ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

TÓPICO DE GRADUACIÓN

"Módulo de Licitación y Subastas de E-guana".

Previa a la obtención del Título de:

INGENIERO EN COMPUTACIÓN ESPECIALIZACIÓN

SISTEMAS DE INFORMACIÓN Y

SISTEMAS TECNOLÓGICOS

Presentada por:

Roberto Carlos Guerrero Navarrete

José Javier Intriago Acuña

Christian Breznhev Tacle Lemos

GUAYAQUIL – ECUADOR

Año 2007

A G R A D E C I M I E N T O

A Dios, nuestros padres, nuestros profesores de tópico y en especial a los compañeros de la comunidad Open Source alrededor del mundo que comparten sus experiencias en la red de manera desinteresada.

D E D I C A T O R I A

A mi Padre celestial, a mis padres terrenales, a mis hermanos y amigos.

Y sobretodo gracias Padre del cielo por lo que esta escrito en Juan 3:16.

Roberto Guerrero N.

D E D I C A T O R I A

A mi familia.

José Intriago

D E D I C A T O R I A

A mis padres

A mis abuelos

A mis hermanos

A mis amigos.

Christian Tacle

TRIBUNAL DE GRADUACIÓN

Ing. Holger Cevallos	Ing. Luis Muñoz
SUB-DECANO DE LA FIEC	DIRECTOR DE TÓPICO
Ing. Otilia Alejandro	Ing. Ana Tapia
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"

Roberto Guerrero

José Intriago

Christian Tacle

RESUMEN

E-guana es un sistema de ventas, compras y subastas desarrollado con tecnología J2EE y con herramientas OpenSource. E-guana, como E-procurement, es modular permitiendo un mejor entendimiento del negocio y reduce el tiempo de respuesta al realizar su mantenimiento, como aplicación web tiene un alcance desde una intranet local o extranet hasta un sitio público web.

Con E-guana el registro de ofertas a licitación es rápido y sencillo, establece una comunicación con las entidades bancarias para realizar pagos automatizados a los proveedores, crea catálogos electrónicos, consolida licitaciones y organiza concesiones de pedidos, y crea una relación comprador/vendedor en las transacciones.

E-guana es un proyecto de tópico diseñado por estudiantes de la FIEC como previo a la obtención del título de Ingeniero en Computación especialización Sistemas de Información y Sistemas Tecnológicos. El proyecto se ha dividido en 4 módulos que son: Administración; Store Front; Licitación & Subastas y de Reportes.

Este texto explica el análisis utilizado para el diseño e implementación del módulo de Licitación y Subastas de E-guana.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Abreviaturas

CAE:	Corporación aduanera ecuatoriana
CMS:	Sistemas de manejo de contenido
CONATEL:	Consejo Nacional de Telecomunicaciones
CORPECE:	Corporación Ecuatoriana de Comercio Electrónico
CPU:	Unidad Central de Procesamiento
EIS:	Sistema de informaciòn empresarial
EJB:	Enterprise Java Beans
ERP:	Sistemas de planificación de recursos empresariales
IDE:	Ambiente integrado de desarrollo
IESS:	Instituto Ecuatoriano de Seguridad Social
J2EE:	Java 2 Enterprises Edition
MVC:	Modelo vista controlador
PC:	Computador personal
POJO:	Plain Old Java Object
SMS:	Mensajes cortos de texto
SRI:	Servicio de rentas internas
TIC:	Tecnologías de Información y Comunicaciones
XP:	Programación extrema

CAPÍTULO 1

1 SISTEMAS DE NEGOCIOS EN INTERNET.

1.1 Introducción

La tendencia del uso de Internet para hacer negocios en la actualidad se acrecentó en los últimos años. A diferencia del resto del mundo los negocios por Internet en el Ecuador, son relativamente escasos y los pocos que existen demandan muchos recursos para levantarlos tratándose de herramientas propietarias. Tomando como fundamental premisa se creó E-guana como una herramienta de compra/venta de licitaciones electrónica.

Para conseguir el éxito deseado dentro de los sistemas de negocios que existen actualmente en internet tomamos en cuenta lo que el consultor Eduardo Navarro, Socio Director Improven Consultores, establece dentro de las cinco oportunidades que ofrece Internet [7]:

- Mejoras en el área de marketing y comercial.
- Mejoras en la gestión de compras.
- Mejora de los procesos de la empresa.
- Aprovechamiento de nuevas oportunidades de negocio y nuevas ventajas competitivas.
- Mejora de la gestión de recursos humanos.

Estas posibilidades repercuten en los resultados empresariales:

- Incremento de ventas
- Disminución de costes
- Incremento de los márgenes de beneficio
- Fidelización de los clientes

Un punto importante del fracaso del uso de la Internet en los negocios es la formación de cómo utilizar Internet como una eficiente herramienta para hacer negocios, cuál es el modelo de negocio a seguir y cuáles son los procesos a nivel operacional que toma como fundamento el modelo. Esto es resultado de un débil conocimiento tanto del alcance del servicio como

del mismo negocio, ya que actualmente es necesario tener una visión amplia dentro del *e-business* o negocio electrónico.

José Ignacio López Sánchez coincide con Navarro al indicar “las empresas que quieran participar de las bondades de la Economía Digital deben definir su Modelo de Negocio, identificando su contenido, estructura y gobierno. Para ello debemos tener en cuenta la proposición de valor de la empresa, el segmento del mercado, la estructura de la cadena de valor, la estructura de costes y los beneficios potenciales y ampliar el sistema a clientes y proveedores” [8].

Así López Sanchez confirma lo que la litertura estratégica ha formulado, es decir, hay dos formas de ser una considerable competencia: una es haciendo lo mismo que nuestros competidores pero más eficientes; y hacer lo mismo con procedimientos totalmente diferentes de manera que demostremos una innovación ofreciendo un valor agregado para los clientes, y este ultimo es muy importante porque el valor agregado que llega de un negocio de internet es vital para que se mantenga con visitas y más aún con ventas.

E-guana así trata de fundir dentro de sus cimientos los fundamentos dictados por estos dos autores.

1.2 Objetivos y alcance del proyecto

El módulo de Licitación y Subastas de E-guana tiene como objetivos principales:

- Satisfacer las necesidades de la empresa y del proveedor, como comprador o subastador.
- Dinamizar procesos manuales como la licitación y la entrega de cotizaciones a la empresa.
- Permitir tomar una mejor decisión de compra comparando aspirantes por ofertas.

El alcance del proyecto E-guana es llegar a empresas que necesiten realizar una requisición de productos de manera automática y rápida, a proveedores que deseen registrarse en la cartera de compras y a usuarios que deseen adquirir productos en modo de subasta. Asi podemos describir el alcance total del sistema a través de los siguientes opciones:

Creación de Requerimiento: Registra el ingreso de un requerimiento nuevo de productos.

Consulta de Requerimiento: Consulta los requerimientos ingresados, su estado y fecha de caducidad.

Modificación de Requerimiento: Registra cambios realizados a un requerimiento de productos existente.

Creación de Licitación: Registra una licitación de uno o varios requerimientos ingresados por las unidades de negocio.

Consulta de Licitación: Consulta información sobre la licitación, sus productos, sus estados, número de ofertas y las ofertas realizadas.

Creación de Oferta: Registra una oferta para una licitación en proceso o una subasta.

Creación de Subasta: Registra datos para el ingreso de una subasta, en la cual se asigna un producto de nuestro inventario.

Estas opciones se explicarán a detalle en el capítulo 3, junto con los procedimientos, casos de uso y escenarios que se han contemplado para el desarrollo de E-guana.

1.3 La era de Negocios en Internet

1.3.1 Introducción

A principios de los noventas, con la aparición del World Wide Web, empezaron a surgir empresas multimillonarias de la noche a la mañana gracias a Internet; no obstante, con el pasar de los años, la gran mayoría de las mismas han desaparecido o tienen resultados negativos o muy pobres.

Una de las principales causas es la falta de conocimientos de negocios de los emprendedores que crearon estas compañías, pues consideraban que eran necesarios el conocimiento informático y la infraestructura. Algunos ofrecían servicios gratuitos esperanzados en las ventas por publicidad, ignorando la audiencia efectiva de la misma y su impacto sobre las ventas. Otra situación era los gastos inesperados productos de la mala planificación, así un sitio de venta de productos no tenía presupuestado los gastos de logística de despacho.

Un factor adicional, fue la sobre valoración de las empresas, ya que se confundió el beneficio que el usuario tiene por el consumo del producto o servicio, que sin dudas es alto (disminución de costos en las comunicaciones, así como los costos de búsqueda de información), pero que no se traduce necesariamente en un beneficio para la empresa y sus inversionistas. Por lo que en algún momento la sostenibilidad del negocio era imposible debido a la acumulación de pérdidas. Además se cometió el error de pensar que el valor previsto podía superar al valor que la compañía podía generar en un plazo previsible.

Ante este gran número de puntos negativos a un seguro nicho empresarial dentro de nuestro medio se pueden tomar en cuenta algunos principios importantes para que una empresa sea negocio:

- Visión del negocio a largo plazo
- Potencial de beneficio real del mercado
- Planteamiento exhaustivo del modelo de negocio
- Robustez del modelo ante el cambio
- Dependencia tecnológica

Visión del negocio a largo plazo

Tener una concepción del negocio y de su entorno en el presente y en un futuro previsible

Potencial de beneficio real del mercado

Distinguir entre el mercado potencial y la demanda real.

Planteamiento exhaustivo del modelo de negocio

Esto contempla la elaboración de un buen plan de negocios que incluye elementos de gestión del proyecto.

Conocer la visión y el objetivo de nuestro negocio no es suficiente, se debe estudiar a fondo en concepto cuál será el origen de nuestros ingresos y cómo lograr mantenerlos.

Es necesaria la modificación de módulos de negocios tradicionales o la combinación de ellos para que sean rentables en Internet.

Robustez del modelo ante el cambio

El cambio del mercado es innevitable en una economía globalizada. Debemos estar preparados para el mismo, reconocer las debilidades para determinar si tenemos control sobre estos cambios. Es imperativo tener en cuenta los factores externos para establecer una estrategia de contingencia a fallas.

Dependencia de la tecnología

Es necesario determinar hasta qué punto se podría depender de la tecnología, si el modelo de negocio es muy dependiente de la tecnología será imprescindible dominarla. Es decir saber gestionar su disponibilidad, su propiedad y sus ciclos de vida.

Los negocios en internet exitosos han sido empresas que han sabido aprovechar ciertos beneficios de la tecnología, como las externalidades de la red para fidelizar a sus usuarios. Se producen externalidades de

red cuando la utilidad percibida por un usuario de la red aumenta al añadirse a esta red más usuarios.

En resumen se debe volver a los principios que rigen cualquier proyecto empresarial, para generar valor a los clientes y obtener suficientes ingresos para desarrollar la actividad.

Para E-guana, como sistema teconológico, se realiza una implementación donde se maximicen todas las ventajas que se ofrecen en un modelo de negocios de ventas.

1.3.2 E-Procurement

Definiríamos un e-procurement desde su traducción al español como procuraduría electrónica, esto se entiende como la automatización de los procesos de adquisición de una organización usando aplicaciones basadas en el Web.

“A diferencia de los sistemas de planificación de recursos empresariales (ERP) que permiten a los negocios automatizar sus procesos internos, e-procurement permite a compradores y proveedores ampliamente dispersos a juntarse, interactuar y ejecutar transacciones de compra directamente sobre la Internet “[2].

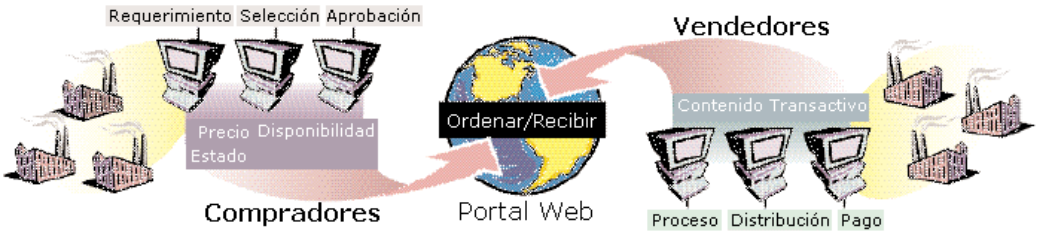

Figura 1-1 Esquema de sistema E-Procurement

Fuente: Hamilton, Myron, Leech. Autor: Entering the Dynamic New E-procurement Marketplace.

El hecho de permitir que una comunidad comercial pueda realizar diferentes operaciones en conjunto nos conduce a concluir que el uso de un e-procurement en el nuevo milenio es una solución a empresas con grandes demandas, puesto que como explica la cita anterior el hacerlo desde la Internet convierte cada operación como oportuna, dando paso a un segundo filtro como es el costo. E-guana, como se lo demuestra en la parte de analisis de factibilidad, tiene una ventaja sobre este último factor puesto que es desarrollado con herramientas de Código Abierto se reducen considerablemente los costos de desarrollo y por lo tanto el precio final.

Para finalizar, una característica interesante es la subasta, la cual permite la venta de excesos de inventario así como las licitaciones de órdenes hechas por el comprador, cuya finalidad es ideal para empresas que desean vender algún bien que pudo haber llegado al final de su vida útil, hablando en términos contables.

1.3.3 Leyes de negocios en línea en el Ecuador

A principios del año 2002 el Congreso Nacional emitió la “Ley de Comercio Electrónico, Firmas y Mensajes de datos”, y establece como disposición final la expedición del reglamento a la ley, a fines del mismo año el Presidente de la República expide el “Reglamento General a la ley de comercio electrónico, firmas electrónicas y mensajes de datos”.

La Ley surge considerando la importancia de las redes electrónicas en los procesos de comercio y producción, además de la necesidad de impulsar el desarrollo del comercio, la educación y la cultura, así como la necesidad de normar, controlar y regular todas las relaciones surgidas de los servicios de las redes electrónicas.

El artículo 1 es “Objeto de la ley.- Esta ley regula los mensajes de datos, la firma electrónica, los servicios de certificación, la contratación electrónica y telemática, la prestación de servicios electrónicos, a través de redes de información, incluido el comercio electrónico y la protección a los usuarios de estos sistemas” [6]. Las firmas electrónicas son un verdadero pilar en la autenticación del usuario porque ya no sólo tendremos un destinatario en un correo o transacción sino también una firma o señal electrónica única que nos permitirá segmentar el uso de

aplicaciones y elementos informáticos con suma precisión, algo que es de vital importancia para transacciones que se puedan dar en un market place.

El artículo 2 indica “Reconocimiento jurídico de los mensajes de datos.- Los mensajes de datos tendrán igual valor jurídico que los documentos escritos. Su eficacia, valoración y efectos se someterán al cumplimiento de lo establecido en esta ley y su reglamento” [6]. Como indica este artículo es de reconocimeinto jurídico todo mensajes de datos convirtiendo a un correo electrónico en un verdadero testimonio ante instancias superiores, en mi opinión esto ya es un paso importante para llegar al nivel cero papel donde no necesitemos usar insumos de oficina para tener constancia ante un superior.

El comercio electrónico, según el glosario de términos de la Ley, se establece como “Es toda transacción comercial realizada en parte o en su totalidad, a través de redes electrónicas de información” [6].

Entre los artículos más directamente relacionados con este punto están:

Artículo 44.- Cumplimiento, de formalidades.- Cualquier actividad, transacción mercantil, financiera o de servicios, que se realice con

mensajes de datos, a través de redes electrónicas, se someterá a los requisitos y solemnidades establecidos en la ley que las rija, en todo lo que fuere aplicable, y tendrá el mismo valor y los mismos efectos jurídicos que los señalados en dicha ley [6].

Artículo 45.- Validez de los contratos electrónicos.- Los contratos podrán ser instrumentados mediante mensajes de datos. No se negará validez o fuerza obligatoria a un contrato por la sola razón de haberse utilizado en su formación uno o más mensajes de datos .

Cualquier negocio resultante del uso de esta tecnología tendrá la misma validez y amparo legal que uno dado por los métodos tradicionales [6].

En resumen la ley busca regular todos los procesos involucrados en los servicios de redes electrónicas estableciendoles un marco jurídico.

Según las predicciones del CORPECE (Corporación Ecuatoriana de Comercio Electrónico), existen claros indicios de que el comercio electrónico tendrá vías importantes para el año 2006 mientras que para el 2007 habrá despegado y asentado [5].

Entre los puntos que les permiten hacer dicha predicción [5]:

- La participación en el comercio electrónico del sector financiero y bancario (Bolivariano, Pichincha, Pacífico).
- La experiencia de instituciones públicas en el uso de Internet para facilitar trámites públicos (SRI, IESS, CAE).
- La inminente aprobación de las empresas de firmas electrónicas por parte del CONATEL.
- La conectividad entre los Concejos Provinciales del País.
- La reducción, aunque aún no óptima, de los costos de los servicios de Internet.
- Las iniciativas conjuntas entre grandes empresas para compartir contenidos y convergencia de medios (Terra, diario El Comercio, diario HOY, ECUAVISA, Movistar).
- El impulso de las tiendas virtuales de la pequeña empresa por parte del programa CONQUITO.
- La creación del sector de Tecnologías de Información y Comunicaciones (TICs) en las Cámaras de la Pequeña Industria.
- El mantenimiento de ferias y eventos tecnológicos.
- El desarrollo de alianzas estratégicas entre las empresas de logística y distribución con los proveedores permiten entregar un producto en cualquier parte del País en un lapso de horas.

- La existencia de 9 millones de aparatos y líneas telefónicas entre móviles y fijas.
- El uso masivo de SMS para comunicación y para micropagos es una señal directa de la posible orientación del mercado de negocios electrónicos en el País.

El mismo artículo indica de igual forma lo que todavía hace falta [5]:

- Normas aduaneras para facilitar las importaciones de bienes de bajo costo y uso personal.
- Aprobación definitiva de las empresas de firma electrónica.
- Decisión del SRI y CAE para implementar la tecnología en sus procesos.
- Impulso de las empresa privada para forzar la atención de la empresa pública en el comercio electrónico.

El ambiente regulatorio es fundamental para el desarrollo empresarial en cualquier área y muy especialmente en los sectores tecnológico y de telecomunicaciones. Un ambiente regulatorio saludable, implica disponer de acciones regulatorias claras y que éstas efectivamente se cumplan por regulador y regulados [4].

Pero Carlos Vera, Presidente del Comité Panamericano de Tecnología y Telecomunicaciones (1994-2002), también considera que “El crecimiento sano del mercado exige un ambiente regulatorio predecible y confiable del cual en el País nos encontramos bastante lejanos” [4]. Pese a que lo mencionado es una realidad, el ambiente al cual ser refiere como inexistente se mantiene de esa forma por comentarios similares que lo único que hacen es fomentar no sólo en el área tecnológica sino en ámbitos de índoles artísticos, entre otras, es por eso que al contrario de apoyar al terror psicológico plasmado por este tipo de mensajes debemos construir primero un ambiente regular dentro de nuestra cotidianidad.

1.3.4 Aplicaciones J2EE

Según la Tutoría J2EE una Aplicación J2EE es “Cualquier unidad desplegable de funcionalidad J2EE. Ésta puede ser un solo módulo J2EE o un grupo de módulos empaquetados en un archivo EAR junto a un descriptor de despliegue J2EE. Las aplicaciones J2EE son típicamente pensadas para ser distribuidas entre múltiples capas de computación” [3].

Extendiendo este concepto, las aplicaciones J2EE son sistemas empresariales implementados en múltiples capas distribuidas. La lógica de la aplicación se divide en componentes dependiendo de la

funcionalidad, y cada componente puede estar instalado en una máquina diferente dependiendo de la capa del ambiente distribuido en la que se encuentre.

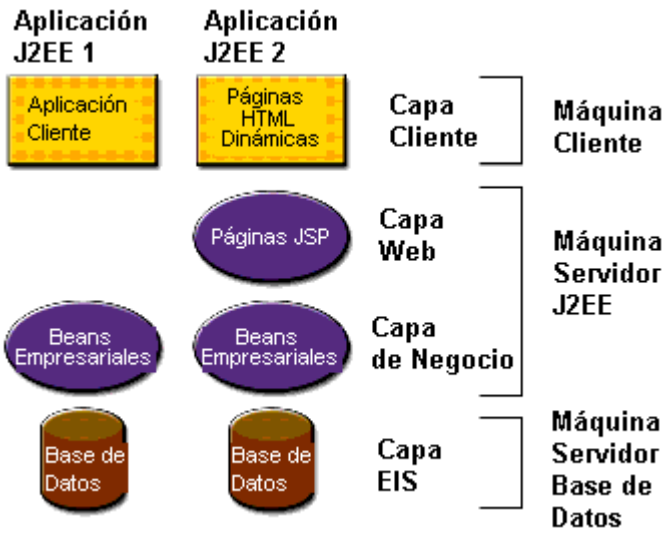

Figura 1-2 Aplicaciones de Múltiples Capas

Fuente: The J2EE Tutorial. Autor: Armstrong, Ball, Bodoff, Carson, Evans, Green, Haase, Jendrock.

La figura muestra dos aplicaciones J2EE multicapas:

- Componentes de la capa de cliente se ejecutan en la máquina del cliente
- Componentes de la capa Web se ejecutan en el servidor J2EE
- Componentes de la capa de negocio se ejecutan en el servidor J2EE
- Software de la capa del sistema de información empresarial (EIS) corre en el servidor empresarial.

Aunque como se muestra en la figura una aplicación J2EE puede estar formada por tres o cuatro capas, las aplicaciones J2EE son consideradas como aplicaciones de tres capas ya que son distribuidas entre tres ubicaciones: el cliente, el servidor J2EE y la base de datos o sistema legado en la parte posterior.

Estos componentes son escritos en el lenguaje de programación Java y son compilados de la misma forma que cualquier programa en el lenguaje. Las diferencia entre estos componentes J2EE y las clases Java comunes es que estos componentes son ensamblados en una aplicación J2EE, son verificados de que estén bien formados y acorde a la especificación J2EE, y son desplegados para producción, donde son ejecutados y controlados por el servidor J2EE.

El uso de J2EE en sistemas de la vida real raramente se presenta en el rango completo de las tecnologías que ofrece, su uso más común se da en la capa Web con la implementación de aplicaciones Web con servlets y JSP debido a la innegable complejidad de la tecnología de EJBs tal como se lo mencionó en el capítulo 1.

1.3.5 E-Guana

E-Guana es una implementación de e-procurement utilizando la tecnología J2EE.

Para facilitar este objetivo se dividió al sistema en varios módulos:

- Módulo de Administración
- Módulo de StoreFront
- Módulo de Licitación y Subastas
- Módulo de Reportes
- Módulo de Pagos de Transacciones

A continuación se definirá cada módulo y su relación con los componentes de un sistema de e-procurement:

Módulo de Administración

Se encarga del registro de empresas en el sistema, mantenimiento de usuarios, creación y verificación de roles y permisos, mantenimiento del catálogo de productos de cada empresa, configuración de variables del sistema y mantenimiento de otras entidades secundarias. Por lo tanto este módulo está cumpliendo la tarea del componente “contenido de catálogo” de un sistema de e-procurement, así como del “manejo de” y “mantenimiento del sistema”.

Módulo de StoreFront

Permite la búsqueda de productos que pueden terminar en una compra, el sistema notifica en todo momento el estado de la compra tanto al comprador como al vendedor. Este módulo también forma parte del componente de “contenido de catálogo” puesto que hace uso de los datos del catálogo de productos para armar el carro de compras, otro componente involucrado con este módulo es “establecimiento de relaciones comprador/vendedor” por tratarse directamente de la compra de productos.

Módulo de Licitaciones & Subastas

Provee el servicio de licitaciones de productos por parte de un comprador y la subasta de productos a cargo de un vendedor, adicionalmente maneja la idea de requerimientos, así un grupo de usuarios, organizados en unidades de negocios, puede hacer pedidos de productos a la unidad principal o matriz de la empresa para que esta última sea la que decida si hacer una compra directa o licitar los productos. Similar al módulo anterior éste forma parte del componente “contenido de catálogo” y “establecimiento de relaciones comprador/vendedor” pero además del componente “establecimiento de precio” por la naturaleza de los dos servicios provistos por este módulo

donde los precios son fijados por compradores, en subastas, y por vendedores, en licitaciones.

Módulo de Reportes de E-guana

Genera varios tipos de reportes gerenciales que pueden utlizarse para mejorar estrategias de ventas o eficiencia en adquisición de recursos como empresa y para mejorar el rendimiento del sistema como administrador, entre otras cosas, por lo tanto este módulo sirve de soporte a la mayoría de componentes del sistema de e-procurement.

Módulo de Pagos de Transacciones

Cuando se realiza una compra, directamente o por medio de una licitación / subasta, el módulo que genera la misma se debe comunicar con este módulo, el que a su vez se comunica con las entidades bancarias de las empresas involucradas en la compra para que hagan el pago respectivo. Otra tarea de este módulo es verificar que cada empresa tenga una cuenta bancaria en alguna de las entidades bancarias registradas, y que tenga los fondos suficientes para hacer la compra. Por lo anterior se ve que este módulo cumple parte de las tareas del componente de “manejo de facturación” de un sistema de e-procurement.

En cuanto al uso de la tecnología J2EE, E-Guana es en definitiva una aplicación J2EE.

Cada módulo implementa un conjunto de componentes J2EE en la capa Web y la capa de negocios.

Los componentes que implementa cada módulo son páginas JSP, servlets y beans empresariales (algunos de los cuales interactúan entre módulos).

Para facilitar el trabajo e independencia cada módulo implementó su propia aplicación Web, algo similar se pretendió con los componentes de negocio, pero debido a restricciones de la especificación se tuvo que implementar una sola aplicación EJB que integre todos los beans de todos los módulos

La capa del sistema de información empresarial es una base de datos.

CAPÍTULO 2

2 HERRAMIENTAS Y TÉCNICAS PARA EL DESARROLLO DEL PROYECTO.

2.1 Open source.

En estos últimos años se escucha con mucha frecuencia el término de Software Libre (Free Software) y más recientemente el de Software de Código Abierto (Open Source Software), estos términos se refieren al modelo de desarrollo y distribución de software desarrollado y distribuido cooperativamente.

Hoy en día el concepto Código Abierto no solo se refiere al software sino que se está introduciendo en los diferentes tipos de disciplinas que van

desde la biotecnología para facilitar el trabajo a los biólogos y poner a su disposición las más reciente tecnología genética y biológica, ya hasta en la fabricación de automóviles, que lo que pretende es crear un automóvil cuyas especificaciones sea de dominio público.

Elegimos usar Código Abierto como base para el desarrollo de nuestro E-procurement no sólo por el costo nulo sino por la infinidad de código que se puede encontrar para facilitar y acelerar el desarrollo del mismo. Esta es una ventaja también porque siendo Código Abierto es posible incluir nuevos componentes a la aplicación principal, que al ser desarrollados pueden adaptarse convenientemente como algunos productos que hemos manejado: CMS, ERP, entre otros.

2.1.1 Filosofía Open Source.

La filosofía Open Source o de código abierto se cimienta, como lo indica Richard Stallman, programador y una figura relevante del movimiento por el software libre, en su libro “Software para una sociedad libre”, en la capacidad de compartir el código. Una de las ventajas de este esquema mencionado por Stallman es el mejoramiento constante del código, la fácil distribución del mismo y la adaptabilidad para ser traducido en diferentes lenguajes.

Esta filosofía dentro de E-guana fue la que maximizó el rendimiento de cada rutina de programación, y de la misma forma el concepto total de

la aplicación como un modelo de negocio por internet. Si tomamos en cuenta lo mencionado a través del capítulo 1, nos podemos dar cuenta que la modalidad de código abierto podría establecer una pequeña brecha de ventaja tecnológica convirtiendola en una ventaja contra modelos de negocio con software propietario, como también una gran desventaja si elegimos un desarrollo que tome mucho tiempo y más minucioso que sacrifique la operación diaria.

2.2 Lenguajes de Programación.

Al desarrollarse las primeras computadoras electrónicas, se vio la necesidad de programarlas, es decir, de almacenar en memoria la información sobre la tarea que iban a ejecutar. Las primeras se usaban como calculadoras simples; se les indicaban los pasos de cálculo, uno por uno.

John Von Neumann desarrolló el modelo que lleva su nombre, para describir este concepto de "programa almacenado". En este modelo, se tiene una abstracción de la memoria como un conjunto de celdas, que almacenan simplemente números. Estos números pueden representar dos cosas: los datos, sobre los que va a trabajar el programa; o bien, el programa en sí.

Los lenguajes más primitivos fueron los lenguajes de máquina. Ya que el hardware se desarrolló antes del software, y además cualquier software finalmente tiene que expresarse en el lenguaje que maneja el hardware.

La programación en esos momentos era sumamente tediosa, pues el programador tenía que "bajarse" al nivel de la máquina y decirle, paso a paso, cada punto de la tarea que tenía que realizar. Además, debía expresarlo en forma numérica; y por supuesto, este proceso era propenso a errores, con lo que la productividad del programador era muy limitada. Sin embargo, hay que recordar que en esos momentos, simplemente no existía alternativa.

El primer gran avance que se dio, fue la abstracción dada por el Lenguaje Ensamblador, y con él, el nacimiento de las primeras herramientas automáticas para generar el código máquina. Esto redujo los errores triviales, como podía ser el número que correspondía a una operación, que son sumamente engorrosos y difíciles de detectar, pero fáciles de cometer. Sin embargo, aún aquí es fácil para el programador perderse y cometer errores de lógica, pues debe bajar al nivel de la forma en que trabaja el CPU, y entender bien todo lo que sucede dentro de él.

En los años 50 y 60 con el desarrollo de algoritmos de más elevado nivel, y el aumento de poder del hardware, empezaron a entrar al uso de computadoras científicos de otras ramas; ellos conocían mucho de Física, Química y otras ramas similares, pero no de Computación, y por supuesto,

les era sumamente complicado trabajar con lenguaje Ensamblador en vez de fórmulas. Así, nació el concepto de Lenguaje de Alto Nivel, con el primer compilador de FORTRAN (FORmula TRANslation), que, como su nombre indica, inició como un "simple" esfuerzo de traducir un lenguaje de fórmulas, al lenguaje ensamblador y por consiguiente al lenguaje de máquina. A partir de FORTRAN, se han desarrollado innumerables lenguajes, que siguen el mismo concepto: buscar la mayor abstracción posible, y facilitar la vida al programador, aumentando la productividad, encargándose los compiladores o intérpretes de traducir el lenguaje de alto nivel, al lenguaje de máquina.

Hay que notar la existencia de lenguajes que combinan características de los de alto nivel y los de bajo nivel (es decir, Ensamblador). Un buen ejemplo es C: contiene estructuras de programación de alto nivel, y la facilidad de usar librerías que también son características de alto nivel; sin embargo, fue diseñado con muy pocas instrucciones, las cuales son sumamente sencillas, fáciles de traducir al lenguaje de la máquina; y requiere de un entendimiento apropiado de cómo funciona la máquina, el uso de la memoria, etcétera. Por ello, muchas personas consideran a lenguajes como C (que fue diseñado para hacer sistemas operativos), lenguajes de nivel medio.

2.2.1 Java.

El lenguaje de programación Java, fue diseñado por la compañía Sun Microsystems Inc, con el propósito de crear un lenguaje que pudiera funcionar en redes computacionales heterogéneas (redes de computadoras formadas por más de un tipo de computadora, ya sean PC, MAC's, estaciones de trabajo, etc.), y que fuera independiente de la plataforma en la que se vaya a ejecutar. Esto significa que un programa de Java puede ejecutarse en cualquier máquina o plataforma. El lenguaje fue diseñado con las siguientes características en mente:

- **Simple**. Elimina la complejidad de los lenguajes como "C" y da paso al contexto de los lenguajes modernos orientados a objetos. La filosofía de programación orientada a objetos es diferente a la programación convencional.
- **Familiar**. Como la mayoría de los programadores están acostumbrados a programar en C o en C++, la sintaxis de Java es muy similar al de éstos.
- **Robusto**. El sistema de Java maneja la memoria de la computadora. No se tiene que preocupar por apuntadores, memoria que no se esté utilizando, etc. Java realiza todo esto sin necesidad de que el programador se lo indique.
- **Seguro**. El sistema de Java tiene ciertas políticas que evitan se puedan codificar virus con este lenguaje. Existen muchas

restricciones, especialmente para los applets, que limitan lo que se puede y no puede hacer con los recursos críticos de una computadora.

- **Portable**. Como el código compilado de Java (conocido como byte code) es interpretado, un programa compilado de Java puede ser utilizado por cualquier computadora que tenga implementado el interprete de Java.
- **Independiente a la arquitectura**. Al compilar un programa en Java, el código resultante un tipo de código binario conocido como byte code. Este código es interpretado por diferentes computadoras de igual manera, solamente hay que implementar un intérprete para cada plataforma. De esa manera Java logra ser un lenguaje que no depende de una arquitectura computacional definida.
- **Multithreaded**. Un lenguaje que soporta múltiples hilos es un lenguaje que puede ejecutar diferentes líneas de código al mismo tiempo.
- **Interpretado**. Java corre en máquina virtual, por lo tanto es interpretado.
- **Dinámico**. Java no requiere que compile todas las clases de un programa para que este funcione. Si realiza una modificación a una

clase Java se encarga de realizar un Dynamic Binding o un Dynamic Loading para encontrar las clases.

Java puede funcionar como una aplicación sola o como un "applet", que es un pequeño programa que se ejecuta en una página Web. Los applets de Java se pueden embeber a una página de Web, y con esto se obtiene un programa que puede correr desde cualquier navegador Web.

Java funciona de la siguiente manera: El compilador de Java deja el programa en un Pseudo-código (no es código maquinal[1]) y luego el intérprete de Java ejecuta el programa (lo que se conoce como el "Java Virtual Machine"). Por eso Java es multiplataforma, existe un intérprete para cada máquina diferente.

Para entender bien como funciona un applet de Java vea el siguiente ejemplo:

1. Existe un código de Java en un servidor de Web. (Los códigos de Java se caracterizan por tener la extensión *.class).
2. Una persona en Internet, con un browser compatible con Java, realiza una conexión al servidor.

[1] Codigo Maquinal es el código binario que la computadora entiende y puede ejecutar

3. El servidor envía el documento HTML y el código en Java (*.class).
4. En la computadora del usuario remoto llegan ambos, y la Máquina Virtual de Java, que está en el browser, transforma el código Java en un código que entienda la máquina local y se ejecuta el programa dentro de la página de Web.
5. Si el usuario realiza otra conexión a otro URL o se sale del browser, el programa se deja de ejecutar y en la computadora no queda rastro de él.

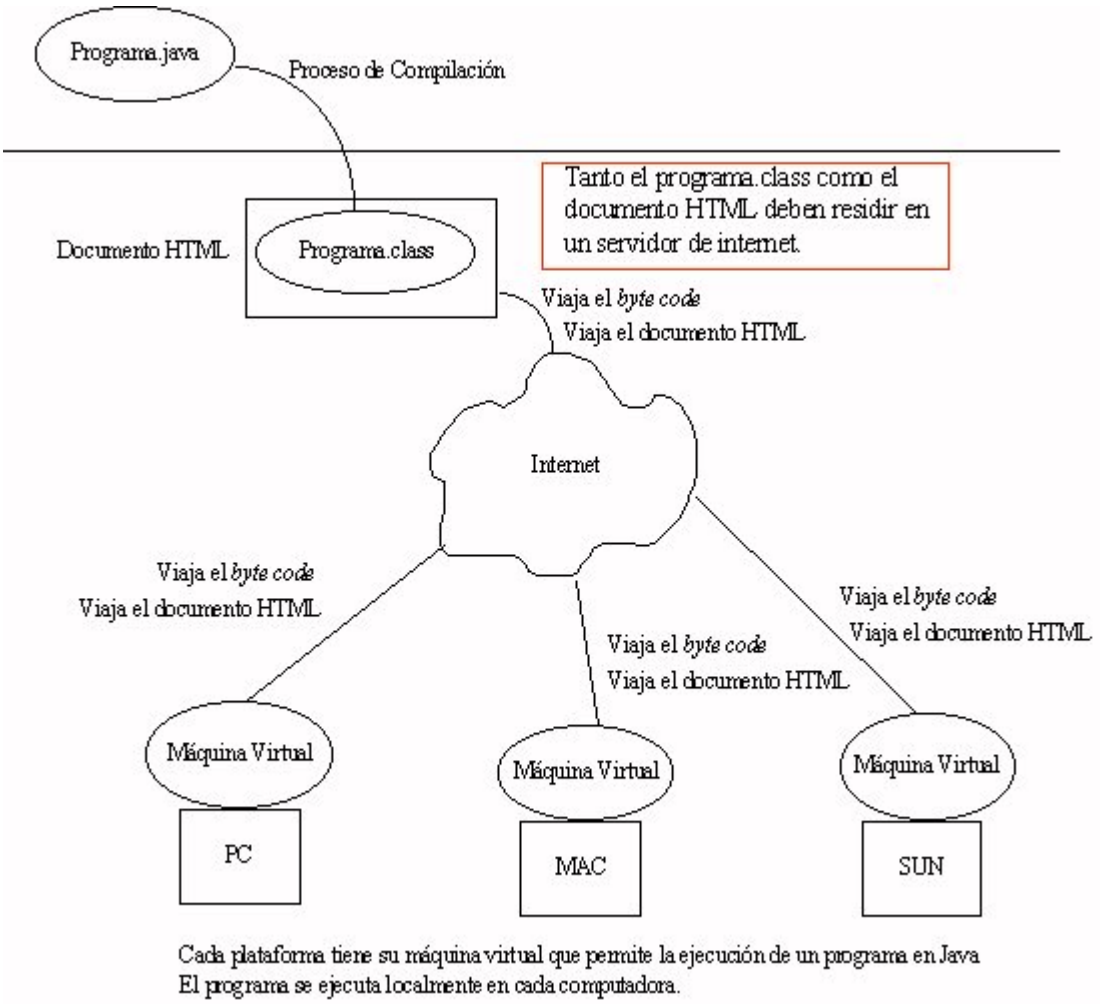

Figura 2-1 Funcionamiento de un Applet Java
Fuente: http://www.monografias.com/trabajos/lengprog/lengprog.shtml

Como base del desarrollo en E-guana, Java reune todas las características para un sistema de ventas en línea por muchos de los puntos mencionados anteriormente. El simple hecho de ser multiplataforma nos brinda una ventaja sobre sistemas que necesitan migrar su código para funcionar en un ambiente diferente.

2.3 Base de datos

Una base de datos es un conjunto de datos que pertenecen al mismo contexto almacenados sistemáticamente para su uso posterior. De esta forma una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta.

> "MySQL continua teniendo una alta aceptación por un gran número de industrias y es conocido por su rentabilidad, fácil uso y rendimiento." [11]

La base datos que usamos para el proyecto E-guana es MySQL, se escogió este motor de base de datos para confirmar lo que Noel Yuhanna, del Forrester Research nos indicaba y de hecho se podría añadir que uno de los puntos importantes es que se necesita muy poco recurso de hardware, y que parte de el rendimiento que esta base de datos nos ofrece se debe a que se puede ejecutar bajo un sistema operativo modo texto.

Las principales características de MySQL Server según su página oficial son:

1. Aprovecha la potencia de sistemas multiprocesador, gracias a su implementación multi-hilo (ejecución de varios procesos simultaneamente).

2. Soporta gran cantidad de tipos de datos para las columnas.
3. Dispone de API's[2] para varios lenguajes como Java, C++, C, PHP, etc.
4. Gran portabilidad entre sistemas.
5. Soporta varios índices por tabla, hasta un máximo de 32 por tabla.
6. Buen nivel de seguridad en los datos, por medio de la gestión de usuarios y contraseñas.

2.4 La plataforma J2EE.

J2EE es, según Ethan Henry columnista de la revista electrónica JAVAWORLD, una plataforma para crear aplicaciones de servidor. La plataforma J2EE es esencialmente un entorno distribuido aplicación – servidor, un entorno Java que ofrece:

1. Un conjunto de varios API de extensiones Java para construir aplicaciones. Estos API definen un modelo de programación para aplicaciones J2EE.

[2] Application Program Interface programa de aplicación de interfaz, parte del sistema operativo que provee a las aplicaciones una interfaz de uso común o interfaz similar

2. Una infraestructura de periodo de ejecución para albergar y gestionar aplicaciones. Éste es el periodo de ejecución en el que residen sus aplicaciones.

Las aplicaciones que pueden desarrollar con estos dos elementos pueden ser programas para controlar páginas Web o componentes para implementar transacciones complejas de bases de datos, o incluso applets de Java, todos ellos distribuidos por la red.

2.5 Herramientas para Programación.

El sistema E-guana fue desarrollado con herramientas de programación de código abierto. Además de ser desarrollado en herramientas de código abierto, E-guana es ejecutado en un ambiente de código abierto. En esta sección se describirán las herramientas de desarrollo que se usaron en este proyecto, la herramienta de programación en Java Eclipse, la Base de Datos MySQL y el servidor de aplicaciones JBOSS.

2.5.1 JBOSS.

JBOSS Application Server como su nombre lo describe es un servidor de aplicaciones, lo elegimos por ser gratuito, ser código abierto y por

usar la licencia LGPL[3]. En E-guana decidimos usar la versión 3.2.4 aunque al momento de escritura de este documento JBOSS se encuentra ya en su versión 4. La decisión de usar JBOSS como servidor de aplicaciones en E-guana fue la que más impacto causó en el desarrollo de la aplicación debido a que este importante componente del proyecto determina la arquitectura general del sistema.

2.5.1.1 Características del JBOSS.

JBOSS es lo que se define también como Middleware, software que se encuentra entre la aplicación y la base de datos. JBOSS es el encargado de proveer servicios a la aplicación para que ésta pueda funcionar. Su funcionamiento interno lo hace en forma de capas como se muestra en la figura 2-2.

[3] LGPL: De sus siglas en inglés Lesser General Public License

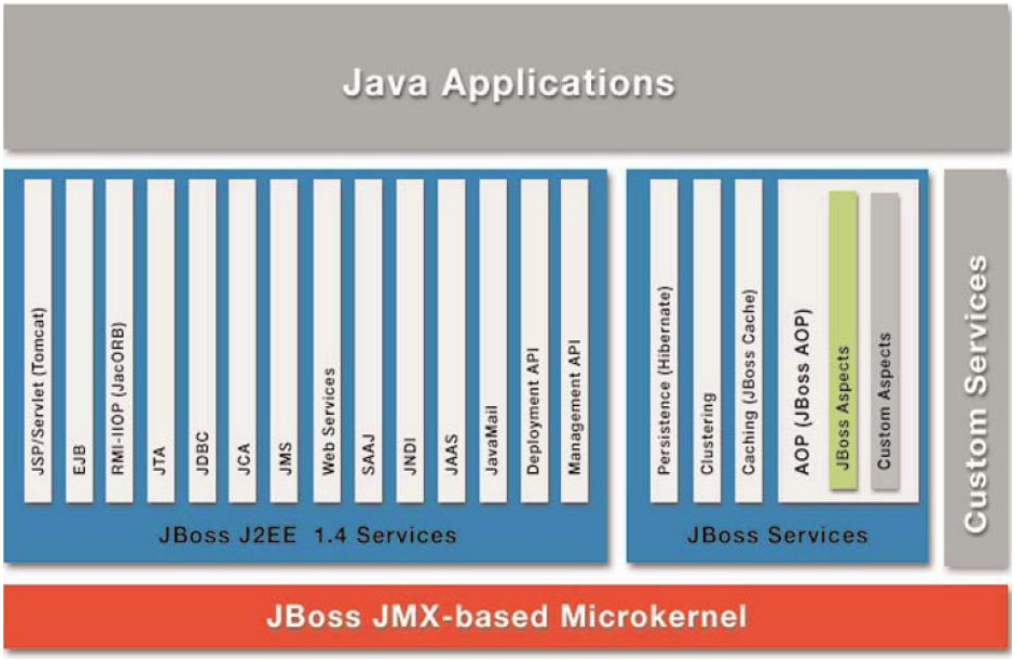

Figura 2-2 Diagrama de capas de JBoss

Fuente: http://www.jboss.org

En la capa más baja tenemos el microkernel basado en JMX y el cuál se encarga de gestionar todas las tareas y servicios que el sistema provee de una manera ordenada.

En la siguiente capa tenemos los servicios de JBOSS, estos servicios pueden ser tanto los servicios que provee J2EE en general como servicios propietarios de JBOSS. Además JBOSS añade la flexibilidad de servicios personalizados, esto es servicios hechos por el desarrollador de la aplicación o por terceros. Esta es la capa más importante del servidor de aplicaciones, es aquí donde se aplican las pautas de J2EE y cumple su estándar, es aquí también donde JBOSS adquiere su identidad y se lo diferencia de otros servidores

de aplicaciones y finalmente es aquí donde el desarrollador saca provecho máximo al poder de este software.

La última capa, la capa superior, es donde se encuentra la aplicación. Aquí es donde la aplicación desarrollada por el equipo de programación llama a los servicios de JBOSS para aspectos que van desde mensajería hasta persistencia

2.5.1.2 Ventajas.

Existe una enorme cantidad de servidores de aplicaciones en el mercado. Prácticamente cada compañía cuyo negocio base es la Tecnología de Información ha desarrollado su propio servidor de aplicaciones. La numerosa población de este tipo de software ha hecho posible destacar las ventajas que JBOSS tiene ante sus competidores.

2.5.1.3 Esquemas de seguridad del JBOSS.

La seguridad es una parte fundamental de cualquier aplicación empresarial. Es necesario restringir quien está permitido a acceder la aplicación y controlar las operaciones que los usuarios puedan ejecutar. La especificación J2EE define un simple modelo de seguridad para los EJBs y los componentes Web basado en roles. El componente de JBOSS que maneja la seguridad es JBossSX. La

seguridad de extensión JBossSX provee soporte para la seguridad declarativa basada en roles del modelo J2EE como también la integración de seguridad especializada a través de una capa de seguridad Proxy. La implementación original del modelo J2EE de seguridad declarativa es basado en los módulos JAAS[4] .

La seguridad en JBOSS, como indica el modelo J2EE, está basada en roles. Es altamente configurable tiene un gran poder y alcance. En E-guana usamos el sistema de roles para definir cuales son las acciones que cada usuario puede ejercer. Los roles se definen en los descriptores de despliegue de cada módulo Web de la aplicación. Aquí también se definen los dominios de acceso para cada rol y los métodos de autenticación de los usuarios.

Además de proveer una manera segura de restringir el acceso a los usuarios, JBOSS también provee en el mismo módulo de seguridad formas confiables de mantener la privacidad e integridad de los datos mediante el uso de encriptación SSL.

JBoss, mediante su modulo JBossSX, provee diversas opciones y herramientas de seguridad para el diseñador de aplicaciones. La seguridad es un factor vital de toda aplicación Web y ésta hasta el

[4] JAAS: Java Authentication and Authorization Service, es un conjunto de APIs usado para autenticación y autorización.

momento de escritura de este documento ha sido inexpugnable en JBoss.

2.5.2 MySQL.

El servidor de base de datos MySQL es la base de datos de código abierto más popular del mundo. Su arquitectura lo hace extremadamente rápido y fácil de configurar. Extensivo reuso de código y una filosofía minimalista en el diseño de características dan como resultado una base de datos compacta, extremadamente rápida, estable y fácil de desplegar.

MySQL es la base de datos de código más popular en el mercado. A pesar de no poseer muchas características estándares en otras bases de datos, como por ejemplo integridad referencial, MySQL se suscribe a la filosofía de la simplicidad. Muchas de las tareas de mantenimiento e integridad de datos las deja en las manos del desarrollador o del servidor de aplicaciones y se preocupa solamente en hacer bien las tareas específicas de una base de datos. Esta filosofía de desarrollo ha permitido que MySQL sea una base datos extremadamente simple y fácil de usar.

2.5.3 Eclipse.

Eclipse es una plataforma de integración de herramientas construida por una comunidad abierta de proveedores de herramientas, esto lo confirman J. des Rivières and J. Wiegand, columnistas de IBM SYSTEMS JOURNALS, en su artículo “Eclipse: A platform for integrating development tools”. Operando dentro del paradigma de código abierto, con una licencia de uso público común que provee código fuente gratis y derechos de distribución mundial, la plataforma Eclipse provee a desarrolladores de herramientas un nivel inigualable de flexibilidad y control sobre sus tecnologías de software.

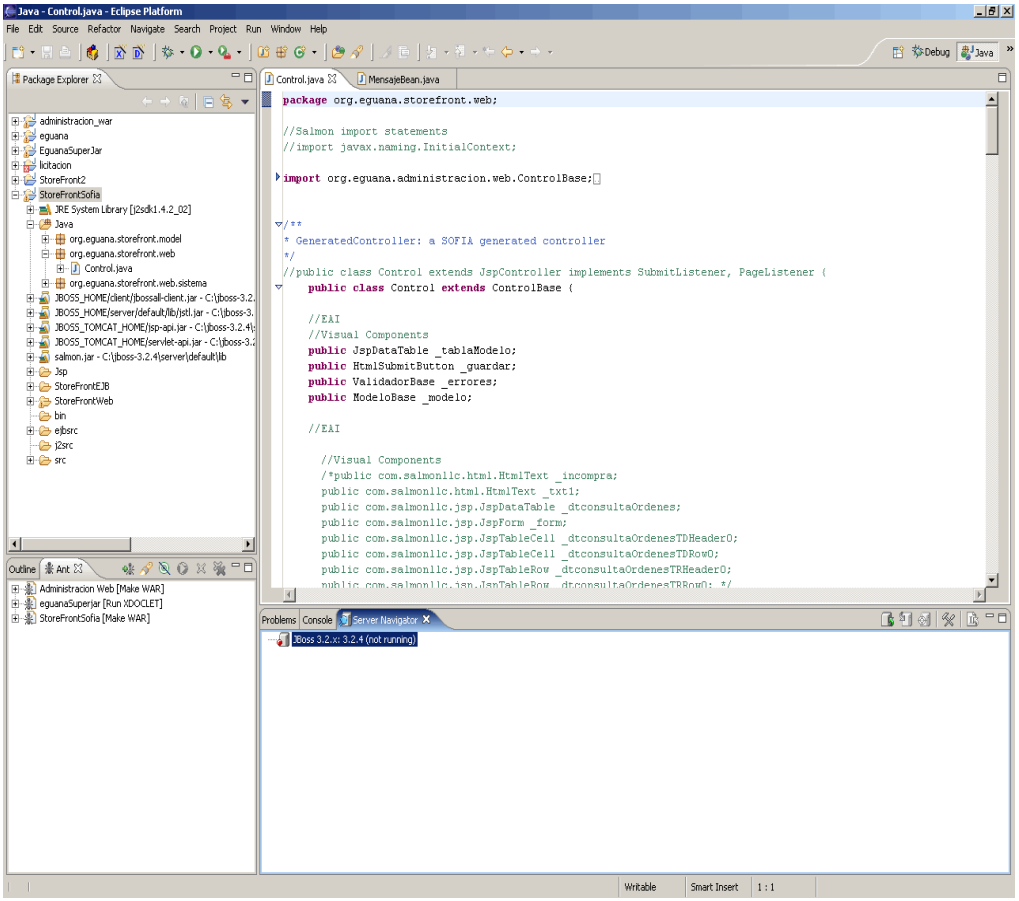

Figura 2-3 Pantalla de Eclipse.

Fuente: E-guana
Autor: E-guana

Generalmente los libros de texto o bibliotecas virtuales como Wikipedia lo llaman simplemente un IDE (Integrated Development Environment) o ambiente de programación, esto subestima el verdadero potencial de la herramienta ya que está enfocada, según su página oficial (www.eclipse.org) como un desarrollador de IDE's e integrador de estos para a su vez crear aplicaciones web, en su mayoría. Eclipse no está limitado a ninguna tecnología de desarrollo, como lo menciona la

documentación dentro su sitio oficial de internet, es abierto mediante una arquitectura de plug-ins, a recibir funcionalidad para cualquier tecnología de cualquier proveedor.

Cómo principal herramienta para desarrollo en el proyecto E-guana, nuestra decisión de uso se cimentó en su flexibidad y su excelente reputación dentro del ambiente Web.

Actualmente, Eclipse está en su versión 3.0 la cuál se utiliza en el proyecto.

2.6 Programación extrema (XP).

La Programación Extrema es una metodología de desarrollo de software que se basa en la simplicidad, la comunicación y la retroalimentación o reutilización del código desarrollado.

Programación Extrema se basa en observar qué es lo que hace que el desarrollo de un programa sea rápido o lento. Programación Extrema es una metodología importante por dos razones. Primero y principalmente, porque constituye un método de control para las actividades de desarrollo de software que se han convertido en métodos operativos estándar. Y segundo, es una de las pocas y nuevas metodologías ligeras desarrolladas para reducir el coste del software. XP va un paso más allá, definir un proceso simple y satisfactorio para la implementación de software.

2.6.1 Introducción a la metodología XP.

Podríamos decir que XP nace “oficialmente” a fines de los años 90 y comienzo del año 2000 en un proyecto desarrollado por Kent Beck en *DaimlerChrysle*r, después de haber trabajado varios años con Ward Cunningham en busca de una nueva aproximación al problema del desarrollo de software que hiciera las cosas más simples.

Kent definió cuatro grandes tareas a realizar en el desarrollo de todo proyecto: planificación, diseño, desarrollo y pruebas.

2.6.2 Objetivos del XP.

El objetivo principal que persigue XP es la satisfacción del cliente. Esta metodología fue diseñada para proporcionar el software que el cliente necesita cuando lo necesite. Deberemos responder de forma rápida a los cambios en las necesidades del cliente, incluso cuando estos cambios se produzcan al final del ciclo de vida de dicho software (simplicidad y retroalimentación).

El segundo objetivo es potenciar al máximo el trabajo en equipo. Tanto los Jefes de Proyecto, como los clientes y desarrolladores, son parte del equipo encargado de la implementación de software de calidad (comunicación). Esto implicará que los diseños deberán ser

claros y sencillos. Y los clientes deberán disponer de versiones operativas cuanto antes para poder participar en el proceso creativo mediante sus sugerencias y aportaciones.

El código será revisado continuamente, mediante la **programación en parejas** (dos personas por máquina).

Se harán pruebas todo el tiempo, no sólo de cada nueva clase **(pruebas unitarias**) sino que también los clientes comprobarán que el proyecto va satisfaciendo los requisitos **(pruebas funcionales**).

Las pruebas de integración se efectuarán siempre, antes de añadir cualquier nueva clase al proyecto, o después de modificar cualquiera existente **(integración continua**), en E-guana se ha utilizado el JUnit.

2.6.3 Las cuatro variables.

XP define cuatro variables para cualquier proyecto software: *coste*, *tiempo*, *calidad* y *alcance*.

Además, especifica que, de estas cuatro variables, sólo tres de ellas podrán ser fijadas por las fuerzas externas al proyecto (clientes y jefes de proyecto), mientras que el valor de la variable libre será establecido por el equipo de desarrollo en función de los valores de las otras tres.

XP hace especial énfasis en equipos de desarrollo pequeños (diez o doce personas como mucho) que se podrán ir incrementando a medida

que sea necesario, pero no antes, o los resultados serán generalmente contrarios a lo esperado.

Sí es bueno, en cambio, incrementar el **coste** del proyecto en aspectos como máquinas más rápidas, más especialistas técnicos en determinadas áreas o mejores oficinas para el equipo de desarrollo.

Con la **calidad** también sucede otro fenómeno extraño: frecuentemente, *aumentar la calidad conduce a que el proyecto pueda realizarse en menos tiempo*. En efecto, en cuanto el equipo de desarrollo se habitúe a realizar pruebas intensivas y se sigan estándares de codificación, poco a poco comenzará a avanzar mucho más rápido de lo que lo hacía antes, mientras la calidad del proyecto se mantiene asegurada –por las pruebas al 100%, lo que conlleva mayor confianza en el código y, por tanto, mayor facilidad para adaptarse al cambio, sin estrés, lo que hace que se programe más rápido.

No tener esto en cuenta conduce a la desmoralización del equipo y, con ello, a la larga, a la ralentización del proyecto mucho más allá del **tiempo** que hubiera podido ganarse al principio con esta reducción de calidad. En cuanto al **alcance** del proyecto, es una buena idea dejar que sea esta la variable libre, de manera que, una vez fijadas las otras tres, el equipo de desarrollo determinaría el alcance mediante:

- La estimación de las tareas a realizar para satisfacer los requisitos del cliente.

- La implementación de los requisitos más importantes primero, de manera que el proyecto tenga en cada instante tanta funcionalidad como sea posible.

2.6.4 El coste del cambio.

Es importante al menos reseñar una de las asunciones más importantes e innovadoras que hace XP frente a la mayoría de los métodos conocidos, y es la referida al coste del cambio. En efecto, siempre ha sido una verdad universal el hecho de que el coste del cambio en el desarrollo de un proyecto se incrementaba exponencialmente en el tiempo, como lo muestra Gerardo Fernandez Escribano autor del texto Introducción a Extreme Programming en la figura 2-9.

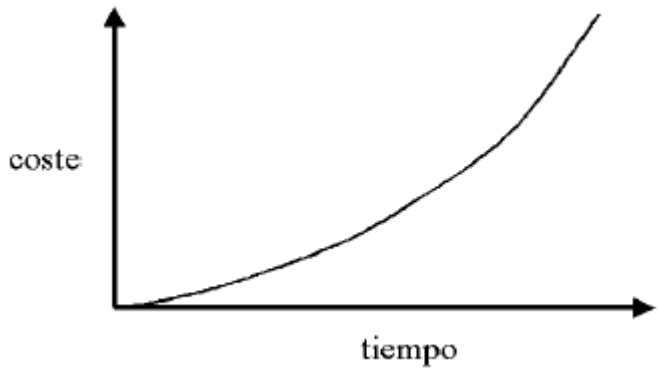

Figura 2-4 Curva Costo vs Tiempo sin usar XP.
Fuente: XP Overview Autor: Gerardo Fernandez

Lo que XP propugna es que esta curva ha perdido validez y que con una combinación de buenas prácticas de programación y tecnología es posible lograr que la curva sea la contraria. Con la metodología XP, se pretende conseguir:

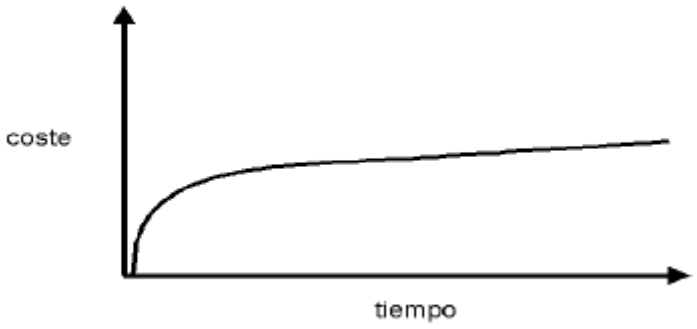

Figura 2-5 Curva Costo vs Tiempo usando XP.
Fuente: XP Overview Autor: Gerardo Fernandez

Si decidimos emplear XP como proceso de desarrollo de software, deberemos adoptarlo basándonos en dicha curva.

La idea fundamental aquí es que, en vez de diseñar para el cambio, diseñaremos tan sencillo como sea posible, para hacer sólo lo que sea imprescindible en un momento dado, pues la propia simplicidad del código, y, sobretodo, los *tests* y la integración continua, hacen posible que los cambios puedan ser realizados tan a menudo como sea necesario.

2.6.5 Ciclo de vida.

Como se ha demostrado, los largos ciclos de desarrollo de los métodos tradicionales son incapaces de adaptarse al cambio, tal vez lo que haya que hacer sean ciclos de desarrollo más cortos.

Esta es una de las ideas centrales de XP, sugeridos por Gerardo Fernandez en su texto, Introducción a Extreme Programming:

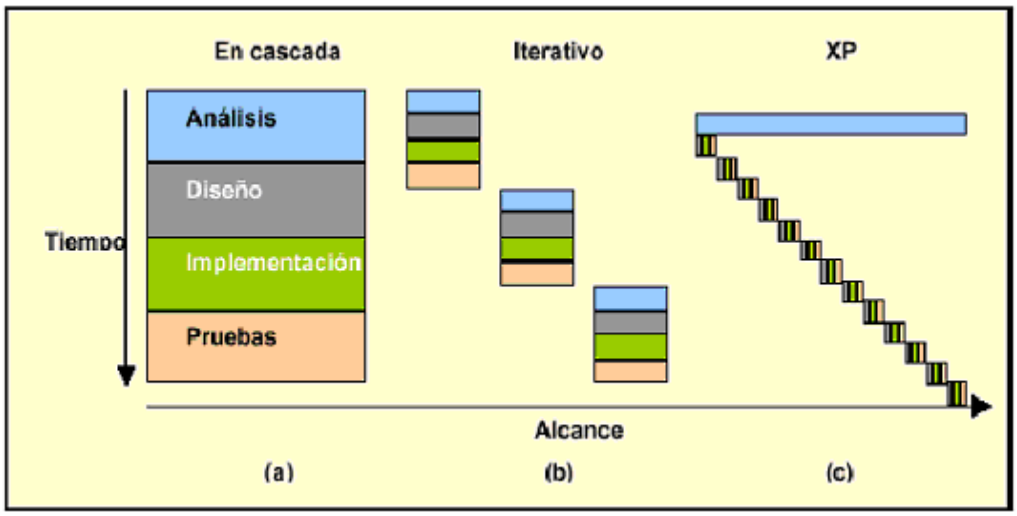

Figura 2-6 Evolución de los largos ciclos desarrollados en cascada (a) a ciclos más cortos (b) y a la mezcla que hace XP (c)

Fuente: XP Overview Autor: Gerardo Fernandez

La figura 2-6 muestra los ciclos de desarrollo y como la programación extrema hace una mezcla de estos.

2.7 Criterios y justificación para la elección de las herramientas seleccionadas.

Debido a la popularidad del movimiento de código abierto existe una inmensa cantidad de programas disponibles para escoger en nuestro proyecto. Por esta razón es necesario justificar la elección de cada una de estas herramientas o programas.

- **JBOSS**: Las razones descritas en la sección 2.5.1.1 y el hecho de que JBOSS es código abierto es razón suficiente para justificar su uso. A continuación mostramos una tabla de comparación de JBOSS con algunos de sus competidores.

Producto	Licencia J2EE	Certificado J2EE	Precio	Plataforma
JBOSS	Sí	1.4	Gratis	Cualquier plataforma con JDK 1.3+
Websphere	Sí	1.3	$12,000	NT, Win2K, Solaris,AIX, OS/400, HP-UX, Red Hat Linux, SUSE Linux, Turbo Linux, Linux/390, NetWare, OS/390.

Oracle	Sí	1.3	$20,000	Solaris, HP-UX, Redhat Linux, United Linux
Resin	NO	1.4	$500	Cualquier plataforma con JDK 1.3+
Jonas	Sí	1.4	Gratis	NT, Linux, Solaris, AIX, HP-UX, Win2K, Netware
JRun	Sí	1.3	$899	NT, Win2K, WinXP, Solaris, SUSE Linux, Red Hat Linux, HP-UX, Compaq Tru64, AIX

Tabla 2-1 Comparación JBOSS y Competencia [16].

Fuente: E-guana
Autor: E-guana

- **MySQL:** Existen varias bases de datos código abierto, entre ellas están PostgreSQL, y Hypersonic. MySQL, además de ser popular, ofrece un excelente compromiso entre simplicidad y robustez. Hypersonic es muy simple y por lo tanto inapropiado para una base de datos muy grande. PostgreSQL a pesar de ser muy robusto puede

volverse complicado de mantener. El resto de bases de datos código abierto no han alcanzado aún una madurez de desarrollo aceptable.

- **Eclipse:** Hasta el momento de escritura de este documento es imposible encontrar una herramienta de desarrollo código abierto que alcance la calidad de Eclipse. Eclipse es superior a sus competidores de código abierto e incluso a muchos competidores de código propietario.

2.8 Análisis

El estudio que hemos realizado basados en lo investigado en el depto. De Adquisiciones de la empresa American Call Center determinamos que las compras no se encuentra completamente automatizadas y presentan algunos problemas, los cuales han sido dividido en tres niveles: nivel operacional, nivel organizacional y nivel tecnológico.

A continuación mencionaremos de manera muy general las dificultades que actualmente atraviesa, según el estudio, el proceso de compras:

2.8.1 Problemas Operacionales

Problema	Causa	Efecto
Lentitud en la actualización de la información.	Se requieren periódicamente cambios de información de proveedores y precios.	Demora en la actualización de la información, debido a la gran demanda de la misma.
	La actualización de la información se realiza manualmente o electrónicamente sin relación alguna.	Pérdida de la información y seguramente tendremos que volver hacer contacto para pedir de nuevo los datos del proveedor.

Tabla 2-2 Cuadro de Problemas operacionales.

Fuente: E-guana
Autor: E-guana

Los problemas operacionales surgen de la necesidad de eficiencia y rapidez en los procesos que se manejan dentro de una empresa.

2.8.2 Problemas Organizacionales

Problema	Causa	Efecto
La información entre unidades no fluye con rapidez.	Procesos de licitación no integrados entre unidades.	Redundancia de procesos de control.
Lentitud en la entrega de solicitud de suministros actualizados.	Inexistencia de procesos que gestionen las solicitudes.	Baja productividad en la empresa y retraso en cosas intrascendentes.

Tabla 2-3 Cuadro de Problemas organizacionales.

Fuente: E-guana
Autor: E-guana

Los problemas organizacionales que se presentan aquí son resultado de los problemas operacionales ya que estos últimos son los que generan los datos centrales.

2.8.3 Problemas Tecnológicos

Problema	Causa	Efecto
Diversidad de fuentes de información, que muchas veces no concuerdan entre sí.	No se dispone de un Sistema de Base de datos	Demora en la actualización de la información.
El comprobar la información de estas fuentes produce retrasos y procesos innecesarios.	No existe un Sistema para unificar los procesos de adquisiciones, o licitaciones.	Dificultad en determinar la situación real del pedido y por consiguiente de las unidades solicitantes.

Tabla 2-4 Cuadro de problemas tecnológicos

Fuente: E-guana

Autor: E-guana

Los problemas tecnológicos no son difíciles de percibir en este caso particular puesto que la falta de un sistema que actúe como agente de control sobre las operaciones dan por resultados los problemas citados en la tabla 2-4.

2.8.4 Solución Propuesta

Después de haber realizado un análisis de los problemas que tiene la empresa en el aspecto organizacional, operacional y tecnológico, se pone a su entera disposición la siguiente solución que está orientada a satisfacer las necesidades de toda empresa interesada en realizar un control en sus adquisiciones:

E-Guana es un sistema de e-procurement, adquisición de productos en línea a través de compras directas o a través de licitaciones y subastas.

El sistema permitirá llevar un control detallado de las actividades de compras y ventas o subastas que se llevan a cabo en toda la Institución. El desarrollo se fundamenta en la tecnología J2EE (Java 2 Enterprise Edition) con una base de datos MySQL bajo una plataforma LINUX.

Para soportar la interoperabilidad con sistemas back-office existentes (sistemas de catálogos), se utiliza XML como interfase desde E-Guana hacia esos sistemas.

El uso de interfaces Web de usuario facilita la utilización y adopción del sistema, puesto que el uso del Internet en servicios bancarios, académicos

y de investigación en la actualidad lo han convertido en un medio común entre la mayor parte de personas.

2.8.5 Análisis Factible

El tiempo estimado de desarrollo es de 4 meses, cada módulo (el módulo de administración también trabaja con el de pagos de transacciones) es desarrollado por 2 personas, considerando un sueldo de 500 dólares mensuales da un total de: $ 16 000 dólares

Para la selección del servidor se debe considerar: la cantidad de sesiones concurrentes a manejar, la cantidad de espacio para almacenamiento, entre otros puntos, no obstante esto no lo constituye en una limitante para la implantación del sistema, por tratarse de características comunes para un servidor de este tipo.

Vale anotar que las características del servidor dependen además de las expectativas del cliente que mantendrá el sistema o de la capacidad de tener al sistema como distribuido entre distintas máquinas con propósitos específicos (una para la base de datos, otra para administración, etc.).

En cuanto al sistema operativo, la propuesta se basa en Linux, mundialmente conocido como un excelente sistema de código abierto para

servidores, con un sinnúmero de plataformas que lo soportan. El costo de implantación de este sistema operativo se reduce a los de la logística necesaria.

Internet será la plataforma de comunicación básica que utilizará el sistema, esto implica la necesaria interoperabilidad con otros sistemas en la red, un enlace a Internet dedicado, capaz, seguro y confiable es vital para el éxito del producto.

A continuación se describirán los costos de las tecnologías asociados a este proyecto:

Costos de desarrollo

Recurso	Cantidad	Valor Mensual	Meses	SubTotal
Programadores	8	$500	4	$16000

Tabla 2-5 Cuadro de Costos de Desarrollo

Fuente: E-guana

Autor: E-guana

El desarrollo se lo hará en grupo de dos personas para cada módulo, esto en un tiempo considerable de 4 meses, dando como resultado $16000.

Costos de software

Recurso	Cantidad	Valor Mensual	Meses	SubTotal
MySql DBMS	1	$0	4	$0
JBOSS Servidor web	1	$0	4	$0
Eclipse IDE's Builder	1	$0	4	$0
Licencias Eclipse IDE	8	$0	4	$0

Tabla 2-6Cuadro de Costos de software.

Fuente: E-guana

Autor: E-guana

Este cuadro muestra el punto fuerte de los costos en el sistema, cuando se trabaja con herramientas de código abierto es posible alcazar este punto de costos 0.

Costos de operación

Recurso	Cantidad	Valor Mensual	Meses	SubTotal
Servidor Intel Xeon 3.0 Server 800 MHZ 2MB	1	$1800	--	$1800
Enlace Internet	--	--	4	$300

Tabla 2-7 Cuadro de costos de operación.

Fuente: E-guana

Autor: E-guana

A nivel de equipos e infraestructura de comunicación, se puede notar que no es posible escatimar costos si se desea que nuestro sistema funcione con eficiencia y rapidez.

La factibilidad operativa del sistema presentado, depende de:

- Alianzas con otras empresas, como por ejemplo las entidades bancarias que permitan la interacción bancaria o implementen una interfaz con E-Guana.
- Cumplimiento de los compromisos de compra establecidos sobre la plataforma, asegurándose la validez de la identidad de la empresa puede facilitar este punto.

- Cantidad de empresas participantes o registradas, y por ende del catálogo ofertado.

Los puntos anteriores pueden ser alcanzados con éxito siguiendo una adecuada política de relaciones públicas.

Al utilizar E-Guana los tiempos requeridos pueden ser reducidos considerablemente ya que se tiene acceso electrónico a un gran catálogo de productos de forma rápida y fácil. Además la información presentada asegura datos actuales, que podrían ser difíciles de obtener a través de los medios tradicionales. Lo anterior lógicamente reducirá los costos para las empresas clientes del sistema, por ejemplo en el uso de papel para la generación de cotizaciones o catálogos escritos. Otro punto fuerte es el control de las compras y ventas que puede hacer cada empresa de forma instantánea. Consecuencia de lo anterior es la resultante eficiencia de la reducción en tiempos destinados para las adquisiciones, el empleado podrá hacer más cosas en el mismo período de tiempo.

En resumen la implantación de un sistema como E-Guana en una empresa es muy recomendable.

CAPÍTULO 3

3 DISEÑO E IMPLEMENTACIÓN DEL MÓDULO LICITACIÓN & SUBASTAS DE E-GUANA.

La parte más importante, al momento de la ejecución de cualquier proyecto de software, es el diseño. Y una herramienta muy útil en esta parte del proceso es el modelado. Asimismo es importante el lenguaje y las herramientas con el que se lo realiza. Existen actualmente muchos lenguajes y herramientas para modelar sistemas. El Lenguaje de Modelamiento Unificado UML es uno de los más utilizados al momento del diseño de aplicaciones informáticas, debido a que ha sido una evolución de otros lenguajes de modelamiento y existen muchos aplicativos que nos facilitan su uso.

Ventajosamente “Eclipse”, nuestra herramienta de desarrollo, con un plug-in (aplicativo complementario), contiene los componentes para realizar los diagramas UML. Además facilita la sincronización entre el modelado y la implementación de las aplicaciones.

Estereotipos UML: Como complemento a los diagramas UML y como parte del Lenguaje de Modelamiento Unificado hemos usado estereotipos para indicar diferentes tipos de objetos y roles en los diagramas. A continuación una lista de los estereotipos utilizados y su significado:

Estereotipo	**Significado**
EJB	Representa un componente Enterprise JavaBean
SessionEJB	Representa un Session Bean como un todo, sin especificar su interface remota, home o implementación.
EntityEJB	Representa un Entity Bean como un todo, sin especificar su interface remota, home o implementación.
View	Representa una Vista (que es la representación de información que el cliente ve).
Interface	Representa a una Interface en Java.
Realice	Indica que una Clase implementa otra Clase o

	Interfaz.
Subsistema	Se refiere a un conjunto de Clases y recursos que realizan una funcionalidad.
Incluye	Es una relación de Casos de Uso en la cual el caso de uso padre utiliza a los casos de uso hijos.
Extend	Es una relación de casos de uso en la cual un caso de uso deriva de otro.

Tabla 3-1 Estereotipos UML utilizados para modelar el proyecto.
Fuente: E-guana. Autor: E-guana

3.1 Arquitectura del Sistema E-Guana.

E-guana comprende una arquitectura n-capas que incluye por lo menos una capa de persistencia, de negocio y una capa de presentación, que mantiene un diseño escalable y modular, que permite la reutilización de componentes y la optimización del desarrollo e implementación.

La capa de presentación es la encargada de interactuar con el usuario, es la que provee un conjunto de Interfaces en tecnología Web para el manejo de solicitudes que el cliente necesite. Es la capa que interactúa directamente con el usuario y absorbe la dificultad del sistema con un conjunto de interfaces fáciles de usar.

La capa de negocio contiene las reglas que E-guana define para el manejo de las transacciones que los usuarios quieren realizar. Es la capa intermedia entre la capa de presentación y la de persistencia.

La capa de persistencia interactúa con la capa de negocio, y maneja la durabilidad de los datos en el tiempo, mediante un motor de base de datos, en nuestro caso MySQL[5].

El siguiente gráfico muestra el diagrama de despliegue, que presenta los componentes o módulos del sistema E-guana, dibujados como subsistemas para efectos de este diagrama, ya que fueron diseñados para operar independientemente pero a su vez con la capacidad de integrarse gracias al diseño de componentes reutilizables con el que ha sido desarrollado.

[5] MySql: Base de datos código abierto.

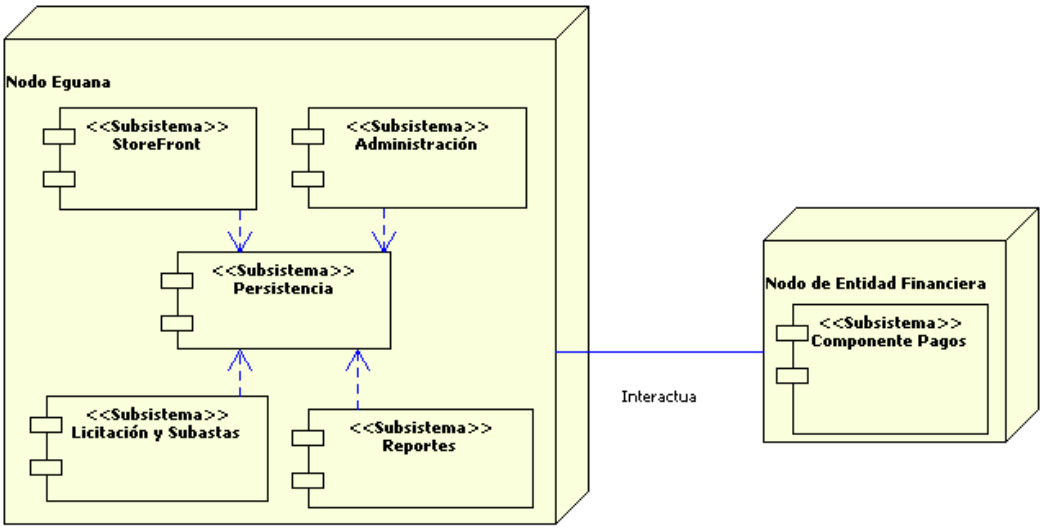

Figura 3-1 Proceso de adquisición de Certificados de Prueba.

Fuente: E-guana Autor: E-guana

Los subsistemas StoreFront, Licitación y Subasta interactúan con el nodo de Entidad Financiera que es una simulación de un sistema de pagos electrónicos.

3.2 Requerimientos del Módulo de Licitación & Subastas dentro de E-guana

Este módulo garantiza la realización de todos los servicios necesarios para la gestión de licitaciones y de subastas.

El objetivo del módulo de Licitación es que el usuario ingrese una necesidad de uno o varios productos y obtenga una lista de ofertas para luego poder elegir la oferta ganadora. El módulo en esta parte involucra la

creación de una Licitación que puede ser realizado de dos maneras, mediante la consolidación de requerimientos y mediante el ingreso directo de la Licitación. A continuación se describen los requerimientos a más detalle:

- Recolección de requerimientos por unidades: Un usuario deberá ingresar el o los productos que solicita. Los requerimientos son almacenados en la base de datos y podrán ser consultados y modificados de acuerdo a las necesidades. Al final el usuario puede realizar la publicación de la Licitación mediante la Consolidación del requerimiento ingresado.
- Ingreso de Licitación: Se podrá ingresar directamente una licitación seleccionando los productos a Licitar y publicándola directamente.
- Oferta de los licitantes: Una persona o empresa puede ofertar sobre el requerimiento publicado, ingresando su precio durante el periodo en el cual la Licitación está en vigencia.
- Elección de la licitación ganadora: Una vez que ha finalizado el tiempo dado para recibir Ofertas de los Licitantes, se procede al análisis de las ofertas ingresadas por parte del usuario que haya creado la licitación respectiva, ya sea por

el precio o calidad. Al final de este paso se selecciona a un ganador. Una vez seleccionado, E-guana tiene la capacidad de Generar una orden de compra.

- Consultas de Licitación y Ofertas: El sistema debe tener la capacidad de listar las licitaciones y ofertas ingresadas, con sus respectivos estados y fechas.

El objetivo del módulo de Subastas es que el usuario ingrese un requerimiento a subastar, y durante un periodo de tiempo, se recibirán las ofertas respectivas y al final se seleccionará la mejor oferta. EL módulo en esta parte involucra:

- Publicación de la subasta: El usuario ingresa en E-guana lo que desea vender.
- Ingreso de Ofertas: Las personas en la Internet realizan sus ofertas comparándolas con las otras ofertas realizadas y con el precio de referencia.
- Elección de la subasta ganadora: Luego de un periodo determinado que finaliza la subasta, el usuario elige la oferta ganadora.
- Consulta de subastas y ofertas: El sistema debe tener la capacidad de listar las subastas y ofertas ingresadas, con sus respectivos estados y fechas.

3.3 Arquitectura del Módulo de Licitación & Subastas del proyecto E-guana.

E-guana en sus procesos de Licitación y Subastas está basada en el modelo de diseño de aplicaciones empresariales MVC[6] que incluye un conjunto de patrones de diseño embebidos que optimizan y mejoran la arquitectura del sistema, esto en conjunto con JSF[7] que facilita la administración de la visualización al usuario y complementándose con los EJB[8´]s que facilita el acceso a los métodos del negocio y a los datos.

La capa de presentación está implementada con la tecnología JSF que en la actualidad facilita el desarrollo de las interfaces al usuario. JSF trae un marco de trabajo para aplicaciones empresariales que además implementa patrones de diseño de aplicaciones en Java y Web. Y un conjunto de componentes visuales que nos permiten realizar interfaces de usuario muy atractivas y con facilidad de uso.

Implementa la parte del Controlador con un manejador de acciones y eventos que reciben y despachan las interacciones de los usuarios sobre

---

[6] MVC: **Model, View and Controller**. Es un Patrón para diseño de aplicaciones donde el Model (Modelo) representa el corazón de la aplicación y la lógica funcional, View (Vista) representa la presentación a usuarios de la aplicación y Controller (Controlador) maneja la interacción del usuario con la aplicación.

[7] JSF: **Java Server Faces**. Es un conjunto de componentes Java, desarrollados por la comunidad de expertos en aplicaciones web y el Java Community Process.

[8] EJB: **Enterprise Java Beans**. Es un conjunto de componentes Java, desarrollados por Sun MicroSystems, que permiten desarrollar aplicaciones distribuidas y multicapas.

la aplicación. La parte de la Vista es implementada con un conjunto de componentes y validadores que facilitan la creación de interfaces de usuario y el manejo de ellas.

El siguiente gráfico muestra la arquitectura de JSF y como los diferentes componentes interactúan entre si dentro del marco de trabajo MVC:

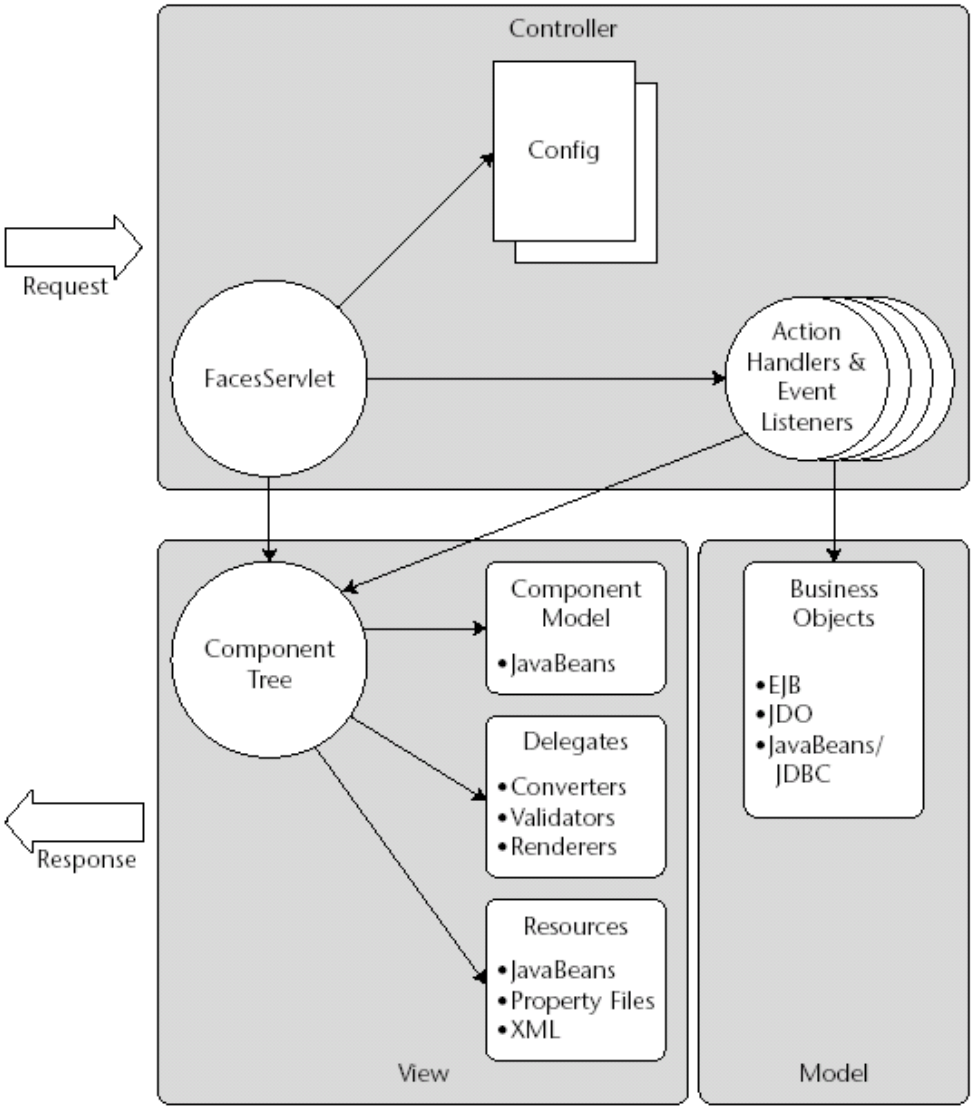

Figura 3-2 Arquitectura MVC de JSF.

Fuente: Mastering JavaServerFaces. Autor: Dudney, Lehr, Willis, Mattingly

La capa de persistencia está implementada con los Entity Beans que son parte de la tecnología EJB[9] que presenta un conjunto de especificaciones

[9] EJB: **Enterprise Java Beans**. Es un conjunto de componentes Java, desarrollados por Sun MicroSystems, que permiten desarrollar aplicaciones distribuidas y multicapas

de arquitectura que separa la lógica de presentación de la lógica de negocio y de los accesos a la base de datos.

La capa de negocio se implementa mediante el uso de Session Beans, que igual son parte de las especificaciones EJB's, y en conjunto de clases POJO[10] las cuales implementan los objetos de transferencia hacia la capa de presentación y es la puerta de acceso a los servicios que se proporcionan en el núcleo de E-guana.

El siguiente gráfico muestra un resumen de la arquitectura del módulo de Licitación y subastas de E-guana la cual fue tomada en cuenta exactamente sin ninguna modificación, dándo un resultado lógico excelente.

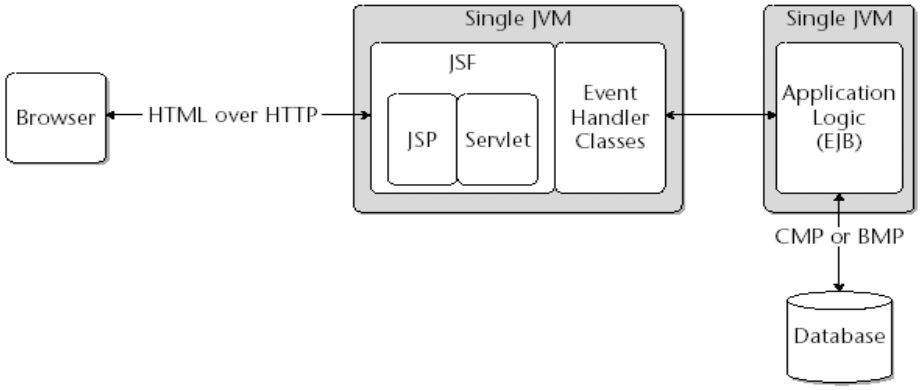

Figura 3-3 Arquitectura de Licitación & Subastas.
Fuente: Mastering JavaServerFaces. Autor: Dudney, Lehr, Willis, Mattingly

[10] POJO: **Poor Old Java Object**. Clases de Java.

3.4 Análisis y Diseño.

En esta sección se ha seleccionado un conjunto mínimo de Artefactos del Lenguaje de Modelamiento Unificado UML, pero que a su vez permiten explicar completamente y con mucha claridad el Diseño del módulo de Licitación y Subastas dentro de E-guana, que van desde plasmar los requerimientos en Casos de Uso, hasta el diseño de las Clases que una vez implementadas, son el motor lógico de la aplicación.

3.4.1 Casos de Uso.

Antes de mostrar los casos de uso y como parte de ellos, vamos a describir los roles (Actores) que tienen los usuarios del sistema.

Roles del Sistema: Dentro de todo el sistema E-Guana existen los siguientes roles:

a) Supervisor: Es el que administra todo el Sistema E-Guana

b) Proveedor: El que puede crear, eliminar empresas virtuales y, adicionar usuario administrador para esa empresa. El Administrador puede crear enlace hacia el inventario de la empresa registrada, puede crear muchos vendedores

c) Vendedor: En nuestro módulo es el encargado de interactuar con el proceso de Subasta. Además es el encargado de establecer contactos

por chat o e-mail, notificaciones, puede ver reportes de cuanto se ha vendido, etc.

d) Comprador: En nuestro módulo es el encargado de realizar el proceso de Licitación. Se pueden crear a titulo personal o como empresas.

Los módulos de Licitación y Subastas no implementan las funcionalidades de todos estos roles, mas son traídos a colación dado a que hay que tenerlos presente dentro de E-guana. En esta sección y las siguientes vamos a tratar a los actores y sus roles simplemente como usuarios y al Actor Comprador como el que interactúa mayormente con la Licitación, y el Actor Vendedor como el que interactúa en la mayoría de casos con la Subasta.

El siguiente diagrama se muestra los Casos de Uso del módulo Licitación y Subastas:

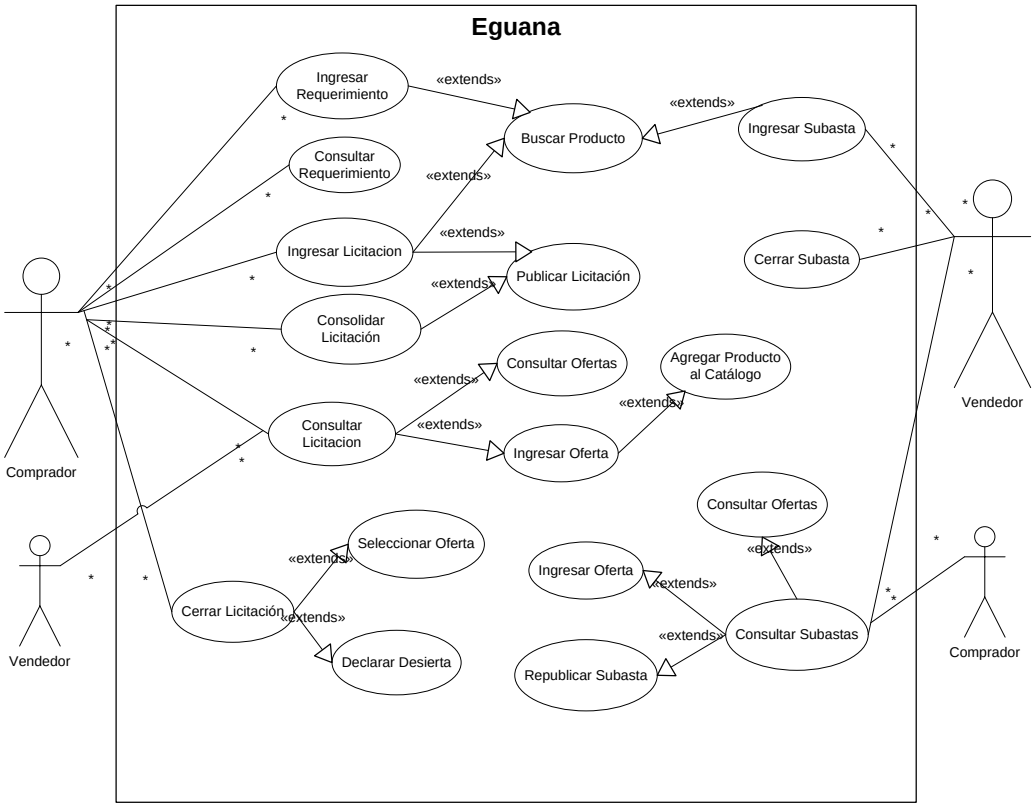

Figura 3-4 Casos de Uso de Licitación y Subastas.

Fuente: E-guana Autor: E-guana

A continuación una descripción de los Casos de Uso:

Requerimientos

Caso de Uso	**Ingresar Requerimiento**
Actores	Comprador.
Suposiciones	El usuario se encuentra logoneado en el sistema y se

	encuentra en la opcion de Requerimiento.
Propósito	Registrar las necesidades de las Unidades de Negocio de una empresa.
Resumen	El comprador busca uno o más productos que requiere y los añade como un requerimiento. Al terminar la transacción, el Requerimiento queda ingresado en la Base de Datos.

Escenarios

Escenario	**Ingreso de Requerimiento exitoso**
Precondición	El usuario se encuentra logoneado en el sistema y se encuentra en la opción de Requerimiento..
Resultado	El usuario ingresa un nuevo Requerimiento.
Descripción	El usuario ha creado un nuevo requerimiento seleccionando de los productos existentes en la base de datos.

Escenario	**Ingreso de Requerimiento fallido**
Precondición	El usuario se encuentra logoneado en el sistema y se encuentra en la opción de Requerimiento.
Resultado	El usuario no puede ingresar un nuevo Requerimiento.
Descripción	El usuario no puede ingresar un nuevo requerimiento ya

que no existen productos que seleccionar en la base de datos.

Caso de Uso	**Buscar Producto**
Actor	Comprador.
Suposiciones	El usuario se encuentra logoneado en el sistema.
Propósito	Realizar una búsqueda de productos por categorías o criterios.
Resumen	El comprador autentificado realiza la consulta de productos para añadirlo como Requerimiento, Licitación o Subasta. Al terminar la transacción, se muestra una lista de productos.

Escenarios

Escenario	**Busqueda Producto exitosa**
Precondición	El usuario se encuentra logoneado en el sistema.
Resultado	Se muestra una lista de productos ordenados alfabéticamente. También pueden ser ordenados por columnas la hacer clic en la cabecera de dicha columna.
Descripción	El comprador realiza una consulta de productos para añadirlo como Requerimiento, Licitación o Subasta.

Caso de Uso	**Consultar Requerimiento**
Actor	Comprador.
Suposiciones	El usuario se encuentra logoneado en el sistema y se encuentra en la opción de Requerimientos.
Propósito	Verificar el listado de requerimientos ingresados en la empresa.
Resumen	El comprador consulta los requerimientos que las unidades de negocio de una empresa han ingresado. La búsqueda puede ser hecha por requerimientos publicados, atendidos, terminados. Al terminar la transacción, se muestra una lista de requerimientos con su respectivo estado.

Escenarios

Escenario	**Consulta de Requerimiento exitosa**
Precondición	El usuario se encuentra logoneado en el sistema y se encuentra en la opción de Requerimientos.
Resultado	Se muestra una lista de requerimientos ordenados alfabéticamente. También pueden ser ordenados por columnas la hacer clic en la cabecera de dicha columna.
Descripción	El comprador realiza una consulta de Requerimientos. Estos requerimientos o algunos de ellos podrán convertirse en Licitación al seleccionar la opción de Consolidar

Licitación.

Licitación

Caso de Uso	**Ingresar Licitación**
Actor	Comprador.
Suposiciones	El usuario se encuentra logoneado en el sistema y se encuentra en la opción de Licitación.
Propósito	Registrar una Licitación.
Resumen	El comprador crea una Licitación seleccionando los productos que van a licitarse, y el tiempo límite para realizar ofertas (caducidad). Al terminar la transacción, la nueva Licitación queda grabada en la Base de Datos.

Escenarios

Escenario	**Ingreso de Licitación exitoso**
Precondición	El usuario se encuentra logoneado en el sistema y se encuentra en la opción de Licitación.
Resultado	El usuario ingresa una nueva Licitación.
Descripción	El usuario ha creado una nueva Licitación seleccionando de los productos existentes en la base de datos.

Escenario	**Ingreso de Licitación fallido**
Precondición	El usuario se encuentra logoneado en el sistema y se

	encuentra en la opción de Licitación.
Resultado	El usuario no puede ingresar un nueva Licitación.
Descripción	El usuario no puede ingresar una nueva Licitación ya que no existen productos que seleccionar en la base de datos.

Caso de Uso	**Publicar Licitación**
Actor	Comprador.
Suposiciones	El usuario se encuentra logoneado en el sistema y se encuentra en la opción de Licitación.
Propósito	Iniciar el proceso de Licitación.
Resumen	El comprador publica la Licitación. Al terminar la transacción, se cambia el estado de la licitación a publicada y se inicia el proceso de licitación.

Escenarios

Escenario	**Publicación de Licitación exitoso**
Precondición	El usuario se encuentra logoneado en el sistema y se encuentra en la opción de Licitación.
Resultado	El usuario ingresa una nueva Licitación.
Descripción	El usuario ha publicado una nueva Licitación seleccionando de los productos existentes en la base de datos.

Caso de Uso	**Consolidar Licitación**
Actor	Comprador.
Suposiciones	El usuario se encuentra logoneado en el sistema y se encuentra en la opción de Licitación.
Propósito	Publicar Licitación en base a Requerimientos ingresados.
Resumen	El comprador consulta los requerimientos ingresados, luego selecciona los que van a ser licitados y la fecha de caducidad de la Licitación. Al terminar la transacción, la Licitación queda publicada y se inicia el proceso de Licitación.

Escenarios

Escenario	**Consolidación de la Licitación Completa**
Precondición	El usuario se encuentra logoneado en el sistema y se encuentra en la opción de Licitación.
Resultado	Se consolida una nueva Licitación
Descripción	Se consolida una nueva Licitación con todos los Requerimientos de todas las unidades de negocio de la empresa que va a licitar.

Escenario	**Consolidación de la Licitación Parcial**
Precondición	El usuario se encuentra logoneado en el sistema y se

	encuentra en la opción de Licitación.
Resultado	Se consolida una nueva Licitación.
Descripción	Se consolida una nueva Licitación seleccionando algunos productos de los Requerimientos por unidades de negocio de la empresa.

Caso de Uso	**Consultar Licitación**
Actor	Comprador, Vendedor.
Suposiciones	El usuario se encuentra logoneado en el sistema y se encuentra en la opción de Licitación.
Propósito	Visualizar las licitaciones y sus respectivos estados.
Resumen	El usuario realiza la búsqueda de las Licitaciones por productos o estados. Al terminar la transacción, se muestra un listado de las Licitaciones consultadas y puede acceder a los detalles de una Licitación.

Escenarios

Escenario	**Consulta de Licitación exitosa**
Precondición	El usuario se encuentra logoneado en el sistema y se encuentra en la opción de Licitación.
Resultado	Se muestra una lista de licitaciones ordenadas por Fecha de publicación. También pueden ser ordenados por

	columnas la hacer clic en la cabecera de dicha columna. Se pueden visualizar los distintos estados de las licitaciones presionando tabs de cada tipo de estado. Se muestra el detalle de la Licitación al darle clic.
Descripción	El comprador realiza una consulta de Licitaciones, se pueden seleccionar los dieferentes estados de las licitaciones.

Caso de Uso	**Consultar Ofertas**
Actor	Comprador, Vendedor
Suposiciones	El usuario se encuentra logoneado en el sistema y se encuentra en la opción de Licitación.
Propósito	Visualizar las Ofertas de una Licitación.
Resumen	El comprador selecciona la Licitación consultada y visualiza las ofertas que ha tenido. Al terminar la transacción, se muestra el listado de ofertas de una Licitación.

Escenarios

Escenario	**Consulta de Ofertas de Licitación exitosa**
Precondición	El usuario se encuentra logoneado en el sistema y se encuentra en la opción de Licitación.
Resultado	Se muestra una lista de oferta de licitaciones ordenadas

	por Fecha de la oferta. También pueden ser ordenados por columnas la hacer clic en la cabecera de dicha columna. Se muestra el detalle de la Oferta de la Licitación al darle clic.
Descripción	El comprador realiza una consulta de las diferentes ofertas que tiene una Licitación.

Caso de Uso	**Ingresar Oferta de una Licitación**
Actor	Vendedor
Suposiciones	El usuario se encuentra logoneado en el sistema y se encuentra en la opción de Licitación.
Propósito	Participar en una Licitación.
Resumen	El vendedor una vez que ha consultado las Licitaciones, sus productos solicitados y sus ofertas, puede participar en una licitación mientras que no haya caducado. Al terminar la transacción, se registra la nueva oferta.

Escenarios

Escenario	**Ingreso de Oferta de Licitación exitoso**
Precondición	El usuario se encuentra logoneado en el sistema, se encuentra en la opción de Licitación, y va a ofertar en una Licitación que este activa. Para poder ofertar debe tener

	esos productos en su catálogo o puede añadirlos a este con la opción añadir producto al catálogo.
Resultado	El usuario selecciona una Licitación a la que va a realizar una Oferta.
Descripción	Se crea una nueva Oferta de Licitación

Escenario	**Ingreso de Oferta de Licitación fallido**
Precondición	El usuario se encuentra logoneado en el sistema, se encuentra en la opción de Licitación y va a ofertar en una Licitación que este activa. Para poder ofertar debe tener esos productos en su catálogo o puede añadirlos a este con la opción añadir producto al catálogo.
Resultado	El usuario no puede ofertar en una Licitación porque no tiene alguno o todos los productos de dicha Licitación en su catálogo de productos.
Descripción	El usuario no puede ofertar en una Licitación

Caso de Uso	**Agregar Producto al Catálogo**
Actor	Comprador, Vendedor.
Suposiciones	El usuario se encuentra logoneado en el sistema y se encuentra en la opción de Licitación.
Propósito	Registrar el producto en el catálogo de una empresa.

Resumen	El comprador puede registrar el producto que está ofertando. Al terminar la transacción, producto queda registrado en el catálogo en el caso de que no exista.

Escenarios

Escenario	**Agregar un nuevo Producto al Catálogo exitosamente**
Precondición	El usuario se encuentra logoneado en el sistema y se encuentra en la opción de Ingreso de Oferta de Licitación.
Resultado	El usuario selecciona un producto que no tiene en su catálogo para añadirlo de manera automática sin tener que ir a las opciones de administración de catálogo. Se puede ingresar los datos de precio, cantidad disponible y una breve descripción del producto.
Descripción	Se añade un nuevo producto al catálogo.

Caso de Uso	**Cerrar Licitación**
Actor	Comprador.
Suposiciones	El usuario se encuentra logoneado en el sistema y se encuentra en la opción de Licitación.
Propósito	Finalizar el proceso de Licitación.
Resumen	El comprador consulta las Licitaciones y termina el proceso seleccionando la Oferta ganadora o declarando desierta la

licitación. Al terminar la transacción, la licitación ya no recibe ofertas, y pasa al estado finalizado o desierto.

Escenarios

Escenario	**Declarar desierta una Licitación**
Precondición	El usuario se encuentra logoneado en el sistema y se encuentra en la opción de mis Licitaciones y se ha cumplido el tiempo tope de la Licitación.
Resultado	El usuario no selecciona ninguna Oferta de Licitación como ganadora y la Licitación es declarada desierta.
Descripción	Se declara desierta una Licitación.

Escenario	**Seleccionar Oferta ganadora en una Licitación**
Precondición	El usuario se encuentra logoneado en el sistema y se encuentra en la opción de mis Licitaciones y se ha cumplido el tiempo tope de la Licitación.
Resultado	El usuario consulta la Licitación y las Ofertas correspondientes. Luego selecciona la oferta ganadora. Al terminar la transacción, la oferta queda seleccionada como ganadora.
Descripción	Se adjudica una Oferta de Licitación ganadora.

Susbasta

Caso de Uso	**Ingresar Subasta.**
Actor	Vendedor.
Suposiciones	El usuario se encuentra logoneado en el sistema y se encuentra en la opción de Subastas.
Propósito	Registrar una nueva Subasta.
Resumen	El vendedor autentificado selecciona el producto que va a Subastar, la fecha tope de duración de la Subasta, puede colocar un precio inicial y la cantidad a subastar. Al terminar la transacción, una nueva Subasta es creada en la base de datos.

Escenarios

Escenario	**Ingreso de Subasta exitoso**
Precondición	El usuario se encuentra logoneado en el sistema y se encuentra en la opción de Subastas.
Resultado	El usuario ingresa una nueva Subasta.
Descripción	El usuario ha creado una nueva Subasta seleccionando de los productos existentes en la base de datos.

Escenario	**Ingreso de Subasta fallido**
Precondición	El usuario se encuentra logoneado en el sistema y se

	encuentra en la opción de Subastas.
Resultado	El usuario no puede ingresar una nueva Subasta.
Descripción	El usuario no puede ingresar una nueva Subasta ya que no existen productos que seleccionar en la base de datos.

Caso de Uso	**Consultar Subasta**
Actor	Comprador, Vendedor.
Suposiciones	El usuario se encuentra logoneado en el sistema y se encuentra en la opción de Subastas.
Propósito	Visualizar las subastas y sus estados.
Resumen	El usuario necesita visualizar las subastas y realiza la búsqueda por productos o estados. Al terminar la transacción, se muestra un listado con las Subastas consultadas y puede acceder a los detalles de las ofertas.

Escenarios

Escenario	**Consulta de Subastas exitosa**
Precondición	El usuario se encuentra logoneado en el sistema y se encuentra en la opción de Subastas.
Resultado	Visualizar las todas las Susbastas activas.
Descripción	El usuario realiza la búsqueda de las Subastas por productos o estados. Al terminar la transacción, se muestra

	un listado de las Subastas consultadas y puede acceder a los detalles de esta al hacer clic sobre ella.

Caso de Uso	**Ingresar Oferta de una Subasta**
Actor	Comprador.
Suposiciones	El usuario se encuentra logoneado en el sistema y se encuentra en la opción de Subastas.
Propósito	Participar en una Subasta.
Resumen	El comprador una vez que consulta las subastas y desea participar en ella, visualiza la subasta e ingresa su precio. Al terminar la transacción, se registra una nueva oferta.

Escenarios

Escenario	**Ingreso de Oferta de Subasta exitoso**
Precondición	El usuario se encuentra logoneado en el sistema, se encuentra en la opción de Subastas y va a ofertar en una Subasta que este activa.
Resultado	El usuario selecciona una Subasta a la que va a realizar una Oferta.
Descripción	Se crea una nueva Oferta de Subasta

Caso de Uso:	**Cerrar Subasta.**
Actor	Vendedor.
Suposiciones	El usuario se encuentra logoneado en el sistema y se encuentra en la opción de Subastas.
Propósito	Finalizar el proceso de Subasta.
Resumen	El vendedor consulta la Subasta y termina el proceso seleccionando la Oferta ganadora o republicando la Subasta. Al terminar la transacción, la Subasta cambia de estado a finalizado o activo.

Escenarios

Escenario	**Republicar Subasta**
Precondición	El usuario se encuentra logoneado en el sistema y se encuentra en la opción de Subastas.
Resultado	Cuando el tiempo de una subasta ha expirado y no se ha elegido a un ofertante ganador, se puede volver a publicar la Subasta.
Descripción	Se publica una nueva Subasta.

Escenario	**Seleccionar Subasta ganadora**
Precondición	El usuario se encuentra logoneado en el sistema y se

encuentra en la opción de Subastas.

Resultado	Cuando el tiempo de una subasta ha expirado y se selecciona entre todos los ofertantes la oferta que mas le convenga al usuario.
Descripción	Se declara una Oferta de Subasta ganadora.

3.4.2 Diagramas E-R

El modelo de datos se lo ha realizado con el tradicional Diagrama Entidad Relación (E-R), que lo hemos dividido en dos diagramas, el primero referente al Sistema E-guana, y el segundo relacionado al módulo de pagos del Sistema de Entidad Financiera.

Diagrama entidad relación de E-Guana

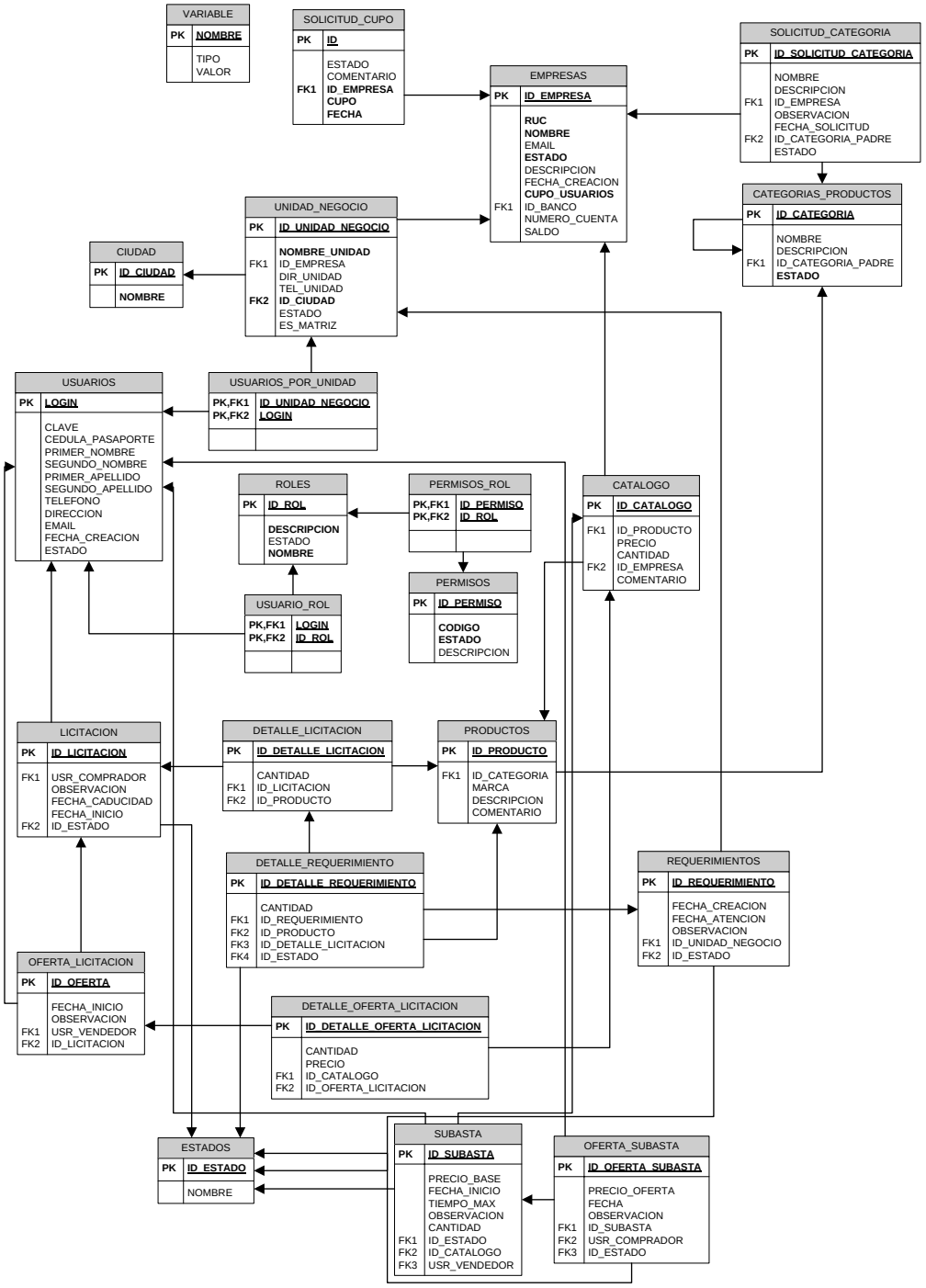

Figura 3-5 Diagrama Entidad-Relación de E-guana.

Fuente: E-guana. Autor: E-guana.

Diagrama entidad relación del módulo de Pagos de Transacciones

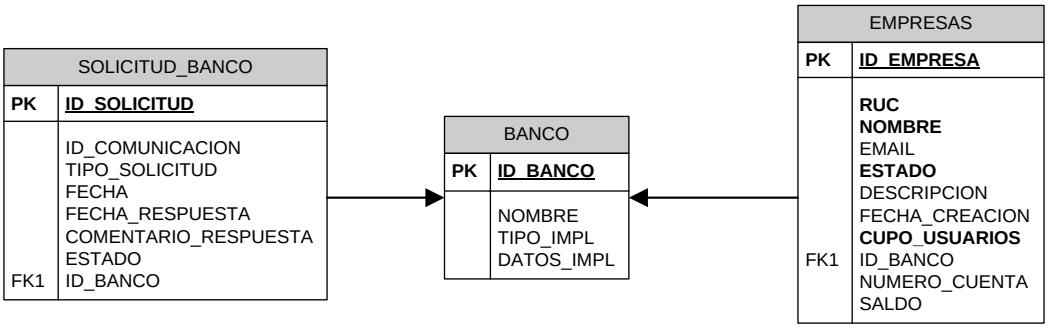

Figura 3-6 Diagrama Entidad-Relación de Módulo de Pagos.

Fuente: E-guana. Autor: E-guana.

En la sección 3.4.6 “Base de Datos” se muestra la descripción de los campos de las tablas.

3.4.3 Diagramas de Clases

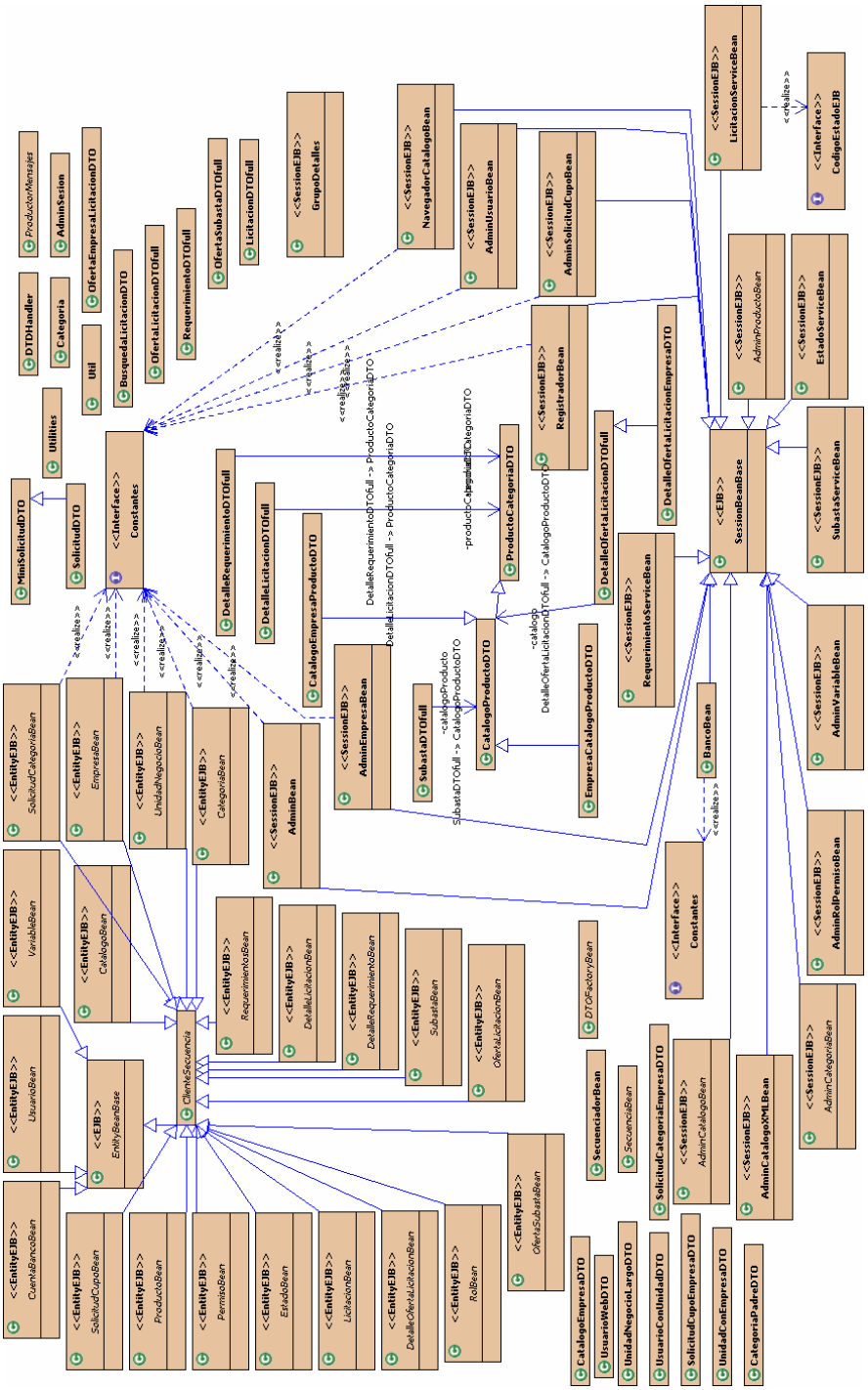

Figura 3-7 Diagrama de Clases.

Fuente: E-guana. Autor: E-guana.

3.4.4 Diagramas de Secuencia.

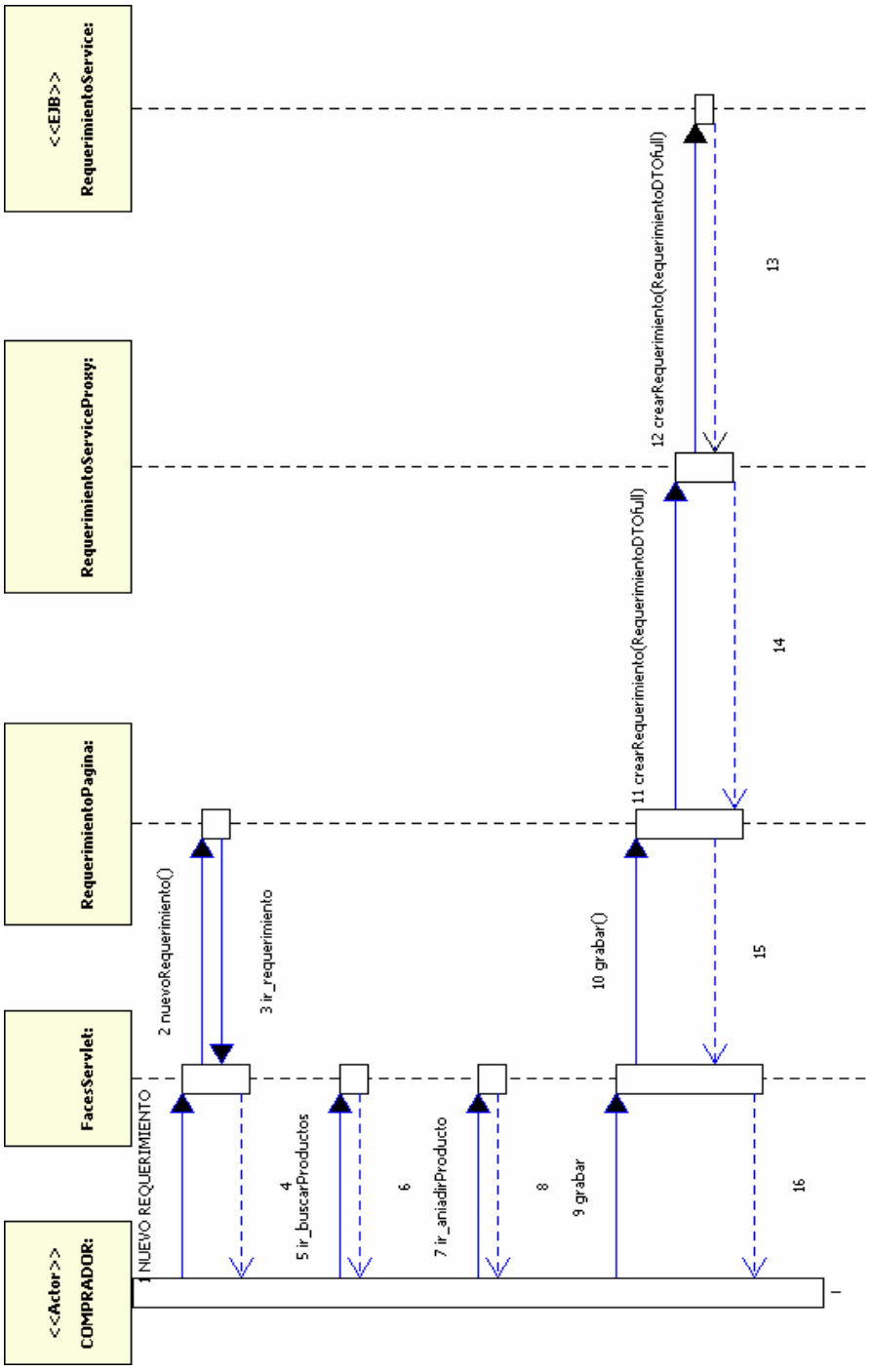

Figura 3-8 Diagrama de Secuencia: Ingreso de Requerimiento.

Fuente: E-guana. Autor: E-guana.

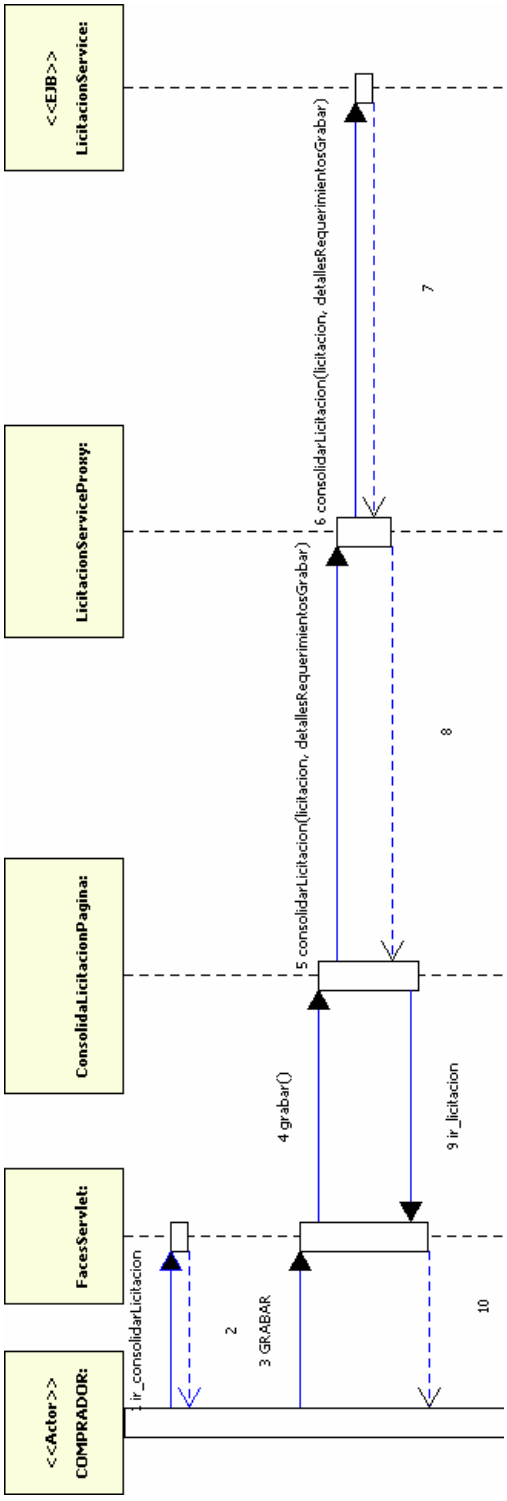

Figura 3-9 Diagrama de Secuencia: Consolidar Requerimiento.

Fuente: E-guana. Autor: E-guana.

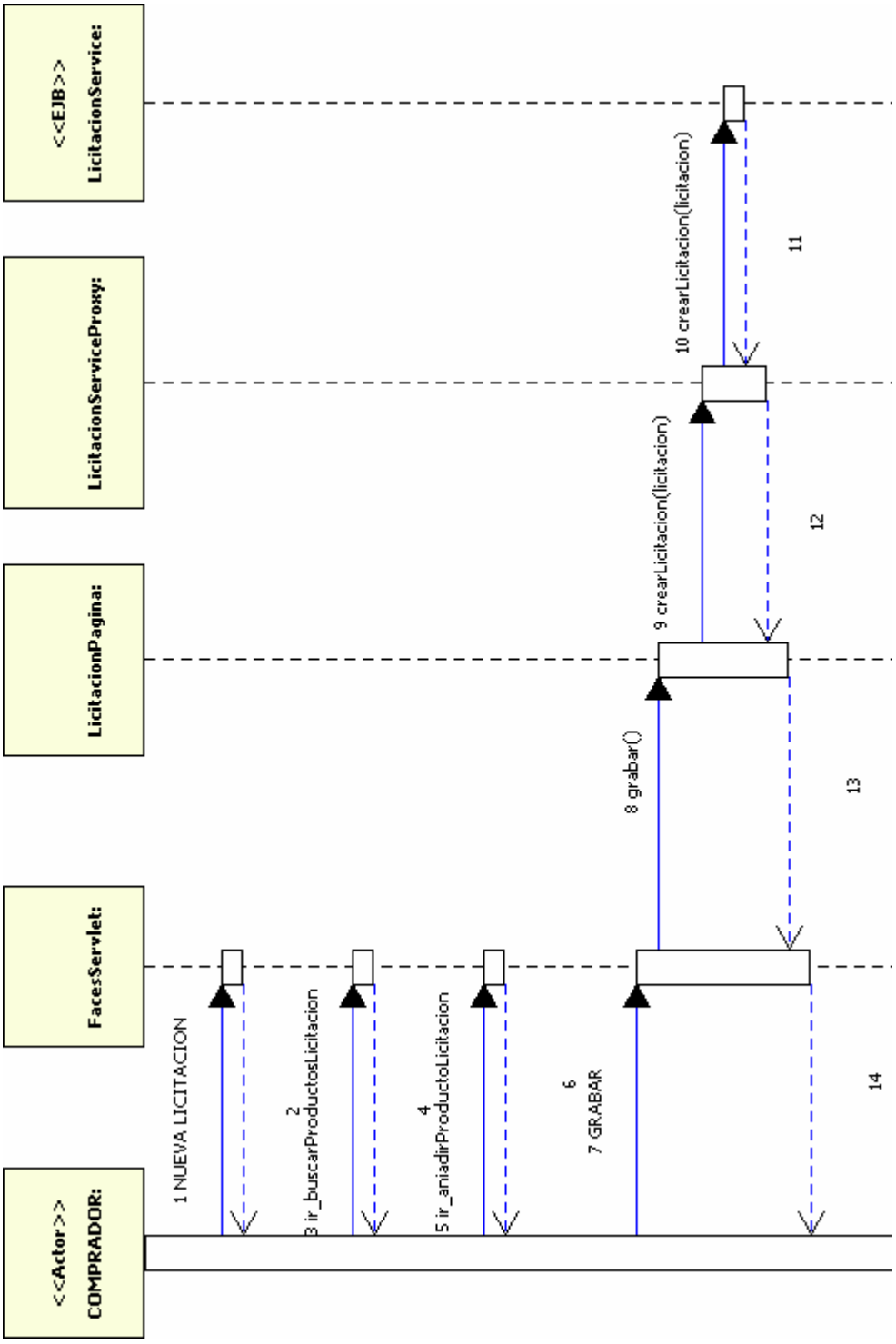

Figura 3-10 Diagrama de Secuencia: Ingreso de Licitación.

Fuente: E-guana. Autor: E-guana.

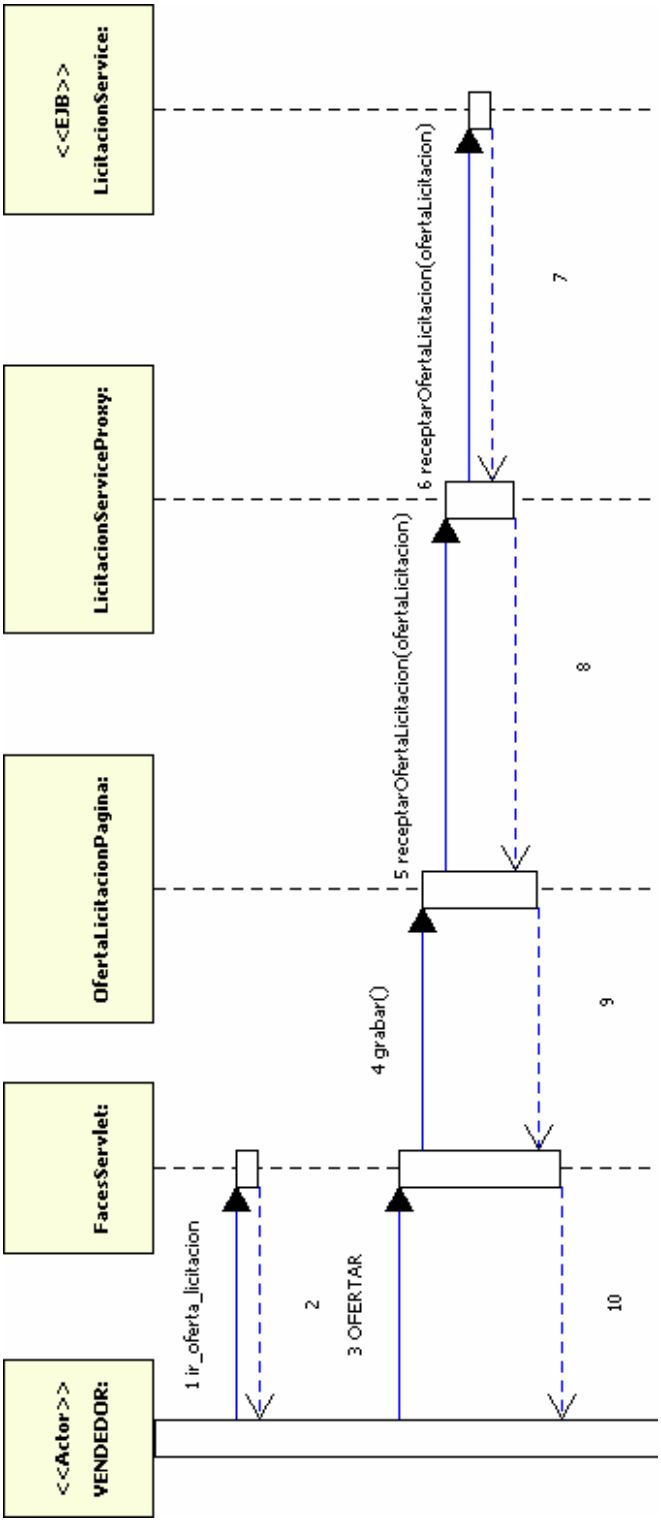

Figura 3-11 Diagrama de Secuencia: Oferta a Licitación.

Fuente: E-guana. Autor: E-guana.

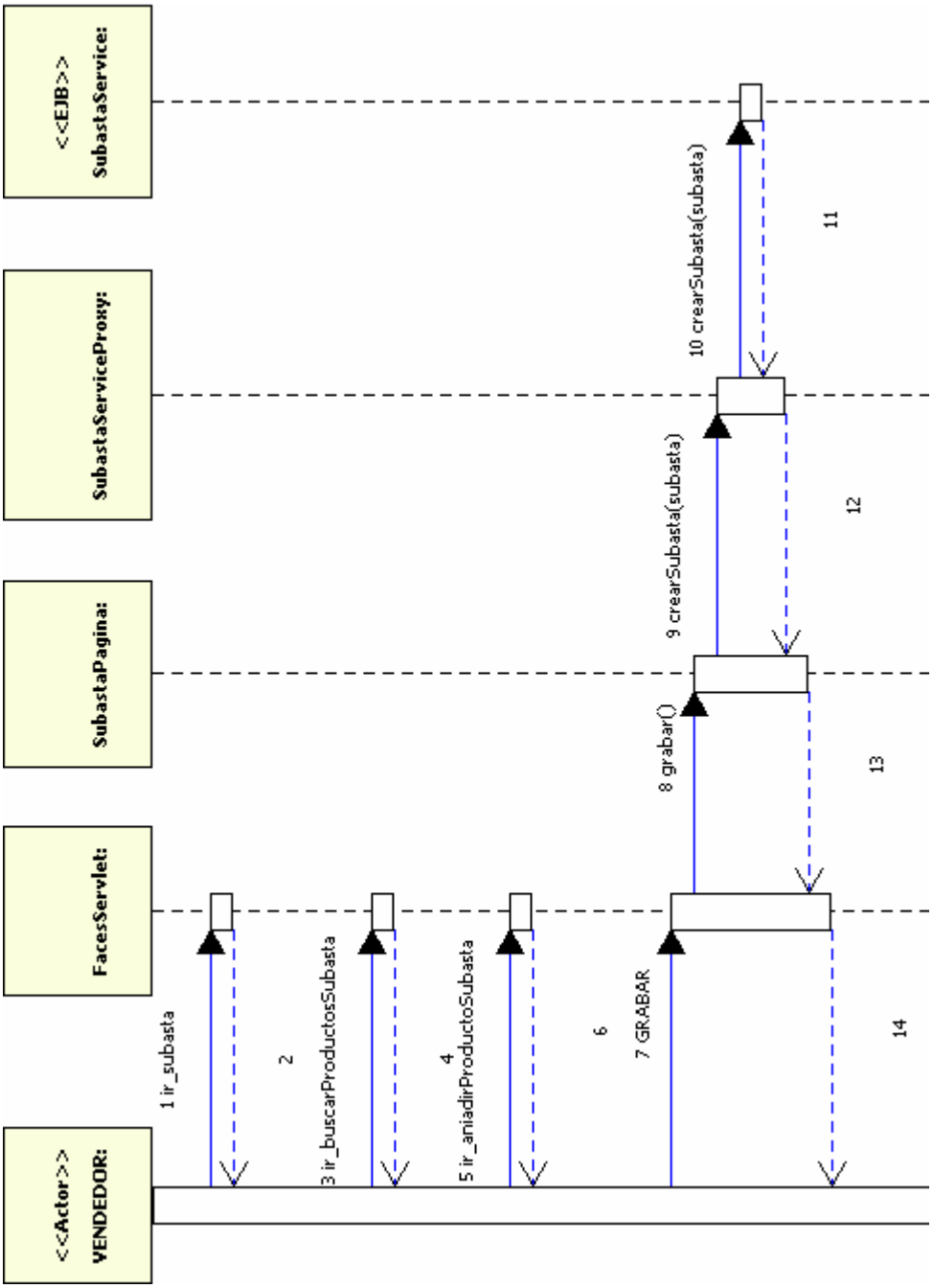

Figura 3-12 Diagrama de Secuencia: Ingreso de Subasta.

Fuente: E-guana. Autor: E-guana.

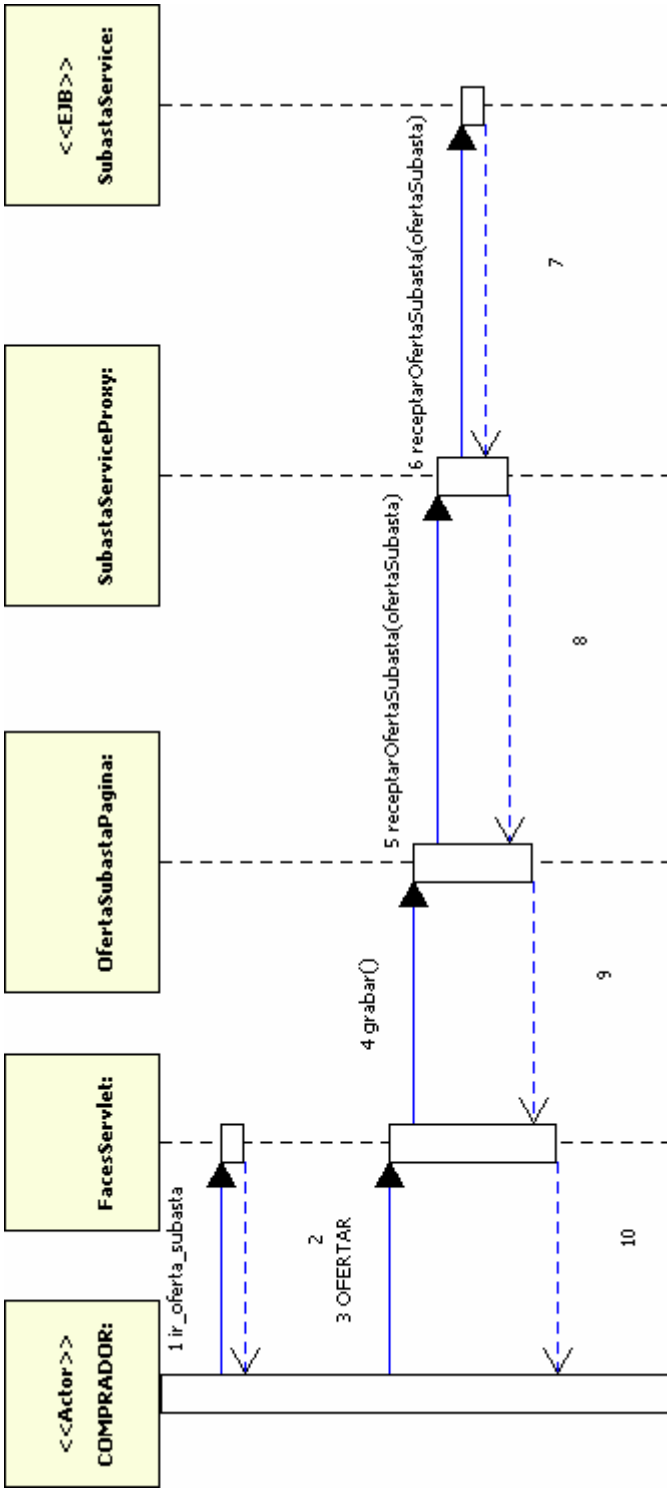

Figura 3-13 Diagrama de Secuencia: Oferta a Subasta.

Fuente: E-guana. Autor: E-guana.

3.4.5 Procesos de Licitación y Subasta.

El siguiente gráfico muestra el proceso de Licitación, el cual inicia con la necesidad de uno o varios productos y finaliza con la selección de la oferta ganadora o declarando desierta la licitación:

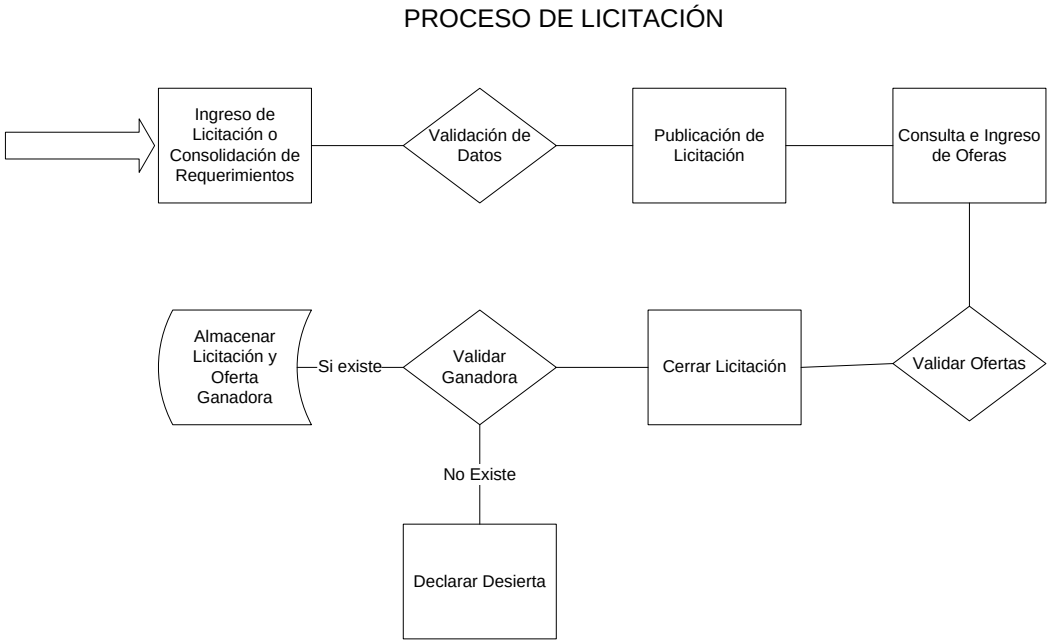

Figura 3-14 Proceso de Licitación.
Fuente: E-guana. Autor: E-guana.

A continuación se muestra el proceso de Subastas, el cual inicia con el Ingreso de la misma y finaliza con la selección de la oferta ganadora:

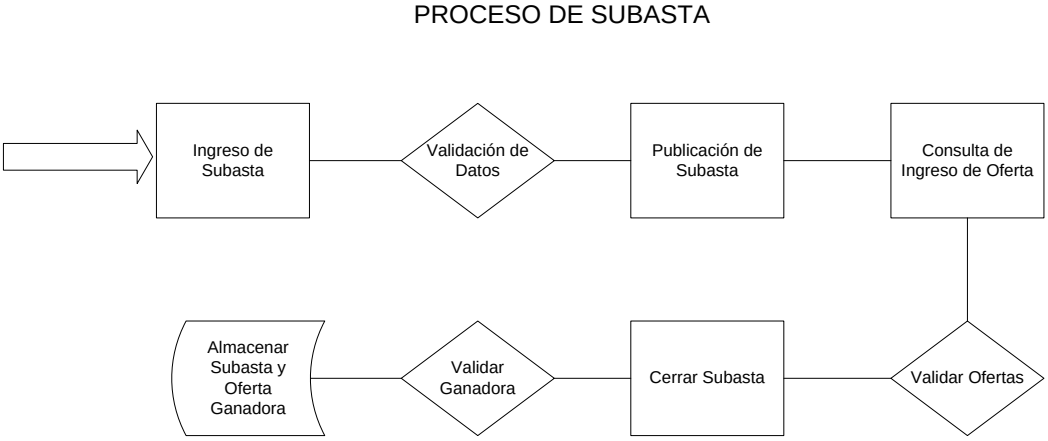

Figura 3-15 Proceso de Subasta.

Fuente: E-guana. Autor: E-guana.

3.4.6 Base de datos.

A continuación mostramos la descripción de los campos de las tablas relacionadas a los módulos de Licitación y Subastas:

Nombre de tabla: **REQUERIMIENTO**				
Nombre	Opcional	Formato	Longitud Máxima	Clave Primaria
ID_REQUERIMIENTO		**Int**	**11**	✔
ID_UNIDAD_NEGOCIO		**Int**	**11**	
FECHA_CREACION		**Date**	**8**	
FECHA_ATENCION	✔	**Date**	**8**	
ID_ESTADO		**Int**	**11**	
OBSERVACION	✔	**Varchar**	**100**	

Tabla 3-2 Descripción de Tabla: REQUERIMIENTO.

Fuente: E-guana. Autor: E-guana.

Nombre de tabla: **DETALLE_REQUERIMIENTO**

Nombre	Opcional	Formato	Longitud Máxima	Clave Primaria
ID_DETALLE_REQUERIMIENTO		**Int**	**11**	✔
ID_REQUERIMIENTO		**Int**	**11**	
ID_PRODUCTO		**Int**	**11**	
CANTIDAD		**Int**	**11**	
ID_DETALLE_LICITACION		**int**	**11**	
ID_ESTADO		**Int**	**11**	

Tabla 3-3 Descripción de Tabla: DETALLE_REQUERIMIENTO.

Fuente: E-guana. Autor: E-guana.

Nombre de tabla: **LICITACION**

Nombre	Opcional	Formato	Longitud Máxima	Clave Primaria
ID_LICITACION		**Int**	**11**	✔
USR_COMPRADOR	✔	**Varchar**	**8**	
OBSERVACION	✔	**Varchar**	**255**	
FECHA_INICIO		**Int**	**11**	
FECHA_CADUCIDAD		**Int**	**11**	
ESTADO		**Char**	**1**	

ID_ESTADO	**Int**	**11**

Tabla 3-4 Descripción de Tabla: LICITACIÓN.

Fuente: E-guana. Autor: E-guana.

Nombre de tabla: **DETALLE_LICITACION**

Nombre	Opcional	Formato	Longitud Máxima	Clave Primaria
ID_ DETALLE_LICITACION		**Int**	**11**	✔
CANTIDAD		**Int**	**11**	
ID_LICITACION		**Int**	**11**	
ID_PRODUCTO		**Int**	**11**	
ID_DETALLE_OFERTA	✔	**Int**	**11**	
ID_ESTADO		**Int**	**11**	

Tabla 3-5 Descripción de Tabla: DETALLE_LICITACIÓN.

Fuente: E-guana. Autor: E-guana.

Nombre de tabla: **OFERTA_LICITACION**

Nombre	Opcional	Formato	Longitud Máxima	Clave Primaria
ID_OFERTA		**Int**	**11**	✔
USR_VENDEDOR		**Varchar**	**8**	
FECHA_INICIO		**Date**	**8**	
ESTADO		**Char**	**1**	

OBSERVACION	✔	**Varchar**	**255**	
ID_LICITACION		**Int**	**11**	

Tabla 3-6 Descripción de Tabla: OFERTA_LICITACION.

Fuente: E-guana. Autor: E-guana.

Nombre de tabla: **DETALLE_OFERTA_LICITACION**				
Nombre	Opcional	Formato	Longitud Máxima	Clave Primaria
ID_DETALLE_OFERTA_LICITACION		**Int**	**11**	✔
CANTIDAD		**Int**	**11**	
PRECIO		**Double**		
ID_CATALOGO	✔	**Int**	**11**	
ID_OFERTA_LICITACION		**Int**	**11**	

Tabla 3-7 Descripción de Tabla: DETALE_OFERTA_LICITACION.

Fuente: E-guana. Autor: E-guana.

Nombre de tabla: **SUBASTA**				
Nombre	Opcional	Formato	Longitud Máxima	Clave Primaria
ID_SUBASTA		**Int**	**11**	✔
USR_VENDEDOR		**Varchar**	**8**	
PRECIO_BASE		**Float**		

PRECIO_RESERVA	✔	**Flota**		
FECHA_INICIO		**Date**	**8**	
TIEMPO_MAX		**Date**	**8**	
OBSERVACION	✔	**Varchar**	**255**	
ESTADO		**Char**	**1**	
ID_ESTADO		**Int**	**11**	
CANTIDAD		**Int**	**11**	
ID_CATALOGO	✔	**Int**	**11**	

Tabla 3-8 Descripción de Tabla: SUBASTA.

Fuente: E-guana. Autor: E-guana.

Nombre de tabla: **OFERTA_SUBASTA**				
Nombre	Opcional	Formato	Longitud Máxima	Clave Primaria
ID_OFERTA_SUBASTA		**Int**	**11**	✔
ID_SUBASTA		**Int**	**11**	
USR_COMPRADOR		**Varchar**	**8**	
PRECIO_OFERTA		**Float**		
ESTADO		**Char**	**1**	
ID_ESTADO		**Int**	**11**	
FECHA		**Date**	**8**	
OBSERVACION	✔	**Varchar**	**100**	

Tabla 3-9 Descripción de Tabla: OFERTA_SUBASTA.

Fuente: E-guana. Autor: E-guana.

Nombre de tabla: **ESTADOS**				
Nombre	Opcional	Formato	Longitud Máxima	Clave Primaria
ID_ESTADO		**Int**	**11**	✔
NOMBRE		**Varchar**	**50**	

Tabla 3-10 Descripción de Tabla: ESTADOS.

Fuente: E-guana. Autor: E-guana.

Nombre de tabla: **SECUENCIA**				
Nombre	Opcional	Formato	Longitud Máxima	Clave Primaria
NOMBRE		**Varchar**	**100**	✔
VALOR		**Int**	**11**	

Tabla 3-11 Descripción de Tabla: SECUENCIA.

Fuente: E-guana. Autor: E-guana.

3.4.7 Interfaz de Usuario

Durante el proyecto E-guana se utilizó una metodología espiral basada en prototipos. Lo cual permitió realizar el diseño de las Interfaces de Usuario en conjunto con la aplicación.

El Desarrollo en Espiral es un modelo de ciclo de vida desarrollado por Barry Boehm en 1985, utilizado generalmente en la Ingeniería de software Las actividades de este modelo son una espiral, cada bucle es una actividad. Las actividades no están fijadas a prioridad, sino que las siguientes se eligen en función del análisis de riesgo, comenzando por el bucle interior. Se basan en 4 tareas: fijar objetivos, análisis de riesgo, desarrollar y planificar [11].

Se tomó mucho en cuenta la Facilidad de Uso, para lo cual JSF nos da muchas ventajas. Cada una de las funcionalidades o servicios expuestos por este módulo pueden ser accedidos de dos forma, la una es el menú ubicado en la parte superior de cada página, la otra es la lista de enlaces de la parte izquierda de cada página.

Figura 3-16 Menú y Lista de Opciones.

Fuente: E-guana. Autor: E-guana.

En la siguiente sección, 3.5 “Implementación”, se muestra el detalle de las funcionalidades del proyecto con sus respectivas Interfaces de Usuario.

3.5 Implementación

A continuación se describirán las funciones de cada una de las páginas que constituyen el módulo en el contexto del servicio que general que prestan, se iniciará con las páginas relacionadas con los requerimientos de productos, luego las utilizadas en todas las etapas de una licitación y por último las páginas referentes a las subastas.

Requerimiento de Productos

Cada Unidad de Negocio de una Empresa puede hacer requerimientos de productos. Para esto se provee el formulario representado en el siguiente gráfico:

Figura 3-17 Página de requerimiento ”requerimiento.jsp”.

Fuente: E-guana. Autor: E-guana.

Cada requerimiento permite el ingreso de una descripción y el listado de productos requeridos. Así como decidir si desea publicar el requerimiento o dejarlo tan solo registrado para agregar más productos en otro momento. (Figura 3-17)

Para seleccionar los productos de interés el usuario debe presionar el botón “Buscar Productos”, lo cual le presentará la página para búsqueda de productos.

Figura 3-18 Página para buscar productos para un requerimiento "buscarProductos.jsp".

Fuente: E-guana. Autor: E-guana.

La misma que permite la búsqueda de productos tanto de acuerdo a la categoría a la que pertenece el producto(Figura 3-18) como de acuerdo a un criterio de búsqueda, pudiendo ser descripción del producto, la categoría o la marca.

Figura 3-19 Búsqueda de productos por criterio.

Fuente: E-guana. Autor: E-guana.

En ambos casos el resultado de la búsqueda muestra una lista de los productos que concuerdan con la búsqueda, además de un enlace para agregar dicho producto al requerimiento. Dicho enlace presenta la página representada en la gráfica siguiente:

Figura 3-20 Agregar producto a requerimiento "aniadirProducto.jsp".

Fuente: E-guana. Autor: E-guana.

En dicho formulario el usuario puede ingresar la cantidad requerida del producto escogido, luego de lo cual debe presionar el botón “Aceptar” para regresar a la página del requerimiento para proseguir a la publicación del mismo.

En la página de requerimiento y luego de presionar el botón “Guardar” se muestran los datos completos del requerimiento pero en modo de lectura.

Figura 3-21 Datos de un requerimiento publicado

Fuente: E-guana. Autor: E-guana.

El usuario que lo requiera podrá consultar los requerimientos de su empresa, para lo cual se tiene la página “consultarRequerimientos.jsp”

Figura 3-22 Consultar requerimientos "consultarRequerimientos.jsp"

Fuente: E-guana. Autor: E-guana.

Esta página permite visualizar los requerimientos de acuerdo a su estado, para lo cual se tienen enlaces presentados en forma de solapas para cada uno de los estados que puede tener un requerimiento. De cada requerimiento se muestra el código, la fecha de publicación, la fecha de atención (estos dos atributos en caso de que corresponda), la descripción y el estado. Además se presentan enlaces para ver, modificar o borrar el requerimiento. Por último cada encabezado de columnas que presentan datos tiene un enlace que permite el ordenamiento de la lista de acuerdo a la respectiva columna.

Para una búsqueda más precisa se permite la búsqueda por criterio, pudiendo ser éste el producto contenido en el requerimiento, la descripción, la fecha de publicación o el código.

Todo requerimiento en estado de “Registrado” puede ser modificado, para lo cual se puede hacer uso del enlace la lista de consulta directamente o presionando el botón “Modificar” que se presenta en la página que muestra los datos del requerimiento.

Licitación

Una licitación puede crearse de dos formas: ingresando el requerimiento realizado por una persona o unidad de negocio; o por consolidación de los requerimientos hechos por varias unidades de negocio.

Para la publicación directa de una licitación, se proporciona el siguiente formulario:

Figura 3-23 Formulario de licitación "licitacion.jsp"

Fuente: E-guana. Autor: E-guana.

Como referencia se permite el ingreso de la descripción, la fecha de inicio, la fecha de caducidad, los productos y por último la decisión de publicar o no la licitación. Cuando se crea una licitación la fecha de inicio debe tener un valor superior a la fecha actual del sistema.

Para la fecha de inicio y la fecha de caducidad se utiliza un componente que permite la selección de la fecha por medio de un componente gráfico para evitar el ingreso de fechas equivocadas o en formato erróneo.

Figura 3-24 Componente para ingreso de fecha

Fuente: E-guana. Autor: E-guana.

Cuando se presiona el botón “Buscar Productos” se presenta la página “buscarProductosLicitacion.jsp”, que tan solo se diferencia de “buscarProductos.jsp” por el destino del enlace “Agregar” de la lista de productos encontrados, que es en este caso “aniadirProductoLicitacion.jsp”, esta última a su vez es similar a “aniadirProducto.jsp” pero cuyo botón “Aceptar” redirige a la página de la licitación.

Para crear una licitación a partir de requerimientos hechos por las unidades de negocio se tiene el formulario siguiente:

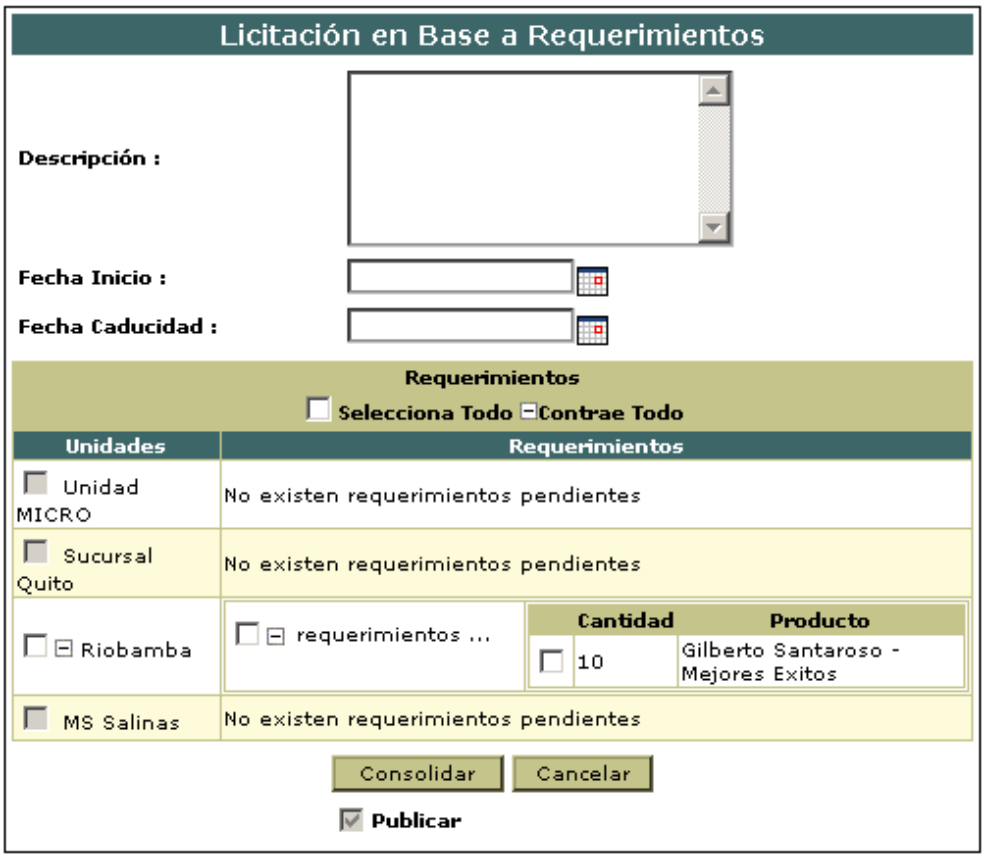

Figura 3-25 Página de consolidación de una licitatación en base a requerimientos "consolidarLicitacion.jsp"

Fuente: E-guana. Autor: E-guana.

La diferencia de este formulario con el anterior es que los productos se seleccionan a partir de los requerimientos que han hecho las unidades que pertenecen a la empresa, para esto se muestra la lista de unidades, junto a cada una, se listan los requerimientos pendientes, en caso de que existan, para finalmente mostrar los productos y cantidad que conforman cada requerimiento. Para facilitar la presentación de estos datos el usuario puede expandir o contraer los datos de cada unidad y requerimiento.

El usuario puede seleccionar un producto individual y un requerimiento, todos los requerimientos de una unidad o todos los requerimientos de todas las unidades.

Una vez que el usuario esté satisfecho con la selección debe presionar el botón “Consolidar”, con lo que el proceso de consolidación estará terminado y se publicará la licitación.

La consulta de las licitaciones se hace a través de la página “consultarLicitaciones.jsp”:

Figura 3-26 Página de consultas de licitaciones “consultarLicitaciones.jsp”

Fuente: E-guana. Autor: E-guana.

Esta página permite buscar licitaciones por estado, producto, descripción, fecha de inicio, fecha de caducidad y código de la licitación.

Los datos que se muestran de la licitación son el código, la fecha de inicio, la fecha de caducidad, la descripción, el número de ofertas recibido y el estado.

Además se presentan enlaces para ver, modificar y eliminar la licitación, siempre y cuando tenga el rol adecuado.

Para facilitar la discriminación entre licitaciones que pueden ser de interés para la empresa del usuario actualmente conectado, se presenta la imagen  junto al código de la licitación si es que dicha licitación contiene algún producto de los existentes en el catálogo de su empresa.

Puesto que en el listado de licitaciones se muestran todas las licitaciones almacenadas en el sistema, para distinguir entre aquellas que pertenecen a otras empresas, se ubica la imagen  junto a los datos de la licitación publicada por el usuario.

Cuando se presiona el enlace para ver la licitación, se presentan junto a los datos de la licitación, el listado de las ofertas que ha recibido la misma:

Ofertas

= Algun producto fue comprado

Fecha Creación	Descripción	Productos ofertados	Precio	Vendedor	
18 enero 2006		1	¤0,00	wramos	Ver Oferta
18 enero 2006	A usted se los dejo en este buen precio	1	¤200,00	wramos	Ver Oferta
18 enero 2006	A usted se los dejo en este buen precio	1	¤180,00	wramos	Ver Oferta
18 enero 2006	otra buena oferta	1	¤180,00	wramos	Ver Oferta
18 enero 2006		1	¤200,00	wramos	Ver Oferta
18 enero 2006		1	¤100,00	wramos	Ver Oferta
18 enero 2006		1	¤150,00	wramos	Ver Oferta

7 ofertas recuperadas.

Figura 3-27 Lista de ofertas a una licitación propia

Fuente: E-guana. Autor: E-guana.

Información de la Licitación

Código : 464
Descripción : Primero consolidada en Calypso
Fecha Inicio : 25 enero 2006
Fecha Caducidad : 27 enero 2006
Estado : Publicado
Comprador : guerrero

Productos Seleccionados

Producto	Categoría	Marca	Cantidad
tejas eternit	Materiales Construcc		1

Un productos recuperado.

Ir a Consulta Hacer Oferta

Ofertas

Fecha Creación	Descripción	Productos ofertados	Precio	Vendedor	
11 febrero 2006	Son de la mejor calidad	1	¤80,00	fdomingu	Ver Oferta

Una ofertas recuperada.

Figura 3-28 Lista de oferta a una licitación de otra empresa

Fuente: E-guana. Autor: E-guana.

Como se puede ver en las gráficas, la página de la licitación muestra componentes diferentes dependiendo de si la persona que ve la página es dueña o no de la licitación. Cuando se trata de una licitación de otra empresa, se presenta el botón “Hacer Oferta”, el mismo que muestra la página para el ingreso de los datos de la oferta. Este botón no se muestra para una licitación que ha caducado.

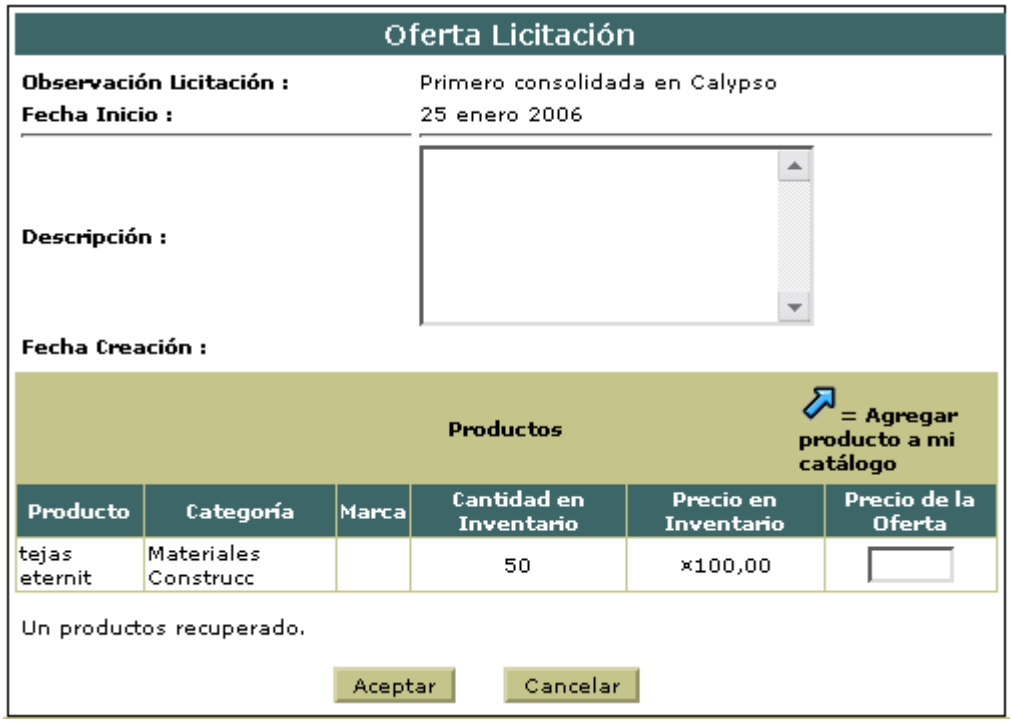

Figura 3-29 Formulario oferta a licitación

Fuente: E-guana. Autor: E-guana.

El formulario representado permite el ingreso de la descripción y el precio del producto ofertado. Como referencia se muestra el precio del producto en el catálogo.

Si el producto solicitado no se encuentra en el catálogo es posible ingresarlo directamente en esta página a través de un formulario del catálogo resumido que se presenta al ubicar el cursor sobre la imagen que se encuentre junto al producto que se desea agregar, este formulario se presenta en la siguiente imagen:

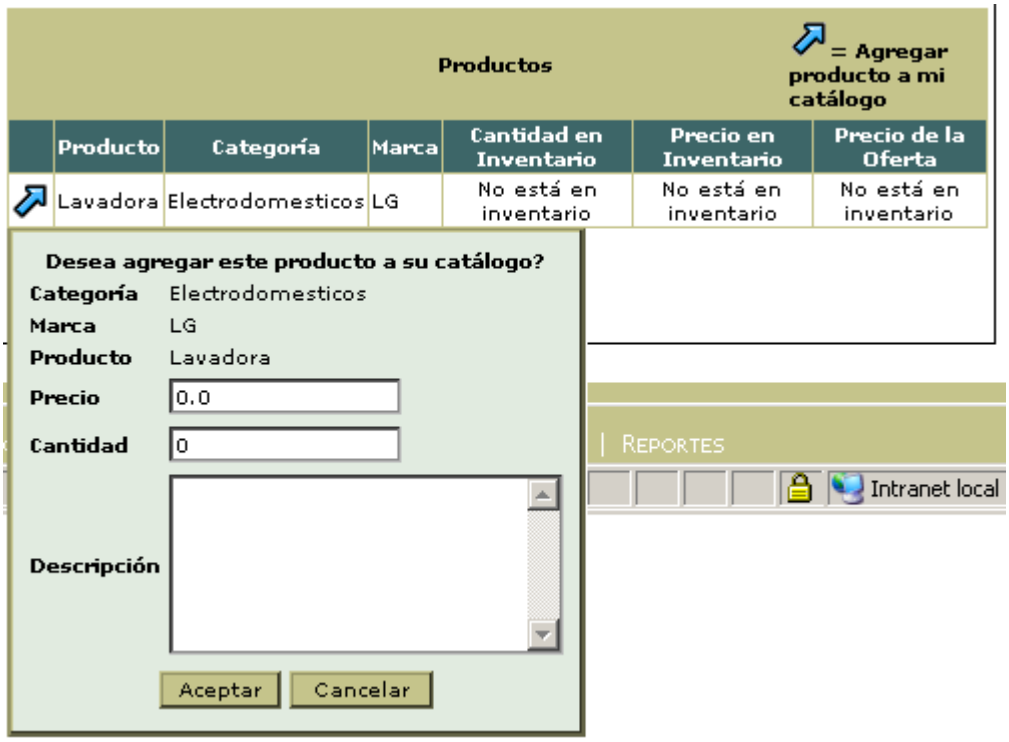

Figura 3-30 Agregar producto al catálogo.

Fuente: E-guana. Autor: E-guana.

Para ver los datos de una oferta se utiliza el botón “Ver Oferta” en el listado de ofertas de la licitación:

Figura 3-31 Datos de una oferta de licitación "ofertaLicitacion.jsp"

Fuente: E-guana. Autor: E-guana.

Si la oferta fue aceptada, se muestra junto al precio del producto el texto "Producto comprado", además en el listado de ofertas se presenta una imagen para destacar este hecho.

Cuando una licitación ha caducado el usuario que publicó la licitación puede proceder a seleccionar la oferta, para esto se le presenta el botón "Selección de Oferta", que lo redirige a la siguiente página:

Figura 3-32 Selección de ofertas "seleccionOfertaLicitacion.jsp"

Fuente: E-guana. Autor: E-guana.

Para seleccionar la oferta el usuario tiene la opción de escoger productos de una o varias ofertas.

Junto a los datos del producto se muestra un componente de selección que al activarse envía un requerimiento al servidor para buscar entre todas las ofertas recibidas, aquellas que contengan los productos que acaban de ser activados. Por defecto estos componentes de selección están activados, por lo tanto la primera vez se retornarán todas las ofertas que contengan todos los productos, en otras palabras las ofertas completas.

El resultado de este pedido es una lista de ofertas indicando el nombre de la empresa ofertante, la fecha de la oferta y el precio total del o los productos.

Si el precio total de los productos es el menor entre las ofertas obtenidas se muestra un mensaje indicando “Mejor Precio”.

Si el usuario está satisfecho con una oferta debe presionar el botón “Seleccionar”, para pasar los productos de la oferta a la lista de productos seleccionados, esta acción puede ser deshecha a través del botón “Quitar” que se muestra junto al nombre del ofertante seleccionado, para comprender mejor vea el gráfico siguiente:

Figura 3-33 Oferta seleccionada

Fuente: E-guana. Autor: E-guana.

Como se puede observar, ahora junto al producto se muestra el nombre de la empresa que hace la oferta del mismo, junto con el precio, además el componente para solicitar la búsqueda del producto entre las ofertas se desactiva puesto que ya no puede ser buscado ya que ha sido seleccionado.

Si el usuario ha completado el proceso de selección debe presionar el botón “Aceptar”. Cuando ninguna oferta satisface las demandas del comprador, puede declarar la licitación como desierta activando el componente etiquetado como “Desierta”.

Para facilitar la búsqueda de licitaciones se proveen dos vistas adicionales, la una es “Mis Licitaciones” que lista únicamente licitaciones pertenecientes al usuario que las ha creado mientras que la otra es “Donde he ofertado” que presenta todas las licitaciones donde el usuario ha hecho al menos una oferta.

Inicio | Requerimiento | Licitación | Subastas

Mis Licitaciones — Nueva

Todos | Registrado | **Publicado** | Atendido | Caducado | Desierta

Buscar Por: Producto [ ] Buscar

= Ver = Modificar = Eliminar

Código	Fecha Inicio	Fecha Caducidad	Descripción	Número de Ofertas	Estado	Opciones
347	29 agosto 2006	29 septiembre 2006	Nueva Cosolidada	0	Publicado	
367	29 agosto 2006	21 septiembre 2006		0	Publicado	

2 licitaciones recuperados.

Figura 3-34 Mis Licitaciones

Fuente: E-guana. Autor: E-guana.

Inicio | Requerimiento | Licitación | Subastas

Licitaciones donde he ofertado

= Ver

Código	Fecha Inicio	Fecha Caducidad	Descripción	Número de Ofertas	Estado	Opciones
304	17 enero 2006	31 enero 2006	es un buen DVD	4	Atendido	
305	17 enero 2006	31 enero 2006	Son dos productos	8	Caducado	
464	25 enero 2006	27 enero 2006	Primero consolidada en Calypso	2	Publicado	
481	16 febrero 2006	28 febrero 2006	son 4 productos	1	Caducado	

4 licitaciones recuperadas.

Figura 3-35 Licitaciones donde he ofertado.

Fuente: E-guana. Autor: E-guana.

Subasta

La funcionalidad de subastas permite al usuario “Vendedor” poner a la venta un producto de su catálogo en forma de subasta mientras que al “Comprador” se le permite participar en dicha subasta para adquirir el producto.

El formulario para la creación de la subasta es el siguiente:

Figura 3-36 Ingreso de Subasta: “subasta.jsp”.

Fuente: E-guana. Autor: E-guana.

Este formulario permite el ingreso de una descripción, la fecha tope de la subasta, el precio base, el producto y la cantidad del mismo a subastar.

Sólo se puede subastar un producto a la vez pero la cantidad o ítems del mismo pueden ser mayores a 1.

Debido a que el usuario subasta productos de su catálogo, la página "buscarProductosSubasta.jsp" en lugar de mostrar todos los productos y todo el árbol de categorías de productos, sólo se muestran las concernientes al catálogo de la empresa del usuario:

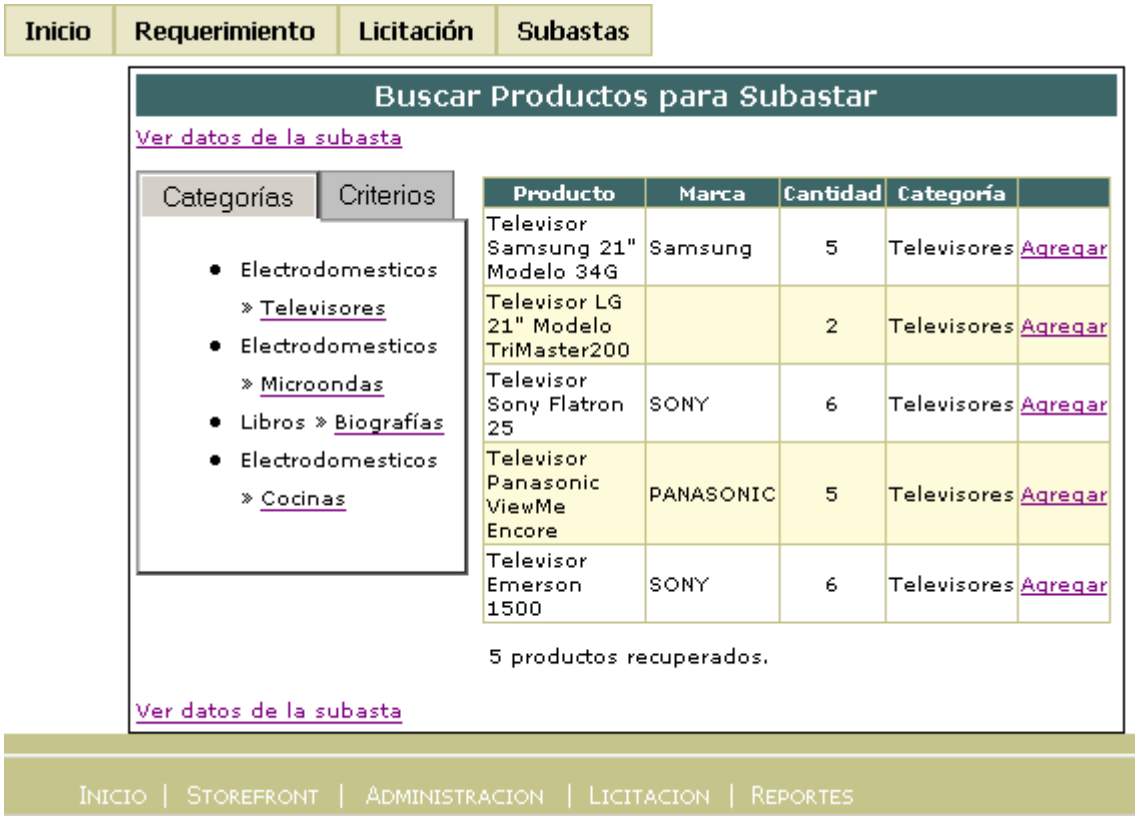

Figura 3-37 Ingreso de Subasta: "buscarProductosSubasta.jsp".

Fuente: E-guana. Autor: E-guana.

Para seleccionar el producto a subastar se debe presionar el enlace “Agregar”, con esto se redirigirá a la página con los datos de la subasta pero ahora con los datos del producto:

Figura 3-38 Datos de la subasta con producto seleccionado.
Fuente: E-guana. Autor: E-guana.

Luego el usuario debe ingresar la fecha tope de la subasta, el precio base y la cantidad de ítems del producto subastado, una vez que esté conforme con los datos debe presionar el botón “Guardar” para que la subasta se

registre en el sistema y puedan hacer las ofertas los usuarios compradores.

Para consultar las subastas existentes en E-Guana se proporciona la página “consultarSubastas.jsp”:

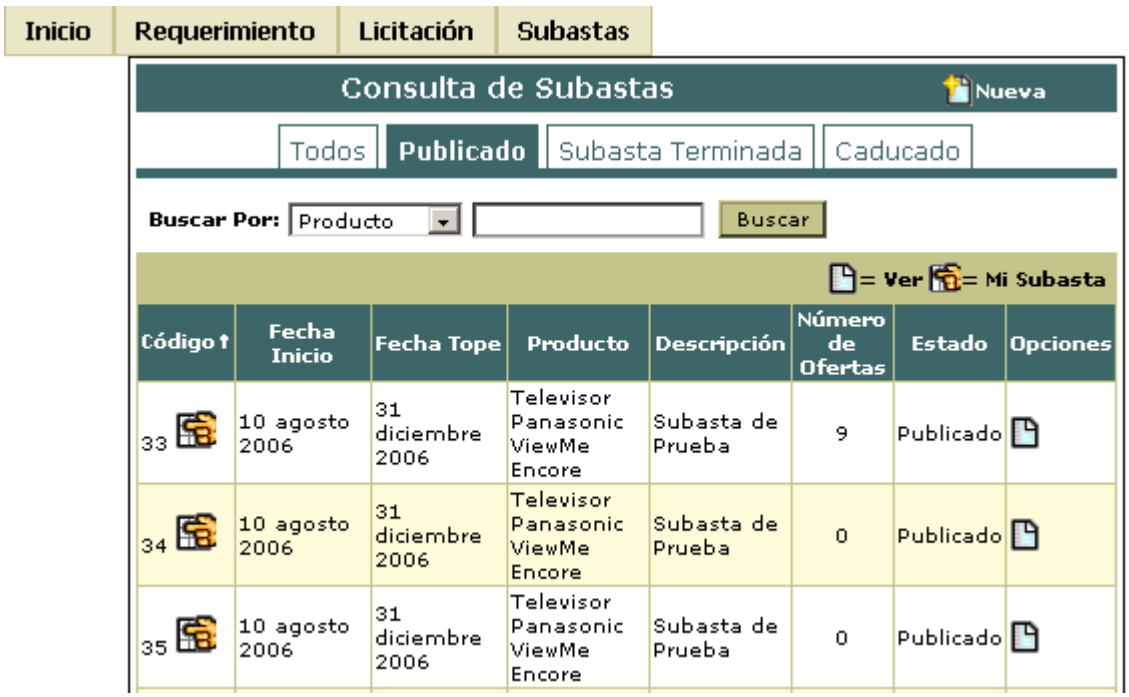

Figura 3-39 Consulta de Subasta: “consultarSubastas.jsp”.

Fuente: E-guana. Autor: E-guana.

Las subastas se pueden listar de acuerdo al estado y por el producto subastado o la descripción de la subasta.

Los datos que se muestran en la lista son el código, fecha de inicio, fecha tope, descripción y estado de la subasta, así como la descripción del producto subastado y el número de ofertas recibido.

Además en el listado se indica si fue hecha por el usuario destacándolo con la imagen  junto al código de la subasta.

Para hacer una oferta a una subasta se debe presionar el botón “Hacer Oferta” que se muestra en el formulario de la subasta si es que la misma no ha caducado ni terminado.

Figura 3-40 Consulta Formulario oferta a subasta.

Fuente: E-guana. Autor: E-guana.

Este formulario permite el ingreso del precio de la oferta y una descripción que se crea necesaria.

Las ofertas presentadas a una subasta se muestran en una lista bajo los datos de la subasta:

Ofertas				
Precio de la Oferta	**Fecha**	**Descripción**	**Comprador**	
×2.200,00 **Mejor Precio**	15 agosto 2006	Oferta Subasta de Prueba	wramos	Ver Oferta
×2.200,00 **Mejor Precio**	15 agosto 2006	Oferta Subasta de Prueba	wramos	Ver Oferta
×2.200,00 **Mejor Precio**	15 agosto 2006	Oferta Subasta de Prueba	wramos	Ver Oferta
×2.200,00 **Mejor Precio**	15 agosto 2006	Oferta Subasta de Prueba	wramos	Ver Oferta
×2.200,00 **Mejor Precio**	15 agosto 2006	Oferta Subasta de Prueba	wramos	Ver Oferta
×2.200,00 **Mejor Precio**	15 agosto 2006	Oferta Subasta de Prueba	wramos	Ver Oferta
×2.200,00 **Mejor Precio**	15 agosto 2006	Oferta Subasta de Prueba	wramos	Ver Oferta
×2.200,00 **Mejor Precio**	15 agosto 2006	Oferta Subasta de Prueba	wramos	Ver Oferta
×1.000,00	15 agosto 2006	Oferta Subasta de Prueba	wramos	Ver Oferta

9 ofertas recuperadas.

Figura 3-41 Ofertas de una subasta en proceso.

Fuente: E-guana. Autor: E-guana.

En la lista de ofertas se muestran el precio de la oferta, la fecha en la que fue publicada, la descripción y el usuario comprador, además se indica junto al precio de la oferta si es el mejor precio ofrecido en caso de ser el mayor.

Si se trata de una subasta que ha terminado, el listado de ofertas destaca la oferta ganadora:

Ofertas — = Oferta Ganadora

Precio de la Oferta	Fecha	Descripción	Comprador
×25,00	12 enero 2006		wramos
×0,00	12 enero 2006		wramos

2 ofertas recuperadas.

Figura 3-42 Ofertas de una subasta terminada.

Fuente: E-guana. Autor: E-guana.

Para ver los datos de la oferta se debe presionar el botón “Ver Oferta” (siempre y cuando la subasta no haya terminado), esto presentará la siguiente página:

Datos de la Oferta

Código :	33
Descripción :	Subasta de Prueba
Fecha Publicación :	10 agosto 2006
Fecha Tope :	31 diciembre 2006
Estado :	Publicado
Producto :	Televisor Panasonic ViewMe Encore
Categoría :	Televisores
Marca :	PANASONIC
Cantidad :	10
Precio Base :	×1.000,00
Fecha Creación :	15 agosto 2006
Precio de la Oferta :	×2.200,00 **Mejor Precio**
Descripción :	Oferta Subasta de Prueba

Ver datos de la subasta | Aceptar Oferta

Figura 3-43 Datos de Oferta: “ofertaSubasta.jsp”.

Fuente: E-guana. Autor: E-guana.

La aceptación de una oferta se la hace a través del formulario de la oferta de la subasta cuando es abierta por el usuario que publicó la subasta y presionando el botón “Aceptar Oferta”, esto pasará la subasta al estado “Terminada”.

Cuando una oferta llega a la fecha tope establecida, el sistema automáticamente la pasa al estado “Caducada”, por lo que no se podrán recibir más ofertas, pero el sistema da la libertad de republicar la subasta, para esto se debe presionar el botón “Republicar” que se muestra en el formulario de una subasta caduca, lo cual permite la edición de los datos del subasta.

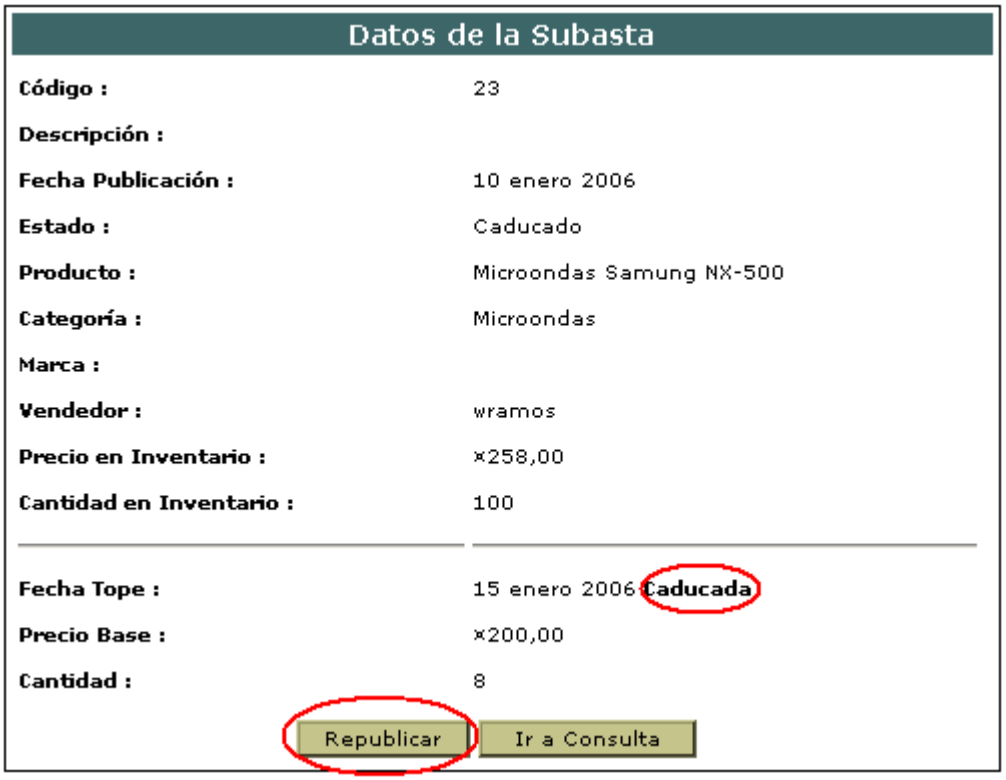

Figura 3-44 Subasta caducada.

Fuente: E-guana. Autor: E-guana.

Para facilitar la búsqueda de subastas se proveen dos vistas adicionales al listado sencillo de las subastas, la primera se denomina “Mis Subastas”, figura 3-45 y la segunda se denomina “Donde he ofertado” figura 3-46..

Código	Fecha Inicio	Fecha Tope	Producto	Descripción	Número de Ofertas	Estado	Opciones
33	10 agosto 2006	31 diciembre 2006	Televisor Panasonic ViewMe Encore	Subasta de Prueba	9	Publicado	
34	10 agosto 2006	31 diciembre 2006	Televisor Panasonic ViewMe Encore	Subasta de Prueba	0	Publicado	
35	10 agosto 2006	31 diciembre 2006	Televisor Panasonic ViewMe Encore	Subasta de Prueba	0	Publicado	
36	10 agosto 2006	31 diciembre 2006	Televisor Panasonic ViewMe Encore	Subasta de Prueba	0	Publicado	
37	10 agosto 2006	31 diciembre	Televisor Panasonic ViewMe	Subasta de Prueba	0	Publicado	

Figura 3-45 Mis Subastas.

Fuente: E-guana. Autor: E-guana.

En “Mis Subastas” sólo presenta las subastas publicadas por el usuario.

Subastas donde he ofertado

= Ver

Código	Fecha Inicio	Fecha Tope	Producto	Descripción	Número de Ofertas	Estado	Opciones
3	22 diciembre 2005	29 diciembre 2005	Microondas Samsung NX-700		2	Caducado	
14	03 enero 2006	08 enero 2006	Televisor Panasonic ViewMe Encore		1	Subasta Terminada	
19	07 enero 2006	12 enero 2006	Microondas Samung NX-500		7	Subasta Terminada	
23	10 enero 2006	15 enero 2006	Microondas Samung NX-500		1	Caducado	
24	10 enero 2006	15 enero 2006	Televisor Samsung 21" Modelo 34G		4	Subasta Terminada	
27	13 enero 2006	25 enero 2006	Televisor Samsung 21" Modelo 34G	mejorado el precio	2	Subasta Terminada	

Figura 3-46 Subastas donde he ofertado.

Fuente: E-guana. Autor: E-guana.

En “Subastas donde he ofertado” sólo se muestran los datos de las subastas donde el usuario ha hecho al menos una oferta.

En estas dos últimas opciones, se encierra un alto potencial de consulta para el usuario conectado al sistema ya que ambos visores de información fueron diseñados pensando en la necesidad de saber cuales son las ofertas y subastas creadas cubriendo las dos papeles en el sistema: crear subastas y ofertar en una de ellas.

3.5.1 Objetos del Modelo

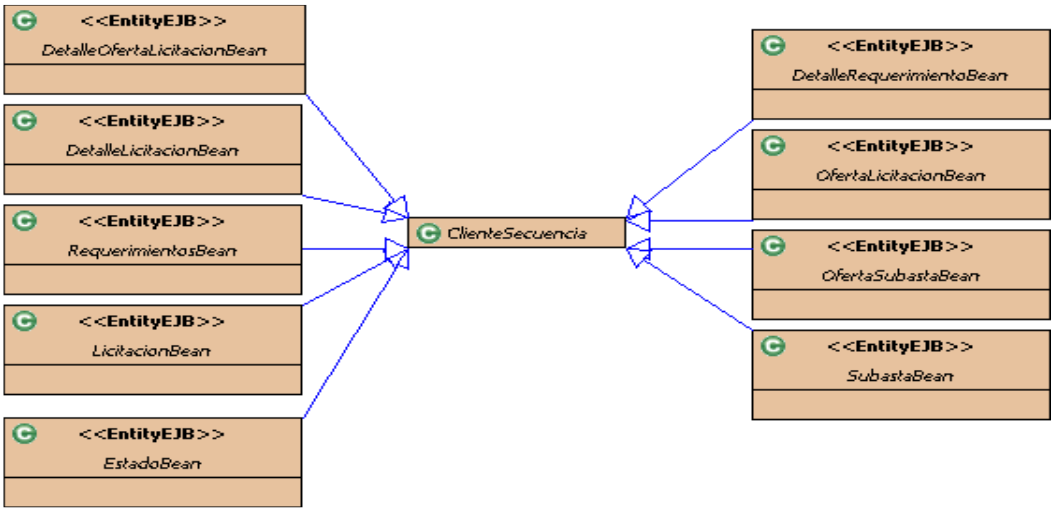

Figura 3-47 Objetos del Modelo: Entity Beans.

Fuente: E-guana. Autor: E-guana.

El modelo mostrado en la figura 3-47, explica cómo funcionan las Entities, siendo clases que están mapeadas con cada tabla de la base de datos, es decir, cada clase representa una tabla.

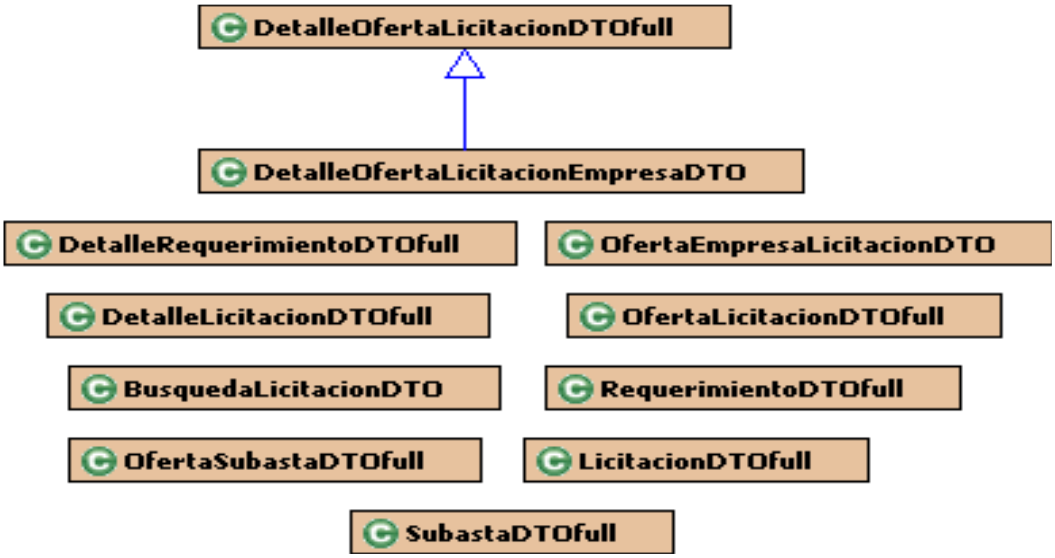

Figura 3-48 Objetos del Modelo: DTOs.

Fuente: E-guana. Autor: E-guana.

El modelo mostrado en la figura 3-48, explica la forma en que estàn organizadas las clases DTO(Data Transfer Object), que son utilizadas para traer los datos de las consultas, como criterios de bùsquedas en consultas y mostrarlos en las páginas.

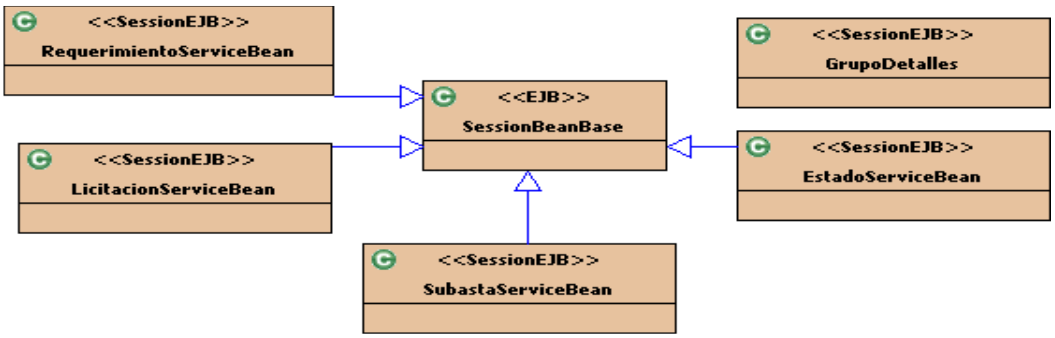

Figura 3-49 Objetos del Modelo: Session Beans.

Fuente: E-guana. Autor: E-guana.

En la figura 3-48 se muestran cómo las clases del sistema heredan algunas caracterìsticas de la clase Session Beans, que son clases que manejan la lógica de negocios y sus operaciones.

3.5.2 Objetos del controlador.

Los Objetos del controlador son específicamente los implementados por el marco de trabajo de JSF, los cuales no forman parte del desarrollo del proyecto, más son traídos a colación para tener en cuenta que E-guana está basado en el esquema Modelo Vista Controlador.

3.5.3 Objetos de la Vista.

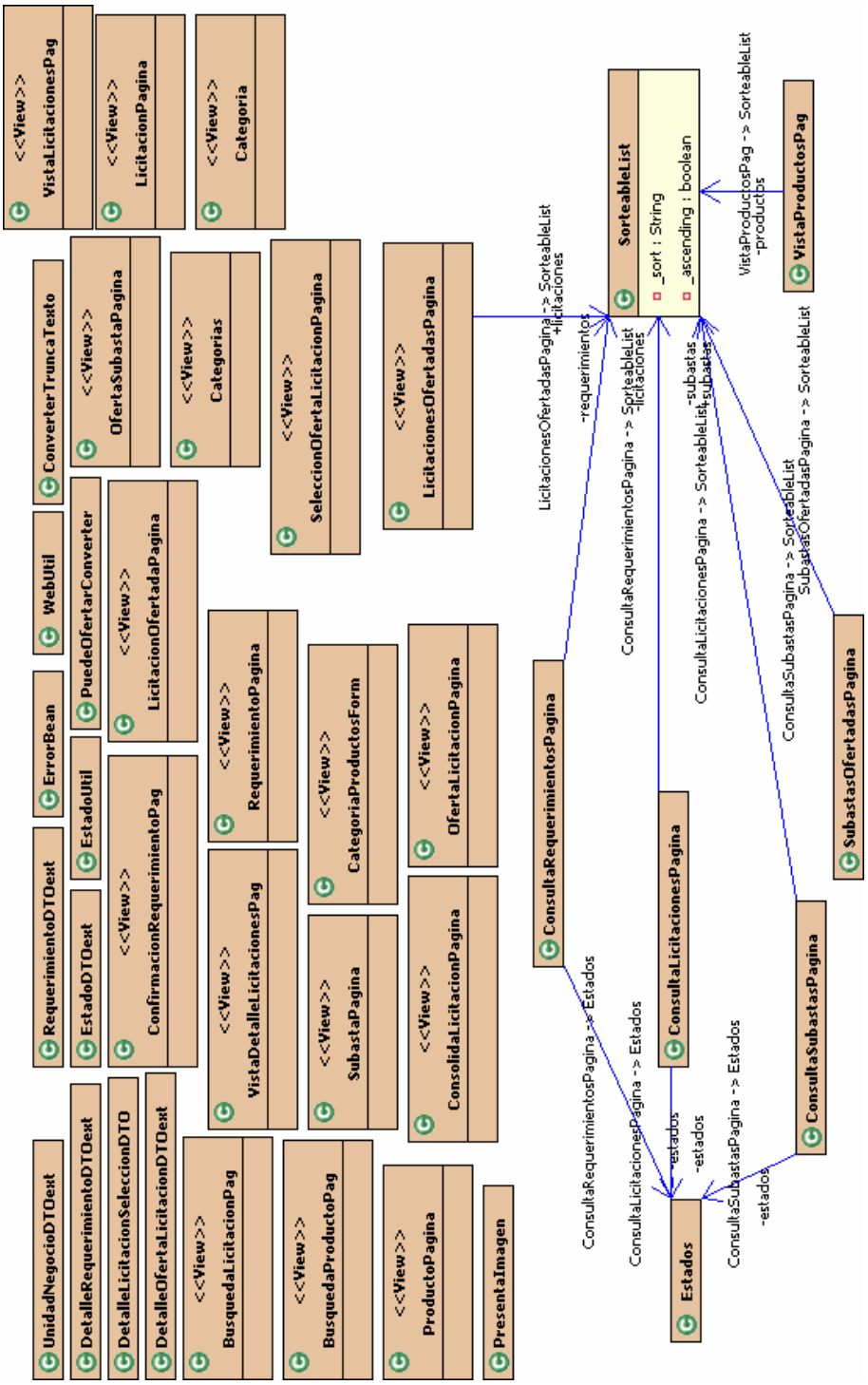

Figura 3-50 Objetos de la Vista.

Fuente: E-guana. Autor: E-guana.

En la figura 3-50 se explica el funcionamiento para poder viualizar los datos. Las clases con estas caracterìsticas son: Consulta de licitaciones, requerimientos y subastas, siendo estas el soporte principal del sistema.

3.5.4 Empaquetado e Implementación (Jar, WAR, EAR).

JARS: "Java Archives" es un formato desarrollado por Sun que permite agrupar las clases diseñadas en el lenguaje Java, este formato es ampliamente utilizado en ambientes Java de todo tipo, esto se debe a que otorga un nivel de compresión y reduce la carga administrativa al distribuir clases en el lenguaje.

EJB-JAR: Un EJB-JAR es la manera en que son distribuidos desarrollos EJB's. La estructura de un EJB-JAR es la siguiente:

/ *.class: Bajo este directorio base se encuentran las diversas clases que conforman a un EJB.

/META-INF/ejb-jar.xml: Este archivo contiene el denominado Deployment Descriptor utilizado en EJB's.

/META-INF/*: Este directorio además del Deployment Descriptor puede contener otros archivos de configuración utilizados por el "Application Server" ("EJB Container" para ser más exacto).

La creación de JARS y otros archivos de empaquetado y despliegue puede realizarse en IDE's o en algún otro programa como Ant, como es el caso de nuestro proyecto.

Se podrá ver el empaquetado EJB-JAR de E-guana en el anexo 1.

En ambientes Java también existen otros tipos de archivos llamados: WARS ("Web-Archives"), EARS ("Enterprise Archives"), sin embargo, es conveniente saber que estos siguen siendo archivos JAR ("Java Archives"), su creación se debe únicamente como convención y su contenido puede ser analizado y examinado de la misma manera que un archivo JAR.

WARS: "Web Archive" es una especificación desarrollada por Sun que permite agrupar un conjunto de clases y documentos que conforman una aplicación Web en Java en un sentido un poco peculiar a la que se acostumbra en un sitio.

La estructura de un (Web-Archive) WAR es la siguiente:

/ *.html *.jsp *.css: Este directorio base contiene los elementos que comúnmente son utilizados en un sitio, Documentos en HTML , JSP's , CSS("Cascading Style Sheets") ,etc.

/WEB-INF/web.xml: Contiene elementos de seguridad de la aplicación así como detalles sobre los Servlets que serán utilizados dentro de la misma.

/WEB-INF/classes/: Contiene las clases Java adicionales a las del JDK que serán empleadas en los JSP's y Servlets

/WEB-INF/lib/: Contiene los JAR's que serán utilizados por su aplicación.

El empaquetado WAR de E-guana se lo puede revisar en Anexo 2.

EARS: "Enterprise Archives" es simplemente un WAR y EJB-JAR agrupados, la razón de su existencia se debe básicamente a la estructura un "Java Application Server" , por lo general ningún vendedor de "Application Servers" hace una clara distinción entre el "Servlet Engine" (ambiente utilizado para WAR's) y el "EJB Container" (utilizado para EJB-JAR's) ; debido a esto se genero el termino EAR, que desde luego también sigue estando en el formato JAR.

En E-guana, no se genera el archivo EAR, debido a que los EJB-JAR están separados y se manejan como dos proyectos. El procedimiento al hacer el despliegue del WAR y ejecutar el proyecto es más rápido ya que en el Servidor de Aplicaciones ya debe estar el EJB-JAR.

CAPÍTULO 4

4 PRUEBAS E INTEGRACIÓN DE LICITACIÓN & SUBASTAS DE E-GUANA.

Los módulos de E-Guana se integran unos con otros en distintos niveles del sistema, desde la base de datos hasta la interfaz gráfica Web.

Todos los módulos hacen uso de la misma base de datos, cuyo acceso generalmente no se lo hace directamente a través de JDBC sino con la capa de persistencia proporcionada por los EJBs.

Cada una de las aplicaciones Web de los módulos permiten que la autenticación de usuario sea necesaria una sola vez que se accede a alguna de las aplicaciones, para que esta integración funcione se requieren configuraciones tanto en el servidor de aplicaciones Web así como en cada una de las aplicaciones Web, el mecanismo descrito anteriormente se conoce como “single sign-on” ó autenticación única.

Configuración Single Sign-On del servidor

Para lograr la autenticación única se debe modificar el archivo “server.xml” ubicado en el subdirectorio “jbossweb-tomcat50.sar” del directorio “deploy” de JBoss. Dicho archivo permite la configuración del servicio provisto por el contenedor Web.

Es necesario que el código siguiente esté presente dentro de las etiquetas que conforman la etiqueta “Host”, puesto que la distribución normal de JBoss no la tienen habilitada:

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

Configuración Dominio de Seguridad del servidor

Toda aplicación EJB o Web que requiera autenticación de usuario debe tener configurado un dominio de seguridad en el servidor, JBoss provee del archivo de configuración “login-config.xml”, donde se agregan y configuran los

dominios de seguridad a través de las políticas de aplicación (application-policy).

El código de la política de aplicación para E-Guana corresponde al siguiente código:

```
<application-policy name = "E-guana">
<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>
```

El atributo “name” de la etiqueta “application-policy” corresponde al nombre del dominio de seguridad que se está configurando. Este nombre debe ser referenciado por todas las aplicaciones que requieran hacer uso del mismo a través del nombre JNDI “java:/jaas/E-guana”.

Como se puede notar dentro de la etiqueta “authentication” se encuentra “login-module” esta última tiene el atributo “flag” con valor “required” para

indicar que el módulo de “login” es requerido para poder completar el proceso de autenticación.

Entre las opciones del módulo de “login” se destacan las consultas SQL para recuperar la clave y los roles que serán asignados al usuario.

Configuración Aplicación Web

Cada aplicación Web debe asegurarse de recuperar los mismos datos para la autenticación del usuario, en este caso todos deberán hacerlo del mismo dominio de seguridad, esto se lo hace agregando el código siguiente en el archivo “jboss-web.xml” de cada aplicación:

<security-domain>java:/jaas/E-guana</security-domain>

Como se nota el valor contenido en las etiquetas “security-domain” corresponde al nombre del dominio de seguridad previamente configurado precedido por “java:/jaas”.

Además dentro del descriptor de despliegue “web.xml” se indicarán el o los roles que pueden hacer uso de sus páginas, de la siguiente manera:

```
<security-constraint>
<web-resource-collection>
        <web-resource-name>Paginas seguras</web-resource-name>
        <url-pattern>/paginas/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
        <role-name>Administrador</role-name>
        <role-name>Comprador</role-name>
        <role-name>Vendedor</role-name>
        <role-name>AdminEmpresa</role-name>
```

```
  </auth-constraint>
  <user-data-constraint>
            <transport-guarantee>INTEGRAL</transport-guarantee>
  </user-data-constraint>
 </security-constraint>

 <login-config>
  <auth-method>FORM</auth-method>
  <form-login-config>
            <form-login-page>/login/login.jsf</form-login-page>
            <form-error-page>/login/loginError.jsf</form-error-page>
  </form-login-config>
 </login-config>

<security-role>
<role-name>Administrador</role-name>
</security-role>
<security-role>
  <role-name>Comprador</role-name>
</security-role>
<security-role>
  <role-name>Vendedor</role-name>
</security-role>
<security-role>
  <role-name>AdminEmpresa</role-name>
</security-role>
```

En este caso los roles “Administrador”, “Comprador, “Vendedor” y “AdminEmpresa” podrán hacer uso de los recursos ubicados dentro de del directorio “paginas”.

Luego de completados las configuraciones anteriores, el usuario sólo deberá ingresar sus datos: nombre de usuario y clave en la página de ingreso del sistema, se validan los datos ingresados y de ser correctos, entra a la aplicación de manera segura.

Lo anterior junto con una estructura de páginas Web similar en cada una de las aplicaciones permite que el usuario perciba a todas las aplicaciones como un todo que es E-Guana, a continuación se presenta una imagen que representa la estructura utilizada por las aplicaciones Web.

Figura 4-1 Estructura de las páginas del módulo de Licitaciones & Subastas

Fuente: E-guana. Autor: E-guana.

La parte superior o encabezado presenta los datos de la empresa y unidad de negocio a la que pertenece el usuario actualmente conectado así como un mensaje de bienvenida y finalmente un enlace para salir del sistema. Esta porción de la página hace uso de las etiquetas personalizadas JSP “Empresa” y “Usuario” lo cual se detallará más adelante.

La parte izquierda es utilizada para presentar un menú de opciones o cualquier otro elemento necesario.

La parte central contiene los elementos necesarios para la funcionalidad de la página.

Finalmente la parte inferior o pie de página presenta un conjunto de enlaces útiles que permiten desplazarse entre las distintas aplicaciones o módulos de E-Guana.

Aunque esta estructura es común a todas las aplicaciones del sistema, su implementación quedó a libertad de los desarrolladores, el módulo StoreFront utilizó frames HTML (cada una de las partes es un frame) mientras que tanto el módulo de Administración como el de Licitación & Subastas utilizaron fragmentos de páginas JSP, que luego son utilizadas por la página que maneja el contenido central o principal de la página a través de la directiva JSP “include”.

Respecto a la integración EJB es necesario mencionar que debido a las restricciones de la especificación J2EE todos los *beans* de entidad y sesión de todas las aplicaciones, se empaquetaron en un solo módulo EJB.

4.1 Plan de pruebas y Resultados.

Las pruebas realizadas al módulo de Licitación & Subastas consiste en un conjunto de pruebas de unidad, es decir pruebas directas al código para verificar que el mismo se comporte como se lo espera.

Debido a la extensión del módulo, se limitaron las pruebas a los *beans* de sesión que proveen los principales servicios que representan al módulo, de tal forma se puede asegurar que la mayor parte del módulo está funcionando de forma correcta.

Los *beans* sometidos a prueba junto con su clase de prueba son:

RequerimientoServiceBean	**RequerimientoPrueba**
LicitacionServiceBean	**LicitacionPrueba**
SubastaServiceBean	**SubastaPrueba**

4.1.1 Consideraciones

Para reducir la complejidad de las clases de prueba se utilizaron las clases denominadas como fachada o *proxy* para acceder a los *beans.*

Puesto que la funcionalidad del módulo requiere que su uso lo realice un usuario autenticado, en el método *setUp()*, que es llamado previo a

cada prueba, se configuró la autenticación de un usuario cuyos roles son: Administrador de Empresa, Comprador y Vendedor.

Las pruebas correspondientes a cada *bean* se ejecutan en cierto orden para obtener un resultado válido, puesto que son dependientes unos de otros siguiendo el ciclo de vida de las entidades involucradas.

Se generó una prueba adicional que comprende todas las pruebas en conjunto, la misma es manejada por la clase *AllTests*..

Un par de pruebas requieren el ID de un producto del sistema, para esto se buscó el producto cuya descripción corresponde a “HP Proliant”.

Cada clase de prueba tiene una instancia de la clase de fachada asociada al *bean* de sesión que está siendo sometido a prueba.

4.1.2 Pruebas a RequerimientoServiceBean

La primera prueba consistió en la creación de un requerimiento, el método del *bean* para esto es *crearRequerimiento*, el mismo recibe como argumento los datos completos del requerimiento los que en este caso corresponden a la observación con un valor de "Requerimiento de Prueba", la cantidad requerida igual a un entero con el valor de 10 y el identificador del producto que se requiere.

El método bajo prueba retorna un valor entero igual al identificador del requerimiento creado, por lo que parte de la prueba es que el método retorne un objeto (un entero) distinto de nulo.

La siguiente prueba consiste en buscar el requerimiento creado, se utiliza el método *testBuscarRequerimientosPorObservacion*, enviando como argumento el valor de la observación en la prueba previa, ya que este método retorna una lista de requerimientos que concuerden con la observación, pero se sabe que en este caso solo se retornará una lista con un elemento, se toma el primer elemento de dicha lista y se verifica que la observación retornada sea igual a la que se utilizó en la creación.

Una vez verificado que se puede encontrar el requerimiento se prueba la publicación del mismo a través del método *publicarRequerimiento*, debido a que el método requiere el identificador del requerimiento se debe primero buscarlo, por lo que se lo busca de acuerdo la observación como en la prueba anterior, lo anterior también se utiliza para la última prueba.

La eliminación de requerimiento es la última prueba de este servicio, el método del *bean* es *buscarRequerimiento.*

El código completo de la prueba de RequerimientoServiceBean se lo puede encontrar en el Anexo 3.

4.1.3 Pruebas a LicitacionServiceBean

La creación de la licitación es la primera prueba, el método respectivo es *crearLicitacion,* que requiere los datos completos de la licitación, la observación tiene un valor de “Licitacion de Prueba”, la fecha de caducidad tiene una correspondiente a "2006-12-31” mientras que la fecha de inicio es "2006-01-01". Cada licitación tiene un detalle que a su vez se refiere a un producto por lo que se hace uso del identificador del mismo, la cantidad licitada corresponde a un valor de 10.

Una vez invocado al método se verifica que el valor retornado por éste sea distinto de nulo, ya que corresponde al ID de la licitación recién creada.

Buscar la licitación es la siguiente prueba, a través del método *testBuscarPorObservacion* se puede realizar esto, el mismo retorna una lista que en esta situación sólo tiene un elemento que es la nueva licitación, se verifica que los valores de observación, fecha de caducidad y fecha de inicio sean los correctos.

La siguiente prueba consiste en la publicación de la licitación, el método de creación crea una licitación en estado de “Registrada”. Para publicar una licitación se invoca *publicarLicitacion* el mismo que recibe como argumento un valor entero correspondiente al identificador de la licitación, por lo que previamente es necesario buscar la licitación a considerando el valor del atributo observación.

La cuarta prueba es la declaración de la licitación como desierta a través del método *declararDesierta* que similar al método anterior recibe como argumento el ID de la licitación de interés.

Finalmente se prueba la eliminación de la licitación llamando a *eliminarLicitacion* del servicio que como en los dos métodos anteriores requiere el ID de una licitación.

Para ver el código completo de la clase de prueba ver anexo 4.

4.1.4 Pruebas a SubastaServiceBean

La primera prueba a la que es sometido el *bean* de sesión SubastaServiceBean es aquella que crea una subasta a través del método *crearSubasta*, los datos que recibe este método son la observación de la subasta con un valor igual a “Subasta de Prueba”, la

cantidad subastada cuyo valor es 10, la fecha tope de la subasta con una fecha correspondiente a “2006-12-31”, el precio base de la subasta con un valor de 1000, además puesto que la subasta se refiere a un producto del catálogo de la empresa del usuario autenticado, se hace referencia al producto cuya descripción corresponde a "Televisor Panasonic ViewMe Encore". Se verifica que el valor retornado por el método sea distinto de nulo.

La siguiente prueba es la búsqueda de la subasta creada, para esto se lo hace considerando el atributo observación de la subasta utilizando el método *buscarPorObservacion*, se verifican que tanto la observación, fecha tope, cantidad y precio base de la subasta sean los correctos, además del identificador y descripción del producto subastado.

Luego se realiza la prueba de recepción de oferta a la subasta a través del método *receptarOfertaSubasta.* Para este simulacro utilizaremos como datos de la oferta: el Codigo de la subasta a la que estamos ofertando, una observación con el valor de "Oferta Subasta de Prueba" y el precio de la oferta con el valor de 2200. Tomaremos como consideración que el Código de la subasta debe ser un código que exista en la base de datos.

Finalmente se busca la oferta invocando el método *buscarOfertasSubasta* que recibe como argumento el identificador de la subasta de la que se desean las ofertas, se comprueba que la observación de la oferta y su precio tengan el valor establecido en la prueba de recepción de oferta.

El código de la clase de prueba se lo puede ver en el anexo 5.

4.1.5 Resultados

La ejecución de las pruebas descritas se realiza utilizando un *script* de Ant:

```
<project name="pruebas" default="pruebas">

  <property file="ant.properties" />
  <property name="resultados.dir" value="./resultados"></property>

  <path id="pruebas.path">
        <pathelement location="${client.jar}"/>
        <pathelement location="../bin"/>
  </path>

  <target name="prepara">
        <mkdir dir="${resultados.dir}" />
  </target>

  <target name="pruebas" depends="prepara">
        <junit printsummary="true">
              <formatter type="xml" usefile="true" />
              <test
              name="org.E-guana.licitacion.pruebas.RequerimientoPrueba" />
              <test
              name="org.E-guana.licitacion.pruebas.LicitacionPrueba" />
              <test name="org.E-guana.licitacion.pruebas.SubastaPrueba" />
              <test name="org.E-guana.licitacion.pruebas.AllTests" />
              <classpath refid="pruebas.path" />
        </junit>

        <junitreport todir="${resultados.dir}">
              <fileset dir=".">
```

```
                    <include name="TEST-*.xml" />
                </fileset>

                <report format="frames" todir="${resultados.dir}"/>
            </junitreport>

    </target>

</project>
```

El código anterior indica que se ejecutarán las pruebas de RequerimientoPrueba, LicitacionPrueba, SubastaPrueba y por último AllTests.

El resultado de la ejecución de las mismas se generará en formato XML que luego será procesado para la generación de un reporte HTML organizado en una estructura de *frames*.

El resumen de los resultados es:

Nombre de la Prueba	Pruebas	Errores/Fallas	Tiempo*
RequerimientoPrueba	**4**	**0**	**28.391**
LicitacionPrueba	**5**	**0**	**2.594**
SubastaPrueba	**4**	**0**	**1.859**
AllTests	**13**	**0**	**6.657**

Tabla 4-1 Resumen de resultados de las pruebas

Fuente: E-guana. Autor: E-guana.

***unidad de tiempo en segundos**

El resultado más significativo de las pruebas es que efectivamente el servicio prestado por el módulo de Licitación & Subastas está comportándose como debe ya que no se presentaron errores ni fallas.

El tiempo para la primera prueba es mucho más elevado que el resto, debido a que en el momento que es ejecutada, la máquina virtual debe cargar y preparar los recursos necesarios para el proceso de pruebas, por lo que para la prueba siguiente todos los recursos ya están disponibles y el tiempo corresponde al de la prueba en sí. Puesto que las pruebas son del tipo prueba de unidad este valor no es de mayor importancia en este caso.

A continuación se presentan las pantallas de los reportes con los resultados de las prueba tomando la unidad de tiempo que se muestra en los cuadros como segundos:

Summary

Tests	Failures	Errors	Success rate	Time
26	0	0	100.00%	39.501

Note: *failures* are anticipated and checked for with assertions while *errors* are unanticipated.

Packages

Name	Tests	Errors	Failures	Time(s)
org.equana.licitacion.pruebas	26	0	0	39.501

Figura 4-2 Resumen del resultado de las pruebas

Fuente: E-guana. Autor: E-guana.

Class org.eguana.licitacion.pruebas.AllTests

Name	Tests	Errors	Failures	Time(s)
AllTests	13	0	0	6.657

Tests

Name	Status	Type	Time(s)
testCrearRequerimiento	Success		0.407
testBuscarRequerimientosPorObservacion	Success		0.609
testPublicarRequerimiento	Success		0.250
testEliminarRequerimiento	Success		0.078
testCrearLicitacion	Success		0.156
testBuscarPorObservacion	Success		0.188
testPublicarLicitacion	Success		0.094
testDeclararDesierta	Success		0.109
testEliminarLicitacion	Success		0.219
testCrearSubasta	Success		0.172
testBuscarPorObservacion	Success		0.172
testReceptarOfertaSubasta	Success		0.203
testBuscarOfertasSubasta	Success		4.000

Figura 4-3 Resultados de la prueba "AllTests"

Fuente: E-guana. Autor: E-guana.

Class org.eguana.licitacion.pruebas.RequerimientoPrueba

Name	Tests	Errors	Failures	Time(s)
RequerimientoPrueba	4	0	0	28.391

Tests

Name	Status	Type	Time(s)
testCrearRequerimiento	Success		27.359
testBuscarRequerimientosPorObservacion	Success		0.468
testPublicarRequerimiento	Success		0.266
testEliminarRequerimiento	Success		0.125

Figura 4-4 Resultados de la prueba "RequerimientoPrueba"

Fuente: E-guana. Autor: E-guana.

Class org.eguana.licitacion.pruebas.LicitacionPrueba

Name	Tests	Errors	Failures	Time(s)
LicitacionPrueba	5	0	0	2.594

Tests

Name	Status	Type	Time(s)
testCrearLicitacion	Success		1.297
testBuscarPorObservacion	Success		0.765
testPublicarLicitacion	Success		0.266
testDeclararDesierta	Success		0.141
testEliminarLicitacion	Success		0.125

Figura 4-5 Resultados de la prueba "LicitacionPrueba"

Fuente: E-guana. Autor: E-guana.

Class org.eguana.licitacion.pruebas.SubastaPrueba

Name	Tests	Errors	Failures	Time(s)
SubastaPrueba	4	0	0	1.859

Tests

Name	Status	Type	Time(s)
testCrearSubasta	Success		1.219
testBuscarPorObservacion	Success		0.265
testReceptarOfertaSubasta	Success		0.219
testBuscarOfertasSubasta	Success		0.156

Figura 4-6 Resultados de la prueba "SubastaPrueba"

Fuente: E-guana. Autor: E-guana.

4.2 Licitación & Subastas con Store Front

El módulo de Store Front es el que utilizan los usuarios para realizar compras de productos directamente, consultar el estado de una compra así como la anulación de la misma.

La integración con el módulo de Store Front se da en la capa de persistencia, más específicamente en los *beans* de sesión SubastaServiceBean y LicitacionServiceBean cuando el proceso respectivo termina en un compra.

4.2.1 SubastaServiceBean

Cuando el usuario acepta una oferta a la subasta se debe generar una orden de compra o compra simplemente, es en este caso que SubastaServiceBean hace uso del *bean* de entidad CompraBean y del *bean* relacionado DetalleCompraBean.

La compra tiene el atributo “tipo” que se establece con el valor “s” para indicar que la compra resulta de una subasta. Éste es de interés para el módulo de Store Front.

4.2.2 LicitacionServiceBean

Similar al caso de subastas, cuando el usuario acepta una oferta a una licitación, más específicamente se acepta un detalle de una oferta. A diferencia de las subastas, que solo hay un producto en cuestión, una licitación puede involucrar varios productos y por lo tanto varias ofertas, en consecuencia el resultado serán varias compras. El atributo “tipo” se establece como “l” para indicar que la compra fue resultado de una licitación.

4.3 Licitación & Subastas con Administración

El módulo de Administración permite entre otras cosas el mantenimiento del catálogo de productos, además provee de servicios necesarios por el

resto de componentes del sistema como acceso a los datos del usuario y de la empresa a la que pertenece.

El módulo de Licitación & Subastas se integra con el módulo de Administración en varios niveles, desde la capa de persistencia manejada por los EJBs hasta la capa de presentación a través del uso de etiquetas personalizadas JSP.

4.3.1 Capa EJB

Inicialmente se detallará el uso de los *beans* de entidad del módulo de Administración por parte de los *beans* de entidad que constituyen el módulo de Licitación & Subastas, más específicamente se describirán la relación existente entre los *beans,* los métodos para lograr dicha relación y el código XDoclet necesario para la generación de adecuados descriptores de despliegue.

4.3.1.1 DetalleLicitacionBean

Este *bean* de entidad tiene una relación directa con su par para el manejo de productos, ProductoBean, ya que el detalle de una licitación se refiere a uno de los productos que se desean adquirir por

medio de la licitación. Esta relación es múltiple del lado del producto por lo indicado anteriormente.

Los métodos para manejar esta relación son:

```
public abstract ProductoLocal getProducto()
public abstract void setProducto(ProductoLocal producto)

La configuración XDoclet es la siguiente:
        @ejb.interface-method
        @ejb.relation name = "Producto-DetalleLicitacion"
        target-ejb = "Producto"
        role-name = "DetalleLicitacion-belongs-to-Producto"
        target-multiple = "yes"
        target-role-name = "Producto-has-many-DetalleLicitacion"
        @jboss.relation related-pk-field = "id"
        fk-column = "id_producto"
```

Del código anterior se debe destacar que el *bean* ProductoBean no tendrá un método de acceso para los detalles de licitación, además esta relación se manejará en la base de datos a través de una clave foránea ubicada en la columna *id_producto* de la tabla asociada.

4.3.1.2 DetalleOfertaLicitacionBean

El detalle de la oferta a una licitación es en otras palabras el detalle de un respuesta hecha por una empresa a la licitación publicada, por este motivo DetalleOfertaLicitacionBean necesita conocer los datos del producto ofertado, que en este contexto es de hecho una entrada

del catálogo de la empresa ofertante, es decir necesita una referencia al *bean* CatalogoBean, los métodos respectivos son:

```
public abstract CatalogoLocal getCatalogo()
public abstract void setCatalogo(CatalogoLocal catalogo)

El código XDoclet es:
  @ejb.interface-method
  @ejb.relation
        name = "DetalleOfertaLicitacion-Catalogo"
        role-name = "detalleOferta-catalogo"
        target-ejb = "Catalogo"
        target-role-name = "catalogo-detalleOferta"
        target-multiple = "yes"
   @jboss.relation
        related-pk-field = "id"
        fk-column = "ID_CATALOGO"
```

La explicación del código es similar a la de DetalleLicitacionBean.

4.3.1.3 RequerimientosBean

Como se ha indicado en secciones anteriores, un requerimiento es publicado por alguna de las unidades de negocio a través de un usuario perteneciente a la misma, por lo tanto cada requerimiento debe relacionarse con una unidad de negocio.

Los métodos para el manejo de esta relación son:

```
public abstract UnidadNegocioLocal getUnidadNegocio()
public abstract void setUnidadNegocio(UnidadNegocioLocal
unidadNegocioLocal)
```

Mientras que el código XDoclet es:

```
@ejb.relation
    name="Requerimiento-Unidad"
```

```
        role-name = "Requerimiento-pertenece-a-UnidadNegocio"
        target-ejb = "UnidadNegocio"
        target-role-name = "UnidadNegocio-tiene-Requerimiento"
        target-multiple = "yes"
    @jboss.relation
        fk-column = "id_unidad_negocio"
        related-pk-field = "id"
```

Cuyo significado es similar a la de los *beans* anteriores.

4.3.1.4 OfertaLicitacionBean

Cada oferta o respuesta a una licitación es hecha por un usuario de alguna de las empresas registradas en E-Guana, que en estas circunstancias el usuario es uno con el rol de vendedor dentro de dicha empresa.

Esta relación se maneja con los métodos:

```
public abstract UsuarioLocal getVendedor()
public abstract void setVendedor(UsuarioLocal vendedor)
```

El código XDoclet es el siguiente:

```
@ejb.relation
        name = "OfertaLicitacion-Usuario"
        role-name = "oferta-es-hecha-por-vendedor"
        target-ejb = "Usuario"
        target-role-name = "vendedor-ofrece-por-licitacion"
        target-multiple = "yes"
@jboss.relation
        fk-column = "USR_VENDEDOR"
        related-pk-field = "usuario"
```

4.3.1.5 DetalleRequerimientoBean

Este *bean* al igual que DetalleLicitacionBean hace uso del *bean* ProductoBean, puesto que el detalle de un requerimiento corresponde al producto que es solicitado por la unidad de negocio.

El código es similar al del *bean* DetalleLicitacionBean.

4.3.1.6 SubastaBean

Una subasta corresponde a la venta de un producto del catálogo de una empresa por alguno de sus usuarios (en este caso un vendedor), por lo que SubastaBean se relaciona con CatalogoBean y UsuarioBean.

Para la primera relación se tienen los métodos:

```
public abstract CatalogoLocal getCatalogo()
public abstract void setCatalogo(CatalogoLocal catalogo)
```

Y el XDoclet:

```
@ejb.relation
        name = "Catalogo-Subasta"
        target-ejb = "Catalogo"
        role-name = "Subasta-referencia-a-Catalogo"
        target-multiple = "yes"
        target-role-name = "Catalogo-es-referenciado-por-Subasta"
```

```
@jboss.relation
        related-pk-field = "id"
        fk-column = "ID_CATALOGO"
```

Mientras que la relación con UsuarioBean se maneja con los métodos:

```
public abstract UsuarioLocal getVendedor()
public abstract UsuarioLocal setVendedor(UsuarioLocal vendedor)
```

El código XDoclet es:

```
@ejb.relation
        name = "Subasta-Vendedor"
        role-name = "Subasta-es-hecha-por-Vendedor"
        target-ejb = "Usuario"
        target-role-name = "Vendedor-arma-Subasta"
        target-multiple = "yes"
@jboss.relation
        related-pk-field = "usuario"
        fk-column = "USR_VENDEDOR"
```

4.3.1.7 LicitacionBean

Una licitación es publicada por un usuario, el comprador, por lo que se debe manejar una relación con UsaurioBean, esto se hace a través de los métodos:

```
public abstract UsuarioLocal getUsuarioComprador()
public abstract void setUsuarioComprador(UsuarioLocal usuario)
```

El XDoclet es el siguiente:

```
@ejb.relation
    name="LicitacionUsuario"
    role-name="usuario-pertenece-licitacion"
    target-ejb="Usuario"
    target-role-name="licitacion-tiene-estado"
    target-multiple="yes"
```

```
@jboss.relation
        related-pk-field="usuario"
        fk-column="USR_COMPRADOR"
```

Adicional a lo anterior, se implementaron métodos *select* para facilitar ciertas búsquedas, estos métodos hacen uso de los *beans* de la empresa (EmpresaBean) y del catálogo (CatalogoBean), y son los siguientes:

```
public abstract Collection ejbSelectCatalogos (
LicitacionLocal licitacion,
EmpresaLocal empresaVendedora) throws FinderException;
```

Cuya consulta es:

```
SELECT OBJECT(c) FROM
Catalogo AS c,Licitacion AS l,
IN(l.detallesLicitacion) AS dl
WHERE l = ?1 AND c.empresa = ?2 AND c.producto.id = dl.producto.id
```

Este método retorna una colección de catálogos que pertenezcan a la empresa dada. Para lo que compara los productos de la licitación con los de su catálogo.

```
public abstract int ejbSelectTieneAlgunCatalogo(
LicitacionLocal licitacion,
EmpresaLocal empresaVendedora) throws FinderException;
```

Con la consulta

```
SELECT COUNT(c) FROM
Catalogo AS c,
Licitacion AS l,
IN(l.detallesLicitacion) AS dl
WHERE l = ?1 AND c.empresa = ?2 AND c.producto.id = dl.producto.id
```

El objetivo del método es determinar la cantidad de entradas en el catálogo de la empresa que cumplen con los productos licitados.

4.3.1.8 OfertaSubastaBean

La oferta a una subasta es hecha por un usuario comprador, el manejo de esta relación se hace con los métodos:

```
public abstract UsuarioLocal getComprador()
public abstract void setComprador(UsuarioLocal comprador)
```

Y el código XDoclet:

```
@ejb.relation
        name = "OfertaSubasta-Usuario"
        role-name = "oferta-es-hecha-por-comprador"
        target-ejb = "Usuario"
        target-role-name = "comprador-ofrece-por-subasta"
        target-multiple = "yes"
@jboss.relation
        fk-column = "USR_COMPRADOR"
        related-pk-field = "usuario"
```

En cuanto a los *beans* de sesión, su uso del módulo de Administración se da básicamente a través de la búsqueda de las instancias necesarias para formar las relaciones descritas entre los *beans* de entidad en las etapas de creación de nuevas entidades.

Adicional a esto todos los *beans* de sesión derivan de la clase SessionBeanBase del módulo de Administración, para tener acceso

a varios métodos útiles como aquellos que permiten determinar los roles del usuario previo a la ejecución de alguna operación.

4.3.2 Capa Web

Toda página de este módulo hace uso de la etiqueta JSP personalizada Empresa, esta etiqueta es proporcionada por el módulo de Administración para tener acceso a los datos de la empresa del usuario que actualmente está conectado. Esta etiqueta trabaja en conjunto con la etiqueta Usuario, que permite acceder a los datos del usuario. En ambos casos estos datos se almacenan en la sesión.

En la capa correspondiente a la aplicación Web de este módulo se utilizan algunos *beans* de sesión del módulo de Administración.
Su llamado lo realizan los controladores JSF de las páginas. A continuación se detalla su uso.

4.3.2.1 BusquedaProductoPag

Este controlador hace uso de los *beans* AdminCatalogoBean y AdminProducto para hacer búsquedas en el catálogo de las empresas y directamente a los productos registrados en el sistema respectivamente.

4.3.2.2 OfertaLicitacionPagina

Cuando el usuario desea ofertar un producto, para una licitación, que no está en su catálogo, utiliza el *bean* AdminCatalogoBean para crear dicho producto.

4.3.2.3 ConsolidaLicitacionPagina

Previo a la consolidación de una licitación (armar una licitación a partir de los requerimientos hechos por sus unidades de negocio) es necesario determinar las unidades que conforman la empresa, para esto utiliza el *bean* AdminBean que le provee de un método adecuado.

4.3.2.4 Categorías

Utiliza los *beans* AdminCatalogoBean y AdminProductoBean similar a BusquedaProductoPag, adicionalmente utiliza el *bean* NavegadorCatalogoBean para obtener una estructura de datos que permite obtener una representación jerárquica de las categorías de productos existentes en el sistema.

4.4 Licitación & Subastas con Reportería

El módulo de Reportes o Reportería generará un sinnúmero de reportes a través del uso de distintas herramientas desarrolladas con ese fin, como Big Faceless Report Generator o Jasper Reports, estos *frameworks* pueden acceder directamente a la base de datos por lo que no hay interacción con este módulo en esos casos, no obstante el módulo de Licitación & Subastas provee de servicios de consultas que son utilizadas dentro de las páginas de la aplicación Web y que por lo tanto también pueden ser utilizados por dichos *frameworks* en caso de ser necesario, con tan solo una configuración adecuada de su *classpath*.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

- El desarrollo de E-guana nos ha dejado varias experiencias interesantes y han cambiado nuestra visión en lo que respecta la construcción de aplicaciones ya sea de escritorio o soluciones Web. Si bien el costo de aprendizaje de las herramientas de programación de Software Libre, a inicios de la construcción de E-guana, fue muy alto, debido a la complejidad de las herramientas seleccionadas y la escasa documentación. La construcción de una robusta arquitectura y el uso de patrones de diseño, y la combinación de herramientas adicionales generadoras de código permitieron que el desarrollo sea una tarea muy sencilla. Esta ventaja se la ve también reflejada en lo que respecta al mantenimiento, implantación de la aplicación y la construcción de módulos adicionales con programadores nuevos que no tendrán que pasar mucho tiempo aprendiendo como funciona internamente la aplicación.

- Otro sabor que nos deja el uso de herramientas de Software Libre comparándolas con nuestra experiencia anterior en el uso de herramientas propietarias, es que cuando se presenta un error a nivel de la herramienta no tenemos acceso al código fuente de esta y por lo tanto

no es posible solucionar esos problemas, lo que ocurre en esos casos es que si se ha pagado por el soporte del uso de esas herramientas, podemos reportar esos errores. Luego en una nueva versión que compremos de esa herramienta esos errores estarán corregidos, pero vendrán unos nuevos que si queremos soporte para reportar estos problemas debemos pagar nuevamente por el servicio de soporte, para que los nuevos errores que se encuentren sean corregidos en la siguiente versión de la herramienta que se esta utilizando y que si queremos esa versión debemos volver a pagar, y así este problema no termina nunca. Cuando se nos presentaban errores de las herramientas de Software Libre que utilizamos en el desarrollo de E-guana, los que eran evidentes los corregimos al momento modificando las fuentes y volviendo a compilarlas. Los que eran más complejos se buscaba en los foros de ayuda que hay en Internet y allí se encontraba la solución para estos problemas, y cuando a nadie le ha ocurrido antes algún error, uno puede dejar el problema en el foro y los más probable es que alguien nos ayude con eso.

- En el desarrollo de la aplicación usando Software Libre nos ha enseñado lo importante de la filosofía del Software Libre que es el compartir. Creo personalmente (Roberto Guerrero) que el término Software Libre, más que un nuevo estilo de programación se convierte en una necesidad a la

que debemos llegar como sociedad. El uso de patentes y el software propietario son enemigos de la cultura y el desarrollo. Por citar un algunos ejemplos si algún laboratorio descubre la cura contra el cáncer según las leyes deben de pasar 70 años para que la gente común pueda tener acceso a esta cura, ya que el laboratorio que lo descubrió por ley tendrá la patente y por lo tanto será el único que lo pueda comercializar la medicina. El uso de herramientas propietarias en las Escuelas y Universidades limita a los estudiantes el desarrollo de aprendizaje, ya que cuando un alumno le pregunta al profesor como funciona alguna herramienta propietaria no podrá contestárselo ya que no se tiene acceso a las fuentes de esta, y si tiene fallos no podrá arreglarlas, ni aunque tenga acceso a las fuentes ya que las como están hecha las leyes actualmente es ilegal. Otro ejemplo es que si una persona tiene un programa que es propietario no lo puede compartir con un amigo que le pida ese programa ya que la licencia de ese programa dice que es ilegal, y aquí existe un dilema ético y moral. Debería o no darle una copia de ese programa a mi amigo? Ya que si no lo hago no estoy siendo un buen ciudadano y si lo hago estoy violando la licencia del programa. Empresas como Microsoft e Intel intentan hacer alianzas para vender computadoras que no se pueda instalar en ellas Software Libre. La popular canción de cumpleaños “Cumpleaños Feliz” tiene derechos de autor y cantarla sin pagarle al autor nos convierte en infractores de la ley. Concluyo que

mucha de las leyes actuales fueron creadas con intereses de beneficiar a unos cuantos y no para beneficiar a toda la sociedad, debería crearse nuevas leyes que beneficien a la sociedad en general y usar la imaginación para obtener ganancias de una manera creativa y no solo de las rentas que pueda dejar un programa o un nuevo descubrimiento en alguna de las ciencias.

- E-guana deja demostrado que se pueden construir aplicaciones Web serias con herramientas no propietarias, es decir, herramientas de Código Abierto y Software Libre y tener un éxito en el desarrollo e implantación de estos sistemas.

- Como conclusión final hacemos énfasis en que el *"éxito de aplicaciones empresariales serias no depende únicamente de la tecnología que se utilice para su desarrollo, sino la mayor parte de este éxito depende del diseño y arquitectura con que se han implementado estas aplicaciones"*

Recomendaciones

- E-guana es un-sistema escalable, es decir, que puede seguir-evolucionando, esto es posible debido a que ha sido implementado con una robusta y flexible arquitectura, que es lo que permite el éxito de las aplicaciones J2EE.

- E-guana si bien es un prototipo de lo que puede llegar a ser una compleja aplicación web. No deja de ser una potente herramienta que puede ser empleado en cualquier empresa que requiera resolver procesos de de adquisición o venta de productos o servicios.

- En E-guana se pueden crear módulos adicionales, que pueden ser integrados sin mucha dificultad, uno de ellos puede ser un módulo de-ERP o un módulo de Data Mining para hacer predicciones y correcciones en los procesos de Compras y Ventas.

- Creemos como equipo los que desarrollamos el módulo de Licitación y Subasta que el código fuente sea liberado y puesto a disposición de las personas o empresas que quieran implementarlo o seguir desarrollándolo. Hacer esto permitirá que E-guana se siga desarrollando, complementando, actualizando y mejorando y de esta manera la sociedad será la más beneficiada por ello.

- Desde los inicios de E-guana hasta la fecha la tecnología J2EE ha dado unos cambios muy interesantes que mejoran y simplifican las aplicaciones J2EE, actualmente esta en la versión J2EE 5, E-guana podría migrarse sin mucha dificultad a esta nueva versión de J2EE5 y de esta manera aprovechar estas nuevas características en los nuevos módulos que se desarrollen en E-guana.

Glosario

Autentificación: Es el proceso de verificar la identidad de un usuario, equipo u otra entidad en un sistema computacional, usualmente como un pre requisito para permitir acceso para recursos en un sistema.

Autorización: Es el proceso por el cual el acceso para un método o recurso es determinado. La autorización depende sobre la determinación si el principal asociado con un requerimiento a través de la autentificación esta en un rol de seguridad dado.

Browser: Navegador de Internet.

API: (Application Programmable Interface), es un conjunto de servicios, interfaces de programación de aplicaciones.

Applet: Un componente J2EE que se ejecuta típicamente en un navegador Web pero que se puede ejecutar en una variedad de otras aplicaciones y dispositivos que soporten el modelo de programación de applets.

Aplicación cliente: Componente J2EE del cliente que se ejecuta en su propia Máquina Virtual. Las Aplicaciones clientes tienen acceso para algunas APIs de la Plataforma J2EE.

Archivo EAR: Enterprise Archive file. Es un JAR que contiene una aplicación Empresarial J2EE.

Archivo WAR: Web Archive file. Es un JAR que contiene una aplicación Web J2EE.

Archivo JAR: Java Archive file. Es un archivo comprimido que contiene un conjunto de archivos y clases en Java.

Contenedor: Es una entidad que provee administración del ciclo de vida, seguridad, despliegue y servicios de tiempo de ejecución a los componentes J2EE. Cada tipo de contenedor (EJB, Web, Appet y Aplication Client) también provee servicios de componentes específicos.

Contenedor Applet: Un contenedor que incluye soporte para el modelo de programación de applet.

Contenedor Application client: Un contenedor que soporta los componentes Application Client.

Contenedor EJB: Es un contenedor que implementa los componentes EJB, en base a la arquitectura J2EE. La arquitectura especifica un ambiente de tiempo de ejecución para Enterprise Beans que incluye seguridad, concurrencia, ciclo de vida, administración, transacciones, despliegue, nombramiento y otros servicios. Un Contenedor EJB es provisto por un Servidor EJB o J2EE.

Context root: Es el nombre que da el mapeo al documento principal de una aplicación web (o Document Root).

CSS: Cascading style sheet. Es una hoja de estilos usada por los documentos HTML y XML para añadir un estilo a todos los elementos en un tag particular.

Deployment: Es el proceso mediante el cual el software es instalado en dentro de un ambiente operacional.

Deployment descriptor: Es un archivo XML proveído por cada módulo y aplicación J2EE que describe como ellos van a ser desplegados.

Document Object Model (DOM): Es un API para acceder y manipular Documentos XML como estructura tipo árbol. DOM provee interfaces de plataforma y lenguaje neutral que habilita a los programas y scripts acceso dinámico y modificación de contenido y estructura en documentos XML.

Document root: O Documento principal. Es el directorio de más alto nivel de una aplicación Web. Document root es donde son almacenados las páginas JSP, clases y archivos de lado del cliente, y recursos Web estáticos.

EJB Home object: Es un objeto que provee operaciones del ciclo de vida(crear, remover, encontrar) para un Enterprise Bean. La clase para un EJB Home object es generada por las herramientas de despliegue del contenedor. El EJB Home object implementa la interface Home del Enterprise Bean. El cliente hace una referencia al EJB Home object para realizar operaciones del ciclo de vida sobre el objeto EJB. El cliente utiliza JNDI para localizar un EJB Home object.

EJB Object: Es un objeto cuya clase implementa la Interface remota del Enterprise Bean. Un cliente nunca referencia una instancia de un Enterprise Bean directamente, el cliente siempre referencia un EJB Object. La clase de

un EJB Object es generada por las herramientas de despliegue del contenedor.

Enterprise Bean: Es un componente J2EE que implementa una tarea del negocio o una entidad del negocio y esta hospedada por un contenedor EJB, puede ser un Entity Bean, un Session Bean, o un Message-Driven Bean.

Enterprise JavaBeans (EJB): Es una arquitectura de componentes para el desarrollo y despliegue de aplicaciones orientada a objetos, distribuidas y de nivel empresarial. Las aplicaciones escritas usando la arquitectura Enterprise JavaBeans son escalables, transaccionales y seguras.

Entity Bean: Es un Enterprise Bean que representa la persistencia de datos mantenidos en una base de datos. Un Entity Bean puede manejar su propia persistencia o puede delegar esta función a su contenedor. Un Entity Bean es identificado por una clave primaria. Si el contenedor en el cual el Entity Bean es hospedado se cae, el Entity Bean, su clave primaria, y cualquier referencia remota sobrevive a la caída.

Finder Method: Es un método definido en la Interface Home y es invocado por un cliente para localizer un Entity Bean.

Home Interface: Una o dos interfaces para un Enterprise Bean. La interface Home define cero o muchos métodos para manejar un Enterprise Bean. La Interface Home de un Session Bean define los métodos create y remove, así como la Interface Home de un Entity Bean define los métodos create, Finder, y remove.

HTML: Hypertext Markup Language. Lenguaje para escribir documentos en Internet. HTML habilita el uso embebido de imágenes, sonido, flujo de video, formularios, referencias de URLs y formateo básico de texto.

HTTP: Hypertext Transfer Protocol. Es el Protocolo de Internet usado para obtener objetos hipertexto desde hosts remotos. Los mensajes HTTP consisten en requerimientos desde clientes hacia servidores y respuesta desde servidores a clientes.

HTTPS: HTTP sobre el protocolo SSL.

JAR: Java archive. Es un formato de archivo independiente de la plataforma que permite que muchos archivos sean agregados en un solo archivo.

Java 2 Platform, Enterprise Edition (J2EE): Es un ambiente para desarrollo y despliegue de aplicaciones empresariales. La plataforma J2EE consiste de

un conjunto de servicios, APIs y protocolos que proveen la funcionalidad para desarrollo multi hilo y aplicaciones basadas en Web.

Java Naming and Directory Interface (JNDI): Es una API que prove funcionalidad de nombramiento y directorios.

JavaBeans component: Es una clase de Java que puede ser manipulada por herramientas y compuesto dentro de aplicaciones. Un componente JavaBeans puede adherirse a ciertas propiedades y eventos de las convenciones de la Interface.

JavaMail: Es un API para envío y recepción de email.

JavaServer Pages (JSP): Es una tecnología Web extensible que usa datos estáticos, elementos JSP, y objetos Java de lado del servidor para generar contenido dinámico en clientes. Típicamente los datos estáticos son elementos HTML o XML, y en muchos casos el cliente es un Navegador o Browser.

JavaServer Pages Standard Tag Library (JSTL): A tag library that encapsulates core functionality common to many JSP applications. JSTL has support for common, structural tasks such as iteration and conditionals, tags

for manipulating XML documents, internationalization and locale-specific formatting tags, SQL tags, and functions.

JDBC: Es un API para conectividad, independiente de la base de datos, entre la plataforma J2EE y un amplio rango de Data Sources (Fuente de Datos).

JSP tag library: Es una colección de tags personalizados descritos mediante un descriptor tag library y clases Java.

Life Cycle (J2EE component): Ciclo de Vida. Se refiere a la existencia de los componentes J2EE del Framework. Cada tipo de componente tiene definido eventos que marcan su transición dentro de estados en los cuales tiene disponibilidad variable para uso.

Lógica del Negocio: Es el código que implementa la funcionalidad de una aplicación.

Método del Negocio: Es un método que implementa una lógica del negocio o las reglas de la aplicación.

Persistencia manejada por el bean: Es el mecanismo por el que la transferencia de datos entre las variables y el manejador de recursos de un Entity Bean es manejado por el Entity Bean.

Persistencia manejada por el Contenedor: Es el mecanismo mediante el cual la transferencia de datos entre las variables y el manejador de recursos de un Entity Bean es manejado por el contenedor del Bean.

Servlet. Programa en Java que se ejecuta como parte de un servicio de red, típicamente un servidor del HTTP y responde a las peticiones de clientes.

SSL: Secure Socket Layer. Es un protocolo para transmitir nuestra información a través del Internet de manera encriptada.

UML: Es un lenguaje gráfico que sirve para modelar, diseñar, estructurar, visualizar, especificar, construir y documentar software. UML proporciona un vocabulario común para toda la cadena de producción, desde quien recaba los requisitos de los usuarios, hasta el último programador responsable del mantenimiento. Es un lenguaje estándar para crear los planos de un sistema de forma completa y no ambigua. Fue creado por el Object Management Group, OMG, un consorcio internacional sin ánimo de lucro, que asienta estándares en el área de computación distribuida orientada a objetos, y

actualmente revisa y actualiza periódicamente las especificaciones del lenguaje, para adaptarlo a las necesidades que surgen. El prestigio de este consorcio es un aval más para UML, considerando que cuenta con socios tan conocidos como la NASA, la Agencia Europea del Espacio ESA, el Instituto Europeo de Bioinformática EBI, Boeing, Borland, Motorla y el W3C, por mencionar algunos[11].

[11] Sacado del VII Congreso Nacional de Informática para la Salud. Tutorial de UML.

Anexo 1

Empaquetado EJB-JAR

```
<?xml version="1.0"?>
<project default="Run XDOCLET" basedir=".">
        <property name="bin.dir" value="../bin" />
   <property name="ejb.dd.dir" value="META-INF"/>
   <property name="web.dd.dir" value="WEB-INF"/>
   <property name="xdoclet.force" value="${xdoclet.force}"/>
   <property name="container.type" value="ejb-jar"/>
        <property name="eclipse.home" value="${eclipse.home}"/>
     <path id="project.class.path">
       <pathelement location="${bin.dir}" />
                <pathelement location="${server.home}/server/default/lib/jboss-j2ee.jar"/>
          <fileset dir="${eclipse.home}/plugins">
          <include name="**/${xdoclet.home}/*.jar"/>
          </fileset>
       <pathelement path="${java.class.path}" />
          <fileset dir="${tomcat.home}">
          <include name="*.jar"/>
          </fileset>
    </path>
  <target name="Run XDOCLET">
        <taskdef name="ejbdoclet" classname="xdoclet.modules.ejb.EjbDocletTask">
                <classpath refid="project.class.path"/>
        </taskdef>
     <ejbdoclet
        destdir="${ejbsrc.dir}"
        mergedir="${ejb.dd.dir}"
        excludedtags="@version,@author,@todo"
        ejbspec="2.0"
        force="${xdoclet.force}"
        verbose="true" >
```

```
      <fileset dir="../src" defaultexcludes="yes" >
      </fileset>
<packageSubstitution packages="ejb,proxy" substituteWith="interfaces"/>
<remoteinterface/>
<localinterface/>
<homeinterface />
<localhomeinterface/>
<entitypk>
            <packageSubstitution packages="ejb" substituteWith="keys"/>
 </entitypk>
 <dataobject pattern="{0}DTO">
    <packageSubstitution packages="bo.ejb" substituteWith="dto" />
 </dataobject>
 <!--valueobject/-->
 <utilobject cacheHomes="true" includeGUID="false"
    kind="physical">
    <packageSubstitution packages="ejb,proxies" substituteWith="util"/>
 </utilobject>
 <!--entitycmp /-->
 <entityfacade pattern="{0}FacadeBean" destdir="${src.dir}">
    <packageSubstitution packages="ejb" substituteWith="proxies" />
 </entityfacade>
 <remotefacade pattern="{0}Proxy">
    <packageSubstitution packages="ejb" substituteWith="proxies" />
 </remotefacade>
 <!--entitybmp/-->
 <session>
    <packageSubstitution packages="ejb,proxies" substituteWith="interfaces" />
 </session>
 <deploymentdescriptor
   destdir="${ejb.dd.dir}"
   validatexml="true"
   mergedir="${ejb.dd.dir}"
   />
            <jboss
```

```
        version="3.0"
                                securityDomain="java:/jaas/E-guana"
        unauthenticatedPrincipal="visitante"
        xmlencoding="UTF-8"
        destdir="${ejb.dd.dir}"
        validatexml="false"
              datasource="${datasource.name}"
        datasourcemapping="${type.mapping}"
        createTable="true"
        removeTable="false"
       />
   </ejbdoclet>
</target>
<target name="Deploy">
   <jar jarfile="${deploy.dir}/${container}.jar">
     <!--<fileset dir="${bin.dir}" excludes="**/web/*.*,**/servlets/*.*">-->
                  <fileset dir="${bin.dir}" excludes="**/servlets/*.*">
       <include name="**/*.*"/>
     </fileset>
     <fileset dir=".">
       <include name="META-INF/**/*.*"/>
       <exclude name="META-INF/build.properties"/>
       <exclude name="META-INF/ant.xml"/>
     </fileset>
   </jar>
</target>
<target name="crearjar">
                  <jar jarfile="${container}.jar">
     <!--<fileset dir="${bin.dir}" excludes="**/web/*.*,**/servlets/*.*">-->
                  <fileset dir="${bin.dir}" excludes="**/servlets/*.*">
       <include name="**/*.*"/>
     </fileset>
     <fileset dir=".">
       <include name="META-INF/**/*.*"/>
```

Anexo 2

Empaquetado WAR

```
<project name="licitacionJSF" basedir="../" default="deploy">
   <!-- Project settings -->
   <property name="project.distname" value="licitacionJSF"/>

   <!-- Local system paths -->
   <property file="${basedir}/ant/build.properties"/>
   <property name="webroot.dir" value="${basedir}/WebContent"/>
   <property name="webinf.dir" value="${webroot.dir}/WEB-INF"/>

         <!-- classpath for MyFaces 1.0.9 -->
   <path id="compile.classpath">
         <!--pathelement path ="C:/jboss-3.2.4/client/jboss-j2ee.jar"/-->
      <pathelement path ="${webinf.dir}/lib/commons-beanutils-1.6.1.jar"/>
      <pathelement path ="${webinf.dir}/lib/commons-codec-1.2.jar"/>
      <pathelement path ="${webinf.dir}/lib/commons-collections-3.0.jar"/>
      <pathelement path ="${webinf.dir}/lib/commons-digester-1.5.jar"/>
      <pathelement path ="${webinf.dir}/lib/commons-el.jar"/>
      <pathelement path ="${webinf.dir}/lib/commons-fileupload-1.0.jar"/>
      <pathelement path ="${webinf.dir}/lib/commons-logging.jar"/>
      <pathelement path ="${webinf.dir}/lib/commons-validator.jar"/>

      <pathelement path ="${webinf.dir}/lib/jakarta-oro.jar"/>
      <pathelement path ="${webinf.dir}/lib/jsp-2.0.jar"/>
      <pathelement path ="${webinf.dir}/lib/jstl.jar"/>
      <pathelement path ="${webinf.dir}/lib/log4j-1.2.8.jar"/>
      <pathelement path ="${webinf.dir}/lib/myfaces.jar"/>
      <pathelement path ="${webinf.dir}/lib/tomahawk.jar"/>
      <pathelement path ="${webinf.dir}/lib/myfaces-impl.jar"/>
      <pathelement path ="${webinf.dir}/lib/myfaces-api.jar"/>
      <pathelement path ="${webinf.dir}/lib/myfaces-wap.jar"/>
```

```
    <pathelement path ="${webinf.dir}/lib/myfaces-xdoclet.jar"/>
    <pathelement path ="${webinf.dir}/classes"/>
    <pathelement path ="${classpath.external}"/>
    <pathelement path ="${classpath}"/>
</path>
<!-- Check timestamp on files -->
<target name="prepare">
    <!--tstamp/-->
</target>
<!-- Copy any resource or configuration files -->
<target name="resources">
    <copy todir="${webinf.dir}/classes" includeEmptyDirs="no">
        <fileset dir="JavaSource">
        <patternset>
            <include name="**/*.conf"/>
            <include name="**/*.properties"/>
            <include name="**/*.xml"/>
        </patternset>
        </fileset>
    </copy>
</target>
<!-- Normal build of application -->
<target name="compile" depends="prepare,resources">
    <javac srcdir="JavaSource" destdir="${webinf.dir}/classes">
        <classpath refid="compile.classpath"/>
    </javac>
</target>
<!-- Remove classes directory for clean build -->
<target name="clean"
  description="Prepare for clean build">
  <delete dir="${webinf.dir}/classes"/>
  <mkdir  dir="${webinf.dir}/classes"/>
</target>
<!-- Build entire project -->
<target name="build" depends="prepare"/>
```

```
    <target name="rebuild" depends="clean,prepare,compile"/>
    <!-- Create binary distribution -->
    <target name="war" depends="build">
      <mkdir dir="${build.dir}"/>
      <war
        basedir="${webroot.dir}"
        warfile="${deploy.dir}/${project.distname}.war"
        webxml="${webinf.dir}/web.xml">
        <exclude name="WEB-INF/${build.dir}/**"/>
        <exclude name="WEB-INF/src/**"/>
        <exclude name="WEB-INF/web.xml"/>
       </war>
    </target>
    <target name="deploy" depends="undeploy,war">
      <copy file="${deploy.dir}/${project.distname}.war" todir="${server.deploy.dir}"/>
    </target>
    <target name="undeploy">
      <delete file="${server.deploy.dir}/${project.distname}.war"/>
    </target>
</project>
```

Anexo 3

Pruebas a RequerimientoServiceBean

```
public class RequerimientoPrueba extends PruebaBase {

   public RequerimientoPrueba(String nombrePrueba) {
      super(nombrePrueba);
   }

   private RequerimientoServiceProxy servicio;

   private static final String OBSERVACION = "Requerimiento de Prueba";

   private static final Integer CANTIDAD = new Integer(10);

   protected void setUp() throws Exception {
      super.setUp();
      servicio = new RequerimientoServiceProxy();
   }

   private RequerimientoDTOfull buscarRequerimiento(Integer estado)
         throws Exception {
      return (RequerimientoDTOfull) servicio
            .buscarRequerimientosPorObservacion(OBSERVACION, estado).get(0);
   }

   public void testCrearRequerimiento() throws Exception {
      RequerimientoDTOfull requerimiento = new RequerimientoDTOfull();
      requerimiento.setObservacion(OBSERVACION);
      DetalleRequerimientoDTOfull detalle = new DetalleRequerimientoDTOfull();
      detalle.setCantidad(CANTIDAD);
      detalle.setIdProducto(new Integer(producto.getId()));
      Vector detalles = new Vector();
```

```
            detalles.add(detalle);
            requerimiento.setDetallesRequerimiento(detalles);
            Integer idRequerimiento = servicio.crearRequerimiento(requerimiento);
            assertNotNull("ID del requerimiento registrado", idRequerimiento);
        }

        public void testBuscarRequerimientosPorObservacion() throws Exception {
    RequerimientoDTOfull requerimiento2 =
    buscarRequerimiento(CodigoEstadoEJB.REGISTRADO);
            assertNotNull("Requerimiento encontrado", requerimiento2);
            assertEquals("Observacion", OBSERVACION, requerimiento2
                    .getObservacion());
        }

        public void testPublicarRequerimiento() throws Exception {
    servicio.publicarRequerimiento(buscarRequerimiento(
    CodigoEstadoEJB.REGISTRADO).getIdRequerimiento());
        }

        public void testEliminarRequerimiento() throws Exception {
    RequerimientoDTOfull
    requerimiento2 = buscarRequerimiento(CodigoEstadoEJB.PUBLICADO);
            assertNotNull("Requerimiento encontrado", requerimiento2);
            servicio.eliminarRequerimiento(requerimiento2.getIdRequerimiento());
        }

        public static Test suite() {
    TestSuite suite = new TestSuite("Pruebas de Requerimiento");
                    suite.addTest(new RequerimientoPrueba("testCrearRequerimiento"));
            suite.addTest(new RequerimientoPrueba(
                    "testBuscarRequerimientosPorObservacion"));
            suite.addTest(new RequerimientoPrueba("testPublicarRequerimiento"));
            suite.addTest(new RequerimientoPrueba("testEliminarRequerimiento"));
            return suite;
    }
```

Anexo 4

Pruebas a LicitacionServiceBean

```
public class LicitacionPrueba extends PruebaBase {

   private static final String OBSERVACION = "Licitacion de Prueba";

   private static final Date FECHA_CADUCIDAD = Date.valueOf("2006-12-31"),
         FECHA_INICIO = Date.valueOf("2006-01-01");

   private static final Integer CANTIDAD = new Integer(10);

   private LicitacionServiceProxy servicio;

   protected void setUp() throws Exception {
      super.setUp();
      servicio = new LicitacionServiceProxy();
   }

   public LicitacionPrueba(String nombrePrueba) {
      super(nombrePrueba);
   }

   private LicitacionDTOfull buscarLicitacion(Integer estado) throws Exception {
      return (LicitacionDTOfull) servicio.buscarPorObservacion(OBSERVACION,
            estado).get(0);
   }

   public void testCrearLicitacion() throws Exception {
      LicitacionDTOfull licitacion = new LicitacionDTOfull();
      licitacion.setObservacion(OBSERVACION);
      licitacion.setFechaCaducidad(FECHA_CADUCIDAD);
      licitacion.setFechaInicio(FECHA_INICIO);
```

```
    DetalleLicitacionDTOfull detalle = new DetalleLicitacionDTOfull();
    detalle.setIdProducto(new Integer(producto.getId()));
    detalle.setCantidad(CANTIDAD);

    Vector detalles = new Vector();
    detalles.add(detalle);
    licitacion.setDetallesLicitacion(detalles);

    assertNotNull("ID de la licitacio creada", servicio
            .crearLicitacion(licitacion));
}

public void testBuscarPorObservacion() throws Exception {
    LicitacionDTOfull
            licitacion2 = buscarLicitacion(CodigoEstadoEJB.REGISTRADO);
    assertNotNull("Licitacion encontrada", licitacion2);
    assertEquals("Observacion", OBSERVACION, licitacion2.getObservacion());
    assertEquals("Fecha caducidad", FECHA_CADUCIDAD, licitacion2
            .getFechaCaducidad());
    assertEquals("Fecha Inicio", FECHA_INICIO, licitacion2.getFechaInicio());
}

public void testPublicarLicitacion() throws Exception {
    servicio
            .publicarLicitacion(buscarLicitacion(CodigoEstadoEJB.REGISTRADO)
                    .getIdLicitacion());
}

public void testDeclararDesierta() throws Exception {
    servicio.declararDesierta(buscarLicitacion(CodigoEstadoEJB.PUBLICADO)
            .getIdLicitacion());
}

public void testEliminarLicitacion() throws Exception {
```

```
            servicio.eliminarLicitacion(buscarLicitacion(
                    CodigoEstadoEJB.LICITACION_DESIERTA).getIdLicitacion());
        }

        public static Test suite() {
            TestSuite suite = new TestSuite("Pruebas de Licitacion");
            suite.addTest(new LicitacionPrueba("testCrearLicitacion"));
            suite.addTest(new LicitacionPrueba("testBuscarPorObservacion"));
            suite.addTest(new LicitacionPrueba("testPublicarLicitacion"));
            suite.addTest(new LicitacionPrueba("testDeclararDesierta"));
            suite.addTest(new LicitacionPrueba("testEliminarLicitacion"));
            return suite;
        }
}
```

Anexo 5

Pruebas a SubastaServiceBean

```
public class SubastaPrueba extends PruebaBase {

   private SubastaServiceProxy servicio;

   private static final Integer CANTIDAD = new Integer(10);

   private static final String OBSERVACION = "Subasta de Prueba",
         PRODUCTO_MIO = "Televisor Panasonic ViewMe Encore",
         OBSERVACION_OFERTA = "Oferta Subasta de Prueba";

   private static final Date FECHA_TOPE = Date.valueOf("2006-12-31");

   private static final float PRECIO_BASE = 1000, PRECIO_OFERTA = 2200;

   private CatalogoProductoDTO catalogoProducto;

   protected void setUp() throws Exception {
      super.setUp();
      servicio = new SubastaServiceProxy();

      AdminCatalogoProxy adminCatalogo = new AdminCatalogoProxy();
      catalogoProducto = (CatalogoProductoDTO) ((List) adminCatalogo
            .buscarPorProducto(PRODUCTO_MIO)).get(0);
   }

   public SubastaPrueba(String nombrePrueba) {
      super(nombrePrueba);
   }

   public void testCrearSubasta() throws Exception {
```

```
    SubastaDTOfull subasta = new SubastaDTOfull();
    subasta.setCantidad(CANTIDAD);
    subasta.setObservacion(OBSERVACION);
    subasta.setFechaTope(FECHA_TOPE);
    subasta.setPrecioBase(PRECIO_BASE);

    subasta.setCatalogoProducto(catalogoProducto);

    assertNotNull("ID de subasta", servicio.crearSubasta(subasta));
}

private SubastaDTOfull buscarSubasta() throws Exception {
    SubastaDTOfull subasta2 = (SubastaDTOfull) ((List) servicio
            .buscarPorObservacion(OBSERVACION, CodigoEstadoEJB.PUBLICADO))
            .get(0);
    return subasta2;
}

public void testBuscarPorObservacion() throws Exception {
    SubastaDTOfull subasta2 = buscarSubasta();
    assertEquals("Observacion", OBSERVACION, subasta2.getObservacion());
    assertEquals("Fecha Tope", FECHA_TOPE, subasta2.getFechaTope());
    assertEquals("Cantidad", CANTIDAD, subasta2.getCantidad());
    assertEquals("Precio Base", PRECIO_BASE, subasta2.getPrecioBase(), 0);

    CatalogoProductoDTO catalogoProducto2 = subasta2.getCatalogoProducto();

    assertEquals("ID Catalogo", catalogoProducto.getIdCatalogo(),
            catalogoProducto2.getIdCatalogo());
    assertEquals("Descripcion Producto/Catalogo", catalogoProducto
            .getDescripcion(), catalogoProducto2.getDescripcion());
}

public void testReceptarOfertaSubasta() throws Exception {
    SubastaDTOfull subasta2 = buscarSubasta();
```

```
        OfertaSubastaDTOfull ofertaSubasta = new OfertaSubastaDTOfull();
        ofertaSubasta.setIdSubasta(subasta2.getId());
        ofertaSubasta.setObservacion(OBSERVACION_OFERTA);
        ofertaSubasta.setPrecio(PRECIO_OFERTA);

        assertNotNull("ID de oferta subasta", servicio
                .receptarOfertaSubasta(ofertaSubasta));
    }

    public void testBuscarOfertasSubasta() throws Exception {
        SubastaDTOfull subasta2 = buscarSubasta();
        OfertaSubastaDTOfull ofertaSubasta2 =
   (OfertaSubastaDTOfull) ((List) servicio
                .buscarOfertasSubasta(subasta2.getId())).get(0);

        assertEquals("Observacion Oferta", OBSERVACION_OFERTA, ofertaSubasta2
                .getObservacion());
        assertEquals("Precio Oferta", PRECIO_OFERTA,
                ofertaSubasta2.getPrecio(), 0);
    }

    public static Test suite() {
        TestSuite suite = new TestSuite("Pruebas de Subasta");
        suite.addTest(new SubastaPrueba("testCrearSubasta"));
        suite.addTest(new SubastaPrueba("testBuscarPorObservacion"));
        suite.addTest(new SubastaPrueba("testReceptarOfertaSubasta"));
        suite.addTest(new SubastaPrueba("testBuscarOfertasSubasta"));
        return suite;
    }
}
```

Bibliografía.

Libros:

J2EE Core Design Patterns.

Design Patterns Explained: A New Perspective on Object-Oriented Design. InformIT. http://www.informit.com

Mastering JavaServerFaces. Dudney, Jonathan Lehr, Bill Willis, LeRoy Mattingly

Documentos:

RFC 2511 - Internet X.509 Certificate Request Message Format http://www.faqs.org/rfcs/rfc2511.html

RFC LENGPROG – Lenguaje de Programación y Reseña de Java. http://www.monografias.com/trabajos/lengprog/lengprog.shtml

Enlaces:

Eclipse. http://eclipse.org/

Oracle. http://www.oracle.com

Osmosis Latina. http://www.osmosislatina.com/index.htm

Bibliografía

1. Josep Valor. Negocios de Internet: las reglas de siempre siguen vigentes en el nuevo tablero de juego
2. Sue Hamilton, Jim Myron, Jeff Leech. Entering the Dynamic New E-procurement Marketplace: A procurement director's guide. White Paper.
3. Eric Armstrong, Jennifer Ball, Stephanie Bodoff, Debbie Bode Carson, Ian Evans, Dale Green, Kim Haase, Eric Jendrock. The J2EE 1.4 Tutorial. 17 junio del 2004.
4. Carlos Vera Quintana. Desarrollo empresarial y ambiente regulatorio en el sector de las Telecomunicaciones en Ecuador. Septiembre 2006. http://www.corpece.org.ec.
5. CORPECE. El comercio electrónico y sus perspectivas en Ecuador. Septiembre 2006. http://www.corpece.org.ec.
6. Congreso Nacional del Ecuador, Ley de Comercio Electrónico, Firmas y Mensajes de datos. 17 abril del 2002.
7. Eduardo Navarro. Cómo puede la empresa mejorar resultados usando Internet. Junio 2001. http://www.improven-consultores.com.
8. José Ignacio López Sánchez. Evolución de los modelos de negocios en Internet: situación actual en España de la Economía Digital. Universidad Complutense de Madrid. Julio del 2002. http://www.ucm.es/info/business/Documentos/articulos/030703.pdf.

9. Java Tools for Extreme Programming. Richard Hightower, Nicholas Lesiecki. 2002
10. Noel Yuhanna ,Forrester Research.
11. Barry Boehm. Desarrollo en espiral. Septiembre del 2006. http://es.wikipedia.org/wiki/Desarrollo_en_espiral