Diseño e Implementación de un Sistema de Información Empresarial para la Administración Académica y de Recursos de un Postgrado

Autor: Br. Julio Daniel Matheus F.
Profesor Tutor: Dr. Wladimir Rodríguez

Universidad de los Andes
Facultad de Ingeniería
Escuela de Sistemas

Enero de 2005

Índice

Resumen

La empresa moderna requiere de la producción de software, que esté disponible las veinticuatro (24) horas del día los siete (7) días de la semana, que sea además, seguro escalable, portable, eficiente en el manejo de recursos, fácil de modificar y mantener.

Las aplicaciones empresariales son la respuesta de la Ingeniería de Software a las nuevas y crecientes necesidades de los usuarios de las empresas modernas. El enfoque actual para la construcción de sistemas distribuidos, se orienta al desarrollo de aplicaciones empresariales basado en componentes de software reutilizables.

Este trabajo de tesis aborda el tema del desarrollo de aplicaciones empresariales, empleando el modelo J2EE (Java 2 Platform Enterprise Edition) de Sun Microsystems. En el capítulo I se introducen los conceptos implicados por el desarrollo de aplicaciones empresariales basado en componentes.

Los capítulos II, III y IV se orientan al diseño del Sistema. El capítulo II se refiere al sistema de actividades humanas que apoyará la aplicación, en este caso, las actividades administrativas del Postgrado de Computación de la Universidad de los Andes. El capítulo III aborda el tema de la captura de requerimientos y el capítulo IV trata sobre el diseño de la base de datos.

Una vez concluida la etapa de diseño del sistema, en el capítulo V se desarrolla en detalle lo referente a la implementación de una aplicación empresarial, basada en componentes y, por ultimo, en el capítulo VI se comenta en forma resumida el proceso de validación y verificación del sistema.

Agradecimientos

Agradezco de todo corazón a Dios Padre, Hijo y Espíritu Santo. A mi mamá Carmen Gisela Flores Acosta; a mis dos papas: Julio Ramón Matheus Matheus y Onécimo Cueva, a mis queridos hermanos: Javier Matheus, Jeanny David Cuevas y Joselyn Cueva. A mis tíos y tías, a mi abuela Eva Acosta y a todos mis familiares y amigos.

Un especial agradecimiento a mi tía Rosa Isabel Cuevas Pereira, quien es pilar fundamental sobre el que se levantó este triunfo y quien me recibió como hijo suyo. Tía, eres mi verdadera Madrina.

Otro agradecimiento muy especial a la gente del Postgrado de Computación de la Universidad de los Andes. A mi tutor Dr. Wladimir Rodríguez, a Taniana y Luisa. Gracias por hacerme sentir como en casa, ustedes son mi familia y las instalaciones del Postgrado, mi hogar.

A la Universidad de los Andes y a toda su gente por permitirme formar parte de esta gran familia.

A todos gracias y mil bendiciones.

Capítulo I: Introducción

1.1- Antecedentes

Desde los inicios de la computación los programas se elaboraban de manera intuitiva y sobre la marcha. En principio, esta forma de desarrollo llenaba las expectativas de programadores y usuarios. Con el paso del tiempo, los programas han tendido a hacerse cada vez más complejos y costosos lo que desencadenó la llamada "crisis de software de los años sesenta".

Como respuesta a esta crisis surge la Ingeniería de Software como un conjunto de métodos, técnicas y herramientas para el desarrollo de software de alta calidad que satisfaga las crecientes necesidades de los usuarios. Pasando entonces, del desarrollo intuitivo al desarrollo metodológico de sistemas programados.

En Venezuela, la demanda de personal calificado en el área de la informática originó en el año 1995 el nacimiento del Programa de Estudios de Postgrado en Computación de la Universidad de los Andes, el cual es el resultado de un largo proceso de preparación académica que inició el Departamento de Computación de la Escuela de Sistemas durante la década de los ochenta.

En la actualidad, el Programa de Estudios de Postgrado ha experimentado un notable crecimiento en el número y complejidad de sus actividades haciéndose necesaria la búsqueda de una solución que apoye de la mejor manera posible el cumplimiento de estas actividades.

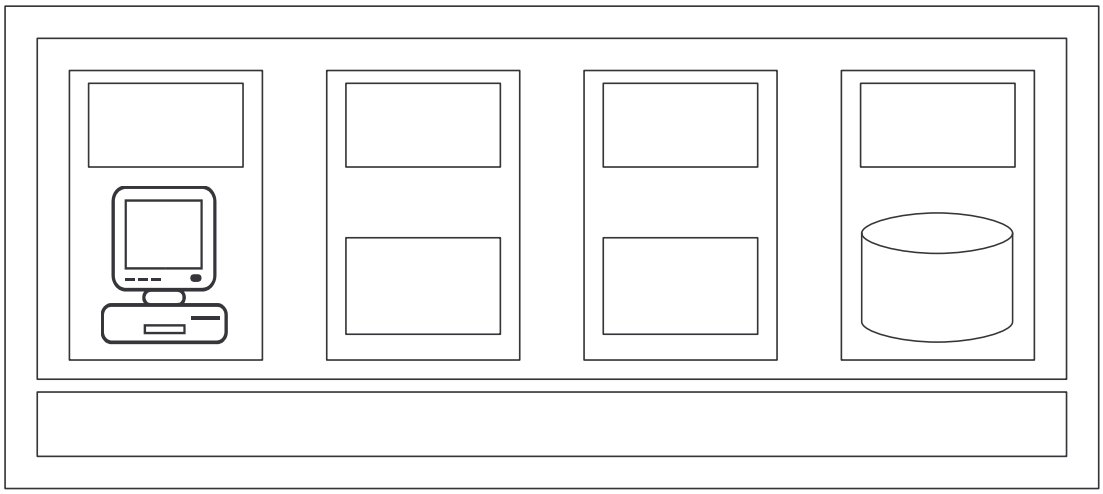
Este proyecto está basado en las tesis: "Desarrollo Orientado a Objeto de un Sistema de Información para la Administración de Recursos de un Postgrado" [Guerrero 99] y "Desarrollo Orientado a Objetos de un Sistema de Información para la Administración Académica de un Postgrado" [Delgado 99].

1.2.-Planteamiento del Problema

Las actividades académicas y administrativas del Postgrado de Computación se realizan de forma manual, lo cual no representaba un problema en sus inicios debido principalmente al bajo número de estudiantes. Pero, como se describió anteriormente, el Postgrado de Computación ha experimentado un notable crecimiento en los últimos años, demandando una solución a sus necesidades de información acorde con la tecnología actual y que contribuya a su promoción tanto a escala nacional como internacional.

Por estas razones se realizaron en el año 1999 las tesis para el apoyo de las actividades académicas de un postgrado [Delgado 99] y para la administración de recursos de un postgrado [Guerrero 99]. Es de resaltar, que de dichos proyectos no se realizó implementación alguna y de haberse realizado se habría hecho sobre una arquitectura de dos capas: cliente-servidor.

Por lo anteriormente expuesto se propone el diseño e implementación de un Sistema de Información Empresarial empleando una arquitectura de cuatro capas (figura 1) utilizando los métodos, técnicas y herramientas de la Ingeniería de Software.



Para tal fin, se tomará como base el diseño obtenido en las tesis anteriores [Guerrero 99] y [Delgado 99] y se rediseñará para adaptar el modelo a las nuevas necesidades del Postgrado de Computación. El desarrollo se realizará sobre una plataforma LINUX y utilizando el lenguaje de programación JAVA por ser orientado a objetos y porque goza actualmente de gran popularidad en la comunidad del software.

1.3.-Objetivos

1.3.1.- Aplicar las técnicas de la Ingeniería de Software para el modelado del sistema

1.3.2.- Diseñar el sistema de información empresarial utilizando los métodos y herramientas de la Ingeniería de Software empleando una arquitectura de cuatro capas: cliente, presentación, aplicación, y datos.

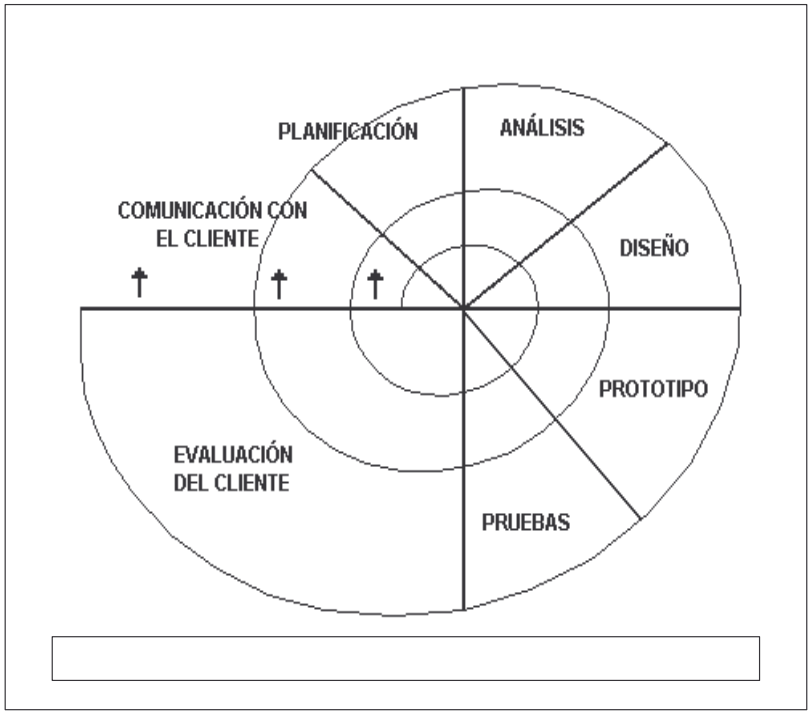
1.3.3.-Implementar el sistema utilizando un lenguaje de alto nivel y orientado a objetos sobre una plataforma Linux.

1.4.-Metodología

Para desarrollar el sistema se usará el modelo de proceso que se observa en la (figura2). Este modelo es una adaptación del modelo lineal secuencial y de construcción de prototipos con el modelo en espiral. A continuación se presenta una breve descripción de cada una de las etapas de este modelo:

- **Planificación:** comprende las actividades requeridas para definir recursos, el tiempo y otras informaciones relacionadas con el proyecto.

- **Análisis:** en esta etapa se agregan, modifican o desechan; según corresponda, los nuevos requerimientos capturados en la fase de evaluación del cliente. Luego, estos requerimientos se incorporan al modelo del sistema.



- **Diseño:** consiste en determinar la solución a las nuevas necesidades del cliente y agregarla al modelo antes de ser implementada para evaluar su efecto en el diseño general. El diseño se centra en cuatro atributos distintos de un programa: estructura de datos, arquitectura del software, interfaz y algoritmos.

- **Construcción de prototipo:** es la fase en que se traduce en código legible por la máquina el diseño obtenido en la etapa anterior.

- **Pruebas:** corresponde a las actividades necesarias para evaluar el software, es decir, para detectar y corregir errores.

- **Evaluación del cliente:** actividades requeridas para atrapar la reacción del cliente ante el software con la finalidad de capturar nuevos requerimientos y validar el sistema.

La dinámica del proceso se inicia en la fase de comunicación con el cliente. Seguidamente, avanzamos a la zona correspondiente a la ingeniería de sistemas de información, que comprende las fases de: planificación, análisis y diseño. A continuación se construye el prototipo, se realizan las pruebas y se le presenta al cliente el software con la finalidad de validar el sistema y capturar nuevos requerimientos. El ciclo se repite tantas veces como sea necesario y a cada vuelta completa corresponde la construcción de un prototipo.

1.5.-Marco Conceptual

1.5.1.- UML:

Para apoyar la etapa de ingeniería de sistemas de información se empleará el UML (Unified Modeling Language) o lenguaje unificado para el modelado de objetos. Esta notación gráfica se usa para representar de una manera fácil y entendible por el usuario la parte estática y dinámica de cualquier sistema de actividades humanas. Esta herramienta nos permitirá modelar en el sentido de la informática, es decir, describir o representar el problema y luego, describir la solución; estas operaciones se denominan respectivamente, el análisis y el diseño.

Un modelo es una descripción abstracta de un sistema o de un proceso, una representación simplificada que permite comprender y simular. Los modelos no son directamente visibles por los usuarios. Estos son vistos y manipulados por ellos mediante vistas gráficas que reciben el nombre de “diagramas”. Algunos de los modelos para representar sistemas definidos por UML son los siguientes:

- Modelo de clases: captura la estructura estática.
- Modelo de estados: expresa el comportamiento dinámico de los objetos.
- Modelo de casos de uso: describe las necesidades del usuario.
- Modelo de interacción: representa los escenarios y los flujos de mensajes.
- Modelo de realización: muestra las unidades de trabajo.
- Modelo de despliegue: precisa el reparto de procesos.

Los diagramas definidos por UML están basados en estos modelos. Recientemente fue liberada la especificación de este lenguaje de modelado en su versión 2.0 y está disponible para todos los usuarios en la página de la OMG [INT1].

1.5.2.- SGDB:

Un Sistema de Gestión de Bases de Datos (SGDB) es una colección de datos interrelacionados y un conjunto de programas para acceder a dichos datos de forma práctica y eficiente. La colección de datos generalmente se denomina Base de Datos. La gestión implica tanto la definición de estructuras para almacenar la información como la provisión de mecanismos para la manipulación de la misma. Los SGBD son herramientas de apoyo al desarrollo de sistemas programados pensados para manipular grandes cantidades de datos y resolver los siguientes inconvenientes encontrados en los sistemas de procesamiento de archivos convencionales:

- **Redundancia:** se presenta cuando la misma información está duplicada en diferentes lugares (archivos).

- **Inconsistencia:** se presenta cuando los duplicados no coinciden y es consecuencia de la redundancia.

- **Dificultad en el acceso a los datos:** debido a que las operaciones sobre los datos no son generalizadas, ni prácticas, ni eficientes; sino que son específicas para cada desarrollo.

- **Problemas de integridad:** porque no existe una forma generalizada de definir y modificar el dominio de los datos.

- **Problemas de atomicidad:** debido a que resulta difícil asegurar la consistencia de la base de datos en caso de fallos o caídas del sistema.

- **Anomalías en el acceso concurrente:** es complicado supervisar los accesos de tal forma que distintas aplicaciones puedan acceder y modificar los mismos datos al mismo tiempo y aun así, conservar el estado de consistencia de los mismos.

- **Problemas de seguridad:** debido a que no existe una forma generalizada de establecer a qué parte de los datos se permite acceder a cada usuario.

1.5.3.- SQL:

El lenguaje de consulta estructurado o SQL (Structured Query Language). "Lenguaje utilizado para interrogar y procesar datos en una base de datos relacional. Fue desarrollado originalmente por IBM para sus "mainframes", sin embargo se han creado muchas implementaciones para aplicaciones de bases de datos en mini y microcomputadores. Las ordenes (mandatos) de SQL se pueden utilizar para trabajar interactivamente con una base de datos, o pueden incluirse en un lenguaje de programación para servir de interfaz a una base de datos" [DCB1].

La evolución en el tiempo de SQL lo ha convertido en un lenguaje extenso para el desarrollo de programas de aplicación. En consecuencia, ya no es estrictamente necesario combinar SQL con algún lenguaje "anfitrión" distinto para construir aplicaciones

completas. La especificación del lenguaje conocida oficialmente como: Estándar Internacional del Lenguaje de Bases de Datos SQL (1992) supera las 600 páginas y la especificación actual SQL (1999) casi triplica su tamaño. Este lenguaje es considerado actualmente un estándar para el desarrollo de bases de datos relacionales, aunque en su especificación no se menciona en ningún momento, y de hecho, ningún producto en el mercado implementa en forma completa el modelo relacional.

1.5.4.- Servidor de Páginas Web:

Un servidor web es un programa de computación especializado en la administración de páginas web y en la transmisión de los datos de imágenes, animaciones, música y sonidos que éstas incluyen. Permite desarrollar y gestionar sistemas de hospedaje de sitios web. El servidor web reside en un computador conectado en forma permanente a la Internet, atendiendo las solicitudes de páginas web y las tareas de proceso de datos que le hacen los usuarios de la red, las 24 horas del día todos los días del año.

El servidor web mas usado es el Apache del ambiente Unix, seguido por el IIS del sistema Windows 2000. Existen unos 20 millones (60%) de servidores APACHE y unos 6 millones (27%) de servidores IIS, ejecutando en Internet. Se explica el predominio del servidor Apache porque es gratis, muy estable y confiable, tiene una gran capacidad y existen versiones para todas las plataformas de computación o se lo puede adaptar con facilidad.

"Un cliente web es un programa capaz de solicitar servicios de un servidor web. Son buenos ejemplos los navegadores o exploradores de Internet. Entre tantos, el Explorer 5.x y el Communicator 6.x de Netscape. El cliente web reside en el PC del usuario bajo su control. Cuando el usuario digita la dirección de una página web, su cliente web inicia la comunicación con el servidor web a través de Internet, y le solicita la página indicada. Cada servidor en Internet se identifica unívocamente mediante el número IP (Internet Protocol Number). El IP es usado por los protocolos TCP/IP de transmisión de datos. Un ejemplo de IP es 206.101.20.201, son cuatro grupos de dígitos separados por puntos.

Los usuarios le asignan un nombre único a los servidores del tipo www.miservidor.com, porque es más fácil recordar una serie de nombres de dominios que un grupo de números. Se ha creado el sistema DNS (Data Name System) para poder mapear los nombres de los servidores en los números IP respectivos. El objetivo es conciliar el mundo de las personas y el de los computadores. Nosotros usamos nombres y los computadores usan números. El sistema DNS tiene una base de datos distribuida en todo el mundo, y servidores DNS que se encargan de mantener dicha base y de resolver las consultas directas e inversas entre los nombres y los números IP. El DNS es fundamental para que la Internet funcione" [INT2].

1.5.5.- Aplicaciones Empresariales (AE):

"Una aplicación empresarial es una aplicación distribuida que apoya la ejecución de procesos de negocios en una empresa. Los sistemas de información web (SIW) son un tipo particular de aplicación empresarial" [DAE]. La definición anterior involucra las dos características fundamentales de una aplicación empresarial. En primer lugar, establece que son elaboradas con la finalidad de asistir las actividades de una organización o empresa y en segundo lugar, refiere que sus componentes están almacenados, en general, en distintos lugares y están disponibles para ser utilizados (distribuidos). Las aplicaciones empresariales poseen, además de las ya mencionadas, las siguientes propiedades y características:

- **Escalabilidad:** un aumento en la carga de trabajo es manejable sin necesidad de modificar el software.

- **Disponibilidad:** idealmente no deben dejar de prestar servicio.

- **Seguridad:** no todos los usuarios pueden acceder a la misma funcionalidad.

- **Arquitectura multi-capa:** separación clara entre: interfaz (U/S), lógica de negocios y gestión de datos (bases de datos).

- **Acceso a bases de datos:** requieren la interacción con diferentes bases de datos locales o distribuidas.

- **Integrabilidad:** deben proveer los mecanismos necesarios para establecer la comunicación con otros sistemas.

La arquitectura de las AE requiere para su implementación de los siguientes componentes físicos: un servidor de páginas para manejar la interfaz, un servidor de aplicaciones para soportar la lógica de negocios y un servidor de datos que administre las conexiones a bases de datos.

El diseño de AE basado en componentes explota al máximo las ventajas de la reutilización de software y consiste en agrupar de forma lógica los componentes implicados por el sistema. En la capa de presentación, los componentes del lado del cliente y los del lado del servidor. En la capa de lógica de negocios, los componentes de entidades u objetos de negocio y los componentes de procesos de negocios. Por último, la capa de datos está formada por una o mas bases de datos locales o distribuidas.

Un componente de software reutilizable (CSR) "es una unidad de software auto-contenida, modular y reemplazable que:

- Encapsula su implementación.
- Hace visible su funcionalidad a través de un conjunto de interfases.
- Se integra a otros componentes mediante sus interfases para formar un sistema u otro componente mayor.
- Puede ser desplegado (instalado y ejecutado) independientemente de los otros componentes" [DAE].

Las reglas que rigen el desarrollo de componentes de software son establecidas por el modelo de componentes. Existen diversos modelos en el mercado entre los cuales podemos citar: el modelo CORBA de la OMG, el modelo J2EE de Sun, el modelo .NET de Microsoft, el modelo SOA para Web Services de W3C, entre otros. Cada una de las diversas implementaciones de un modelo de componentes específico se conoce con el nombre de infraestructura de despliegue (component framework) y ésta es la responsable de ejecutar y desplegar los componentes desarrollados siguiendo el modelo correspondiente. Como ejemplos podemos mencionar:

- Productos CORBA: ORBIX de IONA, Visibroker de Borland, JavaIDL de Sun.
- Productos J2EE: WebSphere de IBM, WebLogic de BEA, JBoss de JBoss Group.
- Productos .NET: .NET framework de Microsoft.

1.5.6.- Modelo J2EE:

J2EE (Java 2 Platform Enterprise Edition) es un estándar para el desarrollo de aplicaciones empresariales establecido por Sun Microsystems. Contiene una serie de APIs (Interfaz para Programas de Aplicación) cuyas clases y métodos son abstractos. Existen diversas implementaciones algunas de las cuales son código abierto como por ejemplo: JBoss. Una aplicación construida con J2EE no depende de una implementación particular.

J2EE proporciona las siguientes tecnologías:

- JDBC: interfaz para acceso a bases de datos.
- Servlets, Páginas JSP y JSTL: para implementar la interfaz gráfica de una aplicación web.
- EJB: tecnología de Componentes Java.
- XML: interfaz para manejo del lenguaje de etiquetas extensible usado entre otras cosas para integrar aplicaciones heterogéneas debido a que permite expresar datos y no aspecto visual a diferencia de HTML.

Todas estas tecnologías son manejadas por un servidor de aplicaciones quien es el verdadero responsable de prestar los servicios antes mencionados.

En este capítulo se introdujeron algunos conceptos correspondientes al entorno del desarrollo de aplicaciones empresariales. En el capítulo siguiente, se describirá en forma rápida el sistema de actividades que debe apoyar el sistema de información empresarial. Luego, en el capítulo III se abordará el tema de la especificación de requerimientos. El capítulo IV se refiere al diseño de la base de datos. El capítulo V trata acerca de la implementación de la aplicación y el capítulo VI sobre la verificación y validación del sistema.

Capítulo II: Sistema de Actividades

Como se mencionó en capítulos anteriores, todo sistema de información empresarial se desarrolla con la finalidad de apoyar un conjunto de actividades humanas que se ejecutan en una organización o empresa particular. Este capitulo se refiere al estudio del sistema de actividades que se llevan a cabo en el Postgrado de Computación de la Universidad de los Andes para apoyar las actividades de administración académica y de recursos.

No pretendemos realizar un estudio detallado, sino mostrar en forma resumida las tareas que se realizan diariamente y que requieren apoyo computacional. Si el lector desea obtener un estudio detallado de las actividades de administración académicas y de recursos del Postgrado le recomendamos revisar las referencias [Delgado 99] y [Guerrero 99] respectivamente.

Un sistema de actividades es un sistema organizacional que lleva a cabo un conjunto de actividades (funciones, procesos, tareas) que son desarrolladas por un grupo de Actores con la finalidad de cumplir un conjunto de objetivos pre-establecidos [Montilva-96].

El estudio comienza por la determinación de los objetivos principales de la organización y las actividades o tares que se realizan para alcanzar dichos objetivos. Por cada actividad, se identifican los actores y de ser necesario la actividad se descompone en tareas. Seguidamente, se construye un modelo de la estructura funcional (figuras 3 y 4) donde se representan las funciones o procesos que se ejecutan y un modelo de la estructura organizacional (figura 5) de la empresa. A partir de estos dos modelos se construye una tabla donde se representa la relación entre ellos, con el fin de facilitar la especificación de requerimientos.

Por último, se identifican los tipos de entidades involucrados en cada proceso y los eventos que se presentan durante el desarrollo de cada uno de ellos.

Figura 3: Diagrama de la estructura Funcional (Actividades Académicas).

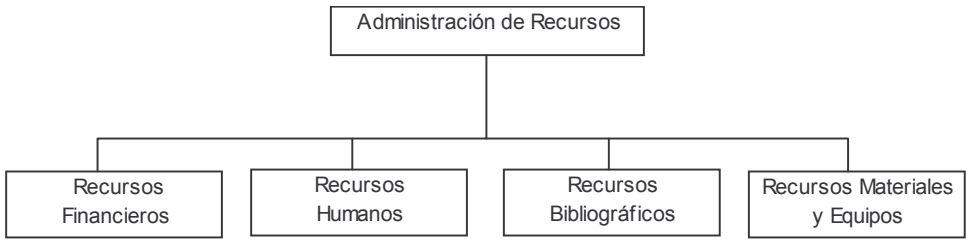


Figura 4: Diagrama de la estructura Funcional (Administración de Recursos).

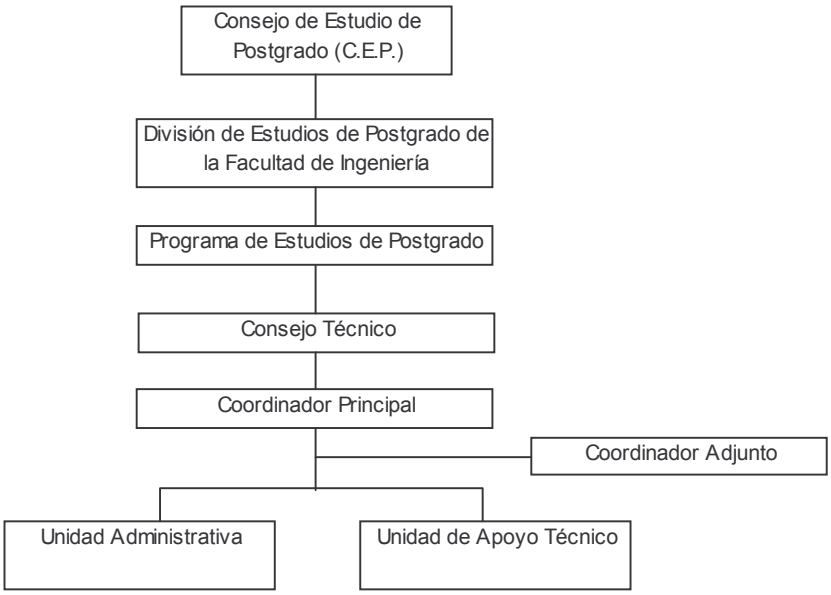


Figura 5: Diagrama de la estructura Organizacional.

2.1.-Identificación del Sistema de Actividades

El sistema de actividades objeto de estudio es el Postgrado de Computación de la Universidad de los Andes. Específicamente, las actividades de administración académica y de recursos.

El Postgrado de Computación, ofrece estudios de Especialización y Maestría en Computación, contribuyendo de esta manera con el fortalecimiento de la investigación y la docencia en esta área a escala nacional e internacional.

2.2.-Objetivos del Sistema de Actividades:

1.- Promover el Programa de Estudios de Postgrado en empresas e instituciones ubicadas dentro y fuera del país.

2.- Planificar el proceso de inscripción para las nuevas cohortes (grupo de estudiantes que ingresan en el mismo semestre y/o año).

3.- Realizar la programación semestral y preparar el proceso de inscripción para los estudiantes de las cohortes existentes y para los de nuevo ingreso.

4.- Controlar y evaluar el desarrollo de las actividades académicas del Postgrado.

5.- Automatizar el control de recursos.

2.3.-Descripción General del Sistema de Actividades

El Programa de Estudios de Postgrado de Computación ofrece dos planes de estudios. El primero de ellos, denominado Programa de Especialización en Computación, está dirigido a la formación de especialistas o expertos en la aplicación de tecnologías de la computación e informática. Su estructura es de cuatro materias obligatorias, tres electivas y un proyecto de grado. Este programa se realiza bajo tres modalidades:

- **Tiempo completo presencial:** Requiere un total de tres semestres consecutivos y se realiza en la sede del Postgrado en Mérida.

- **Medio tiempo presencial:** Requiere un total de cinco semestres consecutivos y se realiza en la sede del Postgrado en Mérida.

- **Medio tiempo a distancia:** Requiere un total de cinco semestres consecutivos y se realiza en la sede de la institución o empresa contratante, la cual debe proveer los recursos y facilidades necesarias para que el programa sea dictado a distancia.

El segundo de estos planes es el denominado, Programa de Maestría en Computación, está orientado a la formación de investigadores y docentes en el área de computación. Su estructura es de cuatro materias obligatorias, cuatro electivas como mínimo, un seminario de carácter obligatorio y un trabajo de grado. Al igual que el programa de especialización, éste se realiza bajo tres modalidades:

- **Tiempo completo presencial:** Requiere un total de cuatro semestres consecutivos y se realiza en la sede del Postgrado en Mérida.

- **Medio tiempo presencial:** Requiere un total de seis semestres consecutivos y se realiza en la sede del Postgrado en Mérida.

- **Medio tiempo a distancia:** Demanda un total de seis semestres consecutivos y se realiza en la sede de la institución o empresa contratante, la cual debe proveer los recursos y facilidades necesarias para que el programa sea dictado a distancia.

La admisión de nuevos estudiantes es realizada por la "Comisión de Ingreso al Programa", la cual está constituida por un Coordinador (Principal o Adjunto) y dos profesores activos del programa. Esta comisión es designada, para cada nueva cohorte o ciclo de ingreso, por el Consejo Técnico.

El proceso de selección de aspirantes se lleva a cabo en dos etapas:

- **Pre-Selección de Aspirantes:** Una vez concluido el lapso de llamado a inscripciones, la Comisión de Ingreso evaluará la solicitud de ingreso de cada aspirante y seleccionará aquellos candidatos que cumplan con todos los requisitos de admisión establecidos.

- **Examen de Admisión:** Una vez que el estudiante ha sido pre-seleccionado, deberá aprobar un examen de conocimientos en diversas áreas básicas de la

computación, las cuales serán establecidas por la Comisión de Ingreso. Este examen se llevará efecto en un plazo no inferior a tres meses, una vez concluido el proceso de pre-selección. Durante este lapso el estudiante podrá seguir el Programa de Nivelación y Actualización, que consta de varios módulos instruccionales o de nivelación, uno por cada área evaluada en el examen y cuyo contenido esta disponible actualmente en la página web del Postgrado. En caso de aprobar el examen, el estudiante será admitido para cursar el Programa que seleccionó.

2.4.-Descripción de Problemas Actuales de Información:

El principal problema que existe en la administración del Postgrado es que toda la información se registra en forma manual, lo que causa ciertos inconvenientes en la ejecución de sus procesos. Algunos de estos inconvenientes son:

1.- Las hojas con los datos básicos de los estudiantes y las notas son archivadas, lo que hace tedioso la búsqueda y ordenamiento de la misma.

2.- Poca velocidad al organizar toda la información de un estudiante, ya que ésta se encuentra dispersa, lo que dificulta su manipulación.

3.- No se lleva un expediente completo de los profesores pertenecientes al Postgrado. Es decir, se guardan sus datos básicos, pero no se tiene información sobre los eventos a los que ha asistido, los libros que ha publicado, etc.

4.- El registro de los materiales y equipos del postgrado se realiza de forma manual, lo que genera problemas de búsqueda y actualización.

5.- No se tienen registrados los recursos bibliográficos del Postgrado, así que no existe un sistema de préstamo de ese material.

6.- La información sobre los presupuestos es difícil de manejar, debido a que se encuentra dispersa entre los diferentes entes que otorgan los presupuestos y éstos a su vez, tardan en enviar información acerca del mismo, es decir, no se sabe con cuanto se cuenta ni cuanto se ha gastado.

Existen muchos mas problemas de esta naturaleza, pero los anteriores son quizás los más prioritarios. Dichos problemas demandan una solución que acelere el registro, la búsqueda y actualización de la información pertinente a la administración académica y de recursos del Postgrado.

Una vez conocido el sistema de actividades al que dará apoyo la aplicación empresarial podemos concentrarnos en lo referente a la especificación de requerimientos. Este tema se abordará en el capítulo siguiente.

Capítulo III: Análisis y Especificación de Requerimientos

Los requerimientos son las necesidades que el sistema debe satisfacer a partir de la información almacenada en la base de datos. Estos pueden ser de diferentes tipos:

3.1.- Requerimientos de Información: Se refieren a los servicios que el Sistema de Información debe proveer a los usuarios. Se deben establecer cuáles serán las consultas más frecuentes a la base de datos y los reportes más importantes que debe presentar el sistema, ya sea por pantalla o en papel.

3.2.- Requerimientos de Almacenamiento: Se describen las tecnologías de almacenamiento que van a ser empleadas, los tipos de entidades del Sistema de Actividades que se representarán en la base de datos, además de los atributos de cada uno de estos tipos de entidades.

3.3.- Requerimientos de Interacción: Definen como será la interacción entre el usuario y el sistema. Estos se dividen en:

3.3.1.-Requerimientos de Entrada de Datos: Se identifican las fuentes y los medios que se emplearán para capturar los datos de entrada. Los datos pueden provenir de distintos tipos de fuentes como por ejemplo: documentos existentes, observación directa, digitalización de imágenes, audio, video o archivos de bases de datos ya existentes, etc.; dependiendo de la fuente de donde provenga el dato, dependerá el medio que se empleará para su captura, por ejemplo, si proviene de una observación directa, éste será introducido a través del teclado. También debe realizarse el diseño de las formas o pantallas que se utilizarán para la introducción de los datos.

3.3.2.-Requerimientos de Interfaz o Diálogo Usuario/Sistema: Se especifica el tipo de interfaz que deberá utilizarse para el manejo del sistema, es decir, debe decidirse si la interfaz será gráfica, por comandos o por menú.

3.3.3.-Requerimientos de Salida de Información: Se determinan las características y el contenido de la información que deberá producir el sistema, además del formato de salida, en el que se deberá presentar dicha información al usuario, el cual puede ser a través de gráficos de barra, histogramas, gráficos circulares, tablas estadísticas, listados, reportes, páginas web, etc.

3.4.- Requerimientos de Atributos de Calidad: Se especifican un conjunto de factores o condiciones que determinarán la calidad del sistema, tales como: confiabilidad, eficiencia, amigabilidad, transportabilidad, etc.

3.5.- Requerimientos de Restricciones: Se establecen el conjunto de restricciones o limitaciones que pueden afectar el desarrollo y la puesta en operación del sistema. Esas restricciones dependerán de la disponibilidad de recursos económicos, recursos en equipos de computación, recursos en cuanto a sistemas programados, recursos humanos, etc.

3.1.- Requerimientos de Información:

Los requerimientos de información se determinan a partir de los modelos funcional y organizativo del sistema de actividades, elaborados en la Fase I. Para cada uno de los procesos se identifica y especifica la información que los usuarios involucrados requieren para ejecutar cada una de las actividades dentro del mismo. Esto se refleja a través de una matriz que relaciona los requerimientos de información establecidos con los procesos.

Un diagrama de clases describe los tipos de objetos involucrados en el sistema, sus características y las relaciones que existen entre ellos. Los elementos de modelado que se utilizan en la representación de los diagramas de clases son los siguientes:

3.1.1.- Clase: La clase es una agrupación de objetos del mundo real que poseen características y comportamiento comunes. Es el elemento fundamental del diagrama de

clases. Se representada mediante un rectángulo con tres divisiones internas: la primera contiene el nombre de clases, la segunda la estructura y la tercera el comportamiento. También se puede representar por un rectángulo que contenga solamente el nombre.

3.1.2.- Atributo: Identifica las características o propiedades de cada clase. La sintaxis para un atributo es la siguiente:

Visibilidad NombreAtributo: TipoAtributo = Valor Inicial

Donde Visibilidad es: + → Pública
→ Protegida
- → Privada

3.1.3- Operación: Describe el comportamiento de los objetos de una clase. La sintaxis para un método es la siguiente:

Visibilidad Nombre Método (Parámetros): ExpresiónRetorno

Donde Visibilidad es: + → Pública
→ Protegida
- → Privada

3.1.4.- Asociaciones: Representan los parentescos o relaciones entre instancias u objetos de clases. Algunos tipos de asociaciones son los siguientes:

3.1.4.1.- Asociación Binaria: Es una asociación sencilla entre dos clases, la cual es representada como una línea sólida que une a las dos clases.

3.1.4.2.- Composición: Es una forma de asociación más fuerte que tiene las características siguientes:

- Pertenencia fuerte: Los objetos componentes son una parte constitutiva y vital del objeto compuesto.
- Dependencia existencial: Coincidencia en el ciclo de vida del objeto compuesto y de los objetos componentes, es decir, una vez creados, éstos viven y mueren con el objeto compuesto (comparten su ciclo de vida). Estos componentes también pueden ser removidos antes de la muerte del compuesto.

Existe una relación menos fuerte que se representa casi igual a la composición, y que es llamada "Agregación".

La composición se denota dibujando un rombo relleno del lado de la clase compuesta y una línea recta no dirigida (sin flecha) que la conecta con la clase componente.

3.1.4.3.- Generalización: Es una relación de herencia de una clase a otra, donde la clase más general es denominada "Superclase", y la clase específica que hereda los atributos y operaciones de la superclase, es denominada "Subclase". Se representa dibujando un triángulo sin rellenar en el lado de la superclase.

Cada asociación tiene dos roles, donde cada rol posee las características siguientes:

- Es un nombre al final de la línea de asociación, que describe la semántica (significado) de la relación en el sentido indicado.
- Puede tener multiplicidad, lo que indica que muchos objetos pueden participar en un parentesco. Las más comunes son:

  1 → Uno

  0...1 → Cero o Uno

  * → Indica un rango de 0...infinito

3.1.5.-Instancia: Es un individuo u ocurrencia de algún tipo determinado.

3.1.6.- Objeto: Un objeto es una instancia de una clase.

El diagrama de clases del sistema se puede consultar en el Apéndice A (figuras 1 y 2). En la figura 1 se muestra el diagrama de clases del sistema de administración académica y en la figura 2 el diagrama de clases del sistema de administración de recursos.

3.3.2.- Requerimientos de Entrada de Datos:

La mayoría de los datos que alimentarán el Sistema de Información provendrán de documentos ya existentes como: expedientes de los estudiantes y profesores, planillas de notas de las materias, listados de los estudiantes pertenecientes a cohortes pasadas, etc. y de documentos que se irán generando en la realización de los procesos dentro de la organización, como: planillas de inscripción de los estudiantes que ingresan a las nuevas cohortes, planillas con las notas de materias terminadas en algún momento, etc. Mientras que los datos restantes, serán generados o calculados, como: promedios de notas de los estudiantes, número de estudiantes pertenecientes a una cohorte, etc.

Todos los datos se capturarán a través de formas o ventanas, que permitirán a los usuarios transcribir los datos por medio del teclado.

3.3.3.- Requerimientos de Salida de Información:

La mayor parte de la información será representada a través de la pantalla, ofreciendo al usuario la posibilidad de obtenerla en forma impresa. También se generarán reportes impresos, en forma tabular. A continuación se presenta una lista de los listados, reportes y consultas, que serán satisfechas por el Sistema:

Listados más importantes:

- Materias vistas por la pasada cohorte.
- Materias que han sido cursadas y las que son cursadas por la cohorte actual.
- Instituciones visitadas por la pasada cohorte.

- Material de Promoción por Tipo: Afiche, Tríptico o folleto.
- Aspirantes que solicitan ingreso al Programa de Estudios.
- Aspirantes pre-seleccionados.
- Aspirantes seleccionados, una vez que han aprobado el examen de admisión.
- Aspirantes seleccionados, junto con las notas del examen de admisión.
- Materias en general.
- Materias vistas por cada cohorte en el último semestre.
- Estudiantes inscritos en alguno de los programas de Estudios.
- Estudiantes inscritos indicando su promedio de notas.
- Estudiantes que estén en realización de tesis.
- Egresados indicando su condición: graduado o retirado.
- Planta profesoral del postgrado.
- Materias que pertenecen a cada Plan de Estudios.
- Promedios de Notas por Materia para cada Cohorte.
- Promedios de Notas por Estudiante por Semestre y por Materia.

Reportes:

- Informe de cada aspirante que solicita ingreso.
- Expediente de cada seleccionado.
- Directorio de profesores.
- Informe para cada inscrito con las materias a cursar en este semestre, junto con sus prelaciones y las notas de estas últimas.
- Informe para cada inscrito con las materias vistas durante sus estudios.
- Listado general de estudiantes pertenecientes a una materia.
- Expediente de notas por cada estudiante.
- Directorio de Egresados.
- Informe General para un Profesor particular.
- Informe sobre Recursos (Laboratorios, Bibliotecas, Financieros, etc.).

Consultas:

- Contenido de alguna materia en particular.
- Planes de Estudios: Maestría y Especialización.
- Aspirante que solicita su ingreso.
- Aspirante seleccionado.
- Notas del Examen para un seleccionado particular.
- Materias que cursa un estudiante en el semestre actual.
- Promedio de notas de un estudiante particular.
- Estudiante que esté realizando proyecto de grado, indicando el Tutor.
- Carga horaria de un profesor.
- Horario de las materias dictadas durante el semestre inmediato pasado.
- Profesor que dicta una materia determinada.
- Datos de un Profesor a través de su cédula o su nombre.
- Disponibilidad de profesores.

3.6.-Diagramas de Casos de Uso:

Un diagrama de casos de uso muestra las distintas operaciones que se esperan de una aplicación o sistema y la relación que éstas tienen con su entorno. Los elementos de modelado utilizados para la representación de los casos de uso son los siguientes:

1.- Caso de Uso: Refleja la interacción entre un usuario y un sistema computarizado. Cada caso de uso es una operación completa desarrollada por los actores y por el sistema en un diálogo. Se representa a través de una elipse y denota un requerimiento solucionado por el sistema.

2.- Actor: Es un usuario del sistema, quien necesita o utiliza algunos de los casos de uso.

3.- Relaciones:

3.1.- Comunica (Communicates): Es una relación entre un actor y un caso de uso. Denota la participación del actor en el caso de uso determinado.

3.2.- Usa (Uses): Es una relación entre dos casos de uso. Indica la inclusión del comportamiento de un caso de uso en otro.

3.3.- Extiende (Extends): Es una relación entre dos casos de uso. Denota una especialización de un caso de uso.

El propósito de los diagramas de casos de uso es reunir los requerimientos de los usuarios y representarlos en una forma compacta y fácil de entender. A manera de ejemplo se muestran los diagramas de casos de uso para dos de los procesos de la administración del Postgrado en las figuras 6 y 7. Se implementarán los casos de uso estudiados en las tesis anteriores [Delgado99] y [Guerrero99]. Para información más detallada acerca de estos casos de uso se le sugiere al lector consultar estas referencias.

3.7.-Atributos de Calidad del Sistema:

El sistema tendrá una interfaz Usuario/Sistema amigable. Esto debido a la utilización de páginas web, las cuales hoy en día, son muy familiares para los usuarios y su manipulación es altamente gráfica e intuitiva, por lo que se requiere poco o ningún entrenamiento para su uso.

El sistema será eficiente y eficaz debido a que hará lo que se supone que debe hacer en un tiempo aceptable.

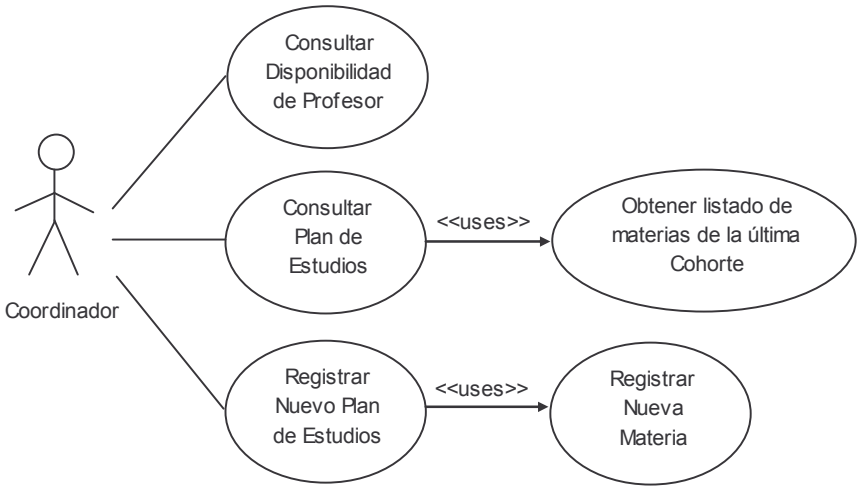


Figura 6: Diagrama de casos de uso para el proceso de "Planificación de Cohorte"

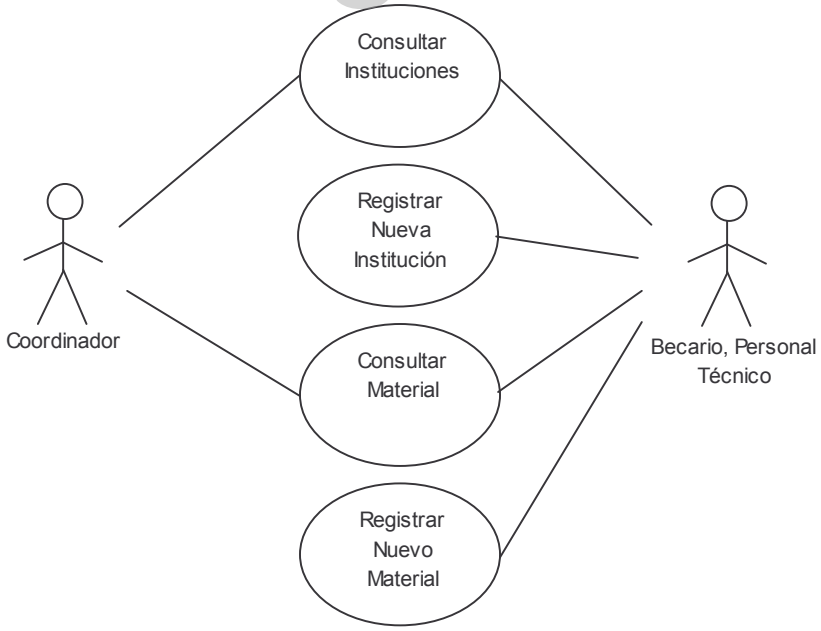


Figura 7: Diagrama de casos de uso para el proceso de "Promoción del Programa"

El sistema será portable y confiable. Estará basado en tecnología estandarizada y probada.

El sistema será fácil de modificar, mantener y será reutilizable. Se usarán componentes distribuidos para su desarrollo y una arquitectura multi-capa.

Una vez, establecidos los requerimientos del sistema. Entramos en la etapa de diseño conceptual de la base de datos. Este tema se aborda en el capítulo siguiente.

Capítulo IV: Diseño de la Base de Datos

4.1.- Transformación del Diagrama de Clases

En el diagrama de clases obtenido en la especificación de requerimientos se observa la estructura estática del sistema, compuesta por un conjunto de clases relacionadas. Dado que se dispone de un manejador de base de datos relacional para la implementación de dicho diagrama; y no de un manejador de base de datos orientado a objeto; es necesario transformar el modelo orientado a objetos (diagrama de clases) (figuras 1 y 2 del Apéndice A) al modelo relacional. Para tal fin, seguiremos los siguientes pasos:

- Para cada clase se debe seleccionar o agregar un atributo que la identificará de forma unívoca (clave primaria). Por ejemplo en el caso de la clase "Persona" se escogió el atributo "Cédula" porque cumple con las propiedades de unicidad e irreductibilidad.

- Cada clase se transforma en un esquema de relación que contiene inicialmente sólo los atributos monovaluados. Por ejemplo, en el caso del esquema correspondiente a la clase "ExpedienteE" se omitieron los atributos multivaluados como: universidades, empresas, proyectos, etc.

- En los casos en que se presente composición de clases, al esquema correspondiente a la clase componente se le agrega el atributo seleccionado como clave primaria del esquema de la clase compuesta. Por ejemplo, al esquema "Materia" se le agregó el atributo "CodigoPE", clave primaria del esquema "PlanEstudio".

- Cada clase derivada (subclase) hereda los atributos de la superclase. Por lo tanto, sólo es necesario agregar al esquema de la subclase el atributo que resulte elegido como clave primaria del esquema de la superclase. Por ejemplo, al esquema de la

clase “Profesor” se le agregó el atributo “Cédula”, clave primaria del esquema “Persona”.

- Si la cardinalidad de la asociación entre clases es:

  Uno a uno (1:1) se debe agregar a cualquiera de los dos esquemas el atributo clave del otro. Por ejemplo, en el caso del esquema de la clase “ExpedienteE” se agregó el atributo “Cédula”, clave primaria del esquema de la clase “Estudiante”. Además, se seleccionó como clave primaria del esquema “ExpedienteE” el atributo “Cédula” porque se consideró sin sentido consultar un expediente usando como parámetro un atributo distinto a la cédula del estudiante.

  Uno a muchos (1:M) se agrega al esquema correspondiente a la clase con asociación múltiple la clave primaria del esquema representante de la clase con asociación unitaria. Por ejemplo, al esquema “Reconocimiento” se agregó el atributo “Cédula”, clave primaria del esquema “Profesor”.

  Muchos a muchos (N:M) se debe crear un nuevo esquema de relación cuya clave primaria es la concatenación de las claves primarias de los esquemas de relación referenciados por la asociación. Por ejemplo, se creó el esquema “PartEvento” y se seleccionó como clave primaria de este esquema los atributos: “Cédula” y “CodEvento”, claves primarias de los esquemas: “Profesor” y “Evento”, respectivamente.

Una vez obtenidos los esquemas resultantes de la aplicación de los pasos descritos anteriormente se procede a normalizar. La normalización es un proceso que permite estructurar la base de datos de tal forma que las actualizaciones de las tuplas (filas de una tabla en SQL) individuales sean más aceptables lógicamente de lo que serían de otro modo, es decir, si el diseño no estuviese totalmente normalizado. Este proceso descansa sobre la base del modelo relacional y está íntimamente ligado al concepto de dependencia funcional.

4.2.- Dependencias Funcionales

De manera intuitiva, una dependencia funcional (DF) es un vínculo muchos a uno que asocia un conjunto de atributos de un esquema de relación con otro conjunto de atributos del mismo esquema. Una definición más formal de dependencia funcional sería la siguiente: Sea R una relación y sean X e Y subconjuntos arbitrarios de atributos de R. Entonces decimos que X->Y "X determina funcionalmente a Y" si y sólo si, cada valor de X en R está asociado exactamente con un valor de Y en R. Lo que implica que siempre que dos tuplas coincidan en su valor X, también coincidirán en su valor Y [ISBD].

Una dependencia funcional (DF) es trivial si y sólo si, la parte derecha (dependiente) es un subconjunto no necesariamente propio de la parte izquierda (determinante). Al conjunto de todas las dependencias funcionales implicadas por un conjunto dado "S" de dependencias funcionales, se le llama cierre de "S" y se escribe "S+". Un conjunto "S" de dependencias funcionales es irreducible si y sólo si, satisface las siguientes propiedades:

1. La parte derecha (dependiente) es un conjunto individual, es decir, formado por sólo un atributo.

2. La parte izquierda (determinante) posee el menor número de atributos posible.

3. No es posible descartar de "S" ninguna dependencia funcional sin cambiar el cierre "S+", es decir, sin convertir "S" en algún conjunto no equivalente a "S".

Un conjunto de dependencias funcionales es irreducible a la izquierda si cumple la segunda propiedad (2) de las mencionadas anteriormente.

4.3.- Formas Normales

- **Primera forma normal (1FN)**

Un esquema de relación está en primera forma normal (1FN) si y sólo si, en cada valor válido de ese esquema, toda tupla contiene exactamente un valor para cada atributo. En otras palabras, un esquema 1FN no posee atributos multivaluados [ISBD].

- **Segunda forma normal (2FN)**

Definición que toma sólo una clave candidata, la cual suponemos que es la clave primaria. Un esquema de relación está en segunda forma normal (2FN) si y sólo si, está en primera forma normal (1FN) y todo atributo que no sea clave es dependiente irreduciblemente de la clave primaria. Dicho de otro modo, un esquema 2FN es un esquema 1FN y además, ningún atributo depende funcionalmente de un subconjunto de la clave primaria [ISBD].

- **Tercera forma normal (3FN)**

Definición que toma sólo una clave candidata, la cual suponemos que es la clave primaria. Un esquema de relación está en tercera forma normal (3FN) si y sólo si, está en segunda forma normal (2FN) y todos los atributos que no son clave son dependientes en forma no transitiva de la clave primaria. Es decir, un esquema 3FN es un esquema 2FN y además, no existen dependencias funcionales que involucren atributos no clave [ISBD].

- **Forma normal de Boyce/Codd (FNBC)**

Un esquema de relación está en forma normal de Boyce/Codd (FNBC) si y sólo si, toda dependencia funcional no trivial, irreducible a la izquierda, tiene una clave candidata como su determinante. Es decir, un esquema está en FNBC si y sólo si, los únicos determinantes son claves candidatas [ISBD].

El resultado de la aplicación del proceso de normalización originó el conjunto de esquemas que se presentan en el programa script de la listado 1 Apéndice B y las tablas donde se documentan los atributos de cada tabla en detalle se pueden consultar en el Apéndice C.

Obtenido el modelo conceptual, el modelo relacional y la documentación de los atributos podemos concluir la etapa del diseño de la base de datos y concentrarnos en la implementación del sistema. Este tema se abordará en el capítulo siguiente.

Capítulo V: Implementación del Sistema

El sistema de información empresarial del Postgrado se desarrollará usando arquitectura de cuatro capas basada en el modelo J2EE (Java 2 Platform Enterprise Edition) que es un estándar para la construcción de aplicaciones empresariales desarrollado por Sun Microsystems. Este modelo es altamente usado en el mercado y está basado en tecnología Java por lo que se requirió la descarga e instalación de diversos productos. Estos productos son los siguientes: JSDK (Java Standard Developer Kit) en su versión 1.4.2, Eclipse IDE en su versión 3.0 y JBoss en su versión 3.2.3.

El JSDK incluye el compilador y el entorno de tiempo de ejecución JRE (Java Runtime Environment) entre otros. Todo producto basado en tecnología Java requiere que esté instalado el JRE y el compilador. El Eclipse IDE es el entorno de desarrollo integrado utilizado como herramienta para la implementación del sistema. Por último, el JBoss es el servidor de aplicaciones. Incluye el contenedor y el servidor de EJB (Enterprise JavaBeans), además del Servidor de páginas web. El servidor de aplicaciones es el responsable de proporcionar los servicios de sistema de bajo nivel a los contenedores y servidores mencionados anteriormente y de mantener en funcionamiento la infraestructura de comunicaciones que hace posible el desarrollo de aplicaciones distribuidas.

Una aplicación empresarial es aquella que:

- Utilizan concurrentemente más que unos pocos usuarios.
- Utilizan recursos distribuidos, como por ejemplo, sistemas de bases de datos.
- La funcionalidad es implementada a partir de la interacción entre distintos objetos que se encuentran dispersos en la infraestructura de la empresa.
- Utiliza una arquitectura de servicios web y tecnología J2EE para enlazar los componentes, es decir, los objetos.

Una aplicación empresarial es vital para el funcionamiento de la organización. Por tal razón, los usuarios del sistema tienes grandes expectativas acerca de la aplicación. Desean entre otras cosas que la aplicación empresarial:

- Esté disponible 24 horas al día, 7 días a la semana, sin ninguna caída.
- Tenga un tiempo de respuesta aceptable incluso si aumenta su uso.
- Se adapte a las necesidades cambiantes de la empresa. Por lo cual, debe ser fácil de modificar para no tener que rediseñarla.
- Sea independiente del proveedor.
- Pueda interactuar con los sistemas existentes.
- Utilice los componentes (es decir, objetos) existentes.

Esto significa que se debe crear una aplicación empresarial que esté disponible y sea escalable de forma que se pueda ajustar a las variaciones en la demanda sin tener que modificar el código. La aplicación debe ser extensible y mantenible de forma que sea fácil agregarle nuevas reglas de negocio. Además, la aplicación debe ser portable para garantizar que la empresa no quede atada a un proveedor específico. También se debe asegurar la interoperabilidad para que la aplicación empresarial pueda interactuar con otras aplicaciones y pueda reutilizar código existente.

5.1.- Implementación de la Capa de Datos o de Sistemas de Información Empresarial:

A partir del diseño conceptual y del modelo relacional de la base de datos obtenido en el capítulo anterior, se construye el programa (script) que implementará este diseño. Cada elemento del modelo relacional representa una tabla de la base de datos que se desarrollará y cada uno de los campos que se mencionaron en el diseño conceptual, pasan a ser los atributos de la tabla. El tipo de dato de cada atributo y su correspondiente dominio de valores se encuentran documentados en el Apéndice C.

El sistema de gestión de bases de datos elegido es Postgresql, por razones de disponibilidad y de libre uso. Este es un sistema manejador de bases de datos relacionales, por lo cual, fue necesario transformar el diagrama de clases del modelo orientado a objetos en tablas del modelo relacional, como se mencionó en el párrafo anterior. Luego, a través del comando "createdb PgcompDB" se crea la base de datos usando la ventana del interprete de comandos de Postgresql y a continuación se carga el script a través de la orden "psql -e PgcompDB < postgradobd.dat", donde postgradobd.dat es el nombre del script que contiene los comandos de creación de tablas y de definición de claves primarias y externas de la base de datos del Postgrado. Para mayores detalles acerca del script utilizado se le sugiere al lector consultar el Apéndice B.

A continuación se presentan fragmentos de código del script con la finalidad de mostrar los detalles de creación la tabla “Persona” y la tabla “Profesor”.


```
/* 1 */
create table Persona ( Cedula text not null, Nombre text, Apellido text, EdoCivil char, Sexo char, Nacionalidad char, Direccion text, Telefono text, FechaN date, LugarN text, Email text, DirTrabajo text, TlfTrabajo text, Estado text, Pais text, primary key ( Cedula ) );
/* 2 */
create table Profesor ( Cedula text, Dedicacion text, Condicion text, Categoria text, FechaIngUla date, FechaIngPostgrado date, HoraSemUla int, HoraSemPost int, primary key ( Cedula ), foreign key ( Cedula ) references Persona );
```

Listado 1: Detalles de creación de las tablas “Persona” y “Profesor”.

En el listado anterior se observa la forma de utilizar el comando “create table” para crear una nueva tabla en la base de datos y los comandos “primary key” y “foreign key” para definir la o las claves primarias y externas, respectivamente. La orden “not null” le indica al SGDB (Sistema de Gestión de Bases de Datos) que no acepte el ingreso a la tabla de filas que no tengan un valor definido para el campo “Cedula”. Esto no es necesario, en este caso debido a que más abajo se define al atributo “Cedula” como clave primaria de la

tabla “Persona” y por defecto, el SGDB aplica a las claves: primarias, secundarias o externas; la política de no permitir valores nulos.

En la siguiente orden se le indica al SGDB que cree la tabla “Profesor”. Nótese que, como la clase “Profesor” es una subclase de la clase “Persona” no se debe crear un profesor si antes no se crea la persona que corresponde a ese profesor particular. Esto se consigue, indicándole al SGDB que el atributo “Cedula” que es clave primaria de la tabla “Persona” va a ser la clave primaria de la tabla “Profesor” a través del segmento: “primary key (Cedula), foreign key (Cedula) references Persona);” de la orden “create table Profesor…”.

5.2.- Implementación de la Capa de Lógica de Negocios

La arquitectura J2EE está formada por componentes que en conjunto permiten la construcción de aplicaciones distribuidas. Los componentes se distribuyen en capas según su funcionalidad.

Se puede afirmar, sin lugar a dudas, que la capa de lógica de negocios es el corazón de toda aplicación empresarial. En ella se alojan los componentes encargados de implementar las reglas de negocio que brindan el apoyo computacional que requiere la empresa u organización. Esta es la capa de los Enterprise JavaBeans (EJB).

“Un Enterprise JavaBean (EJB) es un componente de la arquitectura J2EE cuya función principal es proporcionar la lógica de negocio de una aplicación J2EE e interactuar con otros componentes J2EE del servidor” [J2EEMR]”. Existen tres tipos de EJB: los de entidad (Entity Beans), los de sesión (Session Beans) y los beans orientados a mensajes (Message-Driven Beans). Un bean de entidad se utiliza para representar los datos del negocio, un bean de sesión se utiliza para modelar un proceso de negocio y un bean orientado a mensajes se utiliza para manejar mensajes del servicio de mensajería de java JMS (Java Messaging Service).

Interfaces de los Enterprise JavaBeans

Los beans de sesión y de entidad deben tener dos interfaces: la interfaz home y la interfaz remota. Se deben desarrollar las dos interfaces para que sean implementadas por el contenedor. El EJB contiene la lógica de negocio necesaria para dar servicio a un cliente. El contenedor envuelve al EJB y gestiona sus requerimientos de ciclo de vida, además de proporcionarle comunicación con su entorno a través de las interfaces home y remota.

Los clientes obtienen referencias a la interfaz remota a través de la interfaz home, la cual actúa como factoría de objetos, para luego utilizar la interfaz remota y por medio de ella invocar los métodos de negocio del bean. El EJB se puede acceder de forma remota desde una aplicación que utilice los estándares RMI e IIOP, incluyendo aplicaciones CORBA y programas no escritos en Java. La interfaz remota representa la vista de los clientes del bean. A continuación se presentan fragmentos de código correspondientes a las interfaces del EJB de entidad "Persona" y del EJB de sesión "AccesoBean".


```
package interfaces;
import java.rmi.RemoteException;
import java.util.Collection;
import java.sql.Date;
import javax.ejb.CreateException;
import javax.ejb.EJBHome;
import javax.ejb.FinderException;

public interface PersonaHome extends EJBHome {

   public PersonaRemote create() throws RemoteException, CreateException;
   public PersonaRemote create(...parámetros…) throws RemoteException,
CreateException;
   public PersonaRemote findByPrimaryKey(String key) throws RemoteException,
FinderException;
   public Collection findAll() throws RemoteException, FinderException;

}
```

Listado 2: Fragmento de código de la interfaz home del EJB de entidad “Persona”

El archivo tiene por nombre “PersonaHome.java”, debido a que el compilador exige que toda clase o interface tenga el mismo nombre del el archivo que la contiene. La sentencia “package interfaces;” indica que la interface que se desarrolla pertenece al paquete con nombre “interfaces”. Las sentencias “import” incluyen las clases donde se desarrollan los tipos de datos utilizados en los parámetros de entrada y salida de los métodos declarados en la interface. Por ejemplo, la sentencia: “import java.sql.Date” incluye la clase Date que pertenece al paquete sql del paquete java. Toda interfaz home debe extender a la clase “EJBHome” o “EJBLocalHome”y debe declarar como mínimo un método “create” que regrese una referencia a un objeto de la interfaz remota en este caso: “PersonaRemote”. El método “findByPrimaryKey” es obligatorio para los EJB de entidad.

De igual forma se presenta a continuación en el listado 2, fragmentos del código que declara la interfaz remota de un EJB de entidad. En este caso la interfaz remota del EJB de entidad con nombre “Persona”.


```
package interfaces;

import java.rmi.RemoteException;
import java.sql.Date;
import javax.ejb.EJBObject;

public interface PersonaRemote extends EJBObject {

        public void setCedula( String Cedula ) throws RemoteException;
        public String getCedula() throws RemoteException;
        public void setNombre( String Nombre ) throws RemoteException;
        public String getNombre() throws RemoteException;
…
}
```

Listado 3: Fragmento de código de la interfaz remota del EJB de entidad “Persona”

Al igual que en el caso anterior, la interface pertenece al paquete “interfaces” y tiene por nombre en este caso: “PersonaRemote”. Toda interfaz remota debe extender la clase “EJBObject” o “EJBLocalObject”. Luego se observan los métodos que asignan y regresan los valores de los atributos del bean. Los métodos de las interfaces “PersonaHome” y “PersonaRemote” serán visibles a los clientes del bean “Persona”.

Ahora se presentan fragmentos de código de las interfaces home y remota de un bean de sesión, en este caso el bean de sesión “AccesoBean”.

```
package interfaces;

import java.rmi.RemoteException;
import javax.ejb.CreateException;
import javax.ejb.EJBHome;

public interface AccesoBeanHome extends EJBHome {

        AccesoBeanRemote create() throws RemoteException, CreateException;

}
```


Listado 4: Interfaz home del bean de sesión “AccesoBean”.

```
package interfaces;
import javax.ejb.EJBObject;
import java.rmi.RemoteException;

public interface AccesoBeanRemote extends EJBObject {

        public String getTipo() throws RemoteException;
        public boolean darAcceso(String nombre, String clave) throws RemoteException;
        public boolean getEstado() throws RemoteException;

}
```

Listado 5: Interfaz remota del bean de sesión “AccesoBean”.

Como se observa en el listado 4, prácticamente no hay diferencia entre las interfaces home de un bean de entidad y de sesión. La única diferencia apreciable es que los beans de entidad están obligados a definir el método “findByPrimaryKey“. En el caso de las interfaces remotas no hay ninguna diferencia, debido a que en ellas descansan los métodos de negocio de cualquier Enterprise JavaBean.

El descriptor de despliegue

Un descriptor de despliegue es una archivo con sintaxis XML que describe la forma en los EJB serán gestionados durante su ejecución y permite configurar el comportamiento de Enterprise JavaBean sin modificar el código; por ejemplo, permiten describir los atributos del contexto transaccional durante la ejecución. El comportamiento del EJB se puede alterar sin tener que modificar ni su clase ni sus interfaces. A continuación en el listado 6, se presentan fragmentos de código del descriptor de despliegue de los Enterprise JavaBeans que forman la aplicación empresarial del Postgrado.


```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ejb-jar PUBLIC "-//Sun Microsystems, Inc.//DTD Enterprise JavaBeans 2.0//EN" "http://java.sun.com/dtd/ejb-jar_2_0.dtd">
<ejb-jar>
  <description> Aplicación J2EE Postgrado de Computación </description>
  <display-name> Aplicación J2EE Postgrado de Computación </display-name>
  <enterprise-beans>
    <entity>
      <ejb-name>PersonaEJB</ejb-name>
      <home>interfaces.PersonaHome</home>
      <remote>interfaces.PersonaRemote</remote>
      <ejb-class>ejb.Persona</ejb-class>
      <persistence-type>Bean</persistence-type>
      <prim-key-class>java.lang.String</prim-key-class>
      <reentrant>false</reentrant>
      <session-type>Stateless</session-type>
      <transaction-type>Container</transaction-type>
      <env-entry>
        <env-entry-name>DbUrl</env-entry-name>
          <env-entry-type>java.lang.String</env-entry-type>
```

```
        <env-entry-value>jdbc:postgresql:PgcompDB</env-entry-value>
        </env-entry>
        <env-entry>
        <env-entry-name>User</env-entry-name>
        <env-entry-type>java.lang.String</env-entry-type>
        <env-entry-value>postgres</env-entry-value>
        </env-entry>
        <env-entry>
        <env-entry-name>Clave</env-entry-name>
        <env-entry-type>java.lang.String</env-entry-type>
        <env-entry-value>postgres</env-entry-value>
        </env-entry>
        <env-entry>
        <env-entry-name>DriverName</env-entry-name>
        <env-entry-type>java.lang.String</env-entry-type>
        <env-entry-value>org.postgresql.Driver</env-entry-value>
    </env-entry>
  </entity>
        ...
  <session>
                <ejb-name>AccesoEJB</ejb-name>
                <home>interfaces.AccesoBeanHome</home>
                <remote>interfaces.AccesoBeanRemote</remote>
                <ejb-class>ejb.AccesoBean</ejb-class>
                <session-type>Stateful</session-type>
                <transaction-type>Bean</transaction-type>
  </session>
        ...
</enterprise-beans>
</ejb-jar>
```

Listado 6: Fragmentos de código del descriptor de despliegue de los Enterprise JavaBeans.

Como se observa en el listado anterior el descriptor de despliegue es un archivo con sintaxis XML. Comienza por definir la versión de xml que se usa, el tipo de documento y los archivos que implementan las respectivas especificaciones. Luego se escriben las etiquetas usadas para el despliegue de los Enterprise JavaBean dentro de las etiquetas <ejb-jar> y </ejb-jar> que marcan el inicio y fin del documento.

Los dos pares de etiquetas siguientes son descriptivas de la aplicación y no son obligatorias. Los EJB se describen dentro de las etiquetas <enterprise-beans> y </enterprise-beans>. A este nivel, ya podemos diferenciar los tipos de beans mediante el uso de las etiquetas: <entity> y </entity> para beans de entidad, <session> y </session> para beans de sesión y <message-driven> y </message-driven> para beans orientados a mensajes.

Las etiquetas <ejb-name></ejb-name> rodean el nombre con el que se desplegará el EJB y es a través de este nombre que será visible a los clientes. Los siguientes dos pares de etiquetas contienen la ruta donde se encuentran las interfaces home y remota de EJB. Luego se establece que la persistencia del bean será gestionada por el bean, esto quiere decir que en la clase del bean, en este caso, "Persona" se debe escribir el código SQL que administre la persistencia del bean. Entre las etiquetas <prim-key-class> y </prim-key-class> se escribe el tipo de la clave primaria y luego se establece que no se aceptan autoreferencias para este EJB entre las etiquetas <reentrant> y </reentrant>.

El atributo que se especifica a continuación entre las etiquetas <session-type> y </session-type> es obligatorio par beans de sesión, debido a que estos pueden ser con estado (Stateful) o sin estado (Stateless). Entre las etiquetas <env-entry> y </env-entry> se definen las variables de entorno del bean. El nombre de la variable se rodea con las etiquetas <env-entry-name> y </env-entry-name>, el tipo entre las etiquetas <env-entry-type> y </env-entry-type> y el valor entre las etiquetas <env-entry-value> y </env-entry-value>.

Para este bean de entidad con nombre "PersonaEJB" se definen las variable de ambiente: DbUrl, User, Clave y DriverName. En la que DbUrl es la URL de la base de datos, que incluye el nombre de la misma. User y Clave son el nombre y la clave de acceso del usuario que creo la base de datos y el último parámetro es el nombre del driver.

Las etiquetas del bean de sesión del ejemplo son explícitas por si mismas, una vez que se ha entendido la finalidad de cada una de las etiquetas detalladas en los párrafos anteriores y que corresponden a un bean de entidad.

El descriptor de despliegue debe tener por nombre “ejb-jar.xml” y debe estar almacenado en el directorio META-INF de la aplicación.

El Bean de Entidad

Un bean de entidad (EntityBean) es el representante de un objeto persistente por lo cual es responsable de insertar, borrar, buscar, consultar y actualizar datos y a la vez mantener su integridad. La persistencia de los beans de entidad puede ser manejada por el contenedor o por el bean. Si es manejada por el contenedor se debe escribir código declarativo en el descriptor de despliegue para que el contenedor implemente los métodos. Mientras que, si la persistencia es manejada por el bean, el programador debe escribir el código SQL necesario usando la JDBC.

En la clase del bean se deben implementar los métodos declarados en la interfaces home y remota del bean, además de los métodos de la interfaz EntityBean debido a que todo en EJB de entidad debe implementar la interfaz EntityBean. En el listado 7, se representa el esqueleto de un bean de entidad.

```
public class Persona implements EntityBean {
//atributos de la clase
public void ejbLoad ( ) {
//leer los datos de la base de datos
}
public void ejbStore ( ) {
//almacenar los datos en la base de datos
}
public void ejbCreate (Parámetros) {
```

```
//inserta un registro en la base de datos
}
public void ejbRemove ( ) {
//elimina un registro de la base de datos
}
public String ejbFindByPrimaryKey(String clave) {
//busca el producto por su clave primaria
}
```

Listado 7: Esqueleto de un bean de entidad cuya persistencia es gestionada por el bean.

Como se observa en el listado anterior, un bean de entidad con persistencia gestionada por el bean debe definir un mínimo de cinco métodos: ejbLoad ( ), ejbStore ( ), ejbCreate ( ), ejbRemove ( ) y ejbFindXXX( ). Puede haber tantos métodos ejbCreate( ) como sean necesarios, con distintos parámetros. También se pueden codificar en el bean tantos métodos ejbFindXXX ( ) como se requiera, pero el método ejbFindByPrimaryKey ( ) es obligatorio.

Para ver el código completo correspondiente al bean de entidad "Persona" se le sugiere al lector consultar el apéndice D (listado 3). De igual forma, las interfaces home y remota del bean de entidad "Persona", es decir, "PersonaHome" y "PersonaRemote" se pueden consultar en el apéndice D, listados 1 y 2, respectivamente.

El Bean de Sesión

"Un bean de sesión contiene la lógica de negocio que se utiliza para proporcionar un servicio" [J2EEMR]. Se dice que un bean de sesión es ligero respecto a los datos y pesado respecto a la funcionalidad, mientras que un bean de sesión es todo lo contrario.

Todo bean de sesión debe implementar la interfaz "SessionBean". Dicha interfaz contiene cinco métodos: ejbActivate ( ), ejbPassivate ( ) ejbRemove ( ), setSessionContext(SessionContext), ejbCreate ( ). En la clase del bean se deben definir estos cinco métodos y los métodos que contienen la lógica de negocio. En el listado 8 se

muestra el esqueleto de un bean de sesión.

```
import javax.ejb.*;
public class AccesoBean implements SessionBean {
public void ejbCreate (  ) {}
publi void ejbPassivate (  ) {}
public void ejbActivate (  ) {}
public void ejbRemove (  ) {}
public void setSession Context (SessionContext st) {}
…
}
```

Listado 8: Esqueleto de un bean de sesión.

Para ver el código completo de las interfaces home y remota, así como la clase del bean de sesión "AccesoBean" se le sugiere al lector consultar el apéndice D, listados 4, 5 y 6; respectivamente.

5.3.- Implementación de la Capa Web o de Presentación

Las aplicaciones J2EE utilizan un cliente ligero, por lo general un navegador, el cual contiene poca o ninguna lógica de procesamiento. Los componentes del lado del servidor encargados de recibir la petición, procesarla, interactuar con los componentes necesarios para obtener una respuesta, procesarla y enviar la respuesta de regreso al cliente, son las JSP (JavaServer Pages) y los Servlets.

La interfaz de la aplicación J2EE se controla en esta etapa, a través de los servlets se le agrega dinámica a la generación de páginas en formato HTML. Los JSP (JavaServer Pages) utilizan una combinación de etiquetas HTML y etiquetas JSP para incorporar en un mismo componente páginas estáticas y dinámicas.

Un servlet es una clase Java que recibe peticiones que envía un cliente y envía información de vuelta como respuesta" [J2EEMR]. Esta clase debe extender a la clase HttpServlet y reemplazar los métodos doGet ( ) y/o doPost ( ) de esta clase.

El método doGet ( ) se utiliza para interactuar con aquellas peticiones que se envían desde un formulario utilizando el atributo METHOD= "GET" o bien, al introducir un URL en el campo de dirección del navegador. El método doPost ( ) se usa para atender peticiones que se envían de un formulario con el atributo METHOD = "POST".

Los métodos doGet ( ) y doPost ( ) reciben dos parámetros. El primero es un objeto HttpServletRequest, el cual contiene información proveniente del cliente. Mientras que el segundo es un objeto HttpServletResponse que utiliza el servlet para enviar información al cliente.

"Una JavaServer Page (JSP) es un programa del servidor cuyo diseño y funcionalidad son similares a los de un servlet. Un cliente llama a una JSP para que le proporcione un servicio cuya naturaleza depende de la aplicación J2EE" [J2EEMR].

Crear una JSP es más sencillo que crear un servlet debido a que la JSP está escrita en HTML en vez de lenguaje Java. Esto significa que una JSP no está llena de los métodos println( ) que se encuentran en un servlet. Sin embargo, una JSP proporciona fundamentalmente las mismas características que un servlet, debido a que, una JSP se convierte en un servlet la primera vez que un cliente la solicita.

A continuación en el listado 9 se presenta un ejemplo sencillo de una JSP.

```
<HTM>
<HEAD>  <TITLE>  EJEMPLO SENCILLO DE JSP
</TITLE>  <HEAD>
<BODY>
<%! String cad = "Mundo";  %>
<H1> Hola <%= cad %> JSP </H1>
```

```
<BODY>
</HTML>
```

Listado 9: Ejemplo sencillo de una JSP

En el Apéndice D listado 7, se presenta el código completo del servlet "AccesoServlet" empleado en la aplicación empresarial del Postgrado y en el listado 8 se muestra el código completo de una de las páginas JSP de la aplicación.

5.4.-Dinámica del Desarrollo

En esta etapa de implementación es recomendable seguir las mejores prácticas de J2EE y adoptar los patrones de diseño que sean necesarios. Las mejores prácticas son técnicas probadas de diseño y programación que se utilizan para construir sistemas distribuidos empresariales avanzados. Los patrones son rutinas que resuelven problemas de programación comunes que ocurren durante el desarrollo de dichos sistemas.

Es una buena práctica de programación utilizar una estrategia de validación en el servidor. En primer lugar, porque si se realiza la validación en el cliente, cada vez que se libere una nueva versión del navegador puede ser necesario modificar las rutinas de validación del cliente. En segundo lugar, por razones de seguridad. Supóngase por ejemplo que se le pide al usuario que introduzca su dirección de correo electrónico. El usuario si conoce algo de programación, puede borrar el código de validación, o crear un clon del formulario original y enviar datos no válidos para una dirección de correo electrónico y de esta manera, llenar de basura nuestra base de datos. Por esta razón se creó un bean de sesión para realizar tareas de validación.

Se desarrolló un bean de entidad por cada tabla de la base de datos. Además, de un bean de sesión por cada caso de uso. Luego se agregaron los bean de sesión en el orden en que se hicieron necesarios, por ejemplo, el bean de validación y el bean encargado de dar o negar el acceso al sistema.

Esto se hizo considerando las mejores prácticas de diseño en este caso utilizar para distribuir la funcionalidad de la aplicación la estrategia Modelo-Vista-Controlador (MVC) que apoya Sun Microsystems y que tiene sus fundamentos en la tecnología Smalltalk.

La estrategia consiste en dividir la aplicación en tres grupos: el modelo, la vista y el controlador. El modelo está formado por componentes que controlan los datos que utiliza la aplicación. La vista está formada por componentes que presentan datos al cliente, mientras que el controlador es responsable d la gestión de eventos y de coordinar las actividades del modelo y de la vista.

Los Enterprise JavaBeans de entidad se utilizan para construir componentes del modelo. De la misma forma los servlets y las JSP se utilizan para crear componentes de la vista, mientras que como componentes del controlador se utilizan Enterprise JavaBeans de sesión.

Por ejemplo, para dar acceso a los usuarios se utiliza el EJB de entidad "UsuarioSistema" como componente del modelo, el servlet "AccesoServlet" como componente de la vista y el EJB de sesión "AccesoBean" como controlador.

5.5.-Empaquetado y Despliegue de la Aplicación

En el Directorio META-INF de la aplicación se encuentran dos archivos: "ejb-jar.xml" que es el descriptor de despliegue de los Enterprise JavaBeans y el archivo "application.xml." que es el descriptor de despliegue de la aplicación. En el se describe la relación entre la parte web de la aplicación (empaquetado en un archivo con extensión "war") y la parte de EJB (empaquetada en un archivo con extensión “jar”). Estos archivos se empaquetan en un nuevo archivo con extensión "ear" y éste, es el que se despliega. Una vez desplegados, es el servidor de aplicaciones quien controla los requerimientos de los componentes de la aplicación.

Capítulo VI: Verificación y Validación del Sistema

En este capítulo se describe brevemente la forma como se estructuró el plan de prueba del sistema.

La Ingeniería de Software recomienda realizar prueba de unidades usando métodos caja negra, métodos caja blanca; pruebas de integración de unidades y las pruebas del sistema.

Se realizaron las pruebas de cada componente (unidad) en la que se incluyeron los Enterprise JavaBeans de sesión y de entidad. En el Apéndice E listado 1 se incluye el código del programa que prueba los métodos de un bean de entidad y en el listado 2, la salida de este programa, una vez corregidos los errores.

Las pruebas comunes que se practicaron son las que utilizan la estrategia de caja negra para detectar errores. Por ejemplo, en el caso del bean de sesión de acceso al sistema "AccesoBean" se probó cual era la salida del método para varios casos de prueba que incluían entradas nulas para el nombre del usuario y la clave, nombre de usuario existente y la clave incorrecta, nombre de usuario existente y clave correcta, nombre de usuario correcto y clave diferente a la correcta en un caracter, entre otros.

Para probar el bean de sesión de validación "ValidaBean" fue necesario aplicar pruebas de caja blanca además, de las pruebas de caja negra. Debido a que se presentaban errores en los flujos de los algoritmos de validación como por ejemplo en el método "validaMail" que recibe como parámetro una cadena de caracteres "String".

Una vez finalizadas las pruebas de unidades se realizaron las pruebas de integración de unidades. Por ejemplo, para el proceso de acceso de usuarios se realizaron prácticamente las mismas pruebas que se mencionaron para el bean de entidad "UsurarioSistema". La diferencia es que ahora no estamos probando solamente al bean de entidad, sino la

integración del bean de entidad "UsuarioSistema" con el bean de sesión "AccesoBean" y con el servlet "AccesoServlet".

Las pruebas de integración son necesarias y se justifican debido a que los tres componentes pueden funcionar bien por separado, pero esto no garantiza que integrados funcionen correctamente.

Como el sistema, en la capa de presentación está formada en su mayoría por páginas web que incluyen en su cuerpo enlaces a otras páginas, se incluyeron en el plan de pruebas, las correspondientes a verificar si los enlaces conducían al destino correcto.

Al terminar la fase de pruebas y, una vez hechas las correcciones pertinentes, el sistema está listo para ser instalado en su plataforma de ejecución.

Conclusiones

El nombre J2EE proporcionó las herramientas necesarias para cumplir con la promesa de obtener una aplicación empresarial, disponible veinticuatro (24) a los siete (7) días de la semana, escalable, segura, portable, eficiente en el manejo de recursos y fácil de modificar y mantener, ya que no será necesario el rediseño de la aplicación cuando cambien las reglas de negocio.

Sin duda que la inclusión en el diseño o implementación del sistema de las mejoras prácticas de programación y los factores de diseño contribuyeron, en gran medida, a que el resultado fuese el esperado.

Una aplicación construida bajo esta filosofía, resulta perfecta para ser desarrollada por un equipo de programadores. Debido a que explota al máximo las habilidades de cada uno. Por ejemplo, los especialistas en lógica de negocios, pueden trabajar en la construcción de Enterprise JavaBeans en forma independiente respecto al resto del equipo. De igual manera, los artistas gráficos pueden trabajar en los elementos de presentación de las páginas estáticas y, paralelamente, los expertos en bases de datos y SQL pueden desarrollar la capa de datos.

La aplicación posee una interfaz amigable basada en las experiencias de los usuarios. Esto debido al uso de páginas web desplegadas por navegadores. El sistema es altamente flexible y adaptable a nuevas demandas de uso y a cambios en los requerimientos de los usuarios.

Recomendaciones

El Postgrado de Computación de la Universidad de los Andes es una empresa en crecimiento y por tal razón, requiere de un sistema de información empresarial que apoye, de la mejor manera posible, el desarrollo de sus actividades de administración académica y de recursos. En consecuencia, es mi deber recomendar al Postgrado la instalación en el servidor de la distribución Suse Linux, debido a que ésta en el futuro incluirá soporte para servidores de aplicaciones Java y de esta forma, obtener mejoras en el rendimiento y ejecución de la aplicación empresarial del Postgrado.

Por otra parte, el desarrollo de aplicaciones es una tarea sin fin. En la vida real no existe un prototipo que verdaderamente se pueda llamar "prototipo final". Cada dia surgen nuevas necesidades y nuevas ideas que se pueden incorporar fácilmente al sistema por la forma como fue construido. Por consiguiente, invito al Postgrado de Computación de la Universidad de los Andes a no descuidar el desarrollo y mantenimiento del sistema y a hacer lo necesario para que la aplicación crezca al ritmo de crecimiento de las nuevas ideas y necesidades del Postgrado.

Referencias Bibliográficas

[DCB1] Alan Freedman.
"Diccionario de Computación Bilingüe".
McGraw-Hill 1996.

[DAE] Profesor Jonás Montilva y Wladimir Rodríguez.
Material correspondiente a la materia "Desarrollo de Aplicaciones Empresariales".
Julio 2004.

[INT1] http://www.omg.org

[INT2] http://business.fortunecity.com/bren/126/servidor1.htm

[Guerrero 99] Guerrero Varela, Elizabeth
"Desarrollo Orientado a Objeto de un Sistema de Información para la Administración de Recursos de un Postgrado".
Universidad de los Andes, Facultad de Ingeniería, Escuela de Sistemas.
Tesis de Pregrado, 1999.

[Delgado 99] Delgado, Gabriel
"Desarrollo Orientado por Objetos de un Sistema de Información para la Administración Académica de un Postgrado".
Universidad de los Andes, Facultad de Ingeniería, Escuela de Sistemas.
Tesis de Pregrado, 1999.

[JAVA 1] Naughton, Patrick
Manual de Java
McGraw-Hill/Interamericana de España, S.A.
Traducción de la primera edición en inglés: **The Java Handbook**.

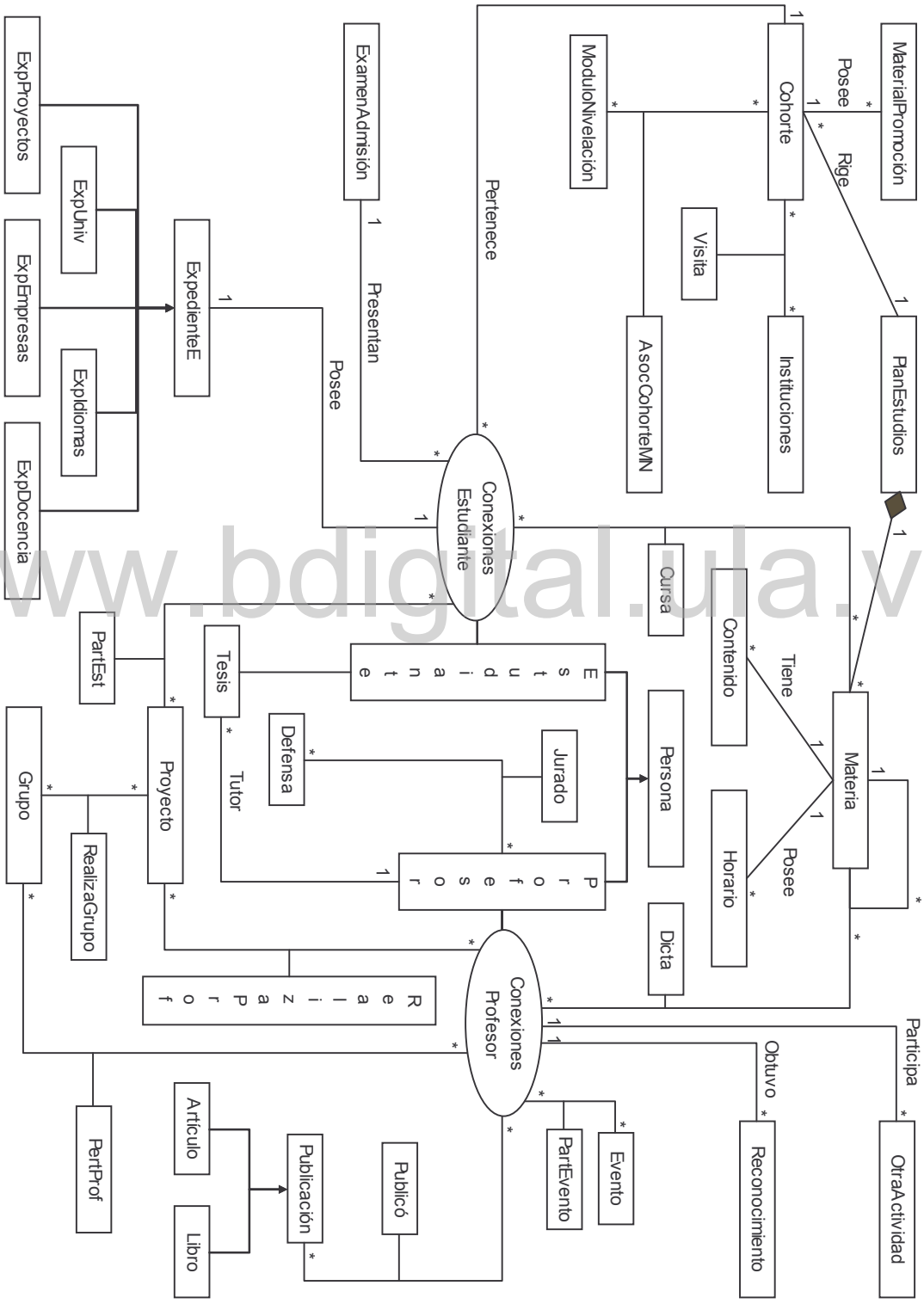
[ING-SOFT 1] Pressman, Roger
INGENIERÍA DEL SOFTWARE: Un Enfoque Práctico
McGraw-Hill/Interamericana de España, S.A.U.
Traducción de la cuarta edición en inglés:
SOFTWARE ENGINEERING: A PRACTITIONER'S APPROACH

[Montilva-96] Montilva, Jonás
Una Metodología Orientada a Objetos para el Desarrollo de Sistemas de Información Gerencial.
Universidad de los Andes, Facultad de Ingeniería, Grupo GIDYC, Informe Técnico: 1996

[J2EEMR] J2EE Manual de Referencia.
Jim Keogh.
McGraw-Hill 2002.

[ISBD] Introducción a los Sistemas de Bases de Datos.
Date, C. J.
Pearson Educación 2001.

Apéndice A: Diagrama de Clases



Apéndice B

Listado 1: Script que implementa la base de datos

```
/*********************************************************************
 * BASE DE DATOS DEL POSTGRADO DE COMPUTACION DE LA UNIVERSIDAD DE    *
 * LOS ANDES                                                          *
 * MERIDA MAYO DEL 2004                                               *
 * TESISTA: JULIO DANIEL MATHEUS F.                                   *
 * CONTACTO jmatheus007@yahoo.com                                     *
 * TUTOR: Dr. WLADIMIR RODRIGUEZ                                      *
 *********************************************************************
*/
/* 1 */
create table Persona ( Cedula text not null, Nombre text, Apellido text,
EdoCivil char, Sexo char, Nacionalidad char, Direccion text, Telefono
text,
FechaN date, LugarN text, Email text, DirTrabajo text, TlfTrabajo text,
Estado text, Pais text, primary key ( Cedula ) );
/* 2 */
create table Profesor ( Cedula text, Dedicacion text, Condicion text,
Categoria text, FechaIngUla date, FechaIngPostgrado date, HoraSemUla int,
HoraSemPost int, primary key ( Cedula ), foreign key ( Cedula )
references Persona );
/* 3 */
create table ExamenAdmision ( CodigoEA text not null , Fecha date,
primary key ( CodigoEA ) );
/* 4 */
create table PlanEstudios ( CodigoPE text not null, Programa text, Tiempo
char,
Modaliadad text, Semestres int, UnidadesA int, UnidadesT int, NotaAprob
float,
Suficiencia char, FechaCT date, FechaCEP date, primary key ( CodigoPE )
);
/* 5 */
create table Cohorte ( CodCohorte text not null, Numero text,
SemInicio text, AñoInicio text, SemFinal text, AñoFinal text,
NumEst int, CodigoPE text, primary key ( CodCohorte ), foreign key
( CodigoPE ) references PlanEstudios );
/* 6 */
create table Estudiante ( Cedula text, EstadoA text, PPonderado
numeric(4,2),
PAritmetico numeric(4,2), NotaExAdm numeric(4,2), Presencial boolean,
Dedicacion text, CodigoEA text, CodCohorte text, primary key ( Cedula ),
foreign key ( Cedula ) references Persona,
foreign key ( CodigoEA ) references ExamenAdmision,
foreign key ( CodCohorte ) references Cohorte );
/* 7 */
create table Tesis ( CodigoT text not null, Titulo text, FechaInicio
date,
FechaFinal date, LineaEspInv text, Estado char, Cedula text,
CedulaP text, primary key ( CodigoT ), foreign key ( Cedula ) references
Estudiante,
foreign key ( CedulaP ) references Profesor );
```

```
/* 8 */
create table Defensa ( Cedula text, CodigoT text, FechaDef date,
Evaluacion float, primary key ( Cedula ), foreign key ( Cedula )
references Estudiante, foreign key ( CodigoT ) references Tesis );
/* 9 */
create table Jurado ( CedulaP text, Cedula text,
primary key ( CedulaP, Cedula ), foreign key ( CedulaP ) references
Profesor,
foreign key ( Cedula ) references Defensa );
/* 10 */
create table Publicacion ( CodigoP text not null, Titulo text, Año text,
primary key ( CodigoP ) );
/* 11 */
create table Articulo ( CodigoP text, Revista text, Volumen int, Numero
int,
Pp text, Fecha date, primary key ( CodigoP ), foreign key ( CodigoP )
references Publicacion );
/* 12 */
create table Libro ( CodigoP text, Editorial text, Edicion text,
primary key ( CodigoP ), foreign key (CodigoP) references Publicacion );
/* 13 */
create table Publico ( Cedula text, CodigoP text, Tipo text,
primary key ( Cedula, CodigoP ), foreign key ( CodigoP ) references
Publicacion,
foreign key ( Cedula ) references Profesor );
/* 14 */
create table Evento ( CodEvento text not null, Nombre text, Tipo text,
Tema text, Lugar text, Ciudad text, Pais Text, FechaInicio date,
FechaFinal date, primary key ( CodEvento ) );
/* 15 */
create table PartEvento ( Cedula text, CodEvento text, Participacion
text,
primary key ( Cedula, CodEvento ), foreign key ( Cedula ) references
Profesor,
foreign key ( CodEvento ) references Evento );
/* 16 */
create table Presentacion ( CodEvento text, CodigoP text,
primary key ( CodEvento, CodigoP ), foreign key ( CodEvento ) references
Evento,
foreign key ( CodigoP ) references Articulo );
/* 17 */
create table OtraActividad ( NombreOA text not null, Fecha date,
Cedula text, primary key ( NombreOA ), foreign key ( Cedula ) references
Profesor );
/* 18 */
create table Reconocimiento ( NombreRec text not null, Tipo text,
Fecha date, CedulaP text, primary key ( NombreRec ), foreign key
( CedulaP ) references Profesor );
/* 19 */
create table Grupo ( CodGrupo text not null, Nombre text, Iniciales text,
LineaInv text, primary key ( CodGrupo ) );
/* 20 */
create table Proyecto ( CodProyecto text not null, Nombre text,
Financinamiento text, Responsable text, Ayudantes text, Ciudad text,
FechaInicio date, FechaFinal date, Ejecucion char,
primary key ( CodProyecto ) );
/* 21 */
```

```
create table PertProf ( CedulaP text, CodGrupo text,
primary key ( CedulaP, CodGrupo ), foreign key (CodGrupo) references
Grupo,
foreign key ( CedulaP ) references Profesor );
/* 22 */
create table RealizaProf ( CedulaP text, CodProyecto text,
primary key ( CedulaP, CodProyecto ), foreign key ( CedulaP ) references
Profesor,
foreign key ( CodProyecto ) references Proyecto );
/* 23 */
create table RealizaGrupo ( CodGrupo text, CodProyecto text,
primary key ( CodGrupo, CodProyecto ), foreign key ( CodGrupo )
references Grupo,
foreign key ( CodProyecto ) references Proyecto );
/* 24 */
create table PartEst ( Cedula text, CodProyecto text,
primary key ( Cedula, CodProyecto ), foreign key ( CodProyecto )
references Proyecto,
foreign key ( Cedula ) references Estudiante );
/* 25 */
create table ExpedienteE ( Cedula text, Codigo text, FuenteIng char,
Programa text, Dedicacion text, LineaEspInv text,
primary key ( Cedula ), foreign key ( Cedula ) references Estudiante );
/* 26 */
create table ExpUniv ( Cedula text, TituloUniv text, Carrera text,
Mencion text, FechaDesde date, FechaHasta date, Instituto text,
primary key ( Cedula ), foreign key ( Cedula ) references Estudiante );
/* 27 */
create table ExpEmpresas ( Cedula text, Nombre text, Direccion text,
Telefono text, Ciudad text, Estado text, Pais text, JefeInmS text,
TlfImnS text, FechaDesde date, FechaHasta date, TipoEmpresa text,
TipoCargo text, Actividad text, primary key ( Cedula ), foreign
key ( Cedula ) references Estudiante );
/* 28 */
create table ExpProyectos ( Cedula text, Nombre text, Fecha date,
primary key ( Cedula ), foreign key ( Cedula ) references Estudiante );
/* 29 */
create table ExpIdiomas ( Cedula text, Idioma text,
primary key ( Cedula ), foreign key ( Cedula ) references Estudiante );
/* 30 */
create table ExpDocencia ( Cedula text, Curso text, Fecha date,
Institucion text, primary key ( Cedula ), foreign key ( Cedula )
references Estudiante );
/* 31 */
create table ModuloNivelacion ( CodigoMN text not null, Nombre text,
Descripcion text, primary key ( CodigoMN ) );
/* 32 */
create table AsocCohorteMN ( CodCohorte text, CodigoMN text,
primary key ( CodCohorte, CodigoMN ), foreign key ( CodCohorte )
references Cohorte,
foreign key ( CodigoMN ) references ModuloNivelacion );
/* 33 */
create table Instituciones ( CodigoInst text not null, Nombre text,
Tipo char, Direccion text, Telefono text, Ciudad text, Estado text,
primary key ( CodigoInst ) );
/* 34 */
create table Visita ( CodCohorte text, CodigoInst text, Fecha date,
```

```
Motivo text, primary key ( CodCohorte, CodigoInst ), foreign key (
CodCohorte ) references Cohorte, foreign key ( CodigoInst ) references
Instituciones );
/* 35 */
create table MaterialPromocion ( CodigoMP text not null, Tipo char,
Descripcion text, PathImagen text, CodCohorte text,
primary key ( CodigoMP ), foreign key ( CodCohorte ) references Cohorte
);
/* 36 */
create table Materia ( CodigoMat text not null, Nombre text, Tipo char,
HoraTeoria int, HoraPractica int, HoraLab int, ActExtras text,
Unidades int, Periodo int, CodigoPE text,
primary key ( CodigoMat ), foreign key ( CodigoPE ) references
PlanEstudios );
/* 37 */
create table Contenido ( CodigoMat text, Titulo text, Descripcion text,
CodigoC text, primary key ( CodigoMat ), foreign key ( CodigoMat )
references
Materia );
/* 38 */
create table Seccion ( SeccionId text not null, CodigoMat text,
primary key ( SeccionId ), foreign key ( CodigoMat ) references Materia
);
/* 39 */
create table BloqueHorario ( BloqueHorarioId text, HoraI text,
HoraF text, TurnoI text, TurnoF text, NombreDia text,
primary key ( BloqueHorarioId ) );
/* 40 */
create table SeccionBloqueH ( SeccionId text, BloqueHorarioId text,
primary key ( SeccionId, BloqueHorarioId ), foreign key ( SeccionId )
references Seccion,
foreign key ( BloqueHorarioId ) references BloqueHorario );
/* 41 */
create table Prela ( CodigoMat text, CodigoMatQuePrela text,
primary key ( CodigoMat, CodigoMatQuePrela ), foreign key ( CodigoMat )
references
Materia, foreign key ( CodigoMatQuePrela ) references Materia );
/* 42 */
create table Dicta ( CedulaP text, CodigoMat text, Semestre text,
Año text, primary key ( CedulaP, CodigoMat ), foreign key ( CedulaP )
references
Profesor, foreign key ( CodigoMat ) references Materia );
/* 43 */
create table Curso ( Cedula text, CodigoMat text, Semestre text,
Año text, NotaMat numeric(4,2), primary key ( Cedula, CodigoMat ),
foreign key
( CodigoMat ) references Materia, foreign key ( Cedula ) references
Estudiante );
/* SISTEMA DE ADMINISTRACION DE RECURSOS */
/* 44 */
create table PersonalFijo ( Cedula text, TipoATS text, FechaInicio date,
primary key ( Cedula ), foreign key ( Cedula ) references Persona );
/* 45 */
create table PersonalContratado ( Cedula text, TipoATS text, FechaIniCont
date,
FechaFinCont date, Monto float, primary key ( Cedula ), foreign key (
Cedula )
```

```
references Persona );
/* 46 */
create table Becario ( Cedula text, tipoEst text, FechaIng date,
OrgFinancia text,
MontoBeca float, MontoCompl float, Horario text, primary key ( Cedula ),
foreign key ( Cedula ) references Persona );
/* 47 */
create table UsuarioMB ( Cedula text, NumCarnet text, NomApeU text,
TipoU text, FechaIni date, Suspendido boolean, FechaIsusp date,
FechaFsusp date,
Observacion text, Estatus text, primary key ( Cedula ), foreign key
( Cedula ) references Persona );
/* 48 */
create table UsuarioSistema ( Cedula text, NombreUsuario text, Clave
text,
TipoUsuario text, primary key ( Cedula ), foreign key ( Cedula )
references Persona );
/* 49 */
create table MaterialBibliografico ( Cota text not null, Titulo text,
NumEjemplares int, NumDisponibles int, NumPrestados int,
primary key ( Cota ) );
/* 50 */
create table Autor ( NombreAutor text, primary key ( NombreAutor ) );
/* 51 */
create table AutorMB ( Cota text, NombreAutor text,
primary key ( Cota, NombreAutor ), foreign key ( Cota ) references
MaterialBibliografico, foreign key ( NombreAutor ) references Autor );
/* 52 */
create table Bibliografia ( Cota text, Materia text, Editorial text,
Edicion text, Año text, Pais text, primary key ( Cota ), foreign key (
Cota ) references MaterialBibliografico );
/* 53 */
create table Revista ( Cota text, Editorial text, Volumen int,
Numero int, Año text, primary key ( Cota ), foreign key ( Cota )
references
MaterialBibliografico );
/* 54 */
create table Manual ( Cota text, Volumen int, Editorial text,
Compañia text, primary key ( Cota ), foreign key ( Cota ) references
MaterialBibliografico );
/* 55 */
create table LibroMB ( Cota text, Materia text, Coleccion text,
Serie text, Editorial text, Edicion int, Año text, primary key ( Cota ),
foreign key ( Cota ) references MaterialBibliografico );
/* 56 */
create table Monografia ( Cota text, Editorial text, NumPaginas int,
Año text, LinInv text, primary key ( Cota ), foreign key ( Cota )
references MaterialBibliografico );
/* 57 */
create table ArticuloMB ( Cota text, Materia text, Revista text,
Volumen int, Numero int, Fecha date, PP text, primary key ( Cota ),
foreign key ( Cota ) references MaterialBibliografico );
/* 58 */
create table MaterialInst ( Cota text, Materia text, Descripcion text,
Año text, primary key ( Cota ), foreign key ( Cota ) references
MaterialBibliografico );
/* 59 */
```

```
create table MemoCongreso ( Cota text, NombreEvento text, Actas text,
Editorial text, Edicion int, NumPaginas int, primary key ( Cota ),
foreign key ( Cota ) references MaterialBibliografico );
/* 60 */
create table TesisMB ( Cota text, FechaPub date, LineaInv text,
primary key ( Cota ), foreign key ( Cota ) references
MaterialBibliografico );
/* 61 */
create table SoftwareMB ( Cota text, MarcaEmp text, Version text,
NumLic text, SistOp text, NumFactura text, CodigoPro text,
primary key ( Cota ), foreign key ( Cota ) references
MaterialBibliografico );
/* 62 */
create table Prestamo ( Cota text, NumCed text, TipoU text,
FechaD date, FechaP date, FechaV date, primary key ( Cota, NumCed ),
foreign key ( Cota ) references MaterialBibliografico,
foreign key ( NumCed ) references UsuarioMB );
/* 63 */
create table Partida ( CodP text not null, Año text,
MonAcep numeric(15,2), MonCcep numeric(15,2), MonEcep numeric(15,2),
MonDcep numeric(15,2), MonAcc numeric(15,2), MonEcc numeric(15,2),
MonDcc numeric(15,2), primary key ( CodP ) );
/* 64 */
create table Tercero ( NiTer text not null, Nombre text, Apellido text,
Direccion text, Telefono text, Estado text, Pais text, NombreEmp text,
Tipo char, Email text, primary key ( NiTer ) );
/* 65 */
create table Proveedor ( CodigoPro text not null, NombreE text,
Direccion text, Telefono text, Fax text, Email text, NombreContacto text,
PaginaWeb text, Niter text, primary key ( CodigoPro ),
foreign key ( Niter ) references Tercero );
/* 66 */
create table Transaccion ( NumeroTran text, NiTer text, FechaT date,
TipoDoc text, NumDoc text, primary key ( NumeroTran ), foreign key
( NiTer ) references Tercero );
/* 67 */
create table Cuenta ( CodCuenta text, Tipo char, Nombre text,
primary key ( CodCuenta ) );
/* 68 */
create table Referencia ( CodCuenta text, NumeroTran text, Saldo float,
primary key ( CodCuenta, NumeroTran ), foreign key ( CodCuenta )
references
Cuenta, foreign key ( NumeroTran ) references Transaccion );
/* 69 */
create table Equipo ( CodigoE text not null, NumCtrlBien text,
Marca text, Modelo text, Serial text, Ubicacion text, Descripcion text,
Tipo text, Costo numeric(15,2), FechaAd date, CodigoPro text,
primary key ( CodigoE ), foreign key ( CodigoPro ) references Proveedor
);
/* 70 */
create table PC ( CodigoE text, TipoProcesador text, Ram int, CapDD text,
CapTG text, VelCD text, TarjFM char, Floppy3 char,
primary key ( CodigoE ), foreign key ( CodigoE ) references Equipo );
/* 71 */
create table Monitor ( CodigoE text, Tamaño text, Dpi text,
CodigoPC text, primary key ( CodigoE ), foreign key ( CodigoE )
references Equipo, foreign key ( CodigoPC ) references PC );
```

```
/* 72 */
create table Periferico ( CodigoE text, TipoPeriferico text,
Caracteristicas text, CodigoPC text, primary key ( CodigoE ), foreign
key ( CodigoE ) references Equipo , foreign key ( CodigoPC )
references PC );
/* 73 */
create table Audiovisual ( CodigoE text, TipoAudio text,
Caracteristicas text, primary key ( CodigoE ), foreign key ( CodigoE )
references Equipo );
/* 74 */
create table Mobiliario ( CodigoMo text not null, NumCtrlBien text,
TipoMobiliario text, Ubicacion text, Caracteristicas text,
NumFactura text, CodigoPro text, primary key ( CodigoMo ), foreign key (
CodigoPro ) references Proveedor );
/* 75 */
create table Materiales ( CodigoMa text not null, TipoMaterial text,
Existencia int, Descripcion text, CodigoPro text,
primary key ( CodigoMa ), foreign key ( CodigoPro ) references Proveedor
);
```

Apéndice C

Representación detallada de los atributos de cada esquema

Persona		
Campo	**Tipo**	**Descripción**
Cédula	Cadena de caracteres (Números)	Número de Cédula
Nombre	Cadena de caracteres (Letras)	Nombres
Apellido	Cadena de caracteres (Letras)	Apellidos
Sexo	Letra {M, F}	Sexo
EdoCivil	Letra {S, C, D, V} S->Soltero C->Casado D->Divorciado V->Viudo	Estado Civil
Nacionalidad	Letra {V, E} V->venezolano E->Extranjero	Nacionalidad
FechaN	Fecha	Fecha de Nacimiento
LugarN	Cadena de caracteres (Letras)	Lugar de Nacimiento
Dirección	Cadena de caracteres (Letras, Números)	Dirección de Habitación
Teléfono	Cadena de caracteres (Números)	Teléfono de Habitación
DirTrabj	Cadena de caracteres (Letras, Números)	Dirección del Lugar de Trabajo
TlfTrabj	Cadena de caracteres (Números)	Teléfono del Lugar de trabajo
Estado	Cadena de caracteres (Letras)	Estado del país donde está radicado
País	Cadena de caracteres (Letras)	País donde está radicado
Email	Cadena de caracteres (Letras, Números, @, .)	Dirección de Correo Electrónico

Profesor		
Campo	**Tipo**	**Descripción**
Dedicación	Cadena de caracteres (Letras) {Exclusiva, Parcial}	Tipo de Dedicación
Condición	Cadena de caracteres (Letras) {Afiliado, Invitado}	Condición
Categoría	Cadena de caracteres (Letras) {Agregado, Instructor, Asociado, Titular, Asistente}	Categoría del Profesor

FechaIngULA	Fecha	Fecha de Ingreso a la ULA
FechaIngPost	Fecha	Fecha de Ingreso al Postgrado
HorasSemULA	Entero Positivo	Horas a la semana que trabaja en la ULA
HorasSemPost	Entero Positivo	Horas a la semana que trabaja en la Postgrado

ExámenAdmisión		
Campo	**Tipo**	**Descripción**
CodExAdm	Cadena de caracteres (Letras)	Código asignado al examen
Fecha	Fecha	Fecha de realización del examen

Cohorte		
Campo	**Tipo**	**Descripción**
Número	Cadena de caracteres (Números)	Número de la Cohorte
CodCohorte	Cadena de caracteres (Letras)	Código de la Cohorte
SemInicio	Cadena de caracteres (Letras, Números)	Semestre de inicio
SemFinal	Cadena de caracteres (Letras, Números)	Semestre de finalización
AñoInicio	Cadena de caracteres (Números)	Año de inicio
AñoFinal	Cadena de caracteres (Números)	Año de finalización
NumEst	Entero Positivo	Número de Estudiantes pertenecientes a la cohorte

Estudiante		
Campo	**Campo**	**Campo**
EstadoA	Cadena de caracteres (Letras) {Aspirante, Pre-Seleccionado, Seleccionado, Admitido, Retirado, Egresado}	Estado Actual del Estudiante
PPonderado	Real entre 0.00 y 20.00	Promedio Ponderado
PAritmético	Real entre 0.00 y 20.00	Promedio Aritmético

Egresado		
Campo	**Tipo**	**Descripción**
NombProm	Cadena de caracteres (Letras)	Nombre de la Promoción
SemEgreso	Cadena de caracteres (Letras, Números)	Semestre de Egreso

AñoEgreso	Cadena de caracteres (Números)	Año de Egreso
FechaGrado	Fecha	Fecha de Graduación
LineaEspInv	Cadena de caracteres (Letras)	Área de especialización

ExpedienteE		
Campo	**Tipo**	**Descripción**
FuenteIng	Letra {P, B, E, O} P-> Personal B->Beca E->Empresa O->Otros	Fuente de ingreso con que cuenta el estudiante
Programa	Letra {E, M} E->Especialización M->Maestría	
Dedicación	Letra {M, C} M->Medio C->Completo	Tiempo dedicado a los estudios de Postgrado
LineaEspInv	Cadena de caracteres (Letras)	Línea de especialización o investigación
ExpProg	Cadena de caracteres (Letras, Números)	Registra la experiencia en Programación y manejo de Software

ExpUni		
Campo	**Tipo**	**Descripción**
TítuloUniv	Cadena de caracteres (Letras)	Título Universitario obtenido
Carrera	Cadena de caracteres (Letras)	Nombre de la carrera
Mención	Cadena de caracteres (Letras)	Nombre de la mención
FechaDesde	Fecha	Fecha de inicio
FechaHasta	Fecha	Fecha final
Instituto	Cadena de caracteres (Letras, Números)	Nombre del Instituto

ExpEmpresas		
Campo	**Tipo**	**Descripción**
Nombre	Cadena de caracteres (Letras, Números)	Nombre de la empresa
Dirección	Cadena de caracteres (Letras, Números)	Dirección de la empresa
Teléfono	Cadena de caracteres (Números)	Número de teléfono de la empresa
Estado	Cadena de caracteres (Letras)	Estado donde está ubicada
País	Cadena de caracteres (Letras)	País
JefeInmSup	Cadena de caracteres (Letras)	Nombre del Jefe

TlfInmSup	Cadena de caracteres (Números)	Número de teléfono del Jefe
FechaDesde	Fecha	Fecha de inicio
FechaHasta	Fecha	Fecha de culminación
TipoEmpresa	Cadena de caracteres (Letras)	Tipo
TipoCargo	Cadena de caracteres (Letras, Números)	Nombre del cargo

ExpProyectos		
Campo	**Tipo**	**Descripción**
Nombre	Cadena de caracteres (Letras)	Nombre del Proyecto
Fecha	Fecha	Fecha de culminación del Proyecto

ExpIdiomas		
Campo	**Tipo**	**Descripción**
Idioma	Cadena de caracteres (Letras)	Nombre del idioma

ExpDocencia		
Campo	**Tipo**	**Descripción**
Curso	Cadena de caracteres (Letras, Números)	Título del curso
Fecha	Fecha	Fecha de culminación del curso
Instituto	Cadena de caracteres (Letras, Números)	Nombre de la instituto

Tesis		
Campo	**Tipo**	**Descripción**
CódigoT	Cadena de caracteres (Letras, Números)	Código de la tesis
Título	Cadena de caracteres (Letras, Números)	Título de la tesis
FechaInicio	Fecha	Fecha inicial
FechaFinal	Fecha	Fecha final
LineaEspInv	Cadena de caracteres (Letras)	Línea de especialización o investigación en que el estudiante realizó la tesis
Estado	Letra {E, T, A} E->En ejecución T->Terminada A->Abandonada	Indica el estado de la tesis
Evaluación	Real entre 0.00 y 20.00	Nota que obtuvo

Evento		
Campo	**Tipo**	**Descripción**
CodEvento	Cadena de caracteres (Letras, Números)	Código del Evento
Nombre	Cadena de caracteres (Letras, Números)	Nombre del evento
Tipo	Cadena de caracteres (Letras) {Congreso, Taller, Coloquio, Plenaria}	Tipo de evento
Tema	Cadena de caracteres (Letras)	Tema principal
Lugar	Cadena de caracteres (Letras)	Lugar donde se realizó el evento
Ciudad	Cadena de caracteres (Letras)	Ciudad
País	Cadena de caracteres (Letras)	País
FechaInicio	Fecha	Fecha inicial
FechaFinal	Fecha	Fecha final

PartEvento		
Campo	**Tipo**	**Descripción**
Participación	Cadena de caracteres (Letras)	Registra la forma de participación

OtraActividad		
Campo	**Tipo**	**Descripción**
NombreOA	Cadena de caracteres (Letras)	Nombre de la otra actividad
Fecha	Fecha	Fecha de culminación

Reconocimiento		
Campo	**Tipo**	**Descripción**
NombreRec	Cadena de caracteres (Letras)	Nombre del reconocimiento
Tipo	Cadena de caracteres (Letras)	Tipo de reconocimiento. Ej. diploma, medalla, etc.
Fecha	Fecha	Fecha de recepción

Publicación		
Campo	**Tipo**	**Descripción**
CódigoP	Cadena de caracteres (Libre)	Cota asignada por la biblioteca
Título	Cadena de caracteres (Libre)	Título de la publicación

Año	Cadena de caracteres (Números)	Año de publicación

Artículo		
Campo	**Tipo**	**Descripción**
Revista	Cadena de caracteres (Libre)	Nombre de la revista
Vol	Entero positivo	Volumen
Num	Entero positivo	Número
Pp	Cadena de caracteres (Números)	Página donde está el artículo
Fecha	Fecha	Fecha de publicación

Libro		
Campo	**Tipo**	**Descripción**
Editorial	Cadena de caracteres (Libre)	Nombre de la editorial
Edición	Entero positivo	Número de la edición

PlanEstudio		
Campo	**Tipo**	**Descripción**
CódigoPE	Cadena de caracteres (Letras, Números)	Código del Plan de Estudios
Programa	Letra {E, M} E->Especialización M->Maestría	
Tiempo	Letra {C, M} C->Completo M->Medio	
Modalidad	Letra {D, P} D->Distancia P->Presencial	Modalidad del plan de estudios
Semestres	Entero positivo	Número de semestres requeridos
UnidadesA	Entero positivo	Unidades aprobatorias
UnidadesT	Entero positivo	Unidades de la tesis
NotaAprob	Real entre 0.00 y 20.00	Nota aprobatoria
Suficiencia	Lógico {true, false}	Registra si el Plan de Estudios contempla la realización de un examen de admisión
FechaCT	Fecha	Fecha de aprobación del Consejo Técnico
FechaCEP	Fecha	Fecha de aprobación del CEP

Materia		
Campo	**Tipo**	**Descripción**

CódigoM	Cadena de caracteres (Libre)	Código de la materia
Nombre	Cadena de caracteres (Libre)	Nombre de la materia
Tipo	Letra {O, E} O->Obligatoria E->Electiva	Tipo de la materia
HorasTeo	Entero positivo	Número de horas de teoría
HorasPrac	Entero positivo	Número de horas prácticas
HorasLab	Entero positivo	Número de horas de laboratorio
ActExtras	Cadena de caracteres (Letras, Números)	Actividades extras de la materia
Unidades	Entero positivo	Número de créditos
Período	Entero positivo	Registra en qué período de la Especialización o Maestría se debe cursar la materia {1,2,3,4} a tiempo completo o {1,2,3,4,5,6} a medio tiempo

Dictó		
Campo	**Tipo**	**Descripción**
Semestre	Cadena de caracteres (Letras, Números)	Semestre en que fue dictada la materia
Año	Cadena de caracteres (Números)	Año

Grupo		
Campo	**Tipo**	**Descripción**
CódigoG	Cadena de caracteres (Letras, Números)	Código del grupo
Nombre	Cadena de caracteres (Letras, Números)	Nombre del grupo
Iniciales	Cadena de caracteres (Letras, Números)	Iniciales del grupo
LineaInv	Cadena de caracteres (Letras)	Línea de investigación

Proyecto		
Campo	**Tipo**	**Descripción**
CodProyecto	Cadena de caracteres (Letras, Números)	Código del proyecto
Nombre	Cadena de caracteres (Letras, Números)	Nombre del proyecto
Financiamiento	Cadena de caracteres (Libre)	Organismo que lo financia
Responsable	Cadena de caracteres (Letras)	Jefe principal del proyecto

Ayudantes	Cadena de caracteres (Letras)	Personas que colaboraron en el proyecto
Ciudad	Cadena de caracteres (Letras)	Ciudad donde se realizó el proyecto
FechaInicio	Fecha	Fecha inicial
FechaFinal	Fecha	Fecha final
Estado	Letra {E, F, S, C} E->En Ejecución F->Finalizado S->Suspendido C->Cancelado	Registra el estado del proyecto

Instituciones		
Campo	**Tipo**	**Descripción**
CódigoInst	Cadena de caracteres (Letras, Números)	Código de la institución
Nombre	Cadena de caracteres (Letras)	Nombre de la institución
Tipo	Letra {E, U, I} E->Empresa U->Universidad I->Instituto	Tipo de institución visitada
Dirección	Cadena de caracteres (Letras, Números)	Ubicación de la institución
Teléfono	Cadena de caracteres (Números)	Número de teléfono
Ciudad	Cadena de caracteres (Letras)	Ciudad donde está ubicada
Estado	Cadena de caracteres (Letras)	Estado en que está ubicado

Visita		
Campo	**Tipo**	**Descripción**
Fecha	Fecha	Fecha de la visita
Motivo	Cadena de caracteres (Letras, Números)	Motivo de la visita

MóduloNivelación		
Campo	**Tipo**	**Descripción**
CódigoMN	Cadena de caracteres (Letras, Números)	Código del Módulo de Nivelación
Nombre	Cadena de caracteres (Letras)	Nombre del Módulo de Nivelación
Descripción	Cadena de caracteres (Libre)	Registra en forma resumida el contenido temático de cada Módulo

MaterialPromoción		
Campo	**Tipo**	**Descripción**

Tipo	Letra {A, T, F} A->Afiche T->Tríptico F->Folleto	Tipo de material
Descripción	Cadena de caracteres (Letras, Números)	Registra en forma resumida el contenido del material
Imagen	Cadena de caracteres (Libre)	Ruta y nombre del archivo de imagen de la portada del material

Cursó		
Campo	**Tipo**	**Descripción**
Semestre	Cadena de caracteres (Letras, Números)	Semestre en que fue cursada la materia
Año	Cadena de caracteres (Números)	Año en que fue cursada la materia
NotaMat	Real entre 0.00 y 20.00	Nota obtenida

Contenido		
Campo	**Tipo**	**Descripción**
Título	Cadena de caracteres (Letras, Números)	Título
Descripción	Cadena de caracteres (Letras, Números)	Breve descripción

Horario		
Campo	**Tipo**	**Descripción**
CódigoH	Cadena de caracteres (Letras, Números)	Código del horario, equivalente a sección

Días		
Campo	**Tipo**	**Descripción**
Dia	Cadena de caracteres (Letras) {Lunes, Martes, Miércoles, Jueves, Viernes}	

HorasTurnos		
Campo	**Tipo**	**Descripción**
Hora	Cadena de caracteres (Números, Signos)	
Turno	Letra {M, T} M->Mañana T->Tarde	

UsuarioSistema		
Campo	**Tipo**	**Descripción**
TipoU	Letra {E, P, A, C, S, O} E->Estudiante P->Profesor A->Administrador C->Coordinador S->Secretaria O->Otros	Tipo de usuario
login	Cadena de caracteres (Letras, Números). Debe comenzar con una letra	Nombre de usuario
Clave	Cadena de caracteres (Letras, Números)	Clave de acceso

UsuarioMB		
Campo	**Tipo**	**Descripción**
TipoU	Letra {E, P, O} E->Estudiante P->Profesor O->Otros	Tipo de usuario de material bibliográfico
FechaIns	Fecha	Fecha de Inscripción
Estado	Letra {A, S} A->Activo S->Suspendido	Estado del usuario
FehcaIniSusp	Fecha	Fecha de suspensión
FechaFinSusp	Fecha	Fecha de vencimiento suspensión
Observación	Cadena de caracteres (Letras, Números)	

PersonalATS		
Campo	**Tipo**	**Descripción**
Tipo	Letra {A, T, S} A->Administrativo T->Técnico S->Servicio	Tipo de Personal
Monto	Numérico (10,2)	Pago mensual

PersonalFijo		
Campo	**Tipo**	**Descripción**
FechaInicio	Fecha	Fecha de inicio en el Postgrado

PersonalContratado		
Campo	**Tipo**	**Descripción**
FechaIniCont	Fecha	Fecha de inicio contrato
FechaFinCont	Fecha	Fecha de culminación del contrato

MaterialBibliográfico		
Campo	**Tipo**	**Descripción**
CódigoMB	Cadena de caracteres (Letras, Números)	Código del material equivalente a cota
Título	Cadena de caracteres (Letras, Números)	Título del material
NumEjemp	Entero positivo	Número de ejemplares
NumDisp	Entero positivo	Número de ejemplares disponibles
NumPres	Entero positivo	Número de ejemplares prestados

Bibliografía		
Campo	**Tipo**	**Descripción**
Autores	Cadena de caracteres (Letras)	Autores del material
Editorial	Cadena de caracteres (Letras)	Editorial
Edición	Entero positivo	Número de la edición
Año	Cadena de caracteres (Números)	Año de publicación
País	Cadena de caracteres (Letras)	Lugar donde fue impreso
Materia	Cadena de caracteres (Letras, Números)	Materias a las cuales pertenece

Revista		
Campo	**Tipo**	**Descripción**
Editor	Cadena de caracteres (Letras)	Nombre del editor
Vol	Entero positivo	Volumen
Año	Cadena de caracteres (Números)	Año de la revista
Num	Entero positivo	Número de la revista

Manual		
Campo	**Tipo**	**Descripción**
Vol	Entero positivo	Volumen
Compañía	Cadena de caracteres (Letras, Números)	Compañía a la que pertenece
Editorial	Cadena de caracteres (Letras)	Nombre del editor

PublicaciónInterna		
Campo	**Tipo**	**Descripción**
Autores	Cadena de caracteres (Letras, punto y coma)	Autores

Libro		
Campo	**Tipo**	**Descripción**
Colección	Cadena de caracteres (Letras, Números)	Colección a la que pertenece
Serie	Cadena de caracteres (Letras, Números)	Serie
Edición	Entero positivo	Número de la edición
Editorial	Cadena de caracteres (Letras)	Nombre del editor
Año	Cadena de caracteres (Números)	Año de impresión

MemoCongreso		
Campo	**Tipo**	**Descripción**
NombreEvento	Cadena de caracteres (Letras, Números)	Nombre del evento
Actas	Cadena de caracteres (Letras, Números)	
Editor	Cadena de caracteres (Letras)	Nombre del editor
Edición	Entero Positivo	Número de la edición
Pp	Cadena de caracteres (Números)	Número de la página en las que se encuentran las actas

Monografía		
Campo	**Tipo**	**Descripción**
Editor	Cadena de caracteres (Letras)	Nombre del editor
Numpp	Entero positivo	Número de páginas
Año	Cadena de caracteres (Números)	Año de publicación
LineaInv	Cadena de caracteres (Letras, Números)	Línea de investigación

Artículo		
Campo	**Tipo**	**Descripción**
Revista	Cadena de caracteres (Letras, Números)	Nombre de la revista donde fue publicado el artículo
Vol	Entero positivo	Volumen de la revista
Num	Entero positivo	Número de la revista
Fecha	Fecha	Fecha de publicación del artículo
Pp	Cadena de caracteres (Números)	Número de la página de la revista donde se encuentra el artículo

Materia	Cadena de caracteres (Letras, Números)	Materias a las cuales pertenece

MateInst		
Campo	**Tipo**	**Descripción**
Descripción	Cadena de caracteres (Letras, Números)	Resumen de lo que trata el material instructivo
Año	Cadena de caracteres (Números)	Año en que fue publicado
Materia	Cadena de caracteres (Letras, Números)	Materias a las cuales pertenece

TesisMB		
Campo	**Tipo**	**Descripción**
FechaPub	Fecha	Fecha de publicación
LineaInv	Cadena de caracteres (Letras, Números)	Línea de investigación en la que se desarrolló la tesis

Presta		
Campo	**Tipo**	**Descripción**
FechaP	Fecha	Fecha de préstamo
FechaVP	Fecha	Fecha de vencimiento del préstamo

Tercero		
Campo	**Tipo**	**Descripción**
NITer	Cadena de caracteres (Letras, Números)	Número de identificación del tercero. En caso de personas naturales la cédula o RIF y en el caso de personas jurídicas el RIF o NIT
Nombre	Nombres de la persona	En caso de personas naturales
Apellido	Apellidos de la persona	
Dirección	Cadena de caracteres (Letras, Números)	
Teléfono	Cadena de caracteres (Números)	Número de teléfono
Estado	Cadena de caracteres (Letras)	Estado del país
País	Cadena de caracteres (Letras)	
NombreEmp	Cadena de caracteres (Letras, Números)	Nombre de la empresa. En caso de personas jurídicas
Tipo	Letra {N, J} N->Natural J->Jurídica	Tipo de Persona

Email	Cadena de caracteres (Letras, punto y la @)	Dirección de correo electrónico

Proveedor		
Campo	**Tipo**	**Descripción**
Fax	Cadena de caracteres (Letras, Números)	Número de fax
NombreContacto	Cadena de caracteres (Letras)	Persona contacto
Http	Cadena de caracteres (Letras, Números, **:**, //, punto)	Página del proveedor en Internet

Mobiliario		
Campo	**Tipo**	**Descripción**
NumCtrlB	Cadena de caracteres (Letras, Números)	Número de control de bienes de la ULA
Tipo	Cadena de caracteres (Letras, Números)	Tipo de mobiliario
Características	Cadena de caracteres (Letras, Números)	Detalles
Ubicación	Cadena de caracteres (Letras, Números)	Lugar donde se encuentra
NumFact	Cadena de caracteres (Letras, Números)	Número de la factura

Software		
Campo	**Tipo**	**Descripción**
Tipo	Cadena de caracteres (Letras, Números)	Tipo de software. Ejemplo: Sistema Operativo, Programa de aplicación, etc.
Nombre	Cadena de caracteres (Letras, Números)	Nombre del software. Ejemplo: Linux Mandrake
Versión	Real	Ejemplo: 9.2
MarcaEmp	Cadena de caracteres (Letras, Números)	Ejemplo: Sun Microsystems
NumLic	Cadena de caracteres (Letras, Números)	Número de licencia
SisOpe	Cadena de caracteres (Letras, Números)	Sistema operativo sobre el que trabaja
NumFact	Cadena de caracteres (Letras, Números)	Número de factura

Materiales		
Campo	**Tipo**	**Descripción**
Tipo	Cadena de caracteres (Letras, Números)	Tipo de material
Existencia	Entero positivo	Cantidad disponible
Descripción	Cadena de caracteres (Letras, Números)	Detalles
Unidad	Cadena de caracteres (Letras)	Unidad en que está expresada la existencia

Equipo		
Campo	**Tipo**	**Descripción**
NumCtrlB	Cadena de caracteres (Letras, Números)	Número de control de bienes de la ULA
Marca	Cadena de caracteres (Letras, Números)	
Modelo	Cadena de caracteres (Letras, Números)	
Serial	Cadena de caracteres (Letras, Números)	
Ubicación	Cadena de caracteres (Letras, Números)	Lugar donde se encuentra
Descripción	Cadena de caracteres (Letras, Números)	Características generales
FechaAd	Fecha	Fecha de adquisición
NumFact	Cadena de caracteres (Letras, Números)	Número de factura

CPU		
Campo	**Tipo**	**Descripción**
TipoProcesador	Cadena de caracteres (Letras, Números)	Tipo de procesador
RAM	Entero positivo	Cantidad de memoria RAM
CapDD	Cadena de caracteres (Letras, Números)	Capacidad del disco duro
CapTG	Cadena de caracteres (Letras, Números)	Capacidad de la tarjeta gráfica
MarcaTG	Cadena de caracteres (Letras, Números)	Marca de la tarjeta gráfica
VelCD	Cadena de caracteres (Letras, Números)	Velocidad del CD
TarjFM	Lógico {true, false}	Tarjeta fax-MODEM
Floppy3	Lógico {true, false}	Unidad de disco de 3.5”

Monitor		
Campo	**Tipo**	**Descripción**
Tamaño	Cadena de caracteres (Números)	Tamaño del monitor en pulgadas
Dpi	Cadena de caracteres (Letras, Números)	Resolución

Periférico		
Campo	**Tipo**	**Descripción**
Tipo	Tipo de periférico	Ejemplo: Ratón, Impresora, etc.
Características	Cadena de caracteres (Letras, Números)	Detalles

Audiovisual		
Campo	**Tipo**	**Descripción**
Tipo	Cadena de caracteres (Letras, Números)	Tipo de equipo audiovisual
Características	Cadena de caracteres (Letras, Números)	Detalles

Transacción		
Campo	**Tipo**	**Descripción**
NumeroTran	Entero Largo consecutivo	Número de Transacción
NITer	Cadena de caracteres (Letras, Números)	Número de identificación del tercero. En caso de personas naturales la cédula o RIF y en el caso de personas jurídicas el RIF o NIT
FechaT	Fecha	Fecha de transacción
TipoDoc	Cadena de caracteres (Letras)	Tipo de documento {letra de cambio, pagaré, recibo, factura, etc.}
NumDoc	Cadena de caracteres (Letras, Números)	Número asociado al documento representante de la transacción

Cuenta		
Campo	**Tipo**	**Descripción**

CodCuenta	Cadena de caracteres (Letras, Números)	Código de la cuenta
Tipo	Letra {I, G} I->Ingreso G->Gasto	Tipo de cuenta
Nombre	Cadena de caracteres (Letras, Números)	Nombre de la cuenta

Referencia		
Campo	**Tipo**	**Descripción**
CodCuenta	Cadena de caracteres (Letras, Números)	Código de la cuenta
NumeroTran	Entero Largo consecutivo	Número de transacción
Saldo	Real	Saldo de la cuenta

Apéndice D

Listado 1: Código ejemplo: Interfaz home de un Bean de Entidad.

```
package interfaces;

import java.rmi.RemoteException;
import java.util.Collection;
import java.util.Vector;
import java.sql.Date;
import javax.ejb.CreateException;
import javax.ejb.EJBHome;
import javax.ejb.FinderException;

public interface PersonaHome extends EJBHome {

    public PersonaRemote create() throws RemoteException,
CreateException;
    public PersonaRemote create(String []c) throws RemoteException,
CreateException;
    public PersonaRemote create(String Cedula, String Nombre, String
Apellido, String EdoCivil, String Sexo, String Nacionalidad, String
Direccion, String Telefono, Date FechaN, String LugarN, String Email,
String DirTrabajo, String TlfTrabajo, String Estado, String Pais) throws
RemoteException, CreateException;
    public PersonaRemote findByPrimaryKey(String key) throws
RemoteException, FinderException;
    public Collection findAll() throws RemoteException, FinderException;
    public Collection findX(int n, String V, String Vi, String Vf,
Collection col, String campo, String ord) throws RemoteException,
FinderException;
    public Collection findX(Vector campos, Vector valores, Vector
parametros, Vector sensibilidades) throws RemoteException,
FinderException;

}
```


Listado 2: Código ejemplo: Interfaz remote de un Bean de Entidad.

```
package interfaces;

import java.sql.Date;
import java.rmi.RemoteException;
import javax.ejb.EJBObject;
import java.util.Collection;

public interface PersonaRemote extends EJBObject {

      public void setCedula( String Cedula ) throws RemoteException;
```

```
      public String getCedula() throws RemoteException;
      public void setNombre( String Nombre ) throws RemoteException;
      public String getNombre() throws RemoteException;
      public void setApellido( String Apellido ) throws RemoteException;
      public String getApellido() throws RemoteException;
      public void setEdoCivil( String EdoCivil ) throws RemoteException;
      public String getEdoCivil() throws RemoteException;
      public void setSexo( String Sexo ) throws RemoteException;
      public String getSexo() throws RemoteException;
      public void setNacionalidad( String Nacionalidad ) throws
RemoteException;
      public String getNacionalidad() throws RemoteException;
      public void setDireccion( String Direccion ) throws
RemoteException;
      public String getDireccion() throws RemoteException;
      public void setTelefono( String Telefono ) throws RemoteException;
      public String getTelefono() throws RemoteException;
      public void setFechaN( Date FechaN ) throws RemoteException;
      public Date getFechaN() throws RemoteException;
      public void setLugarN( String LugarN ) throws RemoteException;
      public String getLugarN() throws RemoteException;
      public void setEmail( String Email ) throws RemoteException;
      public String getEmail() throws RemoteException;
      public void setDirTrabajo( String DirTrabajo ) throws
RemoteException;
      public String getDirTrabajo() throws RemoteException;
      public void setTlfTrabajo( String TlfTrabajo ) throws
RemoteException;
      public String getTlfTrabajo() throws RemoteException;
      public void setEstado( String Estado ) throws RemoteException;
      public String getEstado() throws RemoteException;
      public void setPais( String Pais ) throws RemoteException;
      public String getPais() throws RemoteException;
      public boolean set(String name, String value) throws
RemoteException;
      public boolean Modificar(String []c) throws RemoteException;
      public Collection getParamValues() throws RemoteException;
      public Collection getParamNames() throws RemoteException;
}
```


Listado 3: Código ejemplo: Clase de un Bean de Entidad.

```
package ejb;

import java.rmi.RemoteException;
import java.util.Collection;
import java.util.Vector;
import java.util.Iterator;
import java.sql.*;
```

```
import util.Utilerias;

import javax.ejb.CreateException;
import javax.ejb.EJBException;
import javax.ejb.EntityBean;
import javax.ejb.EntityContext;
import javax.ejb.FinderException;
import javax.ejb.ObjectNotFoundException;
import javax.ejb.RemoveException;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;

public class Persona implements EntityBean {
      private String Cedula;
      private String Nombre;
      private String Apellido;
      private String EdoCivil;
      private String Sexo;
      private String Nacionalidad;
      private String Direccion;
      private String Telefono;
      private Date FechaN;
      private String LugarN;
      private String Email;
      private String DirTrabajo;
      private String TlfTrabajo;
      private String Estado;
      private String Pais;
    private EntityContext context;

      public void setCedula( String Cedula ) {
            this.Cedula = Cedula;
      }
      public String getCedula() {
            return this.Cedula;
      }
      public void setNombre( String Nombre ) {
            this.Nombre = Nombre;
      }
      public String getNombre() {
            return this.Nombre;
      }
      public void setApellido( String Apellido ) {
            this.Apellido = Apellido;
      }
      public String getApellido() {
            return this.Apellido;
      }
      public void setEdoCivil( String EdoCivil ) {
            this.EdoCivil = EdoCivil;
      }
      public String getEdoCivil() {
            return this.EdoCivil;
      }
      public void setSexo( String Sexo ) {
            this.Sexo = Sexo;
```

```
}
public String getSexo() {
      return this.Sexo;
}
public void setNacionalidad( String Nacionalidad ) {
      this.Nacionalidad = Nacionalidad;
}
public String getNacionalidad() {
      return this.Nacionalidad;
}
public void setDireccion( String Direccion ) {
      this.Direccion = Direccion;
}
public String getDireccion() {
      return this.Direccion;
}
public void setTelefono( String Telefono ) {
      this.Telefono = Telefono;
}
public String getTelefono() {
      return this.Telefono;
}
public void setFechaN( Date FechaN ) {
      this.FechaN = FechaN;
}
public Date getFechaN() {
      return this.FechaN;
}
public void setLugarN( String LugarN ) {
      this.LugarN = LugarN;
}
public String getLugarN() {
      return this.LugarN;
}
public void setEmail( String Email ) {
      this.Email = Email;
}
public String getEmail() {
      return this.Email;
}
public void setDirTrabajo( String DirTrabajo ) {
      this.DirTrabajo = DirTrabajo;
}
public String getDirTrabajo() {
      return this.DirTrabajo;
}
public void setTlfTrabajo( String TlfTrabajo ) {
      this.TlfTrabajo = TlfTrabajo;
}
public String getTlfTrabajo() {
      return this.TlfTrabajo;
}
public void setEstado( String Estado ) {
      this.Estado = Estado;
}
public String getEstado() {
      return this.Estado;
```

```
        }
        public void setPais( String Pais ) {
              this.Pais = Pais;
        }
        public String getPais() {
              return this.Pais;
        }
        public Persona() {
              System.out.println("Persona::Persona()");
              this.Cedula = "";
              this.Nombre = "";
              this.Apellido = "";
              this.EdoCivil = "";
              this.Sexo = "";
              this.Nacionalidad = "";
              this.Direccion = "";
              this.Telefono = "";
              this.FechaN = Date.valueOf("1970-01-01");
              this.LugarN = "";
              this.Email = "";
              this.DirTrabajo = "";
              this.TlfTrabajo = "";
              this.Estado = "";
              this.Pais = "";
        }
      public void ejbActivate() throws EJBException, RemoteException {
        System.out.println("Persona::ejbActivate()");
      }
        public void ejbLoad() throws EJBException, RemoteException {
        System.out.println("Persona::ejbLoad()");
        Connection con = null;
        PreparedStatement ps = null;
        ResultSet rs = null;
        try {
                    String sql = "SELECT * FROM Persona WHERE Cedula=?";
            con = getConnection();
            ps = con.prepareStatement(sql);
                    String key = (String) context.getPrimaryKey();
                    ps.setString(1,key);
            rs = ps.executeQuery();
            if (rs.next()) {
this.Cedula = rs.getString(1);
this.Nombre = rs.getString(2);
this.Apellido = rs.getString(3);
this.EdoCivil = rs.getString(4);
this.Sexo = rs.getString(5);
this.Nacionalidad = rs.getString(6);
this.Direccion = rs.getString(7);
this.Telefono = rs.getString(8);
this.FechaN = rs.getDate(9);
this.LugarN = rs.getString(10);
this.Email = rs.getString(11);
this.DirTrabajo = rs.getString(12);
this.TlfTrabajo = rs.getString(13);
this.Estado = rs.getString(14);
this.Pais = rs.getString(15);
            }
```

```
      } catch (SQLException e) {
          System.out.println("SQLException:Persona:ejbLoad()");
          e.printStackTrace();
      } finally {
          try {
            if (rs != null) rs.close();
            if (ps != null) ps.close();
            if (con != null) con.close();
          } catch (SQLException e) {e.printStackTrace();}

      }
    }
    public void ejbPassivate() throws EJBException, RemoteException {
      System.out.println("Persona::ejbPasivate()");
    }
    public void ejbRemove() throws RemoveException,  RemoteException {
      System.out.println("Persona::ejbRemove()");
      Connection con = null;
      PreparedStatement ps = null;
      try {
                  String sql = "DELETE FROM Persona WHERE Cedula=?";
          con = getConnection();
          ps = con.prepareStatement(sql);
                  String key = (String) context.getPrimaryKey();
                  ps.setString(1,key);
          ps.executeUpdate();
      } catch (SQLException e) {
          System.out.println("SQLException:Persona:ejbRemove()");
          throw new RemoveException();
      } finally {
          try {
            if (ps != null) ps.close();
            if (con != null) con.close();
          } catch (SQLException e) {e.printStackTrace();}
      }
    }
      public void ejbStore() throws EJBException, RemoteException {
            System.out.println("Persona::ejbStore()");
      Connection con = null;
      PreparedStatement ps = null;
            try {
                  String sql = "UPDATE Persona SET Nombre=?, Apellido=?,
EdoCivil=?, Sexo=?, Nacionalidad=?, Direccion=?, Telefono=?, FechaN=?,
LugarN=?, Email=?, DirTrabajo=?, TlfTrabajo=?, Estado=?, Pais=? WHERE
Cedula=?";
                  String key = (String) context.getPrimaryKey();
                  con = getConnection();
                  ps = con.prepareStatement(sql);
                  ps.setString(1, this.Nombre);
                  ps.setString(2, this.Apellido);
                  ps.setString(3, this.EdoCivil);
                  ps.setString(4, this.Sexo);
                  ps.setString(5, this.Nacionalidad);
                  ps.setString(6, this.Direccion);
                  ps.setString(7, this.Telefono);
                  ps.setDate(8, this.FechaN);
                  ps.setString(9, this.LugarN);
```

```
                ps.setString(10, this.Email);
                ps.setString(11, this.DirTrabajo);
                ps.setString(12, this.TlfTrabajo);
                ps.setString(13, this.Estado);
                ps.setString(14, this.Pais);
                ps.setString(15, this.Cedula);
                ps.executeUpdate();
            } catch(SQLException e) {
                e.printStackTrace(); }
            finally {
                try {
                    if (ps != null) ps.close();
                    if (con != null) con.close();
            }catch (SQLException e) { e.printStackTrace();}
      }
    }
    public void setEntityContext(EntityContext context) throws
EJBException, RemoteException {
      this.context = context;
      System.out.println("Persona::setContext()");
    }
    public void unsetEntityContext() throws EJBException, RemoteException
{
      context = null;
      System.out.println("Persona::unsetContext()");
    }
    public String ejbCreate() throws RemoteException, CreateException {
      System.out.println("Persona::ejbCreate()");
        return (this.Cedula);
    }
    public void ejbPostCreate() throws RemoteException, CreateException {
      System.out.println("Persona::ejbPostCreate()");
    }
      public String ejbCreate(String Cedula, String Nombre, String
Apellido, String EdoCivil, String Sexo, String Nacionalidad, String
Direccion, String Telefono, Date FechaN, String LugarN, String Email,
String DirTrabajo, String TlfTrabajo, String Estado, String Pais) throws
RemoteException, CreateException {
        System.out.println("Persona::ejbCreate(params)");
            this.Cedula = Cedula;
            this.Nombre = Nombre;
            this.Apellido = Apellido;
            this.EdoCivil = EdoCivil;
            this.Sexo = Sexo;
            this.Nacionalidad = Nacionalidad;
            this.Direccion = Direccion;
            this.Telefono = Telefono;
            this.FechaN = FechaN;
            this.LugarN = LugarN;
            this.Email = Email;
            this.DirTrabajo = DirTrabajo;
            this.TlfTrabajo = TlfTrabajo;
            this.Estado = Estado;
            this.Pais = Pais;
      Connection con = null;
      PreparedStatement ps = null;
      try {
```

```
                String sql = "INSERT INTO Persona(Cedula, Nombre,
Apellido, EdoCivil, Sexo, Nacionalidad, Direccion, Telefono, FechaN,
LugarN, Email, DirTrabajo, TlfTrabajo, Estado, Pais) VALUES (?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
          con = getConnection();
          ps = con.prepareStatement(sql);
                  ps.setString(1, this.getCedula());
                  ps.setString(2, this.getNombre());
                  ps.setString(3, this.getApellido());
                  ps.setString(4, this.getEdoCivil());
                  ps.setString(5, this.getSexo());
                  ps.setString(6, this.getNacionalidad());
                  ps.setString(7, this.getDireccion());
                  ps.setString(8, this.getTelefono());
                  ps.setDate(9, this.getFechaN());
                  ps.setString(10, this.getLugarN());
                  ps.setString(11, this.getEmail());
                  ps.setString(12, this.getDirTrabajo());
                  ps.setString(13, this.getTlfTrabajo());
                  ps.setString(14, this.getEstado());
                  ps.setString(15, this.getPais());
          ps.executeUpdate();
      } catch (SQLException e) {
          System.out.println("SQLException:Persona:ejbCreate()");
          throw new CreateException();
      }
      finally {
          try {
          if (ps != null) ps.close();
          if (con != null) con.close();
          } catch (SQLException e) {e.printStackTrace();}
      }
            return (this.Cedula);
    }
      public void ejbPostCreate(String Cedula, String Nombre, String
Apellido, String EdoCivil, String Sexo, String Nacionalidad, String
Direccion, String Telefono, Date FechaN, String LugarN, String Email,
String DirTrabajo, String TlfTrabajo, String Estado, String Pais) throws
RemoteException, CreateException {
      System.out.println("Persona::ejbPostCreate(params)");
    }
public String ejbFindByPrimaryKey(String key) throws RemoteException,
FinderException {
        System.out.println("Persona::ejbFindByPrimaryKey()");
      Connection con = null;
      PreparedStatement ps = null;
      ResultSet rs = null;
      try {
                  String sql = "SELECT * FROM Persona WHERE Cedula=?";
          con = getConnection();
          ps = con.prepareStatement(sql);
                  Utilerias util = new Utilerias();
                  Vector v = new Vector();
                  v = (Vector) util.SepararX(key);
                  ps.setString(1, v.elementAt(0).toString());
          rs = ps.executeQuery();
          if (rs.next()) {
```

```
this.Cedula = rs.getString(1);
this.Nombre = rs.getString(2);
this.Apellido = rs.getString(3);
this.EdoCivil = rs.getString(4);
this.Sexo = rs.getString(5);
this.Nacionalidad = rs.getString(6);
this.Direccion = rs.getString(7);
this.Telefono = rs.getString(8);
this.FechaN = rs.getDate(9);
this.LugarN = rs.getString(10);
this.Email = rs.getString(11);
this.DirTrabajo = rs.getString(12);
this.TlfTrabajo = rs.getString(13);
this.Estado = rs.getString(14);
this.Pais = rs.getString(15);
            rs.close();
            ps.close();
            con.close();
            return key;
          }
      } catch (SQLException e) {

System.out.println("SQLException:Persona:ejbfindByPrimaryKey()");
          throw new FinderException();
      }
      finally {
          try {
            if (rs != null) rs.close();
            if (ps != null) ps.close();
            if (con != null) con.close();
          } catch (SQLException e) {e.printStackTrace();}
      }
      throw new ObjectNotFoundException();
    }
    public Collection ejbFindAll() throws RemoteException,
FinderException {
        System.out.println("Persona::ejbFindAll()");
      Vector v = new Vector();
      Connection con = null;
      PreparedStatement ps = null;
      ResultSet rs = null;
      try {
                String sql = "SELECT * FROM Persona ORDER BY Cedula";
          con = getConnection();
          ps = con.prepareStatement(sql);
          rs = ps.executeQuery();
          while (rs.next()) {
            v.add(rs.getString(1));
          }
      } catch (SQLException e) {
          System.out.println("SQLException:Persona:ejbfindAll()");
          throw new FinderException();
      } finally {
          try {
            if (rs != null) rs.close();
            if (ps != null) ps.close();
            if (con != null) con.close();
```

```
            } catch (SQLException e) {e.printStackTrace();}
        }
        return v;
    }
    public Connection getConnection() {

        String dbUrl = null;
        String userName = null;
        String password = null;
        String driver = null;
        Connection connection = null;
        Context initialContext;
        Context environment;

        try {
            initialContext = new InitialContext();
            environment = (Context) initialContext.lookup("java:comp/env");
            dbUrl = (String) environment.lookup("DbUrl");
            userName = ((String) environment.lookup("User")).trim();
            password = ((String) environment.lookup("Clave")).trim();
            driver = ((String) environment.lookup("DriverName")).trim();

        } catch (NamingException e) { e.printStackTrace(); };
        try {
            Class.forName(driver);
        } catch (ClassNotFoundException cnfe) {
            System.out.println("ClassNotFoundException: " +
cnfe.getMessage());
            cnfe.printStackTrace(System.out);
        }
        try {
            System.out.println("Obteniendo conexion: " + dbUrl + ": " +
userName + ": " + password);
            connection = DriverManager.getConnection(dbUrl, userName,
password);
            System.out.println("Conexion obtenida: " + connection);
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return connection;
    }
    public Collection getParamNames() {
        System.out.println("Persona::getParamNames()");
        Vector v = new Vector();
              v.add("Cedula");
              v.add("Nombre");
              v.add("Apellido");
              v.add("EdoCivil");
              v.add("Sexo");
              v.add("Nacionalidad");
              v.add("Direccion");
              v.add("Telefono");
              v.add("FechaN");
              v.add("LugarN");
              v.add("Email");
              v.add("DirTrabajo");
              v.add("TlfTrabajo");
```

```
            v.add("Estado");
            v.add("Pais");
      return v;
    }
    public Collection getParamValues() {
        System.out.println("Persona::getParamValues()");
      Vector v = new Vector();
            v.add(this.Cedula);
            v.add(this.Nombre);
            v.add(this.Apellido);
            v.add(this.EdoCivil);
            v.add(this.Sexo);
            v.add(this.Nacionalidad);
            v.add(this.Direccion);
            v.add(this.Telefono);
            v.add(this.FechaN);
            v.add(this.LugarN);
            v.add(this.Email);
            v.add(this.DirTrabajo);
            v.add(this.TlfTrabajo);
            v.add(this.Estado);
            v.add(this.Pais);
      return v;
    }
    public Collection ejbFindX(int n, String V, String Vi, String Vf,
Collection col, String campo, String ord) throws RemoteException,
FinderException {
        System.out.println("Persona::ejbFindX(params)");
      Vector v = new Vector();
      Connection con = null;
      PreparedStatement ps = null;
      ResultSet rs = null;
        if(n == 1 && V.trim().equals("*"))
          return this.ejbFindAll();
      String sql = "SELECT * " + "FROM "+"Persona";
      if(n > 0 && n < 4) {
          if( n == 1) {
            sql += " WHERE "+campo+"='"+V+"'";
          }
          if( n == 2) {
            sql += " WHERE "+campo+">='"+Vi+"' AND "+campo+"<='"+Vf+"'";
          }
          if( n == 3 ) {
            sql += " WHERE ";
            int i=0;
            for (Iterator iter=col.iterator(); iter.hasNext();) {
                if( i == (col.size()-1) )
                  sql += campo +"='" + iter.next() +"'";
                else
                  sql += campo +"='" + iter.next() + "' OR ";
                i++;
            }
          }
          if(ord != null && (!ord.equals("")))
            sql += " ORDER BY "+ord;
          try {
            con = getConnection();
```

```
            ps = con.prepareStatement(sql);
            rs = ps.executeQuery();
            while (rs.next()) {
            v.add(rs.getString(1));
            }
            return v;
          }catch (SQLException e) {
            System.out.println("SQLException:Persona:ejbfindX()");
            throw new FinderException();
          } finally {
            try {
                if (rs != null) rs.close();
                if (ps != null) ps.close();
                if (con != null) con.close();
            } catch (SQLException e) {e.printStackTrace();}
          }
      }
      return v;
    }
    public Collection ejbFindX(Vector campos, Vector valores, Vector
parametros, Vector sensibilidades) throws RemoteException,
FinderException {
        System.out.println("Persona::ejbFindX(Vectors)");
      boolean todos = false;
      Vector v = new Vector();
      Utilerias util =new Utilerias();
      Connection con = null;
      PreparedStatement ps = null;
      ResultSet rs = null;
      String sql = "SELECT * FROM Persona ";
      try {
          for(int i=0;i<valores.size();i++)
            if(valores.elementAt(i).toString().trim().equals("*"))
                todos = true;
          if( todos == false ) {
            sql += "WHERE ";
            for(int i=0;i<campos.size();i++) {
                if(i != 0)
                  sql += " AND ";
                Vector sep =new
Vector(util.SepararX(valores.elementAt(i).toString().trim(),","));
                for(int j=0;j<sep.size();j++) {
                  if(j != 0)
                      sql += " OR ";
                  String param =
parametros.elementAt(i).toString().trim();
                  String val = sep.elementAt(j).toString().trim();
                  String camp = campos.elementAt(i).toString().trim();
                  String sen =
sensibilidades.elementAt(i).toString().trim();
                  sql += camp;
                  if(param.equals("Igual")) {
                      sql += " = ";
                  }
                  if(param.equals("Distinto")) {
                      sql += " != ";
                  }
```

```
                if(param.equals("Subcadena") || param.equals("Inicia")
|| param.equals("Termina") || param.equals("Otra")) {
                    if(sen.equals("true"))
                      sql += " ~* ";
                    else
                      sql += " ~ ";
                    if(param.equals("Subcadena"))
                      sql += "'"+val+"'";
                    if(param.equals("Inicia"))
                      sql += "'^"+val+"'";
                    if(param.equals("Termina"))
                      sql += "'"+val+"$'";
                 }
                 else
                     sql += "'"+val+"'";
               }
            }
          }
          System.out.println("sql = " + sql);
          con = getConnection();
          ps = con.prepareStatement(sql);
          rs = ps.executeQuery();
          while (rs.next()) {
            v.add(rs.getString(1));
          }
          return v;
      }catch(Exception e) {
          e.printStackTrace();
          return null;
      }
    }
    public boolean Modificar(String []c) throws RemoteException {
        System.out.println("Persona::Modificar()");
      try {
            this.Nombre = c[1];
            this.Apellido = c[2];
            this.EdoCivil = c[3];
            this.Sexo = c[4];
            this.Nacionalidad = c[5];
            this.Direccion = c[6];
            this.Telefono = c[7];
            this.FechaN = Date.valueOf(c[8]);
            this.LugarN = c[9];
            this.Email = c[10];
            this.DirTrabajo = c[11];
            this.TlfTrabajo = c[12];
            this.Estado = c[13];
            this.Pais = c[14];
          return true;
      }catch(Exception e) {

System.out.println("Exception:Persona:create(Collection):llamado a
create(...)");
          return false;
      }
    }
```

```
    public String ejbCreate(String []c) throws RemoteException,
CreateException {
        System.out.println("Persona::ejbCreate(String[])");
      try {
            this.Cedula = c[0];
            this.Nombre = c[1];
            this.Apellido = c[2];
            this.EdoCivil = c[3];
            this.Sexo = c[4];
            this.Nacionalidad = c[5];
            this.Direccion = c[6];
            this.Telefono = c[7];
            this.FechaN = Date.valueOf(c[8]);
            this.LugarN = c[9];
            this.Email = c[10];
            this.DirTrabajo = c[11];
            this.TlfTrabajo = c[12];
            this.Estado = c[13];
            this.Pais = c[14];

      this.ejbCreate(this.Cedula,this.Nombre,this.Apellido,this.EdoCivil,
this.Sexo,this.Nacionalidad,this.Direccion,this.Telefono,this.FechaN,this
.LugarN,this.Email,this.DirTrabajo,this.TlfTrabajo,this.Estado,this.Pais)
;
            return (this.Cedula);
      }catch(Exception e) {

System.out.println("Exception:Persona:create(Collection):llamado a
create(...)");
        throw new CreateException();
      }
    }
    public void ejbPostCreate(String []c) throws RemoteException,
CreateException {
      System.out.println("Persona::ejbPostCreate(String[])");
    }
    public boolean set(String name, String value) {
        System.out.println("Persona::set()");
        try {
            if(name.equals("Nombre")) {
                  this.setNombre(value);
            return true;
      }
            if(name.equals("Apellido")) {
                  this.setApellido(value);
            return true;
      }
            if(name.equals("EdoCivil")) {
                  this.setEdoCivil(value);
            return true;
      }
            if(name.equals("Sexo")) {
                  this.setSexo(value);
            return true;
      }
            if(name.equals("Nacionalidad")) {
                  this.setNacionalidad(value);
```

```
                return true;
        }
                if(name.equals("Direccion")) {
                        this.setDireccion(value);
                return true;
        }
                if(name.equals("Telefono")) {
                        this.setTelefono(value);
                return true;
        }
                if(name.equals("FechaN")) {
                        this.setFechaN(Date.valueOf(value));
                return true;
        }
                if(name.equals("LugarN")) {
                        this.setLugarN(value);
                return true;
        }
                if(name.equals("Email")) {
                        this.setEmail(value);
                return true;
        }
                if(name.equals("DirTrabajo")) {
                        this.setDirTrabajo(value);
                return true;
        }
                if(name.equals("TlfTrabajo")) {
                        this.setTlfTrabajo(value);
                return true;
        }
                if(name.equals("Estado")) {
                        this.setEstado(value);
                return true;
        }
                if(name.equals("Pais")) {
                        this.setPais(value);
                return true;
        }
                return false;
          }catch(Exception e) {
                e.printStackTrace();
                return false;
          }
    }
}
```

Listado 4: Código ejemplo: Interfaz home de un Bean de Sesión.

```
package interfaces;

import java.rmi.RemoteException;
```

```
import javax.ejb.CreateException;
import javax.ejb.EJBHome;

public interface AccesoBeanHome extends EJBHome {

      AccesoBeanRemote create() throws RemoteException, CreateException;

}
```

Listado 5: Código ejemplo: Interfaz remota de un Bean de Sesión.

```
package interfaces;

import javax.ejb.EJBObject;

import java.rmi.RemoteException;

public interface AccesoBeanRemote extends EJBObject {

      public String getTipo() throws RemoteException;
      public String darAcceso(String nombre, String clave) throws
RemoteException;
      public boolean getEstado() throws RemoteException;

}
```


Listado 6: Código ejemplo: Clase de un Bean de Sesión.

```
package ejb;

import interfaces.UsuarioSistemaHome;
import interfaces.UsuarioSistemaRemote;

import java.rmi.RemoteException;
import java.util.Properties;

import javax.ejb.SessionBean;
import javax.ejb.SessionContext;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.rmi.PortableRemoteObject;
```

```
public class AccesoBean implements SessionBean {

      private Properties properties;
      private InitialContext jndiContext;
      private Object ref;
      private UsuarioSistemaHome home;
      private UsuarioSistemaRemote remote;
      private boolean estado = false;

      public String getTipo() throws RemoteException {
            return remote.getTipoUsuario();
      }

      public boolean getEstado() throws RemoteException {
            return this.estado;
      }

      public String darAcceso(String nombre, String clave) throws
RemoteException {
            if(this.estado = true && nombre != null && clave != null) {
                  try {
                        remote = home.findByPrimaryKey(nombre);
                        if( clave.equals(remote.getClave()) )
                              return remote.getCedula();
                  }catch(Exception e) {
                        return null;
                  }
            }
            return null;
      }

      public void ejbCreate() {
            properties = new Properties();
            properties.put(Context.INITIAL_CONTEXT_FACTORY,
"org.jnp.interfaces.NamingContextFactory");
            properties.put(Context.PROVIDER_URL, "localhost:1099");
            try {
                  InitialContext jndiContext = new
InitialContext(properties);
                  ref  = jndiContext.lookup("UsuarioSistemaEJB");
            }
            catch(Exception e) {
                  System.out.println("Exception en metodo create:
Referencia no obtenida");
                  this.estado = false;
            }
            try {
                  home = (UsuarioSistemaHome) PortableRemoteObject.narrow
(ref, UsuarioSistemaHome.class);
            }
            catch(Exception e) {
                  System.out.println("Exception en metodo create:
Referencias a las interfaces no obtenida");
                  this.estado = false;
            }
            this.estado = true;
      }
```

```
	public void ejbRemove() {}

	public void ejbActivate() {}

	public void ejbPassivate() {}

	public void setSessionContext(SessionContext sc) {}

}
```

Listado 7: Código ejemplo: AccesoServlet.java.

```
/*
 * Created on 07-ene-2005
 *
 * To change the template for this generated file go to
 * Window - Preferences - Java - Code Generation - Code and Comments
 */
package web;

import interfaces.AccesoBeanHome;
import interfaces.AccesoBeanRemote;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Properties;

import javax.naming.Context;
import javax.naming.InitialContext;
import javax.rmi.PortableRemoteObject;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * @author jmatheus
 *
 * To change the template for this generated type comment go to
 * Window - Preferences - Java - Code Generation - Code and Comments
 */
public class AccesoServlet extends HttpServlet {

	public AccesoServlet() {
		super();
	}

	public void init() throws ServletException {
		super.init();
	}
```

```
      protected void doGet(HttpServletRequest request,
HttpServletResponse response)
      throws ServletException, IOException {
            String user, password;
            boolean res = false;
            Properties properties = new Properties();
            properties.put(Context.INITIAL_CONTEXT_FACTORY,
"org.jnp.interfaces.NamingContextFactory");
            properties.put(Context.PROVIDER_URL, "localhost:1099");
            response.setContentType("text/html");
            PrintWriter out = response.getWriter();
            try {
                  InitialContext jndiContext = new
InitialContext(properties);
                  Object ref  = jndiContext.lookup("AccesoEJB");
                  AccesoBeanHome home = (AccesoBeanHome)
PortableRemoteObject.narrow (ref, AccesoBeanHome.class);
                  user = request.getParameter("NombreUsuario");
                  password = request.getParameter("Clave");
                  if( user == null || password == null || user.equals("")
|| password.equals("")) {
                        response.sendRedirect("http://www.yahoo.com");
                  }
                  else {
                        try {
                              AccesoBeanRemote remote = home.create();
                              try {
                                    res = remote.darAcceso(user,password);
                                    if( res == true) {
                                          try {

      cargarInterfaz(response,remote.getTipo());
                                          }catch(Exception e) {

      response.sendRedirect("http://www.hotmail.com");
                                          }
                                    }
                                    else

      response.sendRedirect("http://localhost:8080/pgcomp/");
                                    }catch(Exception e) {

      response.sendRedirect("http://localhost:8080/pgcomp/");
                                    }
                              }catch(Exception e) {

      response.sendRedirect("http://www.hotmail.com");
                              }
                  }
            }
            catch(Exception e) {
                  response.sendRedirect("http://www.hotmail.com");
            }
      }
      protected void doPost(HttpServletRequest request,
HttpServletResponse response)
```

```
      throws ServletException, IOException {
            doGet(request,response);
      }

      public void cargarInterfaz(HttpServletResponse response, String
tipoUsuario) throws IOException {
            if(tipoUsuario.equals("Adm")) {
                  response.sendRedirect("administrador.jsp");
            }
            else
                  if(tipoUsuario.equals("Est")) {
                        response.sendRedirect("estudiante.jsp");
                  }
                  else
                        if(tipoUsuario.equals("Prof")) {
                              response.sendRedirect("profesor.jsp");
                        }
                        else {

      response.sendRedirect("http://www.google.com");
                        }

      }

}
```

Listado 8: Código ejemplo: formbusqueda1.jsp.


```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 TRANSITIONAL//EN">
<HTML>
<HEAD>
<TITLE> Formulario para Búsqueda de Entidades </TITLE>
</HEAD>
<% session.setAttribute("Title","Búsqueda en grupo: Personal"); %>
<% String accion = (String) request.getParameter("Accion");%>
<%! String t = null; %>
<center>
<%@ page import="javax.servlet.http.Cookie"%>
<%@ page import="util.Utilerias"%>
<% Utilerias util = new Utilerias();%>
<% String color = (String)
util.getCookieValue(request.getCookies(),"Color2");%>
<BODY bgcolor="<%=color%>">
<h2> <%=session.getAttribute("Title")%> </h2><p><br><p><br><p><br>
<form action="formbusquedaentidades2.jsp">
Seleccione la Entidad <%=util.getSelect("TipoEntidad")%>
<option value="Persona">Persona
<option value="UsuarioSistema">Usuario del Sistema
<option>Becario
<option value="PersonalFijo">Personal Fijo
<option value="PersonalContratado">Personal Contratado
<option>Profesor
```

```
<option>Estudiante
</optgroup>
<option value="UsuarioMB">Usuario de Material Bibliografico
</select>
<%=util.getBotonSubmit("Accion","Insertar")%>
<%=util.getBotonSubmit("Accion","Buscar")%>
</form>
</body>
</center>
</HTML>
```

Apéndice E

Listado 1: Prueba de un Bean de Entidad.

```
package client;

import interfaces.EntidadHome;
import interfaces.EntidadRemote;
import javax.ejb.CreateException;
import javax.ejb.FinderException;
import javax.ejb.RemoveException;
import javax.naming.*;
import javax.rmi.PortableRemoteObject;

import java.rmi.RemoteException;
import java.util.Collection;
import java.util.Iterator;
import java.util.Properties;

public class EntidadBeanClient {
    public static void main(String[] args) {
      Collection personas;
      // PREPARANDO UN OBJETO PROPERTIES PARA
      // CONSTRUIR UN OBJETO INITIAL CONTEXT
      Properties properties = new Properties();
      properties.put(Context.INITIAL_CONTEXT_FACTORY,
"org.jnp.interfaces.NamingContextFactory");
      properties.put(Context.PROVIDER_URL, "localhost:1099");
      try {
          System.out.println("Inicio casos de prueba");
          // OBTENIENDO EL CONTEXTO INICIAL (DONDE ESTAN LOS OJBETOS)
          InitialContext jndiContext = new InitialContext(properties);
          System.out.println("Contexto Inicial obtenido 1");
          // OBTENIENDO LA REFERENCIA
          Object ref  = jndiContext.lookup("EntidadEJB");
          System.out.println("Referencia obtenida 2");
          // REFERENCIA A LA INTERFAZ HOME
          EntidadHome home = (EntidadHome) PortableRemoteObject.narrow
(ref, EntidadHome.class);
          System.out.println("Referencia a la interfaz Home obtenida 3");
          // PROBANDO CREATE
          System.out.println("Probando create 4");
          try {
            EntidadRemote p = null;
            try {
                p =  (EntidadRemote) home.findByPrimaryKey("13932871");
            }catch(FinderException e) {
                System.out.print("");
            }
            if( p == null ) {
                p = (EntidadRemote) home.create("13932871","JULIO
DANIEL","MATHEUS F.");
                System.out.println("Entidad creada exitosamente");
            }
```

```
            else
                System.out.println("Entidad no fue creada");
          }catch(CreateException e) {
            System.out.println("Entidad ya existe");
          }
          catch(RemoteException e) {
            System.out.println("Error en conexion remota : " +
e.getMessage());
          }
          // PROBANDO CREATE CON ENTIDAD EXISTENTE
          System.out.println("Probando create con entidad existente 5");
          try {
            EntidadRemote p = null;
            try {
                p =  (EntidadRemote) home.findByPrimaryKey("13932871");
            }catch(FinderException e) {
                System.out.print("");
            }
            if( p == null ) {
                p = (EntidadRemote) home.create("13932871","JULIO
DANIEL","MATHEUS F.");
                System.out.println("Entidad creada exitosamente");
            }
            else
                System.out.println("Entidad no fue creada");
          }catch(CreateException e) {
            System.out.println("Entidad ya existe");
          }
          catch(RemoteException e) {
            System.out.println("Error en conexion remota : " +
e.getMessage());
          }
          // PROBANDO FINDBYPRIMARYKEY CON ENTIDAD EXISTENTE
          System.out.println("Probando findByPrimaryKey con entidad
existente 6");
          try {
            EntidadRemote p = null;
            p = (EntidadRemote) home.findByPrimaryKey("13932871");
            System.out.println("Cedula: " + p.getCedula());
            System.out.println("Nombre: " + p.getNombre());
            System.out.println("Apellido: " + p.getApellido());
          }catch(FinderException e) {
            System.out.println("Entidad no existe");
          }
          catch(RemoteException e) {
            System.out.println("Error en conexion remota : " +
e.getMessage());
          }
          // PROBANDO FINDBYPRIMARYKEY CON ENTIDAD INEXISTENTE
          System.out.println("Probando findByPrimaryKey con entidad
inexistente 7");
          try {
            EntidadRemote p = null;
            p = (EntidadRemote) home.findByPrimaryKey("9999999");
            System.out.println("Cedula: " + p.getCedula());
            System.out.println("Nombre: " + p.getNombre());
            System.out.println("Apellido: " + p.getApellido());
```

```
          }catch(FinderException e) {
            System.out.println("Entidad no existe");
          }
          catch(RemoteException e) {
            System.out.println("Error en conexion remota : " +
e.getMessage());
          }
          // PROBANDO FINDALL CON TABLA NO VACIA
          System.out.println("Probando findAll con tabla no vacia 8");
          try {
            EntidadRemote p = null;
            personas = home.findAll();
            for (Iterator iter=personas.iterator(); iter.hasNext();) {
                p = (EntidadRemote) iter.next();
                System.out.println("Cedula: " + p.getCedula());
                System.out.println("Nombre: " + p.getNombre());
                System.out.println("Apellido: " + p.getApellido());
            }
          }catch(FinderException e) {
            System.out.println("Tabla vacia");
          }
          catch(RemoteException e) {
            System.out.println("Error en conexion remota : " +
e.getMessage());
          }
          // PROBANDO ACTUALIZAR
          System.out.println("Probando Actualizar 9");
          try {
            EntidadRemote p = null;
            // PRIMERO OBTENEMOS LA REFERENCIA
            p = (EntidadRemote) home.findByPrimaryKey("13932871");
            System.out.println("Antes");
            System.out.println("Cedula: " + p.getCedula());
            System.out.println("Nombre: " + p.getNombre());
            System.out.println("Apellido: " + p.getApellido());
            // LUEGO CAMBIAMOS EL CAMPO NOMBRE
            p.setNombre("JULIO RAMON");
            // Y AHORA, BUSCAMOS Y DESPLEGAMOS EL OBJETO
            p = (EntidadRemote) home.findByPrimaryKey("13932871");
            System.out.println("Despues");
            System.out.println("Cedula: " + p.getCedula());
            System.out.println("Nombre: " + p.getNombre());
            System.out.println("Apellido: " + p.getApellido());
          }catch(FinderException e) {
            System.out.println("Entidad no existe");
          }
          catch(RemoteException e) {
            System.out.println("Error en conexion remota : " +
e.getMessage());
          }
          // PROBANDO REMOVE ENTIDAD EXISTENTE
          System.out.println("Probando remove con entidad existente 10");
          try {
            EntidadRemote p = null;
            try {
                p = (EntidadRemote) home.findByPrimaryKey("13932871");
            }catch(FinderException e) {
```

```
                System.out.print("");
              }
              if( p != null ) {
                  p.remove();
                  System.out.println("Entidad eliminada");
              }
              else
                  System.out.println("Entidad no existe");
            }catch(RemoveException e) {
              System.out.println("Entidad no existe");
            }
            catch(RemoteException e) {
              System.out.println("Error en conexion remota : " +
e.getMessage());
            }
            // PROBANDO REMOVE ENTIDAD INEXISTENTE
            System.out.println("Probando remove con entidad inexistente
11");
            try {
              EntidadRemote p = null;
              try {
                  p = (EntidadRemote) home.findByPrimaryKey("99999999");
              }catch(FinderException e) {
                  System.out.print("");
              }
              if( p != null ) {
                  p.remove();
                  System.out.println("Entidad eliminada");
              }
              else
                  System.out.println("Entidad no existe");
            }catch(RemoveException e) {
              System.out.println("Entidad no existe");
            }
            catch(RemoteException e) {
              System.out.println("Error en conexion remota : " +
e.getMessage());
            }
            System.out.println("Ultimo Despliegue: ");
            try {
              EntidadRemote p = null;
              personas = home.findAll();
              for (Iterator iter=personas.iterator(); iter.hasNext();) {
                  p = (EntidadRemote) iter.next();
                  System.out.println("Cedula: " + p.getCedula());
                  System.out.println("Nombre: " + p.getNombre());
                  System.out.println("Apellido: " + p.getApellido());
              }
            }catch(FinderException e) {
              System.out.println("Tabla vacia");
            }
            catch(RemoteException e2) {
              System.out.println("Error en conexion remota : " +
e2.getMessage());
            }
            System.out.println("Fin de casos de prueba");
      }
```

```
      catch(Exception ef) {
          System.out.println(ef.toString());
      }
    }
}
```

Listado 2: Salida generada por el programa anterior.

```
Inicio casos de prueba
Contexto Inicial obtenido 1
Referencia obtenida 2
Referencia a la interfaz Home obtenida 3
Probando create 4
Entidad creada exitosamente
Probando create con entidad existente 5
Entidad no fue creada
Probando findByPrimaryKey con entidad existente 6
Cedula: 13932871
Nombre: JULIO DANIEL
Apellido: MATHEUS F.
Probando findByPrimaryKey con entidad inexistente 7
Entidad no existe
Probando findAll con tabla no vacia 8
Cedula: 13932871
Nombre: JULIO DANIEL
Apellido: MATHEUS F.
Cedula: 14325693
Nombre: JOSELYN
Apellido: CUEVAS F.
Cedula: 14695478
Nombre: JAVIER EDUARDO
Apellido: MATHEUS F.
Cedula: 15698725
Nombre: JEANNYS DAVID
Apellido: CUEVAS F.
Probando Actualizar 9
Antes
Cedula: 13932871
Nombre: JULIO DANIEL
Apellido: MATHEUS F.
Despues
Cedula: 13932871
Nombre: JULIO RAMON
Apellido: MATHEUS F.
Probando remove con entidad existente 10
Entidad eliminada
Probando remove con entidad inexistente 11
Entidad no existe
Ultimo Despliegue:
Cedula: 14325693
```

```
Nombre: JOSELYN
Apellido: CUEVAS F.
Cedula: 14695478
Nombre: JAVIER EDUARDO
Apellido: MATHEUS F.
Cedula: 15698725
Nombre: JEANNYS DAVID
Apellido: CUEVAS F.
Fin de casos de prueba
```