ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

TÓPICO DE GRADUACIÓN

"Módulo de Administración de E-Guana"

Previa a la obtención del Título de:

INGENIERO EN COMPUTACIÓN

Presentada por:

Paola Tatiana Acosta Carvajal

María de Lourdes Barreno Valdivieso

Andrés Fernando Guerrero Vasco

GUAYAQUIL – ECUADOR

2007

DEDICATORIA

Dedicado a nuestros padres quienes con su apoyo hicieron posible el llegar hasta aquí y cumplir nuestras metas.

TRIBUNAL DE GRADUACIÓN

_____	_____
Ing. Holger Cevallos	Ing. Luis Muñoz
SUB-DECANO DE LA FIEC	DIRECTOR DE TÓPICO
_____	_____
Ing. Otilia Alejandro	Ing. Mónica Villavicencio
MIEMBRO PRINCIPAL	MIEMBRO PRINCIPAL

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de este Proyecto, nos corresponde exclusivamente; y el patrimonio intelectual de la misma a la Escuela Superior Politécnica del Litoral"

(Reglamento de exámenes y títulos profesionales de la ESPOL)

Paola Tatiana Acosta Carvajal

María de Lourdes Barreno Valdivieso

Andrés Fernando Guerrero Vasco

RESUMEN

El sistema E-Guana que se describirá a continuación, es una solución E-Procurement desarrollada por los estudiantes del Tópico de Graduación “Desarrollo de Aplicaciones Transaccionales con Java y XML”.

El sistema demuestra el uso del espectro completo de la tecnología J2EE (Java 2 Enterprise Edition) en un proyecto complejo, esta tecnología fue estudiada extensamente en el transcurso del Tópico, convirtiéndose así este documento en una guía práctica para aquellos desarrolladores interesados en utilizarla.

La organización de este documento se detalla a continuación:

En el primer capítulo “Justificación del Sistema” se incluye los antecedentes, conceptos del E-procurement así como los objetivos de E-Guana.

En el segundo capítulo “Análisis y Diseño del Sistema” se detalla la arquitectura utilizada, los requerimientos básicos y mínimos del sistema y breves conceptos de la tecnología utilizada.

El tercer capítulo “Definición e Implementación del Módulo de Administración”, describe la estructura de elaboración del módulo, así como también el diseño y parte de la configuración del entorno realizada.

El cuarto capítulo “Definición e Implementación del Módulo de Pago de Transacciones” explica el análisis, diseño e implementación de este módulo, adicionalmente se describe la forma de interacción con los demás módulos del sistema.

En el quinto capítulo denominado “Pruebas” se detallan los diferentes tipos de pruebas realizadas sobre partes del código del sistema y sus respectivos resultados.

Para finalizar, se presentan las conclusiones y recomendaciones, glosario, anexos y bibliografía.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCION

El presente trabajo tiene como objetivo dar a conocer las múltiples ventajas de uso de una tecnología que proporciona un mercado seguro, libre de barreras geográficas y de horario.

E-Guana es una iniciativa de implementación de un sistema E-Procurement.

El E-Procurement pretende unir en un mercado virtual a pequeñas y medianas empresas para aumentar su poder de negociación en operaciones de compra-venta, a través de licitaciones, subastas, etc. E-Guana implementa E-Procurement usando la tecnología J2EE.

Este proyecto comprende la implementación tanto del Módulo de Administración como del Módulo de Pagos de Transacciones de E-Guana.

El Módulo de Administración permite el manejo y configuración del sistema, desde el mantenimiento de variables del sistema hasta el registro de empresas y mantenimiento del catálogo de productos de dichas empresas.

El Módulo de Pagos de Transacciones permite el mantenimiento de las cuentas bancarias de las empresas y el control sobre los débitos y créditos resultantes de las transacciones de compra-venta, validando los saldos

actuales de las empresas antes de efectuar dichas transacciones sobre nuestra plataforma.

CAPÍTULO 1

1. Justificación del sistema

¿Quién no ha efectuado una compra? ¿Quién no se ha preguntado por qué no se podían hacer compras desde su casa? ¿Y quién cuando tuvo esta funcionalidad no se preguntó por qué tengo que comprar a un solo proveedor? Todas estas preguntas han sido contestadas y desarrolladas con la rápida evolución del proceso de compras y el comercio electrónico. El comercio electrónico está transformando el modelo de gestión de las empresas según se van adaptando e incorporando a la nueva economía. Dentro de esta transformación hay que prestar especial atención a la gestión de las compras que aunque a primera vista puede resultar algo secundario, es la columna vertebral de cualquier negocio.

En este marco surge un nuevo concepto, como un sistema que permite la integración de los procesos de compra y su automatización mediante tecnología Web.

En este primer capítulo se describirá brevemente de la tecnología a utilizar para el desarrollo del sistema, así mismo sus beneficios y el por qué el E-Procurement tiene una tremenda ventaja sobre el uso de sistemas convencionales de compra-venta.

1.1. Antecedentes

Se iniciará indicando que “con la evolución actual de la tecnología, el mantener la competitividad es algo primordial entre las empresas, los elementos críticos para lograrla son la comunicación de los datos del cliente y las transacciones que se realizan” [1], por lo que es necesario tener una herramienta o tecnología que permita un manejo adecuado de estos dos elementos.

“El beneficio principal que ofrecen los negocios electrónicos es hacer más ágil y rápida la comunicación entre el cliente y la fuente (el producto)” [1], por ende hacer más eficiente el proceso de compra para comodidad del cliente, eliminando costos operacionales en los cuales se pudiera incurrir como por ejemplo: transporte, papelería.

El tratar de justificar la adquisición de una herramienta de negocios electrónicos conlleva a tener que calcular, en base a ahorros monetarios que se generan a partir del uso de ésta, el retorno sobre la inversión realizada, lo cual es un poco complejo debido a las diferentes variables que intervienen en

este análisis. El análisis puede estar basado en que la capacidad de generar ahorros está directamente relacionada con la mayor agilidad en los ciclos que conlleva cerrar un negocio, la reducción del tiempo que se dedican a tareas administrativas y por ende reducción de costos operacionales.

El E-Procurement es “una de las tantas herramientas que han surgido para incrementar las ganancias de la empresa y optimizar la comunicación vía transaccional” [3], por lo que, aunque no es la única alternativa, una solución de este tipo será bienvenida por cualquier empresa que incursiona en el negocio electrónico que pretenda estos beneficios y por lo tanto se justifica su implementación.

1.2. Definición

Una vez que se ha indicado con claridad la importancia de los sistemas de comercio electrónico para las empresas, se puede definir formalmente lo que es E-Guana y E-Procurement. Sin embargo previo a esto se recordarán las etapas de un proceso de compra.

El proceso de compra se plantea cuando surge una necesidad que puede ser resuelta con la adquisición de un servicio o producto. Se plantean las características del producto con la participación de expertos asesores. Esta etapa concluye con el envío de una solicitud al departamento de compras.

Se efectúa la búsqueda de productos similares que cumplan las características establecidas, a los cuales se los evalúa en distintos aspectos: calidad, precio, marca, proveedor, valor agregado, etc.

Una vez seleccionado el producto, se efectuará el pedido y se lo evaluará en rendimiento y cumplimiento de quien lo provee; a fin de decidir si dicho proveedor se convierte en el suministrador de compras posteriores.

Ahora bien, este proceso, llevado al Web en esencia sigue siendo el mismo, pero cada paso o etapa se completa con el apoyo de las Tecnologías de Información y Comunicación.

Además se obtienen varios beneficios según E-Business Consultores [22]:

- Disminuir costos.
- Reducir los procesos de compra en tiempo y complejidad.
- Aumentar la comunicación entre empleados, clientes y proveedores.
- Incrementar el mercado.

De lo anterior surgió el E-Commerce o comercio electrónico, que es “la integración de los procesos de comercialización de productos de una empresa en un sitio Web” [22].

Adicionalmente, el esquema de que el Internet era un recurso sólo para publicidad se está dejando a un lado, ahora es un elemento vital y requerido en las relaciones con clientes y proveedores.

Y es aquí donde surgen muchas incógnitas, en especial para la empresa que quiere conectarse con todos quienes conforman su círculo de negocios. “Al momento de querer efectuar transacciones automáticas, empiezan las grandes necesidades de nuevas tecnologías, que junto al hecho de que la tecnología es una de las culturas más difíciles de incorporar en una empresa” [8] (en este medio es aún más notorio y debe ser tomado en consideración en todas las etapas de desarrollo del sistema), hacen que la implantación de un sistema de información de esta naturaleza se convierta en un reto mayor pero que si se sopesa con los beneficios que se pueden obtener, vale la pena enfrentar.

1.2.1. Sistema

Normalmente se encuentra en Internet mercados o tiendas de compra directa, es decir, sitios donde el cliente navega por un catálogo de productos y adquiere uno o más de éstos de acuerdo a sus necesidades o las de su empresa. Con el objetivo de estar un paso más adelante en el proceso de compra-venta de productos y servicios, el sistema E-Guana hace uso de la tecnología de E-Procurement. Una red de E-Procurement “ofrece la posibilidad de integrar todos los procesos de compra de la organización,

generando una sinergia entre todas las áreas de la empresa, que conduce a la optimización y rentabilidad de los procesos relativos a la vida de la compañía” [3]. Lo anterior establece claramente la importancia de un sistema de E-Procurement, así como su diferencia con otros sistemas de comercio electrónico.

En el siguiente punto se verá con mayor detalle lo que es E-Procurement.

1.2.2. E-Procurement

En esta sección se describirá en detalle el E-Procurement y se verá como lleva las compras usando Internet a otro nivel.

Se iniciará con la definición, “E-Procurement es la sistematización de los procesos internos relacionados con la compra de productos de una empresa como el proceso de selección, requerimiento, autorización, pago y compra final” [2]. La siguiente característica del E-Procurement es lo que lo diferencia de otros sistemas de compras por Internet, “permite a una empresa compradora realizar todo tipo de operaciones de compra (licitaciones, subastas y compras por catálogo)” [2]. Con la sistematización se asegura que el producto esté disponible en el momento necesario para el cliente y que existe una comunicación ágil con el proveedor. Así mismo se asegura que el proveedor obtenga toda la información necesaria del cliente para que el

manejo de la compra sea más eficiente. Con esto se logran beneficios para ambas partes.

Otra característica del E-Procurement es “permitir a las empresas administrar sus operaciones internas y tomar ventaja del concepto de *business to business* (comercio entre empresas) ó B2B[1] y del E-Commerce (comercio electrónico) de forma que se obtenga rápidas y eficientes transacciones con proveedores, reduciendo costos y tiempos asociados con la compra de bienes o servicios que la empresa necesita para operar” [22], se puede deducir entonces, que gracias al E-Procurement se eliminan las tareas sin valor agregado, se reduce el volumen de papeles, se eliminan errores de comunicación, se impide que se realicen compras a proveedores no aprobados o certificados y el comprar o contratar productos que no consten en los catálogos.

“El E-Procurement explota el B2B colaborativo, donde con el uso del Internet se logra integrar sistemas y procesos de compañías que en muchos casos operan en colaboración” [22], este trabajo colaborativo entre las empresas resulta en “catálogos electrónicos que mejoran la comunicación de nuevos productos estandarizados, la aceleración de las relaciones dentro de la Cadena de Abastecimiento, la implementación de tarjetas electrónicas de

compra, y el planeamiento conjunto" [3], por lo anotado, un sistema de E-Procurement trae grandes beneficios tanto para la empresa compradora como para la proveedora.

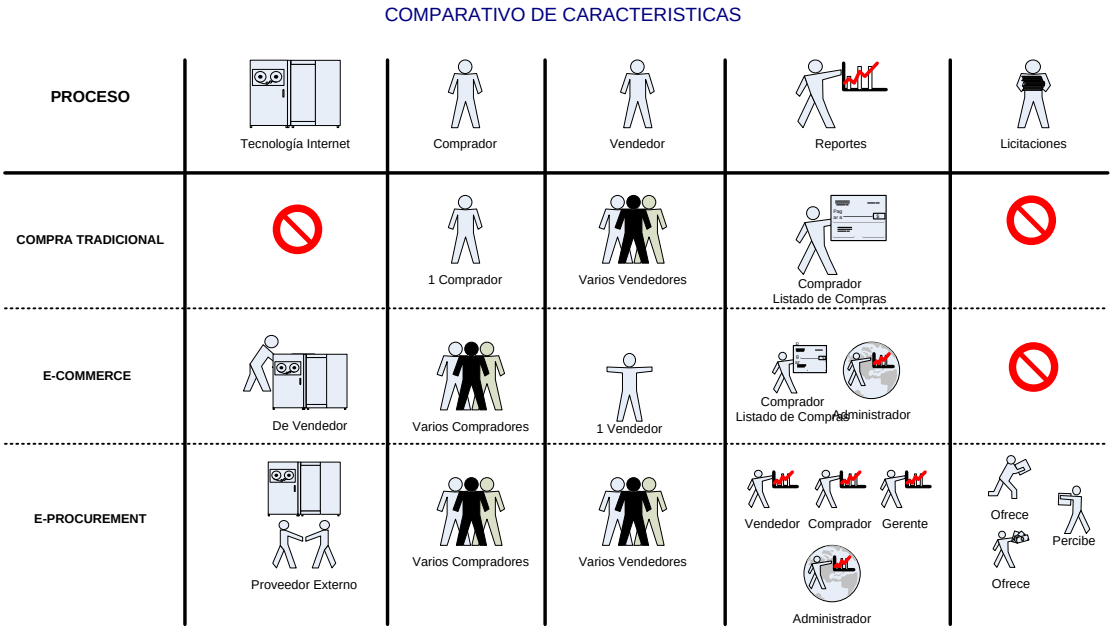


Figura 1-1Comparativo de características de los procesos de compra utilizando diferentes metodologías.
Fuente: Artículos publicados en http://www.eduardoleyton.com, http://eprocurement.interneturl.com.ar y http://www.ebconsultores.com.mx.

En las relaciones comerciales tradicionales, era posible distinguir un contenido que estaba conformado por los bienes o servicios que se intercambiaban; un escenario" en el cual se desarrollaban estas actividades (el lugar de reunión de compradores y vendedores); y las herramientas utilizadas para realizar este intercambio económico entre partes. Sin

[1] B2B: Es la forma contemporánea de llamar a la práctica de ventas denominada (business-to-business) negocio-a-negocio. Las transacciones B2B tienen como objetivo a compañías y compradores mayoristas.

embargo, esta estructura tradicional está modificándose sustancialmente, en E-Procurement “la oferta de materiales y servicios de los proveedores y la demanda de los mismos por parte de los compradores pueden instrumentarse a través de nuevas formas de comunicación: los catálogos electrónicos, las solicitudes de cotización electrónicas, las licitaciones y las subastas electrónicas” [2].

En cuanto a los sistemas de catálogos electrónicos se puede decir que “permiten el manejo de materiales y servicios de diferentes proveedores, incluyendo información comercial, logística, y especificaciones técnicas” [2]. Además estos catálogos “pueden configurarse en forma personalizada por comprador o por proveedor, lo que facilita el seguimiento y control de acuerdos entre empresas” [2], permitiendo identificar claramente los precios acordados, plazos de entrega, tipo de embalaje requerido, etc. Este tipo de catálogos tiene además la gran ventaja de poder informar al solicitante el stock de unidades disponibles del material o equipo buscado. Por otra parte, facilita la adquisición de determinado ítem a través de la selección de diferentes criterios de búsqueda lo que acelera la tarea.

Las licitaciones pueden incluir pliegos de condiciones, fecha de apertura de ofertas, manteniendo seguridad y confidencialidad de los datos ofertados por cada proveedor, además “la realización de licitaciones y subastas electrónicas proporciona a las empresas acceso en línea a toda la

comunidad de negocio, ofreciendo ganancias considerables de productividad, ciclos más cortos y costes reducidos” [2].

A continuación se listarán las 3 formas de implementar una solución de E-Procurement [1]:

- Relacionando un vendedor con varios compradores
- Un comprador y varios vendedores
- Varios compradores con varios vendedores

La primera forma de implementar E-Procurement es relacionando un vendedor con varios compradores. En este caso “el vendedor tiene su propio sitio donde los compradores pueden ingresar, efectuar búsqueda de productos y comprar en-línea” [4], en cuanto al mantenimiento de este sistema se indica que “debe ser realizado por el propio vendedor o por una empresa especializada” [4]. La siguiente figura ilustra este escenario:

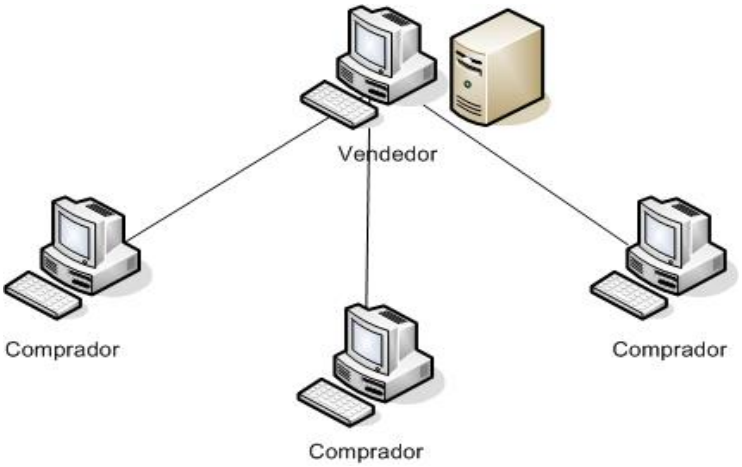


Figura 1-2 Relación de un vendedor con varios compradores. Estrategia de Implementación de E-Procurement.
Fuente: Figura adaptada de NEEF, Dale. E-Procurement from Strategy to Implementation, New Jersey: Prentice Hall 2001. p. 74-93.

Analizando la figura se puede ver que la ventaja de esta arquitectura, es que el vendedor puede mantener sus propios catálogos electrónicos. El mantenimiento inmediato facilita el proceso de compra-venta.

Para el comprador, entre tanto, esa ventaja del vendedor puede ocasionar una dificultad. Puede ser difícil integrar el sistema externo del vendedor con un sistema interno que posea el comprador. Lo cual dificultaría el proceso de compra por la no integración de la información y se acabaría en el proceso tradicional con uso de faxes, teléfono, etc.

La segunda tipología de arquitectura sucede cuando se tiene un comprador con varios vendedores. En este caso “el comprador mantiene su propio sistema de compra electrónica” [4]. Se puede identificar que la principal ventaja para el comprador es su integración con un sistema gerencial interno

y la desventaja el mantenimiento de dicho sistema. La siguiente figura ilustra este escenario:

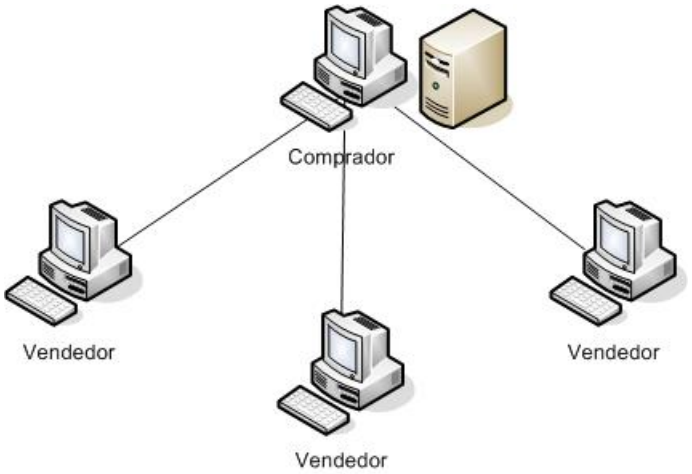


Figura 1-3 Relación de un comprador con varios vendedores. Estrategia de Implementación de E-Procurement.
Fuente: Figura adaptada de NEEF, Dale. E-Procurement from Strategy to Implementation, New Jersey: Prentice Hall 2001. p. 74-93

La última forma de arquitectura es cuando varios compradores se relacionan con varios vendedores, “esta arquitectura (la cual es usada por el sistema E-Guana) es posible normalmente a través de mercados electrónicos, que son comunidades electrónicas formadas por vendedores y compradores con necesidades comunes” [5]. La siguiente figura representa esta forma de arquitectura:

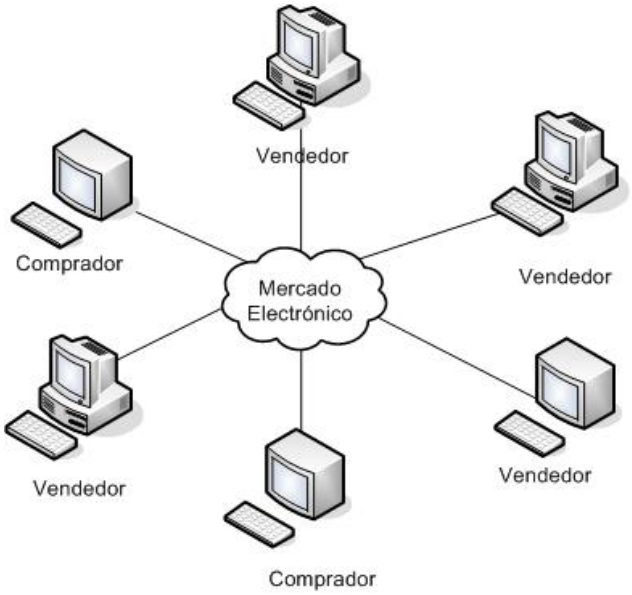


Figura 1-4 Relación de varios compradores – varios vendedores. Estrategia de Implementación de E-Procurement.
Fuente: Figura adaptada de NEEF, Dale. E-Procurement from Strategy to Implementation, New Jersey: Prentice Hall 2001. p. 74-93

Esta arquitectura implica el envío de información directa de los productos/servicios a clientes/proveedores potenciales lo cual "asegura la confiabilidad de los datos que se transfieran, cumpliendo con los parámetros establecidos para los negocios *business to business* (B2B)" [3], esto es algo importante a destacar ya que se le puede asegurar a las empresas que sus datos están protegidos y que sólo podrán ser accedidos por los usuarios pertinentes diluyendo cualquier temor de ingreso al mercado electrónico del sistema, así mismo, debido a que esta comunicación es hecha a través del Internet, el servicio de E-Procurement posee la ventaja de que no importan las distancias geográficas para poder realizar la transacción.

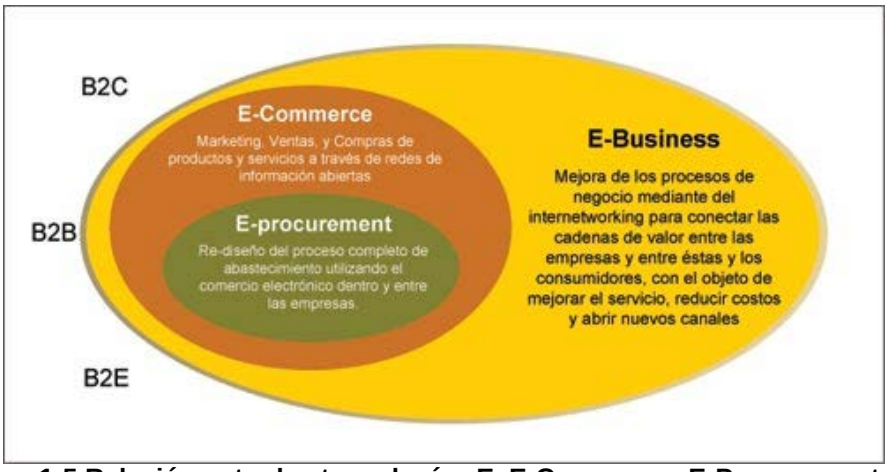


Figura 1-5 Relación entre las tecnologías E. E-Commerce, E-Procurement e E-Business.
Fuente: Artículo Gestión de compras electrónicas: e-Procurement, PC-News, Septiembre 2005.

Previo al desarrollo de una sistema de E-Procurement se debe tener en consideración “para que la solución sea efectiva, ésta debe trabajar bajo una infraestructura basada en datos (bases de datos), definir niveles de seguridad, diseño de la interfaz y experiencia de los usuarios, adicionalmente se debe evaluar los proveedores potenciales de tecnologías que mejor satisfagan las necesidades del negocio” [7], este enunciado claramente está indicando los requerimientos que los módulos del sistema deben cumplir para lograr una culminación exitosa, en cuanto a los proveedores de tecnología, considerando el contexto de este proyecto, la misma está restringida a la versión empresarial de Java (J2EE).

Además, “una vez definidas todas estas necesidades de negocio y el diseño del esquema de la compra, el sistema debe ser integrado de tal manera que garantice que el mapa tecnológico de lo que existe con lo que se desea esté

alineado estratégica y operativamente” [7]. Sin lugar a dudas, una de las mayores preocupaciones a tomar en cuenta, es la integración de este sistema con el de vendedores y compradores, la decisión de cómo hacerlo deberá centrarse en reducir lo más que se pueda la dificultad de ingreso de las empresas al mercado electrónico de E-Guana.

Otras ventajas

En este punto se tiene claro lo que pretende un sistema de E-Procurement, así mismo se han descrito algunas de las ventajas que trae consigo la implementación de un sistema de este tipo, a continuación se hablará de otras ventajas adicionales.

Como consecuencia de la automatización de los procesos, entre ellos el de pagos, “una plataforma de E-Procurement proporciona una transmisión rápida, permite el seguimiento de los pedidos y un proceso eficiente de las órdenes de compra y del pago de facturas” [7].

Además un punto importante es que “ofrece la posibilidad de integrar todos los procesos de la organización” [7], esto lógicamente debido a la participación activa y conjunta de todas las áreas de la empresa que permite la optimización y rentabilidad de los procesos relativos a la vida de la compañía.

Este modelo representa un gran cambio del esquema de trabajo tradicional con relación a procesos de compras y abastecimiento puesto que “a través de los procesos de E-Commerce, se genera un ambiente donde cualquier empresa o gobierno, sin importar su tamaño, puede proveer o comprar mejores productos a un mejor precio, acelerando el proceso de adquisición, el pago de bienes y servicios, por tanto ayudando a reducir costos, tomar decisiones rápidas y obtener valores agregados” [7], en definitiva se crea un mercado electrónico de negocios donde todos tienen las mismas oportunidades.

Además “las compañías pueden centrar sus recursos en la parte estratégica del aprovisionamiento (búsqueda, homologación y negociación de condiciones con proveedores), automatizando la parte administrativa del proceso de compras (formulación, aprobación y seguimiento de ordenes de compra y recepción de pedidos)” [7], por ende finalizando con un aprovisionamiento más eficaz.

Finalmente el E-Procurement es “uno de los pilares principales para la creación de E-Marketplaces, mercados comunes entre una serie de empresas y proveedores que se comunican a través de Internet y forman una verdadera comunidad” [7], esta es una consecuencia natural que resulta de la relación más directa entre las empresas que utilizan un sistema de E-Procurement.

1.3. Objetivos

Se pueden citar algunos objetivos generales sobre el uso del E-Procurement en el sistema E-Guana:

- Facilitar una herramienta en-línea que permita a los compradores y vendedores de productos comunicarse y efectuar transacciones seguras.
- Mejorar los tiempos de espera en los procesos de gestión de compras manteniendo una comunicación efectiva entre compradores y vendedores durante el seguimiento de sus pedidos (contacto).
- Monitorear la gestión de compra que se ha realizado.
- Fortalecer la relación cliente-proveedor brindando varios canales de comunicación (correo electrónico, mensajería) entre los usuarios del sistema dando un valor agregado para poder realizar eficientemente las negociaciones entre empresas participantes.
- Proporcionar una herramienta Web que permita generar reportes gerenciales de las transacciones de ventas y/o compras como soporte a la toma de decisiones, basándose en información integrada y global del negocio.
- Proveer el servicio de licitaciones y subastas.

Los objetivos antes descritos muestran sin lugar a dudas la importante influencia de la tecnología e-procurement en E-Guana, la cual se refleja en el alcance que ofrecen los distintos módulos, entre ellos los descritos en el presente documento.

Objetivos del Módulo de Administración:

A continuación, se pueden citar los objetivos del Módulo de Administración de E-Guana:

- Proveer control sobre el uso de la herramienta a los diferentes usuarios involucrados en el proceso de compra-venta, por medio de roles asignados.
- Proveer acceso a un gran catálogo de productos, facilitando la búsqueda de acuerdo a las preferencias de las empresas.
- Proveer de métodos de búsqueda de productos disponibles de acuerdo a sus características y conocer su stock disponible al momento de efectuar la compra.

Lo antes descrito será desarrollado ampliamente en el Capítulo 3, donde se muestran los requerimientos y las herramientas de implementación de este módulo.

Objetivos del Módulo de Pagos Transaccionales:

En tanto que los objetivos del Módulo de Pagos de Transacciones son los siguientes:

- Reducir los costos de suministros en el proceso de pago por adquisición de los productos.
- Facilitar al usuario un proceso de compra ágil y seguro.
- Minimizar el riesgo de fraude tanto para el cliente como el proveedor por medio del proceso de verificación de cuentas y aprobación de débitos.
- Proveer un mecanismo automático de generación de transacciones de débito y envío de información a los bancos.

El Capítulo 4 describe lo antes mencionado, mostrando ampliamente la definición y las herramientas de implementación de este módulo.

1.4. Ventajas de Uso de E-Procurement

A continuación se listan algunas de las ventajas más relevantes del uso de E-Procurement según José Ochoa en su artículo “E-Procurement: Una herramienta para lograr la integración y colaboración en la relación proveedor-cliente” [3]:

- Comunicación cliente-proveedor más ágil y eficiente.
- Incremento de ganancias sobre la competencia.

- Reducción de costos de operación en la transacción de compra-venta, la cual es consecuencia de:
    - Disminución en los niveles de inventario.
    - Requisición de materia prima en base a necesidades reales de cliente.
    - Eliminación de excesos.
    - Cumplimiento de los planes de producción.
- Reducción de posibles errores humanos en el proceso.
- Mayor capacidad de respuesta.
- Desarrollo sobre requerimientos específicos.
- Pueden establecerse Interfaces con otros sistemas ERP o propietarios.
- Permite el monitoreo del proveedor en tiempo real.
- Proporciona información clave del desempeño del proveedor.
- Administración del inventario (stock actual, establecer niveles mínimos y máximos permitidos).
- Administración del transporte de material (logística) al mostrar el estado en tiempo real del material en tránsito.
- Detectar con anticipación potenciales problemas en el suministro de material.

Una vez que se ha visto las ventajas de un sistema de E-Procurement se puede decir sin lugar a dudas que son de considerable importancia, lo que hace la implantación de un sistema de este tipo muy atractiva.

Tan sólo el “Incremento de ganancias sobre la competencia” [3] y la “Reducción de costos de operación en la transacción de compra-venta” [3] constituirían razón suficiente para lanzarse a la aventura de implementar alguna forma de E-Procurement. Pero las ventajas van más allá de mayores ingresos y menores egresos directos, al permitir un control detallado de los procesos y elementos involucrados en la cadena de aprovisionamiento, donde se tienen mejores servicios para los compradores y mayor prestigio para los vendedores con las obvias consecuencias.

CAPÍTULO 2

2. Análisis y diseño del sistema E-Guana

En este capítulo se tratará el análisis y diseño de E-Guana en su totalidad, de esta forma se tendrá una idea clara del funcionamiento completo y complejidad del sistema, así mismo le dará el contexto adecuado a los Módulos de Administración y Pagos de Transacciones que serán analizados y diseñados en los capítulos 3 y 4 respectivamente.

Se debe recordar que estos 2 últimos son los módulos desarrollados por los autores de este documento.

2.1. Requerimientos del Sistema

Los requerimientos de E-Guana más importantes son los siguientes:

- Permitir la creación, mantenimiento y consulta del catálogo de productos
- Permitir comprar en línea productos
- Permitir licitaciones y subastas de productos
- Permitir el pago en línea de las transacciones

- Generar reportes gerenciales

Una vez declarados los requerimientos se procederá a su análisis.

2.2. Análisis de Requerimientos

En esta sección se analizarán los requerimientos previamente establecidos.

El catálogo de productos es la principal herramienta que tendrán las empresas para exponerse ante las demás en el mercado electrónico creado por el sistema, por lo tanto esta funcionalidad debe ser de fácil uso, de igual forma es necesario tener en consideración la existencia de soluciones internas para el manejo de inventario en cada una de las empresas por lo que es deseable la integración de dichos sistemas con E-Guana.

Este requerimiento también expone la necesidad de proveer un mecanismo de seguridad, puesto que no todos los usuarios podrán actualizar o mantener el catálogo, de igual forma cierta información se considera confidencial por lo que la vía de comunicación también debe ser segura.

La compra en línea debe manejarse de forma similar a otros sistemas que ofrecen dicha funcionalidad, esto es, selección de productos que pasan a un carrito de compras para finalmente generar la orden de compra. Así se apela

a la familiaridad del usuario con otros sistemas de E-Commerce y se reduce cualquier renuencia al cambio.

En cuanto a la licitación y subasta de productos, se vislumbra la consideración de plazos y asignación dinámica de precios, de igual forma el sistema debe permitir el envío de las respuestas respectivas.

Otro punto importante que debe cubrir el sistema es el pago de las transacciones de las compras y que implica la comunicación con sistemas de entidades bancarias externas a E-Guana.

Como parte del soporte para la toma de decisiones de la empresa que utiliza el sistema, E-Guana debe generar un conjunto de reportes gerenciales útiles en base a los datos recopilados por el uso del sistema.

2.3. Arquitectura

La arquitectura de E-Guana se representa en la siguiente figura:

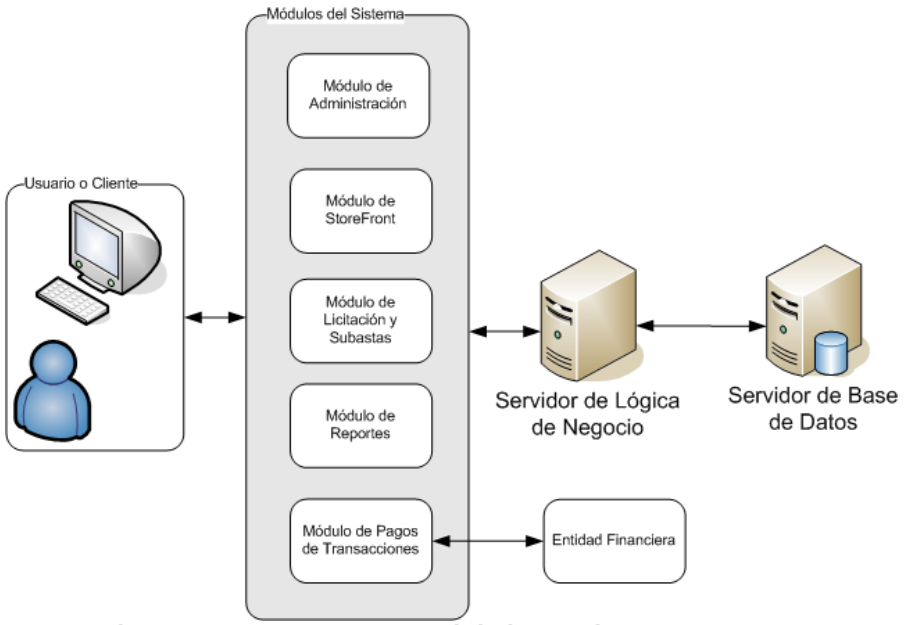


Figura 2-1 Esquema General de la arquitectura E-Guana
Fuente: E-Guana

Del esquema representado se debe aclarar que cada módulo estará conformado tanto por una aplicación Web como por objetos de lógica de negocio, más adelante en este capítulo se describirán las funciones que cumplirá cada módulo. Los clientes no accederán directamente a la base de datos sino a través de una interfaz Web o una aplicación de escritorio basada en el uso de interfaces remotas de objetos de negocio (Descrita en el Anexo 2).

Como se indicó en una sección anterior E-Guana será el punto de encuentro de compradores y vendedores, es decir, será un sistema de E-Procurement de varios compradores y varios vendedores. Por lo tanto deberá existir una empresa de servicios que adquiera E-Guana y que provea la infraestructura adecuada para exponer la funcionalidad del sistema. Mientras que las

empresas compradoras y vendedoras sólo requerirán el acceso a Internet y al menos un computador con un navegador Web.

Esta arquitectura, según el artículo “La Comunidad del E-Procurement” de CIENTEC considera a los componentes básicos de un sistema de E-Procurement los siguientes [7]:

- Una base de datos para almacenamiento de información: Es clave que los administradores tengan toda la información sobre sus proveedores (empresas) y que sea de fácil acceso.
- Catálogo de productos: Normalmente las empresas requieren una actualización periódica de su catálogo de productos. Debe existir la opción de que dicha base sea actualizada por los mismos dueños de empresas sin necesidad de acudir a la ayuda del administrador. Los catálogos deben ser desarrollados en un formato legible o entendible por el sistema para facilitar su carga. En muchas ocasiones además los datos pueden estar conectados a los servidores de los proveedores para tener actualizados datos como el inventario actual de un producto o el estado de algún pedido.
- Esquema jerárquico de autorización: La información debe ser manejada según las normas y necesidades que el administrador del sistema haya previamente establecido y sobre eso se instala un

sistema de acceso restringido a los usuarios autorizados de las empresas.

- Robusto esquema de seguridad: Una arquitectura de un sistema basado en E-Procurement debe poseer en principio políticas de seguridad de datos y accesos al interior de la organización.

En cuanto al componente “catálogo de productos”, esta arquitectura de E-Procurement permitirá el acceso y publicación del mismo a través del sitio provisto por E-Guana, por lo que la alimentación del catálogo deberá ser hecha por cada uno de los proveedores ya sea manualmente o a través de de comunicación basada en XML de manera automática.

2.4. Diseño

El diseño de E-Guana se sustenta en la tecnología J2EE (principal elemento de estudio del Tópico de Graduación “Desarrollo de Aplicaciones Transaccionales con Java y XML”). Por lo tanto, cada módulo diseñará los componentes necesarios partiendo de los elementos e infraestructura provista por este *framework*.

Resultado de esto los módulos expondrán su funcionalidad a través de objetos de negocio, que en el contexto J2EE son los *Enterprise Java Beans* ó EJBs por sus siglas en inglés. Sin embargo, como ya se ha establecido, el principal cliente del sistema es un navegador de Internet, por lo que para

permitir el acceso a los servicios de E-Guana se diseñará una aplicación Web por cada módulo, cada una de estas aplicaciones hará uso de los EJBs adecuados para atender los requerimientos y presentará los resultados a través de páginas JSP ó Java Server Pages, que también forman parte de la especificación J2EE.

El acceso a la base de datos se hará a través de EJBs de entidad en tanto que el manejo de la lógica de negocio corresponderá a los EJBs de sesión (para más detalle sobre estos elementos léanse las secciones posteriores).

La comunicación entre los módulos se hará también a través de los EJBs, por lo que cada módulo deberá hacer pública toda funcionalidad que pueda ser necesaria por otro módulo, pero teniendo en cuenta restricciones de seguridad, afortunadamente J2EE y EJB proveen mecanismos de fácil uso para esto por lo que tan sólo es cuestión de utilizarlos.

En el mismo ámbito de la seguridad, pero a nivel Web, ésta se logrará a través del protocolo HTTPS[2], para lo cual tan sólo es necesario una adecuada configuración de los archivos descriptores de cada aplicación Web.

---

[2] Hypertext Transfer Protocol SSL. Protocolo de transferencia de hipertexto SSL.

2.5. Metodología Utilizada

La metodología de desarrollo utilizada en el proyecto E-Guana es la conocida como Programación Extrema (Extreme Programming), debido a los beneficios de su uso pero sobretodo porque fue parte de estudio del Tópico de Graduación y se deseaba poner en práctica lo aprendido.

En el resto de esta sección se explicará lo que es la Programación Extrema.

La Programación Extrema ó XP (como también se la referirá en adelante) es "una metodología ligera que se enfoca en la codificación como la tarea principal" [19], es decir que en cada etapa del desarrollo del software habrán actividades basadas en el código que está siendo escrito.

Los cuatro valores de XP

Los cuatro valores de la Programación Extrema son [19]:

- Comunicación
- Simplicidad
- Retroalimentación
- Coraje

El objetivo de la **comunicación** es "proveer un lugar donde la gente pueda discutir el proyecto libremente sin miedo a represalias" [19].

En cuanto a **simplicidad** el autor indica que “se debe escoger el diseño, la tecnología, los algoritmos y las técnicas más simple que satisfagan al cliente” [19], en otras palabras no se deben complicar las cosas.

“La **retroalimentación** un elemento crucial se obtiene a través de pruebas de código, historias del cliente, pequeñas iteraciones y programación en pareja” [19], en este caso se hizo uso de las pruebas de código y la programación en pareja para asegurarse que se estaban cumpliendo los objetivos del proyecto.

El **coraje** se refiere a “ser lo suficientemente valiente para hacer lo que es correcto” [19]. El autor destaca este valor en situaciones en las que el código no esté bien hecho y existe el temor de que se si le hace algún cambio se afecta al sistema, por lo que sugiere apoyarse en las pruebas, ya que como él indica “las pruebas son la clave para el coraje” [19], debido a que con ellas se garantiza que el código haga lo que se espera una vez que se hace alguna modificación.

Una vez que se ha hablado de los cuatro valores de XP, se procederá a describir los cinco principios que se fundamentan en estos valores.

Los cinco principios de XP

Los cinco principios para el desarrollo de XP son [19]:

- Proveer retroalimentación rápida
- Asumir simplicidad
- Hacer cambios incrementales
- Acoger el cambio
- Hacer trabajo de calidad

Proveer **retroalimentación rápida** se refiere a que “cuanto más rápido uno tenga retroalimentación, lo más rápido se la podrá responder y guiará mejor el proceso de diseño, codificación y entrega del sistema” [19], se puede notar que para que se de esta adecuada retroalimentación se deberán hacer pequeñas entregas al cliente para su posterior evaluación.

“**Asumir simplicidad** significa tratar cada problema como un problema simple hasta que se demuestre lo contrario” [19]. Como se puede deducir de lo anterior, esta simplicidad se puede poner en práctica gracias a las entregas pequeñas, ya que con ello sólo se diseña para dicha iteración.

Muy relacionado al principio anterior está **hacer cambios incrementales**, como indica el autor “no sobre diseñar el sistema, siempre es mejor hacer un poco a la vez” [19].

Acoger el cambio es una consecuencia de lo descrito anteriormente. “El cambio es esperado, debido a que se está entregando valor de negocio incrementalmente al cliente” [19].

El último punto se explica por si sólo puesto que es sentido común **hacer un trabajo de calidad**, pero con las técnicas de XP se hace una tarea más fácil.

Prácticas de XP usadas en E-Guana

El desarrollo de E-Guana hizo uso de varias prácticas establecidas por la Programación Extrema.

Una de las prácticas utilizadas fue la **prueba de código** que se explicará detalladamente en el capítulo 5.

También se utilizó la **integración continua**, facilitada por la herramienta Ant, en esencia esta integración comprendía el despliegue de todos los objetos de negocio y aplicaciones Web de los distintos grupos de desarrollo, con lo que se garantizaba en cualquier momento el adecuado y completo funcionamiento del sistema, esta tarea también soportada a través de un sistema de control de versiones que podía ser accedido remotamente por los desarrolladores.

Se siguió un **estándar de codificación**, más específicamente el estándar de Sun para la codificación Java.

Finalmente la **programación en pareja** se usó en menor medida, sin embargo se evidenció su utilidad en el mejoramiento de la comunicación

entre los miembros del equipo, el intercambio de conocimientos y al final resultando en un código de mayor calidad.

2.5.1. Tecnología J2EE

El Tutorial J2EE establece que “la plataforma J2EE Java 2 Edición Empresarial provee una aproximación basada en componentes para el diseño, el desarrollo, el ensamblaje y el despliegue de aplicaciones empresariales” [13].

A continuación se representa la arquitectura de una aplicación J2EE:

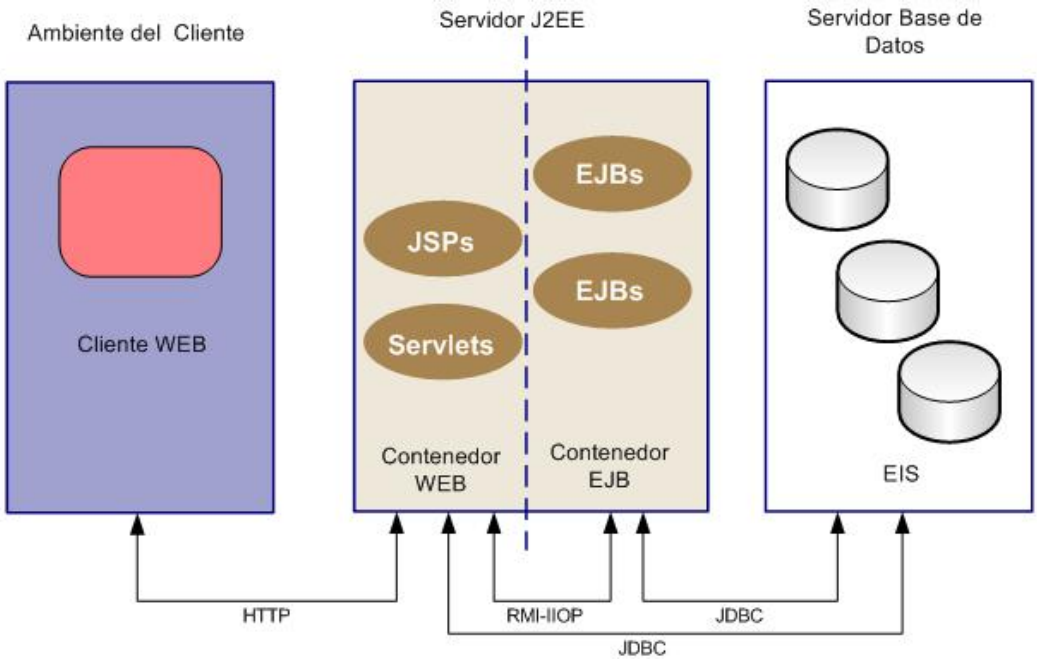


Figura 2-2 Arquitectura en Capas de una Aplicación J2EE.
Fuente: The J2EE Tutorial [13]

Un concepto importante que se debe tener en consideración en sistemas que utilizan J2EE es la idea de componente, definido como “la unidad de software funcional auto-contenida que se ensambla dentro de una aplicación J2EE y que se comunica con otros componentes de aplicación” [13], además se definen los siguientes componentes para una aplicación J2EE [13]:

- Enterprise Java Beans EJB
- Servlets
- Java Server Pages JSP

Adicionalmente “estos componentes requieren de un contenedor para poder exponer sus servicios” [13].

Por lo tanto E-Guana al ser una aplicación J2EE estará formada por varios componentes J2EE, esto implica codificar los varios tipos de componentes dependiendo de la capa de la aplicación en cuestión.

Como se ve en la gráfica los EJBs se ubican en el contendor EJB que en este caso es JBoss versión 3.2.4 mientras que las páginas JSP y los Servlets se desplegarán en el contenedor Web, Tomcat 5 para E-Guana.

2.5.1.1. Enterprise Java Beans (EJB)

Según el Tutorial de J2EE, “Enterprise JavaBeans (EJB) es una especificación creada por Sun Microsystems que define un *framework* para

componentes Java por el lado del servidor. EJB hace que la tecnología de objetos distribuidos sea más accesible y más fácil de usar, ofreciendo un nivel de abstracción mayor y por lo tanto más eficiente para el desarrollo de software" [13], cada especificación EJB tiene una versión, la utilizada en E-Guana es la 1.2 mientras que la que actualmente se encuentra en el mercado es la versión 1.3 pero que se conoce como la versión 3 simplemente.

El Tutorial J2EE también indica que "los Enterprise Java Beans son vagamente similares a los Java Beans. Son componentes reusables los cuales son utilizados en aplicación de gran escala. Sin embargo, aquí es donde la similitud termina. EJBs difieren en varias maneras, la más importante es que son EJBs distribuidos" [13], la similitud de los EJBs con simples Java Beans se da en los métodos de acceso de los atributos de un EJB ya que son del tipo "get" (para obtener el valor) y "set" (para establecer el valor) en ambos casos.

Un punto importante de los EJBs, como se indicó anteriormente, es que "son desarrollados dentro de un contenedor el cual maneja sus interacciones y provee un soporte en tiempo de ejecución" [13].

Además "un solo EJB es un grupo de objetos" [13], como se verá más adelante esto implica que la codificación de un *bean* comprende la escritura

de varias clases. En la versión 1.2 de J2EE esta característica se constituye en el mayor impedimento para iniciar de forma exitosa el desarrollo de una solución con este tipo de tecnología ya que cada EJB requiere 3 ó 4 clases distintas, sin embargo luego surgieron herramientas que automatizan la creación de dichas clases.

Finalmente “ellos no son accedidos directamente por clientes, estos son accedidos por referencia” [13], es decir que la aplicación cliente usará una referencia del *bean,* más específicamente una interfaz del mismo.

El uso de EJBs presenta algunas ventajas y desventajas:

Desventajas:

A continuación se listan algunas de las limitantes que se presentan en el uso de EJBs según el Tutorial J2EE [13]:

- EJBs no pueden crear o manejar hilos.
- EJBs no pueden acceder a los hilos usando el paquete *java.io*.
- EJBs no pueden operar directamente con *sockets*.
- EJBs no pueden cargar librerías nativas.
- EJBs no pueden usar AWT[3] para interactuar con los usuarios.

[3] Abstract Windows Toolkit. Toolkit de Java.

- EJBs únicamente pueden pasar objetos y valores que sean compatibles con RMI/IIOP[4].
- EJBs deben proveer constructores públicos.
- Los métodos no pueden ser estáticos.

A pesar de las desventajas la aplicación puede vivir y tener grandes beneficios.

Ventajas:

Los EJBs simplifican el desarrollo de aplicaciones empresariales por las siguientes razones [13]:

- Subclase de otra clase.
- Las interfaces pueden ser extendidas a otras interfaces que sean descendientes de *EJBObject* o *EJBHome*.
- Los métodos de ayuda pueden pasar cualquier tipo de parámetro o retornar tipos dentro del EJB.
- Los métodos de ayuda pueden ser de cualquier tipo de visibilidad.

La arquitectura EJB provee todas estas clases de beneficios enfocados a desarrolladores y clientes de este tipo de aplicaciones.

---

[4] Protocolo estándar de los servidores J2EE.

Es recomendable usar EJB si la aplicación cumple con cualquiera de los siguientes requisitos: [9]

- Si la aplicación debe ser escalable y distribuida.
- Si las transacciones requieren asegurar integridad de datos.
- Si la aplicación tiene variedad de clientes.

Por lo tanto si existe un número creciente de usuarios, los EJBs permitirán distribuir los componentes de la aplicación entre varias máquinas con su localización transparente para dichos usuarios, así mismo facilitan el uso de transacciones distribuidas y debido a la variedad de clientes, se pueden usar los EJBs para almacenar el modelo del negocio, y acceder a la misma información.

2.5.1.2. Servlets

Otro de los componentes de una aplicación J2EE son los Servlets que “son módulos de código escrito en Java que añaden funcionalidad a un servidor Web. Fueron diseñados para aceptar peticiones de un cliente y generar los mensajes de respuesta correspondiente” [20]. Como se vio en la figura 2.3 el contenedor de Servlet es responsable por el mantenimiento del ciclo de vida del Servlet, en el caso de E-Guana el contenedor de Servlets utilizado en E-Guana es Tomcat 5.

A continuación se presenta la figura que representa el esquema de funcionamiento de un servlet:

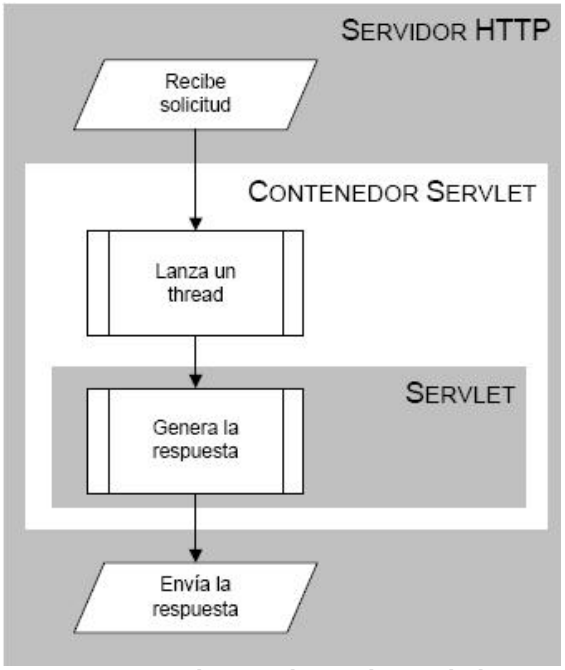


Figura 2-3 Esquema de Funcionamiento de los Servlets.
Fuente: Servlets y JSP
por Alfonso Cubero Moral y Sergio Luna García, p. 9. [20]

Un aspecto importante a destacar del ciclo de vida de los servlets es que “una vez inicializados, son guardados en memoria” [20], así que cada vez que llegue una petición ésta va hacia el servlet en memoria el cual genera una respuesta rápidamente.

A continuación se detallan las tareas más comunes que realiza un servlet [20]:

1. Leer cualquier dato enviado por el usuario: los datos normalmente se introducen por medio de la página Web, pero también pueden obtenerse a partir de un *applet* Java.
2. Obtener otra información sobre la petición que se encuentra embebida en la propia petición HTTP: esta información se refiere por ejemplo a los *cookies*, el nombre del *host* de donde proviene la petición, etc.
3. Generar los resultados: esta parte puede requerir acceder a una base de datos, invocar a una aplicación o simplemente computar los datos de entrada.
4. Generar un documento con los resultados: se debe establecer el tipo de documento que va a ser devuelto (una página HTML, una imagen, un archivo comprimido, etc.).
5. Establecer los parámetros apropiados para la respuesta.
6. Enviar la respuesta al cliente: una vez que se tiene el formato del documento que se entregara como respuesta y se establecieron los parámetros de la comunicación, se envía la respuesta al cliente.

Es necesario hacer una aclaración respecto a las tareas de un servlet, el listado anterior corresponde a servlets implementados de tal forma que atienden el requerimiento y generan la respuesta, sin embargo en otros casos, se implementan como elementos de control que atienden el

requerimiento y redirigen la respuesta a una página JSP que hace la respectiva presentación.

2.5.1.3. Java Server Pages (JSP)

JSP es “la tecnología para generar páginas Web de forma dinámica en el servidor, desarrollado por Sun Microsystems, basado en *scripts* que utilizan una variante del lenguaje Java” [21]. Antes de JSP la alternativa que se tenía para generar contenido dinámico en un entorno Java era a través de servlets.

La tecnología JSP, o de Java Server Pages, es “una tecnología Java que permite a los programadores generar dinámicamente HTML, XML o algún otro tipo de página Web” [9], es decir, aunque su nombre indique lo contrario, con una página JSP la respuesta puede ser inclusive una imagen, todo depende de las necesidades del desarrollador. “Esta tecnología permite al código Java y a algunas acciones predefinidas ser embebidas en el contenido estático” [9], en las páginas JSP se escribe el texto que va a ser devuelto en la salida (normalmente código HTML) poniendo código Java dentro de ella para poder modificar o generar contenido dinámicamente. “El código Java se incluye dentro de las marcas de etiqueta <% y %>” [9], lógicamente crear páginas Web con contenido dinámico de esta forma es

mucho más cómodo que hacerlo a través de pura codificación Java con los servlets que era la alternativa previa.

“La principal ventaja de JSP frente a otros lenguajes es que facilita la integración con clases Java lo que permite separar en niveles las aplicaciones Web” [20], se puede deducir de este enunciado que la separación facilita la división de roles en el equipo de desarrollo, así las páginas podrán ser creadas por un diseñador gráfico y las clases Java ser implementadas por un programador. Inclusive gracias a esta separación es que patrones de diseño como el llamado Modelo Vista Controlador pueden ser implementados.

Además “Java se caracteriza por ser un lenguaje que puede ejecutarse en cualquier sistema” [21], lo que sumado a JSP le da mucha versatilidad.

La siguiente figura representa el esquema de ejecución de una página JSP:

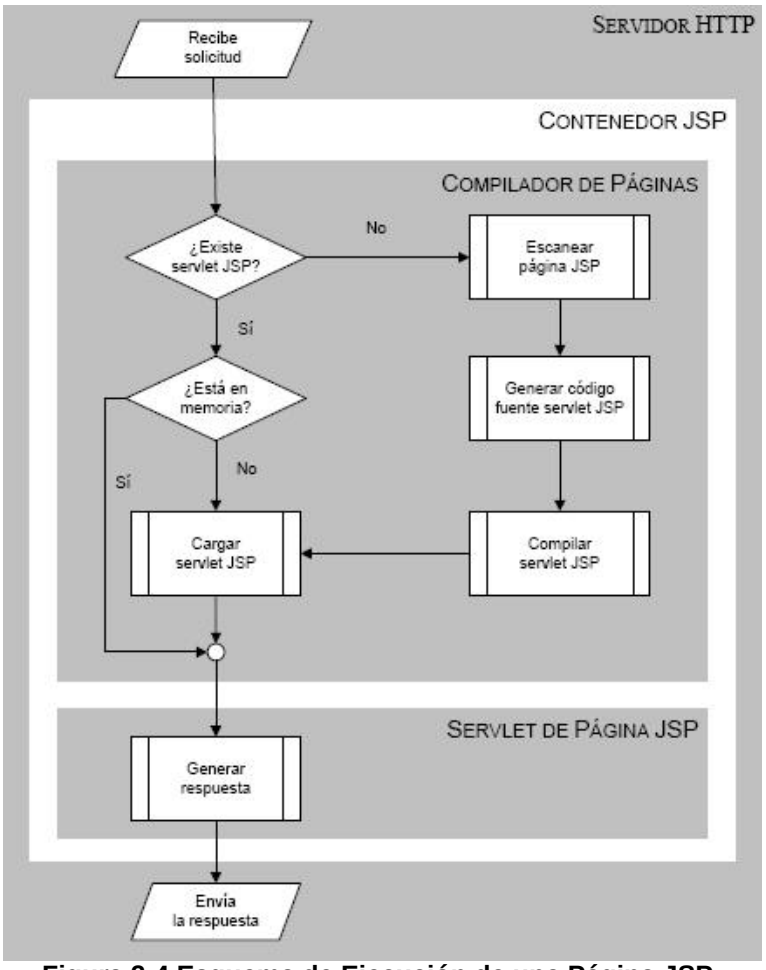


Figura 2-4 Esquema de Ejecución de una Página JSP.
Fuente: Servlets y JSP por Alfonso Cubero Moral y Sergio Luna García, p. 9. [20]

Como se muestra en la figura, si no existe el servlet respectivo de la página, se ejecuta el compilador de páginas en 3 etapas, lógicamente éstas etapas requieren un tiempo de procesamiento por lo que la primera vez que se requiere la página el tiempo de respuesta puede ser considerablemente alto.

Ventajas de JSP

JSP tiene algunas ventajas sobre las otras alternativas disponibles

JSP vs. HTML: el HTML normal no puede contener información dinámica, así que las páginas HTML no pueden estar basadas en la entrada del usuario o en fuentes de datos del lado del servidor.

JSP vs. ASP: Siendo ASP la tecnología competidora de Microsoft, se pueden resaltar 2 grandes ventajas de JSP sobre ASP [20]:

- La parte dinámica está escrita en Java, no en VBScript o cualquier lenguaje específico de ASP, así que es más poderoso para desarrollar aplicaciones que requieren componentes reutilizables.
- JSP es portable a cualquier sistema operativo y servidor Web, no se está encerrado en Windows NT/2000 e IIS. Se puede utilizar el mismo argumento cuando se compara JSP con ColdFusion: con JSP se puede usar Java y no estás atado a un servidor en particular.

JSP vs. PHP: la ventaja de JSP es que “la parte dinámica está escrita en Java, el cual es probable que ya se conozca, tiene una extensa API para el trabajo en red, acceso a bases de datos, objetos distribuidos, en cambio, PHP requiere el aprendizaje de un nuevo lenguaje entero” [20].

Otro punto a destacar de JSP es la existencia de herramientas de soporte de código abierto y gratis, tales como el IDE Eclipse. De igual forma existen

contenedores Web con este mismo tipo de distribución, como el servidor Tomcat utilizado en E-Guana.

2.5.1.4. Contenedores

Otro concepto importante dentro de la tecnología J2EE son los contendedores. “Los contenedores J2EE proporcionan acceso a los servicios del entorno del servidor J2EE. Los contenedores son la interfaz entre un componente y la funcionalidad de bajo nivel que soporta el componente” [9], por lo que a diferencia de una aplicación Java simple, un componente J2EE no puede por si solo exponer los servicios que implementa. “Antes de poder ejecutar un componente Web, un *bean* empresarial o un componente de una aplicación cliente, debe ensamblarse dentro de una aplicación J2EE y desplegarse dentro de su contenedor” [9], el ensamblaje de un componente, en palabras simples, es empaquetar todas las clases y recursos requeridos en un archivo comprimido con extensión JAR o WAR según corresponda.

“El servidor J2EE proporciona contenedores para Enterprise JavaBeans (EJB) y para componentes Web. El contenedor EJB maneja la ejecución de los *beans* empresariales de las aplicaciones J2EE, mientras que el contenedor Web maneja la ejecución de las páginas JSP y los componentes servlets de la aplicación J2EE” [9], como se ha indicado previamente, E-

Guana utiliza el servidor J2EE JBoss versión 3.2.4 que a su vez provee el contenedor Web Tomcat versión 5.0.

“Otros contenedores distintos a estos son el contenedor de aplicaciones clientes y el contenedor de *applets*, que no son parte del servidor J2EE porque residen en la máquina del cliente” [9], el contenedor de aplicaciones cliente es simplemente la máquina virtual de Java instalada en el cliente, en tanto que el contenedor de *applets* está normalmente embebido en los navegadores Web.

2.6. Módulos del Sistema

En esta sección se describirá de forma general cada uno de los módulos que constituyen E-Guana, siendo estos:

- Módulo de Administración
- Módulo de Store Front
- Módulo de Licitación & Subastas
- Módulos Reportes
- Módulo de Pagos de Transacciones

Los módulos de Administración y Pagos de Transacciones se desarrollarán en los capítulos 3 y 4 respectivamente.

2.6.1. Administración

Provee las herramientas para el mantenimiento de las principales entidades que soportan al sistema. Entre dichas entidades están las empresas, usuarios y productos.

Así mismo provee el mecanismo de seguridad que protege la funcionalidad y datos del sistema. La seguridad está basada en la asignación de roles a los usuarios.

2.6.2. Store Front

Store Front es el módulo de E-Guana encargado de manejar la interacción directa entre compradores y vendedores. Este módulo se encarga de implementar la funcionalidad de “tienda”, de ahí su nombre (Store = tienda, front = frente). Entre las principales funciones están:

- Ingresar a la tienda (ingreso al sistema, login).
- Buscar en las perchas el o los producto que se desea comprar (navegación en el catálogo).
- Separar los productos que se desea comprar en una canasta o carrito de compras (Carro de compras virtual).
- Llevar los productos en el carrito a la caja. (Orden de compra virtual).
- La cajera entrega el o los productos cancelados (Orden de entrega virtual).

El sistema mantiene en todo momento notificado a las empresas del estado de las órdenes, ya sea del lado del comprador o del vendedor, en el módulo de Store Front, a través de notificaciones propias del sistema como por vía e-mail. El proceso de compras llega a término cuando se ha realizado con éxito el proceso de pago y se da autorización al despacho.

2.6.3. Licitación y Subastas

Garantiza la realización de todos los servicios necesarios para la gestión de licitación y subasta. A continuación se detallan las principales funcionalidades de este módulo:

Licitación:

Las empresas pueden utilizar Internet para solicitar y comparar precios de diversos proveedores. Las ofertas se pueden publicar en los tablones de anuncios para fomentar la participación de una gama más amplia de posibles proveedores (publicación de una licitación). Con este módulo, se podrá realizar un aprovisionamiento global. Si necesita materiales específicos rápidamente, los postores entran en el sistema de E-Guana para leer los detalles de la llamada a licitación y para introducir sus ofertas, luego el sistema realiza la elección de la licitación ganadora y se emite la orden de compra.

En base a esta explicación del módulo, el proceso se puede resumir en los siguientes pasos:

- Recolección de requerimientos por unidades.
- Consolidación de los requerimientos.
- Publicación de la licitación.
- Oferta de los licitantes.
- Elección de la licitación ganadora.
- Generación de una orden de compra.

Al finalizar el proceso de licitación y de la emisión de la orden de compra, se da por terminado el ciclo de compra-venta del bien, no sin antes haber efectuado el pago.

Subastas:

La empresa vendedora tiene la posibilidad de ofrecer sus productos o artículos y permitir que varios compradores hagan sus propuestas. El proceso de subasta de E-Guana se puede resumir en tres pasos:

- Publicación de la subasta.
- Recepción y análisis de ofertas.
- Elección de la subasta ganadora.

La elección de la subasta ganadora será en base a la mejor propuesta ofrecida por el comprador, es decir, la de mayor precio.

2.6.4. Pago de Transacciones

Este módulo es el encargado del manejo de los débitos bancarios a las diferentes empresas, verificando los saldos en línea y asegurando así que la venta sea exitosa. Por cada orden de compra generada se efectúa el pago electrónico, para lo cual se generan archivos con transacciones de débito/crédito que luego serán enviados a las entidades bancarias respectivas.

2.6.5. Reportes

Este módulo permitirá generar los reportes de E-Guana y puede generar tres tipos de reportes:

Reportes de nivel operacional: sirven para monitorear las actividades elementales y las transacciones de una organización.

Reportes de nivel administrativo: apoyan las actividades administrativas, de monitoreo, de control y de toma de decisiones realizadas por los mandos medios.

Reportes de nivel estratégico: promueven el cambio de los objetivos, procesos, productos, servicios y relaciones con el entorno para ayudar a la organización a obtener ventajas competitivas.

2.7. E-Guana y Otros Sitios Similares

Existe una gran variedad de sistemas E-Procurement en el mercado; herramientas desarrolladas con las mismas características ya descritas en párrafos anteriores; por lo que a continuación se lista en breve algunas particulares que las hacen diferentes a E-Guana.

Agresso Business World

Agresso es una solución E-Procurement cuyos objetivos estratégicos son las organizaciones centradas en servicios y personas. El código no está disponible.

Para acceder al sistema se lo hace por medio del sitio: http://www.unit4agresso.com.

Ketera

Ketera ofrece a las compañías soluciones que le ayuden a disminuir sus costos adquisitivos, mayor actuación de los proveedores y crecimiento del

negocio sin gastos de aplicaciones tradicionales. Elaborado con código propietario no disponible.

Para acceder al sistema se lo hace por medio del sitio: http://www.ketera.com.

Purchasing + Plus

El sistema Purchasing + Plus es un producto con un alto precio cuyas licencias de actualización también tienen costo. Elaborado con código propietario no disponible.

Para acceder al sistema se lo hace por medio del sitio: http://www.palmasdev.com.

Tallion

Sistema elaborado con código propietario no disponible y establecido para un único tipo de negocio.

Para acceder al sistema se lo hace por medio del sitio: http://www.tallion.com.

En la siguiente ilustración se resumen las diferencias entre E-Guana y las otras soluciones de E-Procurement mencionadas:

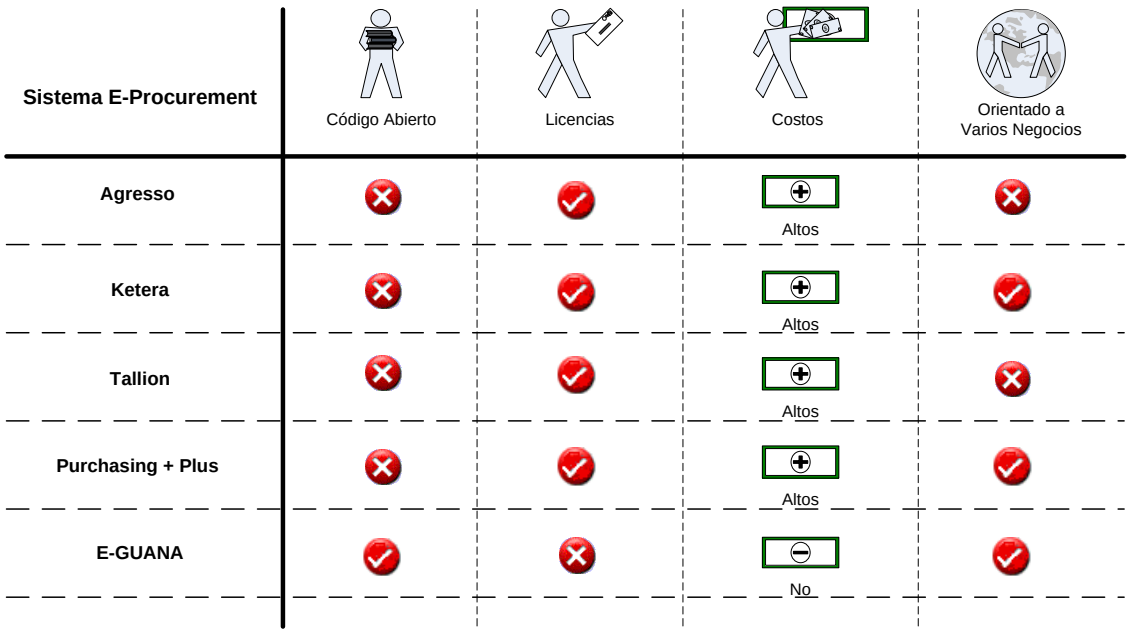


Figura 2-5 Comparación de características de sistemas E-Procurement. Fuente: Sistema E-Guana, Sistema Tallion http://www.tallion.com, Sistema Ketera http://www.ketera.com, Sistema Purchasing+Plus http://www.palmasdev.com y Sistema Agresso http://www.unit4agresso.com

Como se puede observar en las características antes descritas, E-Guana ofrece mayor ventaja competitiva sobre los otros sistemas:

- E-Guana está desarrollado completamente con código abierto.
- E-Guana no necesita de licencias para su implantación.
- E-Guana no es un producto con precios altos.
- E-Guana no está limitada a un único tipo de negocio, ya que es de propósito general.

Esta última ventaja es fácilmente identificada y explorada a través de los distintos módulos que componen E-Guana.

CAPÍTULO 3

3. Definición e Implementación del Módulo de Administración

A lo largo del capítulo se dará una explicación de las principales funciones, objetivos e implementación del módulo de Administración del sistema E-Guana.

3.1. Requerimientos

Los requerimientos del módulo de Administración de E-Guana son los siguientes:

- Proveer de una herramienta completa para el mantenimiento de los usuarios del sistema. Esto incluye creación, eliminación, bloqueo y búsqueda de usuarios.
- Proveer la funcionalidad para mantenimiento y consulta de todas las entidades involucradas en el catálogo de productos (categorías de productos, productos, y catálogos).
- Proveer un mecanismo de seguridad para el acceso a la información que mantiene el sistema; basado en roles con privilegios asociados.

- Proveer un manejo fácil y estándar dentro del sistema, para todas las variables y parámetros del sistema. Esto incluye mensajes, categorías de productos, etc.
- Proveer la interfaz entre los módulos del sistema y la base de datos.
- Control de suscripción de empresas y cupo de usuarios.
- Soportar la internacionalización del sistema al proveer de un mecanismo que permita el mantenimiento y la presentación de las interfaces de usuario en distintos idiomas.

Una vez identificados los requerimientos se procederá a su análisis.

3.2. Análisis de Requerimientos

Los requerimientos de este módulo reflejan que el mismo provee los componentes “Catálogo de Productos”, “Esquema jerárquico de autorización” y “Robusto esquema de seguridad” de la arquitectura de un sistema de E-Procurement (mencionados en la arquitectura del sistema).

En base a las principales funciones que se deben realizar en el sistema, las acciones y las características que encierran cada uno de los individuos, se identificaron 4 roles que deben ser organizados jerárquicamente (la

arquitectura de un sistema de E-Procurement requiere esta organización jerárquica).

- Rol de “Administrador del Sistema”,
- Rol de “Administrador de Empresa”,
- Rol de “Vendedor” (Proveedor) y
- Rol de “Comprador”.

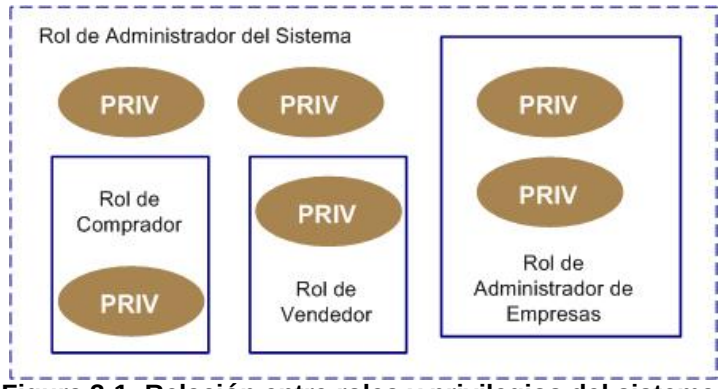


Figura 3-1: Relación entre roles y privilegios del sistema.
Fuente: Diseño del Sistema E-Guana.

El rol de “Administrador del Sistema” tendrá a su cargo toda la administración del sistema, así como también el manejo y control de todas las acciones que se realicen dentro del sistema. Tendrá la capacidad de permitir o negar cualquier solicitud de ingreso o aceptación para uso del sistema.

El usuario con rol de “Administrador de la Empresa” podrá editar los datos descriptivos de la empresa así como crear las unidades de negocio que considere pertinentes (una unidad de negocio es la representación dentro del

sistema de un local de dicha empresa). También deberá controlar que se respeten las normas de cantidad de usuarios asignados por empresa.

Los usuarios con roles de “Vendedor” o “Comprador” serán los encargados de realizar todas las transacciones involucradas en el proceso de negociación y cierre del proceso de compra-venta de un producto.

Un usuario estará asociado a una unidad de negocio de una empresa, esta asociación, adicionalmente, permitirá la asignación de uno o más roles, estos pueden ser: “Administrador de Empresa”, “Vendedor” y “Comprador”, lo anterior no aplica para el “Administrador del Sistema”. Dependiendo de los roles que tenga asociados un usuario con su unidad de negocio, serán las acciones o funciones que podrá desarrollar dentro de dicho ámbito.

A continuación se detallarán los casos de uso, lo que permitirá una clara identificación de lo que va a hacer este módulo.

3.2.1. Casos de Uso

Primero se presentará el diagrama de casos de uso del módulo de Administración.

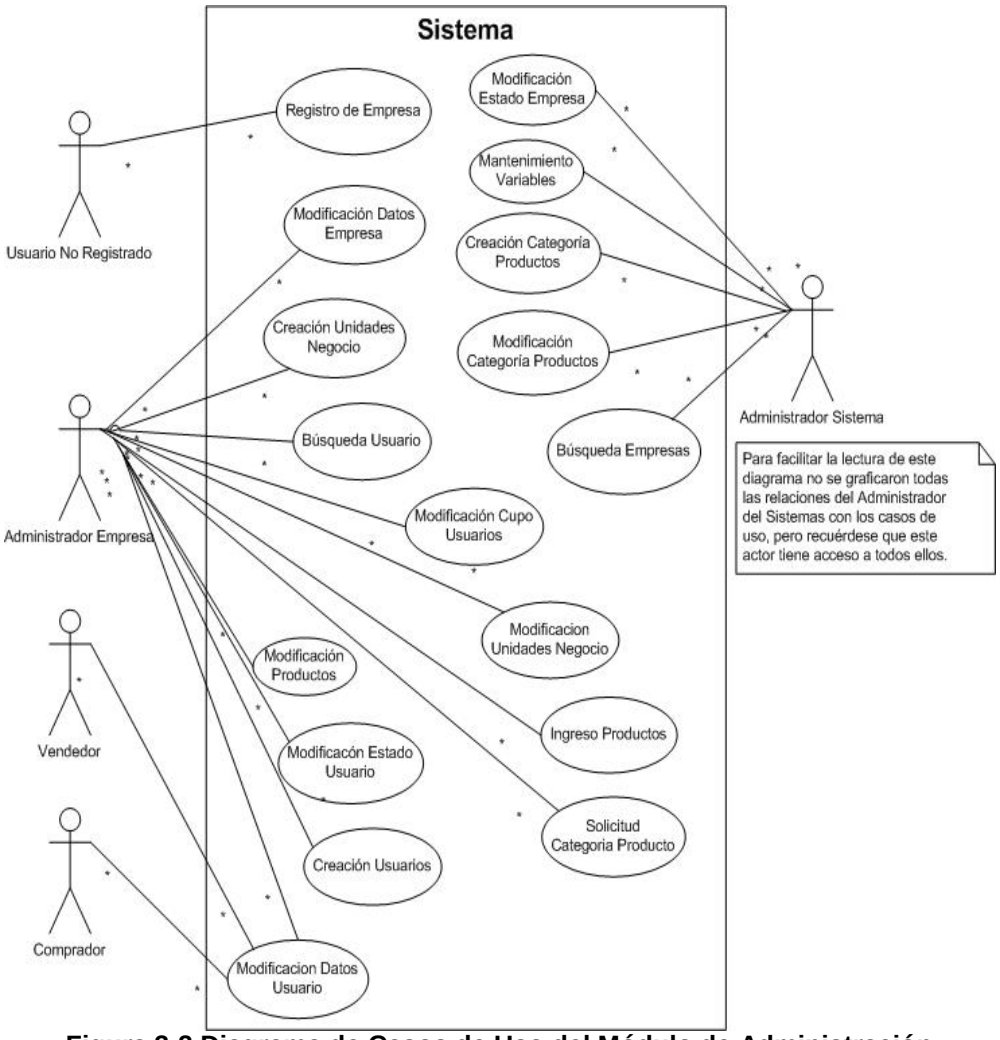


Figura 3-2 Diagrama de Casos de Uso del Módulo de Administración
Fuente: E-Guana

Una vez representados los casos de uso, a continuación se presenta su respectiva descripción:

Caso de uso	**Registro de Empresa**
Objetivo	Registrar a una empresa en el sistema
Descripción	El representante de la empresa que desea utilizar el sistema completa el formulario de solicitud de registro. Luego el “Administrador del Sistema” aprueba o rechaza dicha solicitud. Si se aprueba el registro se crea la empresa, la unidad de negocio conocida como matriz y el usuario con el rol de “Administrador de

	la Empresa".
Valor medible	Los valores que se medirán en este proceso son: una nueva empresa dentro del sistema en estado "Activo", la unidad de negocio matriz de la empresa y la cuenta de usuario con rol de "Administrador de la Empresa" también en estado "Activo".
Involucrados	Las personas que van a intervenir a lo largo del proceso de registro de una empresa son el solicitante de registro y el "Administrador del Sistema".
Controles	Debe existir por lo menos un usuario con rol de "Administrador del Sistema".
Entradas	La entrada del proceso serán todos los datos requeridos por la solicitud de registro.
Salidas	Las salidas que pueden existir en este proceso son: • Correo electrónico dirigido al solicitante con los datos para el ingreso al sistema. • Mensaje de éxito en el registro de la empresa.
Consultas y Reportes	Listas de solicitudes ordenadas por fecha.

Caso de uso	**Modificación de Datos de la Empresa**
Objetivo	Modificar los datos que identifican a la empresa, es decir, nombre, RUC, correo electrónico, etc.
Descripción	El "Administrador de la Empresa" modifica los datos de la empresa a la que pertenece. El "Administrador del Sistema" modifica los datos de cualquier empresa registrada en el sistema. Al término del proceso, el sistema refleja los datos modificados de la empresa.
Valor medible	El valor medible estará en base a los datos modificados.
Involucrados	Los actores involucrados en este proceso son: "Administrador de la Empresa" y "Administrador del Sistema".
Controles	Debe existir la empresa y estar en estado "Activo". Es necesario que el usuario tenga los permisos suficientes para cambiar los datos de la empresa. Los datos ingresados deben de ser datos válidos.
Entradas	La entrada a este proceso son los datos de la empresa que se desean modificar.
Salidas	La salida al proceso será un mensaje de éxito en la modificación de los datos.

Caso de uso	**Modificación de Estado de una Empresa**
Objetivo	Modificar el estado de una empresa
Descripción	El "Administrador del Sistema" cambia el estado de una empresa. Los estados por los que puede pasar una empresa son: • Pendiente: Es el estado inicial de la empresa, luego de generada una solicitud de registro en el sistema, en este estado no hay un solo usuario asociado a la empresa.

	• Activa: Es el estado que permite a la empresa hacer uso completo del sistema. • Bloqueada: La empresa no puede hacer uso del sistema. • Eliminada: La empresa ha sido eliminada del sistema, es el estado final de la empresa. El cambio de estado de la empresa se refleja en el estado de cada una de sus unidades de negocio y de los usuarios que las integran.
Valor medible	Se medirá en base al estado actual de la empresa (Pendiente, Activa, Bloqueada o Eliminada).
Involucrados	El involucrado en este proceso es el “Administrador del Sistema”.
Controles	Debe existir por lo menos un usuario con rol de “Administrador del Sistema”. El estado previo a la modificación no puede ser “Eliminada”.
Entradas	La entrada al proceso es el nuevo estado de la empresa.
Salidas	La salida del proceso será un mensaje de éxito en la modificación del estado de la empresa. Si el estado actual es “Bloqueada” se envía un correo electrónico al “Administrador de la Empresa” indicándole los motivos del bloqueo.

Caso de uso	**Búsqueda de Empresas**
Objetivo	Buscar los datos de las empresas dentro del sistema de acuerdo a algún criterio.
Descripción	El “Administrador del Sistema” realiza la búsqueda de empresas, ésta puede ser de acuerdo al nombre, RUC o por el estado de la empresa. Esta búsqueda resulta en el listado de los datos de las empresas encontradas. Este caso de uso es utilizado por: • Modificación del Estado de una Empresa • Modificación de Datos de la Empresa
Valor medible	Se medirá en base al resultado de la búsqueda de empresas.
Involucrados	El “Administrador del Sistema” es el actor involucrado.
Controles	No se realiza la búsqueda entre empresas con estado “Pendiente”.
Entradas	La entrada al proceso son los datos que permitan hacer la búsqueda, como nombre, RUC, etc.
Salidas	La salida del proceso son los datos de las empresas encontradas de acuerdo al criterio de búsqueda.

Caso de uso	**Creación de Unidades de Negocio**
Objetivo	Crear unidades de negocio en una empresa.
Descripción	El “Administrador de la Empresa” crea una o varias unidades de negocio dentro de su empresa. Una unidad de negocio es una entidad asociada a una empresa, su existencia depende de la empresa. Las unidades de negocio pueden hacer requerimientos de adquisición de bienes, que luego llegan a la empresa, la cual consolida todos los requerimientos de sus unidades de negocio y

	decide si los requerimientos serán atendidos a través de una compra o una licitación.
Valor medible	Se medirá en base a una nueva Unidad de Negocio.
Involucrados	El involucrado es el “Administrador de la Empresa”.
Controles	Todos los datos de la unidad de negocio son obligatorios.
Entradas	La entrada al proceso es toda la información referente a la unidad de negocio como: dirección, teléfono, etc.
Salidas	La salida el proceso será un mensaje de éxito en la creación de la unidad de negocio.

Caso de uso	**Modificación de Unidades de Negocio**
Objetivo	Modificar los datos de una unidad de negocio.
Descripción	El “Administrador de la Empresa” modifica los datos de alguna de las unidades de negocio de su empresa.
Valor medible	El valor medible son los datos modificados de la unidad de negocio.
Involucrados	El actor involucrado es el “Administrador de la Empresa”.
Controles	Deben existir tanto la unidad de negocio como su correspondiente empresa y tener estado “Activo”.
Entradas	Las entradas al proceso son los datos de la unidad de negocio que se desean modificar.
Salidas	La salida del proceso será un mensaje de éxito en la modificación de los datos de la unidad de negocio.

Caso de uso	**Creación de Usuarios dentro de una Empresa**
Objetivo	Crear usuarios en una de las empresas registradas en el sistema.
Descripción	El “Administrador de la Empresa” crea uno o varios o usuarios dentro de alguna de las unidades de negocio de su empresa. Inicialmente el único usuario que tiene una empresa, es aquel cuyo rol es de “Administrador de la Empresa”, y será éste el encargado de crear el resto de usuarios que la empresa necesite. El “Administrador de la Empresa” deberá ingresar datos que identifiquen al usuario de forma única, como: clave, nombre, dirección, teléfono, etc., además de asignarle los roles.
Valor medible	El valor que se medirá es el usuario creado.
Involucrados	El actor involucrado en este proceso es el “Administrador de la Empresa”.
Controles	El nombre de usuario del usuario a crear no existe en el sistema. No se ha excedido el límite de usuarios de la empresa.
Entradas	La entrada al proceso será los datos del nuevo usuario.
Salidas	La salida de este proceso es un mensaje indicando la creación exitosa del usuario. Si el número de usuarios es igual al límite que puede crear, el sistema envía un mensaje al “Administrador del Sistema”.

Caso de uso	**Modificación de Datos de un Usuario**
Objetivo	Modificar los datos de un usuario registrado.

Descripción	El usuario que lo requiera modifica sus datos o los de otro usuario. Los datos de un usuario son: nombre, apellido, empresa, correo electrónico, descripción, usuario, clave, rol. Dependiendo del rol del usuario, éste podrá modificar datos específicos; por ejemplo: el “Administrador de Sistema” podrá cambiar todos los datos, en tanto que el “Administrador de Empresa” no podrá modificar los estados, mientras que otro usuario podrá cambiar cualquier dato excepto su usuario y roles asignados.
Valor medible	El valor medible son los cambios realizados en los datos del usuario.
Involucrados	El actor involucrado es el Usuario del Sistema.
Controles	El usuario debe haber ingresado al sistema. Dependiendo del rol del actor se permiten modificar los datos.
Entradas	La entrada del proceso son los nuevos datos del usuario.
Salidas	La salida del proceso será un mensaje indicando los cambios realizados en los datos del usuario.

Caso de uso	**Búsqueda de Usuario**
Objetivo	Ver los datos de los usuarios que coincidan con un criterio de búsqueda.
Descripción	El usuario hace la búsqueda de acuerdo a usuario, nombre ó número de cédula. El resultado es el listado de los datos de los usuarios que coinciden con los criterios ingresados.
Valor medible	El valor medible es el resultado de la búsqueda de usuarios.
Involucrados	Los actores involucrados en este proceso son: el “Administrador de la Empresa” y el “Administrador del Sistema”.
Controles	El “Administrador de la Empresa” sólo puede buscar usuarios dentro de su misma empresa y el “Administrador del Sistema” puede buscar en todo el sistema.
Entradas	Las entradas pueden ser datos específicos del usuario a buscar como por ejemplo: nombre y apellido, nombre de usuario ó número de cédula.
Salidas	La salida son los datos completos encontrados de cada usuario que coincidan con los criterios de búsqueda.
Consultas y Reportes	El reporte será una lista de los usuarios encontrados. La consulta es la búsqueda.

Caso de uso	**Modificación de Estado de un Usuario**
Objetivo	Modificar el estado de un usuario del sistema.
Descripción	El usuario modifica el estado de un usuario. Los estados por los que puede pasar un usuario son: • Activo: Es el estado inicial del usuario, luego de aprobada la solicitud de registro. Es el estado que permite al usuario hacer completo uso del sistema. • Bloqueado: Es un estado en el que el usuario no puede

	hacer uso de las funciones del sistema. • Eliminado: El usuario no podrá usar ninguna de las opciones del sistema, es el estado final del usuario en el sistema. El “Administrador de la Empresa” será la persona autorizada en modificar el estado de un usuario de su empresa. El estado de un usuario también se ve alterado cuando cambia el estado de una empresa, en este caso todos los usuarios que conforman una empresa cambian su estado.
Valor medible	El valor medible en este caso es el estado actual del usuario.
Involucrados	El involucrado en este caso es el “Administrador de la Empresa”.
Controles	Debe existir por lo menos un usuario con rol de “Administrador de la Empresa”. El estado previo a la modificación no puede ser “Eliminado”.
Entradas	La entrada al proceso es el nuevo estado del usuario.
Salidas	La salida será un mensaje de éxito en la modificación del estado del usuario. Si el estado actual es “Bloqueado” se envía un correo electrónico al usuario correspondiente indicándole los motivos del bloqueo.

Caso de uso	**Modificación del Cupo de Usuarios que Administra una Empresa (Solicitud)**
Objetivo	Solicitar el aumento del cupo de usuarios que se pueden crear dentro de una empresa.
Descripción	El “Administrador de la Empresa” solicita el aumento del cupo de usuarios que puede crear dentro de su empresa a través del formulario de la solicitud.
Valor medible	El valor que se medirá es la solicitud de la modificación del cupo de usuarios.
Involucrados	El actor involucrado es el “Administrador de la Empresa”.
Controles	No puede existir una solicitud pendiente de este tipo.
Entradas	Las entradas al proceso serán: • Los datos de la solicitud: • Motivo de la modificación. • Número de usuarios a incrementar.
Salidas	La salida del proceso es la solicitud.

Caso de uso	**Modificación del Cupo de Usuarios que Administra una Empresa (Aprobación)**
Objetivo	Aprobar o negar una solicitud de modificación del cupo de usuarios de una empresa.
Descripción	El “Administrador del Sistema” aprueba o rechaza una solicitud de aumento del cupo de usuarios por empresa. La solicitud cambia de estado dependiendo de la decisión tomada.
Valor medible	El valor medible será el nuevo cupo de usuarios que tiene una empresa.
Involucrados	El actor involucrado en este caso es el “Administrador del

	Sistema".
Controles	Existe una solicitud pendiente.
Entradas	La entrada al proceso es la solicitud y la decisión del actor.
Salidas	La salida del proceso es el cupo de usuarios modificado.
Consultas y Reportes	El "Administrador de Sistema" tiene un listado de todas las solicitudes de este tipo recibidas y ordenadas por fecha.

Caso de uso	**Solicitud de Nuevas Categorías de Productos**
Objetivo	Solicitar una nueva categoría de productos
Descripción	El "Administrador de la Empresa" completa el formulario de solicitud de una nueva categoría y aplica "Enviar". El sistema genera una nueva solicitud con estado "Pendiente".
Valor medible	El valor medible es la solicitud hecha por el "Administrador de la Empresa"
Involucrados	El "Administrador de la Empresa" es el actor involucrado.
Controles	Se debe seleccionar la categoría padre en la que se ubicará la categoría solicitada.
Entradas	La entrada al proceso son los datos de la solicitud.
Salidas	La salida es un mensaje de la aprobación de la categoría o tipo de producto solicitado por el "Administrador de Empresa".
Consultas y Reportes	El sistema elaborará un listado de todas las solicitudes enviadas por la empresa del solicitante.

Caso de uso	**Creación de Categorías de Productos**
Objetivo	Crear una nueva categoría de productos directamente o a partir de una solicitud de nueva categoría de producto.
Descripción	El "Administrador del Sistema" ingresa una nueva categoría de productos. Los atributos de la categoría de productos son: identificador de categoría, nombre, descripción, identificador de subcategoría. El "Administrador del Sistema" también puede crear una categoría en base a una solicitud, tan solo con la aprobación de la misma.
Valor medible	El valor medible será la categoría de productos creada.
Involucrados	El actor involucrado en este caso es el "Administrador del Sistema".
Controles	No se permitirá el ingreso de una categoría ya existente. Es decir no pueden existir dos categorías con el mismo nombre y la misma categoría padre.
Entradas	La entrada al proceso son los datos de la nueva categoría o la solicitud de una categoría.
Salidas	La salida será la categoría o tipo de producto creado exitosamente.

Caso de uso	**Modificación de Categorías de Productos**
Objetivo	Hacer modificaciones a los datos de una categoría de producto.
Descripción	El "Administrador del Sistema" actualiza los datos referentes a una categoría de producto.
Valor medible	El valor medible corresponde a los datos actualizados de las

	categorías de productos.
Involucrados	El involucrado en este caso es el “Administrador del Sistema”.
Controles	No se puede modificar la categoría padre de la categoría.
Entradas	La entrada son los datos a actualizar de la categoría.
Salidas	La salida será la categoría o tipo de producto actualizado exitosamente.

Caso de uso	**Ingreso de Productos**
Objetivo	Crear un nuevo producto dentro del sistema.
Descripción	El “Administrador de la Empresa” ingresa un nuevo producto en el catálogo de su empresa. Los atributos de los productos son los siguientes: identificador, descripción, precio, marca, categoría (alimentos, electrodoméstico), código de empresa, observación (descuento, tiempo de vida, garantía), etc. El “Administrador de la Empresa” determina si el producto ya existe en el sistema para en ese caso tan solo agregar los datos específicos a su catálogo (ítems en el catálogo, precio).
Valor medible	El valor medible será el producto ingresado al sistema.
Involucrados	El actor involucrado es el “Administrador de la Empresa”.
Controles	El producto a ingresar no existe en el catálogo de la empresa.
Entradas	La entrada serán los datos necesarios para que un producto forme parte de los productos disponibles.
Salidas	La salida serán los datos de los productos disponibles en el sistema.
Consultas y Reportes	Se consultan los productos que ya existen en el sistema.

Caso de uso	**Modificación de Productos**
Objetivo	Modificar los datos de los productos.
Descripción	El “Administrador de la Empresa” modifica los datos de los productos que forman parte del catálogo de su empresa.
Valor medible	El valor medible serán los datos modificados del producto.
Involucrados	El actor involucrado en este caso es el “Administrador de la Empresa”.
Controles	Sólo se podrá modificar valor, cantidad en stock y comentario, es decir los datos propios del catálogo y no los datos relacionados a la naturaleza del producto, como el nombre de producto.
Entradas	Las entradas son los nuevos datos correspondientes al producto a modificar.
Salidas	La salida serán los datos del producto modificados con éxito.
Consultas y Reportes	Se podrá consultar el catálogo de productos de la empresa.

Caso de uso	**Mantenimiento de Variables del Sistema**
Objetivo	Hacer cambios en los valores de las variables utilizadas en el sistema.

Descripción	El “Administrador del Sistema” da mantenimiento a las variables del sistema tales como: variables de entorno (ej. impuestos), de configuración (ej. idioma), etc., las cuales permitirán parametrizar detalles propios del sistema. Cada variable del sistema tiene los siguientes atributos: nombre, valor, tipo. Estos atributos son obligatorios.
Valor medible	El valor medible lo constituyen los datos de la variable.
Involucrados	El involucrado en este caso es el “Administrador del Sistema”.
Controles	El “Administrador de Sistema” es el único que puede modificar los valores de estas variables, pues un cambio inadecuado podría incurrir en el mal funcionamiento del sistema. Los valores de estas variables no pueden ser nulos.
Entradas	Las entradas serán el valor y la variable a cambiar.
Salidas	La salida será la un mensaje de que la variable de sistema ha sido modificada con éxito.
Consultas y Reportes	El sistema elaborará un listado de variables y sus respectivos valores, emitido únicamente para el “Administrador de Sistema”.

Para facilitar el desarrollo de las funciones identificadas en los casos de uso, éstas se agruparán dependiendo del objeto que se vaya a administrar, es decir:

- Administración de Empresas
- Administración de Usuarios
- Administración de Productos
- Administración de Categorías de Productos
- Administración de Variables del Sistema

Esta agrupación se puede considerar como sub-módulos dentro de este módulo. Además esto permite asignar tareas hasta cierto punto independientes y que pueden ser trabajadas en paralelo para ganar tiempo.

3.2.2. Administración de Usuarios

La Administración de Usuarios agrupa todas las funciones relacionadas a la creación y mantenimiento de usuarios y las tareas relacionadas a estos.

Para hacer uso del sistema es necesario tener un usuario registrado en el mismo. La Administración de Usuarios permite aparte de la creación del usuario, la búsqueda de usuarios, modificación de los datos del usuario y cambio del estado de un usuario.

El usuario con rol de “Administrador de la Empresa” es el primer usuario asociado a una empresa, este se creará en el proceso del registro de la empresa en el sistema, es este usuario quien podrá hacer todas las operaciones pertinentes dentro de las unidades de su empresa inclusive la creación de otros usuarios.

Como medida de control para el buen uso del sistema, cada empresa tendrá un cupo de usuarios, que se deberá poder modificar a través de una solicitud.

Para restringir el acceso de un usuario al sistema será posible establecerle el estado de “Bloqueado”, este cambio de estado lo podrán hacer tanto el “Administrador del Sistema” como el “Administrador de la Empresa”.

3.2.3. Administración de Empresas

La Administración de Empresas abarcará el registro y el mantenimiento de empresas, la creación y el mantenimiento de unidades de negocio.

Toda empresa que desee utilizar E-Guana deberá registrarse en el sistema, para esto se proveerá de un formulario en línea de solicitud de registro. Este formulario deberá ser llenado por alguna persona autorizada de la empresa (normalmente el representante legal). Será decisión del “Administrador del Sistema” la aprobación o rechazo de la solicitud.

Si la solicitud fue aprobada, el usuario que registró a la empresa podrá hacer uso del sistema inmediatamente.

El usuario que registró a la empresa poseerá por defecto el rol de “Administrador de la Empresa”, así mismo por defecto, se creará una Unidad de Negocio para representar a la matriz de la empresa.

La Unidad de Negocio es la forma que el sistema representa a los distintos locales que forman parte de la empresa. Además como se indicó anteriormente es una unidad de negocio a la que se asociarán los usuarios que conforman la empresa.

3.2.4. Administración de Productos y Categoría de Productos

La Administración de Productos y Categoría de Productos se encargará del mantenimiento de los productos disponibles para la venta, de la misma forma del mantenimiento de las diferentes categorías de productos que deberán ser creadas en base a las necesidades de agrupamiento de productos.

Entre las principales funciones que proveerá este sub-módulo se tiene:

- Coordinar con el encargado de la empresa proveedora de productos, la actualización periódica del inventario y precios.
- Ingresar, consultar, cambiar el estado y actualizar los productos de las empresas.
- Aprobar, ingresar y modificar las categorías de productos.

Las empresas interesadas en vender sus productos utilizando la plataforma provista por E-Guana, deberán publicar la información de sus productos. Cada producto podrá ser ingresado al sistema a través de formularios en línea o a través de funcionalidad XML que le permitirá agregar datos por lotes (datos del catálogo).

La empresa deberá ingresar tanto datos generales como datos específicos de los productos que ofertarán. Los datos generales corresponden a los

datos que describen al producto en el mercado, mientras que los datos específicos se refieren a los datos pertinentes a su empresa, por ejemplo precio. De esta forma una empresa podrá tan sólo ingresar los datos específicos del producto si es que otra empresa ya ha ingresado ese producto al sistema. Lo anterior facilita la discriminación entre productos, funcionalidad básica para el análisis de ofertas y licitaciones.

Cada producto se agrupará en categorías, durante el proceso de ingreso del producto, se seleccionará la categoría a la que pertenece el producto. Si el conjunto de categorías que ofrece inicialmente el sistema no es suficiente, la empresa podrá solicitar la creación de una nueva categoría y será decisión del usuario con rol de “Administrador del Sistema” aprobar o rechazar dicha solicitud.

3.2.5. Administración de Variables del Sistema

La Administración de Variables del Sistema agrupa las funciones que permitirán la configuración del sistema

Estas funciones de administración estarán disponibles única y exclusivamente para el usuario con el rol de “Administrador del Sistema”.

Los parámetros de configuración del sistema estarán centralizados tanto para el establecimiento de los valores como para la lectura de los mismos.

Adicionalmente se podrá administrar y extender los lenguajes soportados por la plataforma, inicialmente se soportarán el inglés y el español.

3.3. Diseño

Una vez que se ha establecido con claridad lo va a hacer el módulo se debe indicar cómo se lo va a hacer a través de elementos de diseño.

El diseño se plasmará con las siguientes herramientas UML: diagrama de clases, diagramas de interacción de objetos y el modelo entidad-relación.

3.3.1. Diagrama de Clases

El módulo de Administración se compone de las siguientes clases:

- Usuario
- Empresa
- Categoría
- Unidad de Negocio
- Producto
- Variable
- Rol
- Permiso
- Solicitud de Categoría

A continuación se presenta el diagrama de clases de este módulo, indicando en cada clase los atributos y los métodos más importantes.

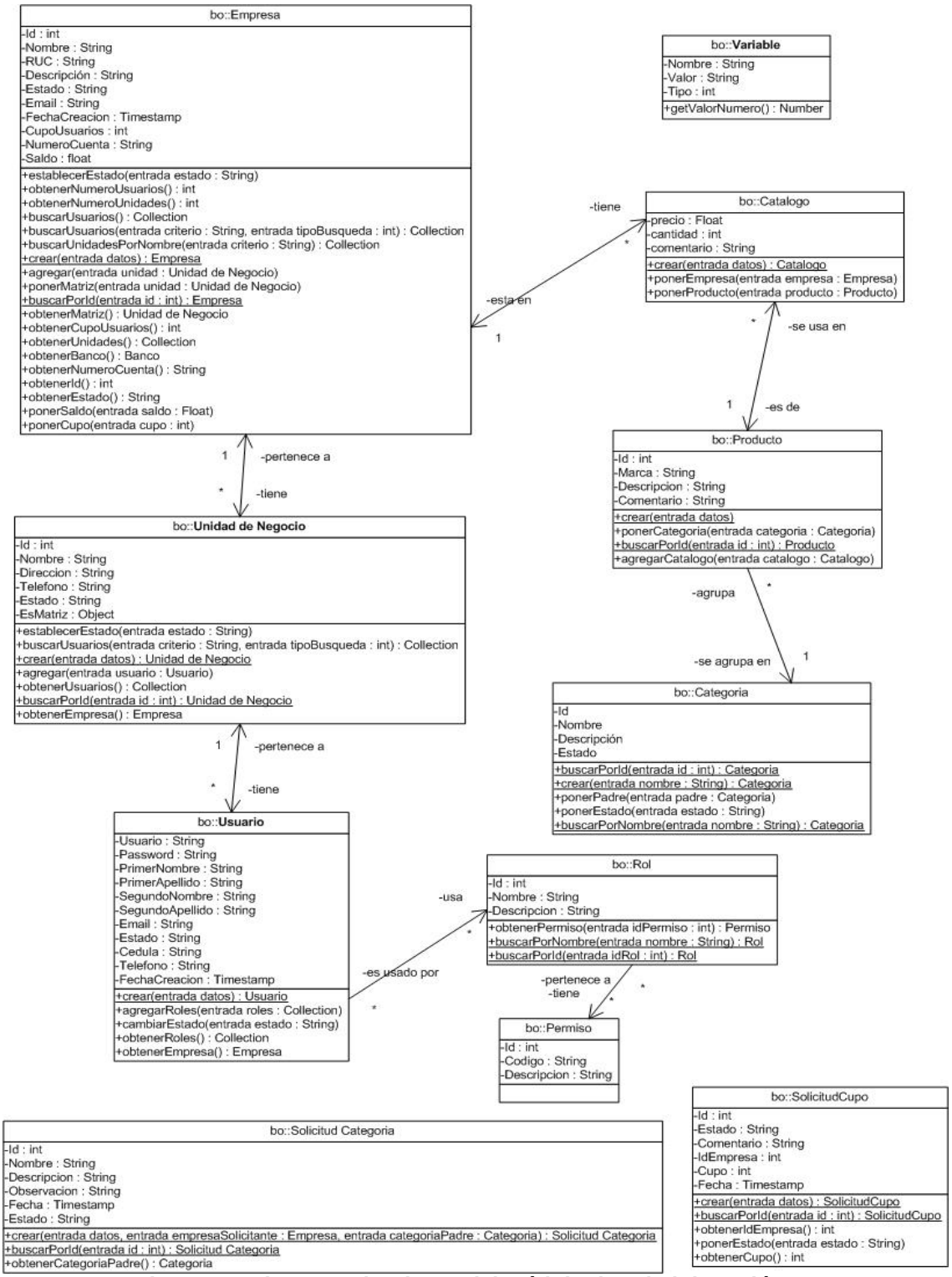


Figura 3-3 Diagrama de Clases del Módulo de Administración.
Fuente: Diagrama principal de clases del Sistema E-Guana.

A continuación se detallarán los atributos de cada una de las clases del módulo de Administración de E-Guana:

Usuario

La clase Usuario representa a una persona que utiliza el sistema.

Los atributos de la clase Usuario son:

- **Primer Nombre:** el primer nombre de la persona.
- **Primer Apellido:** el primer apellido de la persona.
- **Segundo Nombre:** el segundo nombre de la persona.
- **Segundo Apellido:** el segundo apellido de la persona.
- **Email:** el correo electrónico de la persona.
- **Estado:** corresponde al estado actual que tiene el usuario. Los posibles estados del usuario son:
  - Activo**:** el usuario puede usar el sistema normalmente.
  - Eliminado**:** el usuario ha sido eliminado lógicamente del sistema, este es un estado final.
  - Bloqueado**:** el usuario no puede acceder al sistema debido a que ha sido.
- **Cédula:** la cédula o el pasaporte de la persona.
- **Teléfono:** el teléfono donde se puede ubicar a la persona.

- **Usuario**: el nombre de usuario para acceder al sistema.
- **Clave:** clave de acceso al sistema.
- **Fecha de Creación:** es la fecha en la que se creó el usuario dentro del sistema, este atributo es establecido por el sistema automáticamente.

A excepción del usuario “Administrador del Sistema”, cada uno de los usuarios pertenece a una empresa, o propiamente dicho a una unidad de negocio de una empresa, esto se detallará más adelante.

Empresa

Esta clase representa la entidad que agrupa varios usuarios con la capacidad de ofrecer productos para la venta o de adquirir productos sobre la plataforma provista por E-Guana.

Los atributos de la clase Empresa son:

- **Nombre:** es el nombre con el que se identifica a la Empresa. Este valor es único.
- **RUC:** el registro único del contribuyente. Este valor también es único.
- **Descripción:** cualquier descripción adicional que sea necesaria.
- **ID:** el identificador dentro del sistema de la empresa

- **Estado:** corresponde al estado actual de la Empresa, los posibles estados de la empresa son:

  - Pendiente: la empresa aún no ha sido aprobada por el "Administrador del Sistema", por lo que ésta no puede hacer uso del sistema.
  - Pendiente con Cuenta Válida: la empresa tiene una cuenta válida. El módulo de Pagos de Transacciones ha verificado que la empresa si tiene una cuenta bancaria válida y que podría hacer uso del sistema.
  - Pendiente sin Cuenta Válida: la empresa no tiene una cuenta válida. El módulo de Pago de Transacciones recibió del banco asociado a la empresa un mensaje indicando que no tiene una cuenta bancaria válida.
  - Activa: la empresa puede hacer uso completo del sistema.
  - Eliminada: la empresa ha sido eliminada del sistema, éste es un estado final.
  - Bloqueada: la empresa, y por ende sus usuarios, no pueden acceder al sistema, una empresa es bloqueada por la administración o luego de la solicitud de una verificación de la validez de su cuenta bancaria.

- **Email:** el correo electrónico de la empresa.

- **Fecha de Creación:** la fecha en la que la empresa se registró en el sistema, este atributo es asignado automáticamente por el sistema.
- **Cupo de Usuarios:** el número máximo de usuarios que puede tener la empresa dentro del sistema.
- **Número de Cuenta:** el número de la cuenta bancaria de la empresa.
- **Saldo:** el saldo de su cuenta bancaria.
- **Banco:** el ID del banco en el que tiene la cuenta.

Toda empresa está conformada por una o más entidades conocidas como unidades de negocio, los usuarios de la empresa pertenecen de hecho a una unidad de negocio.

Unidad de Negocio

Una unidad de negocio representa uno de los locales que conforman una empresa. Debe existir como mínimo una unidad de negocio en cada empresa para poder hacer uso de sus facultades dentro del sistema, esta unidad es considerada la matriz de la empresa. Los usuarios son asignados a las unidades de negocio.

Los atributos de la clase Unidad de Negocio son:

- **Nombre:** el nombre de la unidad de negocio.
- **ID:** identificador único dentro del sistema de la unidad de negocio.

- **Ciudad:** la ciudad donde se encuentra la unidad de negocio.
- **Dirección:** la dirección de la unidad de negocio.
- **Teléfono:** el número de teléfono de la unidad de negocio.
- **Estado:** corresponde al estado actual de la unidad, los posibles estados de la unidad de negocio son:

    - Activa: los usuarios de la unidad pueden ejercer todos sus roles.
    - Eliminada: la unidad ha sido eliminada del sistema, este es un estado final.
    - Bloqueada: la unidad, y por ende sus usuarios, no puede acceder al sistema.

- **EsMatriz:** indica si la unidad corresponde a la matriz de la empresa.

En la fase de creación de una empresa se crea la unidad de negocio matriz con estado “Activa”.

Producto

Es la descripción general de un artículo ofrecido por una empresa. Permite crear los productos que son ofrecidos por las empresas vendedoras.

Los atributos de la clase Producto son:

- **ID:** identificador único del producto.

- **Marca:** la marca del producto.
- **Descripción:** la descripción del producto.
- **Comentario:** algún comentario que sea necesario.

El ID es un atributo generado automáticamente en la etapa de creación del producto.

Categoría

Es la categoría de un producto. La idea de la categoría es agrupar productos de características similares (ej. televisores, alimentos).

Los atributos de la clase Categoría son:

- **ID:** identificador único de la categoría.
- **Nombre:** el nombre de la categoría.
- **Descripción:** la descripción de la categoría.

Los dos últimos atributos son los únicos que pueden modificarse luego de la creación de una categoría de producto.

Variable de Sistema

Esta clase representa una variable de configuración del sistema.

Los atributos de la clase Variable de Sistema son:

- **Nombre:** identificador único de la variable.
- **Valor:** contenido de la variable.
- **Tipo:** el tipo de valor que puede tomar la variable, puede ser una cadena de texto o un número entero.

Entre las variables establecidas se tienen:

- Cupo de Usuarios por Defecto: corresponde al máximo número de usuarios que puede crear una empresa cuando se registra inicialmente en el sistema.
- Cupo de Usuarios Máximo: corresponde al tope de usuarios que puede solicitar una empresa.
- IVA: es el porcentaje de impuesto al valor agregado que se asigna a las compras.
- Comisión de Compra: el porcentaje que se le descontará al vendedor por cada transacción de compra.

Para acceder al valor de cada una de estas variables se las debe buscar por el nombre respectivo.

3.3.2. Diagramas de Interacción de Objetos

Las siguientes ilustraciones muestran los principales diagramas de interacción de objetos (DIOs):

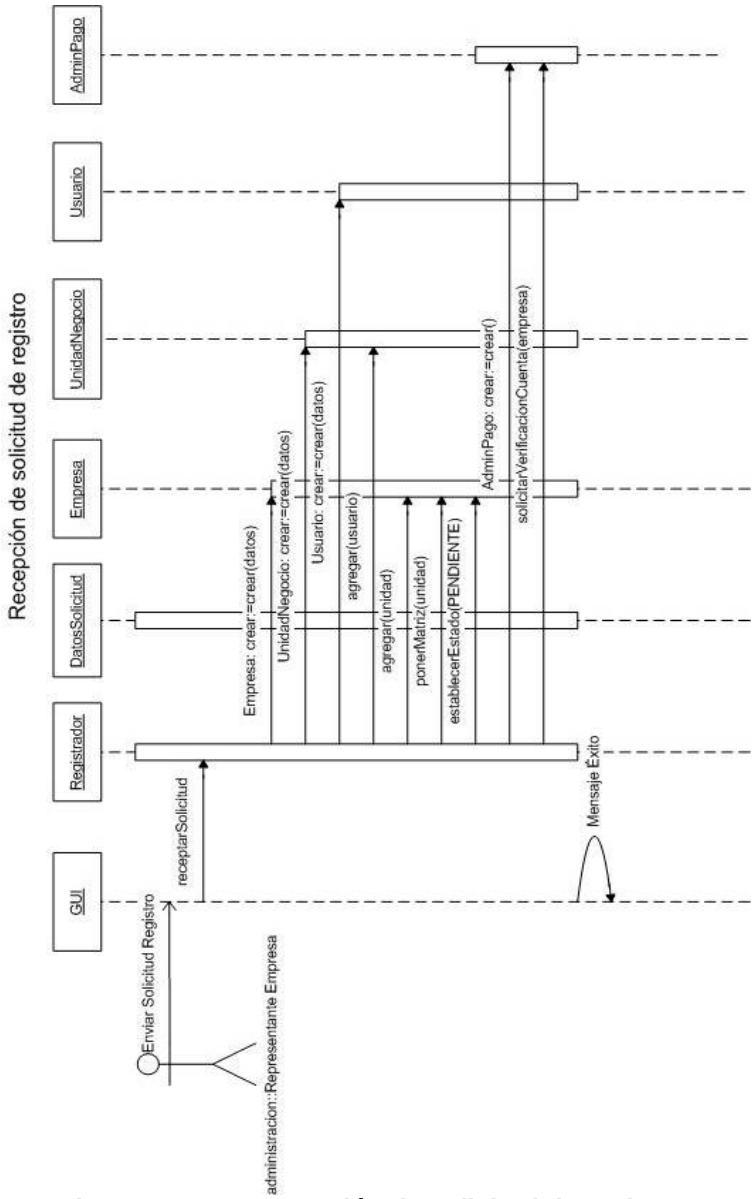


Figura 3-4 DIO Recepción de solicitud de registro.
Fuente: DIOs del Módulo de Administración del Sistema E-Guana.

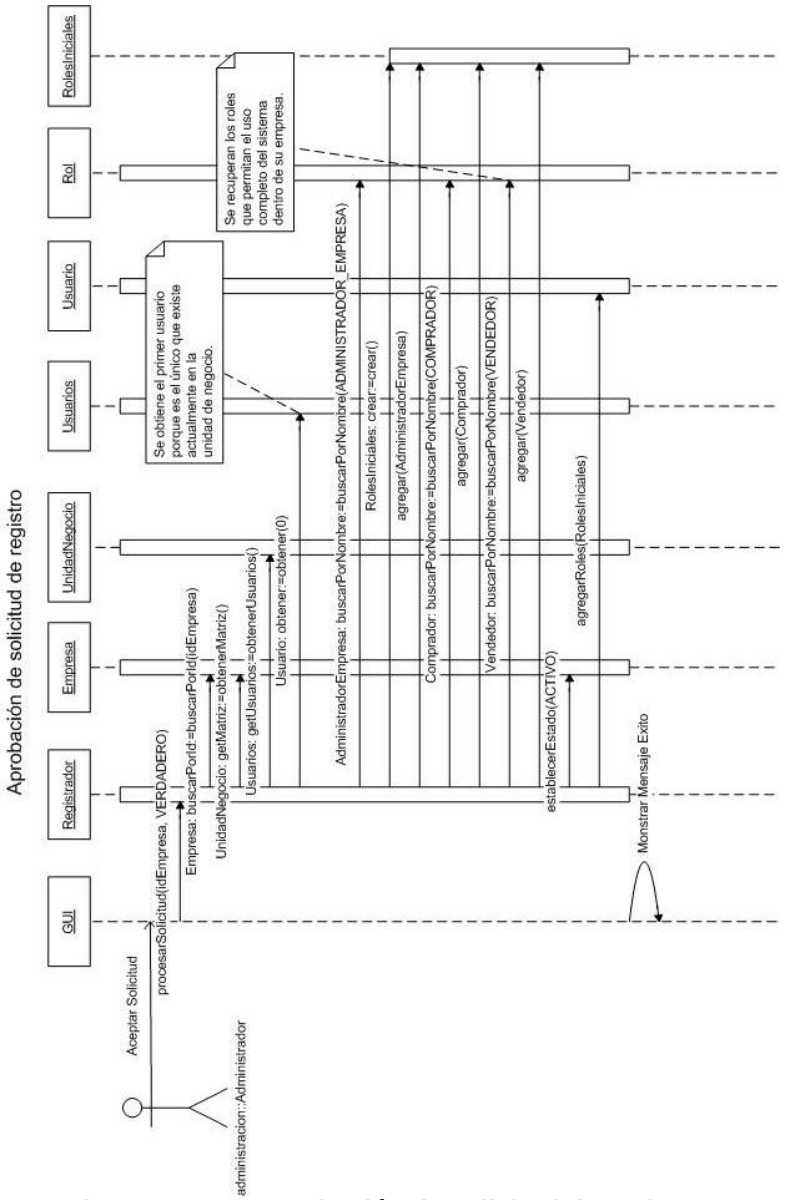


Figura 3-5 DIO Aprobación de solicitud de registro.
Fuente: DIOs del Módulo de Administración del Sistema E-Guana.

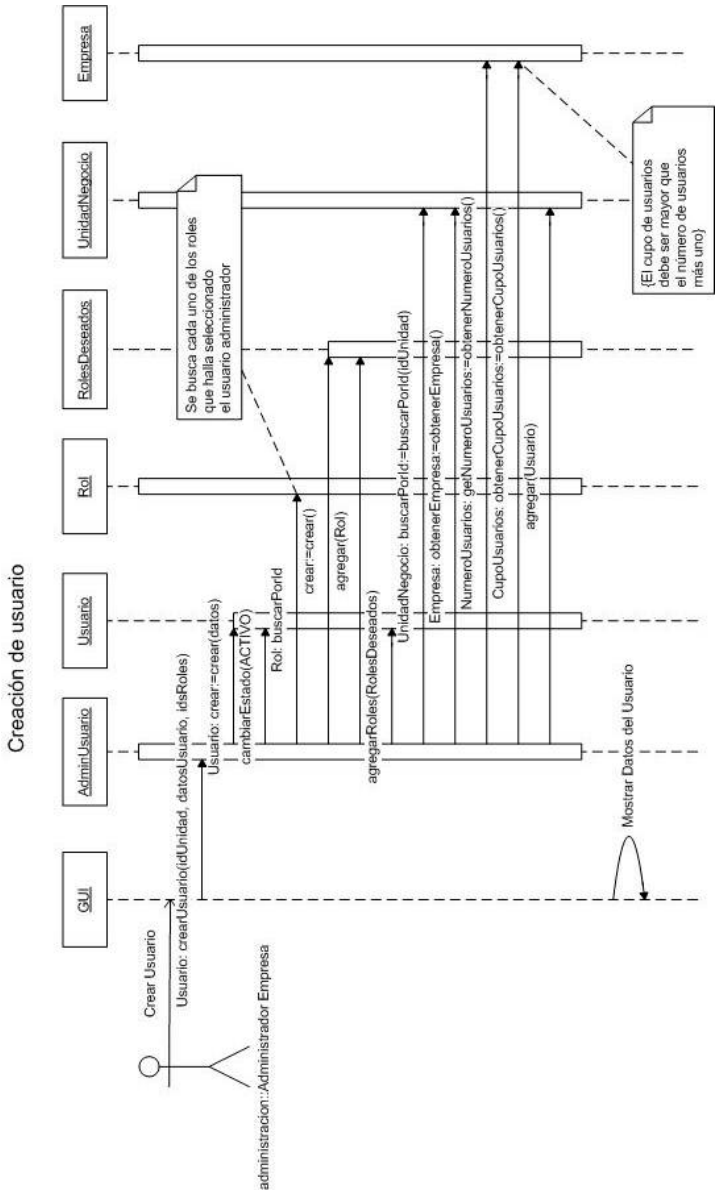


Figura 3-6 DIO Creación de usuario.
Fuente: DIOs del Módulo de Administración del Sistema E-Guana.

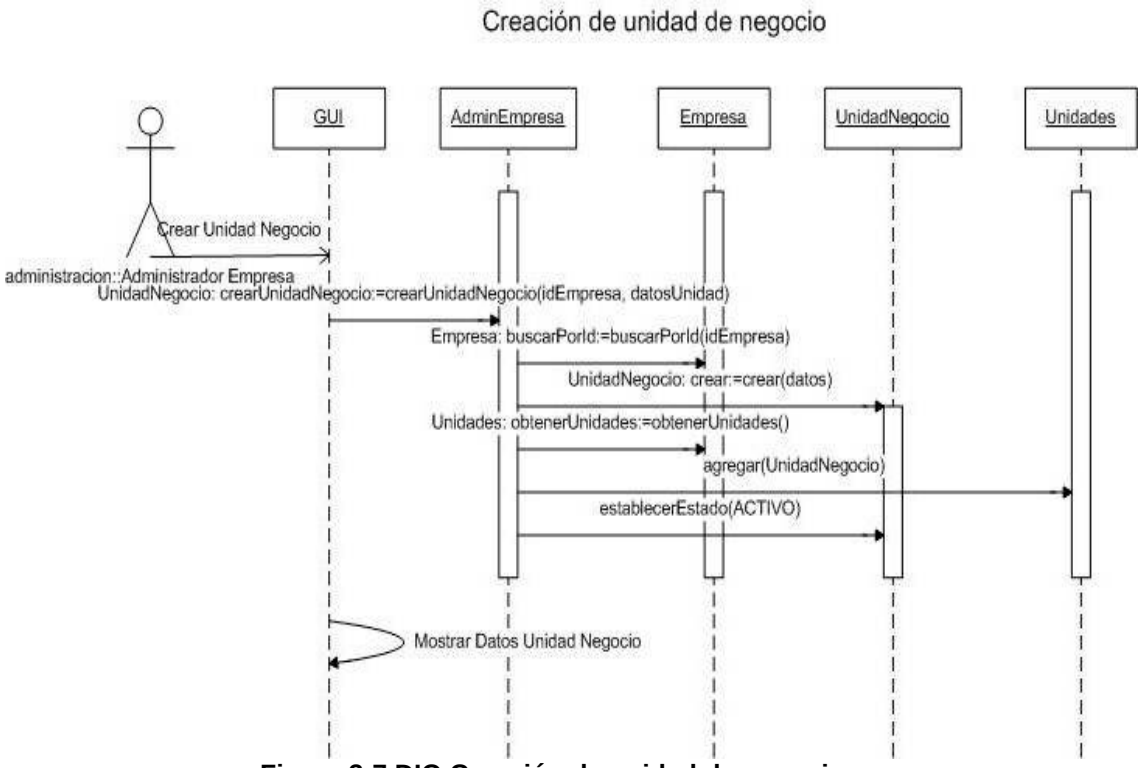


Figura 3-7 DIO Creación de unidad de negocio.
Fuente: DIOs del Módulo de Administración del Sistema E-Guana.

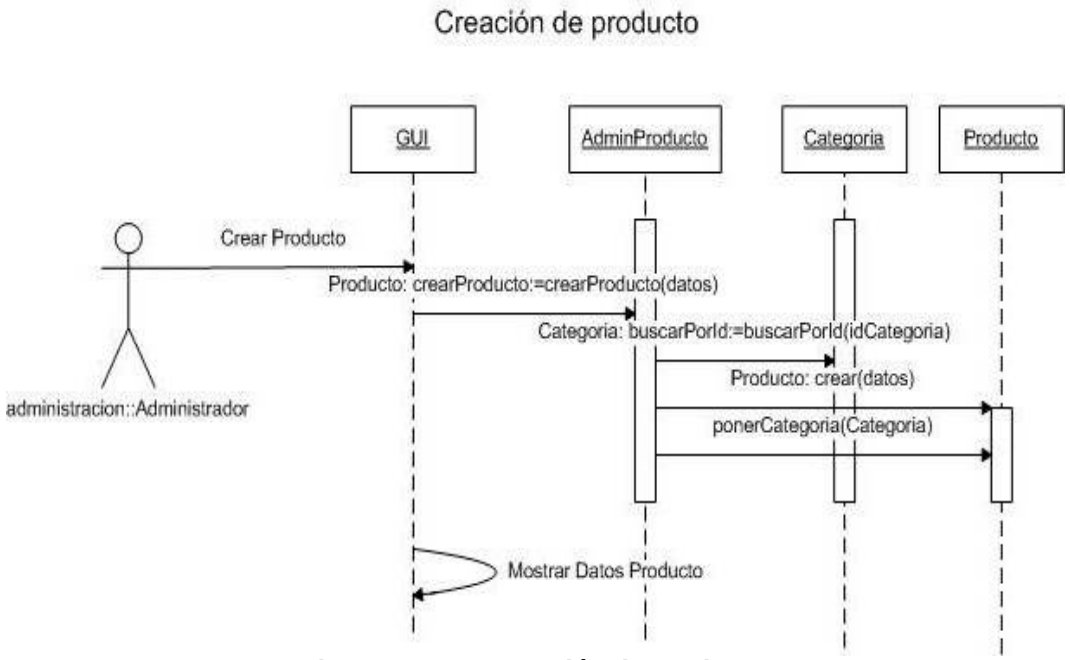


Figura 3-8 DIO Creación de producto
Fuente: DIOs del Módulo de Administración del Sistema E-Guana.

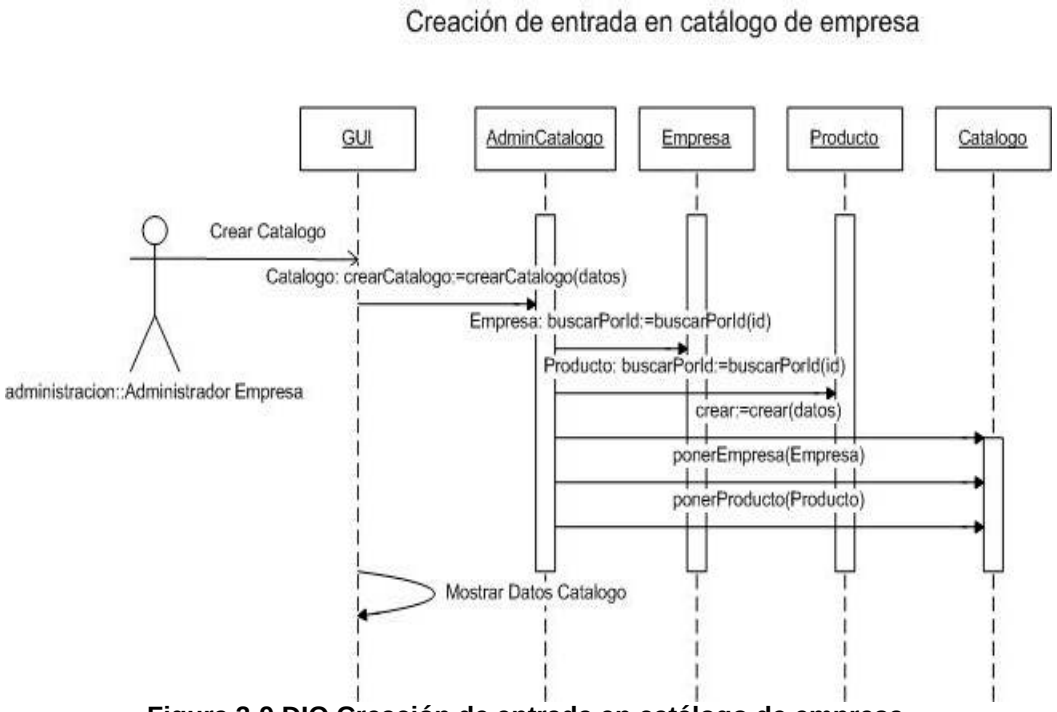


Figura 3-9 DIO Creación de entrada en catálogo de empresa.
Fuente: DIOs del Módulo de Administración del Sistema E-Guana.

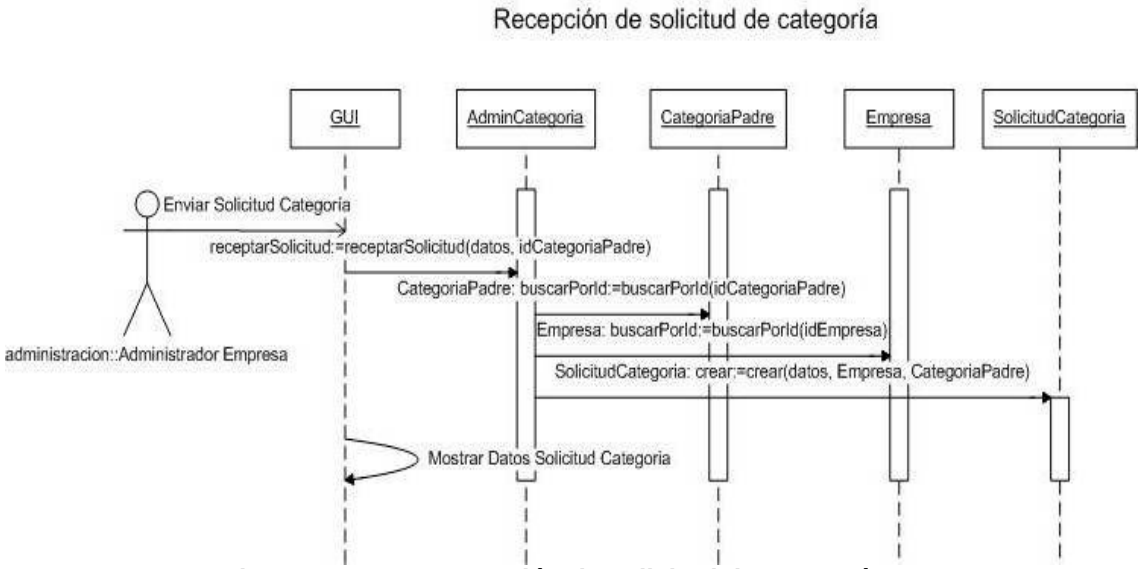


Figura 3-10 DIO Recepción de solicitud de categoría.
Fuente: DIOs del Módulo de Administración del Sistema E-Guana.

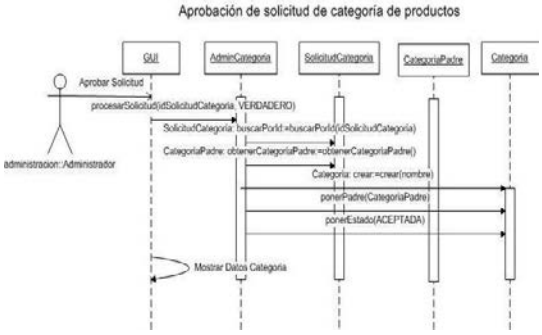


Figura 3-11 DIO Aprobación de solicitud de categoría.
Fuente: DIOs del Módulo de Administración del Sistema E-Guana.

Figura 3-12 DIO Recepción de solicitud de cupo de usuarios.
Fuente: DIOs del Módulo de Administración del Sistema E-Guana.

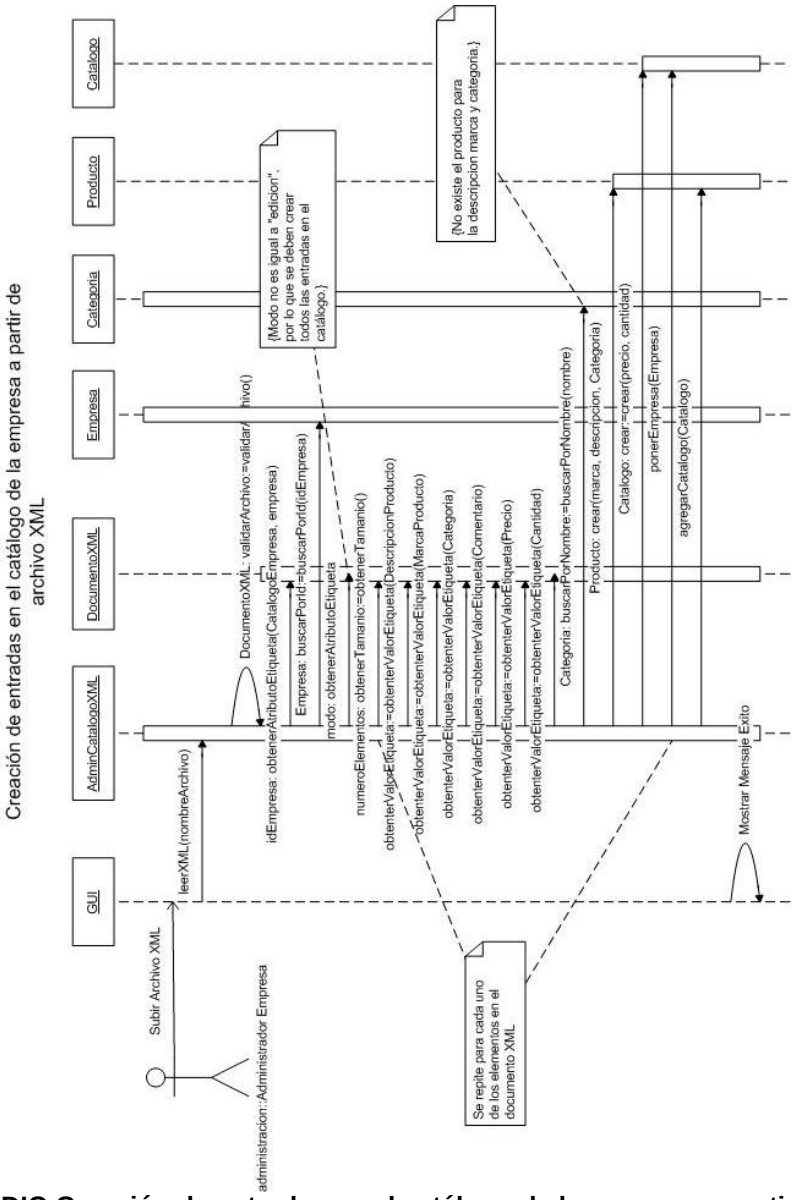


Figura 3-13 DIO Creación de entradas en el catálogo de la empresa a partir de archivo XML.
Fuente: DIOs del Módulo de Administración del Sistema E-Guana.

Figura 3-14 DIO Aprobación de solicitud de cupo de usuarios.
Fuente: DIOs del Módulo de Administración del Sistema E-Guana.

3.3.3. Modelo Entidad-Relación

El modelo Entidad-Relación del Módulo de Administración describe simplificadamente el diseño de la base de datos de todas las entidades que participan en el Módulo.

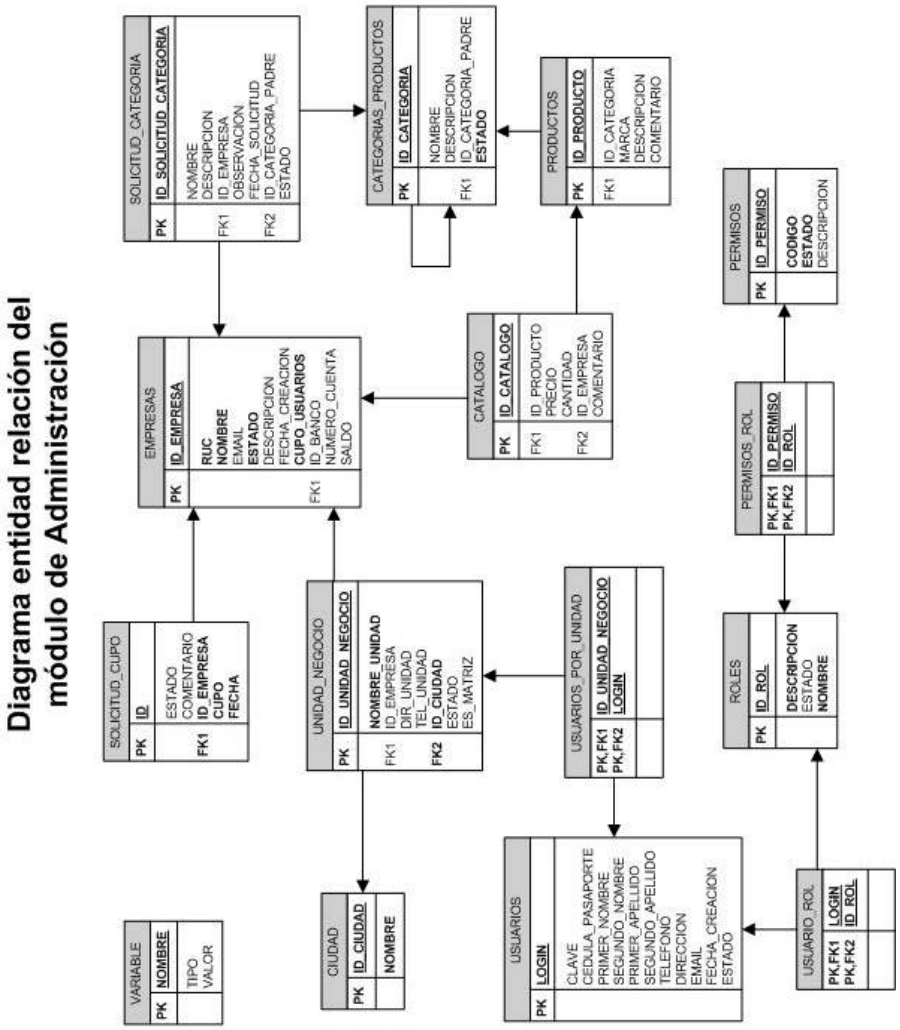


Figura 3-15 Diagrama entidad relación del módulo de Administración.
Fuente: Diagramas Entidad-Relación del Sistema E-Guana.

3.4. Estructura

3.4.1. Componentes utilizados en la implementación

Los componentes utilizados en la implementación se agrupan en EJBs, JSPs y etiquetas JSP personalizadas, todos ellos desplegados dentro de una aplicación J2EE. En esta sección del documento se describirán cada uno de estos componentes.

3.4.1.1. EJBs

Para el Módulo de Administración se utilizan dos tipos de *Enterprise* Java *Beans:* de entidad y de sesión.

Paquete	Nombre
org.eguana.administracion.bo.ejb	EntityBeanBase
org.eguana.administracion.bo.ejb	EmpresaBean
org.eguana.administracion.bo.ejb	ProductoBean
org.eguana.administracion.bo.ejb	CategoriaBean
org.eguana.administracion.bo.ejb	CatalogoBean
org.eguana.administracion.bo.ejb	SolicitudCategoriaBean
org.eguana.administracion.bo.ejb	UnidadNegocioBean
org.eguana.administracion.bo.ejb	UsuarioBean
org.eguana.administracion.bo.ejb	RolBean
org.eguana.administracion.bo.ejb	PermisoBean
org.eguana.administracion.bo.ejb	VariableBean
org.eguana.administracion.bo.ejb	SolicitudCupoBean

Tabla 3-1 *Beans* de Entidad que forman parte del Módulo de Administración

Paquete	Nombre
org.eguana.administracion.wf.ejb	SessionBeanBase
org.eguana.administracion.wf.ejb	AdminBean
org.eguana.administracion.wf.ejb	AdminUsuarioBean
org.eguana.administracion.wf.ejb	AdminEmpresa

org.eguana.administracion.wf.ejb	AdminCategoriaBean
org.eguana.administracion.wf.ejb	AdminSolicitudCupoBean
org.eguana.administracion.wf.ejb	AdminProductoBean
org.eguana.administracion.wf.ejb	RegistradorBean
org.eguana.administracion.wf.ejb	AdminCatalogoXMLBean
org.eguana.administracion.wf.ejb	AdminRolPermisoBean
org.eguana.administracion.wf.ejb	NavegadorCatalogoBean

Tabla 3-2 *Beans* de Sesión que forman parte del Módulo de Administración

Los *beans* de entidad, como se puede deducir por sus nombres, corresponden a la implementación de las clases del módulo, en tanto que los *beans* de sesión implementados exponen la funcionalidad de todo el módulo.

Beans de entidad

En esta sección se describirá cada uno de los *beans* de entidad más importantes implementados en la aplicación.

A menos que se indique lo contrario, cada *bean* está ubicado en el paquete *org.eguana.administracion.bo.ejb*. Se indicarán los atributos del *bean,* luego, de existir, se mencionarán los métodos de acceso para campos de relaciones, es decir los métodos para navegar las relaciones de este *bean* con otros.

Más adelante se describirán los métodos de búsqueda, que pueden ser de tipo *select* o *finder.* Estos métodos “permiten ejecutar consultas, las mismas que son definidas en los descriptores de despliegue *ejb-jar.xml* y *jbosscmp-jdbc.xml*” [15]. Respecto al lenguaje de consulta de estos métodos,

normalmente es EJB-QL pero debido a las limitaciones que se presentaron EJB-QL (la falta del operador *like* por ejemplo) algunas consultas se hicieron en JBOSS QL.

Vale indicar que “los métodos *finder* no son definidos propiamente en el *bean* de entidad sino en las interfaces respectivas” [13], pero al utilizar *XDoclet* es posible, a través de etiquetas o anotaciones especiales, declarar los distintos *finders* en el propio *bean* de entidad para que luego, a través de *Ant* y *XDoclet*, se traduzcan en métodos en las interfaces local o remota, según corresponda, esto facilita enormemente el desarrollo con *EJBs*.

Se describirán también los métodos de negocio, esto es, “los métodos que contienen la lógica del negocio” [13] en los *beans* de entidad.

Cada *bean* de entidad requiere una clave única para diferenciarlos del resto de los *beans* de su tipo, normalmente este atributo se maneja utilizando una cadena de texto (usando la clase de Java java.lang.String) o un número entero (a través de la clase java.lang.Integer), pero en ciertos casos, para claves compuestas por ejemplo, se deben implementar clases personalizadas, en E-Guana se recurrió a esta última forma, en la mayoría de los casos, puesto que se gana flexibilidad en el escenario que se deseen hacer cambios en la implementación. Estas clases se ubican en el paquete

org.eguana.administracion.keys y su nombre tiene como sufijo las letras PK (*primary key*).

EntityBeanBase

Esta es una clase abstracta útil que implementa los métodos establecidos por el contrato dado por la especificación *EJB* y que permite reducir el código en cada *bean* de entidad que derive de esta clase.

Adicionalmente tiene una variable protegida con el contexto del *bean* de entidad para quien lo necesite.

Esta clase tiene un par de métodos útiles para las clases derivadas, *getUsuario* que retorna el usuario actual del cliente que está utilizando el *bean* y *tieneRol (java.lang.String)* que retorna verdadero si el cliente que utiliza el *bean* tiene el rol identificado por el argumento. Estos métodos se apoyan en el contexto del *bean* de entidad.

EmpresaBean

El *bean* de entidad EmpresaBean representa una Empresa. Esta clase extiende la clase *org.eguana.comun.secuencia.ClienteSecuencia* y por tratarse de un *bean* de entidad implementa la interfaz *javax.ejb.EntityBean* además de una interfaz de constantes útiles *org.eguana.administracion.bo.util.Constantes*.

Esta clase implementa los métodos de acceso para cada uno de los atributos definidos en la clase Empresa: ID, nombre, email, RUC, estado, fecha de creación, descripción, cupo de usuarios, número de cuenta (para mayor flexibilidad este atributo se trató como una cadena de texto en lugar de un número) y saldo (de la cuenta de la empresa en el banco respectivo).

- Métodos de acceso para campos de relaciones.

En cuanto a las relaciones, en E-Guana una empresa tiene varias unidades de negocio (de las cuales una es la matriz), para implementar esto, se tiene una pareja de métodos que manejan esta colección de unidades.

De igual forma toda empresa registrada en el sistema puede publicar un catálogo de productos, en E-Guana este catálogo es de hecho un conjunto de entradas tipo catálogo, por lo que es este conjunto el que constituye el catálogo. Con dos métodos que utilizan colecciones de objetos se maneja esta relación.

Finalmente una empresa tiene una cuenta bancaria en alguno de los bancos registrados en E-Guana, para acceder a estos datos se tiene la pareja de métodos que hace uso del *bean BancoBean* del módulo de Pagos de Transacciones.

- Métodos de búsqueda

Este *bean* provee métodos para la búsqueda de empresas de acuerdo a varios criterios: estado, nombre, RUC, tanto en el caso que retorna un solo registro o para colecciones de registros.

Entre los métodos *select,* se tiene uno que retorna todos los usuarios que pertenecen a todas las unidades de negocio de la empresa que corresponde a este *bean.* Otros métodos provistos permiten buscar unidades de negocios de la empresa, incluido uno que busca entre todas las unidades de la empresa, a aquella que es la matriz (o unidad de negocio principal) de la misma. Puesto que los usuarios pertenecen a las unidades y no a la empresa propiamente, esta clase también permite contabilizar el número de total de usuarios a través de un método *select*.

- Métodos de negocio

Los métodos de negocio de este *bean* hacen uso de los métodos *select* descritos anteriormente, tanto para establecer la matriz entre las unidades de negocio, como para retornar el número de usuarios que tiene la empresa.

Otro método de negocio importante es el que establece el estado de la empresa, ya que no sólo cambia el valor del atributo *estado* sino que también lo hace en cada una de sus unidades de negocio.

ProductoBean

Este *bean* de entidad representa un Producto en el sistema. Deriva de la clase ClienteSecuencia.

Los atributos de este *bean* son: ID, marca, descripción y comentario.

- Métodos de acceso para campos de relaciones

Un producto está relacionado con los catálogos de las empresas y con las categorías. El producto puede referenciar a muchos catálogos en tanto que un producto sólo puede pertenecer a una categoría. Por lo que para el primer caso se tiene una pareja de métodos que utiliza colecciones, en tanto que para el segundo, los métodos hacen uso directo de la clase CategoriaBean.

- Métodos de búsqueda

La clase ProductoBean provee métodos *finder* para la búsqueda de productos de acuerdo a varios criterios: marca, descripción tanto entre todos los productos como entre los que están siendo utilizados en el catálogo de una empresa en particular, y por la categoría a la que pertenece.

CategoriaBean

Este *bean* de entidad corresponde a la clase Categoría, se deriva de *ClienteSecuencia* e implementa *EntityBean* y *Constantes*.

Los atributos de la clase Categoría son: ID, nombre, descripción y estado.

- Métodos de acceso para campos de relaciones

Cada categoría tiene un sinnúmero de productos asociados a la misma, además cada categoría puede tener subcategorías, en ambos casos se requieren parejas de métodos que utilicen colecciones de objetos para manejar la relación, aunque para la segunda se requiere una pareja adicional de métodos que permitan establecer y obtener la categoría padre.

- Métodos de búsqueda

Entre los métodos *finder* de esta clase se tiene uno que retorna todas las categorías raíces, es decir, las que no tienen una categoría padre establecida y otro que permite buscar las categorías por el valor del atributo estado.

CatalogoBean

Este *bean* de entidad corresponde a la tabla *catalogo* en la base de datos. La clase deriva de ClienteSecuencia e implementa EntityBean.

Los atributos de la clase Catálogo son: ID, precio, cantidad y comentario.

- Métodos de acceso para campos de relaciones

Cada catálogo (o una entrada en el catálogo propiamente dicho) hace referencia a un producto, esta relación se maneja con métodos que se refieren directamente a la clase *ProductoBean.*

Además, cada catálogo pertenece a una empresa y para manejar esta relación se tiene la pareja de métodos que establece y recupera una referencia a la clase *EmpresaBean*.

SolicitudCategoriaBean

Este *bean* de entidad corresponde a la clase Solicitud de Categoría, que maneja la persistencia de las solicitudes de categorías de productos. Esta clase se deriva de ClienteSecuencia e implementa EntityBean y Constantes.

Los atributos de la clase SolicitudCategoria son: ID, nombre, descripción, observación, estado y fecha.

- Métodos de acceso para campos de relaciones

Cada solicitud de categoría es hecha por una empresa, más precisamente por el “Administrador de la Empresa” solicitante, por lo tanto para manejar esta relación se tienen métodos que apunten a una instancia de la clase *EmpresaBean*.

Además, cada solicitud debe referirse a una categoría de productos denominada padre, es decir la categoría en la que se desea incluir la nueva categoría, debido a esto, se tiene los métodos que referencian a una *CategoriaBean*.

- Métodos de búsqueda

Entre los métodos *finder* de este *bean* de entidad, se tienen los que permiten buscar solicitudes por criterio: estado, y/o solicitante. Los datos retornados se ordenan por la fecha de recepción descendentemente.

UnidadNegocioBean

Es el *bean* de entidad que corresponde a la unidad de negocio. Se deriva de la clase ClienteSecuencia e implementa las interfaces EntityBean y Constantes.

Los atributos de la unidad de negocio son: ID, nombre, dirección, teléfono, ciudad, estado y el atributo que permite discriminar la matriz de la empresa (la unidad de negocio principal).

- Métodos de acceso para campos de relaciones

La unidad de negocio pertenece a una empresa. Los usuarios de una empresa pertenecen de hecho a una unidad de negocio. La relación con la empresa se implementa con métodos que apunten a la instancia de la clase *EmpresaBean* respectiva, mientras que la relación con los usuarios se maneja con colecciones de objetos de tipo *UsuarioBean.*

- Métodos de búsqueda

Entre los métodos *finder* de este *bean* se tienen los que permiten buscar unidades de negocio de acuerdo al estado, a la empresa a la que pertenecen y otro que retorna todas las unidades del sistema.

Esta clase provee métodos *select* que permiten buscar entre los usuarios que pertenecen a la unidad de acuerdo al nombre, al número de cédula o al usuario.

- Métodos de negocio

Esta clase tiene dos métodos de negocio, el primero establece el estado de la unidad de negocio y de todos los usuarios integrantes de la misma, en tanto que el otro hace uso de los métodos *select* de búsqueda de usuarios para exponer dicha funcionalidad a otros *beans* (la especificación EJB sólo permite la disposición de los métodos *select* dentro de la clase que los define).

UsuarioBean

Este *bean* de entidad es la implementación de la clase Usuario. Deriva de EntityBeanBase e implementa EntityBean.

Los atributos de expuestos por este *bean* son: usuario, clave, primer nombre, primer apellido, email, estado, cédula, teléfono y fecha de creación.

- Métodos de acceso para campos de relaciones

Dentro de la empresa, cada usuario debe tener uno o más roles para poder hacer correcto uso del sistema. En el nivel de los EJBs este *bean* provee los métodos necesarios para obtener y establecer los roles, en tanto que en el nivel de la base de datos se utiliza la tabla usuario_*rol* para el manejo de la relación.

- Métodos de búsqueda

Entre los métodos *finder* implementados por esta clase se encuentra uno que permite buscar un usuario en base al nombre de usuario, además de otros para buscar un conjunto de usuarios de acuerdo a su nombre, apellido o nombre de usuario.

- Métodos de negocio

Cada usuario está relacionado directamente a una unidad de negocio, pero una funcionalidad útil es determinar la empresa en la que se encuentra dicha persona, para esto se provee un método que retorna el identificador de la misma.

RolBean

Este *bean* de entidad es la implementación de la clase Rol. Deriva de ClienteSecuencia e implementa EntityBean.

Los campos persistentes de Rol son: ID, nombre y descripción.

- Métodos de acceso para campos de relaciones

Puesto que cada rol tiene un conjunto de permisos asociados, esta clase provee métodos para el manejo de dicha relación a través de listas de objetos. En la base de datos esta relación se maneja a través de la tabla *PERMISOS_ROL* donde cada registro tiene el identificador del permiso (*id_permiso*) y el identificador del rol (*id_rol*).

- Métodos de búsqueda

La clase RolBean expone un método *finder* que permite buscar un rol de acuerdo a su nombre y otro que realiza la búsqueda de acuerdo al nombre y un permiso específico.

Adicionalmente este *bean* de entidad implementa un método de ayuda tipo *select* que retorna una instancia de PermisoBean correspondiente a uno de los permisos del rol actualmente instanciado.

- Métodos de negocio

El único método de negocio expuesto por esta clase es uno que retorna una interfaz local de PermisoBean de acuerdo al método *select* descrito anteriormente.

PermisoBean

Este *bean* de entidad es la implementación de la clase Permiso y trabaja en conjunto con la clase RolBean. Deriva de la clase ClienteSecuencia e implementa la interfaz EntityBean.

Los campos persistentes de Permiso son: ID, código y descripción.

VariableBean

Este *bean* de entidad es la implementación de la clase Variable, que permite la administración de las distintas variables de configuración del sistema. Deriva de EntityBeanBase e implementa la interfaz EntityBean.

Los atributos de una variable son: nombre, valor y tipo. Este último atributo puede tomar los valores “s” si representa una cadena de texto ó “i” se trata de un número.

Esta clase provee métodos que facilitan la búsqueda de una variable y la conversión de los valores numéricos, esto último es importante anotar

considerando que el atributo valor se almacena en la base de datos como texto.

SolicitudCupoBean

Este *bean* de entidad mapea la tabla *solicitud_cupo* y corresponde al modelo para la funcionalidad de la solicitud y aprobación del cupo de usuarios que puede crear una empresa.

Los atributos de la solicitud son: ID, estado, comentario, cupo, ID de la empresa que hace la solicitud y fecha de la solicitud.

El cupo máximo que puede ser solicitado y asignado depende de la variable **CupoUsuariosMaximo**.

Este *bean* provee un método *finder* para la búsqueda de la solicitud que tiene pendiente una empresa (cada empresa sólo puede tener una solicitud pendiente).

Beans de sesión

Esta sección describirá el funcionamiento general de los principales *beans* de sesión implementados en el sistema.

Los *beans* de sesión están ubicados en el paquete *org.eguana.administración.wf.ejb*.

SessionBeanBase

Esta es una clase útil para todos los *beans* de sesión del sistema, ya que implementa los métodos pedidos por la especificación EJB.

Almacena en un variable el contexto del *bean* de sesión (javax.ejb.SessionContext), necesario para soportar otros métodos.

Además tiene una variable, junto con el método de acceso, de la interfaz local del *bean* DTOFactoryBean, otro *bean* útil, que genera objetos de transferencia de datos (DTO: Data Transfer Object) [10].

Otro método implementado es *getUsuario ()*, que retorna el usuario actual del cliente que utiliza el *bean*.

Para determinar si el usuario actual tiene un rol dado, se provee el método *tieneRol (java.lang.String)*, el argumento es el nombre del rol.

Ya que muchas operaciones dependen de la empresa en la cual se están realizando, esta clase tiene el método *getEmpresaPK ()*, que retorna la clave primaria de la empresa a la que pertenece el usuario actualmente conectado.

AdminBean

Este *bean* de sesión agrupa la mayor parte de la funcionalidad disponible para clientes remotos de la aplicación. Esto lo hace a través de la delegación

de funcionalidad a otros *beans* administradores especializados: *AdminUsuarioBean*, *AdminEmpresaBean* y *AdminSolicitudCupoBean*, los anteriores trabajan en el ámbito de interfaces locales, pero como dichas interfaces no pueden ser utilizadas en clientes remotos, AdminBean se encarga de crear, a partir de las interfaces locales y cuando corresponda, los objetos de transferencia de datos (DTO), que son pasados al cliente remoto.

AdminUsuarioBean

Este *bean* de sesión que agrupa los servicios relacionados a la entidad Usuario.

Dentro de las operaciones de mantenimiento que realiza esta clase, se encuentra la etapa de creación de un usuario, en la misma se toma en consideración que el cupo de usuarios no sobrepase el valor establecido.

Entre otras de las tareas de esta clase, están la asignación de roles y el cambio de estado de un usuario.

AdminEmpresa

Esta clase es un *bean* de sesión que permite administrar las empresas y unidades de negocio del sistema. Además de realizar búsquedas de empresas de acuerdo a distintos criterios, desde el estado, pasando por el RUC o el nombre.

Adicionalmente provee métodos para buscar usuarios que pertenecen a una empresa específica.

AdminCategoriaBean

Este *bean* de sesión permite tanto administrar las categorías de productos del sistema, así como el manejo de solicitudes de nuevas categorías.

Las solicitudes de categorías de productos son presentadas por el "Administrador de la Empresa" y es el "Administrador del Sistema" quien las aprueba o rechaza. Esta clase también provee la funcionalidad para consultar las solicitudes hechas por una empresa en particular.

AdminSolicitudCupoBean

Cada empresa registrada en el sistema tiene un cupo de usuarios que puede crear, si dicho cupo lo considera insuficiente, el "Administrador de la Empresa" puede solicitar el aumento del cupo y el "Administrador del Sistema" es el encargado de aprobar o rechazar esta solicitud. Esta funcionalidad es manejada a través del *bean* de sesión AdminSolicitudCupoBean.

AdminProductoBean

AdminProductoBean es el *bean* de sesión que permite administrar productos en el sistema.

Adicional a los métodos de mantenimiento, esta clase provee métodos para la búsqueda de productos a partir de su marca y/o descripción, en ambos casos pudiendo ser una consulta general o dentro de los productos del catálogo de una empresa. Otras búsquedas retornan los productos de acuerdo a la categoría a la que pertenecen.

RegistradorBean

Este *bean* maneja el registro de empresas en el sistema.

Inicialmente el usuario que será el “Administrador de la Empresa”, en el contexto de E-Guana, debe llenar un formulario de registro con los datos que identifican a la empresa y sus datos personales. Este registro debe luego ser revisado por el “Administrador del Sistema” para su aprobación o rechazo.

El resultado del registro exitoso comprende una empresa que tiene una unidad de negocio (la matriz) y esta última tiene usuario con los roles “Administrador de Empresa”, “Comprador” y “Vendedor”, las tres entidades se encuentran en estado “Activo”.

AdminCatalogoXMLBean

AdminCatalogoXMLBean es el *bean* de sesión que permite el mantenimiento del catálogo de productos de la empresa a través de XML.

El “Administrador de la Empresa” puede descargar en su máquina, en formato XML, el catálogo que tiene publicado en el sistema. También puede actualizar la información del catálogo enviando un archivo XML con la nueva información.

De esta forma se agiliza el mantenimiento del catálogo, al poder preparar las modificaciones fuera de línea y luego, cuando las tenga listas, hacer una sola actualización por lotes.

El DTD asociado al archivo XML del catálogo es:

```
<!ELEMENT CatalogoEmpresa (Catalogo+)>
<!ATTLIST CatalogoEmpresa empresa CDATA #IMPLIED modo
(insercion|edicion) #IMPLIED>
<!ELEMENT Catalogo
(DescripcionProducto, MarcaProducto, Precio, Cantidad, Categoria,
Comentario?)>
<!ATTLIST Catalogo id CDATA #IMPLIED>
<!ELEMENT DescripcionProducto (#PCDATA)>
<!ELEMENT MarcaProducto (#PCDATA)>
<!ELEMENT Precio (#PCDATA)>
<!ELEMENT Cantidad (#PCDATA)>
<!ELEMENT Categoria (#PCDATA)>
<!ELEMENT Comentario (#PCDATA)>
```

La explicación del DTD es la siguiente:

El archivo describe el catálogo de una empresa (CatalogoEmpresa) y está conformado por una o más entradas del catálogo (Catalogo).

Para identificar la empresa que está realizando la operación, “CatalogoEmpresa” tiene el atributo “empresa”. Además para identificar si se

está haciendo una modificación o una nueva inserción existe el atributo “modo” que puede tomar los valores de “insercion” o “edicion”.

Cada elemento “Catalogo” está formado por los elementos “DescripcionProducto” que corresponde a la descripción del producto, “MarcaProducto” es la marca del producto, “Precio” es el precio del producto en el catálogo de la empresa, “Cantidad” corresponde a la cantidad de este producto que se ofertará dentro del sistema, “Categoria” es la categoría de productos en la que se desea ubicar al producto y de igual manera “Comentario” corresponde a algún comentario que se considere pertinente.

Adicionalmente el elemento “Catalogo” tiene el atributo “id” que corresponde al identificador del catálogo que se está editando.

AdminRolPermisoBean

Este *bean* de sesión permite administrar los roles y permisos del sistema.

Un permiso corresponde a una acción o función que se puede efectuar dentro del sistema. Cada uno de estos permisos está asociado a uno o más de los roles existentes en el sistema.

Además este *bean* provee un método que es utilizado por otras clases para determinar si un permiso pertenece a una lista de roles dados.

NavegadorCatalogoBean

Este *bean* de sesión provee una funcionalidad útil para la navegación del catálogo de productos de una empresa.

Esta funcionalidad comprende retorno del listado de ítems en el catálogo de acuerdo a la empresa, al producto o a la categoría de productos. Además esta clase soporta los datos necesarios para poder representar gráficamente la jerarquía de categorías de productos.

El problema de las claves secuenciales

Normalmente para la generación de campos identificadores en sistemas de base de datos, se utilizan campos de tipo incremental o serial.

El problema surge cuando se desea referenciar a la entidad a través de este identificador justo luego de la creación de la misma, aunque no se conoce el valor que generó el motor de la base de datos.

Una posible solución es utilizar alguna función específica de la base de datos para obtener el último valor agregado.

El inconveniente de lo anterior, es la dependencia del motor de base de datos, lo que implica disminuir la portabilidad del sistema a otro servidor de aplicaciones o a otro motor de base datos.

En el caso de E-Guana se implementó un mecanismo para la generación de claves que no dependa de la base de datos. Dicha implementación se basa en la solución propuesta por Floy Marinescu en EJB Design Patterns [11].

Los *beans* necesarios para soportar este mecanismo de generación de claves son:

Paquete	Nombre
org.eguana.administracion.bo.ejb	ClienteSecuencia
org.eguana.comun.secuencia	SecuenciaBean
org.eguana.comun.secuencia	Secuenciador

Tabla 3-3 *Beans* que soportan el mecanismo de generación de claves.

A continuación se detalla la participación de estos *beans* en el proceso.

ClienteSecuencia

Esta es una clase abstracta de utilidad y forma parte del mecanismo para la generación de las claves primarias de algunos *beans* de entidad. Debido a que esta clase tiene que ser derivada por clases que serán propiamente *beans* de entidad, ésta a su vez se deriva de *EntityBeanBase* para hacer uso de las utilidades provistas por esta última.

ClienteSecuencia tiene sólo dos métodos el primero *protected abstract java.lang.String getNombreSecuencia ()* es el método que debe ser implementado por las clases que deriven de *ClienteSecuencia*, el valor a

retornar corresponde a una cadena de texto cuyo valor debe existir en la tabla *secuencia* de la base de datos.

El otro método de esta clase es *protected int getNextKey () throws javax.ejb.EJBException*, éste lo que hace es retornar el siguiente número que corresponde a la secuencia dada por *getNombreSecuencia (),* el cálculo del número se lo delega al *bean* de sesión *Secuenciador*, del cual ClienteSecuencia tiene una referencia a través de una variable privada.

SecuenciaBean

Este *bean* de entidad forma parte del mecanismo para la generación de claves secuenciales en el sistema.

Este *bean* tiene dos atributos *name* y *currentKeyValue*, *name* corresponde al nombre de la secuencia para la cual se generan las claves y *currentKeyValue* es el valor de la primera clave del bloque actual.

El método *getNextKeyAfterIncrementBy (int)* retorna el valor de *currentKeyValue* luego de incrementar su valor en la cantidad indicada por el argumento. Es importante indicar que este método se marca con el tipo de transacción *RequiresNew* para que así requerimientos concurrentes de una clave en una misma secuencia no retornen el mismo valor.

Secuenciador

Este *bean* de sesión controla la lógica para la generación de las claves para los *beans* que lo necesiten. Este *bean* trabaja en conjunto con *SecuenciaBean* y con los *beans* que deriven de *ClienteSecuencia*.

El funcionamiento es el siguiente, un ClienteSecuencia solicita una nueva clave a través de *getNexKey*. Este método le pide a Secuenciador que le retorne el siguiente número de la secuencia asociada a ClienteSecuencia. Secuenciador busca en una estructura de datos de soporte el bloque de claves que le corresponden a la secuencia pedida. Si el bloque no existe o está vacío se debe crear y/o llenar el bloque, según corresponda, con el siguiente grupo de claves. Para lo anterior inicialmente hay que determinar la primera clave del bloque, buscando al *bean* SecuenciaBean que corresponde al nombre de secuencia que se está generando y llamando al método *getNextKeyAfterIncrement*. Con este valor se llena el bloque (se lo crea antes si es la primera vez) y con los siguientes valores consecutivos al valor retornado por *getNextKeyAfterIncrement* hasta el tamaño preestablecido del bloque.

El valor que se retorna finalmente es el primer valor del bloque que además debe ser removido del mismo para que no sea tomado por otra instancia.

3.4.1.2. JSPs

En esta sección se describirá el funcionamiento de las páginas más importantes del sistema. Debido a que la funcionalidad de la página no sólo está conformada por el código en la página, también se describirán las clases asociadas más importantes.

Antes se dará una descripción de la manera en que se utilizó el *framework* Sofia en las páginas JSP.

Sofia

El módulo de Administración de E-Guana fue desarrollado en su mayor parte utilizando el *framework* Sofia de Salmon LLC en su versión 2.2.

“El Framework Abierto de Salmon para Aplicaciones de Internet (SOFIA) es una herramienta de desarrollo rápido de aplicaciones para J2EE. Su propósito es permitir a los desarrolladores en Java crear aplicaciones Web, con acceso a base de datos de alta calidad consistentes y mantenibles, y sitios Web rápida y fácilmente” [12].

Sofia está construido sobre el patrón de diseño MVC (Modelo Vista Controlador), en esta implementación, el modelo corresponde a la clase *com.salmonllc.sql.BeanDataStore*, la vista la constituye la página JSP, que

utiliza etiquetas personalizadas de este *framework*, y por último el controlador es la clase *com.salmonllc.jsp.JspController*.

“Sofia provee un completo modelo de persistencia basado en JDBC así como un modelo para envolver otros modelos de persistencia diferentes” [12], es este último acercamiento el que se utilizó para el desarrollo de la capa Web en E-Guana, puesto que este modelo de persistencia ya es provisto por los EJBs.

La clase *BeanDataStore* provee la interfaz entre Sofia y los EJBs, más específicamente, el modelo del *framework* se comunica con un *bean* de sesión que es el que realmente hace las consultas y las modificaciones necesarias, los objetos que se pasan en esta comunicación son DTOs que reflejan el o los *beans* de entidad que se desean consultar o modificar en una página. Esta clase recibe como parámetro en su constructor el tipo de objeto que va a mostrar o manipular, es decir el DTO. Para la reducción de código y facilidad de mantenimiento se implementó una clase abstracta derivada de ésta, *org.eguana.administracion.web.ModeloBase*, que contiene lógica para el borrado, actualización y consulta. Además define métodos abstractos para determinar el *bean* de sesión al cual se le está haciendo la interfaz, así como los nombres de los métodos para la consulta, borrado y actualización.

El controlador interpreta las acciones del usuario sobre la vista y realiza los cambios en el modelo, luego actualiza la presentación para reflejar las modificaciones o redirecciona al cliente a una nueva página.

Otro punto importante a destacar de Sofia, es el de “los componentes inteligentes de interfaz gráfica de usuario, es decir, los componentes pueden automáticamente recordar su estado o pueden ser relacionados a columnas en el modelo para reflejar automáticamente sus valores en la página y el controlador” [12], esto facilita el desarrollo ya que tan solo se necesita relacionar el componente con uno de los atributos del modelo de datos para que el *framework* se encargue de la presentación de los valores.

En resumen cada página o funcionalidad de una página está conformada por un archivo JSP, una clase controladora y un modelo para la manipulación o presentación de los datos.

Controladores generales

Como se señaló previamente todo controlador de Sofia es una clase que deriva de *com.salmonllc.jsp.JspController*, puesto que hay cierta codificación repetitiva o necesaria para cumplir con las interfaces declaradas, se extendió esta clase en dos controladores generales:

- ControlBase

- PaginaEmpresaControl.

Ambos controladores se ubican en el paquete *org.eguana.administracion.web.*

ControlBase

Este es el controlador básico para todos los controladores de las páginas del módulo, implementa interfaces necesarias para la notificación de los eventos de una página así como los eventos de botones o imágenes de tipo *submit*.

Además implementa métodos útiles para acceder a los parámetros del URL, acceso a atributos de sesión, localización de mensajes, notificación de mensajes al cliente entre otras cosas.

PaginaEmpresaControl

Deriva del controlador anterior, debe ser extendido por todos los controladores que requieran los datos de la empresa, trabaja en conjunto con la etiqueta personalizada *empresa*.

Tiene dos variables protegidas de instancia útiles, la una es de tipo *org.eguana.administracion.wf.proxies.AdminProxy*, la que le permite tener acceso al *bean* de sesión *AdministradorBean*, es decir acceso a la mayor parte de la funcionalidad del sistema, la otra variable es *empresa* cuyo tipo es

org.eguana.administracion.dto.UnidadConEmpresaDTO y constituye los datos de la empresa en la que se encuentra el usuario actualmente.

Organización de las páginas

Todas los JSPs del módulo de administración se ubican en subdirectorios del directorio “jsp” de la aplicación Web. El subdirectorio en el que están agrupadas depende del rol al que está dirigida su funcionalidad, así, el directorio “empresa” agrupa las páginas que pueden ser utilizadas por el “Administrador de la Empresa”, mientras que las páginas destinadas al “Administrador del Sistema” se ubican en el directorio “sistema”. Las páginas de errores se ubican en la carpeta “error”, mientras que las páginas que pueden ser accedidas por cualquier usuario, inclusive usuarios anónimos, se ubican en el directorio raíz “jsp”.

Estructura general de las páginas

Todas las páginas del sistema están constituidas por un encabezado, el cuerpo o contenido específico de la misma y el pie de página.

El encabezado es manejado por el fragmento de página “header.jsp”, este fragmento prepara la etiqueta HTML “head” de toda página al incluir los archivos de *Javascript* y hojas de estilo necesarios así como la estructura HTML principal de los componentes, además utiliza la etiqueta personalizada

“usuario” para que la página contenedora pueda acceder a los datos del usuario. Otra función del encabezado es, en caso de estar presentes los datos, mostrar el nombre de la empresa y el nombre de la unidad de negocio a la que pertenece el usuario autenticado. Por último, incluye el menú de funciones de acuerdo al rol del usuario, si se trata del “Administrador del Sistema” incluye el menú “jspf/sistema/menu.jsp” y si es el “Administrador de Empresa” incluye el menú “/jspf/empresa/menu.jsp”.

El pie de página lo maneja el fragmento “footer.jsp”, su función es la de mostrar enlaces a las páginas iniciales de todos los módulos del sistema.

El cuerpo es el contenido en sí de la página, de acuerdo a la vista de la funcionalidad que se está implementando.

El código siguiente muestra la forma en que se construyen las páginas en el sistema:

```
<%@ page extends="com.salmonllc.jsp.JspServlet" %>
…
<salmon:page controller="[nombre de la clase controladora]"/>
…
<jsp:include page="/jspf/header.jsp">
        <jsp:param name="title" value="[título de la página]” />
</jsp:include>
…
<jsp:include page="/jspf/footer.jsp" />
…
<salmon:endpage/>
```

El resultado de lo anterior es algo similar a lo mostrado en la siguiente ilustración:

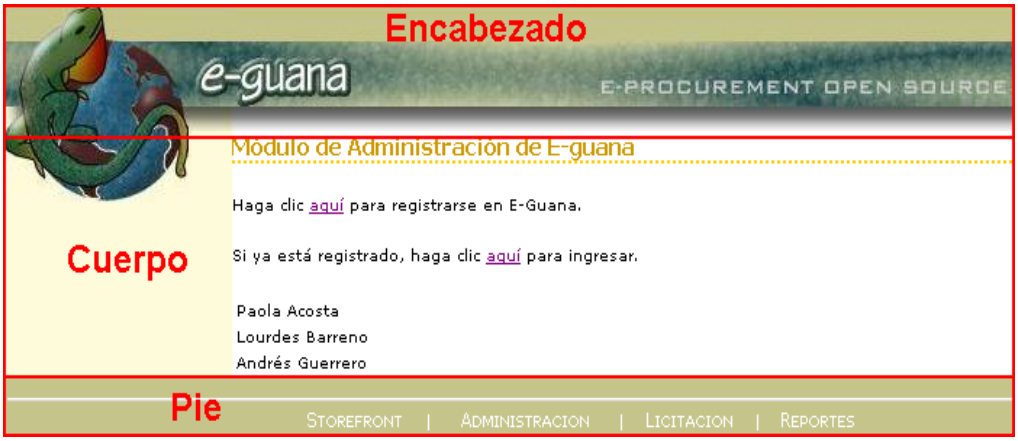


Figura 3-16 Estructura general de las páginas del Sistema E-Guana.

A continuación se entrará en la descripción en sí de las principales páginas del sistema, de acuerdo al orden de los casos de uso relacionados ya presentados.

Registro de empresa

La página JSP para el registro de la empresa en el sistema es “jsp/solicitud.jsp”:

Figura 3-17 Pantalla formulario de registro de empresa

Esta página permite tanto la solicitud de registro de una empresa como la aprobación o negación de dicha solicitud. La primera parte puede hacerla un usuario anónimo, mientras que lo segundo es función exclusiva del "Administrador del Sistema".

Cuando el "Administrador del Sistema" desea aprobar o rechazar una solicitud debe primeramente seleccionar la solicitud, para esto tiene la página "jsp/sistema/solicitudes.jsp":

Figura 3-18 Pantalla solicitudes de registro

Seleccionando la solicitud a revisar, se le presentará la página "jsp/solicitud.jsp" pero en modo de lectura, con los botones para aprobar o negar la solicitud.

Figura 3-19 Pantalla lectura de solicitud de registro

Modificación de Datos de la Empresa

La página para la modificación de los datos de la empresa es "jsp/empresa/empresa.jsp" y luce así:

Figura 3-20 Pantalla modificación de datos de la empresa

Modificación de Estado de una Empresa

El "Administrador del Sistema" es quien puede modificar todos los datos de la empresa, para hacerlo, primero debe seleccionar la empresa entre las registradas en sistema, para esto se tiene la página "jsp/sistema/empresas.jsp":

Figura 3-21 Pantalla empresas registradas

Esta página permite la búsqueda de las empresas por nombre o RUC, le indica el número de unidades de negocio que tiene registradas así como el número de usuarios en total en cada empresa.

Si hace clic en el enlace indicado como “Detalles” se muestra la página de los datos de la empresa aunque con una apariencia ligeramente diferente:

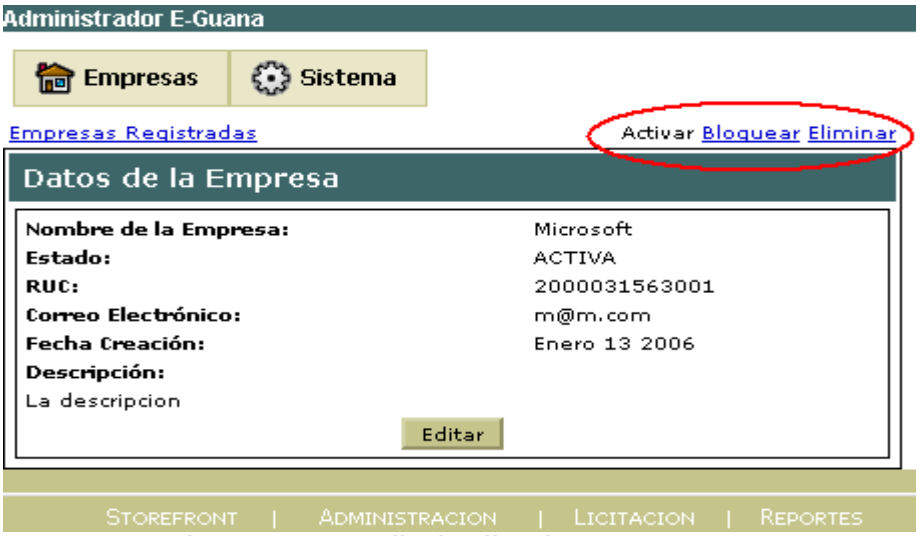


Figura 3-22 Pantalla detalles de una empresa

En esta página se destacan los enlaces que permiten la activación, bloqueo o eliminación de la empresa, es decir el cambio de estado de la empresa.

Modificación del Cupo de Usuarios que Administra una Empresa

La página para la solicitud del aumento en el cupo de usuarios es "/jsp/empresa/solicitudCupo.jsp":

Figura 3-23 Pantalla solicitud de cupo de usuarios

Para mostrar el listado de solicitudes, el "Administrador del Sistema" cuenta con la página "jsp/sistema/solicitudesCupo.jsp":

Figura 3-24 Pantalla solicitudes de cupo

Esta página permite aprobar o rechazar directamente la solicitud sin necesidad de abrirla, pero en caso de hacerlo el sistema mostrará la página "/jsp/empresa/solicitudCupo.jsp" de tal forma que se presenten los botones para la aprobación o negación de la solicitud:

Figura 3-25 Pantalla detalles de solicitud de cupo

Creación y Modificación de Unidades de Negocio

La página “jsp/empresa/unidades.jsp” permite la creación y modificación de las unidades de negocio de una empresa, de igual forma permite establecer las diferentes sucursales o unidades de negocio de una empresa así como la matriz de la misma.

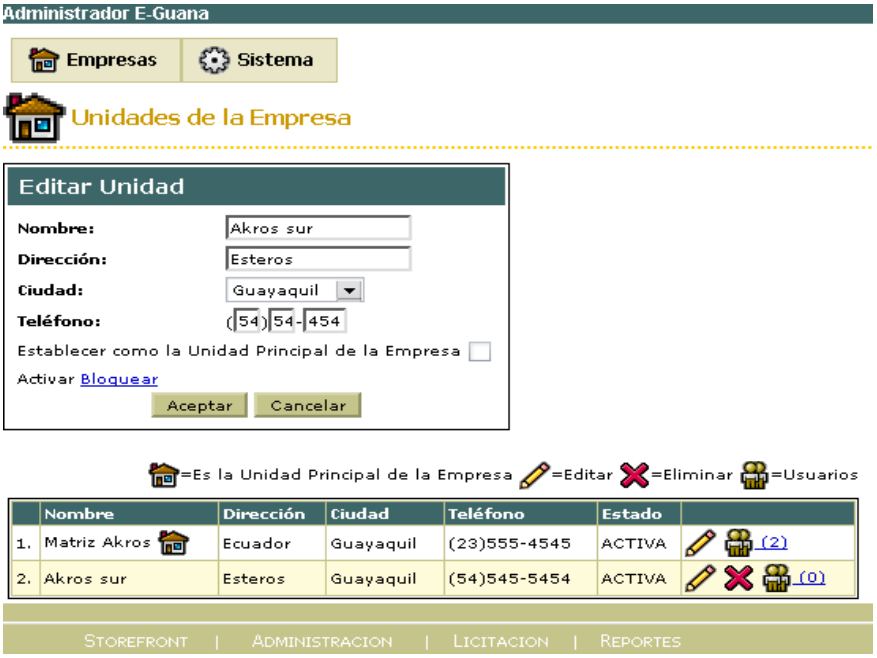


Figura 3-26 Pantalla unidades de empresa

Creación y Modificación de Usuarios dentro de una Empresa y Unidad de Negocio

La creación y modificación de usuarios dentro de una empresa, "jsp/empresa/usuarios.jsp" se presenta con la página:

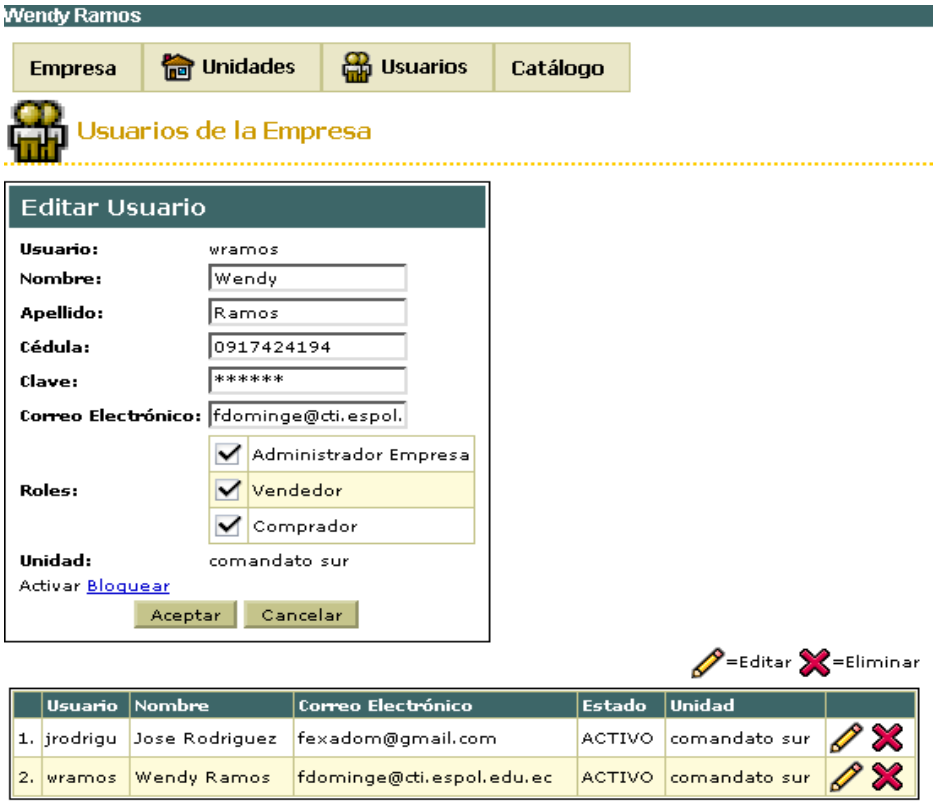


Figura 3-27 Pantalla usuarios del sistema

Para el mantenimiento de usuarios dentro de una unidad se tiene la página "jsp/empresa/usuariosPorUnidad.jsp":

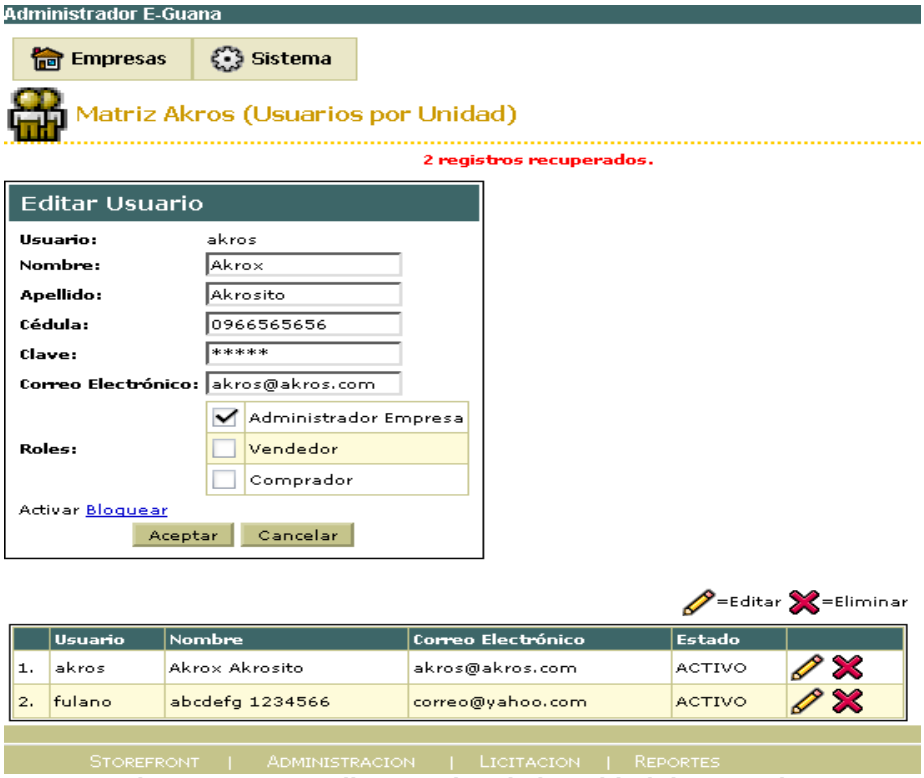


Figura 3-28 Pantalla usuarios de la unidad de negocio

La búsqueda de usuarios se puede hacer en las páginas previamente descritas, puesto que las mismas presentan un formulario. Para lograr esto su modo de presentación es el de lista:

Figura 3-29 Pantalla búsqueda de usuarios en unidad de negocio

Solicitud de Categorías de Productos

El “Administrador de la Empresa” es el usuario que realiza las solicitudes de nuevas categorías de productos, para esto se tiene la página “jsp/empresa/solicitudCategoria.jsp”:

Figura 3-30 Pantalla solicitud de categoría

Esta página presenta en la parte izquierda una estructura de árbol con las categorías de productos existentes en el sistema, en la parte central superior se muestra el formulario de la solicitud propiamente, mientras que en la parte inferior se listan las solicitudes que se han enviado, indicando el estado en el que se encuentra cada una.

El árbol de categorías permite la selección de la categoría padre de la categoría que se está solicitando.

Aprobación de Categorías de Productos

Las solicitudes de categorías pueden ser atendidas por el “Administrador del Sistema” con la página “jsp/sistema/solicitudesCategoria.jsp”:

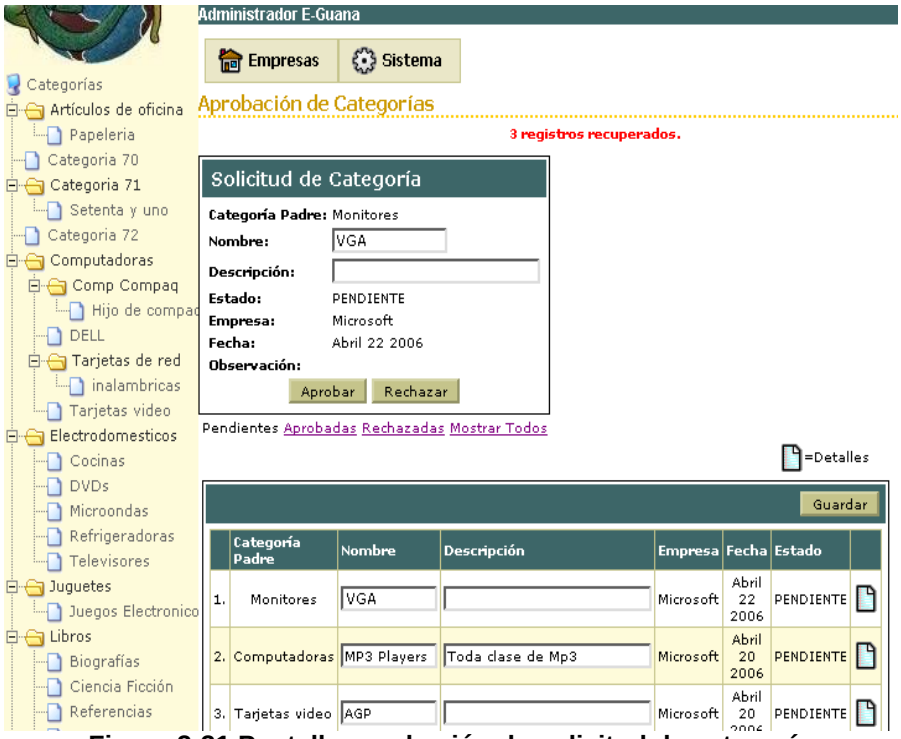


Figura 3-31 Pantalla aprobación de solicitud de categoría

El “Administrador del Sistema” puede aprobar o rechazar la solicitud, además si lo cree necesario, puede modificar los datos de la categoría previo a su aprobación.

Creación de Categorías de Productos

La página de para la creación de categorías puede ser accedida únicamente por el “Administrador del Sistema”, su ubicación es “jsp/sistema/categoria.jsp”:

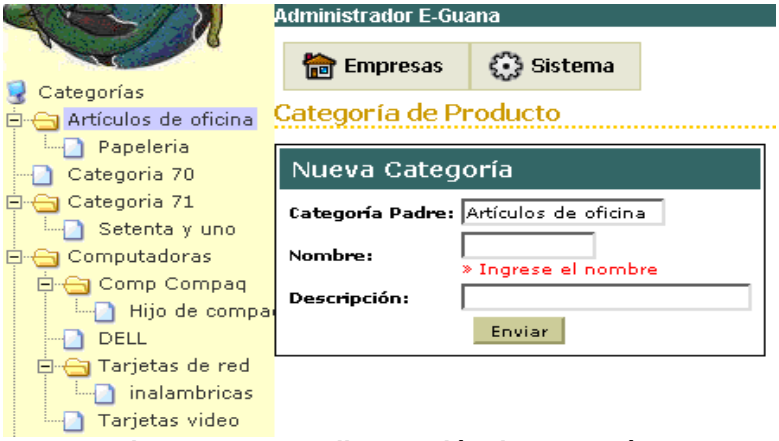


Figura 3-32 Pantalla creación de categorías

Similar a la página para la solicitud de la categoría, aquí se muestra el árbol de categorías para que el “Administrador del Sistema” pueda seleccionar la categoría a la que desea pertenezca la nueva que se está creando.

Modificación de Categorías de Productos

La modificación de las categorías de los productos la realiza el “Administrador del Sistema” a través del JSP “jsp/sistema/categorias.jsp”:

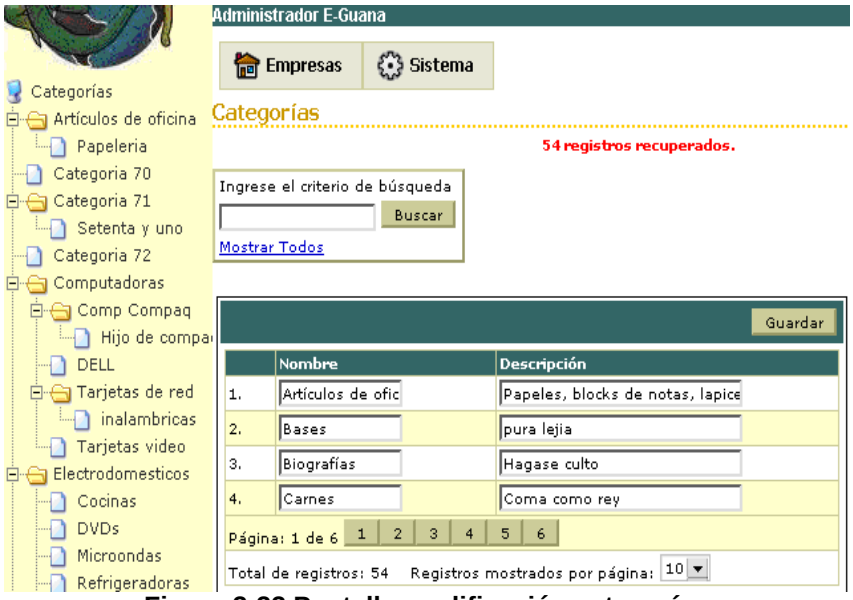


Figura 3-33 Pantalla modificación categorías

Previo a la modificación, el “Administrador del Sistema” puede buscar la categoría a modificar ya sea haciendo clic en el nombre de la categoría en el árbol o ingresando parte del nombre en el campo de texto junto al botón etiquetado como “Buscar”.

Ingreso de Productos

El ingreso de productos lo puede hacer tanto el “Administrador del Sistema” como el “Administrador de la empresa”, con la diferencia que el segundo lo puede hacer sólo a través del ingreso de una nueva entrada en su catálogo, en otras palabras el “Administrador de la Empresa” no puede crear un producto que no tenga algún ítem en su catálogo (Para aclarar esto léanse las definiciones de las clases Producto y Catálogo).

Considerando lo anterior, el “Administrador del Sistema” utiliza la página “jsp/sistema/producto.jsp” para el ingreso de productos:

Figura 3-34 Pantalla inicial creación productos

Luego de presionar el botón etiquetado como “Agregar Nuevo Producto”, la página luce de la siguiente manera:

Figura 3-35 Pantalla ingreso de producto

El “Administrador de la Empresa” para tener acceso a esta funcionalidad utiliza la página “jsp/empresa/agregarEditarCatalogo.jsp”:

Figura 3-36 Pantalla inicial ingreso producto a catálogo

Puesto que esta página de hecho le permite la modificación del catálogo de la empresa, en el listado de productos se muestra el precio y la cantidad de ítems existentes en el catálogo para cada producto, además si no tiene el producto en su catálogo, tiene la opción de agregar un nuevo ítem a su catálogo haciendo clic en la imagen identificada con “Agregar producto al catálogo”. Si va a agregar un producto nuevo debe presionar el botón “Agregar Nuevo Producto”, esto hará que la página luzca de la siguiente forma:

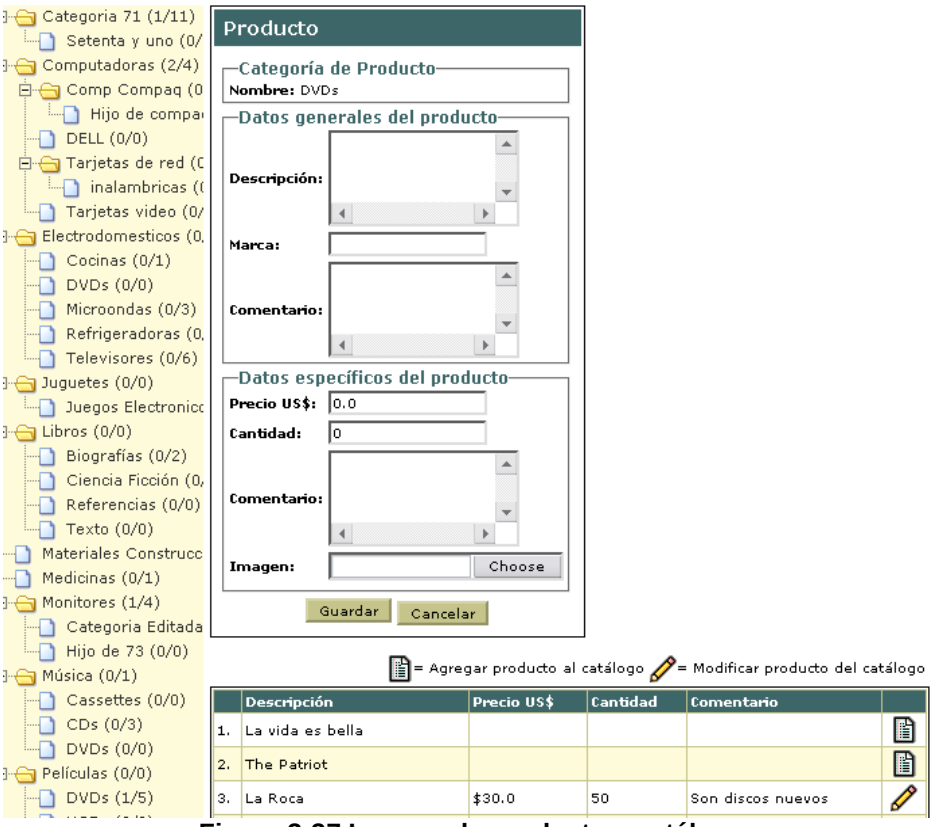


Figura 3-37 Ingreso de producto a catálogo

A diferencia del formulario que se le presenta al “Administrador del Sistema”, éste permite el ingreso de los datos para el catálogo (precio, cantidad, etc.) incluyendo una imagen.

Asignación de Permisos a un Rol

La asignación de permisos a un rol la realiza el “Administrador del Sistema” a través de la página: “jsp/sistema/permisosRol.jsp”:

Figura 3-38 Pantalla asignación de permisos a rol

En la parte superior se muestra un componente para la selección del rol al que se le asignarán los permisos, los mismos que se mostrarán en la lista inferior cada uno junto a un campo de selección que permitirá indicar si el permiso pertenece o no al rol.

Mantenimiento general

Existen algunas páginas que permiten el mantenimiento general de entidades, el funcionamiento de las mismas es similar entre sí por lo que no es necesario hacer la presentación de cada una de ellas, basta con una muestra.

En este caso se mostrará el mantenimiento de las ciudades a las que se puede hacer referencia en el sistema.

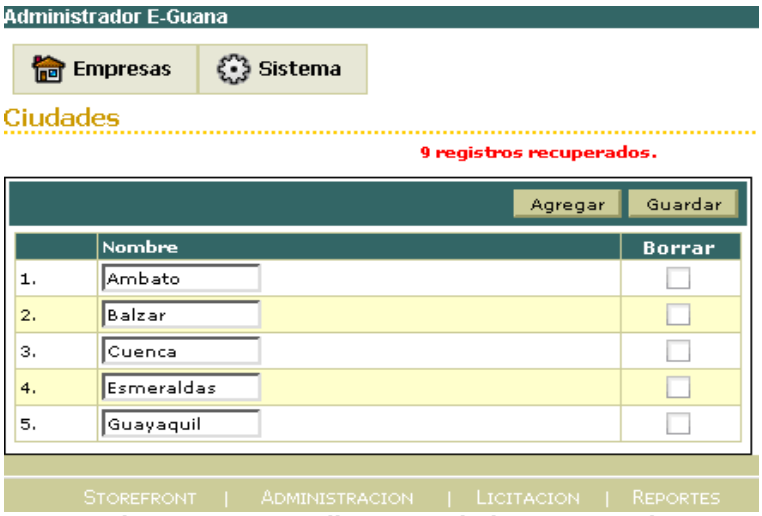


Figura 3-39 Pantalla mantenimiento general

Este tipo de páginas muestran el listado de los datos de todos los elementos de la entidad a la que se le desea dar mantenimiento, adicional a esto se presenta un campo para seleccionar el elemento si se lo desea borrar.

En el encabezado de la lista se muestra un botón para agregar un nuevo elemento a la lista y otro para guardar los cambios realizados. El botón para agregar sólo está presente en casos en los que se permite agregar nuevos elementos.

El controlador para estas páginas es una instancia de *org.eguana.administracion.web.ListaControl* o una extensión de la misma. Esta clase se encarga del control de la actualización, borrado y carga de los

elementos. La clase que corresponde al modelo para hacer este mantenimiento es *org.eguana.administracion.web.ModeloBase*, ya que derivaciones de ésta son las que conocen el *bean* de sesión para acceder a la entidad así como el tipo de DTOs a utilizar en la comunicación.

3.4.1.3. Etiquetas JSP personalizadas

En esta sección se describirán las etiquetas JSP personalizadas implementadas en este módulo.

Las etiquetas personalizadas son “elementos JSP definidos por el usuario que encapsulan tareas repetitivas. Las etiquetas personalizadas se distribuyen en una librería de etiquetas (*tag library*), que define un conjunto de etiquetas personalizadas relacionadas y contiene los objetos que implementan dichas etiquetas” [13], por lo tanto esta implementación dará como resultado una librería de etiquetas y las clases respectivas para ser utilizadas en páginas JSP.

La sintaxis de una etiqueta personalizada es [13]:

```
<prefjo:etiqueta atributo1=”valor” … atributoN=”valor” />
```

Ó

```
<prefijo:etiqueta atributo1=”valor” … atributoN=”valor”>
        Cuerpo
</prefijo:etiqueta>
```

Como se puede ver el código es similar a cualquier etiqueta usada en HTML, aunque se debe considerar que "*prefijo* distingue las etiquetas de una librería, *etiqueta* es el identificador de la etiqueta y *atributo1… atributoN* son atributos que modifican el comportamiento de la etiqueta" [13].

Sin embargo para poder usar las etiquetas no es suficiente saber su sintaxis ya que se debe [13]:

- Declarar la librería que contiene la etiqueta
- Hacer que la implementación de la librería este disponible para la aplicación Web.

A continuación se describirá con detalle cada una de estas tareas.

Declaración de Librerías de Etiquetas

Para declarar que una página JSP va a utilizar las etiquetas definidas en una librería de etiquetas, se debe "incluir la directiva *taglib* previo a la utilización de cualquiera de dichas etiquetas" [13].

El uso de dicha directiva es de la siguiente manera [13]:

```
<%@ taglib prefix="prefijo" uri=URI %>
```

Donde "el atributo *prefijo* define el prefijo que distingue a las etiquetas definidos por una librería dada de aquellos provistos por otras librerías" [13],

el prefijo usado en las páginas fue “eguana” para dejar en claro el origen de la etiqueta y evitar confusión, en tanto que “el atributo *uri* se refiere a un URI que identifica de forma única al descriptor de la librería de etiquetas (TLD), un documento que describe la librería de etiquetas” [13].

En cuanto al nombre del archivo de la librería de etiquetas éste “debe tener la extensión *.tld”* [13], además “los archivos TLD son almacenados en el directorio WEB-INF del archivo WAR o en el directorio META-INF de una librería de etiquetas empaquetada en un JAR” [13], en el caso de E-Guana se utilizó la primera alternativa de almacenamiento ya que las etiquetas sólo serán utilizadas dentro de la aplicación. Se puede referenciar la TLD directa o indirectamente.

Adicionalmente se debe considerar que “para usar la referencia al TLD es necesario utilizar el elemento *taglib* del archivo *web.xml”* [13], de la siguiente forma:

```
<web-app>
…
  <taglib>
    <taglib-uri>/tlt</taglib-uri>
    <taglib-location>/WEB-INF/tlds/tlt.tld</taglib-location>
  </taglib>
…
</web-app>
```

En el caso de este módulo de E-Guana, se declararon en el descriptor *web.xml* tanto la librería de etiquetas de las etiquetas propias del módulo así como la librería de etiquetas del *framework* Sofia:

```
…
<taglib>
   <taglib-uri>eguana.tld</taglib-uri>
   <taglib-location>tlds/administracion.tld</taglib-location>
 </taglib>
 <taglib>
   <taglib-uri>salmon.tld</taglib-uri>
   <taglib-location>tlds/salmon.tld</taglib-location>
 </taglib>
…
```

Por lo tanto en las páginas la directiva taglib lucirá de la siguiente forma:

```
<%@ taglib prefix=”eguana” uri=”eguana.tld” %>
```

Los descriptores TLD se ubicaron en la carpeta “tlds” del directorio de la aplicación Web.

Incluyendo la Implementación de la Librería de Etiquetas

La segunda tarea para poder usar las etiquetas en las páginas es, como ya se indicó, hacer que la implementación de la librería de etiquetas esté disponible para la aplicación Web, para lo que se tienen dos alternativas “aquellas empacadas en un archivo JAR se incluyen en el directorio /WEB-INF/lib/ ó un servidor de aplicaciones que puede cargar una librería de etiquetas en todas las aplicaciones Web que se ejecutan en el servidor” [13].

Para el módulo de administración de E-Guana las clases manejadoras de etiquetas propias del módulo se decidió colocar en el directorio /WEB-INF/classes/, es decir junto con todas las clases de la aplicación, mientras que el JAR con las clases de las etiquetas de Sofia se colocaron en el directorio /lib/ del servidor, así se asegura que las distintas aplicaciones Web que constituyen E-Guana utilizan la misma versión de clases evitando también conflictos, que según nuestra experiencia, es dependiente del servidor de aplicaciones en el que se despliega la aplicación.

La clase de soporte de etiquetas org.eguana.web.tags.TagBase

Esta clase de soporte es el padre de todas las clases que manejarán etiquetas personalizadas dentro del módulo. Extiende *javax.servlet.jsp.tagext.TagSupport* y provee métodos útiles para el acceso a los parámetros del requerimiento, atributos de la sesión, localización, etc.

Las etiquetas

En el módulo de Administración de E-Guana se implementaron algunas etiquetas personalizadas, unas con funcionalidad propia del módulo y otras destinadas a dar soporte en la presentación que no se podía conseguir con las etiquetas provistas por el *framework*. En esta sección se destacarán las primeras, debido a su mayor importancia dentro de la aplicación.

Etiqueta Empresa

La etiqueta empresa no acepta contenido dentro de la página, no acepta atributos y no tiene representación visual. Su objetivo es almacenar ya sea en la sesión o en el requerimiento, los datos de una empresa y una de sus unidades de negocio, para que luego otras etiquetas dentro de la página o clases de control puedan acceder a los mismos.

Cuando el usuario que visita la página es “Administrador de la Empresa” se recuperan los datos de la empresa y unidad de negocio a la que pertenece y se utiliza la sesión para conservar los datos ya que dicho valor no cambia a lo largo de la sesión, pero si se trata del “Administrador del Sistema” se los almacena en el requerimiento asociado a la página ya que este rol tiene la facultad de desplazarse entre páginas de distintas empresas. En este último caso se necesita que el URL contenga el parámetro “empresa” con el valor del identificador de una empresa válida, y puesto que se recuperan tanto los datos de la empresa como de la unidad de negocio, en esta situación se recuperan también los datos de la matriz de la empresa.

Adicional a lo descrito anteriormente, si la etiqueta empresa determina que el usuario no se ha autenticado, lo redirecciona a la página de acceso no permitido y si no se puede determinar cual es la empresa a la que pertenece el usuario o la empresa en la que desea ubicarse el “Administrador del Sistema”, se redirecciona a la página de error interno del servidor.

La clase manejadora de esta etiqueta es: *org.eguana.web.tags.EmpresaTag*.

Toda página cuyo funcionamiento deba estar en el contexto de una empresa debe hacer uso de esta etiqueta de la siguiente manera:

```
<%@ taglib prefix="eguana" uri="eguana.tld" %>
…
<eguana:empresa />
…
```

Si dentro de la página se desea tener acceso a los datos de la empresa se puede proceder de la siguiente manera con etiquetas JSTL:

```
<c:out value="${org_eguana_web_sesion_Empresa}" />
```

Ó en Sofia con los controladores que extienden de *org.eguana.administracion.web.empresas.*

PaginaEmpresaControl:

La variable protegida *empresa* contiene los datos de la empresa.

Etiqueta Usuario

La etiqueta personalizada usuario es manejada por la clase *org.eguana.web.tags.UsuarioTag*. Su función es cargar en la sesión los datos del usuario, incluidos los roles que tiene dentro del sistema. Acepta el atributo opcional “invisible”, cuyo valor debe ser *true, t, yes* ó *y* para que no

se muestren los datos del usuario en la página, esto es útil cuando sólo se desea tener disponibles los datos del usuario en el ambiente de la página para ser usados por otras etiquetas.

La declaración de esta etiqueta en una página JSP es como se muestra a continuación:

```
<%@ taglib prefix="eguana" uri="eguana.tld" %>
 …
<eguana:usuario invisible=”true” />
```

Para acceder a los datos del usuario se puede proceder en páginas con JSTL[5] así:

```
<c:out value="${org_eguana_web_sesion_Usuario.nombre}" />
```

En este ejemplo el atributo que se está mostrando es el nombre del usuario conectado.

Etiqueta Categorías

Esta etiqueta tiene la función de representar gráficamente las categorías de productos en forma de una estructura de árbol, donde cada categoría corresponde a una “hoja” si no tiene subcategorías o un “nodo expandible” si tiene una o más categorías hijas (similar a la representación de los directorios en la mayoría de sistemas de archivos gráficos).

La siguiente figura muestra el resultado de usar esta etiqueta en una página JSP:

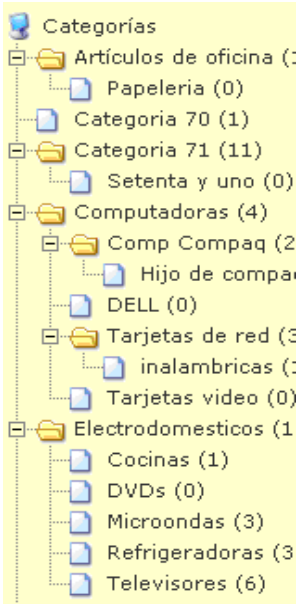


Figura 3-40 Muestra de árbol de categorías

Los atributos de esta etiqueta son “href”, “todas” y “contarProductos”, todos son atributos opcionales.

El atributo “href” corresponde al URL o función *Javascript* que se asignará al enlace mostrado en el árbol para cada categoría presente, si se desea que el enlace contenga el identificador de la categoría, es necesario incluir la cadena “<>” en el valor del atributo.

El atributo “todas” permite indicar que se recuperen y muestren todas las categorías existentes sin importar su estado, para esto se debe enviar un

---

[5] JSTL: Librería de etiquetas JSP que encapsula funcionalidad básica.

valor igual a *true, t, yes* ó *y*, caso contrario se recuperarán las categorías cuyo estado es “Activo”.

El atributo “contar” si tiene un valor igual a *true, t, yes* ó *y* indicará que se muestre junto al nombre de la categoría el número de productos existentes en cada categoría.

La clase manejadora de esta etiqueta es la clase *org.eguana.web.tags.CategoriasTag*.

A continuación se muestra un código de ejemplo de uso de esta etiqueta:

```
<%@ taglib prefix="eguana" uri="eguana.tld" %>
…
<eguana:categorias href="javascript:nombreFuncion(<>)" />
…
```

Ó cuando se desea que se indique la cuenta de productos por categoría:

```
<%@ taglib prefix="eguana" uri="eguana.tld" %>
 …
<eguana:categorias
href="nombrePagina.jsp?categoria=(<>)" contar=”true” />
…
```

Notar en los 2 ejemplos que se utilizó “<>” para que los enlaces en cada categoría tengan el identificador de la categoría de producto.

Etiqueta Permiso

Esta etiqueta tiene la función de determinar si el usuario posee alguno de los permisos indicados en el atributo “nombre” (se debe separar el nombre de cada permiso con una coma), para, en el caso de tenerlos, mostrar el contenido o cuerpo de la etiqueta.

Adicional a lo anterior, si el valor del atributo opcional “redirecciona” toma alguno de los siguientes valores: *true, t, yes* ó *y*, y el usuario no tiene ninguno de los permisos, se lo redirecciona a la página de acceso no permitido.

La clase manejadora de esta etiqueta es *org.eguana.web.tags.PermisoTag*.

El código siguiente muestra la forma de utilizar la etiqueta permiso:

```
<%@ taglib prefix="eguana" uri="eguana.tld" %>
 …
<eguana:permiso
nombre="permiso1, permiso2" redirecciona=”false”>
<!-- Cuerpo de la etiqueta -->
…
</eguana:permiso>
…
```

El ejemplo mostrará el cuerpo de la etiqueta, que puede ser cualquier código JSP, si es que el usuario tiene permiso1 ó permiso2.

Etiqueta Imágenes del Catálogo

Esta etiqueta permite la presentación de las imágenes asociadas a una entrada en el catálogo de una empresa. Debido a que esta etiqueta es utilizada en páginas fuera de este módulo y por lo tanto en páginas que

posiblemente no utilicen el *framework* Sofia, se tienen dos implementaciones de esta etiqueta.

La implementación que funciona dentro de páginas con etiquetas Sofia es manejada por la clase *org.eguana.web.tags.ImagenesCatalogoSofiaTag* mientras que la implementación independiente es manejada por *org.eguana.web.tags.ImagenesCatalogoTag*. Debido a que el mantenimiento de las imágenes de un catálogo se hace en una página que usa Sofia, la implementación para ésta contiene una lógica más compleja que permite el borrado de las imágenes a través del formulario.

La implementación de Sofia requiere dos atributos “name” y “datasource”, el primero es el nombre del componente y el segundo corresponde al modelo de datos que presenta los datos en la página.

Las clases anteriores necesitan de dos clases adicionales para completar el funcionamiento, estas son *org.eguana.web.tags.ImagenesCatalogo* y *org.eguana.web.servlets.DescargarImagenCatalogo*. La primera establece la ubicación física de los archivos de las imágenes asociadas a una entrada en el catálogo, mientras que la otra es un servlet que se encarga de la descarga del archivo al cliente.

El uso de esta etiqueta en páginas sin etiquetas de Sofia se muestra a continuación:

```
<%@ taglib uri="http://java.sun.com/jstl/core" prefix="c" %>
<%@ taglib uri="/eguana" prefix="eguana" %>
<c:set scope="request" var="idCatalogo" value="[id válido del catálogo]" />
<eguana:imagenesCatalogo />
…
```

Mientras que el código para páginas de Sofia es:

```
<%@ taglib uri="/eguana" prefix="eguana" %>
<eguana:sofia-imagenesCatalogo
name="imagenes" datasource="catalogoProducto:idCatalogo" />
…
```

El resto de etiquetas como se indicó previamente comprenden funcionalidades de soporte para presentación, a continuación se listan dichas etiquetas:

Etiqueta	Clase manejadora (Paquete org.eguana.web.tags)	Funcionalidad
sofia-texto	TextoL10NTextTag	Presentar texto localizado en páginas de Sofia
jstl-texto	TextoL10NJSTLTag	Presentar texto localizado en páginas que hagan uso de JSTL con etiquetas de iteración que recuperen propiedades de los objetos
texto	TextoL10NTag	Presentar texto localizado general
par	ParTag	En lazos con etiquetas JSTL, muestra el contenido si corresponde a una iteración par
impar	ImparTag	En lazos con etiquetas JSTL, muestra el contenido si corresponde a una iteración impar
sofia-fila	NumeroFilaSofiaTag	Muestra el número de fila para tablas de datos con Sofia
fila	NumeroFilaTag	Muestra el número de fila

		para iteraciones que usan JSTL
telefono	TelefonoTag	Extiende un componente de Sofia para presentar un número telefónico
validador	ValidadorTag	Extiende un componente de validación de Sofia que permite el manejo de errores de forma personalizada
displaybox	DisplayBoxL10NTag	Extiende un componente de presentación de Sofia para permitir la localización

Tabla 3-4 Etiquetas adicionales implementadas en el módulo de Administración

3.5. Configuración del Entorno de Ejecución

E-Guana es una aplicación multicapa J2EE, por lo que lógicamente la puesta en marcha de la misma implica algún grado de configuración en cada una de las capas que la conforman.

La capa del sistema de información de la empresa es la base de datos, en E-Guana la base de datos utilizada es MySQL versión 3.23.54.

Una capa más arriba se encuentra la capa del negocio, en esta aplicación J2EE dicha capa corresponde a los EJBs, los mismos que residen en el contenedor EJB que es JBoss versión 3.2.4.

Sobre la capa anterior funciona la capa Web, implementada en E-Guana con páginas JSP, estas páginas se ejecutan en el contenedor Web, correspondiente a Tomcat 5.0 embebido en JBoss.

La última capa, la del cliente, no requiere mayor configuración, tan solo que cumpla con los requerimientos mínimos; un navegador Web de última generación como Internet Explorer versión 6, Firefox versión 1.5 u Opera versión 8, en el caso de que sea un cliente Web. También existe la opción de utilizar una aplicación de escritorio que requiere la máquina virtual de Java 1.4 o superior (Descrita en el Anexo 2).

Se iniciará con la configuración de JBoss, el contenedor EJB, luego se procederá con el contenedor Web, Tomcat embebido en JBoss, y se culminará con la configuración del motor de base de datos, MySQL.

3.5.1. Contenedor EJB

En esta sección se describirá la configuración del contenedor EJB, Jboss, para que funcione correctamente E-Guana.

Fuente de datos

Según el Tutorial de Java “una fuente de datos es una fábrica de conexiones a la fuente de datos físicos que este objeto representa” [13], en el caso del sistema la fuente de datos permitirá obtener las conexiones a una base de datos (MySql).

Afortunadamente JBoss viene con un sinnúmero de configuraciones de ejemplo para distintas bases de datos (en la versión 3.2.4 se las puede

encontrar en docs/examples/jca), el archivo de configuración para la fuente de datos en MySql es *mysql-ds.xml*, y su contenido, luego de adaptarlo a E-Guana, es:

```
<datasources>
  <local-tx-datasource>
    <jndi-name>MySqlDS</jndi-name>
    <connection-url>jdbc:mysql://localhost:3306/db_eguana</connection-url>
    <driver-class>com.mysql.jdbc.Driver</driver-class>
    <user-name>root</user-name>
    <password></password>
  </local-tx-datasource>
</datasources>
```

A continuación se describirá el significado de este archivo según lo que mostrado en el código:

"El elemento *jndi-name* corresponde al nombre JNDI[6] de esta fuente de datos" [15], es decir el nombre que se debe utilizar para encontrar a este objeto a través de JNDI.

"El elemento *connection-url* es el URL de la base de datos" [15], en el caso de E-Guana es jdbc:mysql://localhost:3306/db_eguana.

"El elemento *driver-class* es el nombre de la clase que implementa la interfaz java.sql.Driver específico para la base de datos en cuestión" [15], el sistema utiliza el *driver* com.mysql.jdbc.Driver.

---

[6] JNDI: Interfaz de Nombramiento y Directorio de Java.

“El elemento *user-name* corresponde al nombre de un usuario que tenga los permisos de lectura y escritura en la base de datos indicada en *connection – url”* [15], en el caso de E-Guana el usuario con los permisos adecuados es “root”.

Finalmente, “el elemento *password* indica la clave (de ser necesario) asociada al usuario indicado previamente, para acceder a la base de datos” [15], en el código anterior se envía la cadena de texto vacía.

La clase *driver* de la base de datos no es distribuida junto con el servidor de aplicaciones, por lo que es necesario asegurarse de que al iniciar el servidor, éste sea capaz de encontrarla. Para esto se colocó el JAR que contiene el *driver* en la carpeta “lib” del servidor, en el caso de E-Guana el JAR es mysql-connector-java-2.0.14-bin.jar.

Una vez que se tiene listo el archivo de configuración de la fuente de datos y que se ha colocado el JAR en el lugar correcto, “se debe colocar el archivo de configuración de la fuente de datos en la carpeta “deploy” del servidor” [15], esta simple colocación del archivo en dicha carpeta inicia el proceso de despliegue.

Con esto la fuente de datos está lista para ser utilizada por cualquier aplicación desplegada en el servidor.

Toda aplicación J2EE que desee utilizar esta fuente de datos deberá buscarlo a través del nombre JNDI java:/MySqlDS. Sin embargo en el caso de JBoss es necesario “indicar el nombre JNDI de la fuente de datos en el archivo *jbosscmp-jdbc.xml* de cada aplicación a través de la etiqueta *datasource”* [15], esto en la aplicación resulta en algo similar a lo siguiente:

```
<jbosscmp-jdbc>
   <defaults>
     <datasource>java:/MySqlDS</datasource>
     <datasource-mapping>mySQL</datasource-mapping>
   </defaults>
…..
</jbosscmp-jdbc>
```

Se debe notar que el valor de la etiqueta *datasource* es el valor de la etiqueta *jndi-name* del archivo *mysql-ds.xml* con el prefijo “java:/”.

Seguridad

Como se ha indicado en secciones anteriores, el sistema requiere un esquema de seguridad, al respecto el Tutorial J2EE indica ”cuando una aplicación J2EE requiere hacer uso del Servicio de Autorización y Autenticación de Java (JAAS en Inglés), es decir, control de accesos dentro de la aplicación, es necesario configurar el módulo de *login* a utilizarse en la autenticación” [13]. En el servidor de aplicaciones dicha configuración se hizo en el archivo *conf/login-config.xml*, al agregar una nueva entrada de la siguiente forma:

```
<application-policy name = "eguana">
      <authentication>
        <login-module
code = "org.jboss.security.auth.spi.DatabaseServerLoginModule"
flag = "required">
          <module-option
name = "unauthenticatedIdentity">visitante</module-option>
                    <module-option name="dsJndiName">java:/MySqlDS</module-
option>
                    <module-option name="principalsQuery">
                              SELECT clave
                              FROM usuarios
                              WHERE login =? AND estado = 'a'
                    </module-option>
                    <module-option name="rolesQuery">
                              SELECT nombre, 'Roles'
                              FROM roles
                              INNER JOIN usuario_rol
ON(roles.id_rol = usuario_rol.id_rol)
                              INNER JOIN usuarios
ON(usuarios.login = usuario_rol.login)
                              WHERE usuarios.login =?
                    </module-option>
        </login-module>
      </authentication>
 </application-policy>
```

En el caso de JBoss “cada aplicación tiene una política de aplicación (application-policy) en este archivo” [15], lo más importante a destacar de la configuración anterior es lo siguiente:

El atributo “name” de “application-policy”, puesto que este valor es el que deberá ser luego referenciado por las aplicaciones para poder hacer uso de este módulo de autenticación.

El primer elemento “login-module” tiene como valor (org.jboss.security.auth.spi.DatabaseServerLoginModule) el nombre de “un módulo de *login* que recupera la información necesaria para la autenticación

de una base de datos” [15], en E-Guana será la misma base de datos pero es posible que los datos de autenticación estén en otra base de datos e inclusive en un distinto tipo de almacén de datos, lo importante a destacar de esto es que solo se debe modificar este archivo y dejar intacta la aplicación.

“El elemento *module-option* con atributo *name* igual a *dsJndiName* indica la fuente da datos a utilizar para recuperar los datos del usuario y sus roles” [15], en este caso es java:/MySqlDS, es decir la fuente de datos configurada previamente.

Luego están las consultas SQL, “la primera debe retornar la clave del usuario que se desea autenticar” [15], considerando que recibe como parámetro el nombre de usuario, si esta consulta no es exitosa el usuario no podrá utilizar el sistema. La segunda consulta corresponde a “los nombres de roles que el usuario tiene en la aplicación junto con el texto ‘Roles’” [15], de igual forma recibe como parámetro el nombre de usuario.

Una vez que está configurado el módulo de aplicación JBoss indica que “es necesario decirle a la aplicación J2EE el nombre de la política de aplicación que va a utilizar, a través del archivo *jboss.xml* de la aplicación” [15], en la aplicación aquello se lo indica de la siguiente forma:

```
<jboss>
  <security-domain>java:/jaas/eguana</security-domain>
  <unauthenticated-principal>visitante</unauthenticated-principal>
 …
```

```
</jboss>
```

“El elemento *security-domain* indica el nombre de la política de aplicación con el prefijo java:/jaas/ en tanto que elemento *unauthenticated-principal* indica el usuario a utilizar cuando el usuario no está autenticado” [15].

Lo anterior corresponde a las aplicaciones J2EE, de igual forma JBoss requiere que “las aplicaciones *Web* deben indicar el valor de la política de aplicación a utilizar, esto se lo hace en el archivo *jboss-web.xml* de la aplicación” [15], de manera similar que en el archivo *jboss.xml* descrito previamente, de la forma:

```
<jboss-web>
        <security-domain>java:/jaas/eguana</security-domain>
        <context-root>administracion</context-root>
</jboss-web>
```

Vale indicar que gracias al uso de Xdoclet y Ant, no es necesaria la creación manual de ninguno de los descriptores de despliegue (*ejb-jar.xml, jboss.xml* y *jbosscmp-jdbc.xml*), sino que tan solo a través de etiquetas Xdoclet, utilizadas en las clases de los *beans,* archivos de propiedades y la ejecución de un proceso de Ant, se generan dichos descriptores automáticamente, pero es importante conocer cuales son los archivos y atributos a configurar para las situaciones en las que no se cuente con estas herramientas.

Empaquetamiento y despliegue de la aplicación J2EE

Una vez que se han configurado correctamente todos los archivos, es posible hacer el despliegue de la aplicación en el contenedor J2EE, es decir “el proceso de instalación de los componentes en el contenedor para poder ser utilizados” [13], sin embargo previo a esto es necesario empaquetar la aplicación para su distribución.

Más formalmente, “una aplicación J2EE se distribuye en un archivo empresarial EAR (Enterprise Archive), que es básicamente un archivo Java JAR (Java Archive con extensión .ear). Este archivo contendrá todos los módulos (EJB y Web), recursos y el descriptor de despliegue que constituyen la aplicación. El descriptor de despliegue es un documento XML que describe la configuración necesaria de una aplicación, módulo o componente” [14], no obstante lo anterior, para facilitar el desarrollo de E-Guana y la independencia entre los módulos, se decidió que el despliegue de los módulos Web (aplicaciones Web) y el módulo EJB se lo haría por separado. Resultado de esto, cada aplicación Web se desplegará en su propio archivo Web WAR (Web Archive) y el único módulo EJB, de la misma forma, se desplegará en su propio archivo JAR.

En JBoss, el proceso de despliegue consiste simplemente en la ubicación del archivo EAR, JAR o WAR en la carpeta “deploy” del servidor.

El único módulo EJB utilizado por la aplicación E-Guana propiamente se desplegará en el archivo EJB JAR “eguana.jar”, cuya estructura es similar a lo siguiente:

```
/META-INF/MANIFEST.MF
/META-INF/ejb-jar.xml
/META-INF/jboss.xml
/META-INF/jbosscmp-jdbc.xml
/org/eguana/storefront/**/*.class
/org/eguana/licitacion/**/*.class
/org/eguana/administracion/**/*.class
/org/eguana/comun/**/*.class
/org/eguana/banco/**/*.class
```

De esta estructura se puede indicar que “el archivo ejb-jar.xml es el descriptor de despliegue definido por la especificación J2EE” [13] y como ya se ha visto con anterioridad se usó Xdoclet para su generación. Los archivos *jboss.xml* y *jbosscmp-jdbc.xml* son conocidos como “descriptores de despliegue de ejecución, que lógicamente permiten configurar parámetros específicos de la implementación J2EE” [15]. En resumen el primero configura datos requeridos por JBoss que no están definidos en *ejb-jar.xml* mientras que el último configura parámetros necesarios para la persistencia.

El resto del archivo EJB JAR lo conforman las clases que corresponden a los *beans* de sesión, de entidad y de mensajes, clases de soporte, etc., agrupadas en una carpeta cuyo nombre indica el módulo que las implementó o el contexto en que se utilizan, así por ejemplo la subcarpeta *storefront* contiene las clases utilizadas por el módulo de Storefront, mientras que el

subdirectorio *comun* contiene clases utilizadas por todos los módulos de E-Guana.

3.5.2. Contenedor Web

La versión 3.2.4 de JBoss viene con la versión 5 del servidor de aplicaciones *Web* o contenedor *Web*, Tomcat.

Las librerías y archivos de configuración de este servidor se encuentran en la carpeta /deploy/jbossweb-tomcat50.sar de JBoss.

La configuración con la que viene la distribución de JBoss, aunque se comprobó que funcionaba inmediatamente una vez instalado el servidor, se evidenció que se queda un tanto limitada para los objetivos del proyecto. En esta parte se describirá la configuración necesaria para soportar sesiones HTTPS, así como la configuración que permite la distribución de los datos de autenticación del usuario a lo largo de las distintas aplicaciones *Web* que constituyen E-Guana.

Configuración para sesiones HTTPS

Esta configuración requiere modificaciones en el servidor y en las aplicaciones Web de E-Guana.

Primero se debe tener en consideración que para la creación de sesiones HTTPS “el servidor necesita tener acceso al par clave pública/clave privada en la forma de un certificado” [15], para la generación de este certificado se utiliza la herramienta *keytool* distribuída con el JDK.

El comando para la generación del certificado es:

```
keytool -genkey -alias tomcat -keyalg RSA -keystore server.keystore –dname
“CN=localhost, OU=FIEC, O=ESPOL, L=Guayaquil, ST=Guayas, C=ec"
```

Una vez ejecutado el comando se solicitará el ingreso de una contraseña para el almacén de claves y para la clave *tomcat* que se está creando.

Completada la operación anterior se tiene el archivo conocido como almacén de claves con el nombre “server.keystore”, que es una base de datos de claves de seguridad. El archivo correspondiente al almacén de datos se ubicará en la carpeta en la que fue ejecutado el comando *keytool*.

El siguiente paso es indicarle al contenedor *Web* que utilice SSL cuando le sea requerido y que utilice el archivo recién creado. Para esto es necesario modificar el archivo *server.xml* ubicado en la carpeta del contenedor (deploy/jbossweb-tomcat50.sar), habilitando el conector SSL/TLS:

```
<Connector
        port="8443"
        address="${jboss.bind.address}"
        maxThreads="100"
        minSpareThreads="5"
        maxSpareThreads="15"
```

```
scheme="https"
secure="true"
clientAuth="false"
keystoreFile="${jboss.server.home.dir}/conf/server.keystore"
keystorePass="la_clave"
sslProtocol = "TLS" />
```

Los atributos que se deben modificar son “keystoreFile” indicando la ruta adecuada para que el servidor encuentre el almacén de claves, así como “keystorePass” con el valor de la contraseña que se proveyó durante la generación de la clave. Si se desea modificar el puerto donde serán atendidas las peticiones HTTPS, el atributo “port” permite establecer esto. Recordar que el puerto HTTPS por defecto de la configuración no es el puerto estándar HTTPS (443), por lo cual, los navegadores mostrarán el número de puerto 8443 junto con el nombre del servidor.

Con el procedimiento anterior, el servidor está listo para atender los requerimientos HTTPS en el puerto 8443, ahora es necesario que cada aplicación Web le solicite al servidor que transporte la información de autenticación de una forma segura. Esto se lo dice en el archivo web.xml en la sección “security-constraint” de la siguiente forma:

```
<security-constraint>
<web-resource-collection>
<web-resource-name>solo-para-administrador-del-sistema</web-resource-name>
<url-pattern>/jsp/sistema/*</url-pattern>
</web-resource-collection>
<auth-constraint>
<role-name>Administrador</role-name>
</auth-constraint>
<user-data-constraint>
<transport-guarantee>CONFIDENTIAL</transport-guarantee>
</user-data-constraint>
```

```
</security-constraint>
```

El elemento a destacar aquí es “transport-guarantee” dentro de “user-data-constraint” ya que el valor de este elemento hace precisamente lo que se indicó previamente, cuando el mismo es CONFIDENTIAL o INTEGRAL.

Específicamente INTEGRAL significa que “la aplicación requiere que los datos enviados entre el cliente y el servidor se lo haga de una forma tal que no pueda ser modificada” [13], en tanto que CONFIDENTIAL indica que “la aplicación requiere que los datos sean transmitidos en una forma que prevenga que el contenido sea observado por otras entidades” [13], en definitiva estos dos valores harán que el servidor automáticamente inicie una comunicación sobre SSL.

Es importante indicar que si un servidor no está adecuadamente configurado para soportar HTTPS, las aplicaciones *Web* parecerán que no se encuentran desplegadas.

Configuración *Single Sign-On*

E-Guana está conformado por distintas aplicaciones *Web* de acuerdo al módulo del proyecto implementado. Por lo tanto cuando el usuario desea acceder a la funcionalidad de un módulo, utiliza una aplicación *Web* contra la cual debe autenticarse, y si desea la funcionalidad de otro módulo debe

dirigirse a otra aplicación *Web* donde también debe autenticarse, no obstante los datos de autenticación son los mismos en cada una de las aplicaciones.

Para evitar este proceso innecesario y repetitivo, es posible indicarle al servidor que la autenticación de usuarios debe pasar entre aplicaciones, es decir, el usuario se autentica cuando ingresa a alguna de las aplicaciones *Web* y sigue autenticado cuando se dirige a otra aplicación.

El archivo a modificar para indicar lo anterior al servidor es *server.xml* de la carpeta *tomcat*, por medio de la habilitación del elemento siguiente:

```
<Valve
className="org.apache.catalina.authenticator.SingleSignOn" debug="0"/>
```

Una vez que la aplicación Web está configurada se procederá al empaquetamiento y despliegue, lo cual se explicará en el siguiente punto.

Empaquetamiento y despliegue de la aplicación *Web*

Como se lo indicó en la sección 3.4.1, cada aplicación *Web* será desplegada independientemente una de otra. “El despliegue de un módulo Web o aplicación Web se hace en un archivo JAR con extensión .*war”* [13], recuérdese que un archivo JAR es tan solo un archivo comprimido con extensión JAR.

El despliegue en JBoss consiste en “la colocación del módulo Web en la carpeta *deploy* del servidor” [15], si el servidor está ejecutándose se podrá notar que automáticamente se inicia el proceso de despliegue.

La estructura del módulo Web de administración de E-Guana es similar a lo siguiente:

```
/WEB-INF/web.xml
/WEB-INF/jboss-web.xml
/WEB-INF/lib/*.jar
/WEB-INF/classes/**/*.class
/WEB-INF/tlds/*.tld
/images/*.gif  o *.jpg
/styles/*.css
/scripts/**/*.js
/jspf/**/*.jsp
/jsp/**/*.jsp
/index.html
```

Los archivos listados en esta estructura son:

“El archivo *web.xml* es el descriptor de despliegue de la aplicación Web” [13], adicional a lo anotado previamente, en este archivo se declararon entre otras cosas los servlets, las páginas a mostrar para algún error específico, las librerías de etiquetas personalizadas y variables de ambiente necesarias para la aplicación.

“El documento *jboss-web.xml* es el descriptor de despliegue específico de JBoss” [15], en el caso de E-Guana se lo utiliza para establecer el dominio de seguridad que se utilizará así como el nombre del contexto de la aplicación,

es decir el nombre a utilizarse en el navegador Web para acceder a la aplicación.

La subcarpeta “lib” contiene todas las librerías en formato JAR necesarias para la aplicación.

La subcarpeta “classes” contiene las clases de los servlets, etiquetas personalizadas y de cualquier clase implementada.

El subdirectorio “tlds” contiene todos los descriptores de librerías de etiquetas que se utilizan dentro de la aplicación.

Dentro de “images” se ubican todas las imágenes utilizadas en las páginas de la aplicación.

Las hojas de estilos para las páginas se encuentran en el subdirectorio “css”.

En el subdirectorio “scripts” se ubican los archivos de *Javascript* que permiten manejo de menús, estructuras de árboles, etc.

Las páginas JSP se encuentran tanto en la carpeta “jsp” como en la carpeta “jspf”, esta última agrupa porciones de páginas que son incluidas en otras, por ejemplo el encabezado de la página.

Finalmente la página inicial de la aplicación es “index.html”.

3.5.3. Motor de Base de Datos

La configuración del motor de base de datos comprende básicamente en tener operativo el servicio asociado, así como la creación de un usuario que tenga los permisos necesarios para el correcto funcionamiento de las aplicaciones.

En la plataforma Windows la instalación de MySQL crea automáticamente un servicio que se encargará de controlar la operación del motor de base de datos. El administrador del servidor debe asegurarse del correcto funcionamiento del servicio, pudiendo hacerlo a través de aplicaciones para la administración de bases de datos MySQL, o con la herramienta *winmysqladmin.exe* distribuida con el paquete de esta base de datos.

Los datos importantes que necesita tener en consideración para la correcta configuración de una fuente de datos para esta base de datos, son el nombre o dirección IP del servidor en el cual se encuentra funcionando la base de datos y el puerto en el cual atiende los requerimientos (cuyo valor por defecto es 3306).

A través de la herramienta *winmysqladmin.exe* es posible entre otras cosas la edición del archivo “my.ini”, que describe la configuración de MySQL. Es en este archivo que se puede verificar tanto el valor del puerto como la dirección del servidor en la que está atendiendo la base de datos. De no

tener acceso a esta herramienta, es posible encontrar y editar directamente el archivo “my.ini” ubicado generalmente en la carpeta WINDOWS del sistema operativo.

La fuente de datos del contenedor J2EE utiliza un usuario para conectarse con la base de datos. Para crear este usuario se utilizó la aplicación de consola *mysql.exe* ubicada en la carpeta *bin* del directorio donde se instaló MySQL, ejecutando el siguiente comando:

```
GRANT Select, Insert, Update, Delete, Index, Alter, Create, Drop, References ON
db_eguana.* TO 'jboss_usr'@'%' IDENTIFIED BY 'jboss_password'
```

Lo que está haciendo este comando es darle todos los permisos al usuario “jboss_usr” en la base de datos db_eguana, y asignándole la clave “'jboss_password”.

3.6. Interacción con los módulos del Sistema

A continuación se describe la forma en que interactúa el módulo de Administración con el resto de módulos del sistema E-Guana, se deberá notar que este módulo interactúa o provee funcionalidad requerida por cada uno de los módulos del sistema.

Previamente es importante destacar que la correcta interacción de los módulos requiere de la adecuada configuración de la seguridad en cada uno

de ellos (*Ver Sección 3.4 Configuración del Entorno de Ejecución*) así como del contenedor de EJB y del contenedor Web.

Interacción con el Módulo de Store Front

Entre las principales funciones que desempeña el módulo de Store Front están:

- Ingresar a la tienda (ingreso al sistema, usuario)
- Buscar en las perchas el/los producto que se desea comprar (navegación en el catálogo)

La validación del ingreso al sistema lo realiza directamente el módulo de Administración por medio de la información ingresada por el módulo de Store Front. El operador ingresa su usuario por medio de la pantalla y es el módulo de Administración quien posee toda la lógica de validación del usuario, por tipo, empresa, etc., y es quien permite el acceso o tipo de acceso concedido dependiendo del rol que posee el usuario (validación de seguridades).

Para que cualquier componente de la capa de presentación Web de este módulo tenga acceso a los datos y roles del usuario autenticado, se tiene una instancia de la clase *org.eguana.administracion.dto.UsuarioWebDTO*, provista por el módulo de Administración a través de la etiqueta JSP personalizada “usuario”.

Los datos de la empresa a la que pertenece el usuario son provistos por una instancia de la clase *org.eguana.administracion.dto.EmpresaDTO* a través de la etiqueta JSP “empresa”.

Luego del ingreso, el usuario, que en este caso desea comprar productos, realiza la navegación o búsquedas de los productos disponibles. Estos productos son proporcionados por catálogo. El módulo de Administración es quien valida qué catálogo mostrar por medio del Store Front dependiendo de ciertos parámetros como empresa, categoría de producto, etc. Esta interacción se da en la capa del negocio con los *beans* de entidad *CatalogoBean* y *ProductoBean* y los *beans* de sesión *AdminCatalogoBean* y *AdminProductoBean*.

Para la generación de la compra, el módulo de Store Front necesita relacionar a la misma con un usuario comprador y otro vendedor por lo que debe interactuar con el *bean* de entidad *org.eguana.administracion.bo.ejb.UsuarioBean*.

Interacción con el Módulo de Licitación y Subastas

El módulo de Licitación y Subastas interactúa con el módulo de Administración de forma similar al módulo de Store Front en cuanto al acceso del usuario y la navegación del catálogo, además a través del *bean* de sesión

AdminCatalogoBean, este módulo hace uso directo de la funcionalidad del ingreso de elementos a su catálogo cuando genera una oferta de licitación.

Interacción con el Módulo de Pagos de Transacciones

La interacción de estos dos módulos se da cuando el módulo de Administración requiere la verificación de la cuenta bancaria de la empresa, tanto en la etapa de registro como cuando el "Administrador del Sistema" desee hacer dicha solicitud.

De igual forma es necesaria la comunicación entre ambos cuando el módulo de Pagos de Transacciones genera listas de solicitudes de relacionadas a las compras o a las verificaciones de cuentas, ya que se requieren los datos de la empresa asociada a cada proceso y los mismos son manejados por el módulo de Administración.

Interacción con el Módulo de Reportes

Con la excepción del acceso al catálogo de productos, este módulo interactúa de manera similar a los anteriores. Esto se debe a que la mayor parte de este módulo requiere de *frameworks* que interactúen directamente con los datos del catálogo en la base de datos.

Interfaz Gráfica del Usuario

Cada módulo del sistema se presenta al cliente como una aplicación Web, para reducir la sensación de independencia entre ellos se comparten elementos visuales comunes entre los mismos, facilitado esto gracias al uso de hojas de estilos comunes y estructuras de presentación similares (un encabezado, un menú en la parte izquierda, el cuerpo y el pie de página).

Además el pie de cada página presenta enlaces a los demás módulos, a los que se podrá ingresar sin la necesidad de autenticarse nuevamente en cada uno de ellos ya que se tiene configurada la autenticación única (*single sign on*).

CAPÍTULO 4

4. Definición e Implementación del Módulo de Pagos de Transacciones

El siguiente paso, luego del proceso de compra, es el pago del monto al proveedor. El sistema tradicional de pagos de facturas sobre un papel tiene un costo de suministros. La solución E-Procurement en el proceso de pagos, “ayuda a simplificar los procesos de tal manera que los costos se vean reducidos notablemente” [16].

En las secciones posteriores se detallará análisis y diseño de este módulo para el correcto manejo del proceso de pagos resultantes de las transacciones. Se iniciará formalizando los requerimientos de este módulo.

4.1. Requerimientos

Los requerimientos que debe satisfacer este módulo son los siguientes:

- Permitir el mantenimiento de los datos de las cuentas de las empresas registradas en E-Guana

- Permitir el pago de las órdenes de compras resultantes del uso de E-Guana mediante un método automático
- Proveer un mecanismo para la creación y mantenimiento de los datos de los bancos con los que trabajará E-Guana.

En la sección siguiente se desarrollará el análisis de estos requerimientos.

4.2. Análisis de Requerimientos

Lógicamente toda empresa deberá tener una cuenta bancaria en alguno de los bancos que estén trabajando con E-Guana, los datos de la misma serán proporcionados en la etapa de registro de la empresa. El módulo deberá permitir verificar la validez de dicha cuenta.

El pago de las órdenes de compra se manejará a través de las transacciones de débito o crédito a las cuentas correspondientes.

Obviamente los datos de las cuentas no son suficientes para completar el mecanismo de pagos, por lo que se necesita tener información de los bancos a las que pertenecen las mismas y, más importante aún, la forma como este módulo se comunicará con cada entidad bancaria. El mecanismo de comunicación deberá ser flexible, considerando la variedad de entidades bancarias y sistemas.

¿Por qué E-Guana da prioridad al débito bancario?

"Los mecanismos de pago siguen siendo una de las principales barreras para empresas y consumidores a la hora de formalizar un pedido o contratar un servicio directamente desde la Web. Las primeras buscan implantar la mejor solución, la más efectiva para el consumidor y la más segura. Los consumidores evitan dar sus tarjetas de crédito y temen comprar en Internet por miedo a sufrir fraudes y estafas" [23].

El comprador debe tener garantía de que nadie pueda, como consecuencia de la transacción que efectúa, suplantar en un futuro su personalidad efectuando otras compras en su nombre y a su cargo.

Aunque el método de pago con tarjeta de crédito sigue siendo el más usado, tal y como lo muestra las estadísticas de formas de pagos utilizadas, en nuestro país, la gente aún tiene desconfianza de usar su tarjeta de crédito a la hora de realizar pagos a través de algún sitio de compra-venta en Internet. Por lo tanto es necesario buscar otras alternativas que transmitan seguridad al usuario. Es debido a esto que en E-Guana se implementará el método de pago por medio del débito bancario/transferencia bancaria.

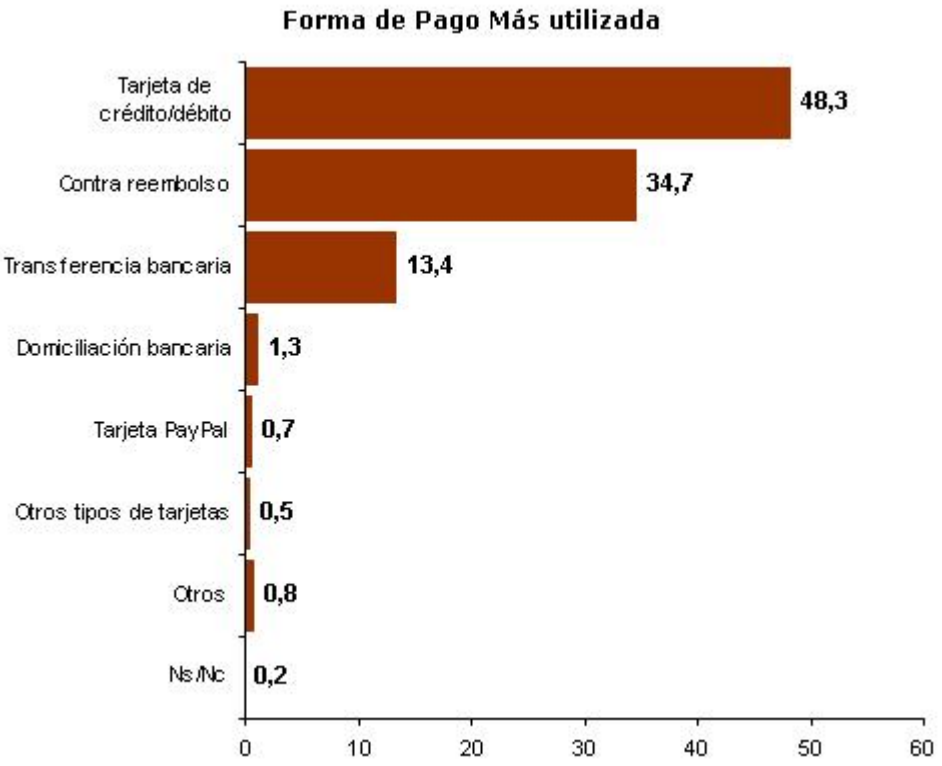


Figura 4-1 Formas de pago más utilizadas en Internet.
Fuente: http://www.ylos.com/spa/item/vendereninternet2.html [24]

Cualquiera que sea el medio de pago utilizado, en el 2006 se reporta como uno de los problemas más comunes a la hora de comprar por Internet el fraude con los medios de pago (14.3% de los problemas más comunes) [24].

Finalmente se indicarán algunas de las ventajas, que se pueden observar del método de cobro por medio de débito bancario:

- La seguridad en la transacción. El cliente no tiene que aportar los datos de su tarjeta de crédito cada vez que desea realizar una

compra, por lo que se evitan fraudes bancarios. Los datos de cuenta sólo son ingresados una vez y validados.

- Acceder a nuevos mercados. Esta forma de pago permite ofertar productos entre el mercado joven, que no dispone de tarjeta de crédito.
- No existe impuesto adicional que pagar por uso de este medio de pago.

Aunque E-Guana plantea el uso de los débitos bancarios para el manejo de los pagos de transacciones, el esquema no excluye el uso futuro de tarjetas de crédito para este pago.

4.2.1. Casos de uso

Para tener un buen entendimiento de lo que hará este módulo se recurrirá una vez más a los casos de uso, partiendo con el diagrama respectivo:

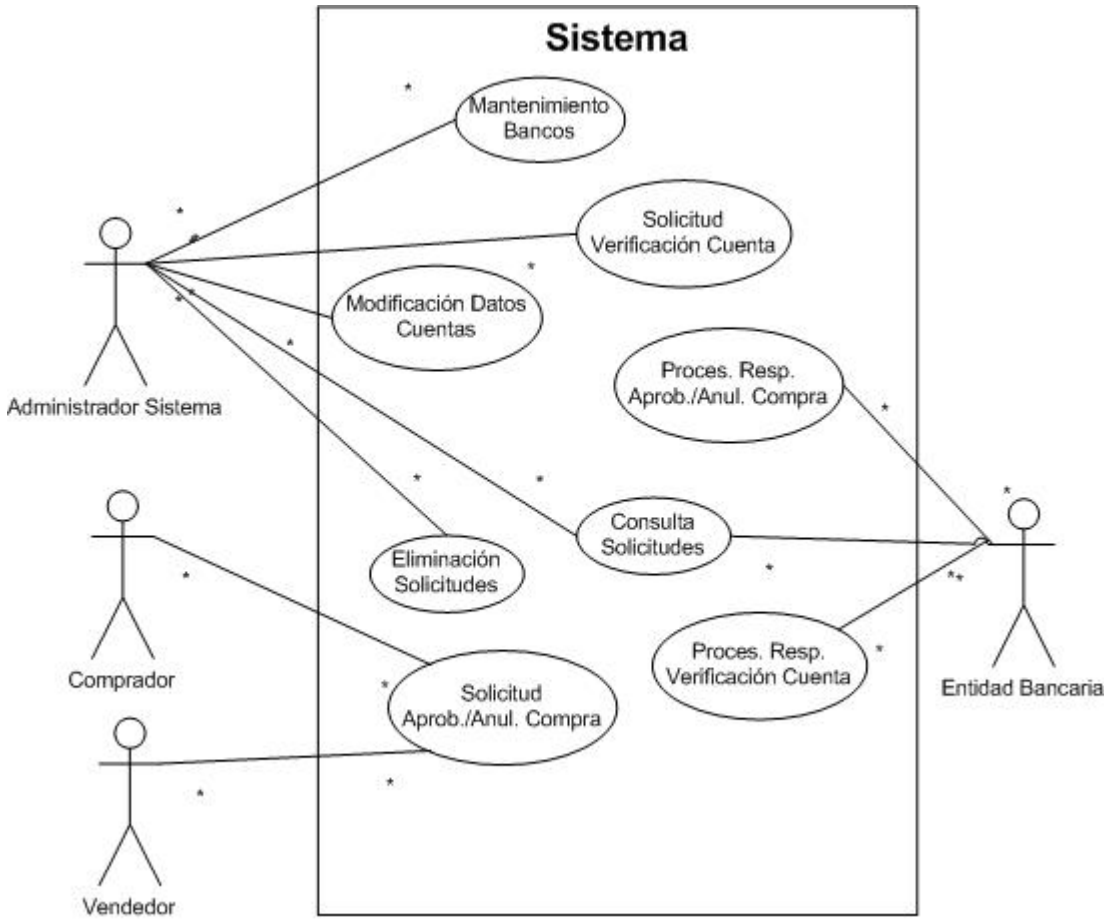


Figura 4-2 Diagrama de Casos de Uso del Módulo de Pagos de Transacciones
Fuente: E-Guana

Ahora se describirán los casos de uso del módulo de Pagos de Transacciones:

Caso de uso	**Mantenimiento de Bancos**
Objetivo	Hacer modificaciones a los datos de los bancos registrados.
Descripción	El "Administrador del Sistema" da mantenimiento a los datos de los bancos registrados, para que los clientes asociados a ellos, puedan hacer uso de E-Guana.
Valor medible	Se medirá en base a los datos modificados del o los bancos (nombre, tipo de comunicación y datos para lograr dicha comunicación).
Involucrados	El actor involucrado en este caso es el "Administrador del

	Sistema”.
Controles	El tipo de implementación para la comunicación del módulo con los bancos sólo puede ser JMS o XML, en el primer caso los datos de implementación debe contener el nombre de la cola de mensajes del banco.
Entradas	Las entradas del proceso son los datos del banco.
Salidas	La salida es un listado con los datos de los bancos existentes en el sistema.

Caso de uso	**Modificación de Datos de la Cuentas**
Objetivo	Realizar modificaciones de los datos de las cuentas bancarias de las empresas.
Descripción	El “Administrador del Sistema” modifica los datos de las cuentas de las empresas registradas, incluyendo el saldo y el número de cuenta.
Valor medible	Se medirá en base a los datos modificados de la cuenta.
Involucrados	El actor involucrado es el “Administrador del Sistema”.
Controles	Sólo se pueden modificar las cuentas de las empresas en estado “Activo”. No se podrá establecer el saldo con un valor negativo.
Entradas	La entrada son los datos de la cuenta a modificar.
Salidas	La salida será un listado con los datos de las cuentas.

Caso de uso	**Generación de Solicitud de Verificación de Cuenta**
Objetivo	Generar una solicitud de verificación de la cuenta bancaria.
Descripción	El “Administrador del Sistema” genera una solicitud de verificación de cuenta bancaria de una de las empresas registradas. También el sistema genera este tipo de solicitud automáticamente en la etapa de registro de la empresa.
Valor medible	Se medirá en base a la solicitud generada por el sistema. Además en caso de que la comunicación se puede iniciar desde E-Guana, otro valor medible es el mensaje JMS respectivo.
Involucrados	Los dos involucrados son el usuario que registra a la empresa y el “Administrador del Sistema”.
Controles	El usuario que registra a una empresa, solo puede escoger entre los banco registrados en el sistema, la cuenta bancaria puede ser cualquier cadena de texto. En el caso de que el banco asociado permita comunicación JMS, se envía el mensaje.
Entradas	Las entradas dependen de dos escenarios; para solicitudes originadas desde un nuevo registro de empresa, se requieren los datos del banco y la cuenta a verificar, mientras que para solicitudes del “Administrador del Sistema” a partir de los datos de la empresa se pueden determinar el banco y número de cuenta.
Salidas	La salida será un mensaje indicando si se generó o no la solicitud.

Caso de uso	Generación de Solicitudes de Aprobación/Anulación Compra
Objetivo	Generar una solicitud de aprobación o anulación de compra, según corresponda.
Descripción	Los módulos de Store Front y de Licitación & Subastas de E-Guana piden la generación de la solicitud de aprobación de la compra cuando crean una orden de compra. Además el módulo de Store Front también pude requerir la generación de la solicitud de anulación de una compra.
Valor medible	El valor medible será la solicitud generada por el sistema con estado pendiente. Además, en caso de que la comunicación se base en mensajería, otro valor medible es el mensaje JMS respectivo.
Involucrados	Los involucrados son los usuarios con rol de “Vendedor” y/o “Comprador” a través de módulos de Licitación & Subastas y el Store Front.
Controles	Las cuentas bancarias involucradas deben ser válidas y pertenecer al mismo banco. El saldo en la cuenta del comprador debe ser superior o igual al valor total de la compra en el caso de tratarse de una aprobación de compra.
Entradas	La entrada es la compra que desea ser aprobada o anulada.
Salidas	En caso de error en la salida se presentan datos que justifiquen el fracaso en la generación de la solicitud.

Caso de uso	Consulta de Solicitudes
Objetivo	Listar los datos de las solicitudes generadas.
Descripción	El “Administrador del Sistema” o una entidad del lado de la empresa bancaria consultan las solicitudes dirigidas al banco. El “Administrador del Sistema” tiene la facultad de acceder a los datos de las solicitudes con el objetivo de consulta o para una futura eliminación. La entidad bancaria que consulta las solicitudes lógicamente sólo podrá ver las solicitudes que le correspondan.
Valor medible	El valor medible será un listado con los datos completos de las solicitudes generadas.
Involucrados	Los involucrados en este caso son el “Administrador del Sistema” y la entidad bancaria.
Controles	En el caso de que sea la entidad bancaria la que solicita los datos de las solicitudes, sólo podrá recuperar aquellas que tiene como destino su Banco. El “Administrador del Sistema” podrá ver todas las solicitudes de todos los bancos.
Entradas	La entrada es el identificador del banco del que se desea mostrar las solicitudes, de igual forma es posible enviar el formato de presentación de los datos.
Salidas	La salida en este caso es el listado con los datos de las solicitudes

	en el formato requerido.

Caso de uso	**Eliminación de Solicitudes**
Objetivo	Eliminar una de las solicitudes previamente generadas.
Descripción	El “Administrador del Sistema” elimina las solicitudes que crea convenientes, para esto primeramente debe listar las solicitudes existentes a través de la “Consulta de Solicitudes”.
Valor medible	Se medirá en base al número de solicitudes eliminadas.
Involucrados	El involucrado es el “Administrador del Sistema”.
Controles	Las claves enviadas deben corresponder a solicitudes válidas.
Entradas	Las entradas serán las claves primarias de las solicitudes que se desean borrar, para esto la interfaz de usuario provee de componentes para la selección de las mismas.
Salidas	La salida es un mensaje con el número de solicitudes eliminadas exitosamente.

Caso de uso	**Procesamiento de Respuestas a Verificación de Cuenta**
Objetivo	Procesar una respuesta de verificación de cuenta bancaria.
Descripción	La cola de mensajes, que recepta las respuestas de las entidades bancarias, procesa una respuesta a una solicitud de verificación de cuenta. La empresa asociada a la cuenta puede cambiar su estado dependiendo de la respuesta obtenida. Adicional a lo anterior la solicitud pasa al estado de “Aprobada” o “Negada” según corresponda. Si la solicitud fue aprobada se actualiza el saldo de la cuenta bancaria de la empresa.
Valor medible	El valor medible será la solicitud en su nuevo estado y con fecha de atención y comentario. De aplicar, la empresa en su nuevo estado y con un valor distinto en su saldo.
Involucrados	En este caso el actor es la entidad bancaria.
Controles	La respuesta debe poder relacionarse con una solicitud en estado “Pendiente”. Además el identificador enviado en la respuesta debe corresponder a la clave primaria de una empresa registrada en E-Guana y el tipo de respuesta debe corresponder a verificación de cuenta. Aunque la cuenta bancaria no sea válida, debe enviarse un valor correspondiente al saldo. Si la empresa se encontraba en estado “Pendiente” pasará al estado “Pendiente-con cuenta válida” siempre que la solicitud fue aprobada, caso contrario pasará al estado de “Pendiente-sin cuenta válida”. Si la empresa se encontraba en estado “Activo” y la respuesta es negativa, la empresa pasará al estado “Bloqueada”, ya que se ha verificado que no tiene una cuenta bancaria que permita el uso del sistema, finalmente si la empresa se encontraba “Bloqueada” y el

	banco indica que si tiene una cuenta válida, el sistema automáticamente reactivará a la empresa.
Entradas	Las entradas serán el identificador de la empresa a la que corresponde la respuesta a la solicitud, el tipo de solicitud a la que se responde, si la respuesta aprueba o niega la solicitud, el saldo que tiene la cuenta bancaria, y un comentario u observación opcional.
Salidas	No hay salidas.

Caso de uso	**Procesamiento de Respuestas a Aprobación/Anulación de Compra**
Objetivo	Procesar una repuesta a una solicitud de aprobación o anulación de compra.
Descripción	La cola de mensajes procesa una respuesta a una solicitud de aprobación o anulación de compra. Para los casos de aprobación y anulación de compras, si la solicitud fue aprobada, y la compra está en estado de “En proceso”, esta última es pasada al estado de “Pagada”. Si el estado en el que se encontraba la compra era en proceso de “Anulación”, es decir, la respuesta corresponde a una anulación de compra, la compra pasa al estado de “Cancelada”. La otra alternativa es que la respuesta sea negativa, en cuyo caso para aprobaciones de compra, la misma es pasada al estado de “No aprobada”, mientras que para anulaciones de compras, la compra involucrada debe pasar al estado anterior al requerimiento de su anulación, además en esta última situación es necesario revertir las transacciones monetarias, es decir, acreditar al comprador y debitar al vendedor.
Valor medible	Se medirá en base a la solicitud asociada a la respuesta con el estado respectivo a la aprobación o no del requerimiento. También el estado de la compra asociada, así como los valores en las cuentas de las empresas.
Involucrados	En este caso el involucrado es la entidad bancaria.
Controles	La respuesta debe poder relacionarse con una solicitud en estado pendiente. Además el identificador de la compra debe ser válido, así como el tipo de respuesta debe ser aprobación o anulación de compra.
Entradas	Las entradas en este caso son el identificador de la compra que asociado a la respuesta, el tipo de solicitud a la que se responde, si la respuesta aprueba o niega la solicitud y un comentario u observación opcional.
Salidas	No hay salidas.

De acuerdo a lo anotado previamente, la funcionalidad del módulo de Pagos de Transacciones se puede agrupar en Administración de Cuentas de

Empresas y Débito Bancario. Estos grupos de funcionalidad se detallarán a continuación.

4.2.2. Administración de Cuentas de Empresas

La Administración de Cuentas de Empresas comprende todas las tareas relacionadas al aseguramiento de la validez de los datos de la cuenta de una empresa. Toda empresa registrada en el sistema deberá proveer un número de cuenta válido en una entidad bancaria activa registrada en E-Guana. La verificación de dicha cuenta es realizada durante el proceso de aprobación de la solicitud de ingreso de la empresa al Sistema.

El administrador de este módulo podrá dar mantenimiento a cada una de las cuentas de las empresas, más específicamente al valor del saldo de cada cuenta y al número de cuenta.

Cuando el administrador lo crea necesario, podrá solicitar al banco respectivo una nueva verificación de cuenta, si la respuesta del banco es que la cuenta no es válida, la empresa asociada a la cuenta automáticamente pasará al estado “Bloqueada”, para de esta forma evitar conflictos con las transacciones resultantes de las compras.

4.2.3. Débito Bancario

El módulo de Pagos de Transacciones tiene como una de sus principales tareas la comunicación con los sistemas de los distintos bancos registrados en E-Guana, la funcionalidad denominada Débito Bancario agrupa todas las tares relacionadas.

La amplia variedad de estos sistemas es una posibilidad indudable, por esta razón se requiere de un mecanismo que permita la comunicación entre ellos con el menor acoplamiento posible. Para lograr esto se ofrecen dos alternativas una basada en mensajes o mensajería y la otra basada en comunicación XML asincrónica. Actualmente en Ecuador, ningún banco cuenta con un sistema de débitos (para entes externos) en línea, ya que ese tipo de transacciones externas deben ser justificadas a la Superintendencia de Bancos, lo anterior fue indicado por el Departamento de Sistemas del Banco Bolivariano.

Es necesario tener en cuenta las posibles limitaciones en la comunicación (infraestructura, deficientes sistemas de integración) que den como resultado considerables tiempos de respuesta a la atención de las solicitudes hechas por este módulo, por lo cual es deseable evitar intercambios de información innecesarios entre el módulo de Pagos de Transacciones y los sistemas bancarios, y además asegurarse de intercambiar solo los datos indispensables en el momento justo. El problema indicado se pretende evitar,

utilizando los datos, obtenidos en el registro de la empresa, de las cuentas bancarias y validando las transacciones (suficiencia de fondos) internamente antes de generar cualquier solicitud a los sistemas externos.

A continuación se describe los dos mecanismos de débito bancario planteados:

Mensajería

Cuando una empresa realice una compra en E-Guana, previo a la verificación de la validez de la transacción por parte de este módulo, se enviará la solicitud en forma de un mensaje JMS al sistema del banco respectivo solicitando el débito del valor correspondiente por la compra entre las cuentas de las dos empresas involucradas. Depende del sistema del banco la celeridad con el que se atienda este requerimiento.

Una vez tomada una decisión, la misma deberá ser comunicada a este módulo, cuya tarea será entonces (dependiendo de la decisión) la de actualizar el estado de la compra además de hacer el débito y el crédito respectivo en las cuentas de las empresas. El módulo provee para la recepción de las respuestas una cola de mensajes.

XML

La diferencia de este mecanismo con el anterior es que el envío de los datos de la solicitud no se lo hace en el momento que se genera una compra, al contrario se almacenan los datos de la solicitud en el sistema y los mismos son enviados al banco ya sea automáticamente o de forma manual.

El envío de la respuesta del banco se lo puede hacer de forma similar a lo hecho en mensajería o a través de un documento XML con los datos de las respuestas de las solicitudes de débito.

El “Administrador del Sistema” es el que configura el mecanismo de comunicación para cada banco.

De la investigación realizada, se implementarán mecanismos para lograr realizar transacciones de débito efectivas con una entidad bancaria del Ecuador, en este caso el Banco Bolivariano.

Banco Bolivariano

El Banco Bolivariano ofrece 2 opciones para el tipo de operación de transacciones de débito, las cuales se describen a continuación:

- Archivos transaccionales de Débitos, Créditos y Pagos

La empresa que genera la transacción hacia el banco registra en un archivo plano (con el formato indicado por la entidad) dicha transacción.

Este archivo será enviado a la entidad bancaria para que ésta procese la transacción y las transacciones interbancarias son enviadas al Banco Central, el cual recibe el archivo hasta las 11AM para procesar el mismo día esas transacciones y otro al final de día contable (alrededor de 5PM o 6 PM) para procesar al siguiente día las transacciones generadas después de las 11AM.

- Servicio de Asistencia Transaccional (SAT)

Este mecanismo trabaja de forma similar, manejando archivos con transacciones, la diferencia radica en que interactúa directamente sobre cuentas locales, es decir, la misma entidad (Bolivariano) procesa las transacciones de sus propios clientes en el mismo instante, para lo que el sistema E-Guana proveerá información con las cuentas a ser procesadas. Si la transacción de fondo sigue siendo una interbancaria se aplicará el primer esquema.

En este caso los costos de operación son un poco más altos.

En el caso de E-Guana la mejor opción será utilizar el Servicio SAT ofrecido por el banco. En este caso el sistema deberá generar el archivo plano, el cual debe poseer 2 registros de transacciones por cada proceso de compra-venta, para lo cual el sistema E-Guana requerirá una cuenta bancaria maestra en esta entidad.

Para la generación de dicho archivo plano con los datos de transacciones, el mecanismo XML planteado previamente es el más adecuado, ya que tan solo se requerirá el diseño de una hoja de estilos que deberá ser aplicada al documento XML resultante.

En el caso de que el cliente A con número de cuenta 1000000 vende un artículo X que es comprado por el cliente B con número de cuenta 2000000, el sistema debe generar los siguientes registros de transacción:

1. Generar un registro para el débito de la cuenta del cliente B con crédito a la cuenta de E-Guana.

2. Generar registro para el crédito a la cuenta del cliente A debitando de la cuenta de E-Guana.

Los costos de operación se cargarían a la cuenta de E-Guana. Las transacciones de cobros y pagos deben ser enviados en archivos diferentes.

A continuación se adjunta el formato del archivo de recaudaciones (cobros) requerido por el banco:

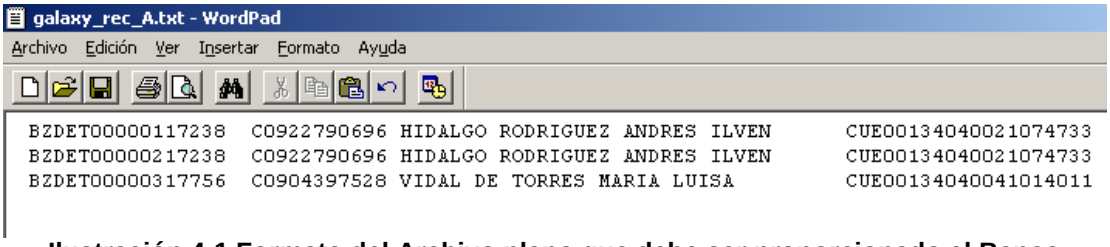


Ilustración 4-1 Formato del Archivo plano que debe ser proporcionado al Banco Bolivariano para procesar las transacciones de pago. (Fragmento 1/2)
Fuente: Departamento de Sistemas del Banco Bolivariano.

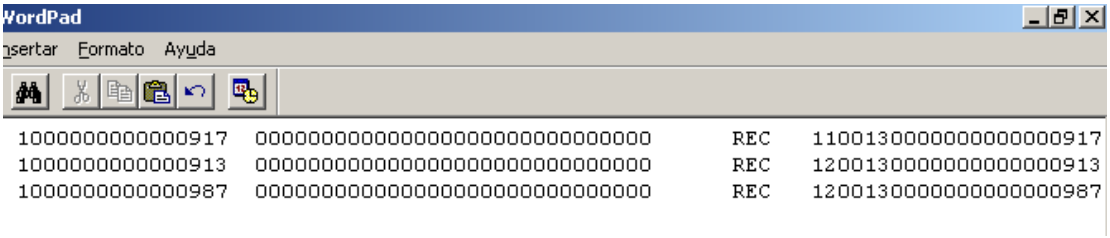


Figura 4-3 Formato del Archivo plano que debe ser proporcionado al Banco Bolivariano para procesar las transacciones de pago. (Fragmento 2/2)
Fuente: Departamento de Sistemas del Banco Bolivariano.

El formato del ejemplo anterior se aplicaría similar a los archivos de créditos, la única diferencia radica en el tipo de transacción (CRE – crédito) identificado en cada registro de dicho archivo.

4.3. Diseño

En esta sección se detallarán los elementos principales del diseño del módulo de Pagos de Transacciones, iniciando con el diagrama de clases,

luego los diagramas de interacción de objetos y finalmente el modelo entidad relación.

4.3.1. Diagrama de Clases

El módulo de Pago de Transacciones se compone de las siguientes clases:

- Banco
- Solicitud al Banco
- Mensaje

La siguiente ilustración describe las clases involucradas, sus atributos y principales métodos.

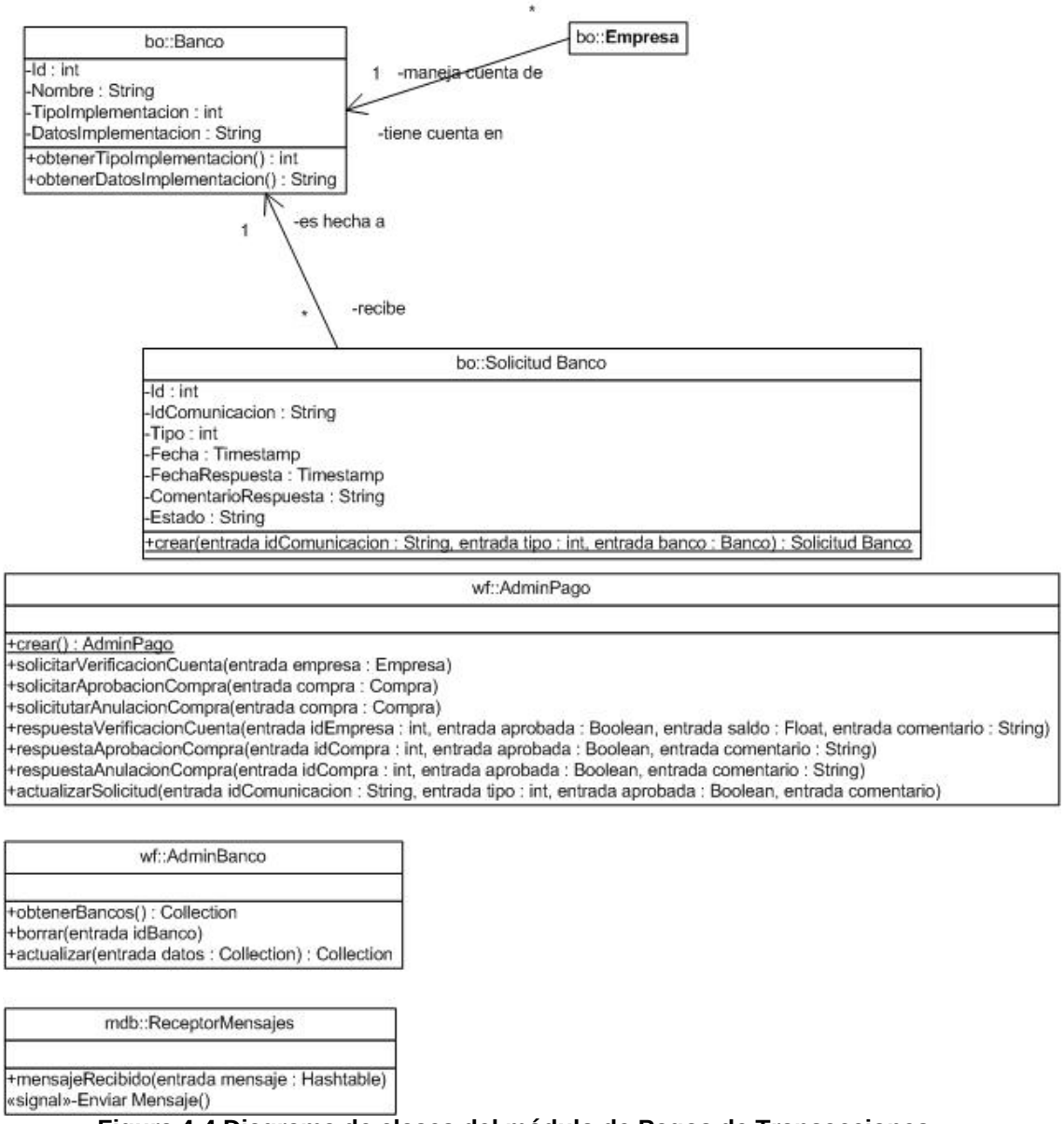


Figura 4-4 Diagrama de clases del módulo de Pagos de Transacciones.
Fuente: Diagrama principal de clases del Sistema E-Guana.

Banco

La clase Banco representa al banco que tiene algún convenio o acuerdo con E-Guana para permitir que sus clientes puedan hacer uso del sistema. Los

datos que se tienen del banco son el nombre y los datos necesarios para poder comunicarse con el mismo.

Los atributos de la clase Banco son los siguientes:

- **ID:** El identificador del banco.
- **Nombre:** El nombre del banco.
- **Tipo de Implementación:** corresponde al tipo de implementación o mecanismo de comunicación usado entre E-Guana y dicho banco. Los valores que puede tomar son JMS para mensajería y XML para la comunicación con documentos XML.
- **Datos de Implementación:** indica algún dato necesario para completar la comunicación. Dependiendo del tipo de implementación de la comunicación es la naturaleza de este atributo, en el caso de JMS, éste contiene el nombre de la cola de mensajes a la que deben ser enviados los mensajes.

Si se desean manejar nuevos tipos de comunicación es necesario configurar los atributos “Tipo de Implementación” y “Datos de Implementación” junto con la lógica de negocio específica.

Solicitud al Banco

Esta clase representa los datos de toda solicitud que debe ser dirigida a alguna de las entidades bancarias registradas en E-Guana.

Existen tres tipos de solicitudes:

1. Solicitud de verificación de validez de una cuenta
2. Solicitud de aprobación de una compra
3. Solicitud de anulación de una compra

Una solicitud es creada por pedido del módulo de Store Front y en el caso de las solicitudes relacionadas a la compra, por pedido del módulo de Administración, cuando se registra una empresa en el sistema y se requiere la verificación de la cuenta que dice tener la empresa en cierto banco y por último por el propio módulo de Pago de Transacciones, cuando también requiere una nueva verificación de la validez de una cuenta.

Considerando lo anterior cada solicitud puede estar refiriéndose a una empresa a la que se le desea verificar la cuenta o a una compra que requiere ser aprobada o anulada, para evitar almacenar estos datos en dos campos con valores nulos, se decidió manejar esto con un solo campo denominado *ID de comunicación* que junto al tipo de solicitud permite inferir la entidad a la que se refiere una solicitud.

Cuando una solicitud es creada su estado es “Pendiente”. Una solicitud en estado “Pendiente” puede luego ser “Aprobada” o “Negada” dependiendo de las respuestas enviadas por los bancos.

Los atributos de la clase Solicitud al Banco son:

- **ID:** identificador de la solicitud.
- **Tipo:** indica si la solicitud es una verificación de cuenta, una aprobación de compra o una anulación de compra.
- **Fecha:** la fecha en la que se creó la solicitud.
- **Fecha de respuesta:** corresponde a la fecha en la que se recibió la respuesta a esta solicitud, inicialmente su valor es nulo.
- **Comentario de respuesta:** es un texto enviado por el banco que considera relevante, o explica la razón de la respuesta a esta solicitud. Su valor inicial es nulo.
- **ID de Comunicación:** el ID de la compra si esta solicitud corresponde a una aprobación de compra o anulación de compra o el ID de la empresa si la solicitud es una verificación de cuenta.
- **Estado:** el estado de la solicitud, los estados posibles son:

    - Pendiente: es el estado inicial de toda solicitud
    - Aprobada: la solicitud recibió una respuesta afirmativa.
    - Negada: la solicitud recibió una respuesta negativa del banco.

No está de más indicar que cada solicitud tendrá sólo uno de los estados anteriormente descritos, mismo que dependerá de la respuesta enviada por el banco.

Mensaje

Es el mensaje que se envía a las colas de mensajes y es atendido por los *bean* manejados por mensajes, es un *hashtable.*

Se procedió de esta forma, en lugar de implementar una clase, debido a que el *hashtable,* al tratarse de una clase propia del JDK, es común a todas las aplicaciones desplegadas y por lo tanto es fácil enviarla entre aplicaciones sin la preocupación de obtener un no deseable *ClassNotFoundException* (la clase no puede ser encontrada dentro de la aplicación)*,* además con esto se puede trabajar de una forma verdaderamente desacoplada siempre y cuando los objetos contenidos en el *hashtable* sean de igual forma, clases propias del JDK.

Los atributos del mensaje, cuando se trata de una respuesta del módulo de algún banco, son los siguientes:

- **IdComunicacion**: el identificador que junto con el tipo de solicitud es necesario para discriminar entre solicitudes.

- **TipoAprobacion**: debe tomar alguno de los siguientes valores: “VerificacionCuenta”, "AprobacionCompra" o "AnulacionCompra".
- **Exito**: Si se aprobó la solicitud este atributo es verdadero, caso contrario debe ser falso.
- **ComentarioAprobacion**: comentario del banco, normalmente debería indicar las razones de su decisión.
- **Saldo**: para verificaciones de cuentas, este valor contiene el saldo que tiene la cuenta en el banco. Para las otras solicitudes este campo no es enviado.

En cambio si el mensaje corresponde a una solicitud de verificación de cuenta generada por el módulo de Pago de Transacciones E-Guana (hacia un banco con comunicación JMS), los atributos del mensaje son los siguientes:

- **IdComunicacion:** en este caso corresponde al identificador de la empresa a la que se le desea verificar la cuenta.
- **TipoAprobacion:** en este caso el único valor que puede tomar es "VerificacionCuenta".
- **NumeroCuenta:** indica el número de cuenta que se desea verificar.
- **Comentario:** contiene datos que pueden ayudar al banco en la verificación de la cuenta (nombre de la empresa, RUC, etc.).

Por último cuando el mensaje corresponde a una solicitud de aprobación o anulación de una compra, los atributos del mensaje son:

- **IdComunicacion:** en este caso corresponde al identificador de la compra a la que se le desea aprobar o anular.
- **TipoAprobación:** si se desea aprobar la compra el valor es “AprobacionCompra”, en el caso de anulación es "AnulacionCompra".
- **Valor:** valor total de la compra.
- **CuentaComprador:** identificador de la cuenta del comprador.
- **CuentaVendedor:** identificador de la cuenta del vendedor.

Cada uno de estos atributos se puede recuperar del mensaje a través del nombre respectivo.

4.3.2. Diagramas de Interacción de Objetos

A continuación algunos diagramas de interacción de objetos involucrados en el Módulo de Pago de Transacciones:

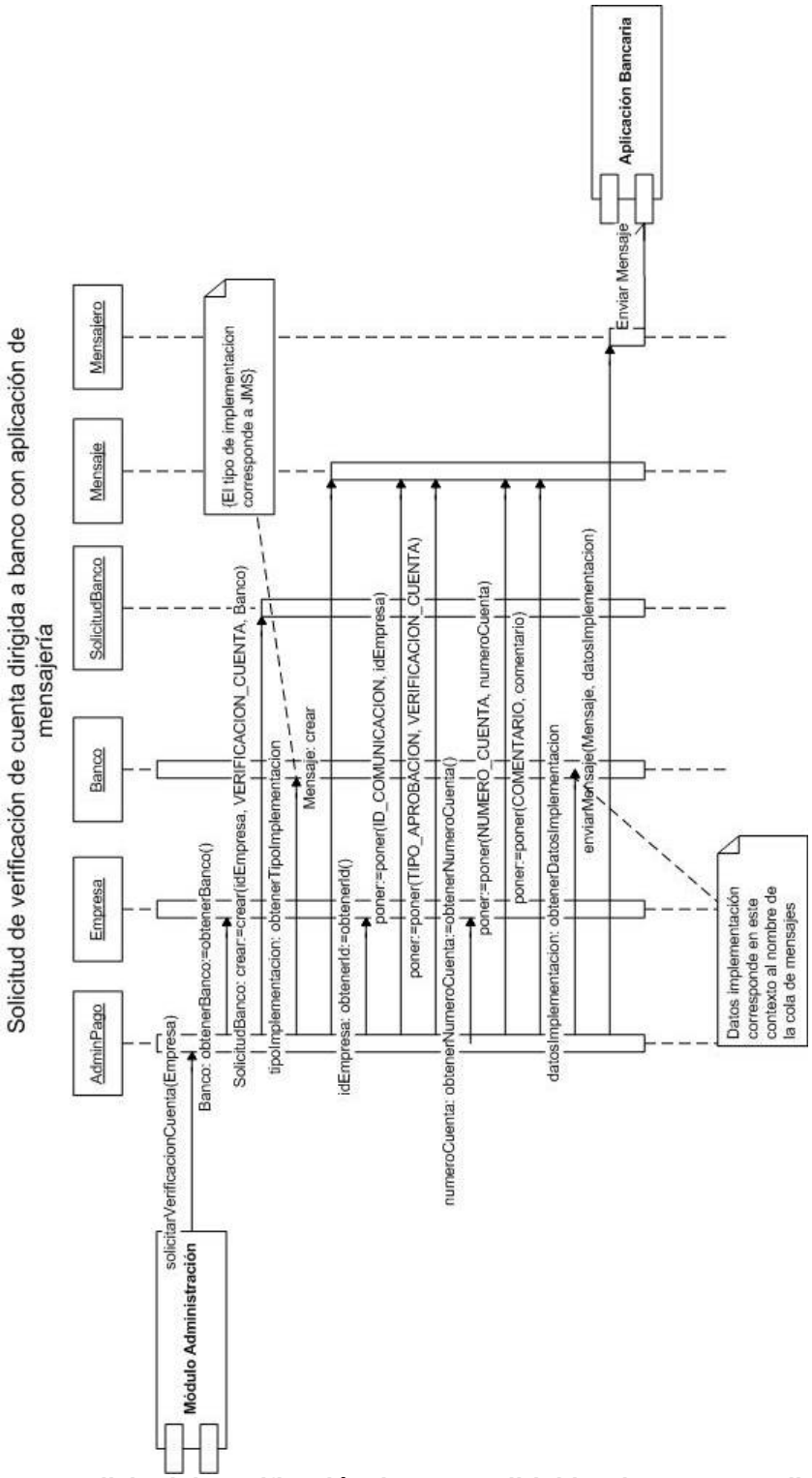


Figura 4-5 DIO Solicitud de verificación de cuenta dirigida a banco con aplicación de mensajería.
Fuente: DIOs del Módulo de Pagos de Transacciones.

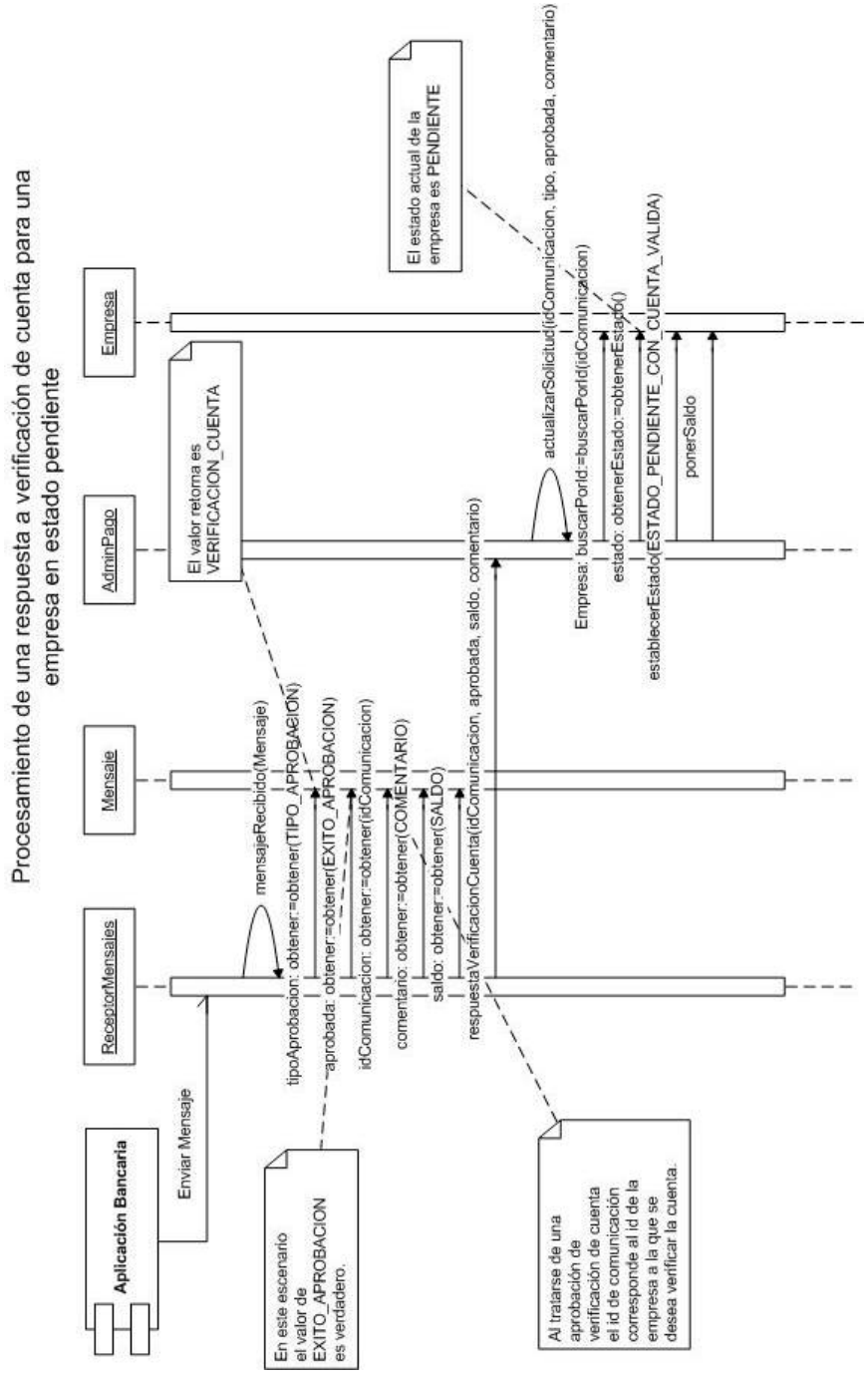


Figura 4-6 DIO Procesamiento de una respuesta a verificación de cuenta para una empresa en estado pendiente.
Fuente: DIOs del Módulo de Pagos de Transacciones

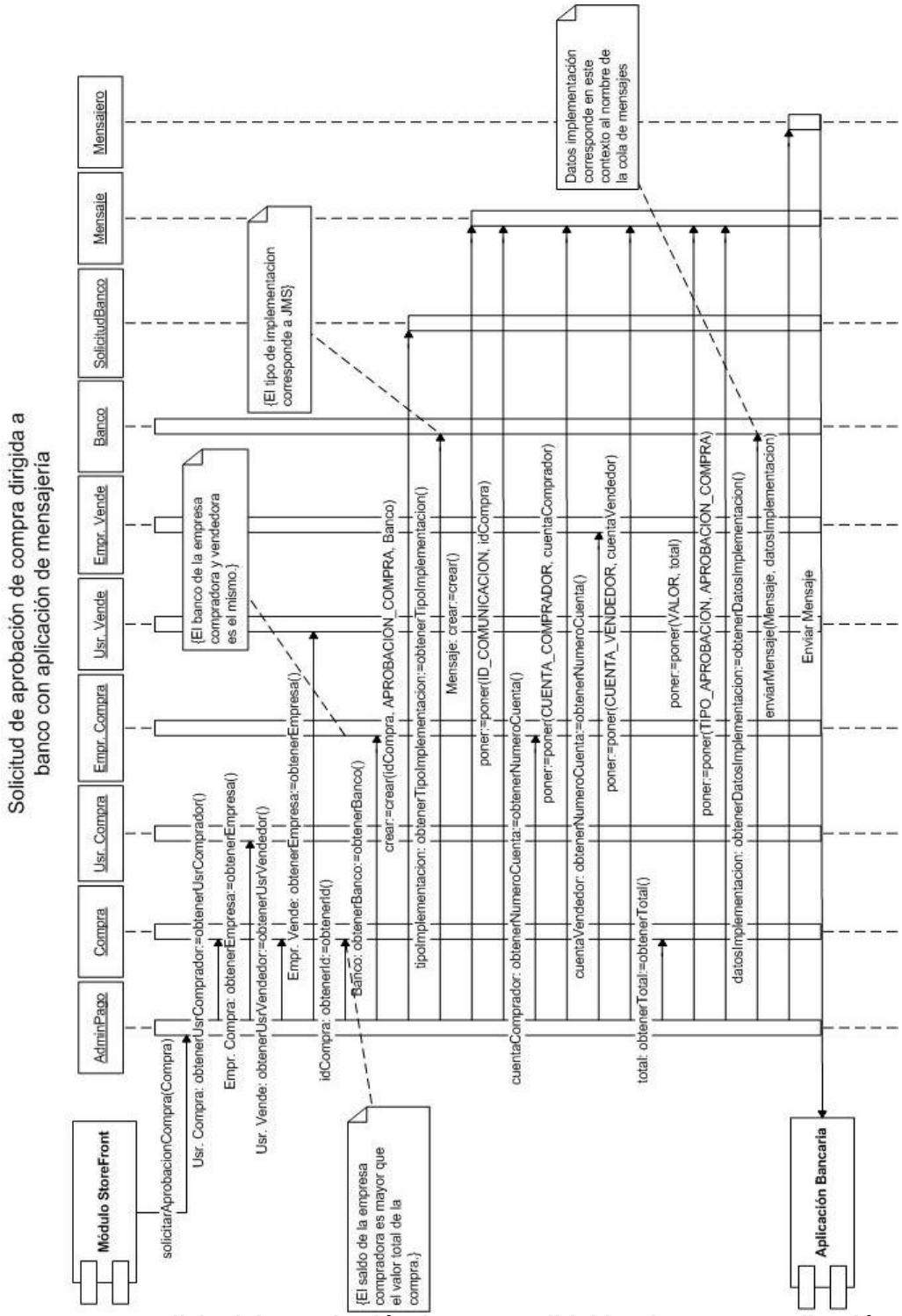


Figura 4-7 DIO Solicitud de aprobación de compra dirigida a banco con aplicación de mensajería.
Fuente: DIOs del Módulo de Pagos de Transacciones

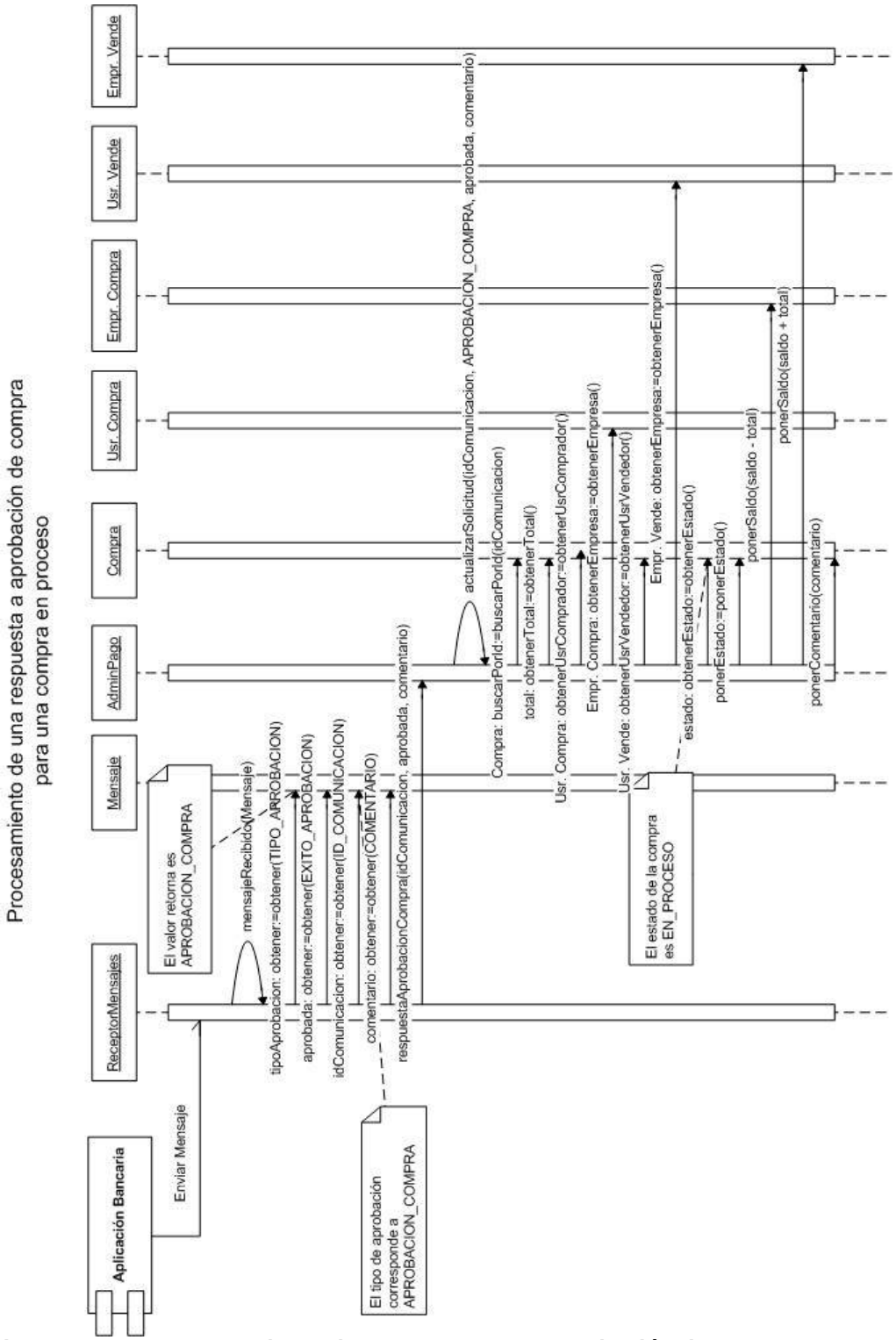


Figura 4-8 DIO Procesamiento de una respuesta a aprobación de compra para una compra en proceso.
Fuente: DIOs del Módulo de Pagos de Transacciones

4.3.3. Modelo Entidad-Relación

El diagrama Entidad-Relación del módulo de Pago de Transacciones se presenta en la siguiente ilustración:

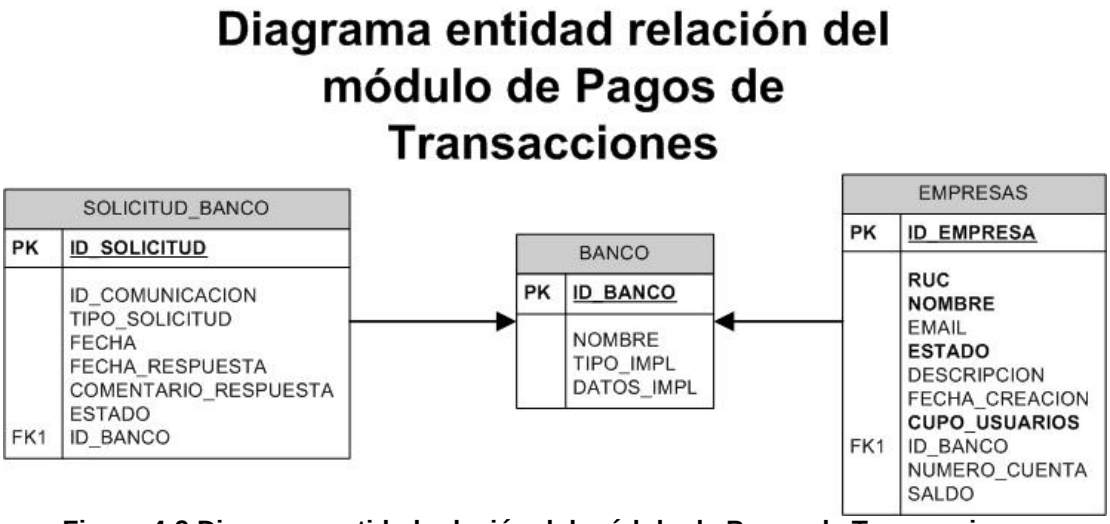


Figura 4-9 Diagrama entidad relación del módulo de Pagos de Transacciones.
Fuente: Diagrama Entidad-Relación del Módulo de Pagos de Transacciones

4.4. Estructura

4.4.1. Componentes utilizados en la implementación

De manera similar a lo indicado en el capítulo 3, se implementaron 3 tipos de componentes para construir la aplicación J2EE que representa a este módulo, cada uno de los cuales se detallará más adelante.

4.4.1.1. EJBs

Para el Módulo de Pago de Transacciones se utilizaron 3 tipos de *beans*, de entidad, de sesión y manejados por mensajes.

Paquete	Nombre
org.eguana.pagos.bo.ejb	BancoBean
org.eguana.pagos.bo.ejb	SolicitudBancoBean

Tabla 4-1 *Beans* de Entidad del Módulo de Pagos de Transacciones.

Paquete	Nombre
org.eguana.pagos.wf.ejb	AdminPagoBean
org.eguana.pagos.wf.ejb	AdminBancoBean

Tabla 4-2 *Beans* de Sesión del Módulo de Pagos de Transacciones.

Paquete	Nombre
org.eguana.pagos.mdb.ejb	ReceptorMensajesBean

Tabla 4-3 *Beans* Manejados por Mensajes del Módulo de Pagos de Transacciones.

Beans de entidad

De manera similar a lo indicado en la sección EJBs del capítulo 3, a continuación se describirán los *beans* de entidad implementados en el módulo de Pago de Transacciones.

En el caso de este módulo los *beans* de entidad están ubicados en el paquete org.*eguana.pagos.bo.ejb*.

BancoBean

Este *bean* corresponde a la clase Banco, es decir controla la persistencia de los datos de todos los bancos que tienen algún convenio con E-Guana para que sus clientes puedan utilizar el sistema. Deriva de la clase ClienteSecuencia e implementa *javax.ejb.EntityBean.*

Los atributos que expone esta clase son: ID, nombre, tipo de implementación y datos de implementación.

SolicitudBancoBean

Este *bean* corresponde a la clase SolicitudBanco, deriva de ClienteSecuencia e implementa EntityBean.

Los atributos de esta clase son: ID, ID de comunicación, tipo, fecha, fecha de respuesta, comentario de respuesta y estado.

- Métodos de acceso para campos de relaciones

Cada solicitud es dirigida a alguno de los bancos registrados con E-Guana, para manejar esta relación se tiene una pareja de métodos que trata con interfaces locales de la clase BancoBean. En la base de datos esta relación se maneja con una clave foránea en la tabla *solicitud_banco*.

- Métodos de búsqueda

Este *bean* provee dos métodos de búsqueda tipo *finder.* El primero retorna una lista de SolicitudBancoBean de acuerdo al identificador de la comunicación y el tipo de solicitud. El segundo método retorna todas las solicitudes pendientes dirigidas a un banco específico.

Beans de sesión

A continuación se describirán los *beans* de sesión del módulo de Pagos de Transacciones.

AdminPagoBean

Este *bean* de sesión es el más importante del módulo de Pagos de Transacciones.

Permite generar todas las solicitudes necesarias (verificación de validez de cuentas bancarias, aprobación de compras, anulación de compras) para el correcto funcionamiento de los pagos asociados a las compras generadas por los otros módulos del sistema.

Otra tarea de esta clase es procesar las respuestas de los bancos a esas solicitudes.

También genera, en base a requerimientos, documentos XML con los datos de las solicitudes recibidas.

Cuando se genera el documento XML para solicitudes de aprobación o anulación de compras, el respectivo DTD es el siguiente:

```
<!ELEMENT aprobaciones (aprobacion*)>
<!ELEMENT aprobacion
(IdComunicacion, Fecha, TipoAprobacion, Valor, CuentaComprador,
CuentaVendedor)>
<!ELEMENT IdComunicacion (#PCDATA)>
<!ELEMENT Fecha (#PCDATA)>
<!ELEMENT TipoAprobacion (#PCDATA)>
<!ELEMENT Valor (#PCDATA)>
<!ELEMENT CuentaComprador (#PCDATA)>
<!ELEMENT CuentaVendedor (#PCDATA)>
```

En tanto que para el listado de solicitudes de verificación de cuenta, el documento XML se valida contra el siguiente DTD:

```
<!ELEMENT verificaciones (verificacion*)>
<!ELEMENT verificacion
(IdComunicacion, NumeroCuenta, Fecha, Comentario)>
<!ELEMENT IdComunicacion (#PCDATA)>
<!ELEMENT NumeroCuenta (#PCDATA)>
<!ELEMENT Fecha (#PCDATA)>
<!ELEMENT Comentario (#PCDATA)>
```

Vale la pena indicar que el uso de estos archivos agilita el procesamiento de las solicitudes, permitiendo ejecutar transacciones por lotes.

AdminBancoBean

Este *bean* de sesión permite la administración de los datos de los bancos que utilizan E-Guana. Deriva de la clase SessionBeanBase e implementa las interfaces javax.ejb.SessionBean y org.eguana.pagos.Constantes.

Beans manejados por mensajes

El módulo de Pago de Transacciones tiene un *bean* manejado por mensajes, ReceptorMensajesBean, ubicado en el paquete *org.eguana.pagos.mdb.ejb*.

MensajesABancoBean

Este *bean* se encarga de atender los mensajes enviados a la cola identificada por el nombre JNDI *queue/queue/EguanaRespuestasASolicitudes*, los mensajes que recepta corresponden a las respuestas de las solicitudes dirigidas a los bancos y enviadas directamente por las aplicaciones bancarias o por el *servlet* que procesa las respuestas a partir de una archivo XML.

Esta clase implementa las interfaces javax.ejb.MessageDrivenBean, javax.jms.MessageListener y la interfaz útil *org.eguana.pagos.Constantes*.

El funcionamiento de esta clase, descrito de manera sencilla, es extraer los datos del mensaje recibido y luego, de acuerdo a los mismos, invocar a uno de los métodos del *bean* de sesión AdminPagoBean, éstos últimos tienen la lógica de negocio para el procesamiento de los tipos de respuestas que acepta el sistema.

4.4.1.2. JSPs

La aplicación *Web* que conforma el módulo de Pago de Transacciones presenta una estructura e implementación similar a la aplicación *Web* del módulo de Administración, por lo que las consideraciones apuntadas en dicho módulo también aplican aquí (uso del *framework* Sofia, páginas y controladores).

Los dos controladores usados en este módulo derivan de una copia del controlador *org.eguana.administracion.web.ControlBase,* por lo que se omitirá su descripción.

Mantenimiento de Bancos

Esta es la página utilizada para el mantenimiento de los bancos registrados en E-Guana, la ubicación de la misma es “/jsp/bancos.jsp”.

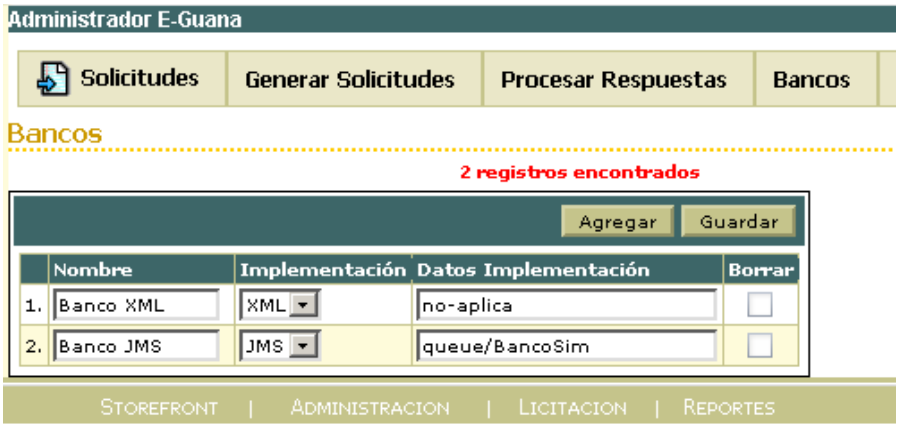


Figura 4-10 Pantalla mantenimiento de bancos

La página presenta controles para la modificación y presentación de los datos, así como controles para agregar un nuevo banco y para guardar los datos editados.

La columna etiquetada como “Implementación”, corresponde al atributo tipo de implementación de la clase Banco y puede tomar los valores a seleccionarse JMS ó XML.

La columna “Datos Implementación” debe contener en el caso de que implementación corresponda a JMS, el nombre de la cola de mensajes a la que se le deben enviar las solicitudes, mientras que para XML, éste dato no es utilizado actualmente.

Para borrar uno o más bancos, se debe activar el componente de selección de la columna “Borrar” y luego presionar el botón “Guardar”.

Modificación de los Datos de las Cuentas

La página JSP para la modificación de los datos de las cuentas bancarias de las empresas es “/jsp/cuentas.jsp”

Figura 4-11 Pantalla mantenimiento de cuentas

El listado muestra el nombre de la empresa y el nombre del banco donde tiene su cuenta bancaria, y permite la modificación de los datos correspondientes al número de cuenta y el saldo de la cuenta.

Adicionalmente se presenta una imagen enlace ✔ que permite al "Administrador del Sistema" la generación de una solicitud de verificación de cuenta.

Consulta y Eliminación de Solicitudes

El "Administrador del Sistema" puede listar todas las solicitudes existentes en el sistema, a través de la página "/jsp/solicitudes.jsp":

	Banco	Tipo	Estado	Fecha de envío	Fecha de respuesta	Comentario Respuesta	Borrar
1.	Banco JMS	APROBACION COMPRA Comandato -> Akros, 1344.0	PENDIENTE	Junio 03 2006, 14:34	Junio 08 2006, 13:49		
2.	Banco JMS	APROBACION COMPRA Comandato -> Akros, 25.76	PENDIENTE	Junio 03 2006, 15:00	Junio 08 2006, 13:49		
3.	Banco JMS	APROBACION COMPRA Comandato -> Akros, 1176.0	PENDIENTE	Junio 03 2006, 15:31	Junio 08 2006, 13:49		

Figura 4-12 Pantalla consulta de solicitudes

Esta página permite también la eliminación de una o más solicitudes, para esto el usuario debe seleccionar cada uno de las solicitudes a eliminar y luego presionar el botón “Enviar”.

	Estado	Fecha de envío	Fecha de respuesta	Comentario Respuesta	Borrar
dato	PENDIENTE	Junio 03 2006, 14:34	Junio 08 2006, 13:49		☑
dato	PENDIENTE	Junio 03 2006, 15:00	Junio 08 2006, 13:49		☑
dato	PENDIENTE	Junio 03 2006, 15:31	Junio 08 2006, 13:49		☐
t	APROBADA	Junio 07 2006, 20:46	Junio 13 2006, 23:40	quinientos	☑

Figura 4-13 Pantalla eliminación de solicitudes

Consulta de Solicitudes con formato:

Es posible acceder a los datos de las solicitudes en distintos formatos a través de la página “/jsp/generaSolicitud.jsp”:

Figura 4-14 Pantalla listado con formato de solicitudes

Los formatos soportados son: XML, HTML y RSS

XML: En el formato XML, el resultado corresponde al documento XML que genera el *bean* de sesión *AdminPagoBean* con sus métodos *obtenerAprobaciones* y *obtenerVerificaciones*.

HTML: El resultado es una página Web con los datos de las solicitudes en una tabla HTML.

RSS: Presenta los datos en un documento RSS versión 0.91.

Esta página no se implementó utilizando Sofia, sino con Servlets, el servlet más importante es *org.eguana.pagos.web.servlets.SolicitudesServlet.*

Para presentar los datos en uno de los formatos indicados, este servlet primero hace uso de los métodos *obtenerVerificacionesCuenta* y *obtenerAprobaciones* del *bean AdminPagoBean* para obtener el documento XML con las datos de las solicitudes, luego de acuerdo al formato deseado selecciona una hoja de estilos para con la misma, transformar el documento XML y finalmente enviar la salida de dicha transformación al cliente.

Para el correcto funcionamiento del servlet, el cliente debe enviar el identificador del banco del que se desean las solicitudes así como el tipo de solicitud y el estilo de la respuesta.

Procesamiento de Respuestas

La página para subir o enviar un archivo en formato XML con las respuestas de un banco a las solicitudes dirigidas a él, es "/jsp/procesaSolicitud.jsp".

Figura 4-15 Pantalla envío de archivo con respuestas

Similar a la página anterior, ésta utiliza Servlets en su funcionalidad.

El servlet principal de la implementación de esta página es *org.eguana.pagos.web.servlets.ReceptorRespuestasServlet*.

Una vez que el cliente envía el archivo, el mismo es verificado para comprobar si es un documento XML válido, luego cada entrada, correspondiente a una respuesta a una solicitud, es procesada resultando en la generación de un mensaje al *bean* manejado por mensajes que atiende las respuestas.

4.4.1.3. Etiquetas JSP personalizadas

Las etiquetas JSP personalizadas del módulo de Pagos de Transacciones están implementadas de tal forma que puedan ser utilizadas por un servidor Web externo a E-Guana que requiera la presentación de los datos de las solicitudes dirigidas a un banco, por lo tanto podrían ser embebidas en las aplicaciones Web existentes de los bancos registrados en el sistema. Los requisitos para su uso son disponibilidad de un servidor de aplicaciones Web JSP y la librería de etiquetas JSTL.

El conjunto de etiquetas implementadas está liderado por dos de ellas: *Solicitudes* y *CampoSolicitud*.

Etiqueta Solicitudes

Esta etiqueta es la principal del conjunto de etiquetas de este módulo, no obstante son dos etiquetas derivadas de ésta las que se usan en las páginas.

Su función es la de poder iterar sobre las solicitudes de cierto tipo dirigidas al banco, por esta razón la clase manejadora, *org.eguana.pagos.web.tags.SolicitudesTag* extiende la clase *javax.servlet.jsp.jstl.core.LoopTagSupport* de las librerías JSTL. La iteración la realiza de hecho sobre el documento XML retornado por el *servlet org.eguana.pagos.web.servlets.SolicitudesServlet* descrito en la sección anterior, esto se explica para separar por completo la implementación J2EE (*beans* de sesión) de los clientes que harían uso de estas etiquetas.

La etiqueta acepta como contenido código JSP y su único atributo obligatorio es “idBanco”, que debe ser un número correspondiente al identificador del banco en E-Guana.

Las dos etiquetas derivadas de ésta son *Aprobaciones* y *Verificaciones,* la primera permite la recuperación de las solicitudes de aprobaciones y anulaciones y la segunda recupera las de verificación de cuenta.

Etiqueta CampoSolicitud

Esta etiqueta se encarga de presentar el valor que tiene un cierto atributo de una solicitud, las derivaciones de esta etiqueta son las que especifican el atributo que se debe mostrar.

La clase manejadora de esta es *org.eguana.pagos.web.tags.CampoSolicitudTag* y deriva *javax.servlet.jsp.tagext.TagSupport.*

La etiqueta no acepta contenido y siempre debe ubicarse dentro de una etiqueta de tipo *Solicitudes,* es decir, dentro de *Aprobaciones* ó *Verificaciones*.

Para recuperar el valor del atributo se busca en el documento XML, expuesto por la etiqueta contenedora, el nodo que concuerde con el nombre del atributo buscado, si el nodo no existe o su valor no es nulo se muestra un espacio en blanco.

Las etiquetas derivadas de ésta son las siguientes:

1. IdComunicacion
2. Fecha
3. Comentario
4. TipoAprobacion
5. Valor

6. Cuenta

A excepción de *Cuenta,* las etiquetas no aceptan atributos, ésta en cambio acepta un atributo opcional llamado “deComprador”, si este último toma el valor de “si” el campo que se recupera es el número de cuenta de la empresa compradora caso contrario se recupera el número de cuenta de la empresa vendedora. Si la etiqueta está contenida en una etiqueta *Verificaciones* el valor que se obtiene es el del número de la cuenta a verificar.

A continuación se muestra el código de una página que hace uso de las etiquetas:

```
<%@ taglib prefix="pagos" uri="/WEB-INF/tlds/pagos.tld" %>
…
<h2>Verificaciones de Cuentas</h2>
<table border="1">
        <tr> …
        </tr>
<pagos:verificaciones idBanco="5">
        <tr>
                <td><pagos:idComunicacion /></td>
                <td><pagos:cuenta /></td>
                <td><pagos:comentario /></td>
                <td><pagos:fecha /></td>
        </tr>
</pagos:verificaciones>
</table>

<h2>Aprobaciones/Anulaciones de Compras</h2>
<table border="1">
        <tr> …
        </tr>
<pagos:aprobaciones idBanco="5">
        <tr>
                <td><pagos:idComunicacion /></td>
                <td><pagos:tipoAprobacion /></td>
                <td><pagos:valor /></td>
                <td><pagos:fecha /></td>
                <td><pagos:cuenta deComprador="si" /></td>
                <td><pagos:cuenta deComprador="no" /></td>
```

```
            </tr>
</pagos:aprobaciones>
…
```

Como se puede observar en el código ejemplo, estas etiquetas deben ser referenciadas con la librería de etiquetas (TLD) correspondiente, declarada a través de la directiva *taglib*.

4.5. Interacción con los módulos del Sistema

El uso de este módulo por parte del resto de módulos se da a través del *bean* de sesión *AdminPagoBean,* que es el elemento principal que expone la funcionalidad del módulo de Pago de Transacciones.

Más específicamente cuando el módulo de Administración desea registrar una empresa, solicita la verificación de la cuenta.

El módulo de Store Front cuando genera una compra debe solicitar la aprobación de la compra, de igual forma cuando dicha compra desea ser cancelada o anulada es a este módulo que se hace el pedido. De forma similar el módulo de Licitaciones y Subastas hace estos requerimientos cuando una subasta o licitación termina en una compra.

Cuando la interacción es iniciada por el módulo de Pago de Transacciones, el mismo interactúa con el módulo de Administración directamente a través del *bean* de entidad *EmpresaBean* que tiene una relación con el *bean* de

entidad *BancoBean,* mientras que la interacción con el módulo de Store Front se da por medio del *bean CompraBean*.

Un punto importante de interacción de este módulo se da con los sistemas bancarios externos ya sea a través de mensajería JMS o con documentos XML.

CAPÍTULO 5

5. Pruebas

Luego de la elaboración del sistema, se procede a preparar una batería de casos de prueba. Esto con el fin de asegurarnos que el sistema hace lo que debería hacer.

Normalmente los casos de prueba pueden ser escritos en papel y ejecutados de forma manual. Sin embargo esto sólo es viable para sistemas pequeños. Por lo tanto, considerando que estos módulos son de tamaño considerable, las pruebas fueron realizadas sobre pedazos de código con ayuda de un conjunto de herramientas que permite la automatización de las mismas.

5.1. Plan de pruebas

Antes de iniciar con la descripción de las pruebas se indicará lo que son pruebas de unidad. “Las pruebas de unidad son las primeras y más críticas, implican tomar unidades de código y probar todo lo que podría salir mal. Para asegurarse que un sistema trabaja completamente las pruebas de unidad cooperan con otros tipos de pruebas: de integración, funcionales y otras auxiliares como de rendimiento y de regresión” [19].

De los tipos de pruebas enumerados se decidió implementar, junto con las de unidad, las de rendimiento, puesto que las últimas hacen uso de las primeras, con lo que se ahorra tiempo de codificación y además se asegura que se está probando la lógica de negocio básica del sistema.

5.1.1. Objetivos

Elaborar pruebas de unidad y rendimiento de las funciones más importantes del módulo de Administración, haciendo énfasis en la lógica de negocio principal del módulo así como la utilizada por los otros módulos del sistema, es decir, consultas de productos, búsqueda de empresas, búsqueda de usuarios, etc.

5.1.2. Herramientas utilizadas

Las herramientas utilizadas para la codificación de las pruebas fueron JUnit y JUnitPerf, mientras que para la ejecución se utilizó un *script* de Ant.

“JUnit es un *framework* de pruebas de regresión escrito por Erich Gamma y Kent Beck. Es usado por desarrolladores que implementan pruebas de unidad en Java” [17], las pruebas se ejecutaron con versión 3.8.1 de este *framework.*

Mientras que para las pruebas de rendimiento se implementaron y ejecutaron con JUnitPerf que “consiste de un pequeño conjunto de decoradores de pruebas y clases relacionadas que extienden el API JUnit” [17], es decir las pruebas JUnitPerf extienden una prueba JUnit usando el patrón decorador[7].

Para escribir las pruebas, o más precisamente las clases Java respectivas, se utilizan los APIs definidos por ambos *frameworks,* en tanto que para la ejecución de las mismas y generación de reportes de resultados se utiliza un *script* Ant, esto último permite la automatización del proceso.

Cada clase de prueba de unidad, en esencia, tiene una variable de instancia del servicio que va a ser sometido a pruebas, y a través del API verifica si los datos retornados por los métodos son iguales a los datos esperados o si existió algún error durante la invocación del método de servicio.

Finalmente, las pruebas de rendimiento hacen uso de las anteriores y se configuran para ser ejecutadas por un número específico de usuarios, además se puede establecer el tiempo transcurrido entre la activación de cada usuario, así como la cantidad de iteraciones que efectuará cada uno de ellos y por último el tiempo luego del cual debe indicarse que la prueba fue fallida.

---

[7] El patrón decorador es usado para agregar funcionalidad adicional a un objeto en particular en lugar de a una clase de objetos.

5.1.3. Definición

Se escribieron seis conjuntos de pruebas, o denominados *suites* en el contexto de JUnit, cada *suite* agrupa funcionalidad relacionada a una entidad del sistema. Además se escribió una *suite* que agrupa todas las pruebas de rendimiento.

El primer grupo de casos de prueba comprende solamente pruebas de unidad, en tanto que el segundo grupo, además de las pruebas de unidad, también incluye las pruebas de rendimiento.

Antes de describir cada prueba se debe tener en consideración, en cuanto a las pruebas de rendimiento, que las mismas corresponden a recuperación de datos, debido en parte a las limitaciones del *framework* y en parte a que se utiliza un mismo caso de prueba de forma simultánea para recrear varios usuarios concurrentes, lo que obliga a utilizar los mismos datos de verificación.

Otro punto a tener en cuenta es que se realizan las pruebas directamente a la capa del negocio (los EJBs) de los módulos de Administración y Pagos de Transacciones de E-Guana, que como se ha descrito en secciones anteriores, no es la única capa del sistema, pero si es la más importante debido a que comprende la funcionalidad básica del sistema y que sobre ésta

se sustentan las capas de presentación. En definitiva se puede concluir que estas pruebas verifican el funcionamiento del núcleo del sistema.

A continuación se detallarán las pruebas a las que fue sometido el sistema.

Pruebas de unidad

Las pruebas de unidad a las que fue sometido el sistema se describen a continuación.

Caso de prueba: Registro de una Empresa:

En este caso el resultado se mide en base a la adición de una nueva empresa en el conjunto de empresas registradas en el sistema, el *bean* de sesión sometido a prueba es *RegistradorBean*.

La prueba consiste en varios pasos, iniciando en la recepción de solicitud de registro, luego la búsqueda de la solicitud de registro y finalmente el procesamiento de dicha solicitud (Rechazo/Aprobación).

Caso de prueba: Solicitud de Aprobación de una transacción de compra:

Esta prueba consiste en la aprobación y finalización exitosa del proceso de compra, el código sometido a verificación es del *bean* de sesión *AdminPagoBean*.

Inicialmente se somete a prueba la recepción de la solicitud de aprobación de compra, luego se verifica que se pueda buscar dicha solicitud y cuyo estado sea “pendiente”, sigue la aprobación de la solicitud, de igual forma se comprueba que se pueda encontrar en el sistema la solicitud “aceptada”, finalmente se verifica que se pueda borrar la solicitud.

Caso de prueba: Mantenimiento de un Producto:

Este caso comprueba todas las etapas involucradas en el mantenimiento exitoso de los datos del producto a través del *bean* de sesión *AdminProductoBean*.

Inicialmente se crea un producto, luego se comprueba que efectivamente se lo pueda encontrar y por último se hace la prueba de borrado.

Pruebas de rendimiento

Las pruebas de rendimiento se detallarán a continuación.

Caso de prueba: Consultas generales

Este *suite* de pruebas agrupa la funcionalidad expuesta por el *bean* de sesión *AdministradorBean*, más específicamente, como se indicó previamente, la funcionalidad de consulta del mismo.

Las búsquedas o consultas probadas, en orden de ejecución, incluyen la de empresas, tanto por ID, estado, nombre y RUC, igualmente la de solicitudes de cupo, la de unidades de negocio, tanto por usuario y nombre, y por último la de usuarios por empresa.

Caso de prueba: Consultas de Categoría de Productos

Este caso de prueba comprende los servicios de consulta expuestas por el *bean* de sesión *AdminCategoriaBean.*

Las consultas sometidas a prueba incluyen a la de solicitudes de creación de categorías pendientes, de categorías de productos por nombre y de solicitudes de creación de categorías de productos por empresa solicitante, tanto sin considerar el estado como en estado “Pendiente”.

Caso de prueba: Consultas de Productos

En este caso de prueba se verifican las búsquedas proporcionadas por el *bean* de sesión *AdminProductoBean.*

Los servicios de búsqueda probados comprenden todos aquellos que permiten encontrar uno o más productos de acuerdo a varios criterios, incluidos el identificador de categoría a la que pertenece, el nombre de la categoría, la descripción del producto, la marca, así como la empresa que lo tiene en su catálogo.

5.1.4. Ejecución

El ambiente de ejecución de las pruebas se resume en las características del sistema en las que fueron ejecutadas, éstas fueron:

- Procesador AMD Sempron 1.4 GHz
- Memoria 480 MB
- Windows XP Professional Service Pack 2
- Java Development Kit 1.4

Es necesario agregar y considerar que en la misma máquina en la que se realizaron las pruebas, también funcionaban el servidor de aplicaciones (JBoss), el contenedor Web (Tomcat 5) y la base de datos (MySQL).

Una parte importante de la ejecución de las pruebas la constituyen los datos de prueba, ya que con estos se puede verificar si la misma fue o no exitosa, a continuación se los detallará para cada caso.

Pruebas de unidad

Para este tipo de pruebas, los datos de prueba corresponden normalmente a los atributos de la entidad que está siendo sometida a verificación.

Caso de prueba: Registro de una Empresa:

Los datos de prueba de este caso corresponden a los datos que deben ser ingresados en el formulario de registro de la empresa, y son los siguientes:

- RUC: “0000000000000”
- Email: “email@prueba.org”
- Descripción: “Descripcion Prueba”
- Teléfono: “555555555”
- Número de cuenta: 1
- Nombre de usuario: “Usuario Prueba”
- Apellido de usuario: “Apellido Prueba”
- Clave: “prueba”
- Usuario: “prueba”
- ID de ciudad: 1
- ID de banco: 1

Cada uno de los datos listados se maneja como atributos del DTO *SolicitudDTO,* ya que es dicha clase la que se utiliza como soporte para la funcionalidad del registro de la empresa.

Caso de prueba: Solicitud de Aprobación de una transacción de compra:

En este caso los datos de prueba se refieren tanto a la compra que quiere ser aprobada así como al banco que debe aprobarla:

- ID de la compra: 389
- ID del banco: 5

Ambos datos en este punto, corresponden a claves primarias de registros que existen en el sistema.

Caso de prueba: Mantenimiento de Producto:

Para el mantenimiento de productos se necesitan tanto los datos del producto en sí, así como el identificador de la categoría en la que se pondrá al producto, por lo tanto los mismos son:

- ID de categoría: 1
- Marca: “Marca Prueba”
- Comentario: “Comentario Prueba”
- Descripción: “Descripción Prueba”

El ID de categoría corresponde a la categoría denominada “Computadoras”.

Pruebas de rendimiento

Para las pruebas de rendimiento se deben considerar, adicional a los datos de prueba propiamente, los tiempos de espera aceptables, el número de usuarios concurrentes, el tiempo entre activación de usuarios y el número de

veces que cada usuario invoca los métodos de prueba, estos últimos parámetros corresponden a los de la prueba de rendimiento en sí.

Los parámetros de rendimiento se configuraron de la siguiente forma: 50 usuarios concurrentes, cada nuevo usuario se activa cada medio segundo y cada usuario ejecutará el conjunto de pruebas una sola vez. Para los tiempos de espera se hizo una distinción y se decidió configurarlos proporcionalmente a la cantidad de funciones sometidas a verificación, ya que no es lógico esperar los mismos tiempos de respuesta entre pruebas de distinta extensión y complejidad.

Caso de prueba: Consultas generales

Este conjunto de pruebas comprende 8 consultas, por ello se estableció un tiempo de espera máximo de 2.5 segundos por usuario, es decir si los 8 requerimientos no se completan en 2.5 segundos, la prueba habrá fallado.

Para las consultas se usaron los siguientes datos:

- ID de la empresa: 2
- Criterio por nombre de la empresa: "%MICROSOFT%"
- Criterio por RUC: “%1234567890%”
- Criterio por nombre de usuario: “wramos”
- Criterio por nombre de la unidad: “%Comandato%”

Como es de esperarse los datos para las consultas retornan al menos un registro.

Caso de prueba: Consultas de Categoría de Productos

Las pruebas a consultas de categorías de productos son sólo 4 por esto se estableció como máximo un segundo en el tiempo de espera.

En cuanto a las consultas, los datos utilizados fueron:

- Nombre de la categoría de productos: “Microondas”
- ID de la empresa: 2

El último dato de prueba se utiliza para las búsquedas de solicitudes de categorías de productos hechas por una empresa.

Caso de prueba: Consultas de Productos

Este conjunto de pruebas agrupa 8 funciones por lo que se configuró un tiempo de espera de 2.5 segundos.

Para las búsquedas de productos se utilizaron los siguientes datos:

- Criterio para la marca: “%SONY%”
- Criterio para la categoría: “%Televisores%”
- Criterio para la descripción: “%Televisor%”

- ID de la categoría en la que se buscarán productos: 13
- ID de la empresa en cuyo catálogo se buscarán productos: 2

El identificador de categorías “13” corresponde a la categoría denominada “Televisores”, por lo que se tendrán resultados similares a la búsqueda de productos por criterio.

5.2. Resultados

JUnit fue configurado para que los resultados de las pruebas se presenten en archivos HTML. Estos documentos de resultados indican el número de pruebas ejecutadas (Tests), la cantidad de fallas ocurridas (Failures), el número de errores (Errors), el porcentaje de éxito (Success Rate) y el tiempo que tomó la prueba. “Las fallas son anticipadas y verificadas, mientras que los errores no son anticipados” [17].

A continuación se presentarán los resultados de las pruebas ejecutadas.

Pruebas de unidad

Los resultados de las pruebas de unidad, como se verá a continuación, no indican fallas ni errores debido a que verifican la validez del código sin someterlo a carga. Sin embargo su utilidad se evidencia en etapas de reescritura de código.

Caso de prueba: Registro de una Empresa:

Los resultados de la prueba de registro de una empresa se resumen en las siguientes tablas:

Tests	Failures	Errors	Success rate	Time
3	0	0	100.00%	44.547

Name	Status	Type	Time(s)
testReceptarSolicitud	Success		42.000
testBuscarSolicitud	Success		1.657
testRechazarSolicitud	Success		0.687

Tabla 5-1 Resultados caso de prueba: Registro de una Empresa

Como se puede observar las 3 pruebas se ejecutaron exitosamente en 44.5 segundos.

Caso de prueba: Solicitud de aprobación de una transacción de compra:

Las tablas que se presentan a continuación recogen los resultados de las pruebas:

Tests	Failures	Errors	Success rate	Time
5	0	0	100.00%	11.066

Name	Status	Type	Time(s)
testSolicitarAprobacionCompra	Success		2.283

testBuscarAprobacionCompraPendiente	Success		2.003
testRespuestaAprobacionCompra	Success		1.873
testBuscarAprobacionCompraAceptada	Success		1.532
testBorrarSolicitudAprobacionCompra	Success		3.325

Tabla 5-2 Resultados caso de prueba: Solicitud de aprobación de una transacción de compra

Las 5 pruebas que constituyen este caso de prueba resultaron exitosas y tardaron en total 11 segundos,

Caso de prueba: Mantenimiento de Producto:

Los resultados de la prueba de mantenimiento de producto se presentan a continuación:

Tests	Failures	Errors	Success rate	Time
3	0	0	100.00%	6.409

Name	Status	Type	Time(s)
testCrearProducto	Success		5.798
testBuscarProducto	Success		0.150
testBorrarProducto	Success		0.421

Tabla 5-3 Resultados caso de prueba: Mantenimiento de producto

Es necesario indicar que el *framework* de prueba JUnit, como toda aplicación Java, tiene considerables tiempos de inicialización, por esta razón la notoria diferencia de tiempos entre algunas pruebas.

Pruebas de rendimiento

Los resultados de las prueba de rendimiento se presentarán a continuación, debido a la extensión de los mismos, tan sólo se presentarán los datos resumidos junto con las fallas y errores.

Caso de prueba: Consultas generales

Los resultados de este caso de prueba son:

Name	Tests	Errors	Failures	Time(s)
ConsultasPrueba	400	1	7	25.625

Tabla 5-4 Resultados caso de prueba: Consultas Generales

Las 400 pruebas corresponden a los 50 usuarios concurrentes configurados ejecutando cada uno 8 pruebas de consulta.

El error indicado en los resultados, corresponde a la prueba “testBuscarEmpresasPorRUC” debido a una excepción de valor nulo, es decir el valor retornado fue nulo cuando debería haber sido una lista con los datos de las empresas.

En cuanto a las fallas, 6 de ellas corresponden a tiempos de espera excedidos, es decir, la prueba tomó más de 2.5 segundos. La última falla se debió a que el método “testBuscarUnidadDeUsuario” no pudo retornar la unidad de negocio a la que pertenece el usuario identificado como “wramos”.

Caso de prueba: Consultas de Categoría de Productos

Las pruebas de consultas de categoría de productos dieron los siguientes resultados:

Name	Tests	Errors	Failures	Time(s)
ConsultasCategoriaPrueba	200	0	0	25.000

Tabla 5-5 Resultados caso de prueba: Consultas de Categoría de Productos

A diferencia de la prueba anterior este *suite* no dio fallas ni errores, un resultado esperado, considerando la menor complejidad de estas pruebas en relación a las otras.

Caso de prueba: Consultas de Productos

Los resultados de este *suite* de pruebas se resumen a continuación:

Name	Tests	Errors	Failures	Time(s)
ConsultasProductoPrueba	**400**	**0**	**15**	**25.125**

Tabla 5-6 Resultados caso de prueba: Consultas de Productos

De las 400 pruebas ejecutadas, 15 de ellas fallaron, de éstas, 14 no pudieron recuperar datos durante su ejecución, mientras que la restante se excedió en el tiempo de espera.

Finalmente se presentan los resultados de la prueba de rendimiento que reúne a las tres anteriores:

Name	Tests	Errors	Failures	Time(s)
AllLoadTests	1000	0	1	75.078

Tabla 5-7 Resultados para todas las pruebas de rendimiento

La única falla resultante de estas pruebas ocurrió en el método que busca la unidad de negocio del usuario de prueba, al no poder encontrar el registro esperado.

El resumen del total de pruebas de rendimiento es el siguiente:

Tests	Failures	Errors	Success rate	Time
2000	23	1	98.80%	150.828

Tabla 5-8 Resumen de resultados de pruebas de rendimiento

Las pruebas de rendimiento, como se dijo anteriormente, se configuraron de tal forma que cada medio segundo un nuevo usuario se creó y ejecutó las consultas, por lo tanto luego de 25 segundos aproximadamente todos los usuarios estaban haciendo uso del sistema o acababan de usarlo, además cada usuario debía completar las pruebas en no más de 2.5 segundos, según los resultados, todos los *suite* de prueba terminaron en alrededor de 25 segundos por lo que se utilizó menor tiempo que lo establecido.

Respecto a las fallas, antes hay que recalcar lo indicado en la parte de ejecución de este capítulo, esto es, en una misma máquina funcionaban todas las capas constituyentes de E-Guana así como el ambiente de las pruebas. Considerando lo anterior y los resultados obtenidos, se pueden

sugerir varias alternativas para asegurar el correcto funcionamiento de E-Guana. La primera es tener servidores dedicados para cada capa del sistema, así una máquina prestaría el servicio de base de datos, otra administraría el servidor de aplicaciones y una tercera proveería el contenedor Web. Si no se cuentan con los recursos necesarios para soportar los tres servidores especializados, la otra opción es un solo servidor con superiores características de hardware, al menos a las de la máquina utilizada en las pruebas.

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

En un mundo competitivo donde la principal ventaja es el uso de la tecnología, es necesario recalcar sin lugar a dudas el desarrollo de los sistemas E-Procurement. A pesar de que no existen muchos sistemas desarrollados para tal efecto y los que existen presentan muy altos precios, el retorno de inversión que estos generan y la amplitud del negocio en las empresas que lo poseen, colocan a éstas muy por encima de sus competencias.

El E-Procurement conlleva algo más que tecnología, habla de la transformación de los negocios, ofrece beneficios a muchas áreas e involucra prácticamente a todos los miembros y componentes de una empresa.

El uso de herramientas de código abierto y la aplicación de las ventajas de E-Procurement ofrecen indudablemente un alto valor agregado al desarrollo de sistemas como E-Guana.

Las aplicaciones E-Procurement permiten un mayor control y visibilidad de la adquisición para el negocio. Al mismo tiempo, todo el proceso de adquisición, tanto interna como externamente, será más funcional, el proceso requerirá

menos pasos y ofrecerá mayor comodidad, especialmente en la aprobación automática de compras y en los trámites de transacciones. Consecuentemente, los beneficios económicos se presentan poco tiempo después de su implementación, adecuándose a una cultura de producción justo a tiempo y de aprovechamiento de espacio para fines productivos y no de almacenamiento. La comodidad de este proceso de compra-venta será lo que anime a los usuarios a adoptarlo. Esta comodidad se basa en ofrecer a los usuarios un solo punto de entrada al sistema de adquisición donde se puede dar seguimiento a las transacciones. Si se puede convencer a los usuarios que este sistema les facilitará el trabajo, sin duda lo usarán, ofreciéndole el conocimiento y la visibilidad que necesitan para seguir adelante.

Con respecto a la tecnología utilizada se puede concluir, luego de la implementación de este proyecto, que el uso EJBs trae consigo beneficios así como desafíos, por lo que la decisión de su uso en el desarrollo de un sistema requiere un profundo análisis por parte de los involucrados en el mismo.

Aunque se debe reconocer que no es una tarea sencilla iniciarse en el uso de EJBs, su adaptación permite construir aplicaciones confiables que facilitan al desarrollador centrarse en la solución del problema del negocio y no en tareas, aunque necesarias, son repetitivas y comunes a toda aplicación

empresarial; como manejo de *pool* de instancias, transacciones, seguridad, persistencia y relaciones manejadas por el contenedor y cacheo de datos; ya que existe un servidor de aplicaciones que cumple dichas tareas de un forma seguramente superior a que si se lo tuviera que hacer desde cero por el grupo de desarrollo del proyecto.

La utilización del paradigma Modelo Vista Controlador se da de forma natural en J2EE, al manejar el modelo con los *beans* de entidad, el control con los *beans* de sesión y la vista con páginas JSP.

El uso de herramientas para la generación de las interfaces y de los descriptores requeridos por la especificación EJB, como Xdoclet se vuelve indispensable, ya que su generación manual, a parte del tiempo requerido, se pude convertir en una tarea frustrante, que a la vez distrae al desarrollador de programar la lógica del negocio, que es uno de los objetivos de EJB.

RECOMENDACIONES

El sistema E-Guana se creó con la finalidad de proveer las ventajas que ofrece la tecnología E-Procurement. Para que el sistema sea comercial se sugiere que la función de Banco emulada por el Módulo de Pagos transaccionales (Descrito en el Anexo 1), sea un servicio que provean los Bancos reales que existan en el mercado los cuales van a ser parte del servicio de E-Guana. El proceso de débito/acreditación de los valores resultantes de las compras, será exclusivo del sistema que maneje cada banco, así como los tiempos de disponibilidad de los valores. Este sistema de pagos puede ser explotado dentro del Ecuador en el caso de que sea un sistema local.

Actualmente el sistema de pagos de E-Guana está basado en un esquema de débito bancario, se sugiere se adicione el esquema de pagos mediante tarjeta de crédito así el alcance del mercado se vuelve internacional. Para este caso, se requiere que el sistema genere y envíe los datos de las tarjetas por medio de una librería (proporcionada por la compañía que provee el servicio) y luego recepte e interprete la respuesta.

El sistema E-Guana se podría conectar con los portales que proveen servicio de rastreo de paquetes, así el usuario tendrá pleno conocimiento de todo el proceso de transporte del paquete desde el origen hasta su destino final

GLOSARIO

API:

Siglas en inglés de Application Programmable Interface. Es un conjunto de especificaciones de comunicación entre componentes de software.

AWT:

Siglas en inglés de Abstract Windows Toolkit. Toolkit de Java.

B2B:

Es la forma contemporánea de llamar a la práctica de ventas denominada (business-to-business) negocio-a-negocio. Las transacciones B2B tienen como objetivo a compañías y compradores mayoristas.

Contexto del *bean*:

Es la vía de comunicación entre en Bean y el Contenedor. A través del contexto se puede recuperar información del contendor EJB, por ejemplo: credenciales, de seguridad, modificar su estado, manejar transacciones, etc.

CTM:

Siglas del Component Transaction Monitor. Es un servidor de aplicaciones que usa un modelo de componente de lado del servidor.

DTD:

Siglas de Data Type Definition. Un archivo de texto que define la estructura válida de un archivo XML.

DTO:

Siglas de Data Transfer Object. Objeto de Transferencia de Datos. Se los utiliza para enviar los datos de uno o más EJB a las aplicaciones clientes que no pueden o no deben tener acceso a los EJB directamente.

EJB QL:

Iniciales de Enterprise JavaBeans Query Language lenguaje para definir las consultas para los métodos *finder* y *select* de un b*ean* de entidad que usa persistencia manejada por el contenedor.

E-Commerce:

Intercambio de bienes y servicios en el cual todo o una parte del mismo se realiza por medios electrónicos.

E-Marketplaces:

Espacios globales en la Web que brindan la posibilidad de relacionar un gran número de empresas, clientes y proveedores, en un sólo punto de encuentro.

Framework:

Es un marco en donde las aplicaciones corren. Las aplicaciones ya no corren directamente bajo el sistema operativo si no que corren bajo este armazón o marco.

Front-end:

Parte de la aplicación que interactúa con el usuario.

Hashtable:

Una clase del JDK que mapea claves a valores. Cualquier objeto no nulo puede ser utilizado como clave o como valor.

HTTPS:

Hypertext Transfer Protocol SSL. Protocolo de transferencia de hipertexto SSL.

JAR:

Archivo Java. Un formato de archivo independiente de la plataforma que permite agregar muchos archivos en uno solo.

JBOSS QL:

Es un súper conjunto de EJB QL.

JNDI:

La Interfaz de Nombramiento y Directorio de Java (JNDI en Inglés) provee funcionalidad de nombramiento y directorio. Le provee a las aplicaciones con métodos para realizar operaciones estándar de directorio, tales como asociación de atributos con objetos y búsqueda de objetos usando sus atributos. Usando JNDI, una aplicación J2EE puede almacenar y recuperar cualquier tipo de objeto Java nombrado. (The J2EE Tutorial).

JMS:

Una API para invocar operaciones en sistemas de mensajería empresarial.

JSTL:

Corresponde a las iniciales de Java Server Pages Standard Tag Library (Librería de Etiquetas Estándar de JSP). Esta librería encapsula funcionalidad básica común a muchas aplicaciones JSP. JSTL tiene soporte para tareas comunes estructurales tales como iteración y condicionales, etiquetas para manipulación de documentos XML, internacionalización y etiquetas de formateo especifico a la localidad, etiquetas SQL y funciones.

Licitación:

Procedimiento administrativo que consiste en una invitación a contratar de acuerdo a bases previamente establecidas con la finalidad de obtener la oferta más beneficiosa.

Método *select*:

Usa consultas EJB QL para retornar objetos e información de estado de *beans* de entidad usando persistencia manejada por el contenedor.

La firma del un método *select* siempre inicia con ejbSelect, el modificador de control de acceso debe ser *public*, debe ser declarado abstracto y la cláusula *throws* debe incluir la excepción *javax.ejb.FinderException*.

Persistencia manejada por el contenedor:

El contenedor maneja todos los accesos a la base de datos requeridos por el *bean* de entidad en lugar de ser el programador quien escriba métodos y consultas para estos accesos.

RMI/IIOP:

Protocolo estándar de los servidores J2EE.

RSS:

Really Simple Syndication. Su principal uso es el de la distribución de noticias en el Web de forma periódica. Toma su nombre del hecho que usa un archivo XML básico para proveer una lista de titulares, descripciones y enlaces y ubica este archivo XML en sitio Web donde puede ser accedido por otros sitios Web o lectores RSS dedicados.

Script:

Programa o una secuencia de instrucciones que es interpretado y llevado a cabo por otro programa en lugar de ser procesado por el procesador de la computadora.

Socket:

Objeto de software utilizado por un cliente para conectarse a un servidor; los componentes básicos incluyen el número de puerto y la dirección de red del host local.

SSL:

Secure Socket Layer. Capa de Socket Seguro. Es una tecnología que permite a los navegadores y servidores Web comunicarse sobre una conexión segura. En esta conexión los datos son encriptados previo al envío y desencriptados en el receptor.

Subasta:

Venta pública de un bien o servicio al mejor postor.

TLD:

Siglas en inglés de Tag Library Descriptor. Descriptor de librería de etiquetas. Un documento XML que contiene información de toda una librería y de cada etiqueta contenida en ésta. Los TLDs son utilizados por el contenedor Web para validar las etiquetas.

URI:

Uniform resource identifier. Un identificador globalmente único para un recurso abstracto o físico.

WAR:

Archivo de aplicación Web. Un archivo JAR que contiene un módulo Web.

XML:

Extensible Markup Language. Lenguaje de marcas extensible. Lenguaje de marcas que permite definir las etiquetas (marcas) necesarias para identificar el contenido, los datos y el texto en documentos XML.

ANEXOS

Anexo 1: Aplicación Web bancaria

Para la demostración de la interacción del sistema con una aplicación bancaria se implementó esta aplicación Web, ésta se comunica con el módulo de Pagos de Transacciones a través de mensajería.

Se la implementó utilizando J2EE y DBForms (una librería de etiquetas personalizadas que presenta y mantiene los datos a través de acceso JDBC).

La aplicación se encargará de responder a las solicitudes de verificación de cuentas así como de aprobación y anulación de órdenes de compra. Estas respuestas serán enviadas directamente a la cola de mensajes destinada para ese objetivo en E-Guana.

Pantallas de la aplicación

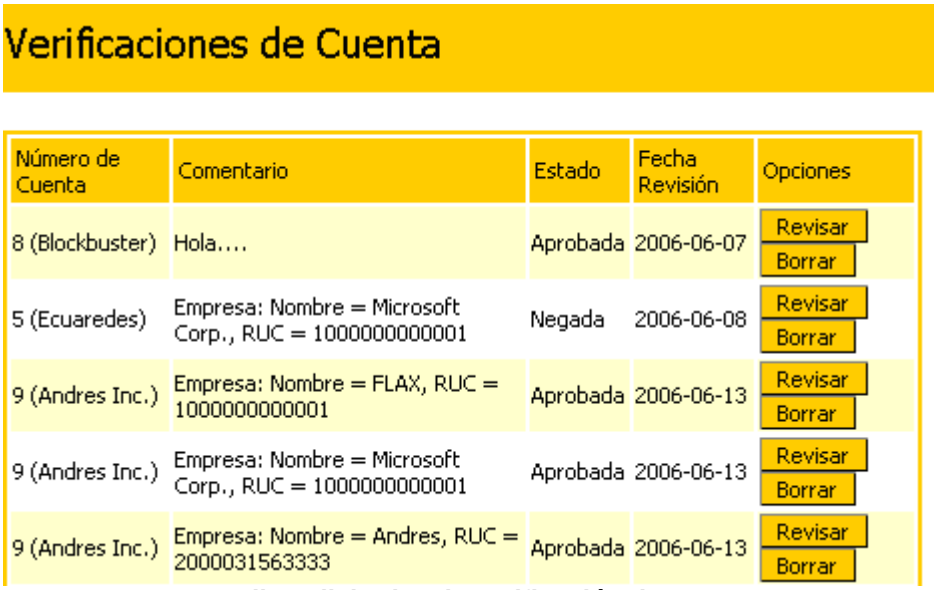
Pantalla cuentas registradas en el banco

La figura corresponde a la página para listar las cuentas bancarias que existen en el sistema, se muestra el RUC y nombre de la empresa, así como la fecha de expiración y saldo de la cuenta.

El banquero puede editar o borrar una cuenta, si decide editarla se presenta la página representada por el siguiente gráfico:

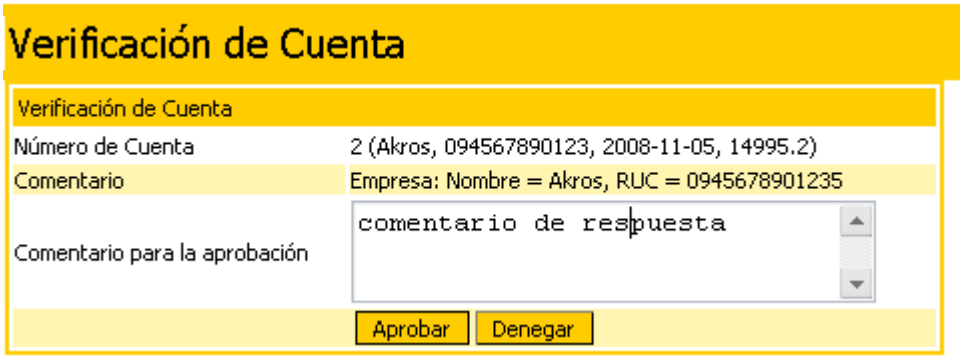
Pantalla detalle de una cuenta

Para revisar las verificaciones de cuentas que le han sido dirigidas, se tiene la página representada por la siguiente figura:



Pantalla solicitudes de verificación de cuenta

Entre los datos que se muestran están todos los enviados por E-Guana, además se presenta el nombre de la empresa que corresponde al número de cuenta envidado desde el módulo de Pago de Transacciones, así como el estado de la solicitud.



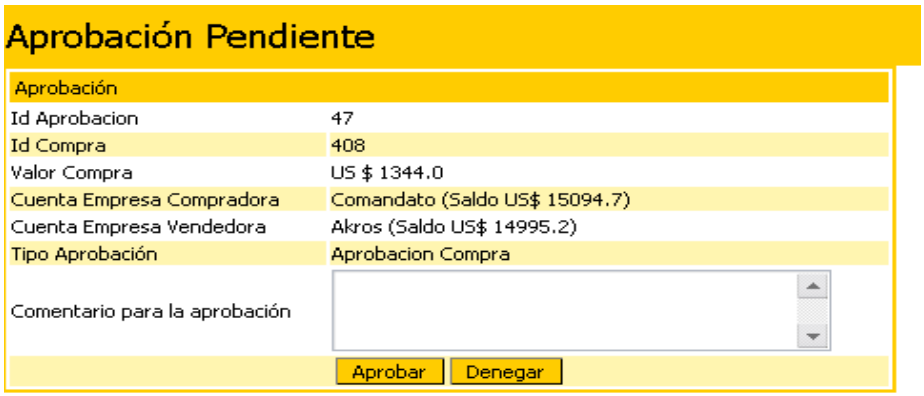
Pantalla aprobación de verificación de cuenta

La figura anterior corresponde al formulario para aprobar o negar la solicitud de verificación de cuenta.

Pantalla solicitudes de aprobación/anulación de compra

El listado de las solicitudes de aprobaciones y anulaciones de compras se presenta en una página similar a la presentada en el gráfico previo. Se muestra el valor de la compra involucrada, el tipo de solicitud y tanto el nombre de la empresa compradora como de la empresa vendedora, estos dos últimos datos siempre y cuando se puedan resolver los valores de los números de cuentas enviados por E-Guana.

El formulario para la aprobación o negación de la compra se grafica a continuación:



Pantalla formulario para aprobación de solicitud de compra

Finalmente el banquero puede listar las solicitudes que ha revisado a lo largo del uso del sistema:

Pantalla aprobaciones revisadas

Anexo 2: Aplicación Java de mantenimiento sencillo

Esta aplicación demuestra el uso de los mismos *beans* de sesión utilizados en la aplicación Web para el mantenimiento del sistema como un servicio que puede ser accedido fuera del servidor de aplicaciones.

Las entidades que pueden ser mantenidas con esta aplicación son: empresa, usuario, ciudades, roles, permisos y variables.

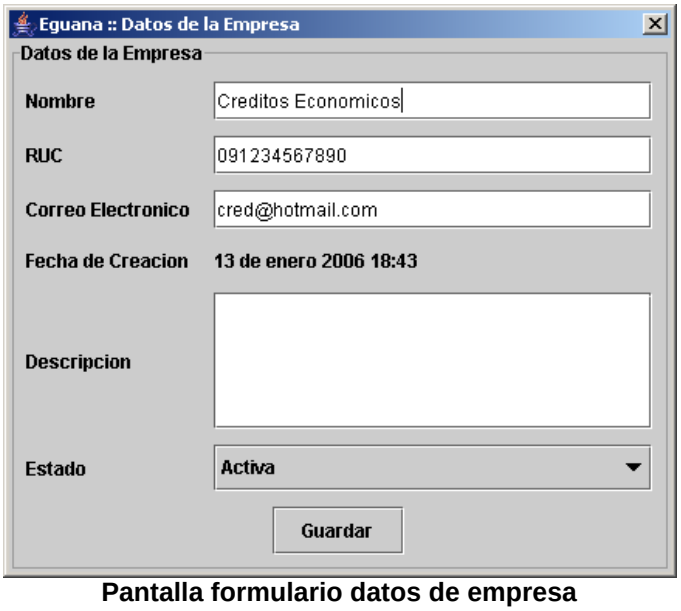
Por tratarse de un mantenimiento sencillo, los datos que pueden ser modificados son los atributos propios de la entidad pero no las relaciones de los mismos con otras entidades.

Como se indicó en el párrafo inicial esta aplicación hace uso de *beans* de sesión y lógicamente lo hace a través de las interfaces remotas de dichos *beans.* Para cada entidad se utilizó el *bean* de sesión respectivo para su mantenimiento según lo descrito en la sección 3.3 Estructura de este documento.

Pantallas de la aplicación

Pantalla listados de datos de una entidad

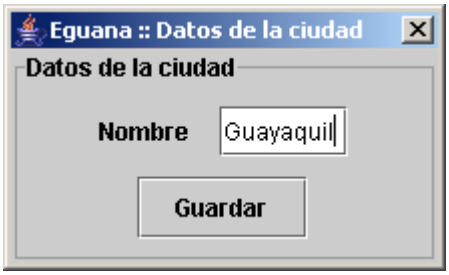
La figura corresponde al listado de los datos de la entidad seleccionada, en este caso los datos de las empresas registradas en E-Guana, haciendo doble clic sobre alguna de las filas de datos se muestra el formulario para la edición de los mismos.



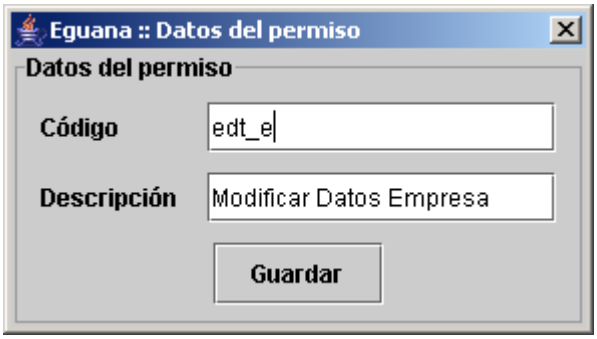
Pantalla formulario datos de empresa

Pantalla formulario datos usuario

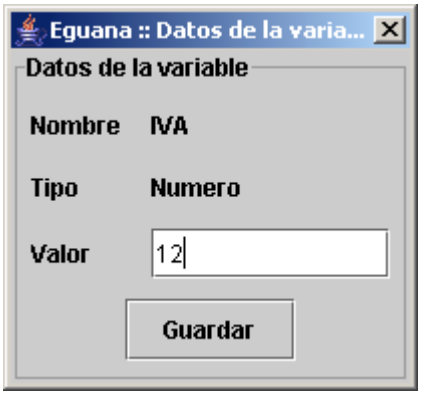
Pantalla formulario datos rol



Pantalla formulario datos ciudad



Pantalla formulario datos permiso



Pantalla formulario datos variable

BIBLIOGRAFÍA

1. R. Elsenpeter, T. Velte, Fundamentos de Comercio Electrónico, McGraw-Hill Interamericana S.A., 2001

2. Conferencia la Gestión de Compras a través del Internet realizada en Madrid el 14-15 Marzo del 2001 – temas tratados y breves de la conferencia.

3. José Ochoa, E-Procurement: Una herramienta para lograr la integración y colaboración en la relación proveedor-cliente, Agosto 2006, http://www.gestiopolis.com/canales2/gerencia/1/eprocur.htm

4. Jocildo Figueiredo Correira Neto, Master en Administración de Empresas, especialista en Administración Financiera, E-Procurement, 2003.

5. Patricia B. Seybold, Ronni T. Marshak, The Customer Revolution. Crown Business, 2001, p. 60-61.

6. Index, Soluciones IP, Diseño Profesional de Páginas Web, 2006, http://www.index.com.mx/services_dis.html

7. Cientec, La comunidad del E-Procurement, Marzo 2006, http://www.cientec.com/analisis/e-procurement.asp

8. José Camilo Daccach T., Transacciones Electrónicas, Marzo 2006 http://www.deltaasesores.com/prof/PRO200.html

9. Glen McCallum, Construir Aplicaciones EJB con Jboss, Lomboz y Eclipse, Servlets Básico, Marzo 2006, http://www.programacion.com/java/tutorial/

10. Sun Microsystems, Core J2EE Pattern Catalog, Noviembre 2005. http://java.sun.com/blueprints/corej2eepatterns/Patterns/index.html

11. Floyd Marinescu, EJB Design Patterns: Advanced Patterns, Processes, and Idioms. Wiley; Bk&Poster edition, Febrero 2002

12. Salmon LLC, SOFIA User Guide, Octubre 5 2002. http://www.salmonllc.com/website/Jsp/vanity/bin/UserGuide.pdf

13. Eric Armstrong, Jennifer Ball, Stephanie Bodoff, Debbie Bode Carson, Ian Evans, Dale Green, Kim Haase, Eric Jendrock, Junio 17 2004, The J2EE Tutorial, http://java.sun.com/j2ee/1.4/docs/tutorial/doc/

14. Ed Roman, Scott Ambler, Tyler Jewell, Ed Roman, Tyler Jewell, Floyd Marinescu, Mastering Enterprise JavaBeans Second Edition, Wiley; 2nd edition, Diciembre 14, 2001

15. Scott Stark y The JBoss Group, JBoss Administration and Development Second Edition, Septiembre 30 2002.

16. American Express, Alianzas en e-Procurement, Marzo 2006, http://corp.americanexpress.com/gcs/intl/spain/corporatecards/corporatepurchasingsolutions/facturacionypagos.aspx

17. JUnit.org, Plan de Pruebas, Julio 2006, http://www.junit.org

18. Bruce Eckel, Thinking in Java 3rd Edition Revision 4.0, Prentice Hall PTR; 3 edition, Diciembre 6, 2002

19. Richard Hightower & Nicolas Lesiecki, Java Tools for Extreme Programming, John Wiley & Sons Inc., 2002.

20. Alfonso Cubero Moral y Sergio Luna García, Servlets y JSP, Mayo 2003, http://tejo.usal.es/~fgarcia/docencia/poo/02-03/trabajos/S2T3.pdf

21. Java Users Group Argentina, Introducción a la Programación de Java Servlets , Noviembre 2003,

http://cricava.com/java/introducci_oacute_n_a_la_programaci_oacute_n_de_java_servlets

22. E-Business Consultores, Noviembre 2006, http://www.ebconsultores.com.mx

23. CECARM.com, Comercio Electrónico en la región de Murcia, Diciembre 2006, http://www.cecarm.com/cecarm/

24. YLOS.com Estadísticas Ventas y Usuarios Internet en 2006, Diciembre 2006, http://www.ylos.com/spa/item/vendereninternet2.html