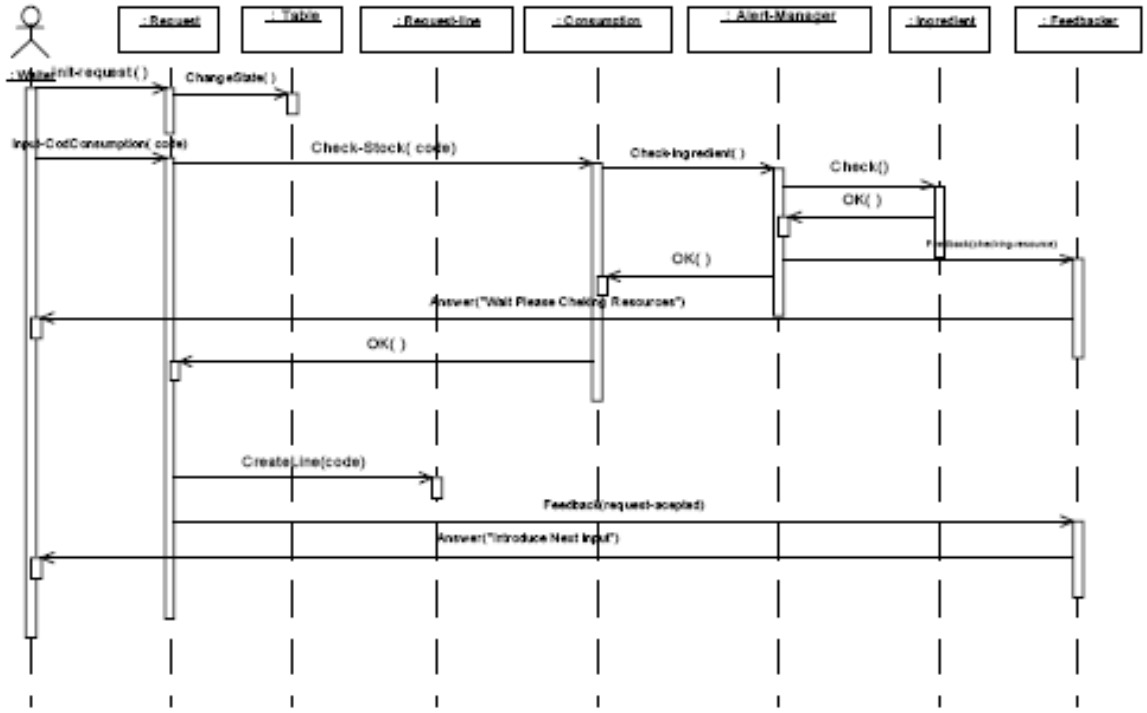
FIGURA C-1: DIAGRAMA DE INTERACCIÓN SIN PATRÓN DE USABILIDAD



La Figura C-1 representa el intercambio de mensajes entre las clases del sistema de gestión de restaurantes a la hora de hacer la petición de consumiciones, no teniendo en cuenta que el sistema deba proporcional al usuario ningún tipo de información sobre lo que está ocurriendo al pedirse sus consumiciones.

La Figura C-2 muestra el diagrama de secuencia, considerando en este caso el efecto de incluir, como requisito de usabilidad, el patrón de usabilidad indication para la funcionalidad anterior. Se aprecia cómo la incorporación de la clase feedbacker proporciona al usuario la información sobre lo que está haciendo el sistema. En este caso el camarero (waiter), no se impacientará por incluir otra consumición porque sabrá que el sistema está comprobando que existen los ingredientes necesarios para cocinar la consumición solicitada y además sabrá en qué momento el sistema le está permitiendo introducir un nuevo código de consumición para los comensales.

FIGURA C-2: DIAGRAMA DE INTERACCIÓN CON PATRÓN DE USABILIDAD



• Ventajas para la usabilidad – Descripción de los aspectos de usabilidad (propiedades de usabilidad) que pueden mejorarse al incluir el patrón apropiado.

• Nombre del patrón arquitectónico – Los patrones deben tener nombres sugestivos que den una idea del problema que abordan y de su solución.

• Problema – Describe cuándo aplicar el patrón y en qué contexto. En el caso de los patrones arquitectónicos, el problema se refiere a un patrón de usabilidad concreto que ha de ser materializado.

• Solución – Describe los elementos que conforman la arquitectura, sus relaciones, sus responsabilidades, etc. la solución para un patrón específico se hará explícita a partir de una:

- Representación gráfica – Una figura que representa los componentes de la arquitectura y sus interacciones. Las flechas numeradas entre los distintos componentes representarán las interacciones. Las flechas con las líneas ininterrumpidas indican el flujo de datos, mientras que las

líneas interrumpidas representan el flujo de control entre los componentes.

- Participantes – Una descripción de los componentes que participan en la solución propuesta y las iteraciones (representadas por flechas) para determinar cómo deben asumir sus responsabilidades.

• Beneficios de usabilidad – Una argumentación razonada acerca del impacto que tiene la aplicación del patrón en la usabilidad.

PATRONES DE DISEÑO[28]

Un patrón de diseño es:

• una solución estándar para un problema común de programación
• una técnica para flexibilizar el código haciéndolo satisfacer ciertos criterios
• un proyecto o estructura de implementación que logra una finalidad determinada
• un lenguaje de programación de alto nivel
• una manera más práctica de describir ciertos aspectos de la organización de un programa
• conexiones entre componentes de programas
• la forma de un diagrama de objeto o de un modelo de objeto.

Se van a presentar algunos ejemplos de patrones de diseño. A cada diseño de proyecto le sigue el problema que trata de resolver, la solución que aporta y las posibles desventajas asociadas. Un desarrollador debe buscar un equilibrio entre las ventajas y las desventajas a la hora de decidir que patrón utilizar. Lo normal es, como se observará a menudo en la ciencia computacional y en otros campos, buscar el balance entre flexibilidad y rendimiento.

[28] http://es.wikipedia.org/wiki/Patr%C3%B3n_de_dise%C3%B1o

CUANDO (NO) UTILIZAR PATRONES DE DISEÑO

La primera regla de los patrones de diseño coincide con la primera regla de la optimización: retrasar. Del mismo modo que no es aconsejable optimizar prematuramente, no se deben utilizar patrones de diseño antes de tiempo. Seguramente sea mejor implementar algo primero y asegurarse de que funciona, para luego utilizar el patrón de diseño para mejorar las flaquezas; esto es cierto, sobre todo, cuando aún no ha identificado todos los detalles del proyecto (si comprende totalmente el dominio y el problema, tal vez sea razonable utilizar patrones desde el principio, de igual modo que tiene sentido utilizar los algoritmos más eficientes desde el comienzo en algunas aplicaciones).

Los patrones de diseño pueden incrementar o disminuir la capacidad de comprensión de un diseño o de una implementación, disminuirla al añadir accesos indirectos o aumentar la cantidad de código, disminuirla al regular la modularidad, separar mejor los conceptos y simplificar la descripción. Una vez que aprendido el vocabulario de los patrones de diseño será más fácil y más rápido comunicarse con otros individuos que también lo conozcan.

La mayoría de las personas utiliza patrones de diseño cuando perciben un problema en su proyecto o algo que debería resultar sencillo y no lo es o su implementación como por ejemplo, el rendimiento.

Los patrones de Diseño se clasifican en:

- Patrones Creacionales
- Patrones Estructurales
- Patrones de Comportamiento

PATRONES CREACIONALES

- Abstract Factory (Fábrica abstracta): Permite trabajar con objetos de distintas familias de manera que las familias no se mezclen entre sí y haciendo transparente el tipo de familia concreta que se esté usando.

- Builder (Constructor virtual): Abstrae el proceso de creación de un objeto complejo, centralizando dicho proceso en un único punto.
- Factory Method (Método de fabricación): Centraliza en una clase constructora la creación de objetos de un subtipo de un tipo determinado, ocultando al usuario la casuística para elegir el subtipo que crear.
- Prototype (Prototipo): Crea nuevos objetos clonándolos de una instancia ya existente.
- Singleton (Instancia única): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia.

PATRONES ESTRUCTURALES

- Adapter (Adaptador): Adapta una interfaz para que pueda ser utilizada por una clase que de otro modo no podría utilizarla.
- Bridge (Puente): Desacopla una abstracción de su implementación.
- Composite (Objeto compuesto): Permite tratar objetos compuestos como si de uno simple se tratase.
- Decorator (Envoltorio): Añade funcionalidad a una clase dinámicamente.
- Facade (Fachada): Provee de una interfaz unificada simple para acceder a una interfaz o grupo de interfaces de un subsistema.
- Flyweight (Peso ligero): Reduce la redundancia cuando gran cantidad de objetos poseen idéntica información.
- Proxy : Mantiene un representante de un objeto.

PATRONES DE COMPORTAMIENTO

- Chain of Responsibility (Cadena de responsabilidad): Permite establecer la línea que deben llevar los mensajes para que los objetos realicen la tarea indicada.
- Command (Comando): Encapsula una operación en un objeto, permitiendo ejecutar dicha operación sin necesidad de conocer el contenido de la misma.

- Interpreter (Intérprete): Dado un lenguaje, define una gramática para dicho lenguaje, así como las herramientas necesarias para interpretarlo.
- Iterator (Iterador): Permite realizar recorridos sobre objetos compuestos independientemente de la implementación de estos.
- Mediator (Mediador): Define un objeto que coordine la comunicación entre objetos de distintas clases, pero que funcionan como un conjunto.
- Memento (Recuerdo): Permite volver a estados anteriores del sistema.
- Observer (Observador): Define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambie de estado se notifique y actualicen automáticamente todos los objetos que dependen de él.
- State (Estado): Permite que un objeto modifique su comportamiento cada vez que cambie su estado interno.
- Strategy (Estrategia): Permite disponer de varios métodos para resolver un problema y elegir cúal utilizar en tiempo de ejecución.
- Template Method (Método plantilla): Define en una operación el esqueleto de un algoritmo, delegando en las subclases algunos de sus pasos, esto permite que las subclases redefinan ciertos pasos de un algoritmo sin cambiar su estructura.
- Visitor (Visitante): Este patrón sirve para indicar que alguien(visitador) visitó a otra cosa, y así saber que eso ha sido visitado, por un visitador

ANEXO D. Plantilla para los Casos de Prueba

MODO DE USO

A continuación daremos una explicación breve de cómo ingresar la información en las plantillas el significado de cada sección.

Para mayor compresión utilizaremos de ejemplo el caso de prueba para el caso de uso Administrar Inspector.

La tabla D-1 establece el Caso de Prueba para un caso de uso específico. Se debe realizar un caso de prueba por cada caso de uso. La Tabla D-1 consta de 2 Columnas: En la primera columna ingresamos el Caso de Uso que va a ser sujeto a un procedimiento de Pruebas; En la columna 2 ingresamos el nombre del caso de prueba con su respectivo procedimiento numerado como se muestra a continuación:

TABLA D-1: CASOS DE PRUEBA PARA CASO DE USO ADMINISTRAR INSPECTOR

Procedimiento de Prueba para Caso de Uso Administrar Inspector	Caso de Prueba Nuevo: 1. Seleccione del menú principal la opción Inspector y escoja la opción Administrar Inspector, se abre la ventana Inspector 2. Pulse sobre el botón Nuevo 3. Ingrese el nombre, cédula, dirección, teléfono y celular del inspector 4. Escoja el estado del inspector activo o pasivo 5. Pulse sobre el botón Grabar Caso de Prueba Editar: 1. Seleccione del menú principal la opción Inspector y escoja la opción Administrar Inspector, se abre la ventana Inspector 2. Seleccione el inspector al cual quiere editar sus datos 3. Pulse sobre el botón Editar 4. Modifique los datos de nombre, dirección, teléfono y celular del inspector

	5. Escoja el nuevo estado del inspector activo o pasivo 6. Pulse sobre el botón Grabar Caso de Prueba Eliminar: 1. Seleccione del menú principal la opción Inspector y escoja la opción Administrar Inspector, se abre la ventana Inspector 2. Seleccione el inspector que quiere eliminar de la tabla inspector 3. Pulse sobre el botón Eliminar 4. Acepte la advertencia

Fuente: Dr. Il-Yeol Song, Object Oriented Analysis and Desing with UML Lecture Notes, College of Information Science and Technology, Drexel University USA, 2005

Una vez establecido el procedimiento para el caso de prueba, ejecutamos el procedimiento y sus resultados los ponemos en las Tablas D-1.1 y D-1.2.

La Tabla D-1.1 muestra los resultados esperados mientras que la Tabla D-1.2 muestra los resultados obtenidos. Cuando los resultados esperados y los obtenidos son iguales la prueba esta concluida caso contrario la prueba queda pendiente hasta que se realicen las debidas correcciones.

A continuación detallaremos cada sección de las Tablas D-1.1 y D-1.2:

Descripción General.- Ingresar la Descripción del Caso de Prueba
Datos de Entrada.- Ingresar los datos de entrada que interviene en el caso de Prueba
Resultados Esperados.- Ingresar los resultados que se desean obtener después de ejecutar el caso de prueba.
Condiciones para esta Prueba.- Ingresar todas las condiciones necesarias para que el caso de prueba se pueda realizar con éxito.
Fechas de Ejecución de la Prueba.- Ingresar la o las fechas de Ejecución del caso de prueba.
Responsables de Ejecución de la Prueba.- Ingresar el o los nombres de los responsables de la Ejecución de la prueba.
Resultados Obtenidos.- Ingresar la descripción de los resultados obtenidos luego de ejecutar la prueba.

TABLA D-1.1: CASO DE PRUEBA #1 NUEVO INSPECTOR

Descripción general:	Registrar un nuevo inspector
Datos de Entrada:	Nombre: Pedro Altamirano Cédula: 1714409602 Dirección: Valle de los Chillos, San Rafael 2da transversal Teléfono: 2862596 Celular: 097412584 Activo: seleccionado Pasivo: en blanco
Resultados Esperados:	Los datos del inspector se registrarán en la tabla inspector
Condiciones para esta prueba:	El inspector a ingresar no debe estar ya registrado. Los datos de entrada deber ser validos.
Fechas de Ejecución de la Prueba:	*Fecha #1: 15 de marzo 2006*
Responsables de Ejecución de la Prueba:	*Responsable #1: Ing. Luis Luna*
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Fuente: Dr. Il-Yeol Song, Object Oriented Analysis and Desing with UML Lecture Notes, College of Information Science and Technology, Drexel University USA, 2005

TABLA D-1.2: CASO DE PRUEBA #2 NUEVO INSPECTOR

Descripción general:	Ingreso fallido de nuevo inspector
Datos de Entrada:	Nombre: en blanco Cédula: 161020923 Dirección: Quito, San Rafael, sector el Triangulo Teléfono: 2833333 Celular: en blanco Activo: seleccionado Pasivo: en blanco
Resultados Esperados:	El sistema no debe permitir registrar el nuevo inspector porque el casillero nombre está vacío y la cédula es menor a 10 dígitos, el sistema debe emitir mensajes de error acordes.
Condiciones para esta prueba:	No ingresar nombre del inspector Ingresar una cedula de 9 dígitos
Fechas de Ejecución de la Prueba:	*Fecha #1: 15 de marzo 2006* *Fecha#2: 17 de marzo 2005*
Responsables de Ejecución de la	*Responsable #1: Ing. Luis Luna* *Responsable #2: Ing. Luis Luna*

Prueba:	
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos no correspondieron a los resultados esperados pues aunque el sistema no registro el inspector, el mensaje de error no fue acorde, por lo cual se concluye que la prueba queda pendiente hasta que se corrija el texto del mensaje de error.* *Resultados obtenidos #2: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Fuente: Dr. Il-Yeol Song, Object Oriented Analysis and Desing with UML Lecture Notes, College of Information Science and Technology, Drexel University USA, 2005

ANEXO E. Descripción de las Clases, Atributos y Métodos de la Aplicación SIABI

CLASES	ATRIBUTO Y OPERACION	DESCRIPCION
LOGIN		**Clase utilizada para registrar el usuario y la contraseña de un Administrador/Bibliotecario**
	Usuario	Se utiliza para registra el nombre de un usuario del sistema
	Password	Se lo utiliza para registrar la contraseña de un usuario del sistema
	crearLogin	Método que se la utiliza para crear el login de un usuario del sistema
	actualizarLogin	Método que se la utiliza para actualizar el login de un usuario del sistema
BIBLIOTECARIO		**Clase que contiene la información del Administrador/Bibliotecario**
	id	Cédula de identidad o identificador único del Biblioteario en el sistema
	Estado	Estado del Administrador o Bibliotecario en el sistema, el cual puede ser activo o inactivo
	Ingreso	Fecha en la que ingresa un nuevo usuario en el sistema
	Salida	Fecha en la cual pasa a ser inactivo un usuario en el sistema
	Tipo	Indica si el Usuario del Sistema es Administrador o Bibliotecario
	Nombre	Registra el nombre de un Administrador o Bibliotecario
	crearBibliotecario	Método que permite crear un Administrador o Bibliotecarios en el sistema
	actualizarBibliotecario	Método que permite actualizar la información de un Administrador o Bibliotecario.
	buscarBibliotecario	Método que permite buscar la información de un Administrador o Bibliotecario

USUARIO CONSULTA		**Clase que registra la información de los usuarios de la biblioteca a quienes se presta los Ejemplares**
	Id	Identificador único del usuario de la Biblioteca
	Nombre	Nombre del Usuario de la Biblioteca al cual se realiza el préstamo de un Ejemplar
	Direccion	Dirección del Usuario de la Biblioteca al cual se le realiza el préstamo de un Ejemplar
	Telefono	Número de Teléfono del usuario de la Biblioteca al que se le realiza el préstamo de un Ejemplar
	Estado	Indica el estado del usuario de la biblioteca en el sistema, el cual puede ser activo o inactivo
	crearUsuario	Método que permite registrar la información de un usuario de la biblioteca
	actualizarUsuario	Método que permite realizar cambios en la información de un usuario de la biblioteca.
	buscarUsuario	Método que permite buscar la información de un usuario de la biblioteca.
RESERVACION		**Clase que administra la reservación de un Ejemplar**
	Id	Identificador único de la reservación
	Fecha	fecha para la cual se realiza la reservación de un Ejemplar
	crearReservacion	Método que permite crear la reservación de un Ejemplar
	consultarReservacion	Método que permite consultar la reservación de un Ejemplar
MULTA		**Clase que registra las multas por préstamo**
	id	Identificador único de la multa
	Monto	Cantidad total que se debe pagar por el atraso en la devolución de un Ejemplar.
	Motivo	Registra el motivo por el cual usuario de la biblioteca se atraso en la devolución del Ejemplar.
	Estado	Indica si la multa fue pagada o no.

	pagarMulta	Método que permite cambiar el estado de la multa (pagado)
	calcularMulta	Método que calcula el valor total de la multa en base a los días de atraso.
CATEGORIA		**Clase que permite distinguir los diferentes tipos de categorías para préstamos de Ejemplares y el tiempo de préstamo.**
	Id	Identificador único del tipo de categoría.
	Valor	Es el costo por día que tendrá la multa por la no devolución del Ejemplar a tiempo
	Dias	El número de días que se puede prestar el Ejemplar.
	Nombre	Nombre de la categoría.
	Estado	Indica el estado de la categoría, si esta activa o inactiva
	crearCategoria	Método que permite crear nuevos tipos de categorías
	actualizarCategoria	Método que permite actualizar la información ingresados de una categoría ya existentes.
	buscarCategoria	Método que permite busca la información de una categoría
PRESTAMO		**Clase que administra los préstamos de los Ejemplares**
	Id	Identificador único del préstamo.
	Fecha	Fecha en la que se realiza el préstamo
	Caducidad	Fecha en la que se debería entregar el Ejemplar.
	devolución	Fecha en la que se entrega el Ejemplar
	crearPrestamo	Método que permite crear un Préstamo
	devolverPrestamo	Método que permite registrar la fecha de devolución del Ejemplar.
	verificarMulta	Método que permite verificar si un usuario de la biblioteca tiene una multa por un Ejemplar que no devolvió a tiempo
	renovarPrestamo	Método que permite renovar el préstamo de un Ejemplar
	verificarFecha	Método que permite determinar

		si existen días de retraso en la entrega de un Ejemplar así como el número de días de atraso.
	buscarPrestamo	Método que permite buscar la información de un préstamo.
EJEMPLAR		**Clase que permite instanciar un documento para realizar un préstamo ya que pueden existir varios ejemplares de un mismo documento**
	Id	Identificador único de un ejemplar
	original	Indica si un ejemplar es original o copia.
	estado_prestamo	Indica el estado de un préstamo, prestado o disponible.
	estado_ejemplar	Indica si el Ejemplar se encuentra en buen estado o no, o si este está perdido.
	crearEjemplar	Método que permite crear un Ejemplar
	buscarEjemplar	Método que permite saber si existe o no un ejemplar.
	actualizarEstadoPrestamo	Método que permite cambiar el estado del préstamo
	verificarEstadoEjemplar	Método que permite cambiar el estado del Ejemplar
DOCUMENTO		**Clase de la cual heredan atributos y métodos las demás clases que contienen información de los diferentes tipos documentos de la Biblioteca**
	Id	Identifica de manera única a un tipo de documento.
	Titulo	Atributo que toma el titulo del documento.
	fecha_ingreso	Fecha en la cual se ingresa el documento a la biblioteca.
	Contenido	Muestra el listado de la información relevante del documento, es decir el índice del documento
	Año	Año en el que se publicó el documento
	crearDocumento	Método que permite crear un nuevo documento.
	actualizarDocumento	Método que permite actualizar la información de un documento existente.

	buscarDocumento	Método que permite buscar la información de un documento
	visualizarContenido	Método que permite visualizar el contenido de un documento
LIBRO		**Clase que hereda los atributos de la clase DOCUMENTO destinada para los Libros**
	Isbn	Identificador único de un libro
	Editorial	Registra el editorial del libro
	Edicion	Registra la edición del libro
	Paginas	Registra el número de páginas del libro
TESIS		**Clase que hereda los atributos de la clase DOCUMENTO destinada para las Tesis**
	Director	Persona que dirige o asesora una Tesis.
REVISTA		**Clase que hereda los atributos de la clase DOCUMENTO destinada para revistas.**
	numero_publicacion	Registra el número de publicación de una revista
AUTOR		**Clase que administra la Información de los Autor/es del documento**
	Id	Identificador única del autor
	Nombre	Registra el o los nombre(s) del o los autores del documento
	Email	Registra el email del o los autores
	dirección_web	Registra la dirección Web de o los autores de un documento
	crearAutor	Método que permite registrar la información del Autor
	actualizarAutor	Método que permite actualizar la información del Autor
	buscarAutor	Método que permite buscar la información del Autor de un Documento
IDIOMA		**Clases que administra la información del Idioma en el que está un documento**
	Id	Identificador único del idioma
	nombre	Nombre del idioma.
	crearIdioma	Método que permite registrar la información del idioma de un documento
	actualizarIdioma	Método que permite actualizar la información del idioma de un

		documento
	buscarIdioma	Método que permite buscar la información de un determinado Idioma

ANEXO F. Código Fuente del modelo Objeto Relacional del Sistema SIABI

```
--Procedimiento que retorna la secuencia

CREATE OR REPLACE FUNCTION retorna_sec(vtipo_sec VARCHAR2) RETURN NUMBER IS vsec
NUMBER;
BEGIN
  EXECUTE IMMEDIATE 'select ' || vtipo_sec || '.nextval  from dual'
  INTO vsec;
  RETURN vsec;

EXCEPTION
WHEN others THEN
  raise_application_error(-20999,   'Error en Retorna_sec ' || vtipo_sec || '-' || sqlerrm);
END;
/

--creación de SECUENCIAS

CREATE sequence secidioma START WITH 100 increment BY 1 nocache nocycle;

CREATE sequence secautor START WITH 100 increment BY 1 nocache nocycle;

CREATE sequence secrev START WITH 1000 increment BY 1 nocache nocycle;

CREATE sequence seclib START WITH 1000 increment BY 1 nocache nocycle;

CREATE sequence sectes START WITH 1000 increment BY 1 nocache nocycle;

CREATE sequence secejerev START WITH 1 increment BY 1 nocache nocycle;

CREATE sequence secejelib START WITH 1 increment BY 1 nocache nocycle;

CREATE sequence secejetes START WITH 1 increment BY 1 nocache nocycle;

CREATE sequence secpre START WITH 1 increment BY 1 nocache nocycle;

CREATE sequence secmul START WITH 1000 increment BY 1 nocache nocycle;

CREATE sequence secres START WITH 1000 increment BY 1 nocache nocycle;

----------CREACION DE TIPOS INCOMPLETOS**********/

CREATE type prestamo;
/

CREATE type ejemplar;
/

CREATE type multa;
/

CREATE type reservacion;
/

CREATE type bibliotecario;
/

CREATE type usuarioconsulta;
/

CREATE type documento;
```

```
/

CREATE type autor;
/

CREATE type revista;
/

CREATE type libro;
/

CREATE type tesis;
/

CREATE type idioma;
/

CREATE type categoria;
/

CREATE type login;
/

------------------CREACION DE COLECCIONES************************/

CREATE type list_prestamo AS TABLE OF ref prestamo;
/

CREATE type list_reservacion AS TABLE OF ref reservacion;
/

CREATE type list_multa AS TABLE OF ref multa;
/

CREATE type list_ejemplar AS TABLE OF ref ejemplar;
/

CREATE type list_doc AS TABLE OF ref documento;
/

CREATE OR REPLACE type list_var_ejem AS TABLE OF VARCHAR2(7);
/



-------Objeto LOGIN***********************************************/

CREATE OR REPLACE type login AS object
(log_usuario VARCHAR2(20),
log_password VARCHAR2(10),
bib_login ref bibliotecario,
static FUNCTION crearlogin(vusuario VARCHAR2,    vpassword VARCHAR2,    vbiblio ref bibliotecario)
RETURN ref login,
static PROCEDURE actualizarlogin(vusuario VARCHAR2,    vpassword VARCHAR2,    vbiblioact ref
bibliotecario));
/

-------- Persistencia de Login

CREATE TABLE login_t OF login(PRIMARY KEY(log_usuario))
/

----paquete de login

CREATE OR REPLACE PACKAGE bibliotecariolog_p AS
  PROCEDURE crearlogin_p(vlogin login);
  PROCEDURE actualizarlog_p(vlogin login);
```

```
  PROCEDURE crearprestamobiblio(vidbiblio NUMBER,   vrefprestamo ref prestamo);
END bibliotecariolog_p;
/

----metodos de object login

CREATE OR REPLACE type BODY login AS static FUNCTION crearlogin(vusuario VARCHAR2,
vpassword VARCHAR2,   vbiblio ref bibliotecario) RETURN ref login IS vloginref ref login;
vlog login := new login(vusuario,   vpassword,   vbiblio);
BEGIN

bibliotecariolog_p.crearlogin_p(vlog);
  SELECT ref(vtemlog)
  INTO vloginref
  FROM login_t vtemlog
  WHERE vtemlog.bib_login = vbiblio;

EXCEPTION
WHEN others THEN
  raise_application_error(-20999,   'Error en crear login ' || vusuario || '-' || vpassword || '-' || sqlerrm);
END;
static PROCEDURE actualizarlogin(vusuario VARCHAR2,     vpassword VARCHAR2,     vbiblioact ref
bibliotecario) IS
vlog login := new login(vusuario,   vpassword,   vbiblioact);
BEGIN
  bibliotecariolog_p.actualizarlog_p(vlog);
END;
END;
/



---objeto MULTA**************************************/

CREATE OR REPLACE type multa AS object
(mul_id NUMBER,
mul_monto NUMBER(5,   2),
mul_motivo VARCHAR2(200),
mul_estado VARCHAR2(1),
pre_multa ref prestamo,
static FUNCTION calcularmulta(vrefprestamo ref prestamo,    vmul_monto NUMBER) RETURN ref multa,
static FUNCTION pagarmulta(vmotivo VARCHAR2,    vpagado NUMBER,    vrefmulta ref multa) RETURN
boolean,
member FUNCTION estadomulta RETURN VARCHAR2);
/

---Persistencia de Multa

CREATE TABLE multa_t OF multa(PRIMARY KEY(mul_id));

--Paquete de multa

CREATE OR REPLACE PACKAGE multa_p AS
  FUNCTION crearmulta_p(vrefprestamo ref prestamo,   vmul_monto NUMBER) RETURN ref multa;
  PROCEDURE pagarmulta_p(vmotivo VARCHAR2,   vpagado NUMBER,   vrefmulta ref multa);
  FUNCTION buscarmulta_p(vrefprestamo IN OUT ref prestamo,    vrefmulta IN OUT ref multa) RETURN
multa;

END multa_p;
/

--cuerpo del paquete Multa

CREATE OR REPLACE PACKAGE BODY multa_p AS
  FUNCTION crearmulta_p(vrefprestamo ref prestamo,     vmul_monto NUMBER) RETURN ref multa IS
vrefmulta ref multa;
  vsecmul NUMBER := retorna_sec('secmul');
```

```
  BEGIN
    INSERT
    INTO multa_t(mul_id,   mul_monto,   mul_estado,   pre_multa)
    VALUES(vsecmul,   vmul_monto,   'D',   vrefprestamo);

    SELECT ref(vmul)
    INTO vrefmulta
    FROM multa_t vmul
    WHERE vmul.mul_id = vsecmul;

    RETURN vrefmulta;
  END crearmulta_p;

  PROCEDURE pagarmulta_p(vmotivo VARCHAR2,   vpagado NUMBER,   vrefmulta ref multa) IS
  BEGIN

    UPDATE multa_t vmul
    SET vmul.mul_estado = 'P',
      vmul.mul_motivo = vmotivo
    WHERE ref(vmul) = vrefmulta;

  END pagarmulta_p;
  FUNCTION buscarmulta_p(vrefprestamo IN OUT ref prestamo,    vrefmulta IN OUT ref multa) RETURN
multa IS vmulta multa;
  BEGIN

    IF vrefprestamo IS NOT NULL THEN

      SELECT ref(vmul),
        VALUE(vmul)
      INTO vrefmulta,
        vmulta
      FROM multa_t vmul
      WHERE vmul.pre_multa = vrefprestamo;

      ELSIF vrefmulta IS NOT NULL THEN

        SELECT VALUE(vmul),
          vmul.pre_multa,
          ref(vmul)
        INTO vmulta,
          vrefprestamo,
          vrefmulta
        FROM multa_t vmul
        WHERE ref(vmul) = vrefmulta;

      END IF;

      RETURN vmulta;

    END buscarmulta_p;

END multa_p;
/

--- Metodos de Multa

CREATE OR REPLACE type BODY multa AS static FUNCTION calcularmulta(vrefprestamo ref prestamo,
vmul_monto NUMBER) RETURN ref multa IS
BEGIN

  RETURN multa_p.crearmulta_p(vrefprestamo,   vmul_monto);

END;

static FUNCTION pagarmulta(vmotivo VARCHAR2,   vpagado NUMBER,   vrefmulta ref multa) RETURN
boolean IS
```

```
BEGIN

multa_p.pagarmulta_p(vmotivo,   vpagado,   vrefmulta);
  RETURN TRUE;

END;

member FUNCTION estadomulta RETURN VARCHAR2 IS
BEGIN
  RETURN mul_estado;
END;
END;
/


----------Objeto PRESTAMO*********************************************/

CREATE OR REPLACE type prestamo AS object
(pre_id NUMBER,
pre_fecha DATE,
pre_devolucion DATE,
pre_caducidad DATE,
mul_prestamo ref multa,
res_prestamo ref reservacion,
static FUNCTION crearprestamo(vpre_fecha DATE,     vres_prestamo ref reservacion, vfecha_fin date)
RETURN ref prestamo,
static PROCEDURE devolverprestamo(vidprestamo NUMBER),
member FUNCTION verificarmulta(vidprestamo NUMBER) RETURN boolean,
static PROCEDURE renovarprestamo(vidprestamo NUMBER),
member FUNCTION verificarfecha(vidprestamo NUMBER) RETURN NUMBER,
static FUNCTION buscarprestamo(vidprestamo NUMBER) RETURN prestamo);
/

----------Persistencia de Prestamo

CREATE TABLE prestamo_t OF prestamo(PRIMARY KEY(pre_id));

--Crear paquete de prestamos

CREATE OR REPLACE PACKAGE prestamo_p AS
  FUNCTION crearprestamo_p(vprestamo prestamo) RETURN ref prestamo;
  PROCEDURE devolver_p(vidprestamo NUMBER);
END prestamo_p;
/

create or replace package  body prestamo_p as
function crearPrestamo_p (vprestamo prestamo) return ref  prestamo is
vrefprestamo ref prestamo;
begin
   insert  into  prestamo_t    (pre_id,  pre_fecha,pre_caducidad,res_prestamo)  values  (vprestamo.pre_id,
vprestamo.pre_fecha,vprestamo.pre_caducidad,vprestamo.res_prestamo);

   begin
   select ref(vpre)
   into vrefprestamo
   from prestamo_t vpre
   where vpre.pre_id=vprestamo.pre_id;

return vrefprestamo;
   exception when others then
   raise_application_error(-20999,'Error en crear préstamo :'||vprestamo.pre_id||'-'||sqlerrm);
end;

end crearPrestamo_p;

procedure devolver_p(vidprestamo number) is
begin
```

```
   update prestamo_t vpre
   set vpre.pre_devolucion= sysdate
   where vpre.pre_id=vidprestamo;

exception
when  others then
raise_application_error(-20999,'Error en devolución del libro '||sqlerrm);
end devolver_p;
end Prestamo_p;
/

--------Métodos de prestamo

CREATE OR REPLACE type BODY prestamo AS
static FUNCTION crearprestamo(vpre_fecha DATE,    vres_prestamo ref reservacion, vfecha_fin date)
RETURN ref prestamo IS
vprestamo                                                                                    prestamo:=NEW
prestamo(retorna_sec('secpre'),vpre_fecha,null,vfecha_fin,null,vres_prestamo);
vrefpre ref prestamo;
BEGIN
     vrefpre := prestamo_p.crearprestamo_p(vprestamo);
     RETURN vrefpre;
exception
when others then
raise_application_error(-20999,'Error Prestano.crearprestamo '||sqlerrm);
END;

static PROCEDURE devolverprestamo(vidprestamo NUMBER) IS

   BEGIN

    prestamo_p.devolver_p(vidprestamo);

   END;

   member FUNCTION verificarmulta(vidprestamo NUMBER) RETURN boolean IS vmulta multa;
   BEGIN

     IF mul_prestamo IS NOT NULL THEN
       BEGIN
         SELECT DEREF(vpre.mul_prestamo)
         INTO vmulta
         FROM prestamo_t vpre
         WHERE vpre.pre_id = vidprestamo;

         IF vmulta.estadomulta = 'P' THEN
           RETURN TRUE;
         ELSE
           RETURN FALSE;
         END IF;

       EXCEPTION
       WHEN others THEN
         raise_application_error(-20999,   'Error en verificar multa ' || sqlerrm);
       END;

       ELSIF TRUNC(pre_caducidad) >= TRUNC(sysdate) THEN
         RETURN TRUE;
       ELSE
         RETURN FALSE;
       END IF;

     END;

static PROCEDURE renovarprestamo(vidprestamo NUMBER) IS
     vcategoria NUMBER;
     BEGIN
```

```
      SELECT to_date(vpre.pre_caducidad,   'yyyy/mm/dd') -to_date(vpre.pre_fecha,   'yyyy/mm/dd') --ojo
revisar
      INTO vcategoria
      FROM prestamo_t vpre
      WHERE vpre.pre_id = vidprestamo;

      UPDATE prestamo_t vpre
      SET vpre.pre_fecha = TRUNC(sysdate),
        vpre.pre_caducidad = TRUNC(sysdate) + vcategoria
      WHERE vpre.pre_id = vidprestamo;

    EXCEPTION
    WHEN others THEN
      raise_application_error(-20999,   'Error en renovar prestamo ' || sqlerrm);
    END;

member FUNCTION verificarfecha(vidprestamo NUMBER) RETURN NUMBER IS vdias NUMBER;
    BEGIN
      SELECT to_date(vpre.pre_caducidad,   'yyyy/mm/dd') -to_date(vpre.pre_fecha,   'yyyy/mm/dd') --ojo
revisar
      INTO vdias
      FROM prestamo_t vpre
      WHERE vpre.pre_id = vidprestamo;
      RETURN vdias;

    EXCEPTION
    WHEN others THEN
      raise_application_error(-20999,   'Error en verificar fecha  ' || sqlerrm);
    END;

static FUNCTION buscarprestamo(vidprestamo NUMBER) RETURN prestamo IS vprestamo prestamo;
    BEGIN
      SELECT VALUE(vpre)
      INTO vprestamo
      FROM prestamo_t vpre
      WHERE vpre.pre_id = vidprestamo;
      RETURN vprestamo;

    EXCEPTION
    WHEN no_data_found THEN
      RETURN NULL;
    WHEN others THEN
      raise_application_error(-20999,   'Error en buscar prestamo ' || sqlerrm);
    END;
END;
/

---objeto EJEMPLAR************************************************/

CREATE OR REPLACE type ejemplar AS object(eje_id VARCHAR2(7),   --?? cambiar en todos los que se
haga referencia
eje_original VARCHAR2(1),
eje_estado_prestamo VARCHAR2(1),
eje_estado_ejemplar VARCHAR2(1),
pre_ejemplar list_prestamo,
static FUNCTION crearejemplar(vdoc_id NUMBER,   veje_original VARCHAR2,   veje_estado_prestamo
VARCHAR2,   veje_estado_ejemplar VARCHAR2,   vtipo VARCHAR2) RETURN ejemplar,   --member
function  crearEjemplar(vejemplar ejemplar) return  ejemplar,--por cuestion de secuencial
member PROCEDURE actualizarestadoprestamo(vestadopres VARCHAR2),
member FUNCTION verificarestadoprestamo(videjemplar NUMBER) RETURN VARCHAR2,
pragma restrict_references(verificarestadoprestamo,   wnds));
/

-- Coleccion para referenciar en Documento

CREATE type list_ejemp_doc AS TABLE OF ejemplar;
```

```
/

---- metodo de Ejemplar

CREATE OR REPLACE type BODY ejemplar as
static function  crearEjemplar(vdoc_id number, veje_original varchar2, veje_estado_prestamo varchar2, veje_estado_ejemplar varchar2,vtipo varchar2 ) return  ejemplar is
vejemplar ejemplar:=new ejemplar(null,veje_original,veje_estado_prestamo,veje_estado_ejemplar,null);
vsecuencia varchar2(12);
begin

  if vtipo = 'R' then
  vsecuencia:='secejerev';
  elsif vtipo = 'L' then
  vsecuencia:='secejelib';
  elsif vtipo = 'T' then
  vsecuencia:='secejetes';
  end if;
  save_log('paso 1');
  vejemplar.eje_id:= to_char(vdoc_id)||'-'||to_char(retorna_sec(vsecuencia));
  save_log('paso 2 '||vejemplar.eje_id);
  return vejemplar;
end;
member procedure actualizarEstadoPrestamo (vestadopres varchar2)  is
ejemplo actualizarRevista
begin
  eje_estado_prestamo:=vestadopres;
  end;
member function verificarEstadoprestamo(videjemplar number) return varchar2 is
begin
return eje_estado_ejemplar;
end;
end;
/

--paquete del ejemplar

CREATE OR REPLACE PACKAGE ejemplar_p AS
  FUNCTION  crearprestamoejemplar(viddocumento  NUMBER,  vtipo  VARCHAR2,  videjemplar VARCHAR2,  vrefprestamo ref prestamo) RETURN ejemplar;
  FUNCTION retornaidprestamo(videjemplar varchar2, vtipodoc varchar2) return number;
  END ejemplar_p;
/

create or replace package body ejemplar_p as
function crearPrestamoEjemplar(viddocumento number,vtipo varchar2,videjemplar varchar2,vrefprestamo ref prestamo) return ejemplar is
vcolejemplar list_ejemp_doc;
vtabla varchar2(12);
VCOUNT NUMBER;
vtemp number:=0;
begin

if vtipo = 'R' then
            vtabla:='revista_t';
elsif vtipo = 'T' then
            vtabla:='tesis_t';
elsif vtipo = 'L' then
            vtabla:='libro_t';
end if;

EXECUTE IMMEDIATE 'SELECT d.eje_documento FROM '||vtabla||' d
             WHERE d.doc_id = '||viddocumento  INTO vcolejemplar;

for vcount in 1..vcolejemplar.count loop
       if vcolejemplar(vcount).eje_id = videjemplar then
              vtemp:=vcount;
```

```
        end if;
end loop;

if vtemp <> 0 then
   vcolejemplar(vtemp).pre_ejemplar:= list_prestamo();
   vcolejemplar(vtemp).pre_ejemplar.extend;
   vcolejemplar(vtemp).pre_ejemplar(vcolejemplar(vtemp).pre_ejemplar.last):=vrefprestamo;

return vcolejemplar(vtemp);
else
return null; --no se encontró el ejemplar
end if;

end crearPrestamoEjemplar;

FUNCTION retornaidprestamo(videjemplar varchar2, vtipodoc varchar2) return number is
vtabla varchar2(11);
vcolejemplar LIST_EJEMP_DOC;
vdocid number;
vprestamo prestamo;
vrefprestamo ref prestamo;

begin

vdocid:=to_number(substr(videjemplar,1,instr(videjemplar,'-',1)-1));
if vtipodoc = 'R' then
            vtabla:='revista_t';
elsif vtipodoc = 'T' then
            vtabla:='tesis_t';
elsif vtipodoc = 'L' then
            vtabla:='libro_t';
end if;

EXECUTE IMMEDIATE 'SELECT d.eje_documento FROM '||vtabla||' d, table(d.eje_documento) e
             WHERE d.doc_id = '||vdocid  INTO vcolejemplar;

for c1 in 1..vcolejemplar.count loop
  if vcolejemplar(c1).eje_id=videjemplar then
    for c2 in 1..vcolejemplar(c1).PRE_EJEMPLAR.count loop
      vrefprestamo:=vcolejemplar(c1).PRE_EJEMPLAR(c2);
      vprestamo:=deref(vrefprestamo);
      if vprestamo.PRE_DEVOLUCION is null then
        return vprestamo.PRE_ID;
      end if;
    end loop;
  end if;
end loop;

return null;
end retornaidprestamo;
end ejemplar_p;
/

--------Objeto BIBLIOTECARIO*******************************************/

CREATE OR REPLACE type bibliotecario AS object
(bib_id VARCHAR2(10),   --number(10),   --   (ojo)cedula
bib_nombre VARCHAR2(100),
bib_estado VARCHAR2(1),
bib_ingreso DATE,
bib_salida DATE,
bib_tipo VARCHAR2(1),
pre_bibliotecario list_prestamo,
log_bibliotecario ref login,
static FUNCTION crearbibliotecario(vidbibliotecario VARCHAR2,   vnombre VARCHAR2,   vingreso DATE,
vbib_tipo varchar2) RETURN ref bibliotecario,
```

```
member PROCEDURE actualizarbibliotecario(vbibliotecario bibliotecario, vid_original VARCHAR2, vprestamo ref prestamo,vlogin ref login),
static FUNCTION buscarbibliotecario(vidbiblio IN OUT VARCHAR2,  vrefbiblio IN OUT ref bibliotecario) RETURN bibliotecario);
/

-------Persistencia de Bibliotecario

CREATE TABLE bibliotecario_t OF bibliotecario --(ojo)
(PRIMARY KEY(bib_id)) nested TABLE pre_bibliotecario store AS bibliotecario_prestamo_t;



----------paquete de bibliotecario
create or replace package bibliotecario_p as
function crearBibliotecario_p(vidbibliotecario VARCHAR2,vnombre VARCHAR2,vingreso DATE,vbib_tipo varchar2) rETURN ref bibliotecario;
procedure actualizaBiblio_p(vbibliotecario bibliotecario, vid_original VARCHAR2, vprestamo ref prestamo,vlogin ref login);
end bibliotecario_p;
/


create or replace package body bibliotecario_p as
function crearBibliotecario_p(vidbibliotecario VARCHAR2,vnombre VARCHAR2,vingreso DATE,vbib_tipo varchar2) rETURN ref bibliotecario is
vrefbiblio ref bibliotecario;
begin
INSERT
  INTO bibliotecario_t(bib_id,  bib_nombre,  bib_estado,  bib_ingreso,bib_tipo)
  VALUES(vidbibliotecario,  vnombre,  'H', sysdate,vbib_tipo);
  SELECT ref(b)
  INTO vrefbiblio
  FROM bibliotecario_t b
  WHERE b.bib_id = vidbibliotecario;
  RETURN vrefbiblio;
exception
when others then
raise_application_error(-20999,'crearBibliotecario_p '||vidbibliotecario||'-'||sqlerrm);
end crearBibliotecario_p;

procedure actualizaBiblio_p(vbibliotecario bibliotecario, vid_original varchar2, vprestamo ref prestamo,vlogin ref login)is
vcolprestamo list_prestamo;
begin

if vid_original is not null and  vbibliotecario is not null then
UPDATE bibliotecario_t
  SET bib_id = vbibliotecario.bib_id,
    bib_nombre = vbibliotecario.bib_nombre,
    bib_estado = vbibliotecario.bib_estado,
    bib_salida = vbibliotecario.bib_salida
  WHERE bib_nombre = vid_original;
end if;


if vprestamo is not null then
 select b.pre_bibliotecario
  into vcolprestamo
  from bibliotecario_t b
  where b.bib_id = vid_original;
  if vcolprestamo is null then
    vcolprestamo:=list_prestamo();
  end if;
  vcolprestamo.extend;
  vcolprestamo(vcolprestamo.last):=vprestamo;
   update bibliotecario_t b
```

```
  set b.pre_bibliotecario = null
  where b.bib_id = vid_original;

  update bibliotecario_t b
  set b.pre_bibliotecario = vcolprestamo
  where b.bib_id = vid_original;
elsif vlogin is not null then
   update bibliotecario_t b
  set b.log_bibliotecario = vlogin
  where b.bib_id = vid_original;
end if;

exception
when others then
raise_application_error(-20999,'Error actualizaBiblio_p '||sqlerrm);
end actualizaBiblio_p;
end bibliotecario_p;
/

--metodos del objeto biblioteario


CREATE OR REPLACE type BODY bibliotecario AS
static FUNCTION crearbibliotecario(vidbibliotecario VARCHAR2,   vnombre VARCHAR2,   vingreso DATE,
vbib_tipo varchar2) RETURN ref bibliotecario is
vrefbiblio ref bibliotecario;
BEGIN
vrefbiblio:=bibliotecario_p.crearBibliotecario_p(vidbibliotecario,vnombre,vingreso,vbib_tipo);
rETURN vrefbiblio;

EXCEPTION
WHEN others THEN
  raise_application_error(-20999,   'Error no se pudo crear el bibliotecario ' || sqlerrm);
END;

member PROCEDURE actualizarbibliotecario(vbibliotecario bibliotecario, vid_original varchar2, vprestamo
ref prestamo,vlogin ref login) is

BEGIN
bibliotecario_p.actualizaBiblio_p(vbibliotecario, vid_original , vprestamo,vlogin);

END;
static FUNCTION buscarbibliotecario(vidbiblio IN OUT VARCHAR2,    vrefbiblio IN OUT ref bibliotecario)
RETURN bibliotecario IS
vbibliotecario bibliotecario;
BEGIN
  IF vidbiblio IS NOT NULL THEN
    SELECT VALUE(b),
      ref(b)
    INTO vbibliotecario,
      vrefbiblio
    FROM bibliotecario_t b
    WHERE bib_id = vidbiblio;

    ELSIF vrefbiblio IS NOT NULL THEN

      SELECT VALUE(b),
        b.bib_id
      INTO vbibliotecario,
        vidbiblio
      FROM bibliotecario_t b
      WHERE bib_id = vidbiblio;

    END IF;

    RETURN vbibliotecario;
```

```
  EXCEPTION
  WHEN others THEN
    raise_application_error(-20999,   'Error no se pudo buscar el bibliotecario' || sqlerrm);
  END;
END;
/



----------paquete  bibliotecario

create or replace package body bibliotecariolog_p as
procedure crearLogin_p( vlogin Login) is
begin
  insert into login_t (log_usuario, log_password,bib_login) values
(vlogin.log_usuario,vlogin.log_password,vlogin.bib_login) ;
exception when others then
raise_application_error(-20999,'Error inserción de login '||sqlerrm);
end crearLogin_p;
procedure actualizarLog_p( vlogin Login) is
begin
  update login_t
  set log_usuario = vlogin.log_usuario, log_password = vlogin.log_password
  where bib_login=vlogin.bib_login;
exception when others then
raise_application_error(20999,'Error actualización de login '||sqlerrm);
end actualizarLog_p;
procedure actrefbiblio(vbiblioref ref Bibliotecario,vloginref ref Login) is
begin
update bibliotecario_t vbiblio
set vbiblio.log_bibliotecario = vloginref
where ref(vbiblio)=vbiblioref;
end actrefbiblio;

procedure crearPrestamoBiblio(vidbiblio number, vrefprestamo ref prestamo) is
vcolprestamo list_prestamo;
vcount number;
begin
select vbiblio.pre_bibliotecario
into vcolprestamo
from bibliotecario_t vbiblio
where vbiblio.bib_id=vidbiblio;

vcolprestamo.Extend;
vcolprestamo( vcolprestamo.last) := vrefprestamo;
update bibliotecario_t b
set b.pre_bibliotecario = vcolprestamo;

end crearPrestamoBiblio;
End bibliotecariolog_p;
/



----Objeto RESERVACION*****************************************/

CREATE OR REPLACE type reservacion AS object
(reser_id NUMBER,
reser_fecha DATE,
pre_reservacion ref prestamo,
eje_reservacion list_var_ejem,
static function crearreservacion(vejemplar varchar2)return ref reservacion,
static function consultarreservacion( vejemplar varchar2, vreservacion in out ref reservacion, vidres in out
number,vnumejemplar out number) RETURN boolean,
```

```
static procedure crearReservaEjemplar(vidreserva number, vejemplar varchar2));
/

---Persistencia de Reservación

CREATE TABLE reservacion_t OF reservacion(PRIMARY KEY(reser_id)) nested TABLE eje_reservacion
store AS reservacion_ejemplar_t
/

-----------------------paquete de reservar

CREATE OR REPLACE PACKAGE reservacion_p AS
  FUNCTION crearreservacion_p(videjemplar VARCHAR2) RETURN ref reservacion;
END reservacion_p;
/

CREATE OR REPLACE PACKAGE BODY reservacion_p AS
  FUNCTION crearreservacion_p(videjemplar VARCHAR2) RETURN ref reservacion IS vidres NUMBER;
  vrefreser ref reservacion;
  vlist_var_ejem list_var_ejem;
  BEGIN
    vlist_var_ejem := list_var_ejem();
    vidres := retorna_sec('secres');
    vlist_var_ejem.extend;
    vlist_var_ejem(vlist_var_ejem.LAST) := videjemplar;

    INSERT
    INTO reservacion_t(reser_id,   reser_fecha,   eje_reservacion)
    VALUES(vidres,   sysdate,   vlist_var_ejem);

    SELECT ref(vres)
    INTO vrefreser
    FROM reservacion_t vres
    WHERE vres.reser_id = vidres;

    RETURN vrefreser;

  EXCEPTION
  WHEN others THEN
    raise_application_error(-20999,   'Error crearReservacion_p ' || sqlerrm);
  END crearreservacion_p;

END reservacion_p;
/

--------  Metodos de Reservacion

CREATE OR REPLACE type BODY reservacion AS
static FUNCTION crearreservacion(vejemplar varchar2) RETURN ref reservacion IS
BEGIN
  RETURN reservacion_p.crearReservacion_p(vejemplar);
END;

static function consultarreservacion( vejemplar varchar2, vreservacion in out ref reservacion, vidres in out
number,vnumejemplar out number) RETURN boolean IS
usuario consulta
  vcolejemplar list_var_ejem;
BEGIN
vnumejemplar:=0;
begin
  select r.EJE_RESERVACION,r.RESER_ID
  into vcolejemplar,vidres
  from reservacion_t r
  where ref(r)= vreservacion
  and trunc(r.RESER_FECHA) = trunc(sysdate);
  exception
  when no_data_found then
```

```
  return true;
  when others then
  raise_application_error(-20999,'Error en buscar Reservación. Revise');
  end;

  for vcount in 1..vcolejemplar.count loop
       vnumejemplar:=vnumejemplar+1;
       if vcolejemplar(vcount) = vejemplar then
         RETURN FALSE;
       end if;

  end loop;

RETURN TRUE;
END;

static procedure crearReservaEjemplar(vidreserva number, vejemplar varchar2) IS
vcolejemplar list_var_ejem;

BEGIN

  select r.EJE_RESERVACION
  into vcolejemplar
  from reservacion_t r
  where r.reser_id=vidreserva;

  vcolejemplar.extend;
  vcolejemplar(vcolejemplar.last):=vejemplar;

  update reservacion_t res
  set res.EJE_RESERVACION = vcolejemplar
  where res.RESER_ID=vidreserva;

exception
when others then
raise_application_error(-20999,'Error en crearReservaEjemplar '||sqlerrm);
  END;
END;
/

---Objeto USUARIOCONSULTA************************************/

CREATE OR REPLACE type usuarioconsulta AS object
(usu_id VARCHAR2(10),
usu_nom VARCHAR2(100),
usu_dir VARCHAR2(200),
usu_telefono VARCHAR2(15),
pre_usuario list_prestamo,
res_usuario list_reservacion,
mul_usuario list_multa,
static FUNCTION crearusuario(vidusuario VARCHAR2,  vnombre VARCHAR2,  vdireccion VARCHAR2,
vtelefono VARCHAR2) RETURN ref usuarioconsulta,
member PROCEDURE actualizarusuario(vusu_id VARCHAR2,  vusu_idorg VARCHAR2,  vusu_nom
VARCHAR2,  vusu_dir VARCHAR2,  vusu_telefono VARCHAR2,  vpre_usuario ref prestamo,
vres_usuario ref reservacion,  vmul_usuario ref multa),
static FUNCTION buscarusuario(vusu_id IN OUT VARCHAR2,  vrefusuario IN OUT ref usuarioconsulta)
RETURN usuarioconsulta);
/

-- Persistencia de Usuarioconsulta

CREATE TABLE usuarioconsulta_t OF usuarioconsulta(PRIMARY KEY(usu_id)) nested TABLE
pre_usuario store AS usuario_prestamo_t,
  nested TABLE res_usuario store AS usuario_reservacion_t,
  nested TABLE mul_usuario store AS usuario_multa_t;

--paquete de bibliotecario
```

```
CREATE OR REPLACE PACKAGE usuarioconsulta_p AS
  FUNCTION crearusuario_p(vidusuario VARCHAR2,  vnombre VARCHAR2,  vdireccion VARCHAR2,
vtelefono VARCHAR2) RETURN ref usuarioconsulta;
  PROCEDURE  actualizarusuario_p(vusu_id  VARCHAR2,  vusu_idorg  VARCHAR2,  vusu_nom
VARCHAR2,  vusu_dir  VARCHAR2,  vusu_telefono  VARCHAR2,  vpre_usuario  ref  prestamo,
vres_usuario ref reservacion,  vmul_usuario ref multa);
  FUNCTION verificareservacion_p(vnumusuario VARCHAR2,  vejemplar VARCHAR2,  vnumejemplar
OUT NUMBER,  vidreser OUT NUMBER) RETURN boolean;
END usuarioconsulta_p;
/

create or replace package body usuarioconsulta_p as
function  crearUsuario_p(vidusuario  VARCHAR2,  vnombre  VARCHAR2,  vdireccion  VARCHAR2,
vtelefono VARCHAR2) return ref usuarioconsulta
is
vrefusuario ref usuarioconsulta;
begin
  INSERT
  INTO usuarioconsulta_t(usu_id,  usu_nom,  usu_dir,  usu_telefono)
  VALUES(vidusuario,  vnombre,  vdireccion,  vtelefono);

  select ref(vusu)
  into vrefusuario
  from usuarioconsulta_t vusu
  where vusu.usu_id=vidusuario;

return vrefusuario;
end crearUsuario_p;
procedure actualizarusuario_p(vusu_id VARCHAR2,vusu_idorg varchar2,vusu_nom VARCHAR2,vusu_dir
VARCHAR2,vusu_telefono  VARCHAR2,vpre_usuario  ref  prestamo,vres_usuario  ref
reservacion,vmul_usuario ref multa)
is
vcolprestamo list_prestamo;
vcolreserva list_reservacion;
vcolmulta  list_multa;
begin
if vusu_idorg is not null then
 UPDATE usuarioconsulta_t
  SET usu_id = vusu_id,
    usu_nom = vusu_nom,
    usu_dir = vusu_dir,
    usu_telefono = vusu_telefono
  WHERE usu_id = vusu_idorg;
elsif vpre_usuario is not null then
  select u.pre_usuario
  into vcolprestamo
  from usuarioconsulta_t u
  where u.usu_id = vusu_id;
  if vcolprestamo is null then
    vcolprestamo:=list_prestamo();
  end if;
  vcolprestamo.extend;
  vcolprestamo(vcolprestamo.last):=vpre_usuario ;
  update usuarioconsulta_t b
  set b.pre_usuario = vcolprestamo
  where b.usu_id = vusu_id;
elsif vres_usuario is not null then
  select u.res_usuario
  into vcolreserva
  from usuarioconsulta_t u
  where u.usu_id = vusu_id;
  if vcolreserva is null then
    vcolreserva:=list_reservacion();
  end if;
  vcolreserva.extend;
  vcolreserva(vcolreserva.last):=vres_usuario;
```

```
  update usuarioconsulta_t b
  set b.res_usuario = vcolreserva
  where b.usu_id = vusu_id;
elsif vmul_usuario is not null then
  select u.mul_usuario
  into vcolmulta
  from usuarioconsulta_t u
  where u.usu_id = vusu_id;
  if vcolmulta is null then
    vcolmulta:=list_multa();
  end if;
  vcolmulta.extend;
  vcolmulta (vcolmulta .last):=vmul_usuario;
  update usuarioconsulta_t b
  set b.mul_usuario = vcolmulta
  where b.usu_id = vusu_id;
end if;
exception
when others then
raise_application_error(-20008,'Error en actualizarusuario_p '||sqlerrm);
end actualizarusuario_p;

function verificaReservacion_p(vnumusuario varchar2, vejemplar varchar2,vnumejemplar out
number,vidreser out number) return boolean is
vcolreservacion list_reservacion;
vrefreservacion ref reservacion:=null;
vtemp number:=0;
cursor creserva is select EJE_RESERVACION from reservacion_t where trunc(RESER_FECHA)
=trunc(sysdate);
begin

select u.RES_USUARIO
into vcolreservacion
from usuarioconsulta_t u
where u.USU_ID=vnumusuario;

if  vcolreservacion is null then
vcolreservacion:=list_reservacion();
save_log('vcolreservacion is null ');
end if;

for c1 in 1..vcolreservacion.count loop
  vrefreservacion:=vcolreservacion(c1);
  if reservacion.consultarreservacion(vejemplar, vrefreservacion, vidreser,vnumejemplar) = true then
  vtemp:=1;
  else
   return false;
  end if;
end loop;
if vrefreservacion is null then
  vnumejemplar:=0;
else
begin
for vcount in creserva loop

  for c1 in 1..vcount.EJE_RESERVACION.count loop
    if vcount.EJE_RESERVACION(c1)= vejemplar then
      return false;
    end if;
  end loop;
end if;
end loop;
end;
return true;
exception
when others then
raise_application_error(-20999,'Error en verificaReservacion_p '||sqlerrm);
```

```
end verificaReservacion_p;
end usuarioconsulta_p;
/

---- METODOS DE USUARIOCONSULTA

CREATE OR REPLACE type BODY usuarioconsulta AS static FUNCTION crearusuario(vidusuario VARCHAR2, vnombre VARCHAR2, vdireccion VARCHAR2, vtelefono VARCHAR2) RETURN ref usuarioconsulta IS
BEGIN
  RETURN usuarioconsulta_p.crearusuario_p(vidusuario, vnombre, vdireccion, vtelefono);

EXCEPTION
WHEN others THEN
  raise_application_error(-20999, 'Error en crear usuario ' || vidusuario || '-' || vnombre || '-' || sqlerrm);
END;

member PROCEDURE actualizarusuario(vusu_id VARCHAR2, vusu_idorg VARCHAR2, vusu_nom VARCHAR2, vusu_dir VARCHAR2, vusu_telefono VARCHAR2, vpre_usuario ref prestamo, vres_usuario ref reservacion, vmul_usuario ref multa) IS

BEGIN
  usuarioconsulta_p.actualizarusuario_p(vusu_id, vusu_idorg, vusu_nom, vusu_dir, vusu_telefono, vpre_usuario, vres_usuario, vmul_usuario);

EXCEPTION
WHEN others THEN
  raise_application_error(-20999, 'Error en actualizar usuario ' || vusu_id || '-' || vusu_nom || '-' || sqlerrm);
END;

static FUNCTION buscarusuario(vusu_id IN OUT VARCHAR2, vrefusuario IN OUT ref usuarioconsulta) RETURN usuarioconsulta
IS vusuario usuarioconsulta;
BEGIN

  IF vusu_id IS NOT NULL THEN
    SELECT VALUE(vusu),
      ref(vusu)
    INTO vusuario,
      vrefusuario
    FROM usuarioconsulta_t vusu
    WHERE vusu.usu_id = vusu_id;
    ELSIF vrefusuario IS NOT NULL THEN
      SELECT VALUE(vusu),
        vusu.usu_id
      INTO vusuario,
        vusu_id
      FROM usuarioconsulta_t vusu
      WHERE ref(vusu) = vrefusuario;
    END IF;

    RETURN vusuario;
  END;
END;
/

----------CATEGORIA**********************************/

CREATE OR REPLACE type categoria AS object
(cat_id NUMBER,
cat_valor NUMBER(4,2),
cat_dias NUMBER,
cat_nombre VARCHAR2(20),
--cat_estado VARCHAR2(1),
cat_ejemplar list_var_ejem,
static PROCEDURE crearcategoria,
member PROCEDURE actualizarcategoria,
```

```
static PROCEDURE buscarcategoria);
/

CREATE TABLE categoria_t OF categoria(PRIMARY KEY(cat_id)) nested TABLE cat_ejemplar store AS categoria_ejemplar_t;

--- Metodos de Categoria

CREATE OR REPLACE BODY categoria AS member PROCEDURE crearcategoria(vvalor NUMBER, vdias NUMBER,   vnombre VARCHAR2,   vestado VARCHAR2) IS
BEGIN
  INSERT
  INTO categoria_t(cat_valor,   cat_dias,   cat_nombre,   cat_estado)
  VALUES(vvalor,   vdias,   vnombre,   vestado);

END;

member PROCEDURE actualizarcategoria(vvalor NUMBER,    vdias NUMBER,    vnombre VARCHAR2, vestado VARCHAR2,   vnombreoriginal) IS

BEGIN

  UPDATE categoria_t
  SET cat_valor := vvalor,
    cat_dias := vdias,
    cat_nombre := vnombre,
    cat_estsdo := vestado
  WHERE cat_nombre = vnombreoriginal
   AND cat_estado = 'H';
END;
member PROCEDURE buscarcategoria(vnombre VARCHAR2) IS
BEGIN
  SELECT cat_valor,
    cat_dias,
    cat_nombre
  FROM categoria_t
  WHERE cat_nombre = vnombre
   AND cat_estado = 'H';
END;
END;

----------  AUTOR   ********************************/

CREATE OR REPLACE type autor AS object
(aut_id NUMBER,
aut_nombre VARCHAR2(40),
aut_email VARCHAR2(40),
aut_direccionweb VARCHAR2(80),   --doc_autor list_doc,
static FUNCTION crearautor(vautnom VARCHAR2,    vautmail VARCHAR2,    vautdirweb VARCHAR2) RETURN ref autor,
member PROCEDURE actualizarautor(vautoract autor),
static FUNCTION buscarautor(vidautor in out number, vautoref IN OUT ref autor) RETURN VARCHAR2);
/


---colección de autor
CREATE type list_autor AS TABLE OF autor;
/

--- Persistencia de Autor

CREATE TABLE autor_t OF autor(PRIMARY KEY(aut_id));


--- paquete administrar Autor

create or replace package Autor_p as
```

```
procedure crearAutor_p(vpautnom varchar2, vpautmail varchar2,  vpautdirweb varchar2);
procedure actualizarAutor_p(vpautor autor );
function retornaAutor_p(vnombre varchar2) return list_autor; --??vnombre es en el caso de restringir la búsqueda a una cadena que se pónga
End Autor_p;
/

create or replace package body Autor_p as
procedure crearAutor_p(vpautnom varchar2, vpautmail varchar2,  vpautdirweb varchar2) is
begin
   insert into autor_t  values (autor(secautor.nextval ,vpautnom,vpautmail,vpautdirweb));
exception when others then
   raise_application_error(-20999,'Error inserción de Autor  '||sqlerrm);
end crearAutor_p;

procedure actualizarAutor_p(vpautor autor) is
begin
  update autor_t
  set aut_nombre = vpautor.aut_nombre,
  aut_email=vpautor.aut_email ,
  aut_direccionWeb=vpautor.aut_direccionWeb
  where aut_nombre = vpautor.aut_nombre;
end actualizarAutor_p;

function retornaAutor_p(vnombre varchar2) return list_autor is
vcol_autor list_autor:=null;
begin
if vnombre is null  then
begin
   select value(vaut)
   bulk collect
   into vcol_autor
   from autor_t vaut
   order by vaut.aut_nombre desc;
   return vcol_autor;
exception
when no_data_found then
return null;
when others then
raise_application_error(-20011,'Error en retornaAutor_p i '||sqlerrm);
end;

else
begin
   select value(vaut)
   bulk collect
   into vcol_autor
   from autor_t vaut
   where upper(vaut.aut_nombre) like upper('%'||vnombre||'%')
   order by vaut.aut_nombre desc;
   return vcol_autor;
exception
when no_data_found then
return null;
when others then
raise_application_error(-20011,'Error en retornaAutor_p e'||sqlerrm);
end;
end if;


end retornaAutor_p;
End Autor_p;
/

--- metodos de Autor

create or replace type body autor as
```

```
static function crearAutor (vautnom varchar2, vautmail varchar2,  vautdirweb varchar2) return ref autor is
vautref ref autor;
  begin
    autor_p.crearAutor_p(vautnom , vautmail ,  vautdirweb);
    select ref(vaut)
    into vautref
    from autor_t vaut
    where vaut.aut_nombre  = vautnom;
     return vautref;
  end;
member procedure actualizarAutor (vautoract autor) is
  begin
    autor_p.actualizarAutor_p(vautoract);

  end;

static FUNCTION buscarautor(vidautor in out number, vautoref IN OUT ref autor) RETURN VARCHAR2 is
vautnom varchar2(40);
  begin


    if vautoref is not null then
            select  vautb.aut_nombre,vautb.AUT_ID
        into vautnom,vidautor
            from autor_t vautb
        where ref( vautb)=vautoref;
  else
       select  ref(vautb),vautb.aut_nombre
       into vautoref,vautnom
       from autor_t vautb
       where vautb.AUT_ID=vidautor;
   end if;
   return vautnom;
  exception when no_data_found then
  return null;
  end;
end;
/

------------------DOCUMENTO ****************************/
create or replace type documento as object
(
doc_id number(6),
doc_titulo varchar2(30),
doc_fecha_ingreso  date,
--doc_estado varchar2(1),
doc_contenido varchar2 (4000),
doc_año number,
idi_documento ref Idioma, --añadido para implementación
aut_documento ref Autor,--añadido para implementación
eje_documento list_ejemp_doc,
static function crearDocumento(vdoc_titulo varchar2,vdoc_fecha_ingreso date,vdoc_contenido
varchar2,vdoc_año number,vidi_documento ref Idioma,vaut_documento ref Autor) return  documento,
member function actualizarDocumento(vdoc documento) return documento,
member function visualizarContenido (vdoc_id number)  return varchar2
)not final;
/


----metodos de Documento

create or replace type body documento as
static function crearDocumento(vdoc_titulo varchar2,vdoc_fecha_ingreso date,vdoc_contenido
varchar2,vdoc_año number,vidi_documento ref Idioma,vaut_documento ref Autor)
return  documento is
```

```
vdoc Documento:= new documento(null,vdoc_titulo,vdoc_fecha_ingreso,vdoc_contenido,vdoc_año,vidi_documento,vaut_documento,null);
begin

  return vdoc;
end;
member function actualizarDocumento(vdoc documento) return documento is

begin
  return vdoc;
end;

member function visualizarContenido (vdoc_id number)  return varchar2 is

begin
  return doc_contenido;
end;
end;
/


----------IDIOMA  **********/

CREATE OR REPLACE type idioma AS object
(idi_id NUMBER(10),
idi_nombre VARCHAR2(50),   --doc_idioma list_doc,
static FUNCTION crearidioma(vidiomanom VARCHAR2) RETURN ref idioma,
member PROCEDURE actualizaridioma(vidiomanv VARCHAR2,   vidiomaoriginal VARCHAR2),
static FUNCTION buscaridioma(vidiomaref IN OUT ref idioma,    vididioma in out number) RETURN VARCHAR2);
/

--colección de idioma

create type list_idioma as table of idioma;
/

-- Persistencia de Idioma

CREATE TABLE idioma_t OF idioma(PRIMARY KEY(idi_id));

-----paquete de administrar idioma

create or replace package Idioma_p as
procedure crearIdioma_p(vidiomanom varchar2);
procedure actualizarIdioma_p(vidiomanv varchar2,vidiomaoriginal varchar2);
function retornaIdioma_p(vnombre varchar2) return list_idioma;
End Idioma_p;
/

create or replace package body Idioma_p as
procedure crearIdioma_p(vidiomanom varchar2) is
begin
  insert into idioma_t (idi_id,idi_nombre) values (secidioma.nextval ,vidiomanom );
exception when others then
raise_application_error(-20999,'Error inserción de Idioma '||sqlerrm);
end crearIdioma_p;

procedure actualizarIdioma_p(vidiomanv varchar2,vidiomaoriginal varchar2) is
begin
  update idioma_t
  set idi_nombre = vidiomaoriginal
  where idi_nombre = vidiomanv ;
end actualizarIdioma_p;

function retornaIdioma_p(vnombre varchar2) return list_idioma is
```

```
vcol_idioma list_idioma:=null;
begin
if vnombre is null  then--debe retornar todos
begin
    select value(idi)
    bulk collect
    into vcol_idioma
    from idioma_t idi
    order by idi.idi_nombre desc;
    return vcol_idioma;
exception
when no_data_found then
return null;
when others then
raise_application_error(-20011,'Error en retornaIdioma_p i '||sqlerrm);
end;

else
begin
    select value(idi)
    bulk collect
    into vcol_idioma
    from idioma_t idi
    where upper(idi.idi_nombre) like upper('%'||vnombre||'%')
    order by idi.idi_nombre desc;
    return vcol_idioma;
exception
when no_data_found then
return null;
when others then
raise_application_error(-20011,'Error en retornaidioma_p e'||sqlerrm);
end;
end if;


end retornaIdioma_p;
End Idioma_p;
/

---- Metodos de idioma

create or replace type body idioma as
static function crearIdioma (vidiomanom varchar2) return ref idioma is
vidiomaref ref idioma;
  begin
    Idioma_p.crearIdioma_p(vidiomanom);
    select ref(vidref)
    into vidiomaref
    from idioma_t vidref
    where vidref.idi_nombre  = vidiomanom;
     return vidiomaref;
  end;

member procedure actualizarIdioma (vidiomanv varchar2,vidiomaoriginal varchar2) is
  begin
    idi_nombre:=vidiomanv;
    Idioma_p.actualizarIdioma_p(vidiomanv ,vidiomaoriginal);
  end;

static FUNCTION buscaridioma(vidiomaref IN OUT ref idioma,     vididioma in out number) RETURN
VARCHAR2 is
vnombre varchar2(50):=null;

  begin
  if    vidiomaref is not null then
            select vid.idi_nombre,vid.IDI_ID
        into vnombre,vididioma
```

```
            from idioma_t vid
            where ref(vid)=vidiomaref;
  else
       select ref(vid),vid.idi_nombre
        into vidiomaref,vnombre
            from idioma_t vid
            where vid.IDI_ID=vididioma;
   end if;
  return vnombre;
  exception when no_data_found then
  return null;
  end;
 end;
/

----tipo REVISTA**********************************************/

create or replace type revista under documento
(
numero_publicacion number(4),
static function crearRevista(vdoc documento, vnumpublica number,vejemplar ejemplar) return ref revista,
member procedure actualizarRevista(vrev revista, vactejemplar varchar2,vcolrev list_ejemp_doc ,veje
ejemplar),
static function buscarRevista(vrevista out varchar2,vrefrevista in out ref revista, vidrevista in out  number,
vcolejemplar list_ejemp_doc, vejemplar out ejemplar, videjemplar varchar2) return  revista,
member function visualizarContenido (vrefrevista ref revista, vidrevista number)  return varchar2,
static procedure actualizarRevistaEjm(vidrevista number, videjemplar varchar2, vejemplar ejemplar)--,
vcolejemplar list_ejemp_doc)
);
/

--coleccion de revistas

CREATE OR REPLACE type list_revista IS TABLE OF revista;
/

------persistencia revista--------------------

CREATE TABLE revista_t OF revista(PRIMARY KEY(doc_id)) nested TABLE eje_documento store AS
revista_ejemplar_t((PRIMARY KEY(eje_id)) nested TABLE pre_ejemplar store AS rev_ejem_prestamo_t);

--paquete de Revista

CREATE OR REPLACE PACKAGE revista_p AS
  FUNCTION crearrevista_p(vdoc documento,   vnumpublica NUMBER, vejemplar ejemplar) RETURN ref
revista;
  PROCEDURE actualizarrevista_p(vrev revista,   vactejemplar VARCHAR2,   vcolrev list_ejemp_doc,
veje ejemplar);
  FUNCTION buscarcontenido(vrevista IN VARCHAR2) RETURN list_revista;
  function buscarContenidoL(vrevista in varchar2) return  list_revista;
  FUNCTION retornaejemplar_p(vrevid NUMBER,   vejeestadopre OUT VARCHAR2,   vejeestadoeje OUT
VARCHAR2,   vcount NUMBER, vejeid varchar2) RETURN VARCHAR2;
END revista_p;
/

create or replace package body revista_p as
function crearRevista_p(vdoc documento, vnumpublica number,vejemplar ejemplar) return ref revista is
vrevista ref revista;
vcoleje  list_ejemp_doc;
begin
  insert   into   revista_t   (doc_id,   doc_titulo,doc_fecha_ingreso,   doc_contenido,   doc_año,
idi_documento,aut_documento,eje_documento,numero_publicacion)
  values(vdoc.doc_id,  vdoc.doc_titulo,  vdoc.doc_fecha_ingreso,vdoc.  doc_contenido,  vdoc.doc_año,
vdoc.idi_documento,vdoc.aut_documento,null,vnumpublica);
  begin
    select ref(vrev),vrev.eje_documento
    into vrevista, vcoleje
```

```
    from revista_t vrev
    where vrev.doc_id = vdoc.doc_id;
  exception when others then
    raise_application_error(-20999,'Error en crear revista '||vdoc.doc_id||'- '||sqlerrm);
  end;
  begin
       if vcoleje is null then
         vcoleje:=list_ejemp_doc();
       end if;
       vcoleje.extend;
       vcoleje(vcoleje.last):=vejemplar;
       update revista_t r
       set   r.eje_documento = vcoleje
       where r.doc_id =vdoc.doc_id ;
  end;
  return vrevista;
end crearRevista_p;

procedure actualizarRevista_p(vrev revista, vactejemplar varchar2, vcolrev list_ejemp_doc, veje ejemplar)
is
vcoleje list_ejemp_doc:=vcolrev;
begin
if vactejemplar in ('R','B')  then
  update revista_t
  set doc_titulo=vrev.doc_titulo,
  doc_fecha_ingreso =vrev.doc_fecha_ingreso,
  doc_contenido  =vrev.doc_contenido,
  doc_año =vrev.doc_año,
  idi_documento =vrev.idi_documento,
  aut_documento =vrev.aut_documento,
  numero_publicacion =vrev.numero_publicacion
  where doc_id= vrev.doc_id;

elsif vactejemplar in ('E','B')  then
  begin
      vcoleje.extend;
       vcoleje(vcoleje.last):=veje;
       update revista_t r
       set   r.eje_documento =NULL
       where r.doc_id= vrev.doc_id;
       vcoleje(vcoleje.last):=veje;  --objeto tipo ejemplar
       update revista_t r
       set   r.eje_documento =vcoleje
       where r.doc_id= vrev.doc_id;
  end;
end if;
exception when others then
raise_application_error(-20999,'Error en actualización de revista ' ||sqlerrm);
end actualizarRevista_p;

function buscarContenido(vrevista in varchar2) return  list_revista  is
vlist_var_ejem  list_var_ejem;
vcol_revista list_revista;
begin

select value(vrev), e.eje_id--, e.EJE_ESTADO_PRESTAMO
bulk collect
into vcol_revista, vlist_var_ejem --, vejeid,vejeestado
from revista_t vrev, TABLE(vrev.eje_documento) e
where upper(vrev.DOC_CONTENIDO) like upper('%'||vrevista||'%')
order by vrev.doc_id desc;
return vcol_revista;
exception
when no_data_found then
return null;
when others then
raise_application_error(-20999,'Error en buscarContenido '||sqlerrm);
```

```
end buscarContenido;

function buscarContenidoL(vrevista in varchar2) return  list_revista  is
vlist_var_ejem  list_var_ejem;
vcol_revista list_revista;

begin

select value(vrev)
bulk collect
into vcol_revista
from revista_t vrev
where upper(vrev.DOC_CONTENIDO) like upper('%'||vrevista||'%')
order by vrev.DOC_TITULO desc;
return vcol_revista;
exception
when no_data_found then
return null;
when others then
raise_application_error(-20999,'Error en buscarContenido '||sqlerrm);
end buscarContenidoL;

function retornaEjemplar_p(vrevid number, vejeestadopre out varchar2, vejeestadoeje out varchar2,
vcount number,vejeid varchar2) return varchar2 is
vcolejemplar list_ejemp_doc;
vejemplar ejemplar;
vexceejemplar exception;
begin
vcolejemplar:= list_ejemp_doc();
begin
SELECT value(e)
bulk collect
into vcolejemplar
FROM revista_t d, TABLE(d.eje_documento) e
where doc_id=vrevid;
exception
when no_data_found then
raise_application_error(-20009,'No existe el ejemplar');
end;

SAVE_LOG('RETORNA EJEMPLAR '||vejeid);
if vejeid is not null then
  for c1 in 1..vcolejemplar.count loop
   if vcolejemplar(c1).eje_id= vejeid then
     vejemplar:=vcolejemplar(c1);
     vejeestadopre:=vejemplar.EJE_ESTADO_PRESTAMO;
     vejeestadoeje:=vejemplar.EJE_ESTADO_EJEMPLAR;
   end if;
   end loop;
   if vejemplar is null then
   raise vexceejemplar;
   end if;
else
vejemplar:=vcolejemplar(vcount);
vejeestadopre:=vejemplar.EJE_ESTADO_PRESTAMO;
vejeestadoeje:=vejemplar.EJE_ESTADO_EJEMPLAR;
end if;
return vejemplar.eje_id;
exception
when vexceejemplar then
raise_application_error(-20009,'No existe el ejemplar');
when others then
raise_application_error(-20009,'Error en retornaEjemplar_p '||sqlerrm);
end retornaEjemplar_p;
end revista_p;
/
```

```
--metodos de Revista


create or replace type body revista as
static function crearRevista(vdoc documento, vnumpublica number,vejemplar ejemplar) return ref revista is
vrevista ref revista;
begin
vrevista:=revista_p.crearRevista_p(vdoc, vnumpublica,vejemplar);
return vrevista;
end;

member procedure actualizarRevista(vrev revista, vactejemplar varchar2,vcolrev list_ejemp_doc ,veje
ejemplar) is
begin
  revista_p.actualizarRevista_p(vrev, vactejemplar , vcolrev, veje);
end;

static function buscarRevista(vrevista out varchar2, vrefrevista in out ref revista, vidrevista in out  number,
vcolejemplar list_ejemp_doc, vejemplar out ejemplar, videjemplar varchar2) return  revista is
vobjrev revista;
vcount  integer;

 begin
necesite
if vrefrevista is not null  then
   select vrefrev.doc_titulo,vrefrev.doc_id ,value(vrefrev)
   into vrevista, vidrevista,vobjrev
   from revista_t vrefrev
   where ref (vrefrev)=vrefrevista;

 elsif vidrevista is not null then
   select vrefrev.doc_titulo, ref(vrefrev), value(vrefrev)
   into vrevista,vrefrevista,vobjrev
   from revista_t vrefrev
   where vrefrev.doc_id = vidrevista;

elsif videjemplar is not null then
    for vcount in 1..vcolejemplar.count loop
        if vcolejemplar(vcount).eje_id = videjemplar then
          vejemplar:=vcolejemplar(vcount);
         end if;
      end loop;

 end if;

return vobjrev;
end;
member function visualizarContenido (vrefrevista ref revista, vidrevista number)  return varchar2 is
vdes varchar2(4000);
begin

if vrefrevista is not null then
 select vrev.doc_contenido
 into  vdes
 from revista_t vrev
 where ref(vrev)=vrefrevista;
elsif vidrevista is not null then
 select vrev.doc_contenido
 into  vdes
 from revista_t vrev
 where vrev.doc_id= vidrevista;
end if;
return vdes;
end;

static procedure actualizarRevistaEjm(vidrevista number, videjemplar varchar2, vejemplar ejemplar) is--,
vcolejemplar list_ejemp_doc) is
```

```
begin

  update table(select d.eje_documento from revista_t d
            where  d.doc_id =vidrevista) p
   set value (p) = vejemplar
   where p.eje_id = videjemplar;

end;
end;
/

---------crear type LIBRO**************************************/

create or replace type libro under documento
(
primary key (doc_id),
lib_isbn varchar2(10),
lib_editorial varchar2(20),
lib_edicion number(2),
lib_paginas number(4),
member function crearLibro(vdoc documento, vlib_isbn varchar2, vlib_editorial varchar2,vlib_edicion
number,lib_paginas number ),
member procedure actualizarLibro(vdoc documento, vlib libro),
--member procedure buscarRevista(vrevista varchar2) implementación en query hecho desde el forms
member function buscarLibro(vrevista varchar2)
);

------persistencia libro----------

create table libro_t of libro
(primary key (doc_Id))
   nested table eje_documento store as libro_ejemplar_t
   ((primary key(eje_id))
      nested table pre_ejemplar store as lib_ejem_prestamo_t
      ((primary key (pre_id))
      )
   );

--------type TESIS**********************************************/

create type tesis under documento
(
primary key (doc_id),
director varchar2(20),
member function crearTesis(vdoc documento, vdirector varchar2),
member procedure actualizarLibro(vdoc documento, vtesis  tesis),
);

-------persistencia tesis------------

create table tesis_t of tesis
(primary key (doc_Id))
   nested table eje_documento store as tesis_ejemplar_t
   ((primary key(eje_id))
      nested table pre_ejemplar store as tes_ejem_prestamo_t
      ((primary key (pre_id))
      )
   );

--paquete que permite administrar la reservación

create or replace package Administra as
procedure crearReservacion_a(vejemplar varchar2, vusuario varchar2, vreserva boolean,vidreserva
number,vnumejemplar number);
function retornadocid_a(vejemplar varchar2) return number;
end Administra;
/
```

```
create or replace package body Administra as
procedure crearReservacion_a(vejemplar varchar2, vusuario varchar2, vreserva boolean,vidreserva number,vnumejemplar number) is
vrefreserva ref reservacion;
vobjusuario usuarioconsulta;
vrefusuario ref usuarioconsulta:=null;
vidusuario varchar2(10):=vusuario;
begin
begin
vobjusuario:=usuarioconsulta.buscarusuario(vidusuario,vrefusuario);
exception
when no_data_found then
Raise_application_error(-20000,'El usuario no se encuentra registrado.');
when others then
Raise_application_error(-20001,'Error en buscar usuario. Revise '||sqlerrm);
end;
error

if vreserva = true and vnumejemplar=0 then
begin
vrefreserva:=reservacion.crearreservacion(vejemplar);
exception
when others then
raise_application_error(-20002,'Crea reserva '||sqlerrm);
end;
else
begin
reservacion.crearReservaEjemplar(vidreserva, vejemplar);--metodo de reservación en el que se añade un ejemplar a una reserva existente
exception
when others then
raise_application_error(-20003,'Crea reserva Ejemplar '||sqlerrm);
end;
end if;
begin
vobjusuario.actualizarusuario(vusuario,null,null,null,null,null,vrefreserva,null);
exception
when others then
raise_application_error(-20004,'Act usuario '||sqlerrm);
end;

end crearReservacion_a;

function retornadocid_a(vejemplar varchar2) return number is
viddoc number;
begin
viddoc:=to_number(substr(vejemplar,1,instr(vejemplar,'-',1)-1));
return viddoc;

end retornadocid_a;

end Administra;
/
```

ANEXO G. Pruebas del Sistema

Procedimiento de Prueba para Caso de Uso Administrar Préstamo	Caso de Prueba Prestar: 1. Seleccione en el TAB principal la opción Préstamo y escoja Prestar, se abre la página Prestar 2. Ingrese el código del usuario, el código del ejemplar 3. Escoja el tipo del documento 4. Pulse el botón Prestar 5. Transacción Realizada Satisfactoriamente Caso de Prueba Devolver: 1. Seleccione en el TAB principal la opción Préstamo y escoja Devolver, se abre la página Devolver 2. Ingrese el código del usuario, el código del ejemplar 3. Escoja el tipo de documento 4. Pulse el botón Devolver 5. Transacción Realizada Satisfactoriamente 6. Pero si el ejemplar tiene multa presenta un mensaje de ejemplar multado, pulse Aceptar. 7. Pulse el botón Pagar Multa, se abre la página Pago de Multa 8. Escoja el motivo de la multa 9. Si el motivo es por atraso el sistema calcula la multa, en caso contrario es por otros motivos ingrese el monto de la multa 10. Pulse el botón Registrar Pago. Caso de Prueba Renovar: 1. Seleccione en el TAB principal la opción Préstamo y escoja Renovar, se abre la página Renovar 2. Ingresar el código del usuario, el código del ejemplar 3. Escoja el tipo de Documento 4. Pulse el botón Renovar 5. Transacción realizada Satisfactoriamente 6. Pero si el ejemplar tiene multa presenta un mensaje de ejemplar multado, pulse Aceptar. 7. Pulse el botón Pagar Multa, se abre la página Pago de Multa 8. Escoja el motivo de la multa 9. Si el motivo es por atraso el sistema calcula la multa, en caso contrario es por otros motivos ingrese el monto de la multa 10. Pulse el botón Registrar Pago.

Descripción general:	Prestar Ejemplar
Datos de Entrada:	Código del Usuario: 9911080 Tipo del Documento: Tesis Código del Ejemplar: 1002-2
Resultados Esperados:	Préstamo de un Ejemplar
Condiciones para esta prueba:	El ejemplar este disponible
Fechas de Ejecución de la Prueba:	*18 de Septiembre del 2006*
Responsables de Ejecución de la Prueba:	*Responsable 1: Jorge Caballero* *Responsable 2: Pablo León*
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Descripción general:	Préstamo Fallido de un Ejemplar
Datos de Entrada:	Código del Usuario: 9911080 Tipo del Documento: en blanco Código del Ejemplar: en blanco
Resultados Esperados:	El sistema no debe permitir registrar el préstamo del Ejemplar porque el casillero Tipo de Ejemplar no fue elegido y Código del Ejemplar está vacío. El sistema debe emitir mensajes de error acordes.
Condiciones para esta prueba:	El Tipo de Ejemplar no fue elegido y el Código del Ejemplar no fue ingresado
Fechas de Ejecución de la Prueba:	*18 de Septiembre del 2006*
Responsables de Ejecución de la Prueba:	*Responsable 1: Jorge Caballero* *Responsable 2: Pablo León*
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Descripción general:	Devolver Ejemplar
Datos de Entrada:	Código del Usuario: 9911080 tipo del Documento: Tesis Código del Ejemplar: 1002-2
Resultados Esperados:	La devolución del Ejemplar sino esta multado, en caso contrario primero realizar el pago de la multa y después la devolución del Ejemplar.
Condiciones para esta prueba:	El ejemplar este prestado
Fechas de Ejecución de la Prueba:	*18 de Septiembre del 2006*
Responsables de Ejecución de la Prueba:	*Responsable 1: Jorge Caballero* *Responsable 2: Pablo León*
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Descripción general:	Devolución Fallida de un Ejemplar
Datos de Entrada:	Código del Usuario: en blanco tipo del Documento: Tesis Código del Ejemplar: en blanco
Resultados Esperados:	El sistema no debe permitir registrar la devolución del ejemplar porque los casilleros Código de Usuario y Código del Ejemplar están vacíos. El sistema debe emitir mensajes de error acordes
Condiciones para esta prueba:	No ingresar el Código de Usuario y Código del Ejemplar
Fechas de Ejecución de la Prueba:	*18 de Septiembre del 2006*
Responsables de Ejecución de la Prueba:	*Responsable 1: Jorge Caballero* *Responsable 2: Pablo León*
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Descripción general:	Renovar Ejemplar
Datos de Entrada:	Código del Usuario: 9910185 tipo del Documento: Revista Código del Ejemplar: 1003-3
Resultados Esperados:	Renovar el préstamo del Ejemplar sino esta multado, en caso contrario primero realizar el pago de la multa y después la renovación del Ejemplar.
Condiciones para esta prueba:	El ejemplar este prestado
Fechas de Ejecución de la Prueba:	*18 de Septiembre del 2006*
Responsables de Ejecución de la Prueba:	*Responsable 1: Jorge Caballero* *Responsable 2: Pablo León*
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Descripción general:	Renovación Fallida de un Ejemplar
Datos de Entrada:	Código del Usuario: 99101 tipo del Documento: Revista Código del Ejemplar: en blanco
Resultados Esperados:	El sistema no debe permitir registrar la renovación del ejemplar porque en el casillero Código de Usuario se ingresó un código no válido y el Código del Ejemplar está vacío. El sistema debe emitir mensajes de error acordes
Condiciones para esta prueba:	Ingresar un código de usuario no válido y no ingresar el Código del Ejemplar
Fechas de Ejecución de la Prueba:	*18 de Septiembre del 2006*
Responsables de Ejecución de la Prueba:	*Responsable 1: Jorge Caballero* *Responsable 2: Pablo León*
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Procedimiento de Prueba para Caso de Uso Administrar Documento y/o Ejemplar	Caso de Prueba Crear Documento Libro: 1. Seleccione en el TAB principal la opción Documento y escoja Crear, se abre la página Crear 2. Escoja el tipo de documento (Libro) 3. Ingrese el título del Documento, ingrese o escoja el autor, escoja el idioma, ingrese el contenido, ingrese el año, escoja categoría, Ingrese el Editorial, ingrese la Edición 4. Pulse el botón Crear. 5. Muestra el código del Ejemplar que se crea cuando se crea un Documento Caso de Prueba Crear Documento Tesis: 1. Seleccione en el TAB principal la opción Documento y escoja Crear, se abre la página Crear 2. Escoja el tipo de documento (Tesis) 3. Ingrese el título del Documento, ingrese o escoja el autor, escoja el idioma, ingrese el contenido, ingrese el año, Ingrese el Director. 4. Pulse el botón Crear 5. Muestra el código del Ejemplar que se crea cuando se crea un Documento Caso de Prueba Crear Documento Revista: 1. Seleccione en el TAB principal la opción Documento y escoja Crear, se abre la página Crear 2. Escoja el tipo de documento (Revista) 3. Ingrese el título del Documento, ingrese o escoja el autor, escoja el idioma, ingrese el contenido, ingrese el año, Ingrese el número de publicaciones. 4. Pulse el botón Crear 5. Muestra el código del Ejemplar que se crea cuando se crea un Documento Caso de Prueba Actualizar Documento Libro: 1. Seleccione en el TAB principal la opción Documento y escoja Actualizar, se abre la página Actualizar 2. Escoja el tipo de documento (Libro) 3. Ingrese el título del Documento, ingrese o escoja el autor, escoja el idioma, ingrese el contenido, ingrese el año, escoja categoría, Ingrese el Editorial, ingrese la Edición 4. Pulse el botón Actualizar.

	5. Transacción Realizada Satisfactoriamente Caso de Prueba Actualizar Documento Tesis: 1. Seleccione en el TAB principal la opción Documento y escoja Actualizar, se abre la página Actualizar 2. Escoja el tipo de documento (Tesis) 3. Ingrese el título del Documento, ingrese o escoja el autor, escoja el idioma, ingrese el contenido, ingrese el año, Ingrese el Director. 4. Pulse el botón Crear 5. Transacción Realizada Satisfactoriamente Caso de Prueba Actualizar Documento Revista: 1. Seleccione en el TAB principal la opción Documento y escoja Actualizar, se abre la página Actualizar 2. Escoja el tipo de documento (Revista) 3. Ingrese el título del Documento, ingrese o escoja el autor, escoja el idioma, ingrese el contenido, ingrese el año, Ingrese el número de publicaciones. 4. Pulse el botón Actualizar 5. Transacción Realizada Satisfactoriamente Caso de Prueba Crear Ejemplar: 1. Seleccione en el TAB principal la opción Ejemplar y escoja Crear, se abre la página Crear 2. Escoja el tipo de documento (libro/tesis/revista), título del documento, categoría (original o copia). 3. Pulse el botón Crear. 4. Muestra el código del Ejemplar Caso de Prueba Actualizar Ejemplar: 5. Seleccione en el TAB principal la opción Ejemplar y escoja Actualizar, se abre la página Actualizar 6. Ingresar el código del Ejemplar 7. Escoja el tipo del documento (libro/tesis/revista), categoría (original o copia), estado del ejemplar 8. Pulse el botón Actualizar. 9. Transacción Realizada Satisfactoriamente

Descripción general:	Crear Documento (Revista)
Datos de Entrada:	Tipo del Documento: Revista Titulo: PC Word Autor: Jorge Ortiz Idioma: Español Contenido: La nueva tecnología Core Duo en Portátiles Hacker vs Craker Arquitectura de 64 bit en computadores personales Tecnología UMA Año: 2006 Numero de Publicación: 9
Resultados Esperados:	Crear un documento de tipo Revista con su respectivo Ejemplar
Condiciones para esta prueba:	Ninguna
Fechas de Ejecución de la Prueba:	*18 de Septiembre del 2006*
Responsables de Ejecución de la Prueba:	*Responsable 1: Jorge Caballero* *Responsable 2: Pablo León*
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Descripción general:	Creación Fallida de un Documento (Revista)
Datos de Entrada:	Tipo del Documento: en blanco Titulo: en blanco Autor: Jorge Ortiz Idioma: Español Contenido: La nueva tecnología Core Duo en Portátiles Hacker vs Craker Arquitectura de 64 bit en computadores personales Tecnología UMA Año: 2006 Numero de Publicación: 9
Resultados Esperados:	El sistema no permita registrar la creación de un documento con su respectivo ejemplar porque el casillero Tipo de Documento no es escogido y el título está vacío. El sistema debe emitir mensajes de error acordes
Condiciones para esta prueba:	No escoger el Tipo del Documento y no ingresar el título del documento.
Fechas de Ejecución de la Prueba:	*18 de Septiembre del 2006*
Responsables de Ejecución de la Prueba:	*Responsable 1: Jorge Caballero* *Responsable 2: Pablo León*
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Descripción general:	Actualizar Documento (Revista)
Datos de Entrada:	Tipo del Documento: Revista Titulo del Documento: PC World
Resultados Esperados:	Actualizar la información de la revista

Condiciones para esta prueba:	Que la revista esté creada en el sistema
Fechas de Ejecución de la Prueba:	*18 de Septiembre del 2006*
Responsables de Ejecución de la Prueba:	*Responsable 1: Jorge Caballero* *Responsable 2: Pablo León*
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Descripción general:	Actualización Fallida del Documento (Revista)
Datos de Entrada:	Tipo del Documento: Revista Titulo del Documento: en blanco
Resultados Esperados:	El sistema no permita registrar la actualización del Documento porque el casillero del Título del Documento no fue ingresado. El sistema debe emitir mensajes de error acordes
Condiciones para esta prueba:	No ingresar el Título del Documento
Fechas de Ejecución de la Prueba:	*18 de Septiembre del 2006*
Responsables de Ejecución de la Prueba:	*Responsable 1: Jorge Caballero* *Responsable 2: Pablo León*
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Descripción general:	Crear Ejemplar (Revista)
Datos de Entrada:	Tipo del Documento: Revista Titulo: PC Word Categoría: Copia
Resultados Esperados:	Crear un Ejemplar de tipo Revista
Condiciones para esta prueba:	Que exista un documento ingresado
Fechas de Ejecución de la Prueba:	*18 de Septiembre del 2006*
Responsables de Ejecución de la Prueba:	*Responsable 1: Jorge Caballero* *Responsable 2: Pablo León*
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Descripción general:	Creación Fallida del Ejemplar (Revista)
Datos de Entrada:	Tipo del Documento: en blanco Titulo: PC Word Tipo de Ejemplar: Copia
Resultados Esperados:	El sistema no debe permitir registrar la creación del Ejemplar (Revista) porque el casillero Tipo del Documento no fue escogido. El sistema debe emitir mensajes de error acordes.
Condiciones para esta prueba:	No ingresar el tipo del documento

Fechas de Ejecución de la Prueba:	*18 de Septiembre del 2006*
Responsables de Ejecución de la Prueba:	*Responsable 1: Jorge Caballero* *Responsable 2: Pablo León*
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Descripción general:	Actualizar Ejemplar (Revista)
Datos de Entrada:	Código del Ejemplar: 1001-4 Tipo del Documento: Revista
Resultados Esperados:	Actualización de la información del ejemplar en el sistema
Condiciones para esta prueba:	Que exista el ejemplar a actualizar en el sistema
Fechas de Ejecución de la Prueba:	*18 de Septiembre del 2006*
Responsables de Ejecución de la Prueba:	*Responsable 1: Jorge Caballero* *Responsable 2: Pablo León*
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Descripción general:	Actualización Fallida del Ejemplar (Revista)
Datos de Entrada:	Código: 0000-3 Tipo del Documento: Revista
Resultados Esperados:	El sistema no debe permitir registrar la actualización del Ejemplar porque en el casillero del código se ingreso un código no válido. El sistema debe emitir mensajes de error acordes.
Condiciones para esta prueba:	Ingresar un código no valido en el sistema
Fechas de Ejecución de la Prueba:	*18 de Septiembre del 2006*
Responsables de Ejecución de la Prueba:	*Responsable 1: Jorge Caballero* *Responsable 2: Pablo León*
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Procedimiento de Prueba para Caso de Uso Administrar Categoría	Caso de Prueba Actualizar Categoría : 1. Seleccione en el TAB principal la opción Categoría y escoja Actualizar, se abre la página Actualizar 2. Escoja el nombre de la Categoría 3. Ingrese el valor por día de atraso, días de préstamo. 4. Pulse el botón Actualizar 5. Transacción Realizada Satisfactoriamente

Descripción general:	Actualizar Categoría
Datos de Entrada:	Nombre de la Categoría: Original Valor por días de atraso: 0.50 Días de Préstamo: 1
Resultados Esperados:	Actualizar la información de la categoría
Condiciones para esta prueba:	Que este ingresado el nombre de la categoría en el sistema
Fechas de Ejecución de la Prueba:	*18 de Septiembre del 2006*
Responsables de Ejecución de la Prueba:	*Responsable 1: Jorge Caballero* *Responsable 2: Pablo León*
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Descripción general:	Actualización Fallida de la Categoría
Datos de Entrada:	Nombre de la Categoría: Original Valor por días de atraso: en blanco Días de Préstamo: 1
Resultados Esperados:	El sistema no debe permitir registrar la actualización de la categoría por el casillero del valor por días de atraso no fue ingresado. El sistema debe emitir mensajes de error acordes.
Condiciones para esta prueba:	No ingresar el valor por días de atraso
Fechas de Ejecución de la Prueba:	*18 de Septiembre del 2006*
Responsables de Ejecución de la Prueba:	*Responsable 1: Jorge Caballero* *Responsable 2: Pablo León*
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Procedimiento de Prueba para Caso de Uso Administrar Idioma	Caso de Prueba Crear Idioma: 1. Seleccione en el TAB principal la opción Idioma y escoja Crear, se abre la página Crear 2. Ingrese el nombre del Idioma 3. Pulse el botón Crear 4. Transacción Realizada Satisfactoriamente Caso de Prueba Actualizar Idioma: 1. Seleccione en el TAB principal la opción Autor y escoja Actualizar, se abre la página Actualizar 2. Escoja el nombre del Idioma. 3. Modificar el Idioma 4. Pulse el botón Actualizar 5. Transacción Realizada Satisfactoriamente

Descripción general:	Crear Idioma
Datos de Entrada:	Nombre del Idioma: Inglés
Resultados Esperados:	Crear el idioma Inglés
Condiciones para esta prueba:	Que el idioma no se encuentre registrado en el sistema
Fechas de Ejecución de la Prueba:	*18 de Septiembre del 2006*
Responsables de Ejecución de la Prueba:	*Responsable 1: Jorge Caballero* *Responsable 2: Pablo León*
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Descripción general:	Creación Fallida del Idioma
Datos de Entrada:	Nombre del Idioma: Inglés
Resultados Esperados:	El sistema no debe permitir registrar la creación del Idioma porque el Idioma que se esta ingresando ya existe. El sistema debe emitir mensajes de error acordes
Condiciones para esta prueba:	Que el idioma que se esta ingresando ya exista en el sistema
Fechas de Ejecución de la Prueba:	*18 de Septiembre del 2006*
Responsables de Ejecución de la Prueba:	*Responsable 1: Jorge Caballero* *Responsable 2: Pablo León*
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Descripción general:	Actualizar Idioma
Datos de Entrada:	Nombre del Idioma: Inglés
Resultados Esperados:	Actualizar el nombre del Idioma
Condiciones para esta prueba:	Que el idioma esté registrado en el sistema
Fechas de Ejecución de la Prueba:	*18 de Septiembre del 2006*
Responsables de Ejecución de la Prueba:	*Responsable 1: Jorge Caballero* *Responsable 2: Pablo León*
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Procedimiento de Prueba para Caso de Uso Administrar Autor	Caso de Prueba Crear Autor: 1. Seleccione en el TAB principal la opción Autor y escoja Crear, se abre la página Crear 2. Ingrese el nombre del autor, email y la dirección Web. 3. Pulse el botón Crear

	4. Transacción Realizada Satisfactoriamente Caso de Prueba Actualizar Autor: 1. Seleccione en el TAB principal la opción Autor y escoja Actualizar, se abre la página Actualizar 2. Escoja el nombre del autor. (además se lo puede modificar) 3. Modificar el email y la dirección Web 4. Pulse el botón Actualizar 5. Transacción Realizada Satisfactoriamente

Descripción general:	Crear Autor
Datos de Entrada:	Nombre del Autor: Robert Pressman Email: robert@editorialhall.com Dirección Web: www.ingenieriadesw.com
Resultados Esperados:	El autor pueda ser ingresado
Condiciones para esta prueba:	Que el autor no esté registrado en el sistema
Fechas de Ejecución de la Prueba:	*18 de Septiembre del 2006*
Responsables de Ejecución de la Prueba:	*Responsable 1: Jorge Caballero* *Responsable 2: Pablo León*
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Descripción general:	Creación Fallida del Autor
Datos de Entrada:	Nombre del Autor: en blanco Email: robert@editorialhall.com Dirección Web: www.ingenieriadesw.com
Resultados Esperados:	El sistema no debe permitir registrar la creación del autor del documento porque el casillero del Nombre del Autor no fue ingresado. El sistema debe emitir mensajes de error acordes.
Condiciones para esta prueba:	No ingresar el nombre del autor
Fechas de Ejecución de la Prueba:	*18 de Septiembre del 2006*
Responsables de Ejecución de la Prueba:	*Responsable 1: Jorge Caballero* *Responsable 2: Pablo León*
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Descripción general:	Actualizar Autor
Datos de Entrada:	Nombre del Autor: Robert Pressman
Resultados Esperados:	Modificar la información del Autor

Condiciones para esta prueba:	El autor este registrado en el sistema
Fechas de Ejecución de la Prueba:	*18 de Septiembre del 2006*
Responsables de Ejecución de la Prueba:	*Responsable 1: Jorge Caballero* *Responsable 2: Pablo León*
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Procedimiento de Prueba para Caso de Uso Administrar Usuario del Sistema	Caso de Prueba Crear Usuario del Sistema: 1. Seleccione en el TAB principal la opción Usuario del Sistema y escoja Crear, se abre la página Crear 2. Ingrese el código, el Nombre, Fecha de Ingreso, Usuario, Contraseña. 3. Escoja el tipo de Usuario del Sistema (Administrador/Bibliotecario) 4. Pulse el botón Crear 5. Transacción Realizada Satisfactoriamente Caso de Prueba Actualizar Usuario del Sistema: 1. Seleccione en el TAB principal la opción Usuario del Sistema y escoja Actualizar, se abre la página Actualizar 2. Escoja el nombre o escriba el código del Usuario del Sistema (además se lo puede modificar) 3. Modificar Fecha de Retiro, Usuario, Contraseña del Usuario del Sistema. 4. Escoja el tipo y el estado del Usuario del Sistema 5. Pulse el botón Actualizar 6. Transacción Realizada Satisfactoriamente

Descripción general:	Crear Usuario del Sistema
Datos de Entrada:	Código:1716456877 Nombre: Isabel Mendoza Fecha de Ingreso: 10/07/2006 Usuario: izha Contraseña: izha25 Dirección: Av. Gran Colombia Tipo: Bibliotecario
Resultados Esperados:	Crear el Bibliotecario
Condiciones para esta prueba:	El bibliotecario no esté registrado en el sistema
Fechas de Ejecución de la Prueba:	*18 de Septiembre del 2006*
Responsables de Ejecución de la	*Responsable 1: Jorge Caballero* *Responsable 2: Pablo León*

Prueba:	
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Descripción general:	Creación Fallida del Usuario de Sistema
Datos de Entrada:	Código:1716456877 Nombre: en blanco Fecha de Ingreso: 10/07/2006 Usuario: izha Contraseña: en blanco Tipo: Bibliotecario
Resultados Esperados:	El sistema no debe permitir registrar la creación del Usuario del Sistema porque el casillero Nombre y contraseña no fueron ingresados. El sistema debe emitir mensajes de error acordes
Condiciones para esta prueba:	No ingresar Nombre y Contraseña del Usuario del Sistema
Fechas de Ejecución de la Prueba:	*18 de Septiembre del 2006*
Responsables de Ejecución de la Prueba:	*Responsable 1: Jorge Caballero* *Responsable 2: Pablo León*
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Descripción general:	Actualizar Usuario del Sistema
Datos de Entrada:	Nombre: Isabel Mendoza
Resultados Esperados:	Modificar la información del Usuario del Sistema
Condiciones para esta prueba:	El Usuario esté registrado en el sistema
Fechas de Ejecución de la Prueba:	*18 de Septiembre del 2006*
Responsables de Ejecución de la Prueba:	*Responsable 1: Jorge Caballero* *Responsable 2: Pablo León*
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Procedimiento de Prueba para Caso de Uso Administrar Usuario Consulta	Caso de Prueba Crear Usuario Consulta: 1. Seleccione en el TAB principal la opción Usuario Consulta y escoja Crear, se abre la página Crear 2. Ingrese la identificación del usuario, el nombre del usuario, dirección del usuario, teléfono del usuario 3. Pulse el botón Crear 4. Transacción Realizada Satisfactoriamente Caso de Prueba Actualizar Usuario Consulta:

	1. Seleccione en el TAB principal la opción Usuario del Sistema y escoja Actualizar, se abre la página Actualizar 2. Escoja el nombre del Usuario del Sistema (además se lo puede modificar) 3. Modificar la dirección, teléfono del Usuario del Sistema. 4. Escoger el Estado del Usuario del Sistema 5. Pulse el botón Actualizar 6. Transacción Realizada Satisfactoriamente

Descripción general:	Crear Usuario Consulta
Datos de Entrada:	Código: 9911080 Nombre: Leonel Caballero Dirección: El Dorado Teléfono: 2226598
Resultados Esperados:	Crear un Usuario Consulta
Condiciones para esta prueba:	El Usuario no esté registrado en el sistema
Fechas de Ejecución de la Prueba:	*18 de Septiembre del 2006*
Responsables de Ejecución de la Prueba:	*Responsable 1: Jorge Caballero* *Responsable 2: Pablo León*
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Descripción general:	Creación Fallida del Usuario Consulta
Datos de Entrada:	Código:9911080 Nombre: en blanco Dirección: El Dorado Teléfono: 2226598
Resultados Esperados:	El sistema no debe permitir registrar la creación del Usuario Consulta por que el nombre no fue ingresado. El sistema debe emitir mensajes de error acordes.
Condiciones para esta prueba:	No ingresar el nombre del Usuario Consulta
Fechas de Ejecución de la Prueba:	*18 de Septiembre del 2006*
Responsables de Ejecución de la Prueba:	*Responsable 1: Jorge Caballero* *Responsable 2: Pablo León*
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Descripción general:	Actualizar Usuario Consulta
Datos de Entrada:	Nombre: Leonel Caballero
Resultados Esperados:	Modificar el Usuario Consulta

Condiciones para esta prueba:	El Usuario esté registrado en el sistema
Fechas de Ejecución de la Prueba:	*18 de Septiembre del 2006*
Responsables de Ejecución de la Prueba:	*Responsable 1: Jorge Caballero* *Responsable 2: Pablo León*
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Procedimiento de Prueba para Caso de Uso Reportes	Caso de Prueba Ejemplares no Devueltos a Tiempo: 1. Seleccione en el TAB principal la opción Reportes, se abre la página de Reportes 2. Ingrese la fecha inicial y la fecha final. 3. Escoja el tipo de reporte (Ejemplares no Devueltos a Tiempo) 4. Pulse el botón generar reporte Caso de Prueba Usuarios Multados: 1. Seleccione en el TAB principal la opción Reportes, se abre la página de Reportes 2. Ingrese la fecha inicial y la fecha final. 3. Escoja el tipo de reporte (Usuarios Multados) 4. Pulse el botón generar reporte

Descripción general:	Ejemplares no Devueltos a Tiempo
Datos de Entrada:	Fecha Inicial: 02/08/2006 Fecha Final: 02/09/2006 Tipo de Reporte: Ejemplares no Devueltos a Tiempo
Resultados Esperados:	Generar el Reporte
Condiciones para esta prueba:	Ingresar adecuados rangos de Tiempo
Fechas de Ejecución de la Prueba:	*18 de Septiembre del 2006*
Responsables de Ejecución de la Prueba:	*Responsable 1: Jorge Caballero* *Responsable 2: Pablo León*
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Descripción general:	Reporte Fallido de Ejemplares no Devueltos a Tiempo
Datos de Entrada:	Fecha Inicial: en blanco Fecha Final: 02/09/2006 Tipo de Reporte: Ejemplares no Devueltos a Tiempo
Resultados Esperados:	El sistema no debe permitir generar el reporte de Ejemplares no Devueltos a Tiempo. El sistema debe emitir mensajes de error acordes.
Condiciones para	No ingresar Fecha Inicial

esta prueba:	
Fechas de Ejecución de la Prueba:	*18 de Septiembre del 2006*
Responsables de Ejecución de la Prueba:	*Responsable 1: Jorge Caballero* *Responsable 2: Pablo León*
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Descripción general:	Usuarios Multados
Datos de Entrada:	Fecha Inicial: 02/08/2006 Fecha Final: 02/09/2006 Tipo de Reporte: Usuarios Multados
Resultados Esperados:	Generar el Reporte
Condiciones para esta prueba:	Ingresar adecuados rangos de Tiempo
Fechas de Ejecución de la Prueba:	*18 de Septiembre del 2006*
Responsables de Ejecución de la Prueba:	*Responsable 1: Jorge Caballero* *Responsable 2: Pablo León*
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Descripción general:	Reportes Fallidos de Usuarios Multados
Datos de Entrada:	Fecha Inicial: 02/09/2006 Fecha Final: 02/08/2006 Tipo de Reporte: Usuarios Multados
Resultados Esperados:	El sistema no debe permitir generar reportes de Usuarios Multados. Sistema debe emitir mensajes acordes.
Condiciones para esta prueba:	El rango de tiempo de donde se realizar el reporte no es adecuado
Fechas de Ejecución de la Prueba:	*18 de Septiembre del 2006*
Responsables de Ejecución de la Prueba:	*Responsable 1: Jorge Caballero* *Responsable 2: Pablo León*
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Procedimiento de Prueba para Caso de Uso Reserva y Visualización de Contenido de Documento	Caso de Prueba Consultar: 1. Escriba la dirección Web del sistema de Biblioteca para el Usuario Consulta y escoja Consultar, se abre la página Consultar 2. Ingrese su consulta 3. Pulse el botón Consulta 4. Despliega la lista de documentos encontrados 5. Pulse el botón Visualizar Contenido

	6. Despliega el contenido del Documento 7. Si desea Reservar el Documento 8. Pulse el botón Reservar 9. Se abre la página de reservación con el código del ejemplar que se desea reservar 10. Ingrese el código del Usuario Consulta 11. Pulse el botón reservar Caso de Prueba Reservación: 1. Seleccione en el TAB principal la opción reservar, se abre la página de reservación 2. Ingrese el código del Usuario Consulta, tipo de documento y el código del ejemplar 3. Pulse el botón Reservar 4. Transacción Realizada Satisfactoriamente

Descripción general:	Consultar Documento
Datos de Entrada:	Ingrese el o las palabras para la búsqueda: ingeniería de software Tipo de Documento: Libro
Resultados Esperados:	Muestra el título del Documento y su contenido, además permita realizar la reservación
Condiciones para esta prueba:	Debe estar ingresado un Documento con la palabra o frase buscada
Fechas de Ejecución de la Prueba:	*18 de Septiembre del 2006*
Responsables de Ejecución de la Prueba:	*Responsable 1: Jorge Caballero* *Responsable 2: Pablo León*
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Descripción general:	Consulta Fallida de un Documento
Datos de Entrada:	Ingrese el o las palabras para la búsqueda: ingeniería de software Tipo de Documento: Libro
Resultados Esperados:	El sistema no desligue información. El sistema debe emitir mensajes acordes
Condiciones para esta prueba:	El documento buscado no este registrado en el sistema
Fechas de Ejecución de la Prueba:	*18 de Septiembre del 2006*
Responsables de Ejecución de la Prueba:	*Responsable 1: Jorge Caballero* *Responsable 2: Pablo León*
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Descripción general	Reservar Ejemplar
Datos de Entrada:	Código del Usuario: 9910185 Tipo de Documento: Revista Código del Ejemplar: 1001-1
Resultados Esperados:	La reservación de un Ejemplar
Condiciones para esta prueba:	Que exista un Ejemplar y este disponible, además que el usuario este registrado en el sistema.
Fechas de Ejecución de la Prueba:	*18 de Septiembre del 2006*
Responsables de Ejecución de la Prueba:	*Responsable 1: Jorge Caballero* *Responsable 2: Pablo León*
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Descripción general	Reservación Fallida de un Ejemplar
Datos de Entrada:	Código del Usuario: 9914 Tipo de Documento: Revista Código del Ejemplar: 1001-1
Resultados Esperados:	El sistema no debe permitir registrar la reservación de un ejemplar porque el código del usuario no es válido. El sistema debe emitir mensajes de error acordes.
Condiciones para esta prueba:	Ingresar un código de usuario no registrado en el sistema.
Fechas de Ejecución de la Prueba:	*18 de Septiembre del 2006*
Responsables de Ejecución de la Prueba:	*Responsable 1: Jorge Caballero* *Responsable 2: Pablo León*
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Procedimiento de Prueba para Login	Caso de Prueba Ingresar al sistema: 1. Escriba la dirección Web del sistema de administración de biblioteca 2. Ingrese el nombre del usuario y la contraseña 3. Pulse el botón Aceptar 4. Ingresa en el Sistema de Administración de Biblioteca

Descripción general:	Ingresar al sistema
Datos de Entrada:	Nombre del Usuario: Pablo Contraseña: Pablo
Resultados Esperados:	Ingresar al sistema
Condiciones para esta prueba:	Que el usuario este registrado en el sistema
Fechas de Ejecución de la Prueba:	*18 de Septiembre del 2006*

Responsables de Ejecución de la Prueba:	*Responsable 1: Jorge Caballero* *Responsable 2: Pablo León*
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*

Descripción general:	Ingreso Fallido al sistema
Datos de Entrada:	Nombre del Usuario: nathaly Contraseña: nata
Resultados Esperados:	El sistema no debe permitir al usuario ingresar al sistema. El sistema debe emitir mensajes de error acordes.
Condiciones para esta prueba:	Nombre de Usuario y Contraseña incorrectos
Fechas de Ejecución de la Prueba:	*18 de Septiembre del 2006*
Responsables de Ejecución de la Prueba:	*Responsable 1: Jorge Caballero* *Responsable 2: Pablo León*
Resultados Obtenidos:	*Resultados obtenidos #1: Los resultados obtenidos correspondieron a los resultados esperados, por lo tanto la prueba esta concluida.*